# THREE DIMENSIONAL HYPERBOLIC GRID GENERATION

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

BY

UMUT CAN DİNÇGEZ

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

APRIL 2006

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan ÖZGEN
Director

I certify that this satisfies all the requirements as a thesis for the degree of Master of Science.

Prof Dr. Sıtkı Kemal İDER
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

| | |
|---|---|
| Dr. Ali Ruhşen ÇETE<br>Co-Supervisor | Prof. Dr. Mehmet Haluk AKSEL<br>Supervisor |

Examining Committee Members

| | |
|---|---|
| Prof. Dr. Kahraman ALBAYRAK | (METU, ME) |
| Prof. Dr. Mehmet Haluk AKSEL | (METU, ME) |
| Dr. Ali Ruhşen ÇETE | (TAI) |
| Instructor Dr. Tahsin ÇETİNKAYA | (METU, ME) |
| Prof. Dr. İbrahim Sinan AKMANDOR | (METU, AEE) |

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

**Name, Last name :** Umut Can DİNÇGEZ

**Signature :**

## ABSTRACT

## THREE DIMENSIONAL HYPERBOLIC GRID GENERATION

DİNÇGEZ, Umut Can
M.Sc., Department of Mechanical Engineering
Supervisor: Prof. Dr. M. Haluk AKSEL
Co-Supervispor: Dr. A. Ruhşen ÇETE

April 2006, 137 pages

This thesis analyzes procedure of generation of hyperbolic grids formulated by two constraints, which specify grid orthogonality and cell volume. The procedure was applied on a wide range of geometries and high quality two and three dimensional hyperbolic grids were generated by using grid control and smoothing procedures, which supply grid clustering in all directions and prevent grid deformation (grid shock), respectively.

Keywords: Hyperbolic Grid Generation, Two Dimensional, Three Dimensional, Grid Control and Smoothing

# ÖZ

## ÜÇ BOYUTLU HİPERBOLİK AĞ ÜRETİMİ

DİNÇGEZ, Umut Can
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. M. Haluk AKSEL
Tez Yardımcı Yöneticisi: Dr. A. Ruhşen ÇETE

Nisan 2006, 137 sayfa

Bu tezde ağı meydana getiren eğrilerin kesişme noktasının birbirine dik olması ve ağ elemanlarının hacimlerinin önceden belirlenmesi esasına dayanan hiperbolik ağ üretim tekniği incelenmiştir. Bu teknik birçok değişik geometriye uygulanmış, yüksek kalitede iki ve üç boyutlu hiperbolik ağlar elde edilmiştir. Yüksek kalitedeki bu ağları elde ederken, ağ çizgilerinin her yönde sıkıştırılmasını sağlayan ağ kontrol metodu ve ağ çizgilerinin bozulmasını önleyen yumuşatma metodu kullanılmıştır.

Anahtar Kelimeler: Hiperbolik Ağ Üretimi, İki Boyutlu, Üç Boyutlu, Ağ Kontrolü, Yumuşatma

To My Family

**ACKNOWLEDGEMENTS**

I would like to express my deepest gratitude to my supervisor Prof. Dr. Haluk AKSEL for his guidance, patience and invaluable help throughout this study.

I would also like to thank Dr. A. Ruhşen ÇETE for sharing his valuable theoretical and practical knowledge throughout my Master of Science program.

I would like to express my great appreciation to my friends; Emre GÜRDAMAR, Onur BAŞ, Kıvanç ÜLKER and Özlem CEYHAN for their help, contribution and encouragement throughout the preparation of this thesis.

Finally, I would like to thank my family for their patience, understanding, altruism and support throughout my education.

## TABLE OF CONTENTS

## LIST OF FIGURES

FIGURES

## LIST OF TABLES

TABLES

## LIST OF SYMBOLS

### SYMBOLS

| | |
|---|---|
| i, j, k | Grid indices, related to ξ, η and ζ directions respectively |
| x, y, z | Cartesian coordinates of physical space |
| ξ, η, ζ | Cartesian coordinates of computational space |
| A, B, C | Coefficient matrices of linear form of finite difference equation of hyperbolic equations |
| I | Identity matrix |
| H | Right hand side matrix |
| F | Cell area function or inverse Jacobian |
| J | Jacobian |
| $\Delta, \delta, \nabla$ | Forward, central and backward operators respectively |
| $\Delta\nabla$ | Laplacian operarator |
| $\lambda$ | Eigenvalues |
| $\beta$ | Fourth order damping coefficient |
| $\Delta c$ | Chord length |
| $\Delta s$ | Marching distance |
| $\Delta s_1$ | Initial grid spacing in the marching direction |
| ε | Ratio of successive spacing in the marching direction |
| α | Half angle formed by the convex corner |
| JS | The level to be stretched |
| INTJ | The interval of stretching |
| SJ | The value of stretching at the required level |
| $^{k, o}$ | Scripts to denote the known level |
| $\vec{R}$ , $\vec{r}$ | Position vectors |

| | |
|---|---|
| $l$ | Distance between two successive grid points along the profile |
| $\Gamma$ | Grid curves |
| cf | Cell size control factor |
| $\Delta V$ | Cell volume |
| AA, BB, CC | Coefficient matrices of simplified finite difference equation of hyperbolic system of equations |
| DD | Right hand side matrix of simplified finite difference equation of hyperbolic system of equations |
| $\Delta A$ | Cell area |
| $\vec{e}$ , $\vec{f}$ | Right hand side vectors of linearized form of finite difference equation of 3-D hyperbolic equation |
| $\Delta S$ | Surface area of three dimensional grid |
| $\varepsilon_{i\xi}, \varepsilon_{i\eta}$ | Implicit second order smoothing parameters |
| $\varepsilon_{e\xi}$ , $\varepsilon_{e\eta}$ | Explicit second order smoothing parameters |
| $\| \|$ | Norm of a matrix |
| $H_o(T)$, $H_1(T)$ | Hermitian polynomials |
| rc | Clustering coefficient |
| $\Delta x_1$ | Arclength between first two grid points in $\xi$ direction |
| $\Delta x_n$ | Arclength between $n^{th}$ and $(n-1)^{th}$ grid points |
| N | Number of grid points in $\xi$ direction |
| $S_n$ | Total arc-length up to $n^{th}$ grid point in $\xi$ direction |
| *dissipation* | Fourth order artificial dissipation term |
| *im* | Maximum index of grid points in $\xi$ direction |
| *jm* | Maximum index of grid points in $\eta$ direction |
| $\det C$ | Determinant of C matrix |
| $v$ | Complex plane |
| $\vec{d}$ | Intermediate solution vector |

# CHAPTER 1

# INTRODUCTION

## 1.1 General

In order to solve the governing partial differential equations (PDE) of fluid mechanics numerically, approximations to the partial differentials are made. By the help of these approximations, partial derivatives are converted to finite difference expressions. These finite difference expressions are used to rewrite the partial differential equations as algebraic equations and these algebraic equations are solved at discrete points within the domain of interest. So, in order to solve these equations, a set of grid points within the domain and the boundaries should be specified.

In order to identify grid points easily, the computational domain has rectangular shape in general and the grid points are placed along the grid lines within the computational domain.

If the physical domain is rectangular in shape, generation of a grid is easy since there is no need any effort to convert the physical domain to rectangular computational domain. Therefore, the specification of boundary conditions is easy and grid points can be specified on the boundaries of physical domain.

However, the majority of the physical domains of interest are nonrectangular. Therefore, imposing a rectangular computational domain on such a physical domain requires interpolations for the implementation of boundary conditions. Since the boundary conditions have the greatest effect on the solution of the

equations, these kinds of interpolations can cause inaccuracies especially at the places of greatest sensitivity. Besides this, if the grid spacing near the boundaries is unequal, more complications with the finite difference equations occur since approximations with unequal step sizes should be used. Also, difficulties in programming can occur since these kind of finite difference equations changes from node to node. To overcome such difficulties, a transformation from physical space to computational space should be introduced. This transformation is accomplished by specifying a generalized coordinate system which will map the nonrectangular grid system in the physical space to a rectangular uniform grid spacing in the computational space. [1]

The critical point is identifying the location of grid points in the physical domain, that is identifying the $x$ and $y$ coordinates of a grid point in the physical space corresponding to the same grid point in the computational space. Some constraints should be regarded while determining the grid points. Grid lines of the same family can not cross each other, which is known as one to one mapping in literature. In addition to this, some properties like smooth grid point distribution with minimum grid line skewness, orhogonality of grids and clustering of grid points in regions where high flow gradients occur are required. However, all these requirements can not be achieved by any particular grid generation technique.

The grid system may be categorized as fixed or adaptive. A fixed grid system is generated prior to the solution of the governing equations of fluid motion and remains fixed independent of the solution. [1] On the other hand, an adaptive grid system evolves as a result of the solution of the equations of fluid motion. As an example to the adaptive grid system, grid points may move toward regions of high gradients such as in the neighborhood of a shock wave. [1]

Detailed information about grid generation techniques is given under the title "Grid Generation Techniques".

## 1.2 Grid Generation Techniques

Grid generation schemes can be classified into two, which are structured grid generation and unstructured grid generation schemes. In structured grid generation schemes, the physical space is transformed to the computational space. This computational space has rectangular boundaries and uniform step sizes.

In unstructured grid generation scheme, there is no need to the transformation and this type of grid is imposed on the physical space directly. Various types of elements may be used, but triangular elements are the most popular elements to construct the grid system. An example to unstructured grid generated by triangular elements is shown in Figure 1.1.

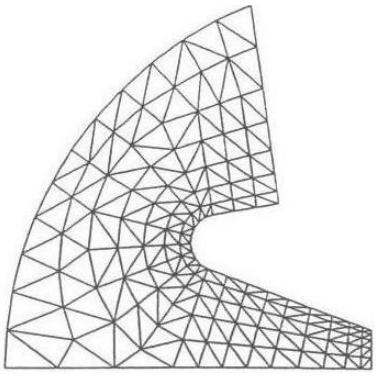

**Figure 1.1** Unstructured grid with triangular elements

More detailed information about unstructured and structured grid generation techniques is given under the following headings.

### 1.2.1 Unstructured Grid Generation Techniques

Unstructured meshes have been developed mainly for the finite element method. There is a large range of shapes for finite elements. These are tetrahedra, prisms, blocks and triangles and there can be arbitrary connectivity between these shapes leading to unstructured meshes. Among these shapes, triangles in two dimension and tetrahedra in three dimensions are the only shapes that can be used to generate unstructured meshes fully automatically.

There are several methods for unstructured grid generation. The most common methods are *Advancing Front* and *Delaunay Triangulation.* Brief information about these two methods is given below.

#### 1.2.1.1 Advancing Front Method

In this method, the boundary is discretized first. This discretization is done by fitting the boundary with polygons in two dimensions. This is known as initial front. Then, by adding triangles or tetrahedral into the domain, with at least one edge or face on the front, initial front is updated. This procedure is shown in Figures 1.2, 1.3, 1.4, 1.5 and 1.6. This update process continues until the front is empty. When the front is empty, mesh generation is completed. This requires that the domain should be bounded but for unbounded domain, the front can be advanced until it is at some large distance from the object. As the algorithm progresses, the front will advance to fill the remainder of the area with triangles.

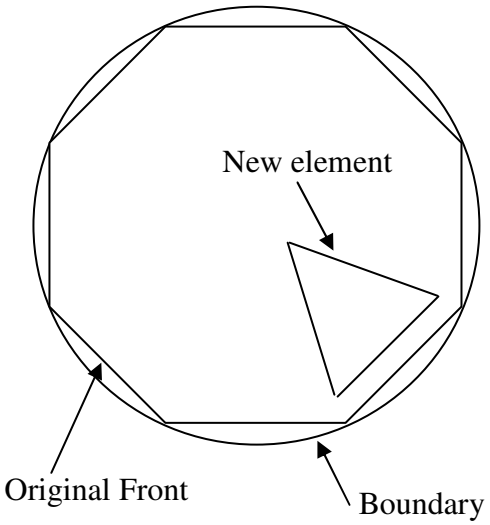


**Figure 1.2** Discretization of boundary and adding a new element

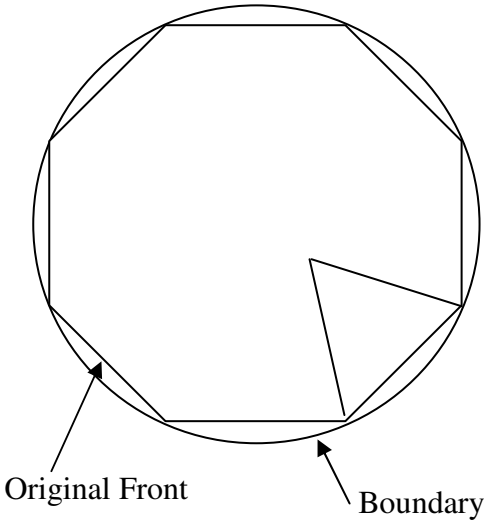


**Figure 1.3** Sketch after the new element is added

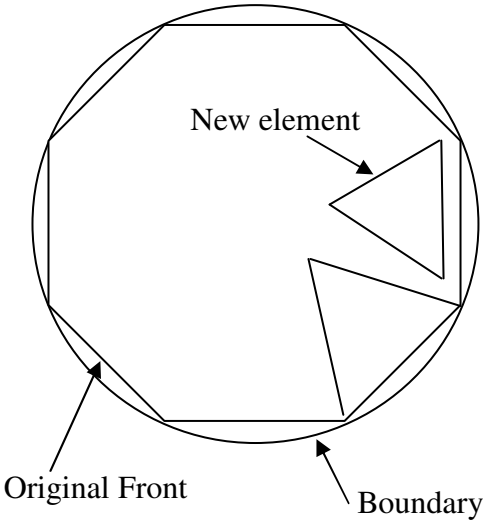


**Figure 1.4** Sketch showing addition of another element

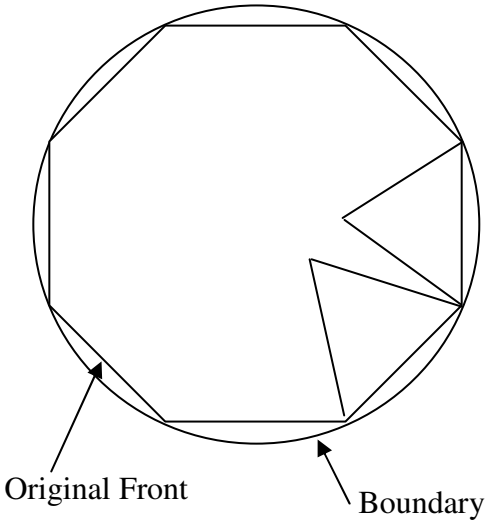


**Figure 1.5** Sketch after another element is added

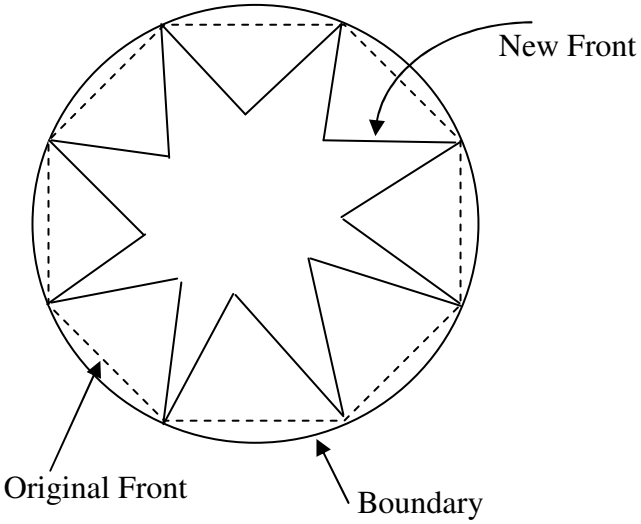


**Figure 1.6** Sketch after all the elements are added and original front is updated

#### 1.2.1.2 Delaunay Triangulation

This method has two phases; placement of mesh vertices and triangulation. If the mesh vertices are placed well, the triangulation phase can be simple.

In the first phase, vertices are placed along the domain boundary and then new points are added to the interior of the domain. There are some methods to insert the point to interior point set, but the methods will not be given here.

The second phase is known as *Delaunay Triangulation*. It is defined by empty circle condition, i.e., the triangle is a valid triangle if and only if there is no other point within its circumcircle. An example to the invalid triangle is given in Figure 1.7.

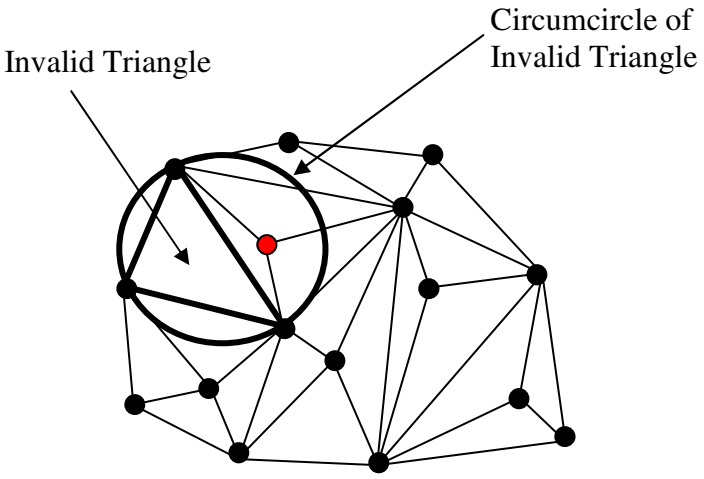


**Figure 1.7** An example to an invalid triangle

### 1.2.2 Structured Grid Generation Techniques

A wide variety of grid generation techniques have been developed and many of these methods currently in use are documented [2]. Grid generation techniques can be divided into three subsections. These are conformal mapping technique, algebraic grid generation technique and partial differential equation techniques. Partial differential equation techniques can be subdivided into three, which are elliptic grid generation technique, parabolic grid generation technique and hyperbolic grid generation technique. Detailed information about these techniques, advantages and disadvantages of them are given in the following sections.

#### 1.2.2.1 Conformal Mapping

In this technique, each point in the two dimensional plane is defined by a complex value $v = x + iy$, where $x$ and $y$ are Cartesian coordinates. So, the

location of each point is characterized by a complex number rather than real numbers. If any grid in the complex plane $v$ is considered, it is seen that the shape, location or orientation of the grid is changed by the conformal transformation.

The idea behind the conformal transformation is that if the original grid lines are orthogonal, the transformed grid lines will be orthogonal too. In literature, conformal mapping is defined as mapping that preserves angles.

This technique is most often applied to fluid dynamic problems. For example, numerical conformal mapping has been applied for the calculation of transonic flow over airfoils [3]. Such a mapping greatly simplifies the boundary conditions and ensures that the nonlinear partial differential equations in the computational plane are slightly more complicated than in the physical domain. Numerical conformal mapping has been found particularly useful in two dimensional incompressible, irrotational free surface flow problems [4, 5]. Normally, the partial differential equations are less distorted and the Laplace equation remains particularly unchanged under conformal mapping. [4]

Application of conformal mappings is limited to two dimensional problems and reasonable knowledge about complex variables is needed. Its main disadvantage is that, construction of numerical conformal mapping usually requires the solution of a nonlinear integral equation and the determination of mapping functions is a difficult task.

#### 1.2.2.2 Algebraic Grid Generation Techniques

The simplest grid generation technique is the algebraic method, advantage of which is the speed with which a grid can be generated. The grid points in the computational domain and those in the physical domain are related with each

other by using algebraic equations. These equations are derived by using an interpolation scheme between the specified boundary grid points [1]. Interior grid points can be generated by using these algebraic equations.

In many grid generation techniques, the metrics and Jacobian of transformation should be evaluated before solving any transformed partial differential equations. However, when an algebraic model is used, the metrics may be calculated analytically in many instances[1]. This brings a very big advantage when compared with other methods. This is because in other methods metrics are computed numerically and this can introduce some errors into the system of equations. Besides this, numerical computation of metrics requires additional time.

In flow problems, accurate computation of flow gradients in regions where large gradients occur needs many grid points in these regions. Instead of using uniform grids with smaller step sizes in the physical domain, grid points may be clustered in these regions. Advantage of this is reducing total number of grid points and increasing the efficiency. Some examples of such algebraic expressions with clustering options are provided in [1].

For many applications, algebraic models provide a reasonable grid system with continuous and smooth metric distributions. However, if grid smoothness, skewness and orthogonality are of concern, grid systems generated by solving partial differential equations must be used [1].

As mentioned earlier, partial differential equation techniques include elliptic, parabolic and hyperbolic grid generation techniques. For the time being, consider the elliptic grid generation technique briefly. In this technique, elliptic partial differential equations are solved in order to generate grids in the physical domain. The difference between algebraic model and elliptic grid generation technique is that a system of partial differential equations is solved

instead of algebraic equations. However, to start such a method, an initial guess of coordinates of grid points is needed. While making this initial estimate, algebraic methods can be used. Therefore, algebraic equations can be used both for generating grid points and for determining initial grid point distribution of elliptic grid generation technique.

Advantages and disadvantages of algebraic grid generation technique are given below.

*Advantages*

1) It is very fast when compared to other grid generation techniques.
2) Numerical errors can be avoided since metrics may be computed analytically.
3) Grid point clustering can be applied to different regions easily.

*Disadvantages*

1) In this technique, propagation of discontinuity at the boundary to interior domain is possible.
2) Skewness and smoothness of grids can not be controlled easily.

#### 1.2.2.3 Partial Differential Equation Techniques

In these methods, in order to determine the location of grid points in the physical space, a system composed of partial differential equations is solved. The computational domain transformed from physical domain is rectangular in shape and has uniform grid spacing. These methods include elliptic, parabolic and hyperbolic system of partial differential equations which are explained in the following sections.

#### 1.2.2.3.1 Elliptic Grid Generation Technique

Among the other partial differential equation techniques, elliptic grid generation method is the most extensively used method. It is used for two dimensional problems in general, but it can be extended to three dimensional problems.

In order to use elliptic grid generation method, all the physical boundaries of domain should be specified. There are three types of domain on which elliptic grid generation technique can be applied. These are known as simply connected domain, doubly connected domain and multiply connected domain.

Simply connected domain can be defined as the domain which can be reducible to a point. So, for a simply connected domain, there should be no objects within the domain. An example of a simply connected domain and the corresponding computational domain are given in Figures 1.8 and 1.9 respectively.

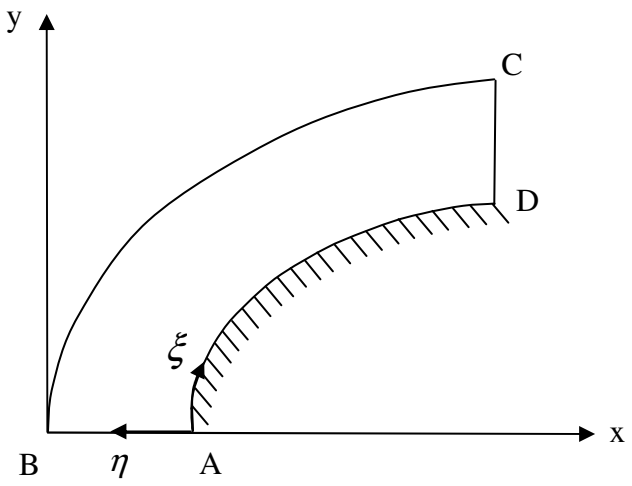


**Figure 1.8** An example to a simply connected domain

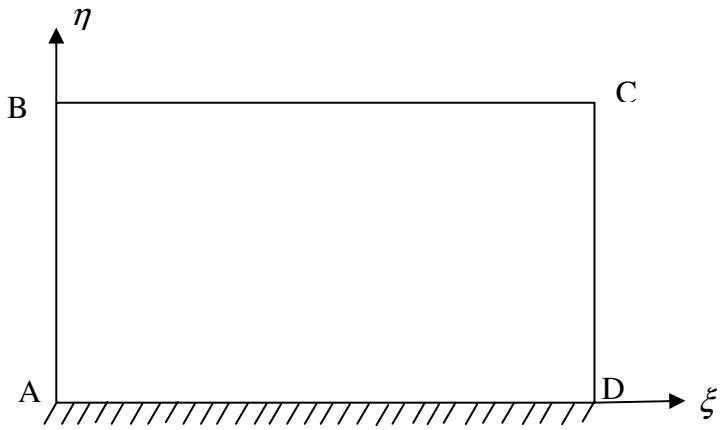


**Figure 1.9** Computational domain of Figure 1.8

Doubly connected domain has one configuration within the domain. So, this domain can not be reducible. However, a doubly connected domain may be converted to simply connected domain by adding branch-cut. This branch-cut extends from a point on the boundary configuration within the domain to a point on the boundary of the domain. As an example, doubly connected region with branch-cut is given in Figure 1.10, unwrapping of doubly connected region is given in Figure 1.11 and computational domain is given in Figure 1.12.

Multiply connected domain has more than one configuration within the domain. This type of domain can be converted to a simply connected domain by adding several branch-cuts. One branch-cut is added like in doubly connected domain, that is, it extends from a point on the boundary of configuration within the domain to a point on the boundary of the domain. Other branch-cuts are added between the boundaries of configurations within the domain. As an example, physical domain for a multiply connected region, unwrapping of a multiply connected region and the corresponding computational domain are given in Figures 1.13, 1.14 and 1.15 respectively.

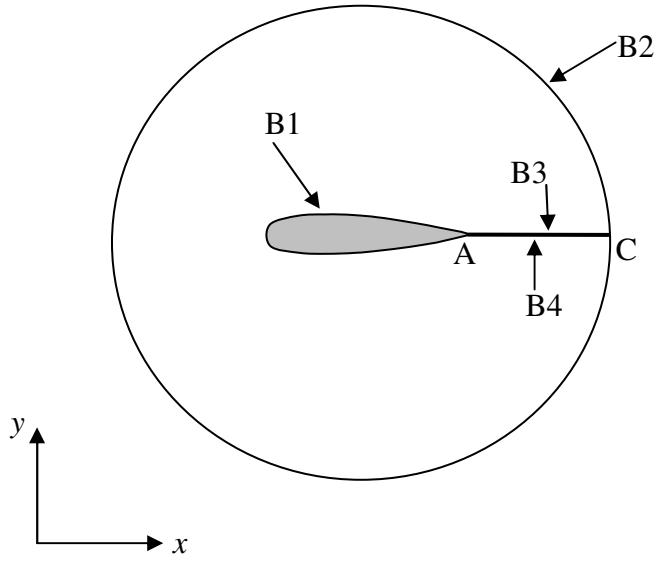


**Figure 1.10** An example to a doubly connected region with branch-cut

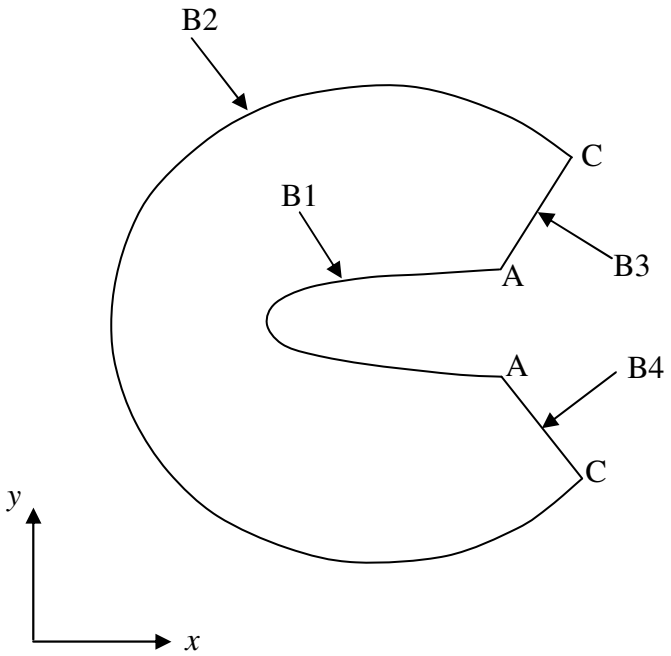


**Figure 1.11** Unwrapping of Figure 1.10

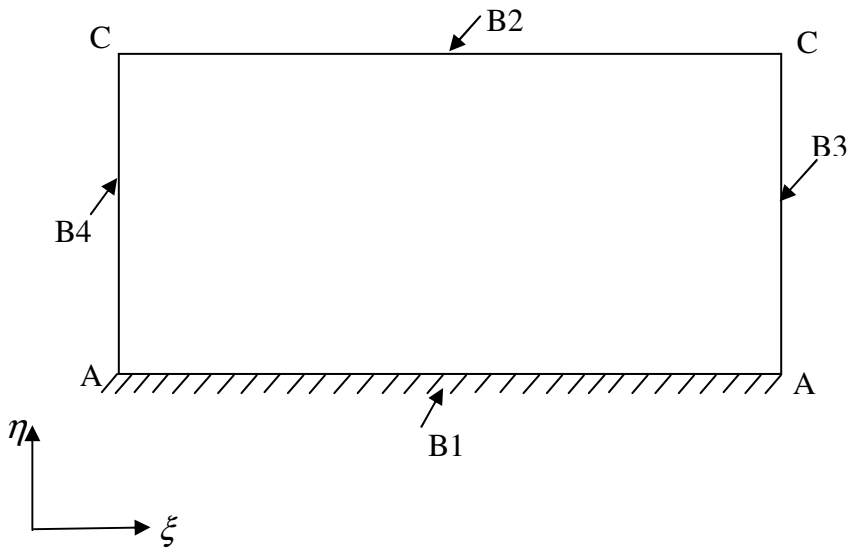


**Figure 1.12** Computational domain of Figure 1.10

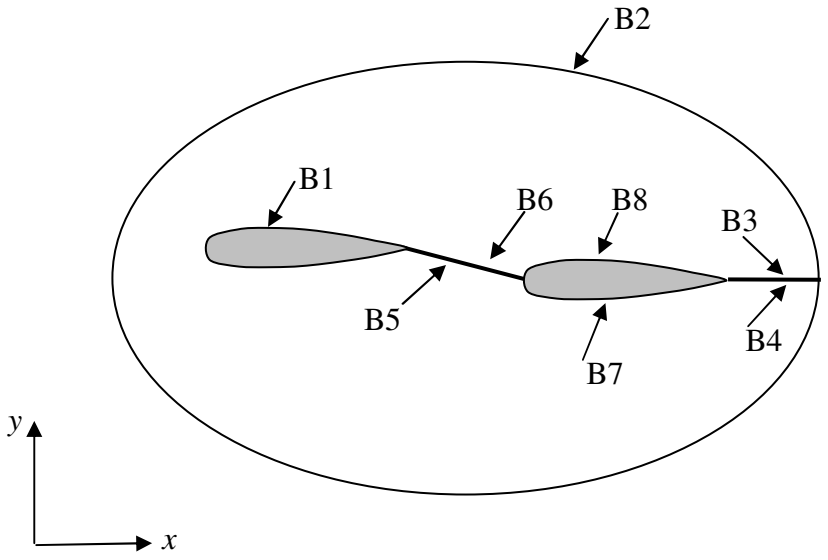


**Figure 1.13** An example to a multiply connected region with branch-cuts

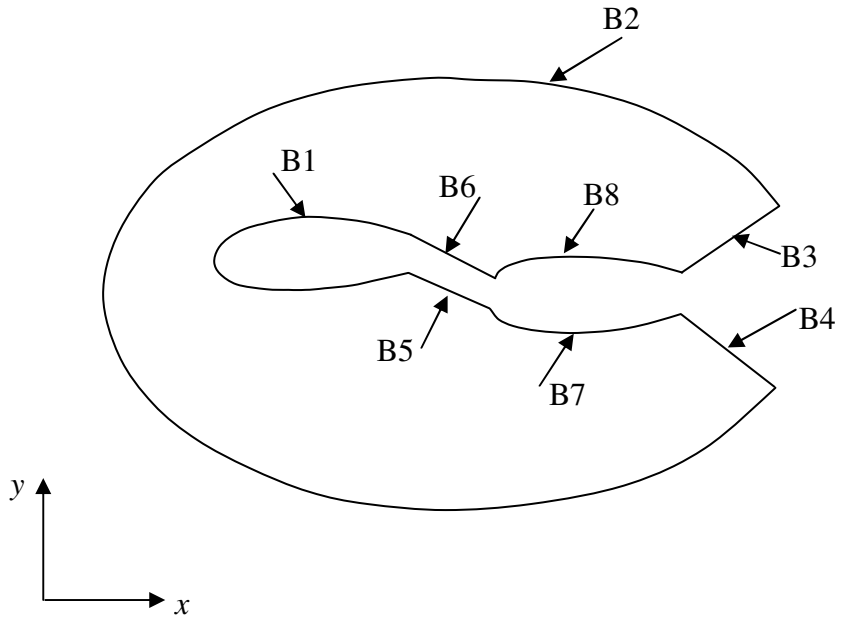


**Figure 1.14** Unwrapping of Figure 1.13

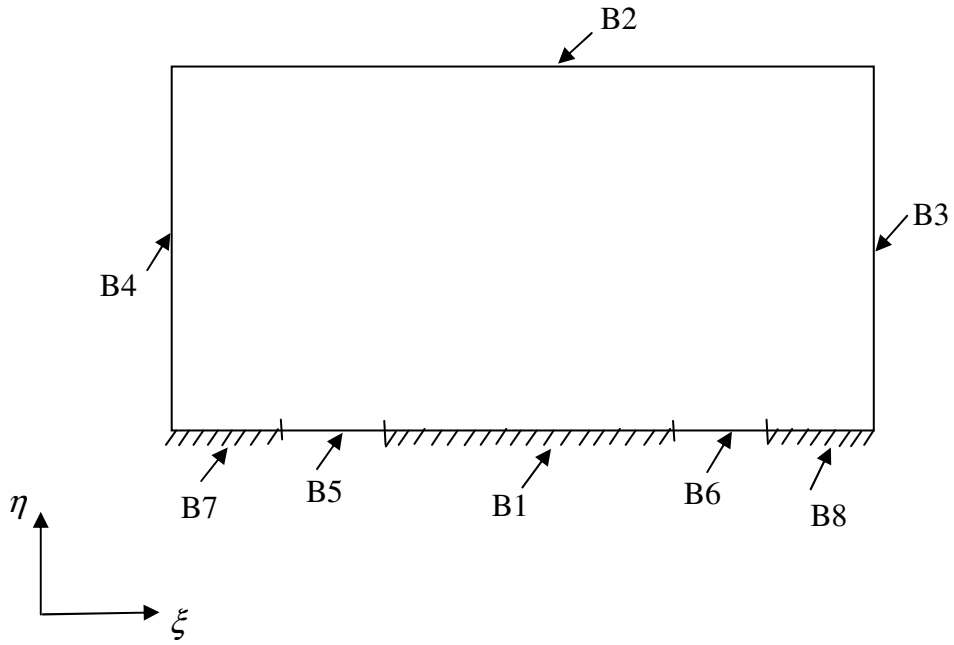


**Figure 1.15** Computational domain of Figure 1.13

Advantages and disadvantages of elliptic grid generation technique are given below.

