DETERMINATION OF COMPUTATIONAL DOMAIN BOUNDARIES FOR VISCOUS FLOW AROUND TWO DIMENSIONAL BODIES

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

BY

MUSTAFA MAZHAR BASA

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

NOVEMBER 2006

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan Özgen
Director

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

Prof. Dr. Erdal Çokça
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.

Assoc. Prof. Dr. İsmail Aydın
Supervisor

Examining Committee Members

Prof. Dr. Cahit Çıray	(METU, AEE)	
Prof. Dr. Mustafa Göğüş	(METU, CE)	
Assoc. Prof. Dr. İsmail Aydın	(METU, CE)	
Assoc. Prof. Dr. Zafer Bozkuş	(METU, CE)	
Assist. Prof. Dr. Burcu Altan Sakarya	(METU, CE)	

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 : Mustafa Mazhar Basa

Signature :

ABSTRACT

DETERMINATION OF COMPUTATIONAL BOUNDARIES
FOR VISCOUS FLOW AROUND TWO DIMENSIONAL BODIES

Basa, Mustafa Mazhar
M.Sc., Department of Civil Engineering
Supervisor: Assoc. Prof. Dr. İsmail Aydın

November 2006, 36 pages

Borders of flow field around immersed bodies can be extended to long distances since there are no physical boundaries. In computational practice however, the flow domain must be restricted to a reasonable size by imposing appropriate boundary conditions at the edges of the computational space. In this thesis work, streamlines obtained from potential flow solution in a relatively large spatial domain are utilized to specify the boundaries and boundary conditions for a more restricted computational domain to be used for detailed viscous flow computations. A grid generation code is adopted for generation of unstructured triangular grid systems for domains involving multiple immersed bodies of any shape at arbitrary orientations such as a group of tall buildings in horizontal plane. Finite volume method is used in the solution of Laplace equation for the stream function. A deformation modulus is introduced as a probe parameter to aid locating the viscous flow boundaries. The computer code acts as a preprocessor for viscous flow computations, specifying the computational boundaries, the boundary conditions and generating the computational grid.

Keywords: Unstructured Grid, Potential Flow, Finite Volume Method, Viscous Boundary

ÖZ

2 BOYUTLU CİSİMLER ETRAFINDAKİ VİZKOZ AKIM İÇİN HESAPLAMA ALANI SINIRLARININ BULUNMASI

Basa, Mustafa Mazhar
Yüksek Lisans, İnşaat Mühendisliği Bölümü
Danışman: Assoc. Prof. Dr. İsmail Aydın

Kasım 2006, 36 sayfa

Batık cisimler etrafındaki akımın sınırları, fiziksel sınırlar olmadığından, uzak mesafelere kadar uzatılabilir. Fakat, sayısal çözümler söz konusu olduğunda, hesaplama alanının kenarlarına uygun sınır koşulları uygulanarak, akım alanının makul bir noktada sınırlanması gerekmektedir. Bu tez çalışmasında, detaylı vizkoz çözümlerinde kullanılacak bir hesaplama alanı oluşturulması amacıyla, bu alanın sınırlarının ve sınır koşullarının belirlenmesi için, potansiyel akımın çok daha geniş bir uzaysal alanda çözümünden elde edilen akım çizgileri kullanılır. Yatay düzlemde yüksek bina grupları gibi birden fazla, farklı şekil ve konumlardaki batık cisimlerden oluşan akım alanları için yapısız üçgen ağ sistemi oluşturan bir yazılım uyarlanmıştır. Akım çizgisi fonksiyonu için Laplace denkleminin çözümünde sonlu hacim yöntemi kullanılır. Viskoz akım sınırlarının tespit edilebilmesi için, bir deformasyon modülü tanımlanır. Bilgisayar programı viskoz çözümler için, hesaplama sınırlarını ve sınır koşullarını belirleyen ve hesaplama ağını oluşturan bir ön işlemci olarak çalışmaktadır.

Anahtar Kelimeler: Potansiyel akım, Yapısal olmayan ağlar, Sonlu Hacim Yöntemi, viskoz akım sınırı

Shoot for the stars,
even if you miss you'll be among the stars.