*Advantages*

1) Distribution of generated grid points is smooth.
2) There are many options available for grid clustering and surface orthogonality.
3) It can be extended to three dimensional problems.

*Disadvantages*

1) It needs more computational time when compared to algebraic or hyperbolic grid generation technique.
2) Forcing function used in this technique can not be specified easily.
3) Metrics should be calculated numerically.

#### 1.2.2.3.2 Parabolic Grid Generation Technique

The idea behind using parabolic grid generation technique is to combine the benefits of elliptic and hyperbolic systems. One of the benefits of the elliptic system is its diffusive nature, that is; a boundary discontinuity does not propagate into the domain. However, a boundary discontinuity may propagate into the domain when hyperbolic grid generation technique is used. When the relative computation time is compared, hyperbolic grid generation is much faster than elliptic grid generation. Since the parabolic grid generation technique is combination of two, it has a diffusive nature and needs less computer time when compared with elliptic grid generation. Parabolic system includes second order derivatives, which result in natural diffusion of

propagation of discontinuities and uses marching schemes to solve parabolic partial differential equations which reduces computational time.
Much work needs to be done for perfection and robustness of this technique, especially in relation to the specification of control functions [1].

#### 1.2.2.3.3 Hyperbolic Grid Generation Technique

For open domains, where the outer boundary can not be prescribed, hyperbolic partial differential equations are used [1]. It is computationally faster than other partial differential equation techniques since it uses a marching scheme. Its main advantages are its speed and orthogonality of grid lines. A major disadvantage of this system is that the outer boundary can not be defined, however this is not a critical problem for external aerodynamic problems. The grid can be marched out away from the body beyond a reasonable range. The details of hyperbolic grid generation technique are given throughout the study.

## 1.3 Review of Literature

The numerical generation of a boundary fitted coordinate system has been attracting many researchers studying on various branches of science and engineering. The idea behind the numerical generation of boundary fitted coordinate system is to transform an arbitrary shaped physical domain into a simple canonical domain, such as rectangle or a unit circle. So, the problems under consideration are modified in terms of the new coordinates and their solution can be obtained more easily. The easiness of the solution stems from the simpler boundary conditions in the canonical domain. After the problem is solved in canonical domain, it is transformed back to the original physical domain.

Most of the Computational Fluid Dynamics (CFD) problems are solved by finite difference techniques. In most of them, defining boundary conditions is complicated. So, the domain transformation can simplify the definition and CFD coding. The numerical generation of boundary fitted coordinate system is also an important tool for finite element grid generations.

There are three ways to generate boundary fitted coordinate system. First one is algebraic transformations. The second is constructing boundary fitted coordinate systems using partial differential equations which include elliptic, parabolic and hyperbolic equations. Chu [6] seems to be the first to apply this method in finite difference solution of two dimensional flow problems. Then, Mastin and Thompson [7] extended this idea to three dimensional problems. Sparis [8] used biharmonic equations for boundary fitted coordinate system generation as a third way. With this method, the mesh point location on the boundary and the angle of intersection of coordinate lines with the boundary can be controlled completely. There have been many investigations on controlling the distribution of coordinate lines and generating orthogonal systems. A very comprehensive survey about this subject was presented by Thompson et. al. It contains nearly four hundred references.

A procedure for generating body-fitted orthogonal grids has been advanced by Graves [9] which has been originated from a scheme developed some years earlier by McNally [10]. According to the procedure given in [9], normals are created from the initial distribution of points on the body surface to an adjacent level line. Then, normals are projected to another adjacent level line. This procedure continues until normals are completely formed between all of the level lines between the body and outer boundary.

The notion of using hyperbolic equations to construct grids has been proposed by Steger and Chausee in 1980[11]. In that approach, an initial surface is propagated outward by using volume and orthogonality constraints. This

technique has three limitations. Discontinuities in the initial data are propagated, crossing grid lines may occur and boundaries other than an initial surface may not be specified.
The first two of these may be overcome by numerical techniques as demonstrated by Kinsey and Barth [12]. A partial solution to the third problem was achieved by Nakamura who considered parabolic-hyperbolic schemes.

In order to construct meshes in two dimensional regions, a method which applies a hyperbolic grid generation scheme was developed by Cordova and Barth in 1988 [13]. Actually, this approach uses the theory developed by Steger and Chausee [11] and the algorithm outlined by Kinsey and Barth [12]. This method brings improvements to local grid control. This was achieved by using a new method to compute the volume source term and adding angle control source term to the equations. These new approaches prevented propagation of initial discontinuities and formation of grid shocks.

Three dimensional body fitted coordinates using hyperbolic partial differential equations were first generated by Steger and Rizk [14]. This method works well for bodies with sharp edges and bodies which are concave. Further enhancements to the three-dimensional hyperbolic grid generation were made by Chan and Steger [15]. High quality three dimensional grids were generated by applying metric correction procedures, local treatment of severe convex corners, and new extrapolation treatments of floating and axis boundaries.

### 1.4 Present Study

This thesis analyzes procedure of hyperbolic grid generation formulated from two constraints, which specify grid orthogonality and cell-volume. Throughout this study, all derivations are investigated for two and three dimensional hyperbolic grid generation.

In Chapter 1, brief information about grid generation and grid generation techniques are given. Also, a literature survey about mesh generation is presented.

In Chapter 2, the governing equations of hyperbolic grid generation technique are introduced and derivation of these governing equations for two dimensional and three dimensional hyperbolic grid generation is given. The governing non-linear equations are linearized and equations are rearranged in order to ease the numerical solution of the system. Also, linearization procedure for nonlinear two dimensional and three dimensional equations is discussed in this chapter.

In Chapter 3, the solution algorithm for the solution of the linearized system is explained. Solution algorithm includes cell volume specification procedure, which is one of the critical factors affecting the grid quality, for two and three dimensional hyperbolic grid generation. Besides, it includes implementation of boundary conditions for two and three dimensional mesh.

In Chapter 4, procedure of smoothing and clustering grids is presented. Clustering method in $\xi$ and $\eta$ directions for two dimensional hyperbolic grid generation is explained.

In Chapter 5, various application results, that is examples from two dimensional and three dimensional grids produced by the hyperbolic grid generator are given.

Finally, conclusion of this study is given in Chapter 6.

# CHAPTER 2

# HYPERBOLIC GRID GENERATION TECHNIQUE

## 2.1 Two Dimensional Hyperbolic Grid Generation Technique

For open domains, where the outer boundary can not be prescribed, hyperbolic partial differential equations may be developed to provide the required grid generators. Since a marching procedure is used to solve such a system, computationally they are faster [1].

The mathematical development is based on two constraints. First one depends on orthogonality of the grid lines at the surface and in the interior domain. Mathematically, if two lines are orthogonal with each other, multiplication of their slopes should be equal to -1. The slope of constant $\xi$ and $\eta$ lines are determined as follows. The physical and computational domains for O-type grids are shown in Figures 2.1 and 2.2 respectively and the physical and computational domains for C-type grids are shown in Figures 2.3 and 2.4 respectively. In these figures *x-y* coordinate system (physical space) and $\xi - \eta$ coordinate system (computational space) are shown.

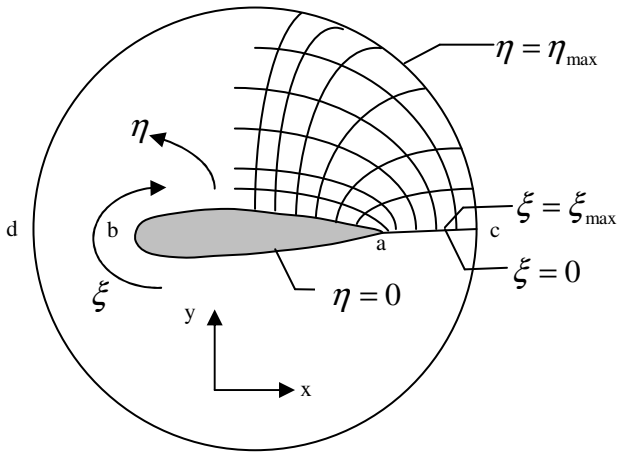


**Figure 2.1** Physical domain of an O-type grid

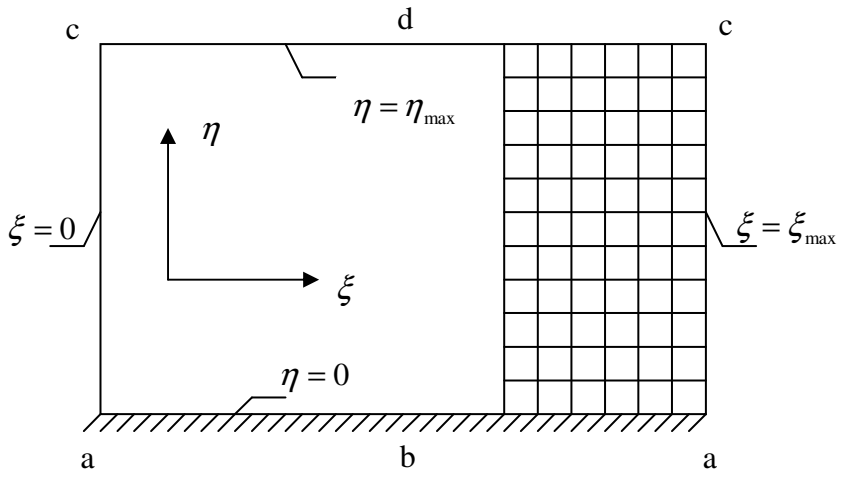


**Figure 2.2** Computational domain of Figure 2.1

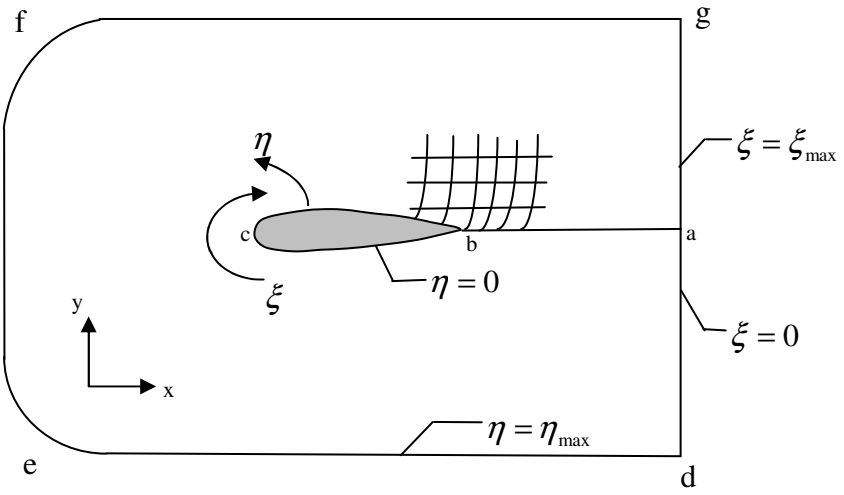


**Figure 2.3** Physical domain of a C-type grid

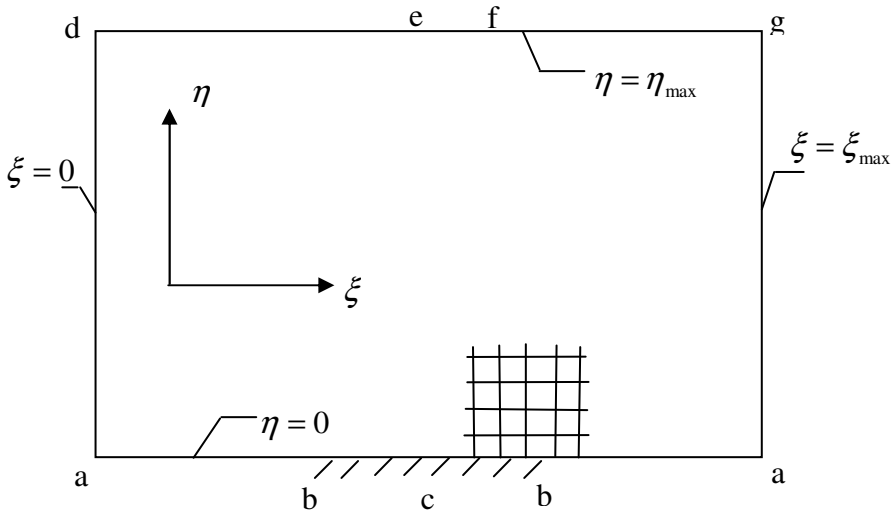


**Figure 2.4** Computational domain of Figure 2.3

Along a line of constant $\xi$, the differential $d\xi$ is zero. Considering a 2-D problem, it is possible to write,

$$d\xi = \xi_x dx + \xi_y dy \tag{2.1}$$

so that

$$\left.\frac{dy}{dx}\right|_{\xi=c} = -\frac{\xi_x}{\xi_y} \tag{2.2}$$

Similarly, for lines of constant $\eta$,

$$d\eta = \eta_x dx + \eta_y dy \tag{2.3}$$

so that

$$\left.\frac{dy}{dx}\right|_{\eta=c} = -\frac{\eta_x}{\eta_y} \tag{2.4}$$

The orthogonality condition can now be imposed as,

$$\left.\frac{dy}{dx}\right|_{\xi=c} \left.\frac{dy}{dx}\right|_{\eta=c} = -1 \tag{2.5}$$

Now using Equations (2.2) and (2.4), the above equation takes the following form.

$$\left(-\frac{\xi_x}{\xi_y}\right)\left(-\frac{\eta_x}{\eta_y}\right) = -1 \tag{2.6}$$

or

$$\xi_x \eta_x + \xi_y \eta_y = 0 \tag{2.7}$$

At this point, it is useful to give Jacobian of transformation for clear understanding. This transformation procedure is given below.

### 2.1.1 Jacobian of Transformation

Relations between the physical and computational spaces can be defined as follows,

$$\xi = \xi(x, y) \tag{2.8a}$$

$$\eta = \eta(x, y) \tag{2.8b}$$

The chain rule for partial differentiation yield the following expression,

$$\frac{\partial}{\partial x} = \frac{\partial \xi}{\partial x} \cdot \frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x} \cdot \frac{\partial}{\partial \eta} \tag{2.9}$$

The partial derivatives will be denoted using the subscript notation, i.e, $\partial \xi / \partial x = \xi_x$ . In this case, Equation (2.9) can be written as

$$\frac{\partial}{\partial x} = \xi_x \cdot \frac{\partial}{\partial \xi} + \eta_x \cdot \frac{\partial}{\partial \eta} \tag{2.10}$$

and similarly,

$$\frac{\partial}{\partial y} = \xi_y \cdot \frac{\partial}{\partial \xi} + \eta_y \cdot \frac{\partial}{\partial \eta} \tag{2.11}$$

In Equations (2.10) and (2.11), terms such as $\xi_x$, $\xi_y$, $\eta_x$ and $\eta_y$ appear. These terms are defined as the metrics of transformation and can be approximated by using finite difference equations.

For example,

$$\xi_x = \frac{\partial \xi}{\partial x} \cong \frac{\Delta \xi}{\Delta x} \tag{2.12}$$

The above expression indicates that the metrics represent the ratio of arc lengths in the computational space to that of the physical space.

By using Equations (2.8a) and (2.8b), the following differential equations are obtained.

$$d\xi = \xi_x dx + \xi_y dy \tag{2.13}$$

$$d\eta = \eta_x dx + \eta_y dy \tag{2.14}$$

which can be written in matrix form as,

$$\begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{2.15}$$

Reversing the role of independent variables, i.e,

$$x = x(\xi, \eta) \tag{2.16}$$

$$y = y(\xi, \eta) \tag{2.17}$$

the following equations can be written,

$$dx = x_\xi d\xi + x_\eta d\eta \tag{2.18}$$

$$dy = y_\xi d\xi + y_\eta d\eta \tag{2.19}$$

or in matrix form,

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \tag{2.20}$$

If equation (2.20) is solved for $\begin{bmatrix} d\xi \\ d\eta \end{bmatrix}$ matrix, the following result is obtained.

$$\begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \frac{1}{x_\xi y_\eta - x_\eta y_\xi} \begin{bmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{2.21}$$

If the above equation and equation (2.15) are equated each other, following relation is obtained.

$$\begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} = \frac{1}{x_\xi y_\eta - x_\eta y_\xi} \begin{bmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{2.22}$$

From the above relation, the following transformations are obtained.

$$\xi_x = J \cdot y_\eta \tag{2.23}$$

$$\xi_y = -J \cdot x_\eta \tag{2.24}$$

$$\eta_x = -J \cdot y_\xi \tag{2.25}$$

$$\eta_y = J \cdot x_\xi \tag{2.26}$$

where

$$J = \frac{1}{x_\xi y_\eta - y_\xi x_\eta}$$

is the Jacobian of transformation.

In literature, the Jacobian $J$ is regarded as the ratio of the areas in two dimension or volumes in three dimensions in the computational space to that of the physical space.

Equation (2.7) is rearranged using Equations (2.23) through (2.26) and the following equation is obtained.

$$x_\xi x_\eta + y_\xi y_\eta = 0 \tag{2.27}$$

This equation is the governing equation related to the orthogonality constraint.

As an alternative, the orthogonality condition can be obtained from the vector approach. By the definition of position vector, $\vec{r}$, the orthogonlity condition can be obtained from the dot product of vectors defining change in $\xi$ and $\eta$ directions.

$$\vec{r}_\xi \cdot \vec{r}_\eta = 0 \Rightarrow x_\xi x_\eta + y_\xi y_\eta = 0 \tag{2.28}$$

These definitions are shown in Figure 2.5.

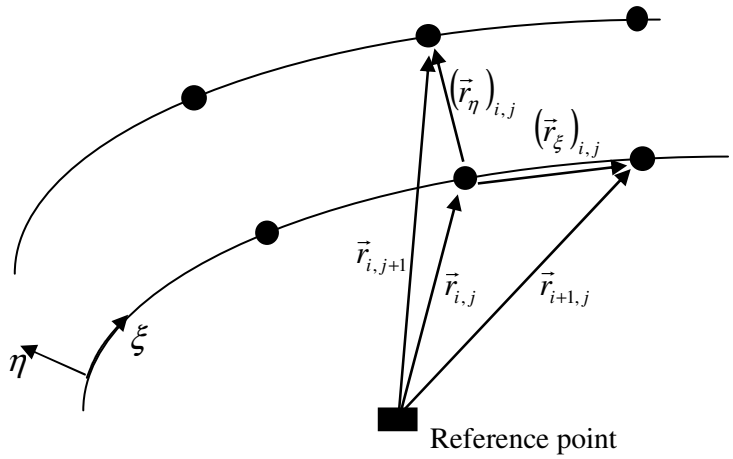


**Figure 2.5** Definition of position vectors

The second constraint must include some geometrical consideration. To achieve this objective, two approaches, i.e ***cell area approach*** & ***arc length approach***, can be found in literature. In the first approach, the Jacobian of transformation is specified. Various schemes can be used for this purpose which will be discussed briefly in the following chapters. In the second approach, the arc length from one grid point to the next is prescribed. This procedure will be referred to as the ***arc length approach***. In this thesis, the first approach was used.

In the first approach, the definition of the Jacobian of transformation (inverse of Jacobian function defines cell area in 2-D or cell volume in 3-D) is used as the second hyperbolic equation, i.e.

$$x_\xi y_\eta - x_\eta y_\xi = F(\xi,\eta) \tag{2.29}$$

where $F(\xi,\eta)$ denotes the cell area function. Thus, the system of hyperbolic equations which is to be solved for the grid point distribution in the physical space,

$$x_\xi x_\eta + y_\xi y_\eta = 0 \tag{2.30a}$$

$$x_\xi y_\eta - x_\eta y_\xi = F(\xi,\eta) \tag{2.30b}$$

In order to solve this hyperbolic system, the reciprocal of the Jacobian of transformation, i.e. F, must be provided. As mentioned before, various schemes for this purpose will be discussed in next chapter.

## 2.2 Three Dimensional Hyperbolic Grid Generation Technique

The body surface is chosen to coincide with $\zeta(x, y, z) = 0$ and the surface grid line distributions of $\xi = const$ and $\eta = const$ are user specified [14].

The outer boundary is not specified; however it is sufficient if the outer boundary is far from the inner boundary. Selecting $\zeta$ as the marching direction, partial differential equations are solved in order to obtain surfaces of constant $\xi$, $\eta$ and $\zeta$ to form a non-singular mesh system.

The set of governing equations (2.30) of two dimensional hyperbolic grid generation technique are rewritten here for better understanding of the three dimensional grid generation equations.

$$x_\xi x_\eta + y_\xi y_\eta = 0 \tag{2.30a}$$

$$x_\xi y_\eta - x_\eta y_\xi = F(\xi,\eta) \tag{2.30b}$$

As it is explained before, one of the above equations is related with the orthogonality and the other is cell area constraint. In three dimensions, however, there are three orthogonality relations and one cell volume constraint. Therefore, there are four equations available to find the three unknowns x, y, z at any point. So, one equation is redundant. Since $\zeta$ is selected to be the marching direction, the orthogonality relations that involve $\zeta$ should be used. This leads to the governing equations below.

$$\vec{r}_\xi \cdot \vec{r}_\zeta = 0 \tag{2.31}$$

$$\vec{r}_\eta \cdot \vec{r}_\zeta = 0 \tag{2.32}$$

$$\frac{\partial(x, y, z)}{\partial(\xi,\eta,\zeta)} = J^{-1} = \Delta V \tag{2.33}$$

In an open form, Equations (2.31) to (2.33) can be written as,

$$x_\xi x_\zeta + y_\xi y_\zeta + z_\xi z_\zeta = 0 \tag{2.34}$$

$$x_\eta x_\zeta + y_\eta y_\zeta + z_\eta z_\zeta = 0 \tag{2.35}$$

$$x_\xi y_\eta z_\zeta + x_\zeta y_\xi z_\eta + x_\eta y_\zeta z_\xi - x_\xi y_\zeta z_\eta - x_\eta y_\xi z_\zeta - x_\zeta y_\eta z_\xi = \Delta V \tag{2.36}$$

Equations (2.34) and (2.35) represent orthogonality relations between $\xi$ and $\zeta$ and between $\eta$ and $\zeta$, respectively and equation (2.36) is the cell volume constraint.

The system of these partial differential equations is nonlinear and should be linearized. The linearization procedure for two and three dimensional equations is given below.

## 2.3 Linearization of 2-D Hyperbolic Grid Generation Equations

To linearize the equations, Newton's iterative scheme is used. A nonlinear term is approximated according to the following formula.

$$AB = A^{k+1}B^{k} + B^{k+1}A^{k} - A^{k}B^{k} \tag{2.37}$$

where the level *k* is the known state and level *k+1* is the unknown state. For the following notations, superscript *k+1* is dropped. Therefore, any variable without a superscript denotes the unknowns at the *k+1* level. Linearization of the terms in Equations (2.30a) and (2.30b) are given one by one.

*Linearization of Equation (2.30a)*;

With the aid of Equation (2.37), the following relations can be obtained.

$$x_\xi x_\eta = x_\xi x_\eta^k + x_\eta x_\xi^k - x_\xi^k x_\eta^k \tag{2.38}$$

$$y_\xi y_\eta = y_\xi y_\eta^k + y_\eta y_\xi^k - y_\xi^k y_\eta^k \tag{2.39}$$

When these linearized terms are substituted into the Equation (2.30a);

$$x_\xi x_\eta^k + x_\eta x_\xi^k - x_\xi^k x_\eta^k + y_\xi y_\eta^k + y_\eta y_\xi^k - y_\xi^k y_\eta^k = 0 \tag{2.40}$$

is obtained. If Equation (2.30a) is written for level *k*;

$$x_\xi x_\eta + y_\xi y_\eta = 0 \Rightarrow x_\xi^k x_\eta^k + y_\xi^k y_\eta^k = 0 \tag{2.41}$$

is obtained. When this equation is substituted into the Equation (2.40); the linearized form of Equation (2.30a) is obtained.

$$x_\xi x_\eta^k + x_\eta x_\xi^k + y_\xi y_\eta^k + y_\eta y_\xi^k = 0 \tag{2.42}$$

*Linearization of Equation (2.30b);*

With the aid of Equation (2.37), the following relations can be obtained.

$$x_\xi y_\eta = x_\xi y_\eta^k + y_\eta x_\xi^k - x_\xi^k y_\eta^k \tag{2.43}$$

$$x_\eta y_\xi = x_\eta y_\xi^k + y_\xi x_\eta^k - x_\eta^k y_\xi^k \tag{2.44}$$

When these linearized terms are substituted into the Equation (2.30b);

$$x_\xi y_\eta^k + y_\eta x_\xi^k - x_\xi^k y_\eta^k - x_\eta y_\xi^k - y_\xi x_\eta^k + x_\eta^k y_\xi^k = F(\xi,\eta) \tag{2.45}$$

is obtained. If Equation (2.30b) is written for level *k*,

$$x_\xi y_\eta - x_\eta y_\xi = F(\xi,\eta) \Rightarrow x_\xi^k y_\eta^k - x_\eta^k y_\xi^k = F^k(\xi,\eta) \tag{2.46}$$

is obtained. When the above equation is substituted into the Equation (2.45); the linearized form of Equation (2.30b) is obtained as,

$$x_\xi y_\eta^k + y_\eta x_\xi^k - x_\eta y_\xi^k - y_\xi x_\eta^k = F(\xi,\eta) + F^k(\xi,\eta) \tag{2.47}$$

## 2.4 Linearization of 3-D Hyperbolic Grid Generation Equations

Let $x^0$, $y^0$, $z^0$ represent a nearby known state so that,

$$x = x^0 + \Delta x \tag{2.48}$$

$$y = y^0 + \Delta y \tag{2.49}$$

$$z = z^0 + \Delta z \tag{2.50}$$

where $\Delta x$, $\Delta y$ and $\Delta z$ are small. When Equations (2.48), (2.49) and (2.50) are substituted into Equations (2.34), (2.35) and (2.36), linearization procedure is required. Linearization of Equation (2.34) is given in detail as follows.

*Linearization of $x_\xi x_\zeta$ term;*

$$x_\xi x_\zeta = \frac{\partial x}{\partial \xi} \cdot \frac{\partial x}{\partial \zeta} \tag{2.51}$$

$$x_\xi x_\zeta = \frac{\partial \left(x^o + \Delta x\right)}{\partial \xi} \cdot \frac{\partial \left(x^o + \Delta x\right)}{\partial \zeta} \tag{2.52}$$

$$x_\xi x_\zeta = \left(x_\xi^o + \Delta x_\xi\right)\left(x_\zeta^o + \Delta x_\zeta\right) \tag{2.53}$$

$$x_\xi x_\zeta = x_\xi^o x_\zeta^o + x_\zeta^o \Delta x_\xi + x_\xi^o \Delta x_\zeta + \Delta x_\xi \Delta x_\zeta \tag{2.54}$$

Since, $\Delta x_\xi$ and $\Delta x_\zeta$ are small, the term $\Delta x_\xi \Delta x_\zeta$ is neglected and Equation (2.54) takes the following form.

$$x_\xi x_\zeta = x_\xi^o x_\zeta^o + x_\zeta^o \Delta x_\xi + x_\xi^o \Delta x_\zeta \tag{2.55}$$

*Linearization of $y_\xi y_\zeta$ term;*

$$y_\xi y_\zeta = \frac{\partial y}{\partial \xi} \cdot \frac{\partial y}{\partial \zeta} \tag{2.56}$$

$$y_\xi y_\zeta = \frac{\partial \left(y^o + \Delta y\right)}{\partial \xi} \cdot \frac{\partial \left(y^o + \Delta y\right)}{\partial \zeta} \tag{2.57}$$

$$y_\xi y_\zeta = \left(y_\xi^o + \Delta y_\xi\right) \cdot \left(y_\zeta^o + \Delta y_\zeta\right) \tag{2.58}$$

$$y_\xi y_\zeta = y_\xi^o y_\zeta^o + y_\zeta^o \Delta y_\xi + y_\xi^o \Delta y_\zeta + \Delta y_\xi \Delta y_\zeta \tag{2.59}$$

Since, $\Delta y_\xi$ and $\Delta y_\zeta$ are small, the term $\Delta y_\xi \Delta y_\zeta$ can be neglected and Equation (2.59) takes the following form.

$$y_\xi y_\zeta = y_\xi^o y_\zeta^o + y_\zeta^o \Delta y_\xi + y_\xi^o \Delta y_\zeta \tag{2.60}$$

*Linearization of $z_\xi z_\zeta$ term;*

$$z_\xi z_\zeta = \frac{\partial z}{\partial \xi} \cdot \frac{\partial z}{\partial \zeta} \tag{2.61}$$

$$z_\xi z_\zeta = \frac{\partial \left(z^o + \Delta z\right)}{\partial \xi} \cdot \frac{\partial \left(z^o + \Delta z\right)}{\partial \zeta} \tag{2.62}$$

$$z_\xi z_\zeta = \left(z_\xi^o + \Delta z_\xi\right) \cdot \left(z_\zeta^o + \Delta z_\zeta\right) \tag{2.63}$$

$$z_\xi z_\zeta = z_\xi^o z_\zeta^o + z_\zeta^o \Delta z_\xi + z_\xi^o \Delta z_\zeta + \Delta z_\xi \Delta z_\zeta \tag{2.64}$$

Since $\Delta z_\xi$ and $\Delta z_\zeta$ are small, the term $\Delta z_\xi \Delta z_\zeta$ can be eliminated and the above equation takes the following form;

$$z_\xi z_\zeta = z_\xi^o z_\zeta^o + z_\zeta^o \Delta z_\xi + z_\xi^o \Delta z_\zeta \tag{2.65}$$

When these three linearized terms are substituted into Equation (2.34), the linearized form of this equation is obtained as follows.

$$\vec{r}_\xi \cdot \vec{r}_\eta = x_\xi^o x_\zeta^o + x_\zeta^o \Delta x_\xi + x_\xi^o \Delta x_\zeta + y_\xi^o y_\zeta^o + y_\zeta^o \Delta y_\xi + y_\xi^o \Delta y_\zeta + z_\xi^o z_\zeta^o + z_\zeta^o \Delta z_\xi + z_\xi^o \Delta z_\zeta \tag{2.66}$$

In the above equation, the term $x_\xi^o x_\zeta^o + y_\xi^o y_\zeta^o + z_\xi^o z_\zeta^o$ is zero from Equation (2.34). Therefore, the final form of the linearized equation of Equation (2.34) is as follows.

$$x_\xi^o \Delta x_\zeta + x_\zeta^o \Delta x_\xi + y_\xi^o \Delta y_\zeta + y_\zeta^o \Delta y_\xi + z_\zeta^o \Delta z_\xi + z_\xi^o \Delta z_\zeta = 0 \tag{2.67}$$

Linearization of Equation (2.35) and (2.36) can be done, similarly. Linearized terms and final linearized form of Equations (2.35) and (2.36) are given below.

*Result of Linearization of $x_\xi x_\zeta$ term;*

$$x_\eta x_\zeta = x_\eta^o x_\zeta^o + x_\zeta^o \Delta x_\eta + x_\eta^o \Delta x_\zeta \tag{2.68}$$

*Result of Linearization of $y_\eta y_\zeta$ term;*

$$y_\eta y_\zeta = y_\eta^o y_\zeta^o + y_\zeta^o \Delta y_\eta + y_\eta^o \Delta y_\zeta \tag{2.69}$$

*Result of Linearization of* $z_\xi z_\zeta$ *term;*

$$z_\xi z_\zeta = z_\xi^o z_\zeta^o + z_\zeta^o \Delta z_\xi + z_\xi^o \Delta z_\zeta \tag{2.70}$$

When these three linearized terms given in Equations (2.68), (2.69), (2.70) are substituted into Equation (2.35), the linearized form of Equation (2.35) is obtained as follows.

$$\vec{r}_\eta \cdot \vec{r}_\zeta = x_\eta^o x_\zeta^o + x_\zeta^o \Delta x_\eta + x_\eta^o \Delta x_\zeta + y_\eta^o y_\zeta^o + y_\zeta^o \Delta y_\eta + y_\eta^o \Delta y_\zeta + z_\xi^o z_\zeta^o$$
$$+ z_\zeta^o \Delta z_\xi + z_\xi^o \Delta z_\zeta = 0 \tag{2.71}$$

In the above equation, the term $x_\eta^o x_\zeta^o + y_\eta^o y_\zeta^o + z_\xi^o z_\zeta^o$ is zero from Equation (2.35). Therefore, the final linearized form of Equation (2.35) is,

$$x_\zeta^o \Delta x_\eta + x_\eta^o \Delta x_\zeta + y_\zeta^o \Delta y_\eta + y_\eta^o \Delta y_\zeta + z_\zeta^o \Delta z_\xi + z_\xi^o \Delta z_\zeta = 0 \tag{2.72}$$

Linearization of Equation (2.36) is more complex, since there are products of three terms, however the linearization procedure is the same.

*Result of Linearization of* $x_\xi y_\eta z_\zeta$ *term;*

$$x_\xi y_\eta z_\zeta = x_\xi^o y_\eta^o z_\zeta^o + y_\eta^o z_\zeta^o \Delta x_\xi + x_\xi^o z_\zeta^o \Delta y_\eta + x_\xi^o y_\eta^o \Delta z_\zeta \tag{2.73}$$

*Result of Linearization of* $x_\zeta y_\xi z_\eta$ *term;*

$$x_\zeta y_\xi z_\eta = x_\zeta^o y_\xi^o z_\eta^o + y_\xi^o z_\eta^o \Delta x_\zeta + x_\zeta^o z_\eta^o \Delta y_\xi + x_\zeta^o y_\xi^o \Delta z_\eta \tag{2.74}$$

*Result of Linearization of* $x_\eta y_\zeta z_\xi$ *term;*

$$x_\eta y_\zeta z_\xi = x_\eta^o y_\zeta^o z_\xi^o + y_\zeta^o z_\xi^o \Delta x_\eta + x_\eta^o z_\xi^o \Delta y_\zeta + x_\eta^o y_\zeta^o \Delta z_\xi \tag{2.75}$$

*Result of Linearization of* $x_\xi y_\zeta z_\eta$ *term;*

$$x_\xi y_\zeta z_\eta = x_\xi^o y_\zeta^o z_\eta^o + y_\zeta^o z_\eta^o \Delta x_\xi + x_\xi^o z_\eta^o \Delta y_\zeta + x_\xi^o y_\zeta^o \Delta z_\eta \tag{2.76}$$

*Result of Linearization of* $x_\eta y_\xi z_\zeta$ *term:*

$$x_\eta y_\xi z_\zeta = x_\eta^o y_\xi^o z_\zeta^o + y_\xi^o z_\zeta^o \Delta x_\eta + x_\eta^o z_\zeta^o \Delta y_\xi + x_\eta^o y_\xi^o \Delta z_\zeta \tag{2.77}$$

*Result of Linearization of* $x_\zeta y_\eta z_\xi$ *term;*

$$x_\zeta y_\eta z_\xi = x_\zeta^o y_\eta^o z_\xi^o + y_\eta^o z_\xi^o \Delta x_\zeta + x_\zeta^o z_\xi^o \Delta y_\eta + x_\zeta^o y_\eta^o \Delta z_\xi \tag{2.78}$$

When Equations (2.73), (2.74), (2.75), (2.76), (2.77) and (2.78) are substituted into the Equation (2.36), the following form of Equation (2.36) is obtained.

$$\begin{aligned}\Delta V = {} & x_\xi^o y_\eta^o z_\zeta^o + y_\eta^o z_\zeta^o \Delta x_\xi + x_\xi^o z_\zeta^o \Delta y_\eta + x_\xi^o y_\eta^o \Delta z_\zeta + x_\zeta^o y_\xi^o z_\eta^o + y_\xi^o z_\eta^o \Delta x_\zeta + x_\zeta^o z_\eta^o \Delta y_\xi \\ & + x_\zeta^o y_\xi^o \Delta z_\eta + x_\eta^o y_\zeta^o z_\xi^o + y_\zeta^o z_\xi^o \Delta x_\eta + x_\eta^o z_\xi^o \Delta y_\zeta + x_\eta^o y_\zeta^o \Delta z_\xi - x_\xi^o y_\zeta^o z_\eta^o - y_\zeta^o z_\eta^o \Delta x_\xi \\ & - x_\xi^o z_\eta^o \Delta y_\zeta - x_\xi^o y_\zeta^o \Delta z_\eta - x_\eta^o y_\xi^o z_\zeta^o - y_\xi^o z_\zeta^o \Delta x_\eta - x_\eta^o z_\zeta^o \Delta y_\xi - x_\eta^o y_\xi^o \Delta z_\zeta - x_\zeta^o y_\eta^o z_\xi^o \\ & - y_\eta^o z_\xi^o \Delta x_\zeta - x_\zeta^o z_\xi^o \Delta y_\eta - x_\zeta^o y_\eta^o \Delta z_\xi\end{aligned} \tag{2.79}$$

Notice that,

$$x_\xi^o y_\eta^o z_\zeta^o + x_\zeta^o y_\xi^o z_\eta^o + x_\eta^o y_\zeta^o z_\xi^o - x_\xi^o y_\zeta^o z_\eta^o - x_\eta^o y_\xi^o z_\zeta^o - x_\zeta^o y_\eta^o z_\xi^o = \Delta V_o \tag{2.80}$$

Therefore, the final linearized form of Equation (2.36) is as follows.

$$y_\eta^o z_\zeta^o \Delta x_\xi + x_\xi^o z_\zeta^o \Delta y_\eta + x_\xi^o y_\eta^o \Delta z_\zeta + y_\xi^o z_\eta^o \Delta x_\zeta + x_\zeta^o z_\eta^o \Delta y_\xi + x_\zeta^o y_\xi^o \Delta z_\eta + y_\zeta^o z_\xi^o \Delta x_\eta$$
$$+ x_\eta^o z_\xi^o \Delta y_\zeta + x_\eta^o y_\zeta^o \Delta z_\xi - y_\zeta^o z_\eta^o \Delta x_\xi - x_\xi^o z_\eta^o \Delta y_\zeta - x_\xi^o y_\zeta^o \Delta z_\eta - y_\xi^o z_\zeta^o \Delta x_\eta - x_\eta^o z_\zeta^o \Delta y_\xi$$
$$- x_\eta^o y_\xi^o \Delta z_\zeta - y_\eta^o z_\xi^o \Delta x_\zeta - x_\zeta^o z_\xi^o \Delta y_\eta - x_\zeta^o y_\eta^o \Delta z_\xi = \Delta V - \Delta V_o \tag{2.81}$$

# CHAPTER 3

# SOLUTION ALGORITHM

## 3.1 Two Dimensional Hyperbolic Grid Generation Algorithm

The linearized form of the system of hyperbolic equations (2.30) which is to be solved for the grid point distribution in the physical space can be written in a compact form as,

$$[A]R_\xi + [B]R_\eta = H \tag{3.1}$$

where

$$R = \begin{bmatrix} x \\ y \end{bmatrix} \qquad A = \begin{bmatrix} x_\eta^k & y_\eta^k \\ y_\eta^k & -x_\eta^k \end{bmatrix} \qquad B = \begin{bmatrix} x_\xi^k & y_\xi^k \\ -y_\xi^k & x_\xi^k \end{bmatrix} \quad H = \begin{bmatrix} 0 \\ F + F^k \end{bmatrix}$$

If the eigenvalues of $[B]^{-1}[A]$ are real, the system given in Equation 3.1 is hyperbolic. Noting that,

$$[B]^{-1} = \frac{1}{\left(x_\xi^k\right)^2 + \left(y_\xi^k\right)^2} \begin{bmatrix} x_\xi^k & -y_\xi^k \\ y_\xi^k & x_\xi^k \end{bmatrix} \tag{3.2}$$

and

$$[C] = [B]^{-1}[A] = \frac{1}{\left(x_\xi^k\right)^2 + \left(y_\xi^k\right)^2} \begin{bmatrix} x_\xi^k x_\eta^k - y_\xi^k y_\eta^k & x_\xi^k y_\eta^k + x_\eta^k y_\xi^k \\ x_\xi^k y_\eta^k + x_\eta^k y_\xi^k & -\left(x_\xi^k x_\eta^k - y_\xi^k y_\eta^k\right) \end{bmatrix} \tag{3.3}$$

The eigenvalues of [C] are,

$$\lambda_{1,2} = \pm \left[ \frac{\left(x_\eta^k\right)^2 + \left(y_\eta^k\right)^2}{\left(x_\xi^k\right)^2 + \left(y_\xi^k\right)^2} \right]^{\frac{1}{2}} \tag{3.4}$$

It is recognized that the eigenvalues are always real; however, it is required that

$$\left(x_{\xi}^{k}\right)^2+\left(y_{\xi}^{k}\right)^2\neq 0 \tag{3.5}$$

In order to obtain finite difference form of Equation (3.1), some numerical approximations are needed. Therefore, for the approximation of $\eta$ and $\xi$ derivatives, first order backward difference and second order central difference approximations are used respectively. Thus, Equation 3.1 can be rewritten as follows.

$$[A]\frac{R_{i+1,j}-R_{i-1,j}}{2\Delta\xi}+[B]\frac{R_{i,j}-R_{i,j-1}}{\Delta\eta}=H_{i,j} \tag{3.6}$$

If both sides of the above equation are multiplied by inverse of B matrix, the following equation is obtained.

$$[B]^{-1}[A]\frac{R_{i+1,j}-R_{i-1,j}}{2\Delta\xi}+[I]\frac{R_{i,j}-R_{i,j-1}}{\Delta\eta}=[B]^{-1}H_{i,j} \tag{3.7}$$

where [I] is the identity matrix. In this equation, $[B]^{-1}$ and $[A]$ matrices are evaluated at (j-1) grid line, that is at known state. If the above equation is rearranged, following equation is obtained.

$$-\frac{1}{2\Delta\xi}[C]_{i,j-1}R_{i-1,j}+\frac{R_{i,j}}{\Delta\eta}+\frac{1}{2\Delta\xi}[C]_{i,j-1}R_{i+1,j}=[B]^{-1}_{i,j-1}H_{i,j}+\frac{R_{i,j-1}}{\Delta\eta} \tag{3.8}$$

More simplification can be done by taking $\Delta\xi=\Delta\eta=1$ and by using following definitions.

$$[AA]=-\frac{1}{2}[C]_{i,j-1} \tag{3.9}$$

$$[BB]=[I] \tag{3.10}$$

$$[CC]=\frac{1}{2}[C]_{i,j-1} \tag{3.11}$$

$$[DD]=[B]^{-1}_{i,j-1}[H]_{i,j}+R_{i,j-1} \tag{3.12}$$

Thus, Equation (3.8) takes the following form.

$$[AA]R_{i-1,j} + [BB]R_{i,j} + [CC]R_{i+1,j} = [DD]_{i,j} \tag{3.13}$$

In order to solve this equation to find $R_{i,j}$, that is the coordinates of grid points at j grid line, a block tridiagonal system should be solved. This block-tridiagonal system is given below.

$$\begin{bmatrix} [BB]_2 & [CC]_2 & & & & & \\ [AA]_3 & [BB]_3 & [CC]_3 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & [AA]_{im-2} & [BB]_{im-2} & [CC]_{im-2} & \\ & & & & [AA]_{im-1} & [BB]_{im-1} \end{bmatrix} \begin{bmatrix} R_2 \\ R_3 \\ \vdots \\ \vdots \\ R_{im-2} \\ R_{im-1} \end{bmatrix} = \begin{bmatrix} [DD]_2 \\ [DD]_3 \\ \vdots \\ \vdots \\ [DD]_{im-2} \\ [DD]_{im-1} \end{bmatrix} \tag{3.14}$$

Before solving this system, the right hand side matrices which are affected by boundary conditions should be updated according to the following formula.

$$[DD]_2 = [DD]_2 - [AA]_2 R_1 \tag{3.15}$$

$$[DD]_{im-1} = [DD]_{im-1} - [CC]_{im-1} R_{im} \tag{3.16}$$

With the known grid point distribution at the surface and boundaries, $x_\xi^k$ and $y_\xi^k$ terms can be calculated easily. At boundaries, forward or backward difference approximations are used. For the internal grid points, $x_\xi^k$ and $y_\xi^k$ are computed by central difference approximation.

In order to calculate $x_\eta^k$ and $y_\eta^k$ by using difference approximations, the grid point distribution on the next level line in the marching direction is needed. Since the aim is to find the grid point distribution on the next level line and this

distribution is not known, difference approximations for $x_\eta^k$ and $y_\eta^k$ can not be used. However, when Equations (2.41) and (2.46) are solved simultaneously, the following expressions for $x_\eta^k$ and $y_\eta^k$ terms are obtained.

$$x_\eta^k = -\frac{y_\xi^k F^k}{\left(x_\xi^k\right)^2 + \left(y_\xi^k\right)^2} \tag{3.17}$$

$$y_\eta^k = \frac{x_\xi^k F^k}{\left(x_\xi^k\right)^2 + \left(y_\xi^k\right)^2} \tag{3.18}$$

After block matrices are formed using the above equations, the block tridiagonal system can be solved by any standard technique. Such a procedure is presented in reference [1].

### 3.2 Three Dimensional Hyperbolic Grid Generation Algorithm

The linearized form of the governing equations was given in Equations (2.67), (2.72) and (2.81). The system of these linearized equations can be written in matrix form as follows.

$$A_o\left(\vec{r} - \vec{r}_o\right)_\xi + B_o\left(\vec{r} - \vec{r}_o\right)_\eta + C_o\left(\vec{r} - \vec{r}_o\right)_\zeta = \vec{f} \tag{3.19}$$

where,

$$A_o = \begin{bmatrix} x_\zeta & y_\zeta & z_\zeta \\ 0 & 0 & 0 \\ \left(y_\eta z_\zeta - y_\zeta z_\eta\right) & \left(x_\zeta z_\eta - x_\eta z_\zeta\right) & \left(x_\eta y_\zeta - x_\zeta y_\eta\right) \end{bmatrix}_o$$

$$B_o = \begin{bmatrix} 0 & 0 & 0 \\ x_\zeta & y_\zeta & z_\zeta \\ \left(y_\zeta z_\xi - y_\xi z_\zeta\right) & \left(x_\xi z_\zeta - x_\zeta z_\xi\right) & \left(x_\zeta y_\xi - x_\xi y_\zeta\right) \end{bmatrix}_o$$

$$C_o = \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ \left(y_\xi z_\eta - y_\eta z_\xi\right) & \left(x_\eta z_\xi - x_\xi z_\eta\right) & \left(x_\xi y_\eta - x_\eta y_\xi\right) \end{bmatrix}_o$$

$$\vec{f} = \begin{bmatrix} -\left(\dfrac{\partial \vec{r}}{\partial \xi} \cdot \dfrac{\partial \vec{r}}{\partial \zeta}\right)_o \\ -\left(\dfrac{\partial \vec{r}}{\partial \eta} \cdot \dfrac{\partial \vec{r}}{\partial \zeta}\right)_o \\ \Delta V - \Delta V_o \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \Delta V - \Delta V_o \end{bmatrix}$$

Notice that

$$\left(\vec{r} - \vec{r}_o\right)_\xi = \Delta \vec{r}_\xi \tag{3.20}$$

$$\left(\vec{r} - \vec{r}_o\right)_\eta = \Delta \vec{r}_\eta \tag{3.21}$$

$$\left(\vec{r} - \vec{r}_o\right)_\zeta = \Delta \vec{r}_\zeta \tag{3.22}$$

where $\Delta \vec{r} = (\Delta x, \Delta y, \Delta z)^t$ and the first, second and third rows of these matrices correspond to the first orthogonality, second orthogonality and the volume constraints, respectively.

If Equation (3.19) is rewritten in explicit form as,

$$A_o \vec{r}_\xi - A_o \vec{r}_{o\xi} + B_o \vec{r}_\eta - B_o \vec{r}_{o\eta} + C_o \vec{r}_\zeta - C_o \vec{r}_{o\zeta} = \vec{f} \tag{3.23}$$

If the above equation is rearranged such that the unknown values are on the left hand side and known values are on the right hand side, it is possible to obtain

$$A_o \vec{r}_\xi + B_o \vec{r}_\eta + C_o \vec{r}_\zeta = \vec{f} + A_o \vec{r}_{o\xi} + B_o \vec{r}_{o\eta} + C_o \vec{r}_{o\zeta} = \vec{e} \tag{3.24}$$

In order to obtain the final form of matrix system, $\vec{e}$ should be derived. The derivation is given in detail as follows.

$$A_o \vec{r}_{o\xi} = \begin{bmatrix} x_\zeta & y_\zeta & z_\zeta \\ 0 & 0 & 0 \\ \left(y_\eta z_\zeta - y_\zeta z_\eta\right) & \left(x_\zeta z_\eta - x_\eta z_\zeta\right) & \left(x_\eta y_\zeta - x_\zeta y_\eta\right) \end{bmatrix}_o \cdot \begin{bmatrix} x_\xi \\ y_\xi \\ z_\xi \end{bmatrix}_o$$

$$= \begin{bmatrix} x_\zeta x_\xi + y_\zeta y_\xi + z_\zeta z_\xi \\ 0 \\ x_\xi\left(y_\eta z_\zeta - y_\zeta z_\eta\right) + y_\xi\left(x_\zeta z_\eta - x_\eta z_\zeta\right) + z_\xi\left(x_\eta y_\zeta - x_\zeta y_\eta\right) \end{bmatrix}_o \tag{3.25}$$

The first row of the column vector is equal to zero as indicated by Equation (2.34) and the third term of this column vector is equal to $\Delta V_o$ as indicated by Equation (2.36). Therefore,

$$A_o \vec{r}_{o\xi} = \begin{bmatrix} 0 \\ 0 \\ \Delta V_o \end{bmatrix} \tag{3.26}$$

$$B_o \vec{r}_{o\eta} = \begin{bmatrix} 0 & 0 & 0 \\ x_\zeta & y_\zeta & z \\ \left(y_\zeta z_\xi - y_\xi z_\zeta\right) & \left(x_\xi z_\zeta - x_\zeta z_\xi\right) & \left(x_\zeta y_\xi - x_\xi y_\zeta\right) \end{bmatrix}_o \cdot \begin{bmatrix} x_\eta \\ y_\eta \\ z_\eta \end{bmatrix}_o$$

$$= \begin{bmatrix} 0 \\ x_\zeta x_\eta + y_\zeta y_\eta + z_\zeta z_\eta \\ x_\eta\left(y_\zeta z_\xi - y_\xi z_\zeta\right) + y_\eta\left(x_\xi z_\zeta - x_\zeta z_\xi\right) + z_\eta\left(x_\zeta y_\xi - x_\xi y_\zeta\right) \end{bmatrix}_o \tag{3.27}$$

Since, $\vec{r}_\eta \cdot \vec{r}_\zeta = x_\zeta x_\eta + y_\zeta y_\eta + z_\zeta z_\eta = 0$ , the second row of the column vector is equal to zero and the third row is $\Delta V_o$ as indicated by equation (2.36). Therefore,

$$B_o \vec{r}_{o\eta} = \begin{bmatrix} 0 \\ 0 \\ \Delta V_o \end{bmatrix} \tag{3.28}$$

$$C_o\vec{r}_{o\zeta} = \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ \left(y_\xi z_\eta - y_\eta z_\xi\right) & \left(x_\eta z_\xi - x_\xi z_\eta\right) & \left(x_\xi y_\eta - x_\eta y_\xi\right) \end{bmatrix}_o \cdot \begin{bmatrix} x_\zeta \\ y_\zeta \\ z_\zeta \end{bmatrix}_o$$

$$= \begin{bmatrix} x_\xi x_\zeta + y_\xi y_\zeta + z_\xi z_\zeta \\ x_\eta x_\zeta + y_\eta y_\zeta + z_\eta z_\zeta \\ x_\zeta\left(y_\xi z_\eta - y_\eta z_\xi\right) + y_\zeta\left(x_\eta z_\xi - x_\xi z_\eta\right) + z_\zeta\left(x_\xi y_\eta - x_\eta y_\xi\right) \end{bmatrix}_o \tag{3.29}$$

First and second row of this column vector are both equal to zero as indicated by Equation (2.34) and (2.35) respectively and the third term of the column vector is equal to $\Delta V_o$ as indicated by Equation (2.36). Therefore,

$$C_o\vec{r}_{o\zeta} = \begin{bmatrix} 0 \\ 0 \\ \Delta V_o \end{bmatrix} \tag{3.30}$$

Now, $\vec{e}$ matrix takes the following form:

$$\vec{e} = \begin{bmatrix} 0 \\ 0 \\ \Delta V - \Delta V_o \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \Delta V_o \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \Delta V_o \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \Delta V_o \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \Delta V + 2\Delta V_o \end{bmatrix} \tag{3.31}$$

Recall that, the equation system (3.24),

$$A_o\vec{r}_\xi + B_o\vec{r}_\eta + C_o\vec{r}_\zeta = \vec{e} \tag{3.24}$$

In order to find *x, y, z* values in the $\zeta$ direction, the term $r_\zeta$ should be calculated. Then, both sides of Equation (3.24) should be multiplied by $C_o^{-1}$ in order to isolate $r_\zeta$.

$$C_o^{-1}A_o\vec{r}_\xi + C_o^{-1}B_o\vec{r}_\eta + \vec{r}_\zeta = C_o^{-1}\vec{e} \tag{3.32}$$

In order to simplify the above equation, $\vec{r}$ can be defined as,

$$\vec{r} = \vec{r}_o + \Delta\vec{r} \tag{3.33}$$

If Equation (3.33) is substituted into Equation (3.32), then the following equation is obtained.

$$C_o^{-1}A_o\left(\vec{r}_o+\Delta\vec{r}\right)_\xi+C_o^{-1}B_o\left(\vec{r}_o+\Delta\vec{r}\right)_\eta+\vec{r}_\zeta=C_o^{-1}\vec{e} \tag{3.34}$$

If equation (3.34) is expanded and rearranged, it is possible to obtain

$$C_o^{-1}A_o\vec{r}_{o\xi}+C_o^{-1}A_o\left(\Delta\vec{r}\right)_\xi+C_o^{-1}B_o\vec{r}_{o\eta}+C_o^{-1}B_o\Delta\vec{r}_\eta+r_\zeta=C_o^{-1}\vec{e} \tag{3.35}$$

$$C_o^{-1}A_o\left(\Delta r\right)_\xi+C_o^{-1}B_o\left(\Delta\vec{r}\right)_\eta+\vec{r}_\zeta=C_o^{-1}\vec{e}-C_o^{-1}B_o\vec{r}_{o\eta}-C_o^{-1}A_o\vec{r}_{o\xi} \tag{3.36}$$

$$C_o^{-1}A_o\left(\Delta\vec{r}\right)_\xi+C_o^{-1}B_o\left(\Delta\vec{r}\right)_\eta+\vec{r}_\zeta=C_o^{-1}\left\{\vec{e}-B_o\vec{r}_{o\eta}-A_o\vec{r}_{o\xi}\right\} \tag{3.37}$$

$$C_o^{-1}A_o\left(\Delta\vec{r}\right)_\xi+C_o^{-1}B_o\left(\Delta\vec{r}\right)_\eta+\vec{r}_\zeta=C_o^{-1}\left\{\begin{bmatrix}0\\0\\\Delta V+2\Delta V_o\end{bmatrix}-\begin{bmatrix}0\\0\\\Delta V_o\end{bmatrix}-\begin{bmatrix}0\\0\\\Delta V_o\end{bmatrix}\right\} \tag{3.38}$$

Finally, Equation (3.34) takes the following form.

$$C_o^{-1}A_o\left(\Delta\vec{r}\right)_\xi+C_o^{-1}B_o\left(\Delta\vec{r}\right)_\eta+\vec{r}_\zeta=C_o^{-1}\vec{g} \tag{3.39}$$

where

$$\vec{g}=\begin{bmatrix}0\\0\\\Delta V+2\Delta V_o\end{bmatrix}-\begin{bmatrix}0\\0\\\Delta V_o\end{bmatrix}-\begin{bmatrix}0\\0\\\Delta V_o\end{bmatrix}=\begin{bmatrix}0\\0\\\Delta V\end{bmatrix}$$

Since, the marching direction is the ζ direction, the following definition can be used.

$$\vec{r}=\vec{r}_o+\vec{r}_\zeta\rightarrow\vec{r}-\vec{r}_o=\Delta\vec{r}=\vec{r}_\zeta \tag{3.40}$$

If Equation (3.40) is substituted into equation (3.39), the following equation is obtained.

$$C_o^{-1}A_o\left(\vec{r}_\zeta\right)_\xi+C_o^{-1}B_o\left(\vec{r}_\zeta\right)_\eta+\vec{r}_\zeta=C_o^{-1}\vec{g} \tag{3.41}$$

In the above equation, central differences in the $\xi$ and $\eta$ directions, and backward difference in the marching direction ζ can be employed by using the following operators.

$$\delta_\xi \phi_{i,j} = \frac{1}{2}\left(\phi_{i+1,j} - \phi_{i-1,j}\right) \tag{3.42}$$

$$\delta_\eta \phi_{i,j} = \frac{1}{2}\left(\phi_{i,j+1} - \phi_{i,j-1}\right) \tag{3.43}$$

$$\nabla_\zeta = \left(\phi_k - \phi_{k-1}\right) \tag{3.44}$$

where $\delta_\xi$, $\delta_\eta$ and $\nabla_\zeta$ defines the central difference in $\xi$, central difference in $\eta$ and backward difference in $\zeta$ direction for an arbitrary variable $\phi$, respectively.

In this case, Equation (3.41) can be written in following form,

$$C_k^{-1} A_k \delta_\xi \left(\vec{r}_{k+1} - \vec{r}_k\right) + C_k^{-1} B_k \delta_\eta \left(\vec{r}_{k+1} - \vec{r}_k\right) + I\left(\vec{r}_{k+1} - \vec{r}_k\right) = C_k^{-1} \vec{g}_{k+1} \tag{3.45}$$

where the subscripts *k* and *k+1* denote the known and unknown levels in the $\zeta$ direction. To reduce the inversion cost, the difference equations are approximately factorized as in [14]

$$\left(I + C_k^{-1} B_k \delta_\eta\right)\left(I + C_k^{-1} A_k \delta_\xi\right)\left(\vec{r}_{k+1} - \vec{r}_k\right) = C_k^{-1} \vec{g}_{k+1} \tag{3.46}$$

so that $\vec{r}_{k+1}$ is obtained by solving sequences of one-dimensional like block tridiagonal systems,

$$\left(I + C_k^{-1} B_k \delta_\eta\right)\vec{d}_{k+1} = C_k^{-1} \vec{g}_{k+1} \tag{3.47}$$

$$\left(I + C_k^{-1} A_k \delta_\xi\right)\nabla_\zeta \vec{r}_{k+1} = \vec{d}_{k+1} \tag{3.48}$$

$$\vec{r}_{k+1} = \vec{r}_k + \nabla_\zeta \vec{r}_{k+1} \tag{3.49}$$

The coefficient matrices A, B, C contain derivatives in $\xi$, $\eta$ and $\zeta$. The derivatives in $\xi$ and $\eta$ directions are obtained by central differencing, while the derivatives in $\zeta$ are obtained from the following formulation,

$$\begin{bmatrix} x_\zeta \\ y_\zeta \\ z_\zeta \end{bmatrix} = \frac{\Delta V}{\det C} \begin{bmatrix} y_\xi z_\eta - y_\eta z_\xi \\ z_\xi x_\eta - z_\eta x_\xi \\ x_\xi y_\eta - x_\eta y_\xi \end{bmatrix} \tag{3.50}$$

### 3.3 Cell Volume Specification for Two Dimensional Grids

J. F. Thompon, Z. U. Warsi and C. W. Mastin suggest employing concentric circles to determine $F(\xi, \eta)$ distribution. According to this procedure, a circle whose perimeter is equal to perimeter of inner boundary of the physical domain is defined. Then, by using an algebraic function, a set of concentric circles at various radii is specified. The same grid point distribution is applied to the inner circle (whose perimeter is that of the inner boundary) and the other grid points can be determined by rays emanating from the origin and passing from the grid points of inner circle as shown in Figure 3.1. After the grid point distribution is determined, inverse of Jacobian of transformation, i.e. cell area function F, can be computed for this grid point distribution. The manner in which concentric circles are specified is used for grid line clustering.

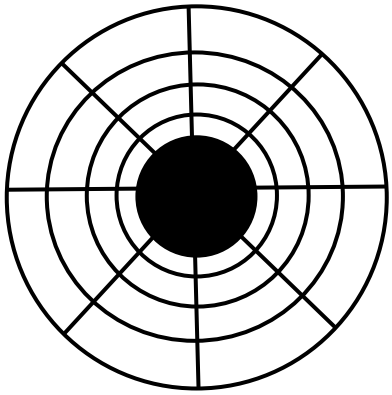

**Figure 3.1** Sample algebraic grid by concentric circles

A second approach to determine $F$ is discussed in reference [16]. In this procedure, the length of the inner boundary is drawn as a straight line with the same grid point distribution. Then parallel grid lines (constant $\eta$) are created to produce a nonuniform grid spacing in a rectangular domain. After this procedure, the Jacobian is calculated to provide the cell area function. This procedure is illustrated graphically in figure 3.2 and 3.3.

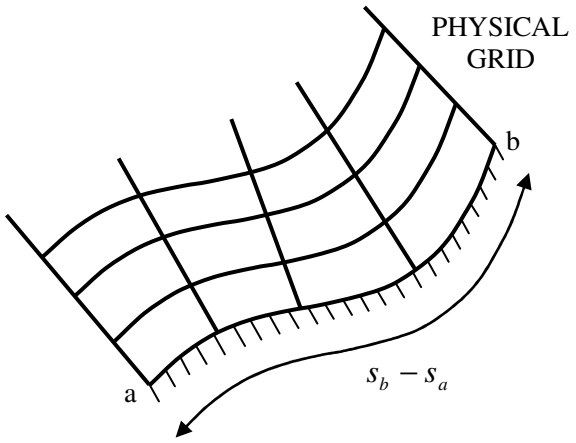


**Figure 3.2** Physical grid created on boundary curve *ab*

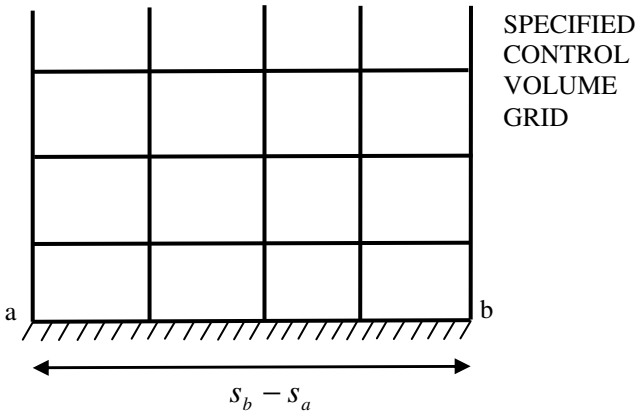


**Figure 3.3** Sample algebraic grid by parallel grid lines

One simple and popular way to specify the cell volume, i.e. *F* distribution, is suggested in reference [14] and in thesis this procedure was used. The procedure is explained below.

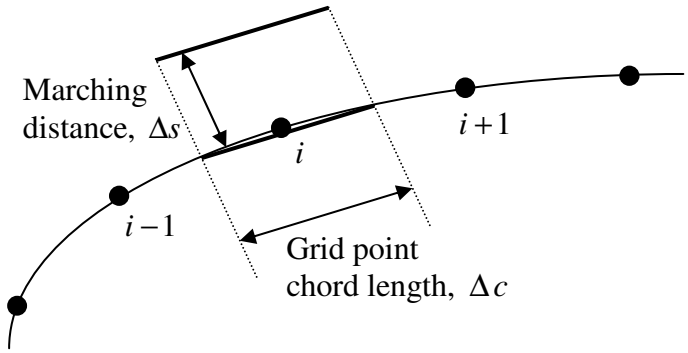


**Figure 3.4** Topology for the cell area specification

$$\Delta V_{i,j} = \Delta c_{i,j} \Delta s_{i,j} \tag{3.51}$$

where $\Delta c_{i,j}$ defines the grid point chord length and $\Delta s_{i,j}$ is the marching distance. The grid point chord length $\Delta c_{i,j}$ is given by

$$\Delta c_{i,j} = \frac{1}{2}\left(\left|\vec{r}_i - \vec{r}_{i-1}\right| + \left|\vec{r}_{i+1} - \vec{r}_i\right|\right) \tag{3.52}$$

Marching distance $\Delta s_{i,j}$ can be taken as constant or determined by using different functions which match the required criteria for generated grids [14].
If exponential function is used, $\Delta s_{i,j}$ can be defined by the following formulation.

$$\Delta s_{i,j} = \Delta s_1 (1+\varepsilon)^{j-1} \tag{3.53}$$

where $\Delta s_1$ is the initial grid spacing in the marching direction and $\varepsilon$ is the ratio of successive spacing in the marching direction. These two parameters are input given by the user.

### 3.4 Cell Volume Specification for Three Dimensional Grids

In order to calculate cell volumes, the method used for the calculation of cell area in two dimensional hyperbolic grid generation is extended to three dimensional hyperbolic grid generation.

In this method, the specified volume at each point is set equal to the computed surface area element times a user specified arc-length [14].

Specifically,

$$\Delta V_{i,j,k+1} = \Delta s_{i,j,k} \Delta S_{i,j,k} \tag{3.54}$$

where $\Delta s$ is the user specified arc-length and $\Delta S$ is the surface area. $\Delta S$ can be found as follows.

$$\Delta S_{i,j,k} = \frac{1}{4}\left[\left|\vec{r}_{i,j,k} - \vec{r}_{i-1,j,k}\right| + \left|\vec{r}_{i+1,j,k} - \vec{r}_{i,j,k}\right|\right] \cdot \left[\left|\vec{r}_{i,j,k} - \vec{r}_{i,j-1,k}\right| + \left|\vec{r}_{i,j+1,k} - \vec{r}_{i,j,k}\right|\right] \tag{3.55}$$

$\Delta s$ can be a constant or can be an exponential function used for clustering in the marching direction.

### 3.5 Boundary Conditions and Implementation for Two Dimensional Grids

In this part, boundary conditions for two dimensional hyperbolic C-type grid and O-type grids are considered.

### 3.5.1 C- Type Grids

At boundaries where $i = 1$ and $i = im$ , $x$ is held constant and $y$ is floated using the equations as follows,

$$x_{1,j+1} = x_{1,j} \tag{3.56}$$

$$y_{1,j+1} = y_{1,j} - \Delta s(j) \tag{3.57}$$

$$x_{im,j+1} = x_{im,j} \tag{3.58}$$

$$y_{im,j+1} = y_{im,j} + \Delta s(j) \tag{3.59}$$

Physical and computational spaces for C-type mesh are shown in Figure 3.5 and figure 3.6 respectively.

### 3.5.2 O – Type Grids

At boundaries where $i = 1$ and $i = im$ , $y$ is held constant and $x$ is floated using the equations as follows,

$$x_{1,j+1} = x_{1,j} + \Delta s(j) \tag{3.60}$$

$$y_{1,j+1} = y_{1,j} \tag{3.61}$$

$$x_{im,j+1} = x_{im,j} + \Delta s(j) \tag{3.62}$$

$$y_{im,j+1} = y_{im,j} \tag{3.63}$$

Physical and computational spaces for O-type mesh are shown in Figure 3.7 and 3.8 respectively.

Recall Equation (3.13) and note that the boundary conditions affect $[DD]_2$ and $[DD]_{im-1}$ according to Equations (3.15) and (3.16). $R_1$ and $R_{im}$ matrices at all $j$ levels can be calculated by using Equations (3.56) through (3.63). After the

calculation of $R_1$ and $R_{im}$ matrices, $[DD]_2$ and $[DD]_{im-1}$ matrices are updated as given by Equations (3.15) and (3.16).