ACKNOWLEDGEMENTS

The author wishes to dedicate this thesis work to the memory of his father, Ethem Basa.

The author wants to thank to his mother Melahat, his sister Ayşe and his beloved Burcu, for their support, patience and understanding.

The author wishes to express his deepest gratitude to his supervisor Assoc. Prof. Dr. İsmail Aydın for his guidance, advice, inspiration, encouragement.

The author also wishes to express his deepest gratitude to his managers Alper Uzman, Kemal Güleryüz and Necdet Demir in GULERMAK Heavy Ind. Const. &Contr. Co. Ltd; Yusuf Erbay and Doğan Güneş from EKATEK Eng. Co. Ltd. for their support and understanding during his work in these companies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

A	Area
A_{abc}	Area of triangle *abc*
A_{CV}	Area of control volume
C.S.	Control surface
C.V.	Control volume
D	Diameter of circumbsribing circle
D_{DM}	Extend of deformation modulus contours or viscous boundary
DM	Deformation modulus
E	First derivative of stream function in x direction
E_i	First derivative of stream function in x direction at the i^{th} grid node
E_k	First derivative of stream function in x direction at the k^{th} control surface
F	First derivative of stream function in y direction at the i^{th} grid node
F_i	First derivative of stream function in y direction at the i^{th} grid node
F_k	First derivative of stream function in x direction at the k^{th} control surface
L_P	Width and length of the potential flow domain
t	Time
u	Velocity component in horizontal direction
U_m	Approach velocity of uniform flow
v	Velocity component in vertical direction
Δ_{max}	Maximum grid size
Δ_{min}	Minimum grid size

Δt	Time step
Δx_k	Difference in x coordinates of starting and end points of k^{th} control surface
Δy_k	Difference in y coordinates of starting and end points of k^{th} control surface
α	Diffusivity parameter
ζ	Vorticity
Ψ	Stream function for potential flow
Ψ_{max}	Maximum value of stream function applied as upper free stream boundary condition
Ψ^n	Value of stream function in the n^{th} iteration
ρ	Fluid density
$\forall$	Volume of control volume

CHAPTER 1

INTRODUCTION

1.1 Approaches in the Analysis of Fluid Flow Problems

Until the middle of the twentieth century, fluid dynamics have been operating in the two-approach world of theory and experiment. However, the advent of high-speed digital computers combined with the development of accurate numerical algorithms for solving problems on these computers has revolutionized the way of studying and practicing fluid dynamics. It has introduced a fundamentally important new third approach in fluid dynamics - the approach of Computational Fluid Dynamics (CFD). Today, computational fluid dynamics is an equal partner with pure theory and pure experiment in the analysis and solution of fluid dynamic problems (Anderson,1995).

In theoretical fluid dynamics, analytical methods are used in the solution of problems. These analytical methods provide quick, closed form solutions, but they can treat only simplified configurations. In order to overcome this shortage, experimental fluid dynamics or namely wind tunnel testing is utilized. In experimentation, actual configurations are used and complete flow field data can be produced. However, wind tunnel has its own limitations on size and flow velocity. Other two important drawbacks of wind tunnel testing are high-energy consumption rates and engineering costs of model and adequate tunnels (Chapman, 1979 and Kutler, 1985).

Relative to analytical techniques, computational procedures require very few restrictive assumptions so they can be used to treat complicated configurations. Moreover, computational procedures have fewer Reynolds number limitations that result in complete surface and external flow field definition, which is extremely difficult in experimental fluid dynamics. The details of these comparisons may be found in references (Chapman, 1979 and Kutler, 1985).

As a research tool, computational fluid dynamic results are analogous to wind tunnel results obtained in a laboratory. This analogy consequences in a tool that can be used for numerical experiments. These numerical experiments, carried out in parallel with physical experiments, can be used to interpret physical phenomena. However, as a design tool, computational fluid dynamics results have become lusty standards for various areas of engineering (Anderson, 1995 and Dogru, 2000).

However, it should be noted that computational fluid dynamics nicely and synergistically complements the other two approaches of pure theory and pure experiment, but it will never replace either of these approaches. The advancement of fluid dynamics rest upon a proper balance of all three approaches, with computational fluid dynamics helping to interpret and understand the results of theory and experiment, and vice versa (Anderson,1995).