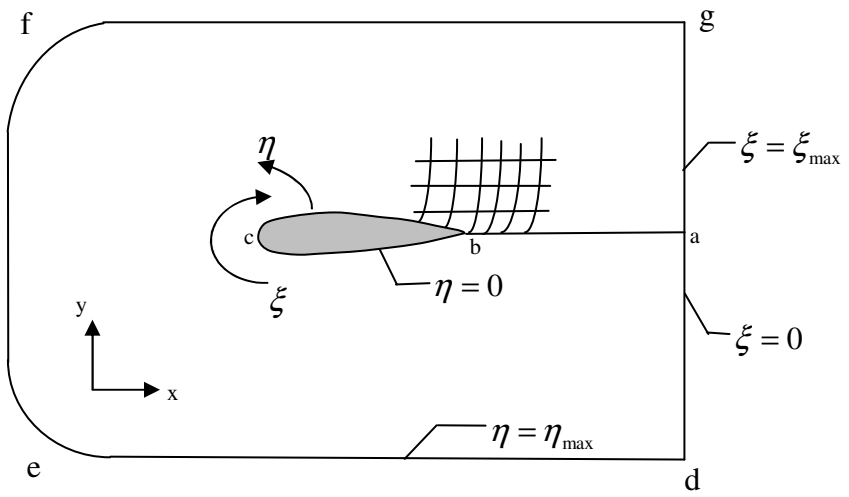


**Figure 3.5** Physical space for C-type mesh

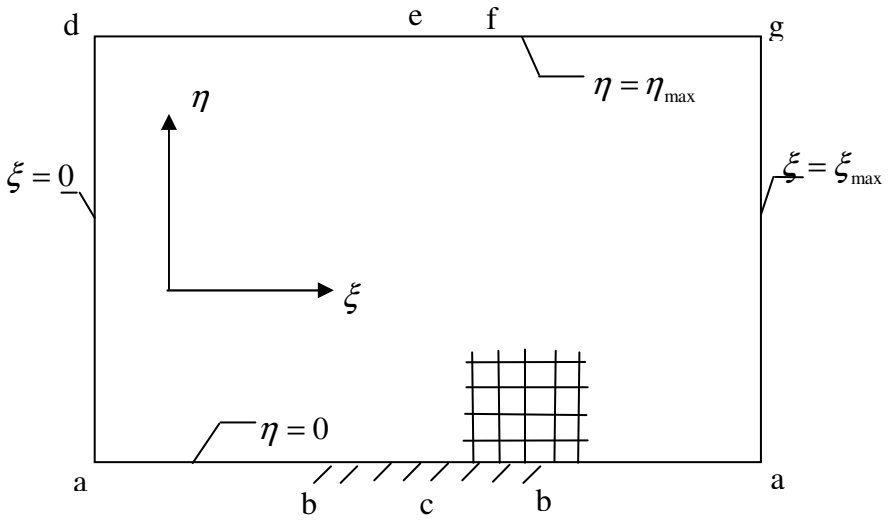


**Figure 3.6** Computational space for C-type mesh

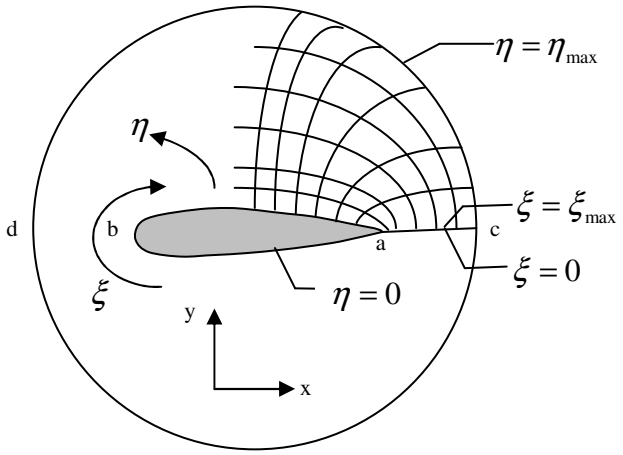


**Figure 3.7** Physical space for O-type mesh

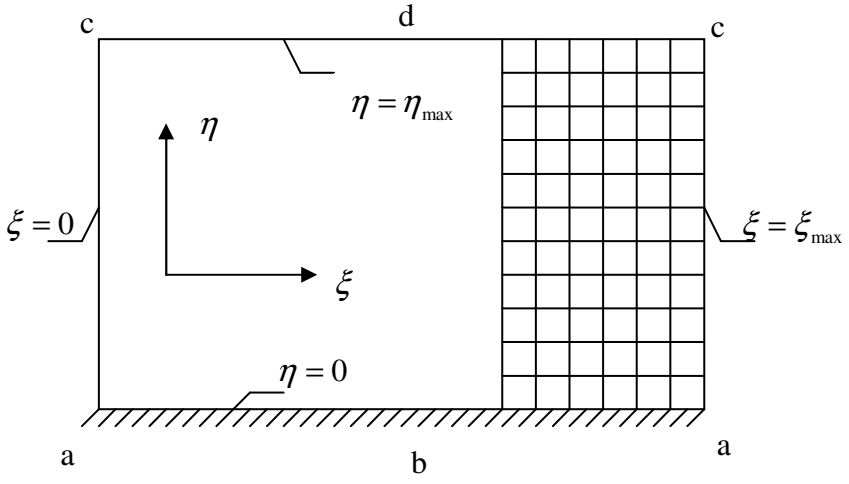


**Figure 3.8** Computational space for O-type mesh

## 3.6 Boundary Conditions and Implementation for Three Dimensional Grids

In literature, it is seen that five types of implicit boundary conditions can be implemented to the grid generation code at the $\xi$ and $\eta$ boundaries (except for the axis condition, which is only implemented in $\xi$). These five types of implicit boundary conditions are discussed in Reference [15]. Throughout this section, coordinate increments $\Delta\vec{r} = \vec{r}_{k+1} - \vec{r}_k$ are represented by $(\Delta x, \Delta y, \Delta z)^t$.

### 3.6.1 Periodicity Boundary Condition

All derivatives at the end points in the periodic direction are evaluated by "wrapping around". A periodic block tridiagonal solver should be used in order to solve the system.

### 3.6.2 Constant Cartesian Plane Boundary Condition

If a $\xi$ or $\eta$ boundary is restricted to an $x$=constant, $y$=constant or $z$=constant plane, then this value is enforced and the other variables are "floated". For example, for $x$=constant plane at $i$=1 boundary, $x$ is held constant and $y$ and $z$ are floated using the following condition.

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}_{i=1} = \begin{bmatrix} 0 \\ \Delta y \\ \Delta z \end{bmatrix}_{i=2} \tag{3.64}$$

### 3.6.3 Symmetry Plane Boundary Condition

Conventional reflection planes are used to impose symmetry about any x=0, y=0 and z=0 plane and values are updated implicitly. For example, to update a reflected plane at i=1 for a symmetry condition about x=0 corresponding to i=2, x reflects odd and y and z reflect even as,

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}_{i=1} = \begin{bmatrix} -\Delta x \\ \Delta y \\ \Delta z \end{bmatrix}_{i=3} \tag{3.65}$$

This condition is illustrated in Figure 3.9.

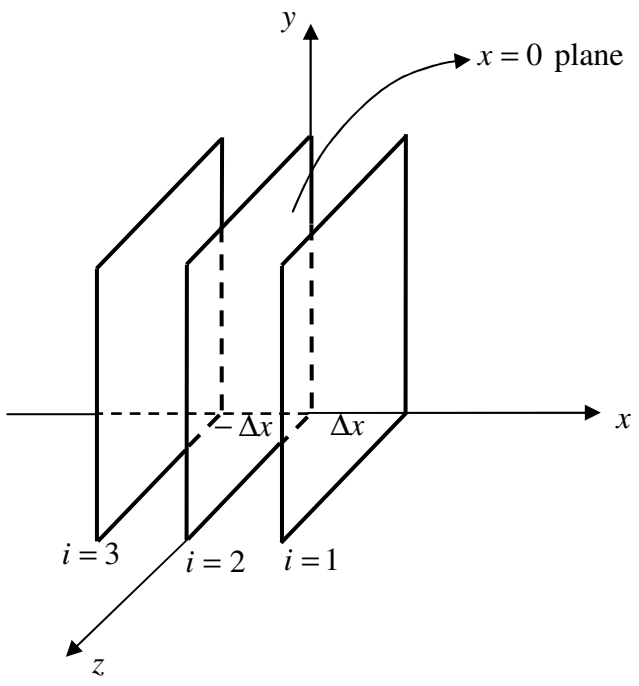


**Figure 3.9** Illustration of symmetry plane boundary condition

### 3.6.4 Floating Edge Boundary Condition

An entire ξ or η boundary can be floated using the simple hyperbolic equation $\vec{r}_{\xi\zeta} = 0$ or $\vec{r}_{\eta\zeta} = 0$ to update a boundary plane. This is essentially a zeroth order extrapolation of $\Delta\vec{r}$ from the adjacent interior value.

As shown in Figure 3.10, there is not any change for $\vec{r}_{\xi}$ and $\vec{r}_{\eta}$ in $\zeta$ direction, i.e., $\vec{r}_{\xi\zeta} = 0$ and $\vec{r}_{\eta\zeta} = 0$. So, these two simple hyperbolic equations provide the orthogonality between $\xi$ and $\zeta$ directions and between $\eta$ and $\zeta$ directions.

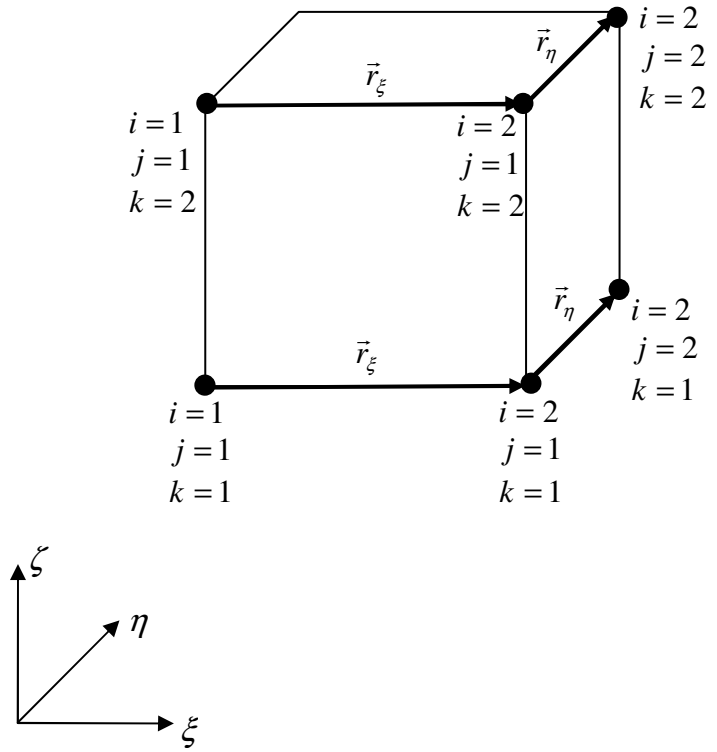


**Figure 3.10** Sketch illustrating $\vec{r}_{\xi}$ and $\vec{r}_{\eta}$ vectors

### 3.6.5 Axis Boundary Condition

The axis point can not be included in a normal calculation, because at the axis point a whole line contracts to one point which makes the determinant of $B^{-1}A$ zero. This makes the governing equation not well posed. Therefore, these points must be user specified and can not be obtained by calculation for the initial surface. [18]

As stated at the beginning of this section, the information about these five types of boundary conditions is taken from Reference [15] and more detailed information can be found in Reference [15].

### 3.6.6 Implementation of Boundary Conditions

In this thesis, a floating type boundary condition is used and it is applied in both $\xi$ and $\eta$ sweeps.

#### 3.6.6.1 $\eta$ sweep implementation

During the solution procedure, the terms at *j*=1 should be implemented to form $\eta$ sweep matrix containing the unknown interior elements.

If Equation (3.47) is written at *j*=2 station, the discretized equation is obtained as follows.

$$\frac{-1}{2}\cdot\left(C^{-1}B\right)_{i,2,k}\vec{d}_{i,1,k+1}+I\cdot\vec{d}_{i,2,k+1}+\frac{1}{2}\cdot\left(C^{-1}B\right)_{i,2,k}\vec{d}_{i,3,k+1}=C^{-1}_{i,2,k}\vec{g}_{i,2,k+1} \tag{3.66}$$

So, zeroth order mixed derivative for vector $d$ with respect to $\eta$ and $\zeta$ at the boundary should be calculated and implemented to the system. Calculation and implementation are given in detail as follows.

$$\frac{\partial^2 \vec{d}}{\partial \zeta \partial \eta} = \frac{\partial}{\partial \zeta}\left(\frac{\partial \vec{d}}{\partial \eta}\right) = 0 \tag{3.67}$$

If $\frac{\partial \vec{d}}{\partial \eta}$ is expanded by using second order forward difference at $j$=1 station, the following equation is obtained.

$$\frac{\partial\left[\left(-3\vec{d}_{j=1} + 4\vec{d}_{j=2} - \vec{d}_{j=3}\right)/2\right]}{\partial \zeta} = 0 \tag{3.68}$$

Then, expanding the above equation by using the first order backward difference for $\zeta$ derivative, the following equation is obtained.

$$\frac{-3\vec{d}_{j=1,k+1} + 4\vec{d}_{j=2,k+1} - \vec{d}_{j=3,k+1}}{2} - \frac{-3\vec{d}_{j=1,k} + 4\vec{d}_{j=2,k} - \vec{d}_{j=3,k}}{2} = 0 \tag{3.69}$$

Rearranging the above equation, it is possible to obtain

$$\vec{d}_{j=1,k+1} = \frac{1}{3}\left(4\vec{d}_{j=2,k+1} - \vec{d}_{j=3,k+1} + 3\vec{d}_{j=1,k} - 4\vec{d}_{j=2,k} + \vec{d}_{j=3,k}\right) \tag{3.70}$$

If Equation (3.70) is substituted into Equation (3.66), the following equation is obtained as,

$$\left(I - \frac{2}{3}\left(C^{-1}B\right)_{i,2,k}\right)\vec{d}_{i,2,k+1} + \frac{2}{3}\left(C^{-1}B\right)_{i,2,k}\vec{d}_{i,3,k+1} =$$

$$C^{-1}_{i,2,k}\vec{g}_{i,2,k+1} + \frac{1}{6}\left(C^{-1}B\right)_{i,2,k}\left(3\vec{d}_{i,1,k} - 4\vec{d}_{i,2,k} + \vec{d}_{i,3,k}\right) \tag{3.71}$$

For the boundary $j$=$j$m, the same procedure of calculation and implementation is applied. Again, zeroth order mixed derivative for vector $d$ with respect to $\eta$ and $\zeta$ at the boundary $j$=jm should be calculated and implemented to the system. Noting that

$$\frac{\partial^2 \vec{d}}{\partial \zeta \partial \eta} = \frac{\partial}{\partial \zeta}\left(\frac{\partial \vec{d}}{\partial \eta}\right) = 0 \tag{3.72}$$

If $\frac{\partial \vec{d}}{\partial \zeta}$ is expanded by using second order backward difference at *j*=*j*m station, the following relation is obtained.

$$\frac{\partial\left[\left(3\vec{d}_{j=jm} - 4\vec{d}_{j=jm1} + \vec{d}_{j=jm2}\right)/2\right]}{\partial \zeta} = 0 \tag{3.73}$$

Then, expanding the above equation by using first order backward difference for $\zeta$ derivative, the following equation is obtained.

$$\frac{3\vec{d}_{j=jm,k+1} - 4\vec{d}_{jm1,k+1} + \vec{d}_{jm2,k+1}}{2} - \frac{3\vec{d}_{j=jm,k} - 4\vec{d}_{jm1,k} + \vec{d}_{jm2,k}}{2} = 0 \tag{3.74}$$

Rearranging the above equation it is possible to obtain

$$\vec{d}_{j=jm,k+1} = \frac{1}{3}\left(4\vec{d}_{j=jm1,k+1} - \vec{d}_{j=jm2,k+1} + 3\vec{d}_{j=jm,k} - 4\vec{d}_{j=jm1,k} + \vec{d}_{j=jm2,k}\right) \tag{3.75}$$

If this equation is substituted into Equation (3.47) at *j*=*j*m-1 station, following equation is obtained.

$$-\frac{2}{3}\left(C^{-1}B\right)_{i,jm-1,k}\vec{d}_{i,jm-2,k+1} + \left(I + \frac{2}{3}\left(C^{-1}B\right)_{i,jm-1,k}\right)\vec{d}_{i,jm-1,k+1} =$$

$$C^{-1}_{i,jm-1,k}\vec{g}_{i,jm-1,k+1} - \frac{1}{6}\left(C^{-1}B\right)_{i,jm-1,k}\left(3\vec{d}_{i,jm,k} - 4\vec{d}_{i,jm-1,k} + \vec{d}_{i,jm-2,k}\right) \tag{3.76}$$

### 3.6.6.2 $\xi$ sweep implementation

During the solution procedure, the terms at *i*=1 should be implemented to form $\xi$ sweep matrix containing the unknown interior elements.

If Equation (3.48) is written at *i*=2 station, the discretized equation can be obtained as follows.

$$-\frac{1}{2}\left(C^{-1}A\right)_{2,j,k}\left(\nabla_{\zeta}\vec{r}\right)_{1,j,k+1}+I\left(\nabla_{\zeta}\vec{r}\right)_{2,j,k+1}+\frac{1}{2}\left(C^{-1}A\right)_{2,j,k}\left(\nabla_{\zeta}\vec{r}\right)_{3,j,k+1}=\vec{d}_{2,j,k+1} \tag{3.77}$$

Therefore, zeroth order mixed derivative for vector $\left(\nabla_{\zeta}\vec{r}\right)$ with respect to $\xi$ and $\zeta$ at the boundary should be calculated and implemented to the system. Noting that

$$\frac{\partial^2\left(\nabla_{\zeta}\vec{r}\right)}{\partial\zeta\partial\xi}=\frac{\partial}{\partial\zeta}\left(\frac{\partial\left(\nabla_{\zeta}\vec{r}\right)}{\partial\xi}\right)=0 \tag{3.78}$$

If $\dfrac{\partial\left(\nabla_{\zeta}\vec{r}\right)}{\partial\xi}$ is expanded by using second order forward difference at $i$ =1 station, the following relation is obtained.

$$\frac{\partial\left[\left(-3\left(\nabla_{\zeta}\vec{r}\right)_{i=1}+4\left(\nabla_{\zeta}\vec{r}\right)_{i=2}-\left(\nabla_{\zeta}\vec{r}\right)_{i=3}\right)/2\right]}{\partial\zeta}=0 \tag{3.79}$$

Then, expanding the above equation by using first order backward difference for $\zeta$ derivative, the following equation is obtained.

$$\frac{-3\left(\nabla_{\zeta}\vec{r}\right)_{i=1,k+1}+4\left(\nabla_{\zeta}\vec{r}\right)_{i=2,k+1}-\left(\nabla_{\zeta}\vec{r}\right)_{i=3,k+1}}{2}+\frac{3\left(\nabla_{\zeta}\vec{r}\right)_{i=1,k}-4\left(\nabla_{\zeta}\vec{r}\right)_{i=2,k}+\left(\nabla_{\zeta}\vec{r}\right)_{i=3,k}}{2}=0 \tag{3.80}$$

Rearranging the above equation, it is possible to obtain

$$\left(\nabla_{\zeta}\vec{r}\right)_{i=1,k+1}=\frac{1}{3}\left(4\left(\nabla_{\zeta}\vec{r}\right)_{i=2,k+1}-\left(\nabla_{\zeta}\vec{r}\right)_{i=3,k+1}+3\left(\nabla_{\zeta}\vec{r}\right)_{i=1,k}-4\left(\nabla_{\zeta}\vec{r}\right)_{i=2,k}+\left(\nabla_{\zeta}\vec{r}\right)_{i=3,k}\right) \tag{3.81}$$

If this equation is substituted into Equation (3.77), following equation can be obtained as,

$$\left(I-\frac{2}{3}\left(C^{-1}A\right)_{2,j,k}\right)\left(\nabla_\zeta \vec{r}\right)_{2,j,k+1}+\frac{2}{3}\left(C^{-1}A\right)_{2,j,k}\left(\nabla_\zeta \vec{r}\right)_{3,j,k+1}=$$

$$\vec{d}_{2,j,k+1}+\frac{1}{6}\left(C^{-1}A\right)_{2,j,k}\left(3\left(\nabla_\zeta \vec{r}\right)_{i=1,k}-4\left(\nabla_\zeta \vec{r}\right)_{i=2,k}+\left(\nabla_\zeta \vec{r}\right)_{i=3,k}\right) \tag{3.82}$$

For the boundary at *i=im*, the same procedure of calculation and implementation can be applied. Again, zeroth order mixed derivative for vector $\left(\nabla_\zeta \vec{r}\right)$ with respect to $\xi$ and $\zeta$ at the boundary *i=im* should be calculated and implemented to the system. Noting that

$$\frac{\partial^2\left(\nabla_\zeta \vec{r}\right)}{\partial\zeta\partial\xi}=\frac{\partial}{\partial\zeta}\left(\frac{\partial\left(\nabla_\zeta \vec{r}\right)}{\partial\xi}\right)=0 \tag{3.83}$$

If $\frac{\partial\left(\nabla_\zeta \vec{r}\right)}{\partial\xi}$ is expanded by using second order backward difference at *i=im* station, the following equation is obtained.

$$\frac{\partial\left[\left(3\left(\nabla_\zeta \vec{r}\right)_{im}-4\left(\nabla_\zeta \vec{r}\right)_{im1}+\left(\nabla_\zeta \vec{r}\right)_{im2}\right)/2\right]}{\partial\zeta}=0 \tag{3.84}$$

Then, expanding the above equation by using first order backward difference for $\zeta$ derivative, the following relation is obtained.

$$\frac{3\left(\nabla_\zeta \vec{r}\right)_{im,k+1}-4\left(\nabla_\zeta \vec{r}\right)_{im1,k+1}+\left(\nabla_\zeta \vec{r}\right)_{im2,k+1}}{2}-\frac{3\left(\nabla_\zeta \vec{r}\right)_{im,k}-4\left(\nabla_\zeta \vec{r}\right)_{im1,k}+\left(\nabla_\zeta \vec{r}\right)_{im2,k}}{2}$$

$$=0 \tag{3.85}$$

Rearranging the above equation, it is possible to obtain

$$\left(\nabla_\zeta \vec{r}\right)_{im,k+1}=\frac{1}{3}\left(4\left(\nabla_\zeta \vec{r}\right)_{i=im1,k+1}-\left(\nabla_\zeta \vec{r}\right)_{i=im2,k+1}+3\left(\nabla_\zeta \vec{r}\right)_{i=im,k}-4\left(\nabla_\zeta \vec{r}\right)_{i=im1,k}\right)$$

$$+\frac{1}{3}\left(\nabla_\zeta \vec{r}\right)_{i=im2,k} \tag{3.86}$$

If Equation (3.86) is substituted into Equation (3.48) at *i=im*-1 station, following equation is obtained as,

$$-\frac{2}{3}\left(C^{-1}A\right)_{im1,j,k}\left(\nabla_\zeta \vec{r}\right)_{im2,j,k+1}+\left(I+\frac{2}{3}\left(C^{-1}A\right)_{im-1,j,k}\right)\left(\nabla_\zeta \vec{r}\right)_{im1,j,k+1}=$$

$$\vec{d}_{im1,j,k+1}-\frac{1}{6}\left(C^{-1}A\right)_{im1,j,k}\left(3\left(\nabla_\zeta \vec{r}\right)_{im,k}-4\left(\nabla_\zeta \vec{r}\right)_{im1,k}+\left(\nabla_\zeta \vec{r}\right)_{im2,k}\right) \qquad (3.87)$$

As explained before, after calculating $\left(\nabla_\zeta \vec{r}\right)$ terms, grid points can be found by using Equation (3.49).

# CHAPTER 4

# SMOOTHING AND CLUSTERING

## 4.1 Smoothing for Two Dimensional Grid Generation

In hyperbolic grid generation technique, the derivatives in $\xi$ and $\eta$ directions are calculated by numerical approximations. So, artificial dissipation terms must be added in order to control oscillations due to odd-even uncoupling of grid points [1]

The form and magnitude of the added dissipation are very important for grid quality. Artificial dissipation terms (damping function) prevent propagation of slope discontinuities from the initial data onto the domain. However, these terms cause deformation of orthogonality. So, there is a trade-off between the grid smoothness and deformation of orthogonality because of the addition of artificial numerical dissipation terms.

In this study, in order to get rid of propagation of discontinuities, the right hand side of the Equation (3.13) is modified with fourth order artificial dissipation terms.

$$[AA]R_{i-1,j} + [BB]R_{i,j} + [CC]R_{i+1,j} = [DD]_{i,j} + dissipation \tag{4.1}$$

where

$$R_{i,j} = \begin{bmatrix} x_{i,j} \\ y_{i,j} \end{bmatrix}$$

$$
\begin{aligned}
dissipation &= \beta(\Delta\nabla)_{\xi}{}^{2} R_{i,j} \\
&= \beta(\Delta\nabla)_{\xi}[(\Delta\nabla)_{\xi} R_{i,j}] \\
&= \beta(\Delta\nabla)_{\xi}[R_{i+1,j} - 2R_{i,j} + R_{i-1,j}] \\
&= \beta[(R_{i+2,j} - 2R_{i+1,j} + R_{i,j}) - 2(R_{i+1,j} - 2R_{i,j} + R_{i-1,j}) + (R_{i,j} - 2R_{i-1,j} + R_{i-2,j})] \\
&= \beta(R_{i+2,j} - 4R_{i+1,j} + 6R_{i,j} - 4R_{i-1,j} + R_{i-2,j}) \qquad (4.2)
\end{aligned}
$$

For stability, the maximum value of $\beta$ can be taken as 0.125.

So, the right hand side of equation (4.1) can be rearranged as follows,

$$[AA]R_{i-1,j} + [BB]R_{i,j} + [CC]R_{i+1,j} = [DD]_{i,j} \qquad (4.3)$$

where

$$[DD]_{i,j} = \begin{bmatrix} \{[DD]_{i,j}\}_{1,1} + \beta(x_{i+2,j} - 4x_{i+1,j} + 6x_{i,j} - 4x_{i-1,j} + x_{i-2,j}) \\ \{[DD]_{i,j}\}_{2,1} + \beta(y_{i+2,j} - 4y_{i+1,j} + 6y_{i,j} - 4y_{i-1,j} + y_{i-2,j)} \end{bmatrix}$$

and there is no change for the other coefficient matrices.

Such numerical methods, though elegant are complex and time consuming. Simple algebraic relations based on analytical geometry can also be used for ensuring the orthogonality and providing required cell area. [17]

In this approach, the area of cell formed over the edge ab on $\Gamma_j$ is obtained by the following formula.

$$\Delta A_{abed} = \left[l_{ab}^{j} cf + l_{ab}^{1}(1 - cf)\right]\Delta s(j) \qquad (4.4)$$

where *cf* is cell size control factor, $\Delta s_j$ is the cell height or marching distance and *l* is the distance between two successive grid points along the profile. Since this is a marching procedure, $l_{ab}^{1}$ denotes the distance between grid points a and b, $l_{ab}^{2}$ denotes the distance between grid points d and e. The superscript

denotes the level in the marching direction. See Figure 4.1. A suitable value for cf (between 0 and 1) is selected to get an appropriate cell size distribution within the domain. Same technique is used in Reference [11]. The difference appears only in selecting the cell size control factor. Cell size control factor is determined by a function in [11], while it is taken as constant in [17]. In this study, it is taken as constant and given by the user.

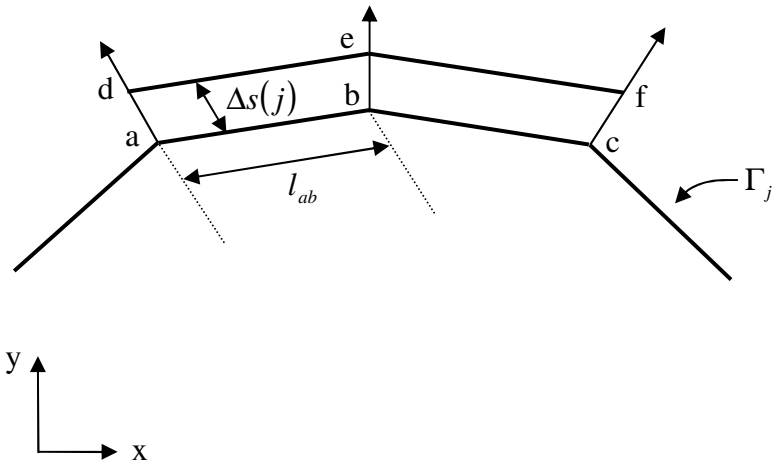


**Figure 4.1** Schematic diagram illustrating the modified cell area calculation

In this thesis, both of the above approaches, i.e. adding fourth order artificial dissipation terms and approach given in [17], were used to obtain a good grid quality. If these approaches are not used, a mesh like figure 4.2 may be obtained. At this point, it is useful to note that these approaches should be used when necessary.

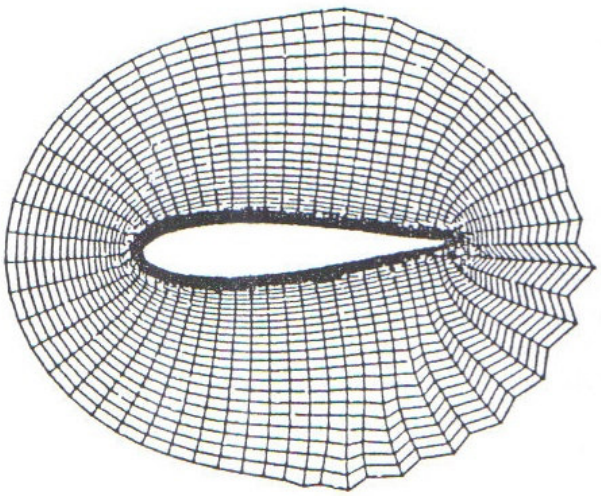

**Figure 4.2** A mesh showing the propagation of a discontinuity generated from a hyperbolic system of equations

Moreover, in many physical domains convex corners are usually met and in order to provide extra robustness at convex corners, switching from solving the hyperbolic grid generation equation to some other equations at the convex corner point is required [15].

The exact location of the grid point in the next marching step out from a convex corner is predicted in advance. The predicted point is located by marching the grid a distance of Δs. The marching distance is scaled as

$$\Delta s = (\Delta s)\sin(\alpha_{i,j}) \tag{4.5}$$

where $\alpha_{i,j}$ is the half angle formed by the corner.

The scaling with size of the half angle causes the corner point to march out a smaller distance than its neighbors, thus helping to bend the neighboring grid lines towards the corner [15].

Since the aim is making the corner point to march out a smaller distance than its neighbors, this approach can be used more than once to bend the neighboring grid lines towards the corner.

$$\Delta s = (\Delta s)\left[\sin(\alpha_{i,j})\right]^n \tag{4.6}$$

where $n$ is the power and specified by the user arbitrarily.

## 4.2 Smoothing for Three Dimensional Grid Generation

In practice, numerical dissipation terms are added in $\xi$ and $\eta$ directions since all $\xi$ and $\eta$ derivatives are calculated by central differencing.

Typically, a combination of fourth and second order differences, which are explicitly and implicitly included into the basic algorithm, is used. However, in this study only second order differences are used for simplicity.

The formulation with these dissipation terms is given below.

$$\left[I + C_k^{-1} B_k \delta_\eta - \varepsilon_{i\eta} (\Delta\nabla)_\eta\right] \cdot \left[I + C_k^{-1} A_k \delta_\xi - \varepsilon_{i\xi} (\Delta\nabla)_\xi\right] \cdot (\vec{r}_{k+1} - \vec{r}_k) =$$
$$C_k^{-1} \vec{g}_{k+1} - \left[\varepsilon_{e\xi} (\Delta\nabla_\xi) + \varepsilon_{e\eta} (\Delta\nabla)_\eta\right] \cdot \vec{r}_k \tag{4.7}$$

where

$$(\Delta\nabla)_\eta \vec{r} = \vec{r}_{j+1} - 2\vec{r}_j + \vec{r}_{j-1} \tag{4.8}$$

$$(\Delta\nabla)_\xi \vec{r} = \vec{r}_{i+1} - 2\vec{r}_i + \vec{r}_{i-1} \tag{4.9}$$

In this study, $\varepsilon_{i\eta} = \varepsilon_{e\eta}$ and $\varepsilon_{i\xi} = \varepsilon_{e\xi}$ are applied for simplicity. Although constant explicit and implicit smoothing parameters are used, they can be scaled using $\varepsilon_{e\xi} = 0.5\left\|C^{-1}A\right\|$ , $\varepsilon_{e\eta} = 0.5\left\|C^{-1}B\right\|$ and $\varepsilon_i = 3\varepsilon_e$. For simplicity, matrix norms $\left\|C^{-1}A\right\|$ and $\left\|C^{-1}B\right\|$ have been approximated by the square root of the sum of the squared elements.

### 4.3 Clustering in $\eta$ Direction

Many functions can be used in order to control grid. Most common ones used for this purpose are sinusoidal and exponential functions. In many applications, grid control includes the surface clustering and the interior domain. In general, most of the important actions take place along the body surface. Thus, surface clustering is very important to investigate these actions.

One of the means of controlling the grids at the surface is to use an exponential function for the marching distance, as suggested in [18],

$$\Delta s(j) = \Delta s_1 (1+\varepsilon)^{j-1} \tag{4.10}$$

where $\Delta s(j)$ is the marching distance at $j$ levels ($\eta$ direction) and the values of $\Delta s_1$ (initial grid spacing in the marching direction) and $\varepsilon$ (the ratio of successive spacing in the marching direction ) are the required parameters and specified by the user.

For the clustering of interior grids in $\eta$ direction, exponential function is used as suggested in reference [18]. The input parameters for $\eta$ clustering are intj, js, and sj which determine interval of stretching, level to be stretched and value of stretching respectively.

In order to have a good grid control, the functions used for this purpose should be continuous. Thus, in order to get a continuously changing marching function at the interval of clustering, the ratio of successive spacing, $\varepsilon$ should be recalculated as,

$$\varepsilon = \left[ \frac{\Delta s(js - \text{int } j)}{sj} \right]^{\frac{1}{\text{int } j}} - 1 \tag{4.11}$$

where $\Delta s(js - \text{int } j)$ is the marching distance at the $(js - \text{int } j)$ level in the $\eta$ direction.

Finally, the marching function is redefined as,

$$\Delta s(j) = sj(1+\varepsilon)^{js-j} \quad \text{for} \quad js - \text{int } j \le j \le js \tag{4.12}$$

$$\Delta s(j) = sj(1+\varepsilon)^{j-js} \quad \text{for} \quad js \le j \le js + \text{int } j \tag{4.13}$$

### 4.4 Clustering in $\xi$ Direction

For $\xi$ clustering, various methods can be found in literature. The procedure of $\xi$ clustering, used in this thesis, is given below.

At the beginning, arc-length of the profile is calculated and normalized. Every three points of the profile is used and formulations for $x$ and $y$ coordinates, which are functions of arc-length, are derived as a function of arc length by using cubic spline method. If number of points of profile is not a multiple of three, this can be achieved by adding required number of points. For this process, last two points are used and arithmetic mean of coordinates of these last two points is calculated and coordinates of the new required points are found.

After the derivation of cubic spline formulas for $x$ and $y$ coordinates as a function of arc length, coordinates of any point on the profile can be calculated by giving arc-length as input. After this work, clustering procedure is applied to arc length. This procedure is given below.

The following series formulations can be introduced for this process.

$$a_n = a_1 \cdot rc^{n-1} \tag{4.14}$$

$$S_n = \frac{a_1 \cdot \left(1 - rc^n\right)}{1 - rc} = \frac{rc \cdot a_n - a_1}{rc - 1} \tag{4.15}$$

where $S_n$ denotes total arc length up to $n^{th}$ grid point in the $\xi$ direction and $rc$ denotes the clustering coefficient.

From general series formulation, the following equations can be written.

$$\Delta x_n = \Delta x_1 \cdot rc^{n-1} \tag{4.16}$$

$$S_n = \frac{\Delta x_1 \cdot \left(1 - rc^n\right)}{\left(1 - rc\right)} \tag{4.17}$$

where $\Delta x_1$ denotes the arc length between first two grid points and $\Delta x_n$ denotes the arc length between $n^{th}$ and $(n-1)^{th}$ grid points.

Rearranging Equation (4.17), one can obtain

$$rc^n - \frac{S_n}{\Delta x_1} \cdot rc + \frac{S_n}{\Delta x_1} - 1 = 0 \tag{4.18}$$

By using this equation, clustering coefficient is calculated by bisection root finding method.

If $\Delta x_1$ and $\Delta x_n$ are given as input, two clustering coefficients $rc_1$ and $rc_2$ should be calculated using the following equations and applying bisection root finding method.

$$rc_1^n - \frac{S_n}{\Delta x_1} rc_1 + \frac{S_n}{\Delta x_1} - 1 = 0 \tag{4.19}$$

$$rc_2^n - \frac{S_n}{\Delta x_n} rc_2 + \frac{S_n}{\Delta x_n} - 1 = 0 \tag{4.20}$$

Using the coefficients $rc_1$ and $rc_2$, $S_n^{(1)}$ and $S_n^{(2)}$ are found and these two $S_n$ values are blended using Hermitian polynomials. Hermitian polynomials used for this procedure are given below.

$$H_o(T) = 2T^3 - 3T^2 + 1 \tag{4.21}$$

$$H_1(T) = -2T^3 + 3T^2 \tag{4.22}$$

Finally, $S_n$ is calculated using the following formula,

$$S_n = H_o\left(\frac{n-2}{N-2}\right)\times S_n^{(1)} + H_1\left(\frac{n-2}{N-2}\right)\times S_n^{(2)} \tag{4.23}$$

where N denotes number of grid points in $\xi$ direction. This equation provides clustering of arclength and since coordinates of grid points are function of arclength, it provides clustering of grid points.

# CHAPTER 5

# APPLICATIONS AND RESULTS

## 5.1 Two Dimensional Grid Generation Applications

In this section, two-dimensional hyperbolic grid generation code is applied to different geometries in order to show robustness of the code. Also, the input parameters used for the generation of grids are given.

### 5.1.1 O-Mesh around an Ellipse

An O-mesh around an ellipse centered at the origin and having a major axis of 2 units and minor axis of 0.2 units is generated for general purpose. This profile and its enlarged view are shown in Figures 5.1 and 5.2 respectively. The input parameters are given in Table 5.1.

**Table 5.1** Input parameters for O-mesh around an ellipse

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.05 |
| Surface clustering coefficient, *scc* | 0.01 |
| Fourth order damping coefficient, *dc* | 0.02 |
| Volume scaling coefficient, *vsc* | 1.0 |
| Number of points $(\xi x \eta)$ | 201x50 |

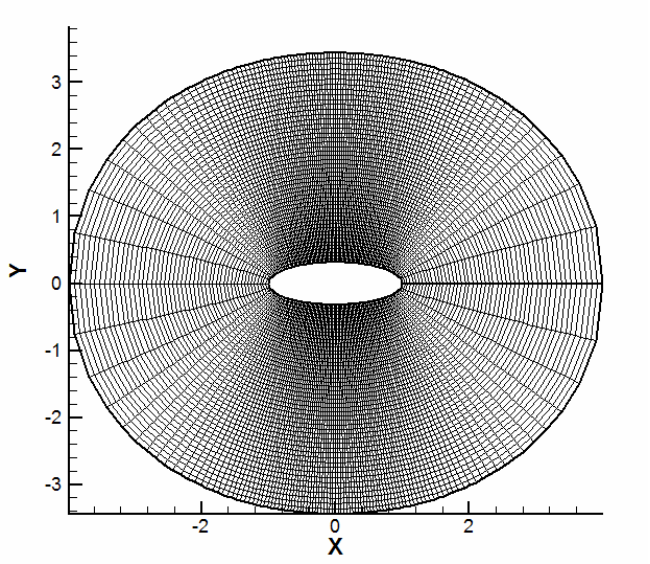


**Figure 5.1** O-mesh around an ellipse

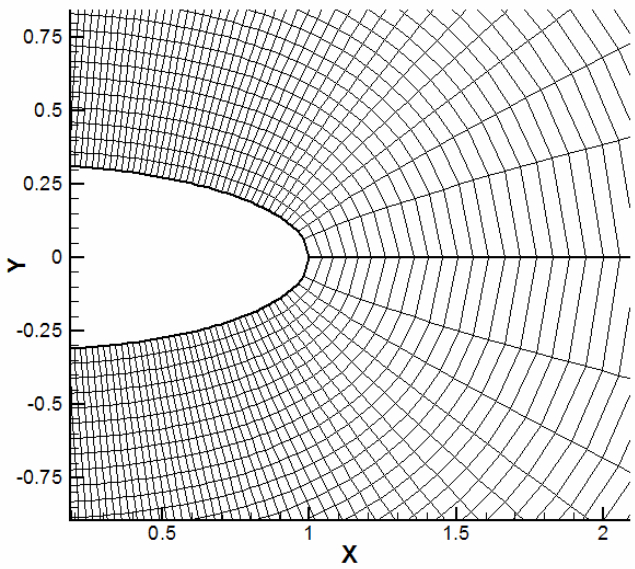


**Figure 5.2** Enlarged view of Figure 5.1

### 5.1.2 O-Mesh around a Circle

An O-mesh around a circle having a radius of 0.5 units and centered at the origin is generated for general purpose, too. This profile and enlarged view of it are given in Figures 5.3 and 5.4 respectively. The input parameters are given in table 5.2.

**Table 5.2** Input parameters for O-mesh around a circle

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.03 |
| Surface clustering coefficient, *scc* | 0.05 |
| Fourth order damping coefficient, *dc* | 0.02 |
| Volume scaling coefficient, *vsc* | 1.0 |
| Number of points $(\xi x \eta)$ | 201x50 |

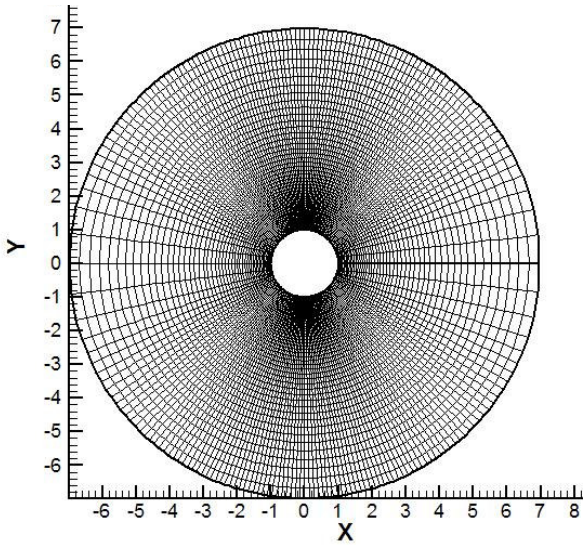


**Figure 5.3** O-mesh around a circle

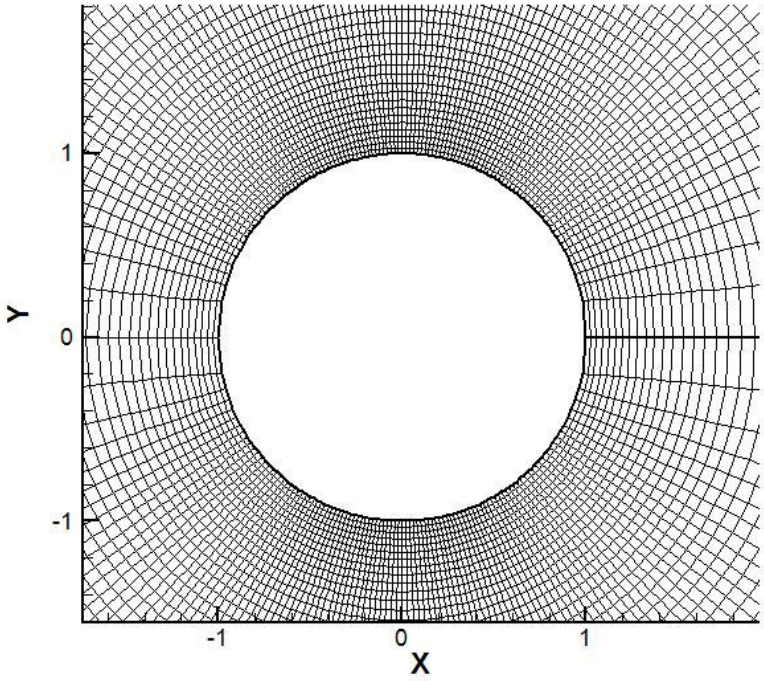


**Figure 5.4** Enlarged view of Figure 5.3

### 5.1.3 Mesh around a convex corner

Hyperbolic grid generation around a convex corner is a difficult task and a special treatment, which is explained in this study, at these points is needed. So, mesh around a $90^o$ convex corner is generated in order to show the robustness of the code. For this case, code works satisfactorily so that there is no need any special treatment in this region. Mesh around a convex corner and enlarged view of it are given in Figures 5.5 and 5.6 respectively. The input parameters are given in Table 5.3.

**Table 5.3** Input parameters for mesh around a convex corner

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.02 |
| Surface clustering coefficient, *scc* | 0.01 |
| Fourth order damping coefficient, *dc* | 0.02 |
| Volume scaling coefficient, *vsc* | 1.0 |
| Number of points $(\xi x \eta)$ | 101x40 |

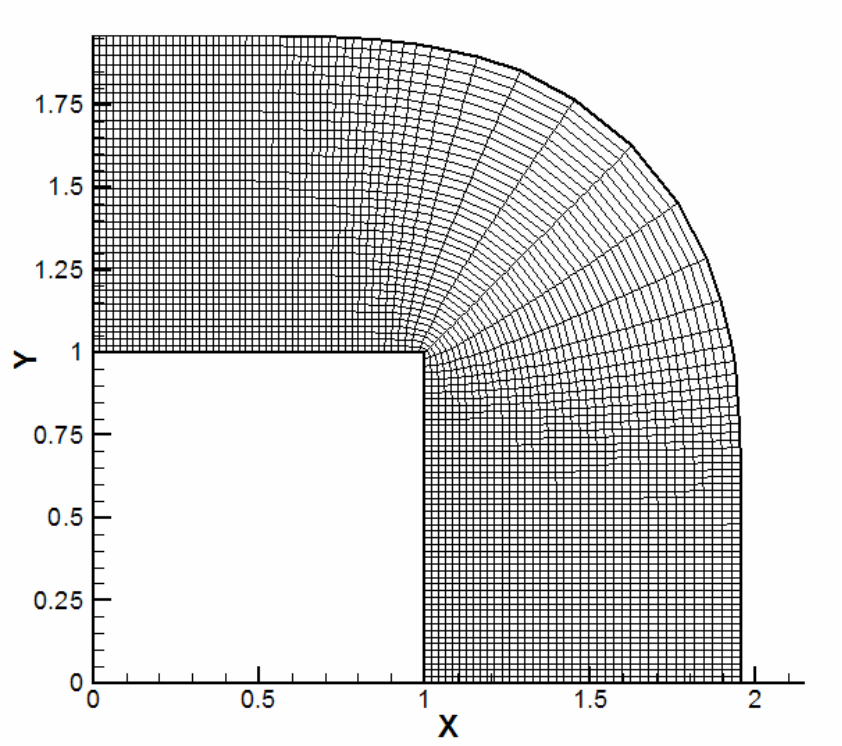


**Figure 5.5** Mesh around a convex corner

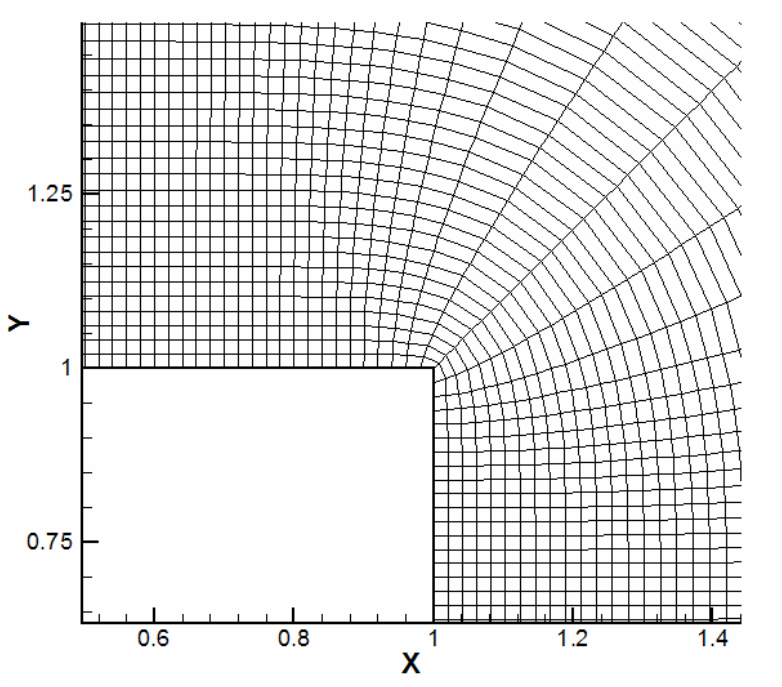


**Figure 5.6** Enlarged view of Figure 5.5

### 5.1.4 Mesh around a concave corner

Hyperbolic grid generation technique can fail for concave profiles since grid lines overlap and cell volumes become negative as the solution proceeds. So, mesh around a $90^o$ concave corner, the most critical profile in order to test robustness of code, is generated in order to show the robustness of the code. Mesh around a concave corner and enlarged view of it are given in Figures 5.7 and 5.8, respectively. The input parameters are given in Table 5.4.

**Table 5.4** Input parameters for mesh around a concave corner

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.007 |
| Surface clustering coefficient, *scc* | 0.01 |
| Fourth order damping coefficient, *dc* | 0.05 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Number of points $(\xi x \eta)$ | 51x30 |

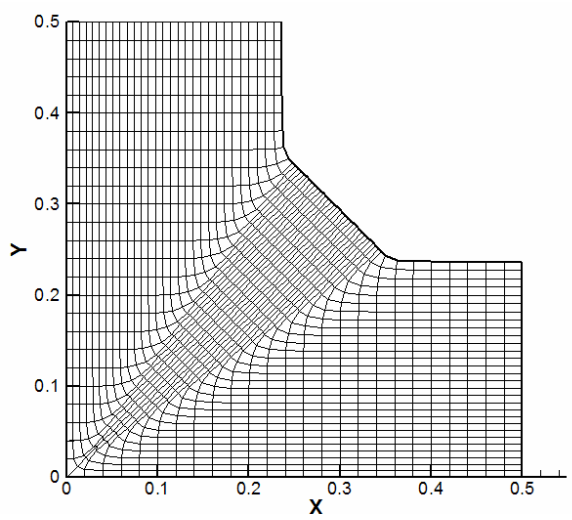


**Figure 5.7** Mesh around a concave corner

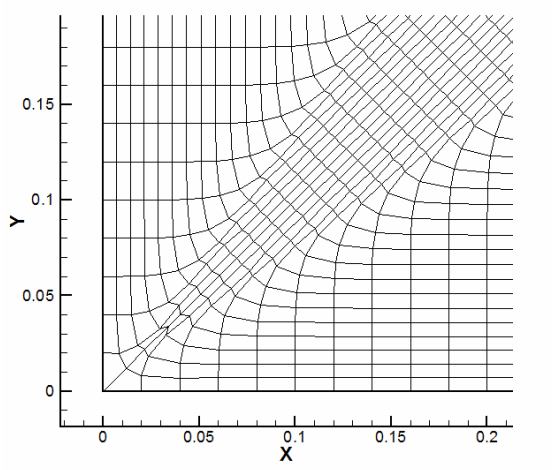


**Figure 5.8** Enlarged view of Figure 5.7

### 5.1.5 C-Mesh and O-Mesh around NACA0012 profile

C-mesh and O-mesh around NACA0012 airfoil profile is generated since it is the most common airfoil profile used in Computational Fluid Dynamics (CFD) applications. In Figures 5.9 and 5.10, C-mesh around NACA0012 profile and its enlarged view are given, respectively. The input parameters for C-mesh are given in Table 5.5.

**Table 5.5** Input parameters for C-mesh around NACA0012

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Surface clustering coefficient, *scc* | 0.005 |
| Fourth order damping coefficient, *dc* | 0.0 |
| Volume scaling coefficient, *vsc* | 1.0 |
| Number of points $(\xi x \eta)$ | 351x100 |

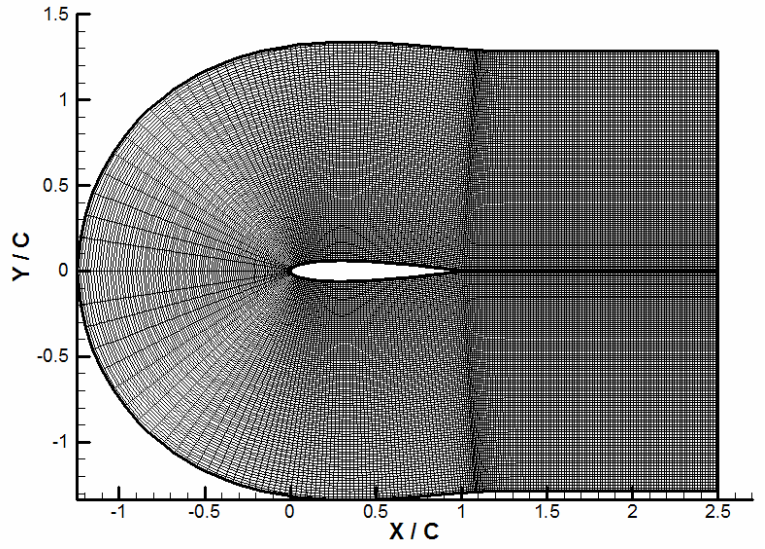


**Figure 5.9** C-mesh around NACA0012 profile

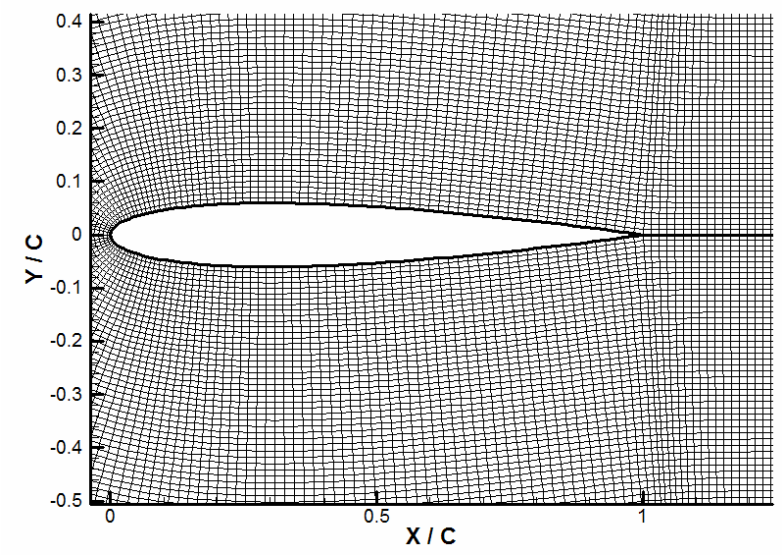


**Figure 5.10** Enlarged view of Figure 5.9

O-mesh around NACA0012 profile and enlarged view of it are given in Figures 5.11 and 5.12, respectively. The input parameters for O-mesh around NACA0012 profile are given in Table 5.6.

**Table 5.6** Input parameters for O-mesh around NACA0012

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Surface clustering coefficient, *scc* | 0.01 |
| Fourth order damping coefficient, *dc* | 0.07 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Number of points $(\xi x \eta)$ | 151x100 |

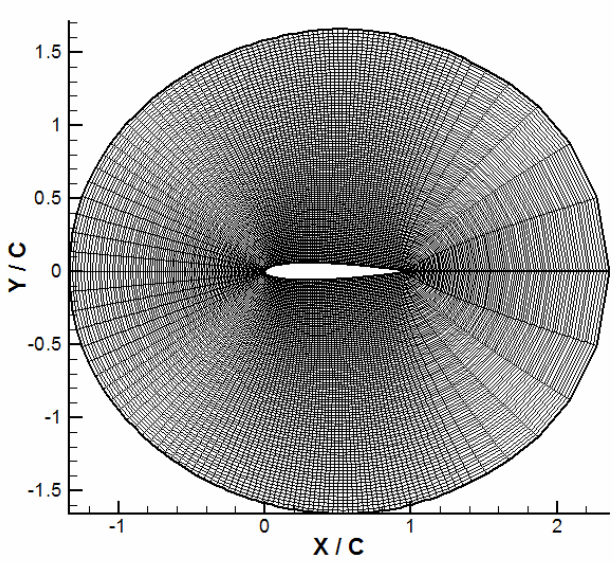


**Figure 5.11** O-mesh around NACA0012 profile

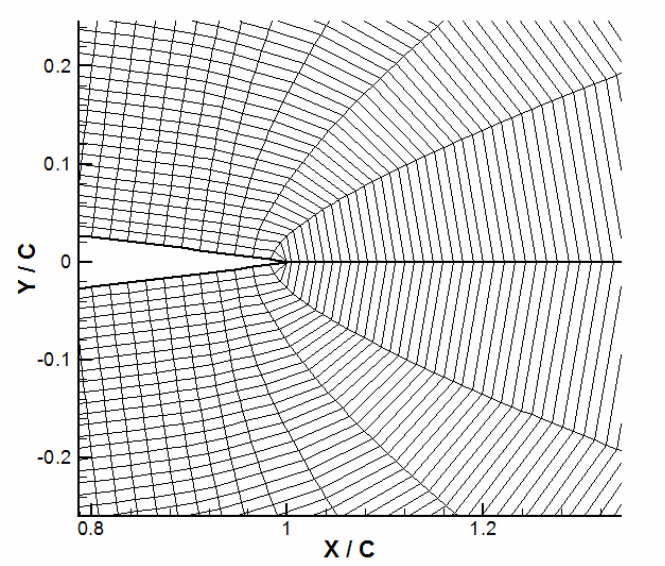


**Figure 5.12** Enlarged view of Figure 5.11

### 5.1.6 C-Mesh and O-Mesh around NLR7301 profile

C-mesh and O-mesh around an NLR7301 airfoil profile, which is a transonic airfoil profile, are generated. In Figures 5.13 and 5.14, C-mesh around NLR7301 and enlarged view of it are given, respectively. Input parameters for C-mesh are given in Table 5.7.

**Table 5.7** Input parameters for C-mesh around NLR7301

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Surface clustering coefficient, *scc* | 0.005 |
| Fourth order damping coefficient, *dc* | 0.03 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Number of points $(\xi x \eta)$ | 301x100 |

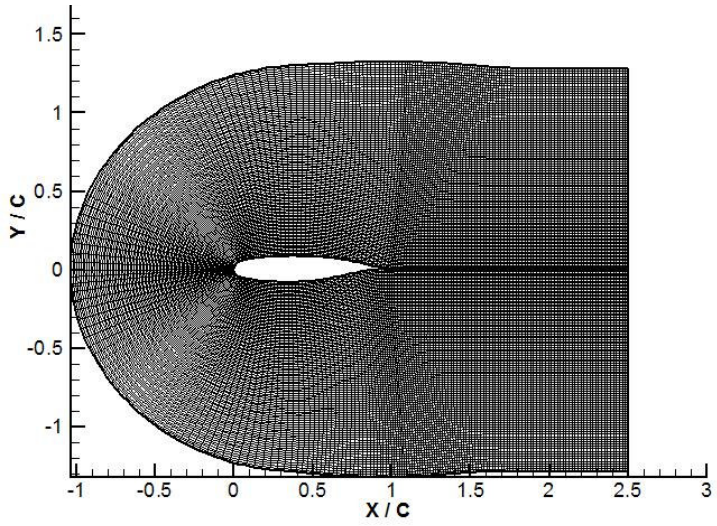


**Figure 5.13** C-mesh around NLR7301 profile

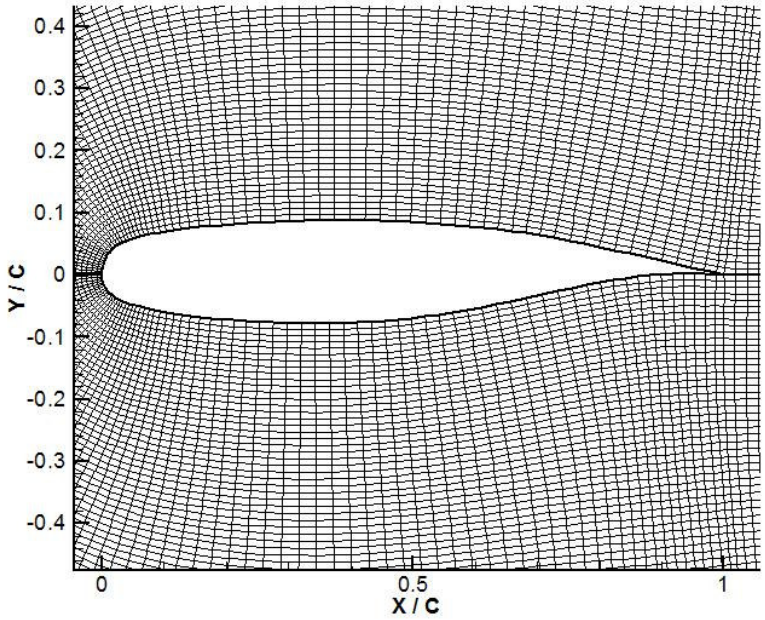


**Figure 5.14** Enlarged view of Figure 5.13

O-mesh around NLR7301 profile and enlarged view of it are given in Figures 5.15 and 5.16 respectively. Input parameters for O-mesh are given in Table 5.8.

**Table 5.8** Input parameters for O-mesh around NLR7301

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Surface clustering coefficient, *scc* | 0.01 |
| Fourth order damping coefficient, *dc* | 0.06 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Number of points $(\xi x \eta)$ | 79x80 |

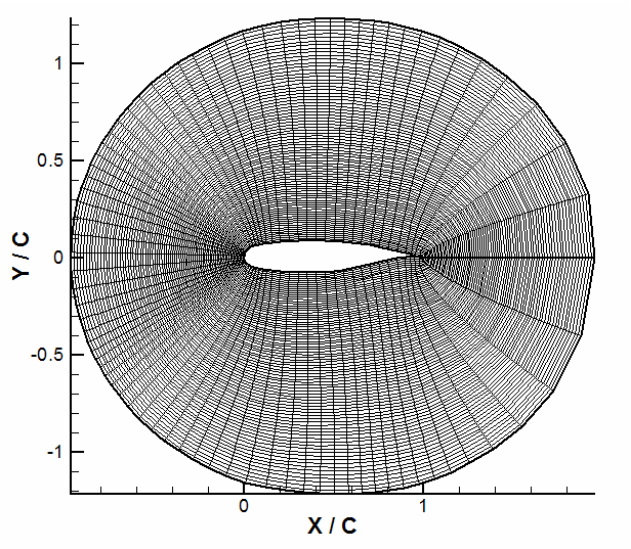


**Figure 5.15** O-mesh around NLR7301 profile

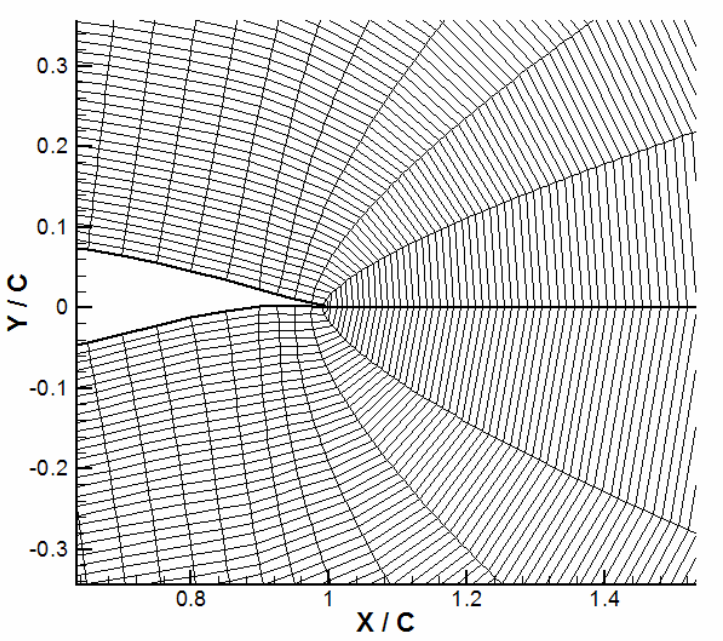


**Figure 5.16** Enlarged view of Figure 5.15

### 5.1.7 C-Mesh and O-Mesh around RAE2822 profile

C-mesh and O-mesh around RAE2822 profile are generated as another example to transonic airfoils. C-mesh around RAE2822 and enlarged view of it are given in Figures 5.17 and 5.18, respectively. Input parameters for C-mesh around RAE2822 profile are given in Figures 5.9.

**Table 5.9** Input parameters for C-mesh around RAE2822

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Surface clustering coefficient, *scc* | 0.005 |
| Fourth order damping coefficient, *dc* | 0.0 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Number of points $(\xi x \eta)$ | 241x100 |

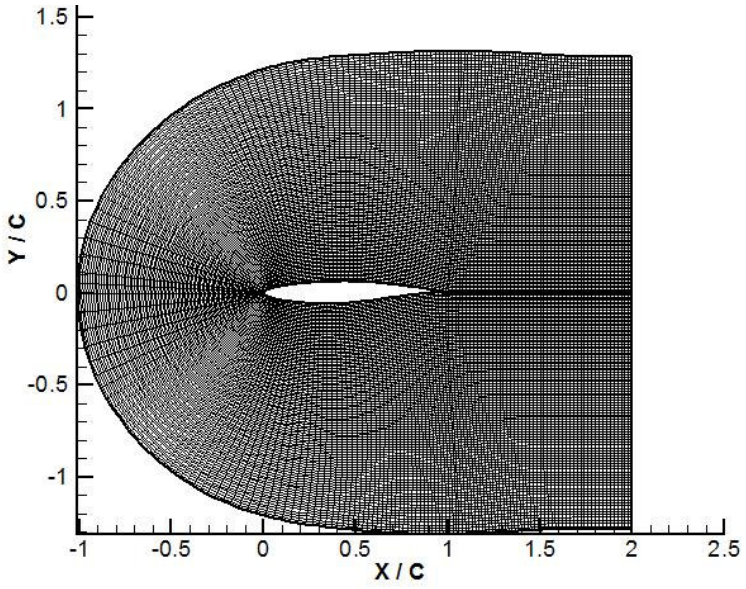


**Figure 5.17** C-mesh around RAE2822 profile

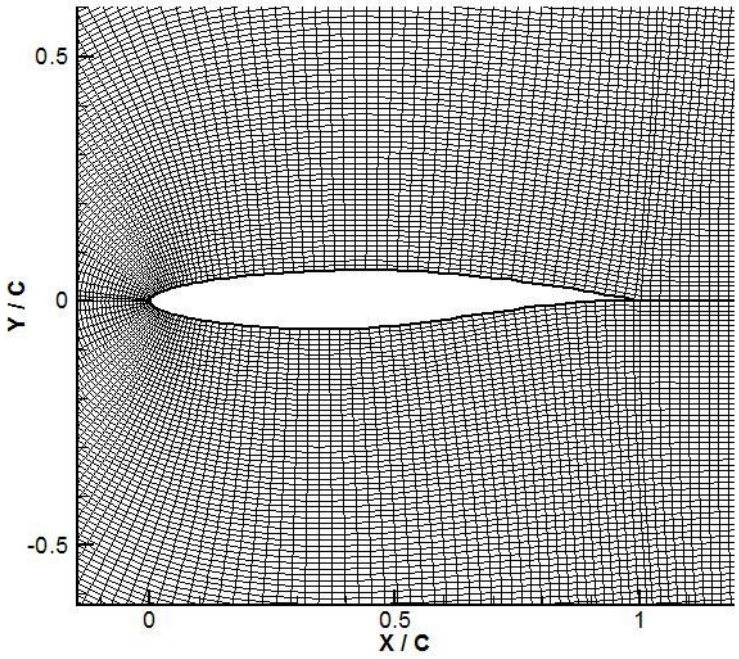


**Figure 5.18** Enlarged view of the Figure 5.17

O-mesh around RAE2822 profile and enlarged view of it are given in Figures 5.19 and 5.20 respectively. Input parameters of O-mesh are given in Table 5.10.

**Table 5.10** Input parameters for O-mesh around RAE2822

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Surface clustering coefficient, *scc* | 0.02 |
| Fourth order damping coefficient, *dc* | 0.06 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Number of points $(\xi x \eta)$ | 141x70 |

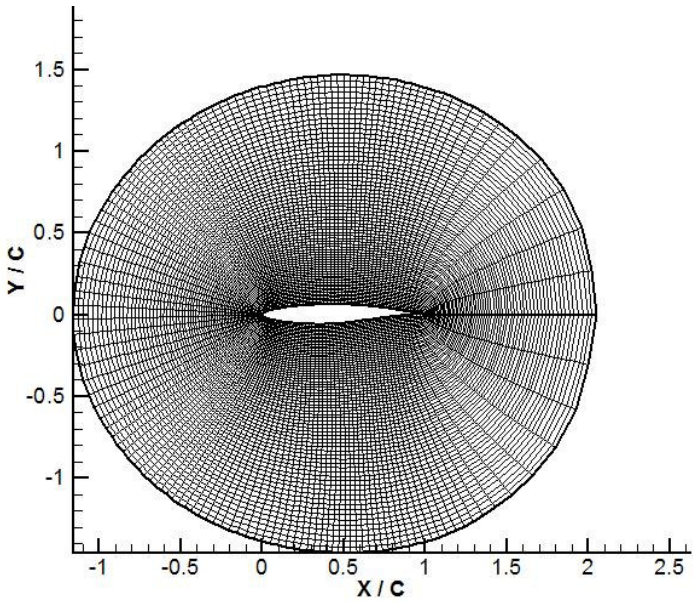


**Figure 5.19** O-mesh around RAE2822 profile

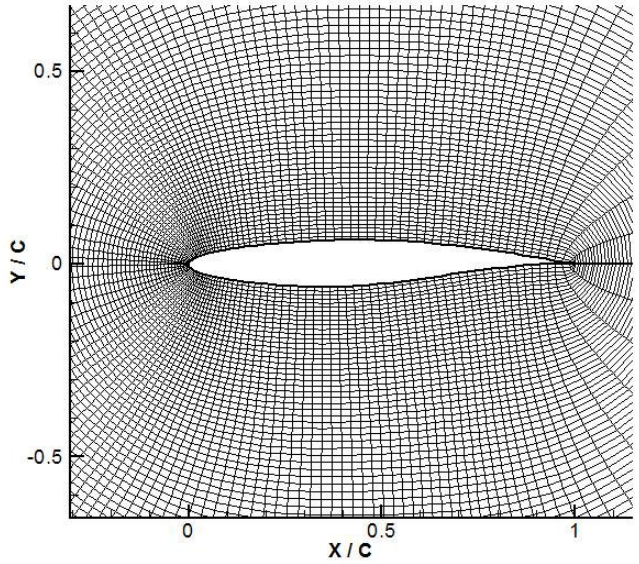


**Figure 5.20** Enlarged view of Figure 5.19

## 5.2 Two Dimensional Applications with Clustering Option

As one can notice from the input parameters given, surface clustering can be applied and determined by surface clustering coefficient. The aim behind this is to produce grids which can be used in Navier-Stokes' solver. Surface clustering is optional and left to the users' preference. If surface clustering is not desired, surface clustering coefficient should be taken as zero.

In addition, clustered grids in $\xi$ and $\eta$ directions can be generated by using hyperbolic grid generation code. Two dimensional applications with clustering option is given under the following headings.

### 5.2.1 O-Mesh around a Circle with Clustering in $\eta$ Direction

O-mesh around a circle having a radius of 0.5 and centered at the origin with clustering in $\eta$ direction and its enlarged view are given in Figures 5.21 and 5.22, respectively. Input parameters are given in Table 5.11.

**Table 5.11** Input parameters for O-mesh around a circle with clustering in $\eta$

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.04 |
| Surface clustering coefficient, *scc* | 0.0 |
| Fourth order damping coefficient, *dc* | 0.02 |
| Volume scaling coefficient, *vsc* | 1.0 |
| Level to be stretched, *js* | 25 |
| Interval of stretching, int *j* | 10 |
| Value of stretching at the required level, *sj* | 0.004 |
| Number of points $(\xi x \eta)$ | 201x50 |

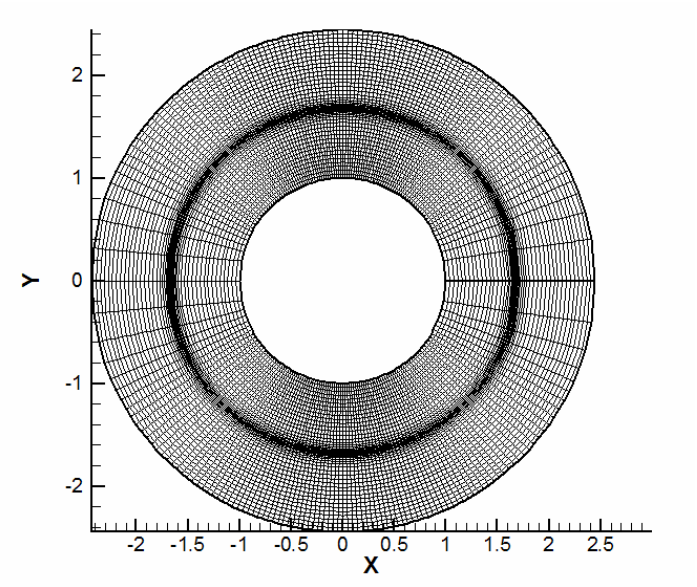


**Figure 5.21** O-mesh around a circle with clustering in $\eta$ direction

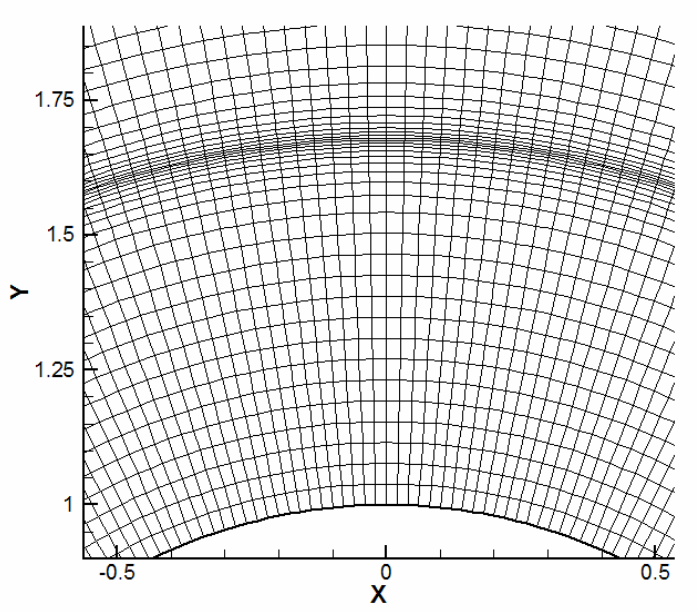


**Figure 5.22** Enlarged view of Figure 5.21

### 5.2.2 O-Mesh and C-Mesh around NACA0012 Profile with Clustering in $\eta$

O-mesh around NACA0012 profile with clustering in $\eta$ direction and the enlarged view are given in Figures 5.23 and 5.24, respectively. The input parameters are given in Table 5.12.

**Table 5.12** Input parameters for O-mesh around NACA0012 with clustering in $\eta$

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Surface clustering coefficient, *scc* | 0.0 |
| Fourth order damping coefficient, *dc* | 0.07 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Level to be stretched, *js* | 70 |
| Interval of stretching, int *j* | 20 |
| Value of stretching at the required level, *sj* | 0.004 |
| Number of points $(\xi x \eta)$ | 159x150 |

C-mesh around NACA0012 profile with clustering in $\eta$ direction and its enlarged view of it are presented in Figures 5.25 and 5.26, respectively. The input parameters are given in Table 5.13.

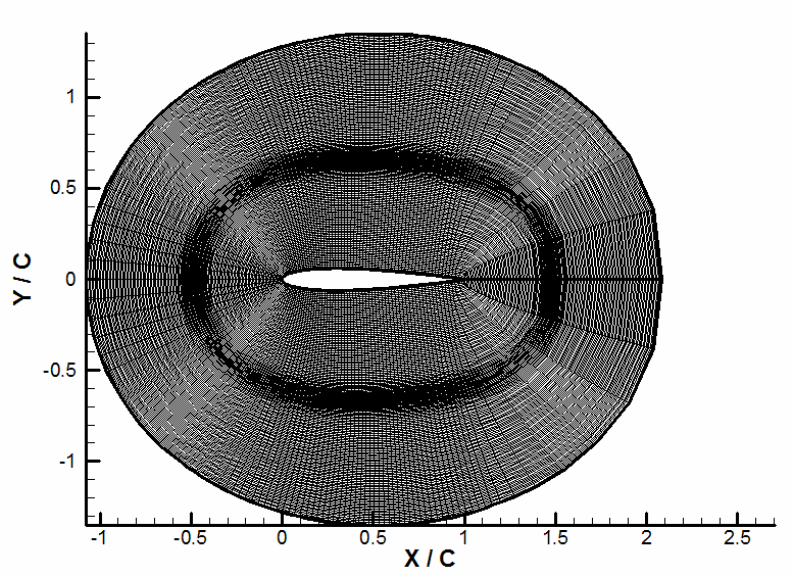


**Figure 5.23** O-mesh around NACA0012 profile with clustering in $\eta$ direction

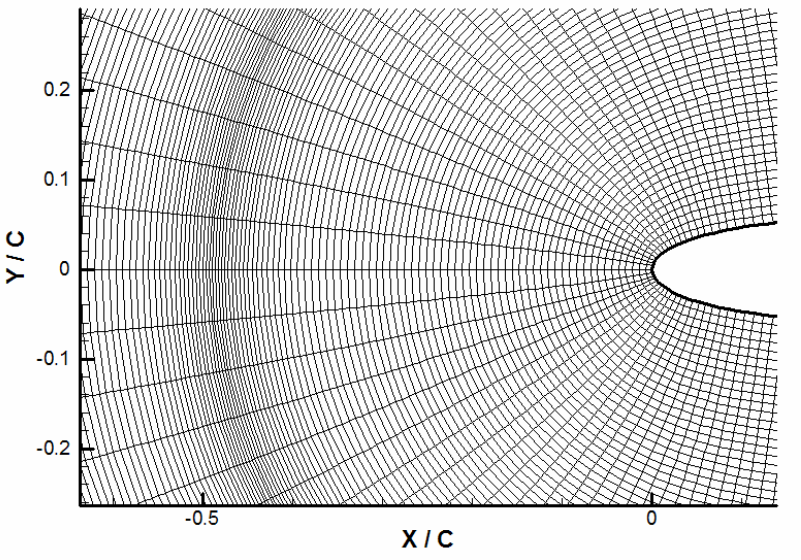


**Figure 5.24** Enlarged view of Figure 5.23

**Table 5.13** Input parameters for C-mesh around NACA0012 with clustering in $\eta$

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Surface clustering coefficient, *scc* | 0.01 |
| Fourth order damping coefficient, *dc* | 0.0 |
| Volume scaling coefficient, *vsc* | 1. |
| Level to be stretched, *js* | 50 |
| Interval of stretching, int *j* | 15 |
| Value of stretching at the required level, *sj* | 0.005 |
| Number of points $(\xi x \eta)$ | 301x90 |

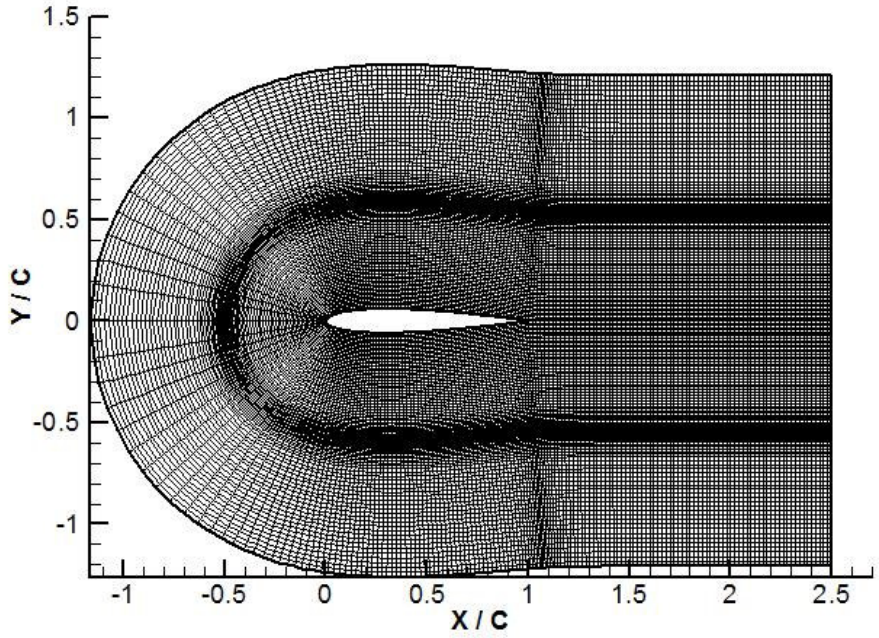


**Figure 5.25** C-mesh around NACA0012 profile with clustering in $\eta$ direction

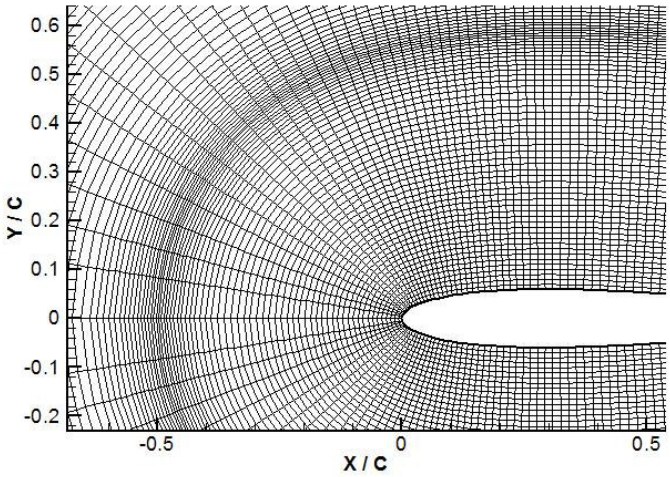


**Figure 5.26** Enlarged view of Figure 5.25

### 5.2.3 C-Mesh around RAE2822 Profile with Clustering in $\eta$

C-mesh around RAE2822 profile with clustering in $\eta$ direction and enlarged view of it are given in Figures 5.27 and 5.28 respectively. The input parameters are given in Table 5.14.

**Table 5.14** Input parameters for C-mesh around RAE2822 with clustering in $\eta$

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.02 |
| Surface clustering coefficient, *scc* | 0.005 |
| Fourth order damping coefficient, *dc* | 0.0 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Level to be stretched, *js* | 35 |
| Interval of stretching, int *j* | 15 |
| Value of stretching at the required level, *sj* | 0.005 |
| Number of points $(\xi x \eta)$ | 291x70 |

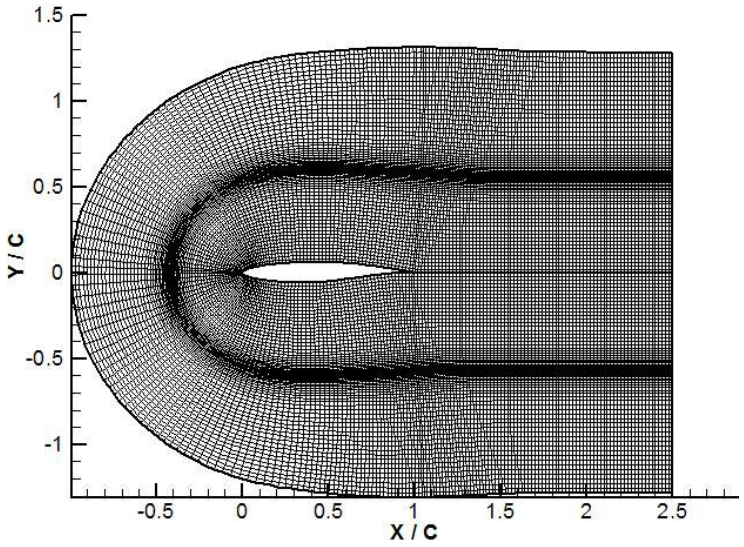


**Figure 5.27** C-mesh around RAE2822 profile with clustering in $\eta$ direction

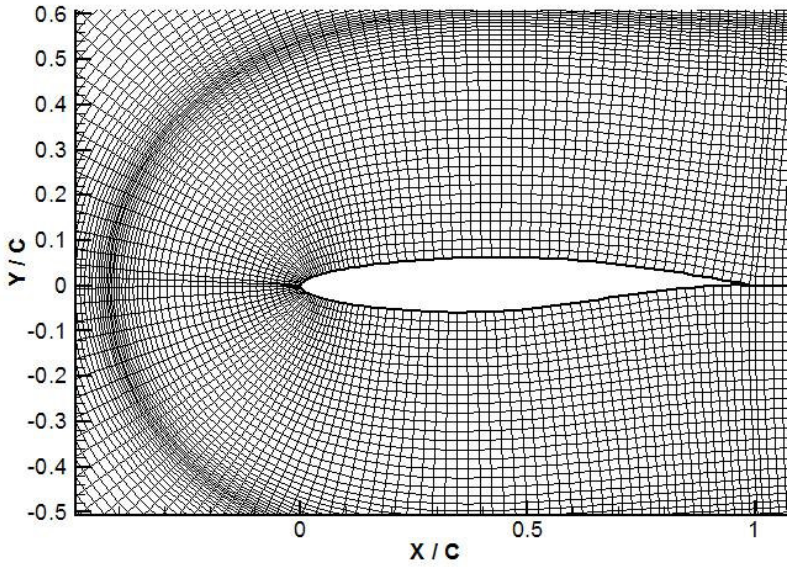


**Figure 5.28** Enlarged view of Figure 5.27

### 5.2.4 C-mesh and O-mesh around NACA0012 Profile with Clustering in $\xi$

C-mesh around NACA0012 profile with clustering in $\xi$ direction and its enlarged view are given in Figures 5.29 and 5.30 respectively. The input parameters are given in Table 5.15 and inputs of input_cluster.txt which is used for clustering in $\xi$ direction are given in Table 5.16.

**Table 5.15** Input parameters for C-mesh around NACA0012 profile with clustering in $\xi$

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Surface clustering coefficient, *scc* | 0.0 |
| Fourth order damping coefficient, *dc* | 0.07 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Number of points $(\xi x \eta)$ | 231x100 |

**Table 5.16** Input parameters of Input_cluster.txt

| ***Clustering Terminal*** | ***Spacing*** | ***Number of Points between Terminals*** |
|---|---|---|
| 0<br>0.65 | 0.01<br>0.001 | 80 |
| 0.65<br>0.75 | 0.001<br>0.01 | 20 |
| 0.75<br>1.0 | 0.01<br>0.01 | 30 |

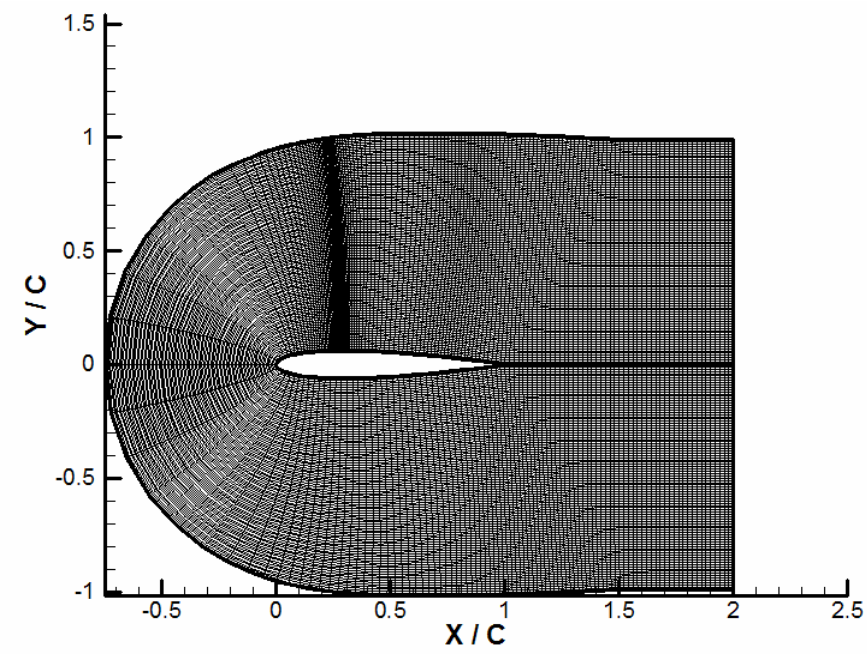


**Figure 5.29** C-mesh around NACA0012 profile with clustering in $\xi$ direction

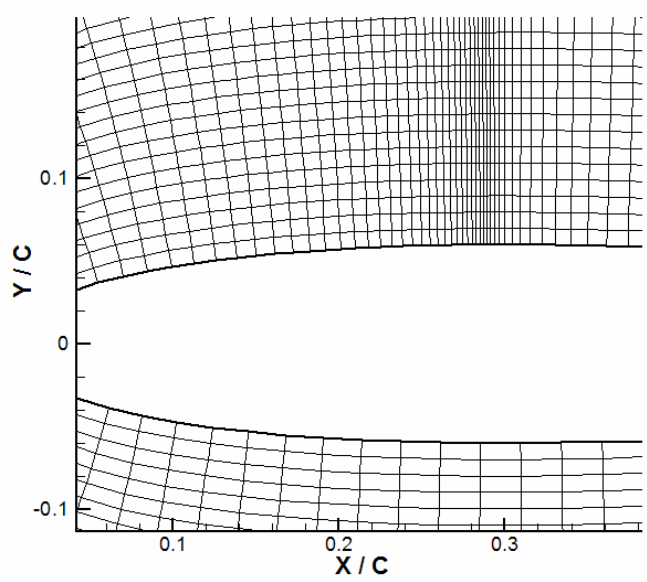


**Figure 5.30** Enlarged view of Figure 5.29

O-mesh around NACA0012 airfoil profile with clustering in $\xi$ direction and its enlarged view are given in Figures 5.31 and 5.32, respectively. Input parameters for this profile are given in Table 5.17 and input parameters of input_cluster.txt are given in Table 5.18.

**Table 5.17** Input parameters for O-mesh around NACA0012 profile with clustering in $\xi$

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Surface clustering coefficient, *scc* | 0.0 |
| Fourth order damping coefficient, *dc* | 0.07 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Number of points $(\xi x \eta)$ | 131x100 |

**Table 5.18** Input parameters for input_cluster.txt

| ***Clustering Terminal*** | ***Spacing*** | ***Number of Points between Terminals*** |
|---|---|---|
| 0<br>0.65 | 0.01<br>0.001 | 80 |
| 0.65<br>0.75 | 0.001<br>0.01 | 20 |
| 0.75<br>1.0 | 0.01<br>0.01 | 30 |

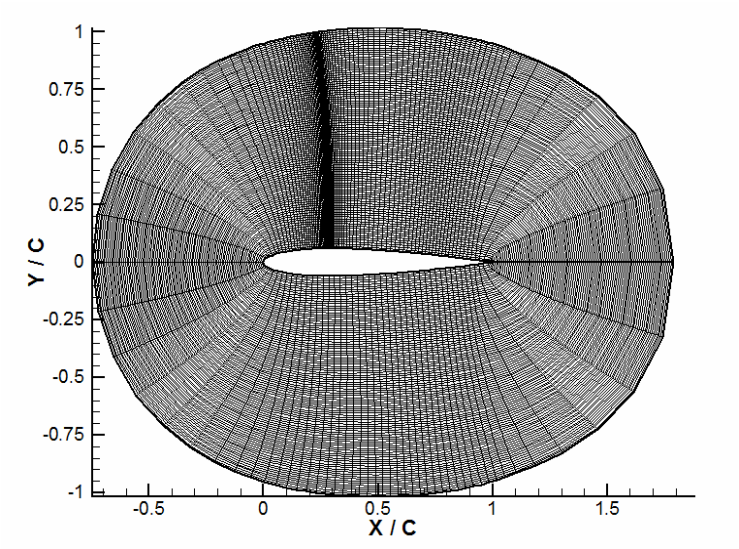


**Figure 5.31** O-mesh around NACA0012 profile with clustering in $\xi$ direction

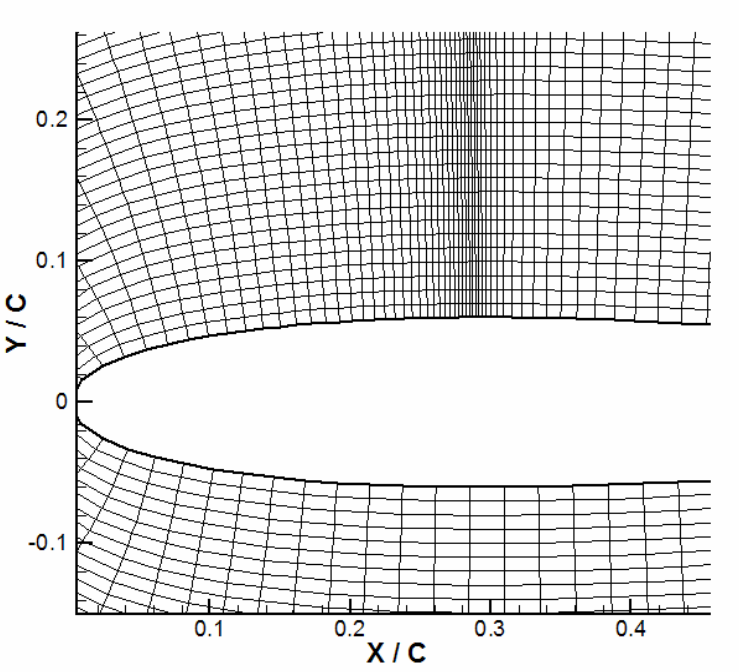


**Figure 5.32** Enlarged view of Figure 5.31

Besides, clustering in $\xi$ or in $\eta$ direction, clustering option can be applied for both directions simultaneously. As an example to this, C-mesh and O-mesh around NACA0012 profile with clustering in both directions are generated.

### 5.2.5 C-mesh and O-mesh around NACA0012 Profile with Clustering in both $\xi$ and $\eta$ Directions

C-mesh around NACA0012 profile with clustering in both directions and its enlarged view are given in Figures 5.33 and 5.34, respectively. The input parameters are given in Table 5.19 and input parameters of input_cluster.txt are given in Table 5.20. In Table 5.19, parameters used for $\eta$ clustering are given as well.

O-mesh around NACA0012 profile with clustering in both directions and its enlarged view are given in Figures 5.35 and 5.36, respectively. The input parameters are given in Table 5.21 and input parameters of input_cluster.txt are given in Table 5.22. In Table 5.21, parameters used for $\eta$ clustering are given as well.

**Table 5.19** Input parameters for C-mesh around NACA0012 profile

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Surface clustering coefficient, *scc* | 0.0 |
| Fourth order damping coefficient, *dc* | 0.07 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Level to be stretched, *js* | 40 |
| Interval of stretching, int *j* | 20 |
| Value of stretching at the required level, *sj* | 0.004 |
| Number of points $(\xi x \eta)$ | 231x100 |

**Table 5.20** Input parameters of Input_cluster.txt

| *Clustering Terminal* | *Spacing* | *Number of Points between Terminals* |
|---|---|---|
| 0<br>0.65 | 0.01<br>0.001 | 80 |
| 0.65<br>0.75 | 0.001<br>0.01 | 20 |
| 0.75<br>1.0 | 0.01<br>0.01 | 30 |

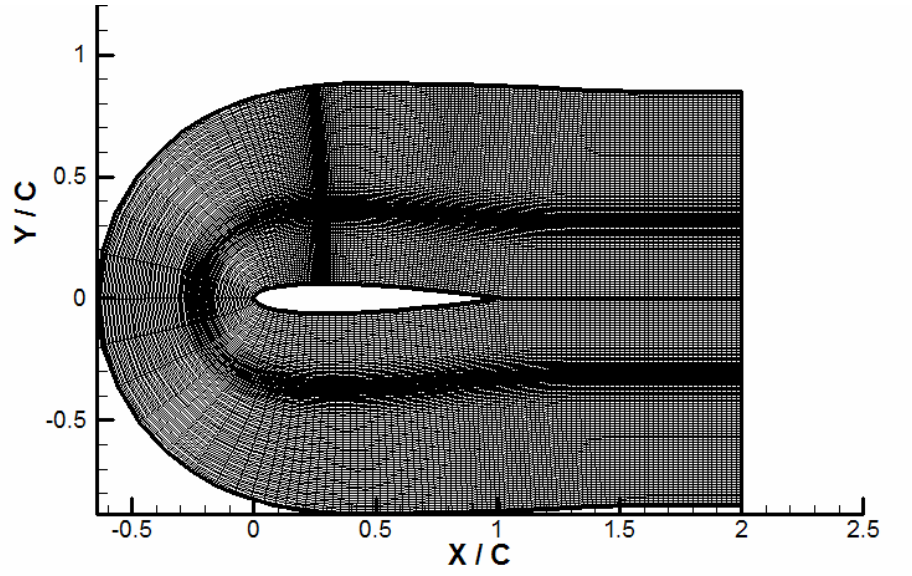


**Figure 5.33** C-mesh around NACA0012 profile with clustering in both $\eta$ and $\xi$ directions

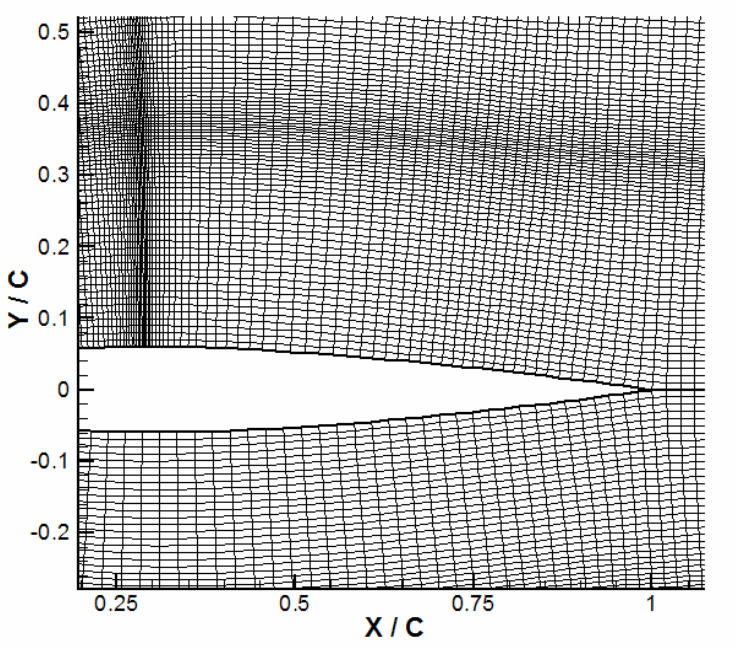


**Figure 5.34** Enlarged view of Figure 5.33

**Table 5.21** Input parameters for O-mesh around NACA0012 profile with clustering in both $\xi$ and $\eta$

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Surface clustering coefficient, *scc* | 0.0 |
| Fourth order damping coefficient, *dc* | 0.07 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Level to be stretched, *js* | 40 |
| Interval of stretching, int *j* | 20 |
| Value of stretching at the required level, *sj* | 0.004 |
| Number of points $(\xi x \eta)$ | 231x100 |

**Table 5.22** Input parameters of input_cluster.txt

| *Clustering Terminal* | *Spacing* | *Number of Points between Terminals* |
|---|---|---|
| 0<br>0.65 | 0.01<br>0.001 | 80 |
| 0.65<br>0.75 | 0.001<br>0.01 | 20 |
| 0.75<br>1.0 | 0.01<br>0.01 | 30 |

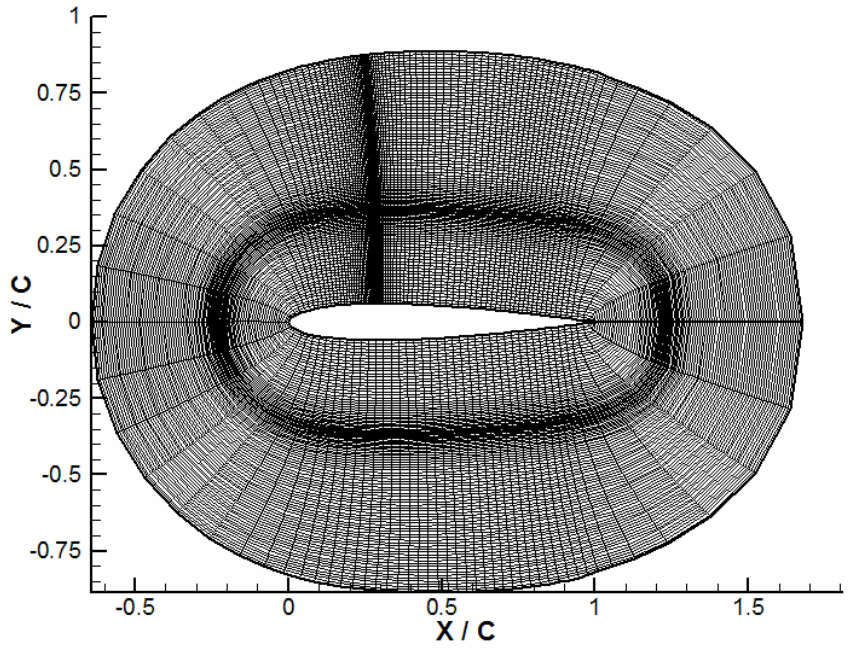


**Figure 5.35** O-mesh around NACA0012 profile with clustering in both $\eta$ and $\xi$ directions

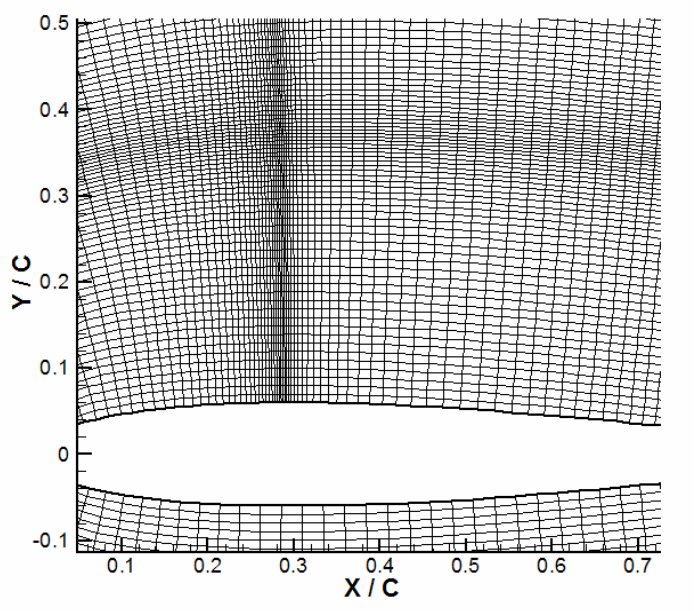


**Figure 5.36** Enlarged view of Figure 5.35

### 5.3 Three-Dimensional Applications

In this section, three-dimensional hyperbolic grid generation code is applied to different geometries. Most of these profiles are the extensions of two-dimensional grid applications given in Sections 5.1 and 5.2. The input parameters used for the generation of three dimensional grids are given as well.

#### 5.3.1 Flat Plate

As a first profile, three dimensional grids for flat plate are generated since it is a simple geometry. Mesh generated for flat plate and sections of this profile (on *x-z*, *y-z* and *x-y* planes) are given in Figures 5.37 and 5.38, respectively. In the three dimensional grid generation code, output of two dimensional grid generation code is used as the input. Thus, the only input parameters used in

this code are marching distance in $\zeta$ direction and number of points in $\zeta$ direction. Input parameters are given in Table 5.23.

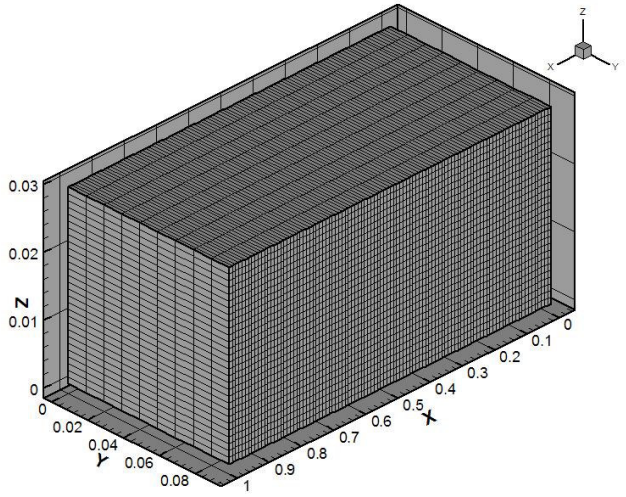


**Figure 5.37** Mesh generated for a flat plate

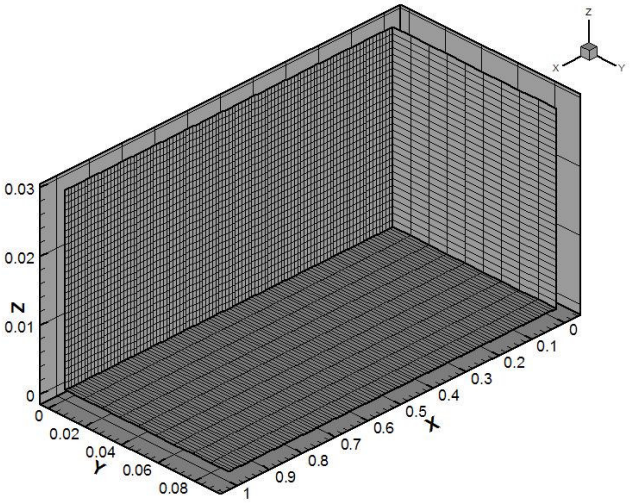


**Figure 5.38** Sections of meshes on *x*-*z*, *y*-*z* and *x*-*y* planes

**Table 5.23** Input parameters for flat plate

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.001 |
| Number of points $(\xi x \eta x \zeta)$ | 101x10x30 |

### 5.3.2 O-Mesh around an Ellipse

In two dimensional grid generation applications, two dimensional O-meshes around an ellipse having a major axis of 2 units and minor axis of 0.2 units and centered at the origin was shown. Also, the parameters used in this generation were given. As a three-dimensional application, these two dimensional O-meshes are extended to three dimensional O-meshes. This profile and its enlarged view are given in Figures 5.39 and 5.40, respectively. Input parameters are given in Table 5.24.

**Table 5.24** Input parameters for 3-D O-mesh around an ellipse

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.003 |
| Number of points $(\xi x \eta x \zeta)$ | 201x50x30 |

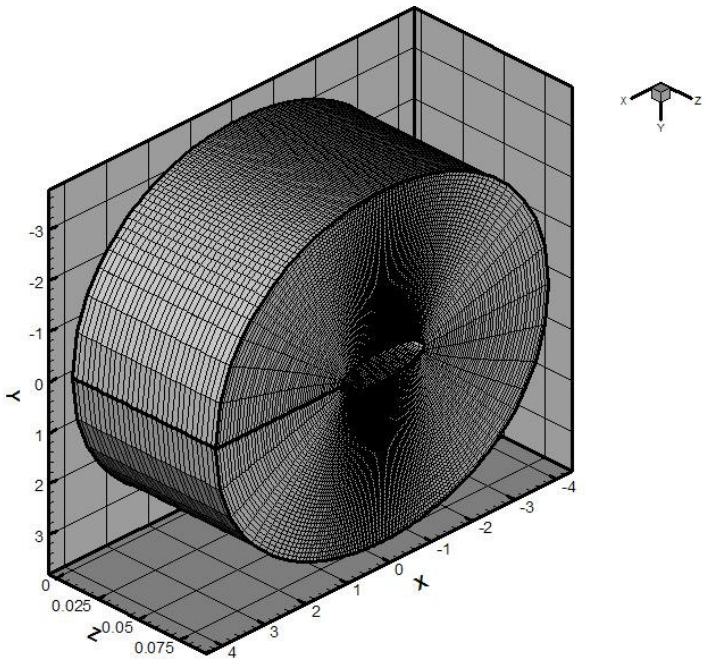


**Figure 5.39** O-mesh around an ellipse

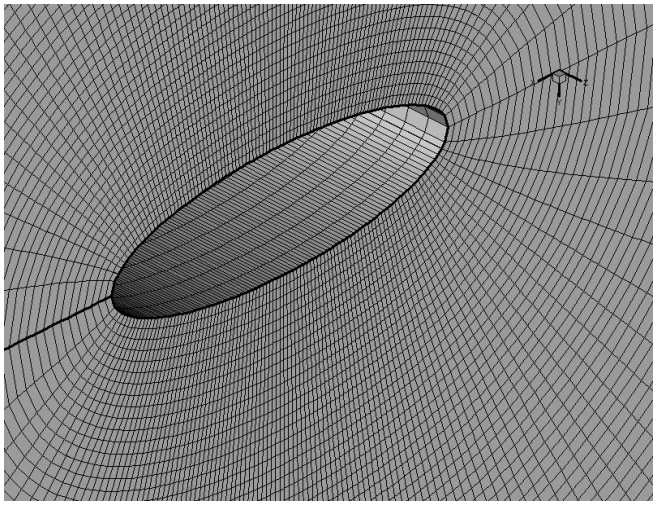

**Figure 5.40** Enlarged view of Figure 5.39

### 5.3.3 O-mesh around a Circle

For an O-mesh around a circle having a radius of 0.5 and centered at the origin, the parameters used in the generation of two dimensional grids were given in section 5.1.2. These meshes are extended to three dimensions. This profile and its enlarged view are given in Figures 5.41 and 5.42, respectively. Input parameters are given in Table 5.25.

**Table 5.25** Input parameters for 3-D O-mesh around a circle

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Number of points $(\xi x \eta x \zeta)$ | 201x50x40 |

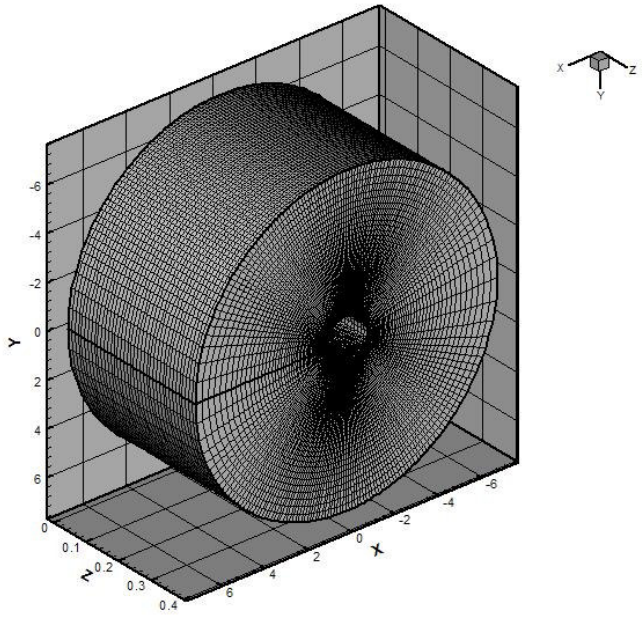


**Figure 5.41** O-mesh around a circle

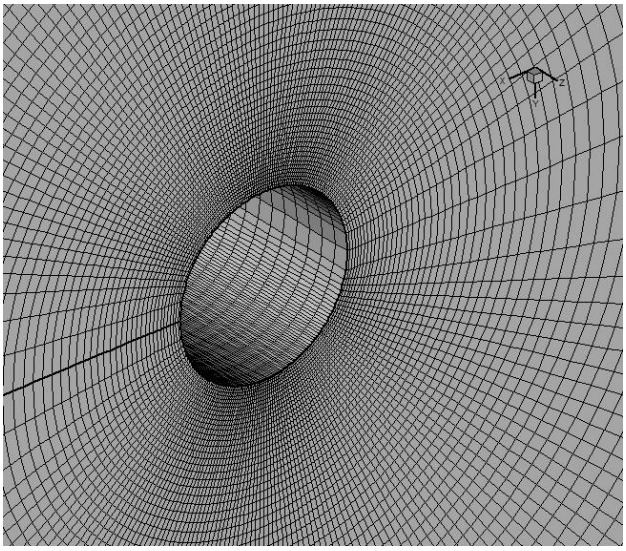

**Figure 5.42** Enlarged view of Figure 5.41

### 5.3.4 C-mesh and O-mesh around NACA0012 Profile

Two-dimensional C-mesh around NACA0012 profile is extended to three dimensions. The input parameters for two-dimensional C-mesh were given in Section 5.1.5. Input parameters for three-dimensional C-mesh are given in Table 5.26. This profile and its enlarged view are given in Figures 5.43 and 5.44, respectively.

**Table 5.26** Input parameters for 3-D C-mesh around NACA0012 profile

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Number of points $(\xi x \eta x \zeta)$ | 251x50x45 |

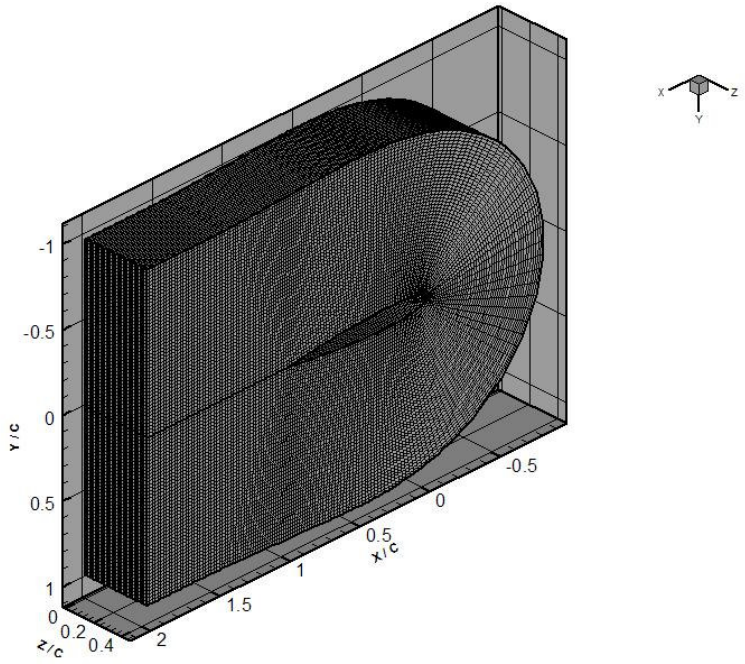


**Figure 5.43** C-mesh around NACA0012 profile

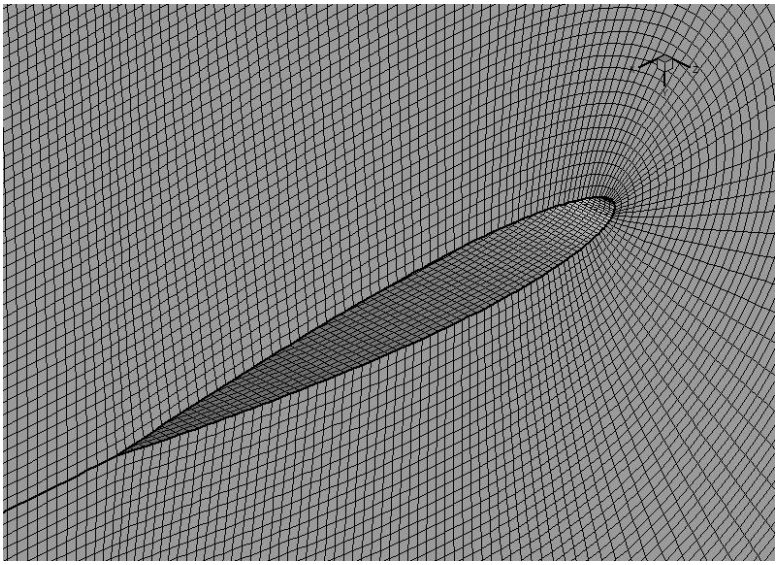

**Figure 5.44** Enlarged view of Figure 5.43

Three-dimensional O-mesh around NACA0012 profile and its enlarged view are given in Figures 5.45 and 5.46, respectively. Input parameters are given in Table 5.27.

**5.27** Input parameters for 3-D O-mesh around NACA0012 profile

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.02 |
| Number of points $(\xi x \eta x \zeta)$ | 151x100x40 |

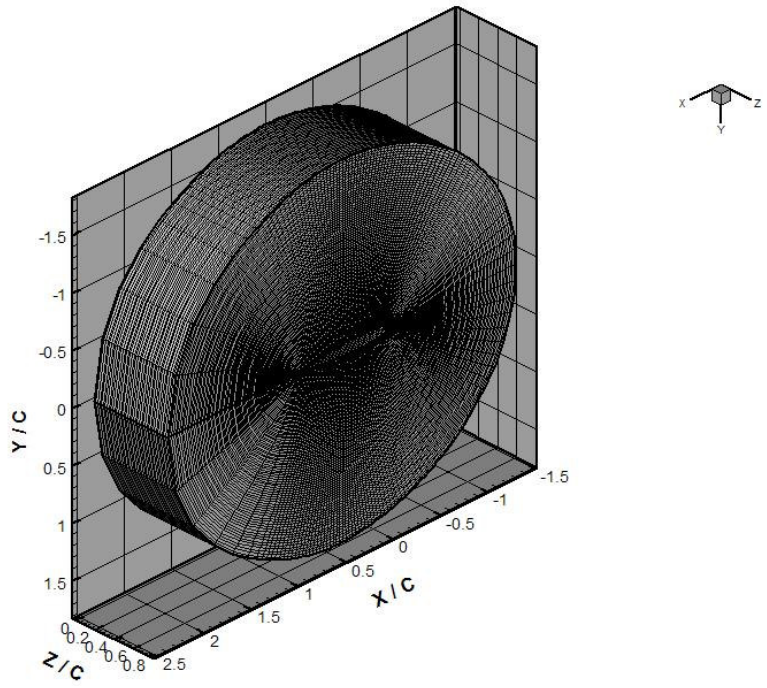


**Figure 5.45** O-mesh around NACA0012 profile

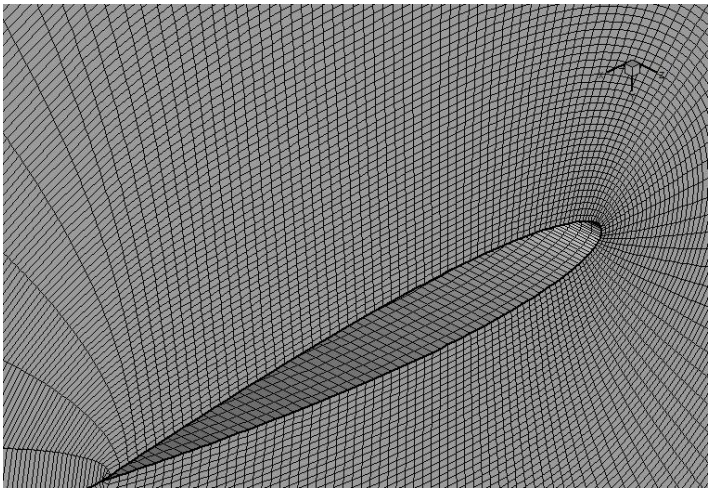

**Figure 5.46** Enlarged view of Figure 5.45

### 5.3.5 C-Mesh and O-Mesh around NLR7301

Three-dimensional C-mesh and enlarged view of it are given in Figures 5.47 and 5.48, respectively. It should be noticed that the values of input parameters used for two-dimensional C-mesh are different. So, all the parameters related with two and three dimensional grids are given in Table 5.28.

**Table 5.28** Input parameters for 3-D C-mesh around NLR7301 profile

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance in $\eta$ direction | 0.01 |
| Surface clustering coefficient, *scc* | 0.005 |
| Fourth order damping coefficient, *dc* | 0.03 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Marching distance in $\zeta$ direction | 0.01 |
| Number of points $(\xi x \eta x \zeta)$ | 301x100x40 |

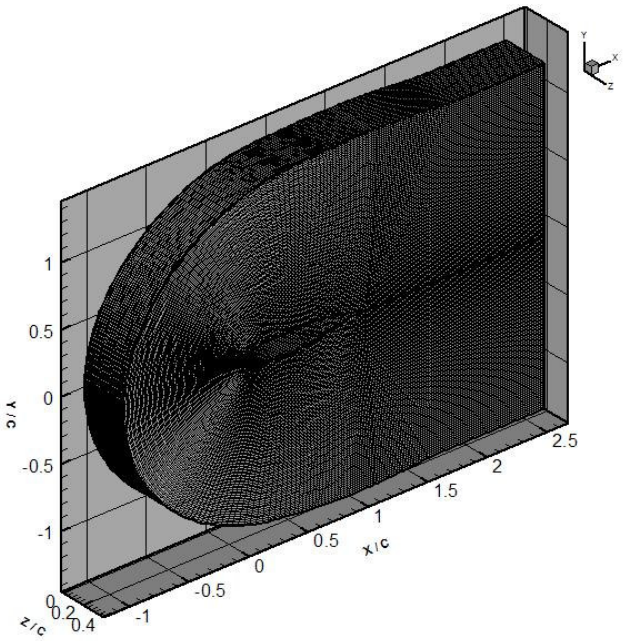


**Figure 5.47** C-mesh around NLR7301 airfoil profile

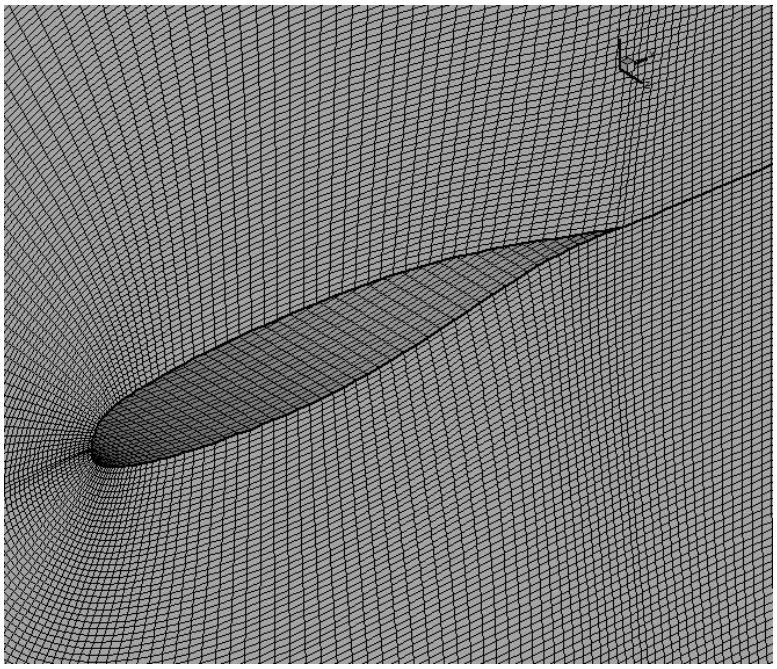

**Figure 5.48** Enlarged view of Figure 5.47

Three-dimensional O-mesh and its enlarged view are given in Figures 5.49 and 5.50, respectively. Input parameters of this profile are given in Table 5.29.

**Table 5.29** Input parameters for 3-D O-mesh around NLR7301 profile

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance in $\eta$ direction | 0.01 |
| Surface clustering coefficient, *scc* | 0.01 |
| Fourth order damping coefficient, *dc* | 0.06 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Marching distance in $\zeta$ direction | 0.01 |
| Number of points $(\xi x \eta x \zeta)$ | 141x80x60 |

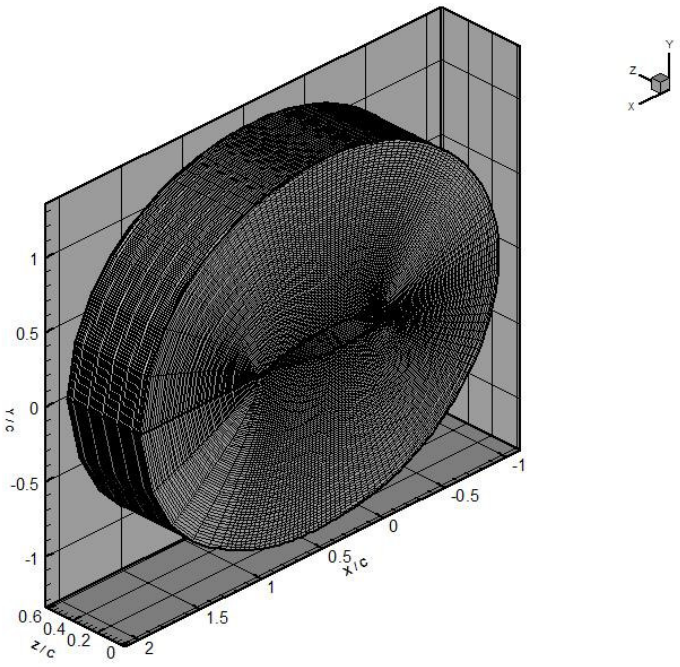


**Figure 5.49** O-mesh around NLR7301 profile

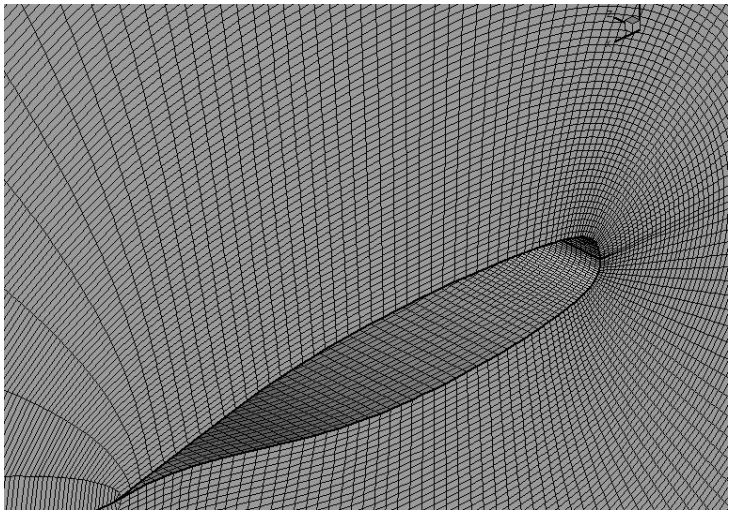

**Figure 5.50** Enlarged view of Figure 5.49

### 5.3.6 Mesh around a Convex Corner

Two dimensional grid for profile having convex corner is extended to three dimensions. Three-dimensional mesh around a convex corner and its enlarged view are given in Figures 5.51 and 5.52, respectively. Input parameters are given in Table 5.30.

**Table 5.30** Input parameters for 3-D O-mesh around a convex corner

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance, $\Delta s$ | 0.01 |
| Number of points $(\xi x \eta x \zeta)$ | 101x40x45 |

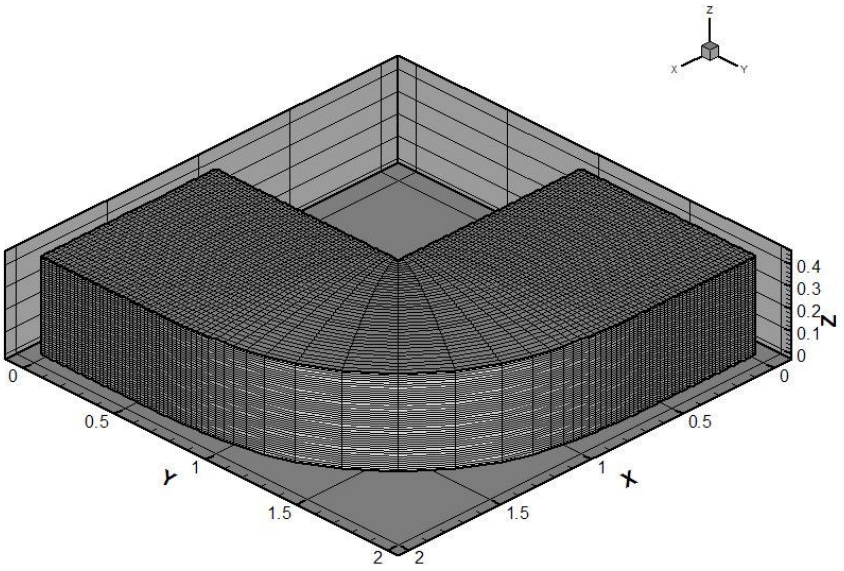


**Figure 5.51** Mesh around a convex corner

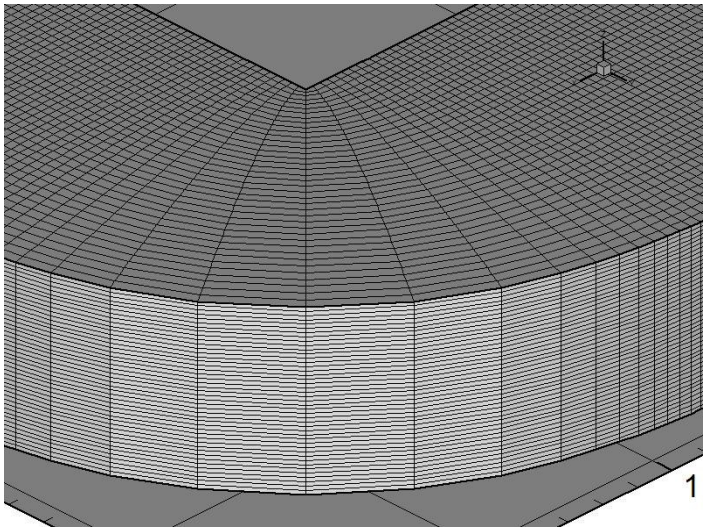


**Figure 5.52** Enlarged view of Figure 5.51

## 5.4 Three Dimensional Applications with Clustering Option

In order to show clustering options, three dimensional C-mesh and O-mesh around NACA0012 airfoil profile with clustering in $\xi$ direction, in $\eta$ direction and in both $\xi$ and $\eta$ directions are generated. Details of these profiles are given under the following headings.

### 5.4.1 C-Mesh and O-Mesh around NACA0012 with Clustering in $\xi$ Direction

C-mesh around NACA0012 with clustering in $\xi$ direction and its enlarged view are given in Figures 5.53 and 5.54, respectively. Input parameters are given in Table 5.31. Input parameters for $\xi$ clustering are given in Table 5.32.

**Table 5.31** Input parameters for 3-D C-mesh with clustering in $\xi$

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance in $\eta$ direction | 0.01 |
| Surface clustering coefficient, *scc* | 0.0 |
| Fourth order damping coefficient, *dc* | 0.07 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Marching distance in $\zeta$ direction | 0.005 |
| Number of points $(\xi x \eta x \zeta)$ | 231x100x40 |

**Table 5.32** Input parameters for $\xi$ clustering of 3-D C-mesh

| *Clustering Terminal* | *Spacing* | *Number of Points between Terminals* |
|---|---|---|
| 0<br>0.65 | 0.01<br>0.001 | 80 |
| 0.65<br>0.75 | 0.001<br>0.01 | 20 |
| 0.75<br>1.0 | 0.01<br>0.01 | 30 |

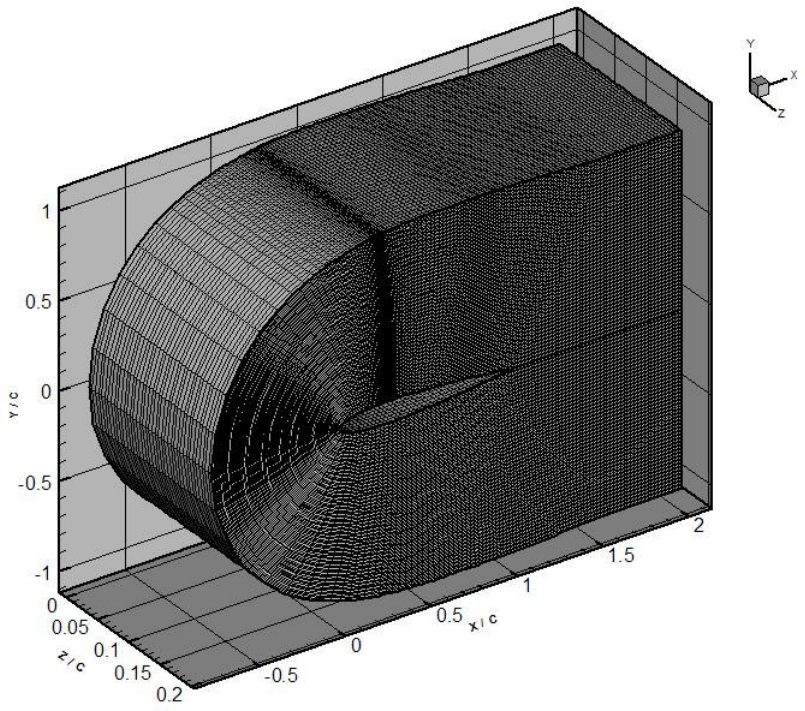


**Figure 5.53** C-mesh around NACA 0012 profile with clustering in $\xi$ direction

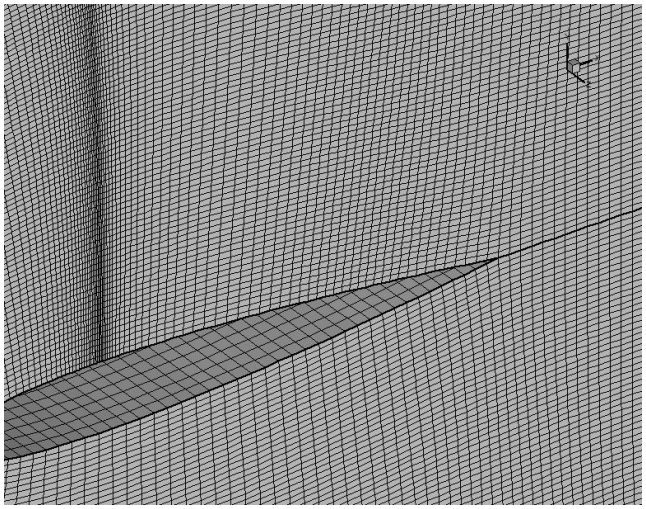

**Figure 5.54** Enlarged view of Figure 5.53

O-mesh around NACA0012 with clustering in $\xi$ direction and its enlarged view are given in Figures 5.55 and 5.56, respectively. Input parameters are given in Table 5.33. Input parameters used for clustering in $\xi$ direction are given in Figure 5.34.

**Table 5.33** Input parameters for 3-D O-mesh around NACA0012 profile with clustering in $\xi$ direction

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance in $\eta$ direction | 0.01 |
| Surface clustering coefficient, *scc* | 0.0 |
| Fourth order damping coefficient, *dc* | 0.07 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Marching distance in $\zeta$ direction | 0.007 |
| Number of points $(\xi x \eta x \zeta)$ | 131x100x60 |

**Table 5.34** Input parameters for $\xi$ clustering of 3-D O-mesh around NACA0012 profile

| *Clustering Terminal* | *Spacing* | *Number of Points between Terminals* |
|---|---|---|
| 0<br>0.65 | 0.01<br>0.001 | 80 |
| 0.65<br>0.75 | 0.001<br>0.01 | 20 |
| 0.75<br>1.0 | 0.01<br>0.01 | 30 |

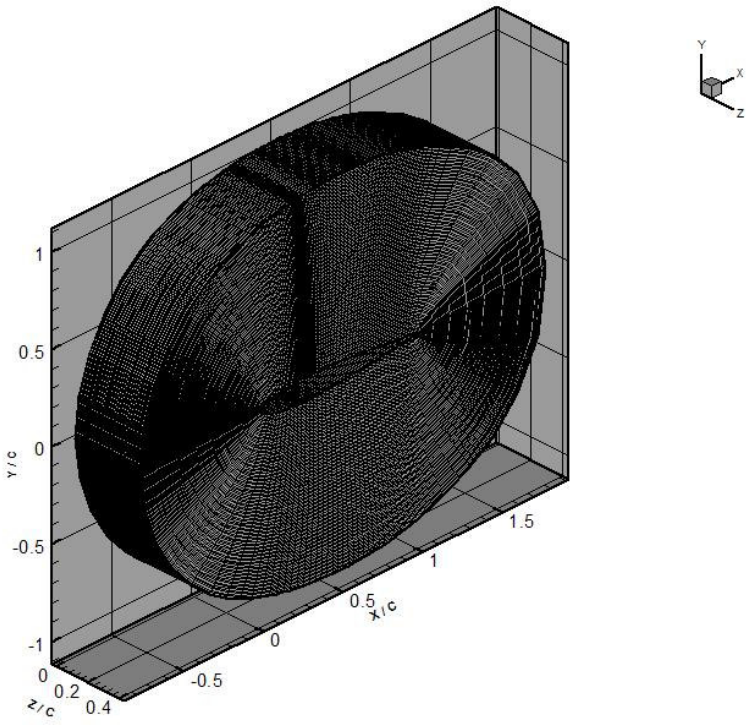


**Figure 5.55** O-mesh around NACA0012 profile with clustering in $\xi$ direction

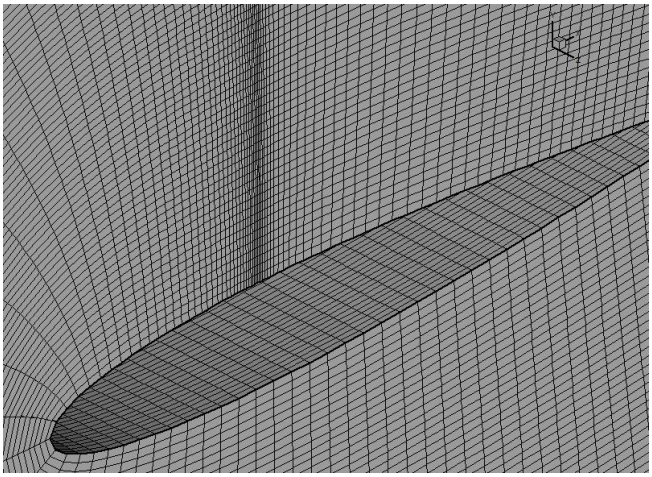

**Figure 5.56** Enlarged view of Figure 5.55

### 5.4.2 O-Mesh around NACA0012 with Clustering in $\eta$ Direction

O-mesh around NACA0012 with clustering in $\eta$ direction and its enlarged view are given in Figures 5.57 and 5.58, respectively. Input parameters, including clustering in $\eta$ parameters, are given in Table 5.35.

**Table 5.35** Input parameters for 3-D O-mesh with clustering in $\eta$

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance in $\eta$ direction | 0.01 |
| Surface clustering coefficient, *scc* | 0.0 |
| Fourth order damping coefficient, *dc* | 0.07 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Level to be stretched, *js* | 60 |
| Interval of stretching, int *j* | 20 |
| Value of stretching at the required level, *sj* | 0.005 |
| Marching distance in $\zeta$ direction | 0.01 |
| Number of points $(\xi x \eta x \zeta)$ | 151x100x40 |

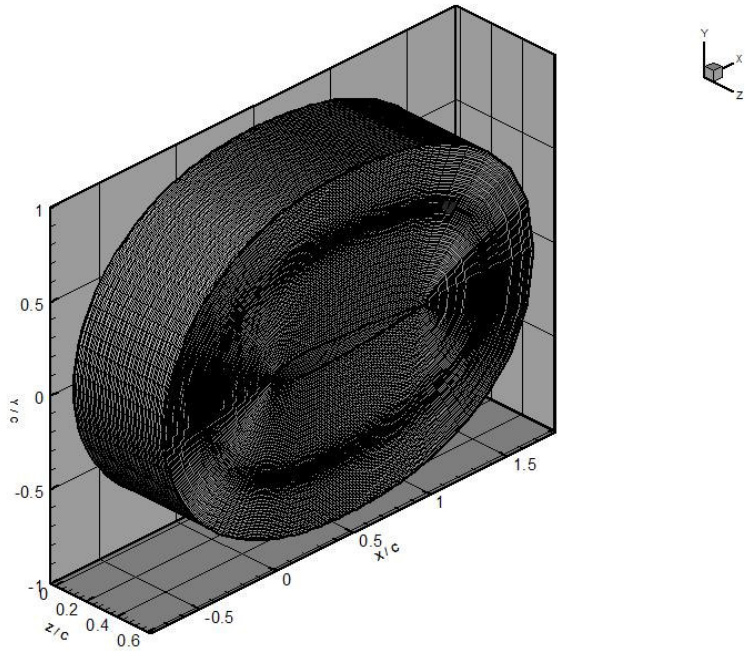


**Figure 5.57** O-mesh around NACA0012 profile with clustering in $\eta$ direction

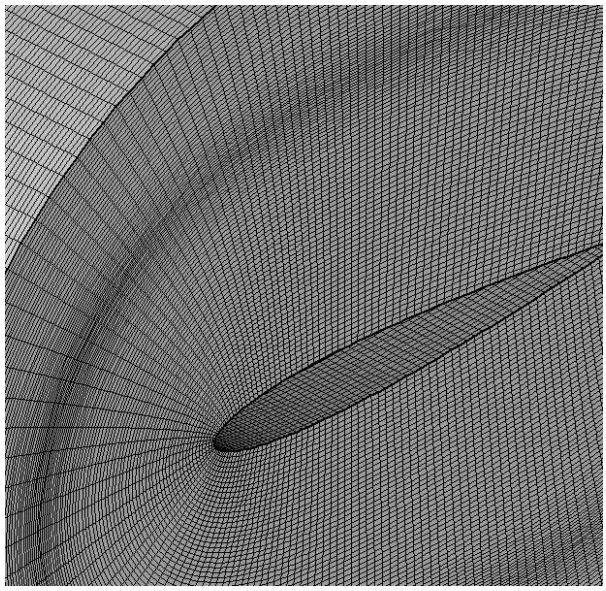

**Figure 5.58** Enlarged view of Figure 5.57

### 5.4.3 O-mesh around NACA0012 with Clustering in both $\xi$ and $\eta$ Directions

O-mesh around NACA0012 with clustering in both $\xi$ and $\eta$ directions and enlarged view of it are given in Figures 5.59 and 5.60, respectively. Input parameters are given in Table 5.36. Input parameters used for clustering in $\xi$ direction are given in Table 5.37.

**Table 5.36** Input parameters for O-mesh with clustering in $\xi$ and $\eta$

| ***Parameters*** | ***Values*** |
|---|---|
| Marching distance in $\eta$ direction | 0.02 |
| Surface clustering coefficient, *scc* | 0.005 |
| Fourth order damping coefficient, *dc* | 0.05 |
| Volume scaling coefficient, *vsc* | 0.8 |
| Level to be stretched, *js* | 60 |
| Interval of stretching, int *j* | 20 |
| Value of stretching at the required level, *sj* | 0.005 |
| Marching distance in $\zeta$ direction | 0.004 |
| Number of points $(\xi x \eta x \zeta)$ | 161x100x50 |

**Table 5.37** Input parameters for $\xi$ clustering of 3-D O-mesh around NACA0012 profile

| *Clustering Terminal* | *Spacing* | *Number of Points between Terminals* |
|---|---|---|
| 0<br>0.65 | 0.01<br>0.001 | 80 |
| 0.65<br>0.75 | 0.001<br>0.01 | 20 |
| 0.75<br>1.0 | 0.01<br>0.01 | 30 |

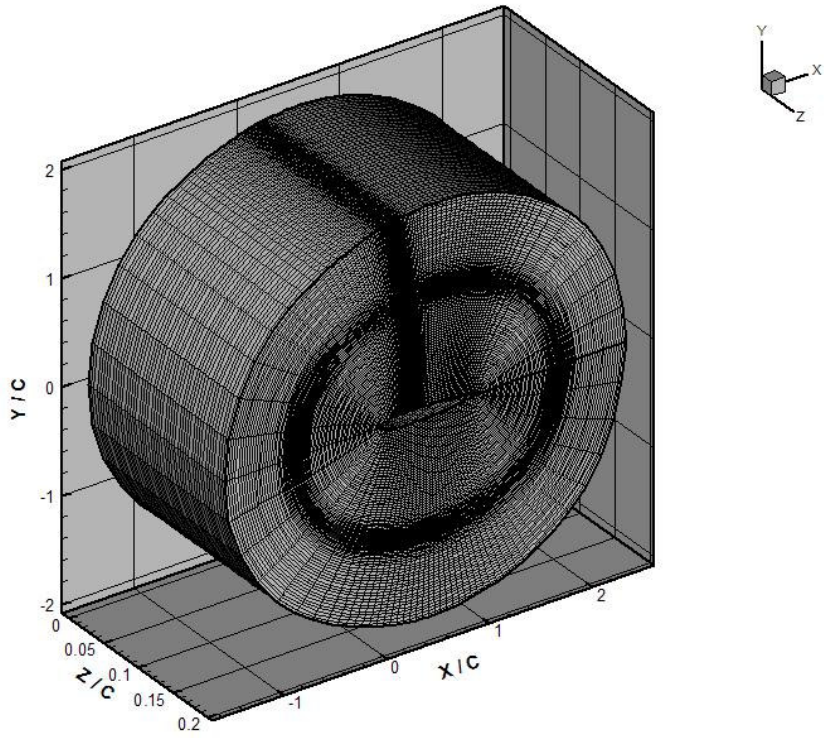


**Figure 5.59** C-mesh around NACA0012 profile with clustering both in $\xi$ and $\eta$ directions

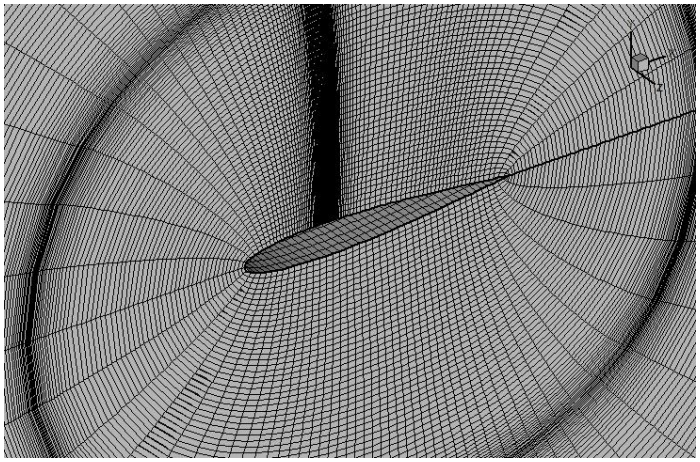

**Figure 5.60** Enlarged view of Figure 5.59

### 5.4.4 C-Mesh around a Tapered Airfoil

In all three-dimensional cases given in this thesis, output file of HYPERGEN2 is used as an input file of HYPERGEN3, that is, input file of HYPERGEN3 includes two-dimensional surface point distribution. Therefore, if three-dimensional mesh around airfoils having tapered three-dimensional geometry is desired, it can not be generated by using output file of HYPERGEN2. Three-dimensional surface point distribution should be used as an input for this purpose. As an example to this situation, three-dimensional surface point distribution of a tapered airfoil profile, whose cross-section is NACA0012 profile, is created and used as an input file of HYPERGEN3 in order to generate three-dimensional C-mesh around this airfoil profile. In Figures 5.61 and 5.62, three-dimensional surface point distribution of a tapered airfoil and enlarged view of Figure 5.61 are given, respectively. In Figures 5.63 and 5.64, generated C-meshes around this profile and enlarged view of Figure 5.63 are given, respectively. For this application, marching distance and number of levels in the marching direction are selected as 0.02 units and 40, respectively.

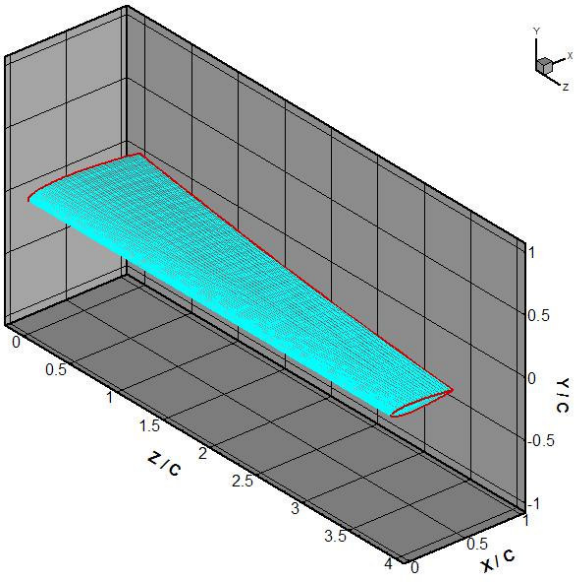


**Figure 5.61** 3-D Surface Point Distribution of a Tapered Airfoil

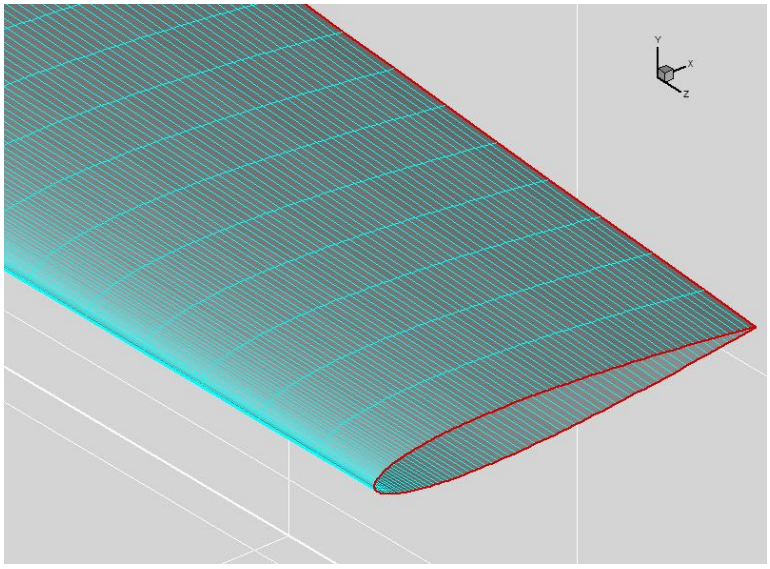


**Figure 5.62** Enlarged view of Figure 5.61

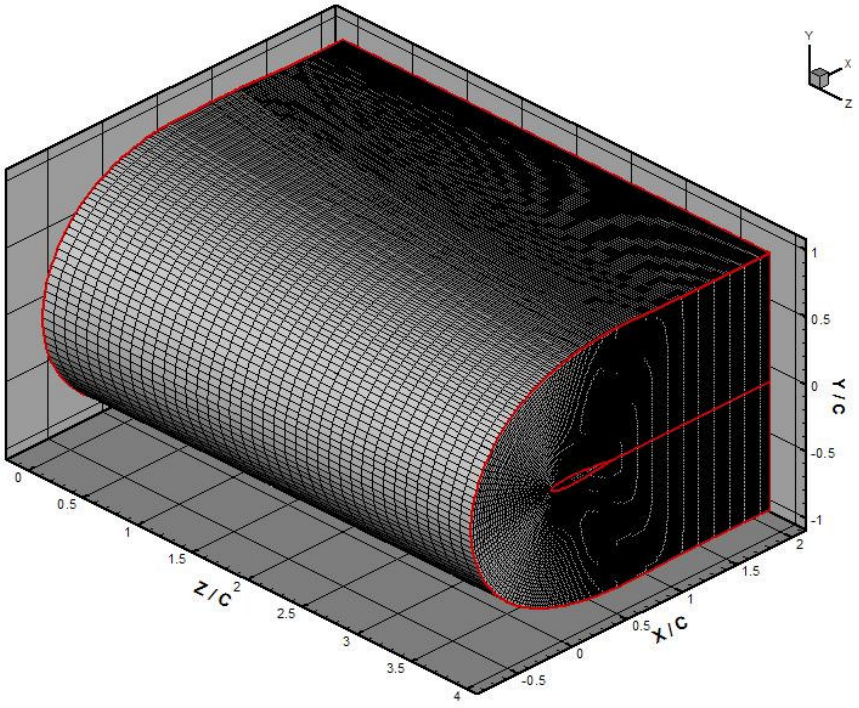


**Figure 5.63** 3-D C-mesh around a Tapered Airfoil

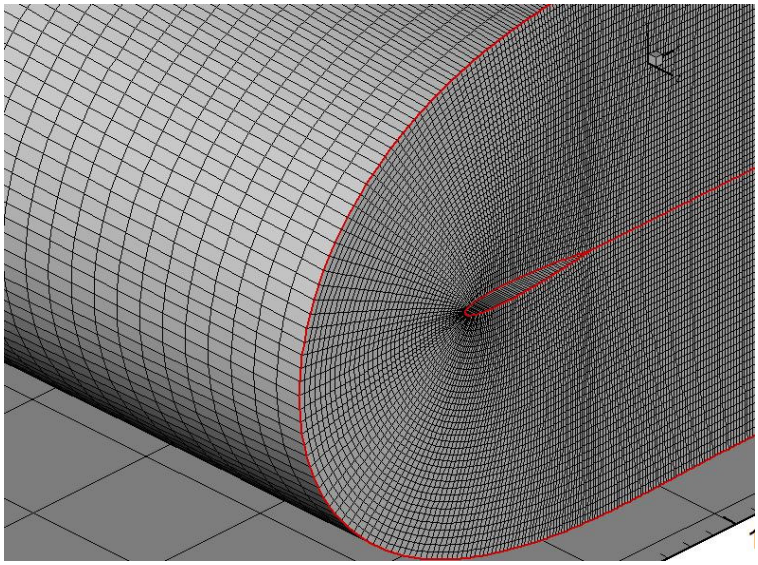

**Figure 5.64** Enlarged view of Figure 5.63

# CHAPTER 6

## CONCLUSION

In this study, two dimensional hyperbolic grid generation technique is presented and extended to three dimensions. In order to generate two and three dimensional grids, a computer program is written using finite difference method. By the application of grid control techniques, high quality two dimensional grids are generated. Two dimensional hyperbolic grid generation technique is applied to the concave and convex geometries and the results show the robustness of the code. In addition to these, clustering procedure is added to the code and adjusted to be used without much effort by the user.

The output of the two dimensional hyperbolic grid generation code hypergen2 is used as the input to the three dimensional hyperbolic grid generation program hypergen3. This code is used in order to generate three dimensional grids for different geometries.

For convex and concave regions, a different scheme should be used in order to eliminate the oscillations generated from odd-even decouplings. In this study, however, two dimensional grids is added on top of each other and by this way, three dimensional grids are generated for concave and convex geometries. The robustness of this code can be improved by using upwind scheme.

As a further study, hyperbolic grid generation technique can be used in order to generate grids for multiple zone geometries. This can be accomplished by adding an angle control source term to the equations and using a new algorithm for computing the volume source term and using shooting techniques as suggested in Reference [13].

## REFERENCES

[1] Klaus A. Hoffmann, Steve T. Chiang, “Computational Fluid Dynamics Volume I”, Wichita, Kansas, August 2000

[2] J. F. Thompson, Z. U. Warsi and C. W. Mastin, “Numerical Grid Generation Foundations and Applications“ , Elsevier 1985

[3] D. C. Ives, “A Modern Look at Conformal Mapping Including Multiply Connected Regions”, AIAA Journal, 14, (1976), pp. 1006-1011

[4] O. Hübner, “The Newton Method for Solving the Theodorsen Equation”, Journal of Computational Applied Mathematics, 14, (1986), pp. 19-30

[5] D. I. Meiron, S. A. Orszag, and M. Israeli, “Applications of Numerical Conformal Mapping”, Journal of Computational Physics, 40, (1981), pp. 345-360

[6] Chu, W. H. , “Development of a General Finite Difference Approximation for a General Domain”, Journal of Computational Physics, 8 , (1971) , pp. 392-408

[7] C. W. Mastin and J. F. Thompson, “Transformation of Three Dimensional Regions onto Rectangular Regions by Elliptic Systems, Numer. Math., 29, (1978), pp. 397-407

[8] P. D. Sparis, “A Method for Generating Boundary Orthogonal Curvilinear Coordinate Systems Using a Biharmonic Equations”, Journal of Computational Physics, 61, (1985), pp. 445-462

[9] A. Graves, “Application of a Numerical Orthogonal Coordinates Generator to Axisymmetric Blunt Bodies” , NASA TM 80131, Oct. 1979

[10] D. McNally, “Fortran Program for Generating a Two-Dimensional Orthogonal Mesh Between Two Arbitrary Boundaries”, NASA TN D-6766, May 1972

[11] Joseph L. Steger, and Denny S. Chausee, “Generation of Body Fitted Coordinates Using Hyperbolic Partial Differential Equations”, SIAM Journal of Scientific and Statistical Computing, Volume I, Number 4, December 1980, pp 431-437

[12] D. W. Kinsey and T. J. Barth, “Description of a Hyperbolic Grid Generation Procedure for Arbitrary Two-Dimensional Bodies” AFWAL TM 84-191-FIMM, Wright Aeronautics Labarotory, Wright Petterson-AFB, Ohio 1984

[13] J. R. Cordova and T. J. Barth, “Grid Generation for General 2-D Regions Using Hyperbolic Equations”, AIAA Paper 88-0520 F, 1988

[14] J. L. Steger and Y. M. Rizk , “Generation of Three Dimensional Body-Fitted Coordinates Using Hyperbolic Partial Differential Equations”, NASA TM 86753 , NASA-Ames Research Center, Motted Field, California, 1985

[15] W. M. Chan and J. L. Steger, “Enhancements of a Three Dimensional Hyperbolic Grid Generation Scheme“, Applied Mathematics and Computation, Vol. 51, No. 1 , 1992 , pp. 181-205

[16] Steger, J. L. and Sorenson, R. L. , “Use of Hyperbolic Partial Differential Equations to Generate Body Fitted Coordinates”, NASA CP-2166, 1980

[17] S. Anil Lal, B. V. S. S. S. Prasad and N. Sitaram, “Geometry-Based Hyperbolic Grid Generation for Computational Fluid Dynamics” AIAA Journal, Vol. 39, No. 8, Technical Notes, August 2001, pp 1631- 1633

[18] Gökhan Durmuş, “Three Dimensional Hyperbolic Grid Generation” M.S Thesis of The Department of Aeronautical Engineering of Middle East Technical University, September 1998, pp 47-48

# APPENDIX A

## DESCRIPTION AND FLOW CHART OF TWO DIMENSIONAL HYPERBOLIC GRID GENERATION PROGRAM CODE, HYPERGEN2

The two dimensional grid generation computer code, hypergen2 has three parts which are preprocessor, processor and postprocessor.

In preprocessor part, three input files are read. These are profil.dat, input.dat and input_cluster.txt. Profil.dat file has coordinates of geometry of profile and input.dat file has the values of lots of parameters used in grid generation program. These parameters define mesh type, maximum marching level, surface clustering coefficient, volume scaling coefficient, fourth order damping coefficient, initial marching distance, wake length, number of points at wake, coefficient which determines clustering in $\xi$, clustering in $\eta$ or clustering in both directions. Input_cluster.txt has parameters used for clustering in $\xi$ direction. This file is read if clustering in $\xi$ direction is desired.

In processor part, two dimensional hyperbolic grid generation technique described in thesis is applied, that is all linearized equations derived for two dimensional grid generation are solved here. This part includes formation of matrices, application of boundary conditions, smoothing procedure, grid control and solution of the system, i.e. finding grid coordinates. There are two main subroutines, and other several subroutines in these main subroutines, in processor part of the program. These are BTDMS and body_cluster subroutines used for solving block tridiagonal matrix system and clustering profile in $\xi$

direction respectively. Body_cluster is used if ξ clustering is needed and input file of this subroutine is input_cluster.txt as explained before.

In postprocessor part, coordinates of all grid points are written on a file which is prepared according to TECPLOT format. So, after running the program, an output file is created which can be opened in TECPLOT. All application results given in this thesis are TECPLOT outputs.

Parameters that should be given by the user in Input.dat file are explained below.

***jm*** : Maximum marching level normal to profile
***scc*** : Surface clustering coefficient
***vsc*** : Volume scaling coefficient which can take 1 as maximum value
***cleta*** : Clustering in η direction. If clustering in η direction is desired, cleta shold be 1 and if clustering in η direction is not desired, cleta should be 0.
***clxi*** : Clustering in ξ direction. If clustering in ξ direction is desired, clxi should be 1 and if clustering in ξ direction is not desired, clxi should be 0.
***dc*** : Fourth order damping coefficient. It can take 0.125 as maximum value due to stability condition.
***mt*** : Mesh type. If C-type grid is desired, mt should be 1 and if O-type grid is desired, mt should be 0. Actually, it is the control variable which adjusts the boundary conditions.
***convex*** : Used if profile has convex corner. If there is a convex corner, this parameter should be 1, if not it should be 0. This parameter is used if O-type grid is chosen, that is if mt = 0.
***cx*** : Power of sine function related with convex corner procedure. So, this parameter is used if convex = 1 and mt = 0.
***deltas(1)*** : Initial marching distance

***js*** : Used if cleta equals to 1. This parameter defines level to be stretched in η direction.

***intj*** : Used if cleta equals to 1. This parameter defines interval of stretching in η direction.

***sj*** : Used if cleta equals to 1. This parameter defines the value of stretching at the required level in η direction, that is the minimum marching distance at clustered regions.

***wl*** : This parameter defines wake length and of course it is used if mt = 1 (C-type grid).

***np*** : Number of points at wake.

Parameters that should be given in the Input_cluster.txt file by the user are explained below.

Clustering terminals, spacing and the desired number of points should be given. In this file 0 and 1 should be given as clustering terminals in all ξ clustering cases since the arclength is normalized and 0 and 1 defines the start and end point of the geometry profile. Also, spacings at these terminals should be equal. The terminals between 0 and 1 and the spacings are given by the user. Since this input file is used for clustering in ξ direction, the spacings given for the terminals between 0 and 1 should be less than the spacings given for 0 and 1 terminals.

Output of this two dimensional hyperbolic grid generation code, Hypergen2, is also an input for three dimensional hyperbolic grid generation program code, Hypergen3. Flow chart of the computer program Hypergen2 is given in Figure A.1.

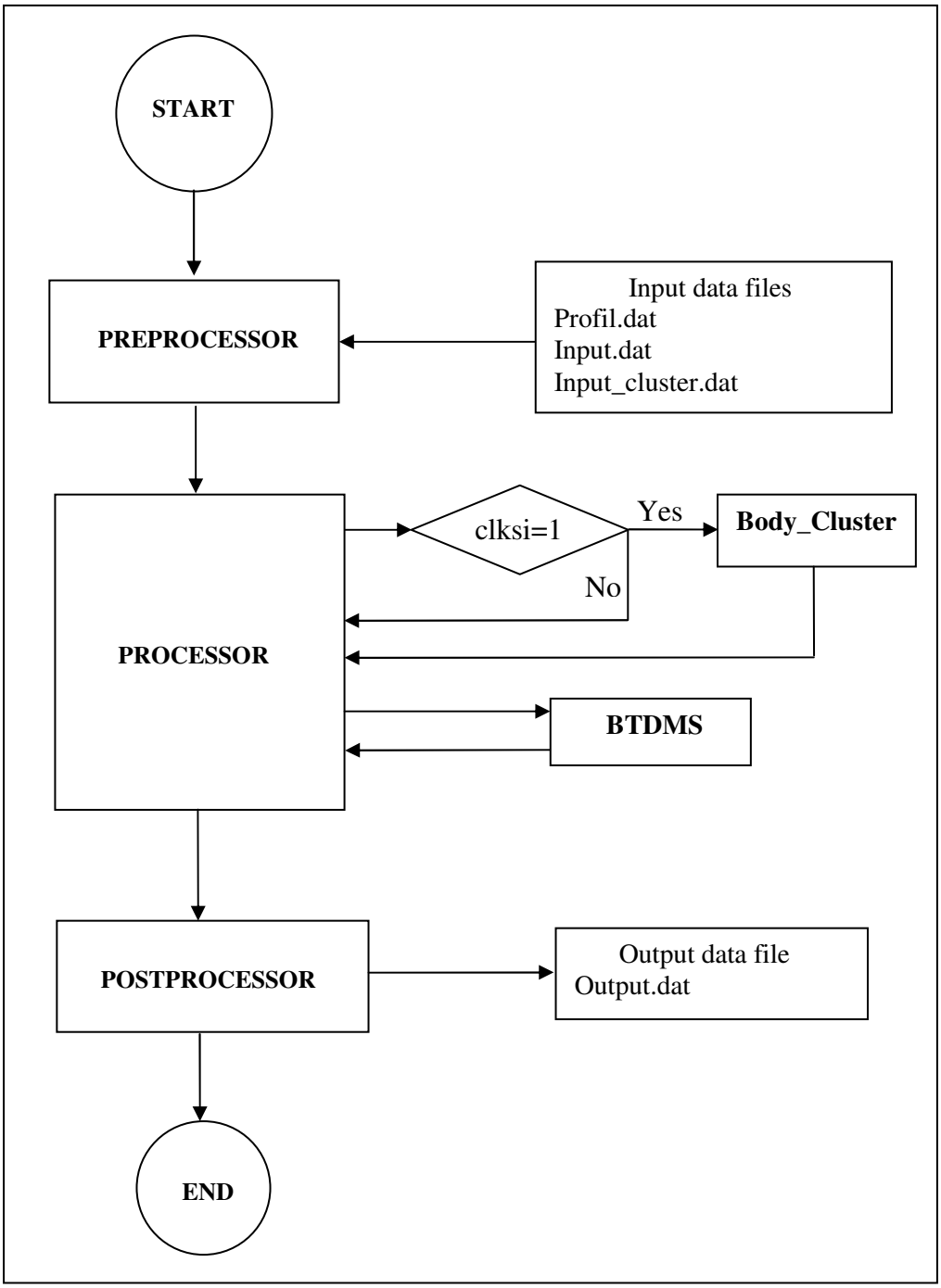


**Figure A.1** Flow Chart of the Computer Program HYPERGEN2

# APPENDIX B

## DESCRIPTION AND FLOW CHART OF THREE DIMENSIONAL HYPERBOLIC GRID GENERATION PROGRAM CODE, HYPERGEN3

The three dimensional hyperbolic grid generation program code, hypergen3 has three parts which are preprocessor, processor and postprocessor.

In preprocessor part, input file given by the user is read. This input file is output file of the hypergen2 and given without any changes. The parameters which define mesh type, maximum marching level in $\eta$ direction, surface clustering coefficient, volume scaling coefficient, fourth order damping coefficient, initial marching distance in $\eta$ direction, clustering in $\xi$ direction, clustering in $\eta$ or clustering in both directions are used in hypergen2. Therefore, there is no need any parameter which controls the grid in hypergen3. Grid control can be done in two dimensional grids and the generated three dimensional grids have the properties of two dimensional grids since output file of hypergen2 is the input file of the hypergen3.

In processor part, three dimensional hyperbolic grid generation technique described in thesis is applied, that is linearized form of equations, derived from approximate factorization, is solved here. This part includes the calculation of the terms in $\eta$ sweep and $\xi$ sweep and calculation of derivatives in $\xi$, $\eta$ and $\zeta$ directions. Also, this part includes formation of matrices, application of boundary conditions for $\eta$ and $\xi$ sweep and solution of the system, that is finding coordinates of grids. Also, there is a control parameter for convex

profiles. In this part, there is only one subroutine which is used for solving block tridiagonal matrix system.

In postprocessor part, coordinates of all grid points are written on a file which is prepared according to TECPLOT format. So, after running the program, an output file is created which can be opened in TECPLOT.

Flow chart of three dimensional hyperbolic grid generation program code, Hypergen3 is given in Figure B.1.

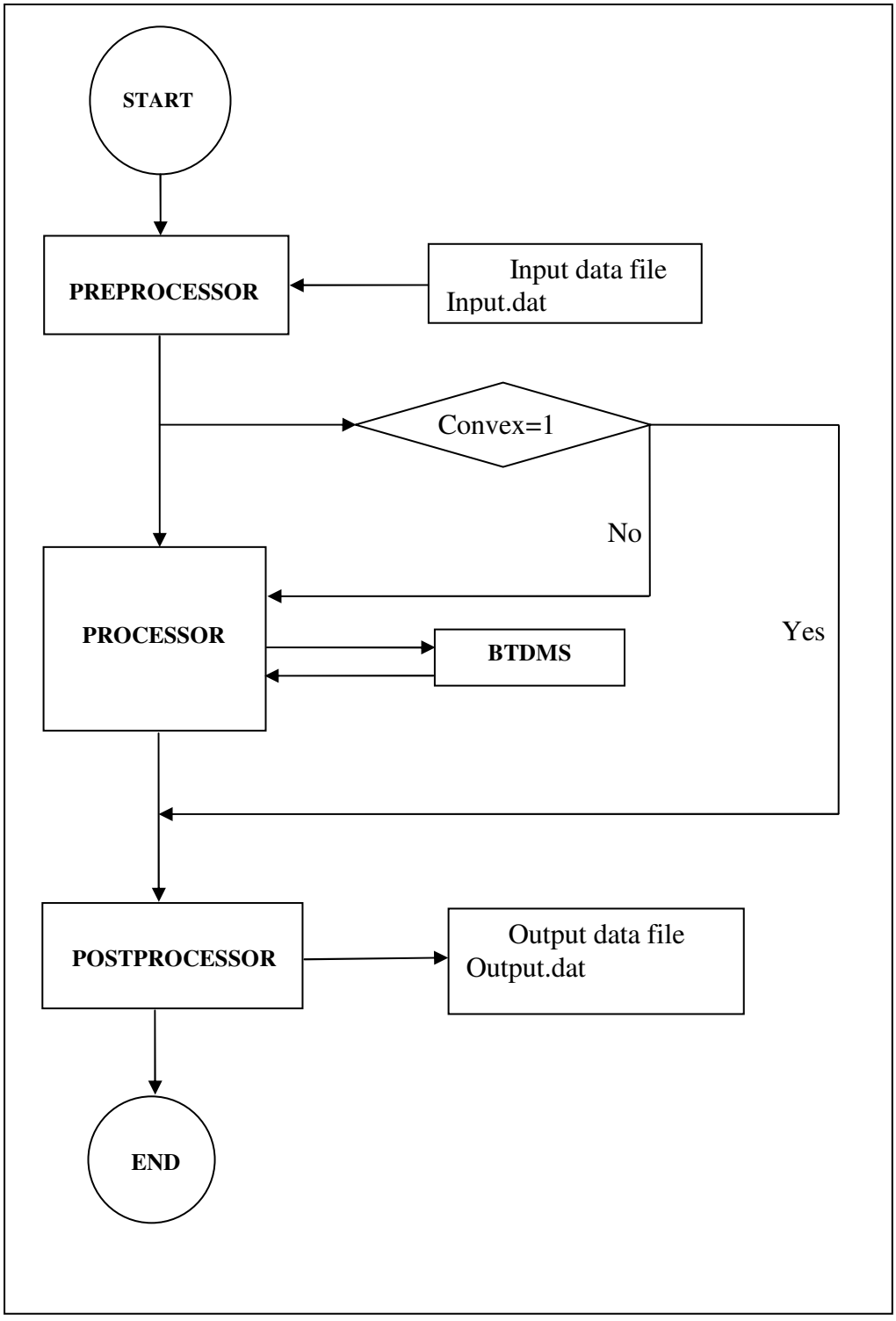


**Figure B.1** Flow Chart of the Computer Program HYPERGEN3