1.2 Numerical Methods Used in CFD

Numerous restrictions apply to the methods used in CFD. First of all, mathematical model that will be used in the calculations should be decided according to the desired level of approximation. This approximation is a consequence of available computing power and validated numerical schemes in hand. After that, discretization models for equations (equation discretization) and for solution domain (space discretization) should be decided.

Space discretization consists of setting up a mesh or a grid by which continuum of space is replaced by finite number of points where numerical values of the variables will have to be determined. Thus, it is clear that the accuracy of a numerical

approximation will be directly dependent on the size of the mesh. That is; the better the discretized space approaches to continuum, the better the numerical scheme (Hirsch, 1989). There are two main types of grids used in CFD, namely structured and unstructured grids.

First step in equation discretization is the selection of a numerical scheme among finite difference, finite element and finite volume methods. The basis of all these numerical methods consists of transformation of the physical equations into an algebraic, linear or non-linear system of equations. For time dependent problems an intermediate step is obtained, namely a system of ordinary differential equations in time, which through an integration scheme in time, will lead to an algebraic system for the unknowns at a given time level (Telçeker, 1992).

It is important to distinguish time-dependent problems from time-independent ones. Because the numerical schemes associated with time-dependent problems differ from the ones used in time-independent problems. Some examples of such problems are transient flow problems or the ones connected to time-varying boundary conditions. For these problems, time-dependent mathematical models must be used to preserve the time accuracy of the solution (Hirsch, 1989).

According to the solution methods, there are two families; namely explicit and implicit methods. In explicit methods, the matrix of the unknown variables at the new time level as a diagonal matrix and the right-hand side of the system is dependent only on the flow variables at the previous times. This leads to a trivial matrix inversion with minimal number of arithmetic operations at each step. Whereas in implicit methods, the matrix to be inverted is not diagonal since more than one set of variables are unknown at the same time level.

However, in many cases the solution of the matrix is not so cumbersome because of its block tridiagonal or block bidiagonal structure, only the number of arithmetic operations increase at each time step when compared with explicit methods. But

implicit methods are advantageous when stability and convergence criteria are considered.

Two families of methods for solving algebraic systems can be distinguished; direct and iterative methods. The former can be defined as leading to the solution of a linear system in one step, while the latter will require many iterative steps (Hirsch, 1989).

1.3 Scope of the Study

Main goal of this thesis is to develop a computational tool to determine the location of the outer computational boundaries required for viscous flow solutions around immersed bodies in 2D domains. The outer boundaries will be defined as the location where there is a transition in the characteristics of fluid flow from viscous to inviscid, which means that vorticity approaches zero and shear stresses vanish.

In order to determine the location of the transition from viscous to inviscid flow, an inviscid flow solution for a sufficiently large domain will be obtained by solving the potential flow equations using finite volume method over unstructured grid system. Use of unstructured grid system is preferred to be able to study any number of bodies of any shape at arbitrary orientations. After potential flow solution, influence of the immersed body on the surrounding streamline patterns will be investigated by defining a probe parameter. The outer boundary streamlines for the viscous flow domain will be identified by setting an appropriate value for the probe parameter. The viscous flow outer boundary streamline will be the one which is the first almost-straight streamline closest to the immersed body and after which all the streamlines are nearly parallel to each other.

The values of the flow variables such as velocity and pressure over the viscous boundary obtained from the inviscid solution will also be valid as boundary values for the viscous solution of the interior domain. This approach of specifying the boundaries of viscous flow domain from the potential flow solution will result in

significant reduction in size of the computational domain for viscous flow and therefore savings in the computational costs for viscous flow.

In chapter 2, general information about unstructured grid generation will be given, including information about DELAUNDO program, which will be used as unstructured grid generator in this study. In chapter 3, governing equations of potential flow and numerical solution technique will be presented. Results and discussions will be given in chapter 4. Main conclusions of this study will be presented in chapter 5. Some examples and explanations for possible use of the code developed in this study for educational purposes are presented in the Appendices.

CHAPTER 2

UNSTRUCTURED GRID GENERATION

2.1 General

There are two fundamental classes of grid popular in the numerical solution of Computational Fluid Dynamics problems: structured and unstructured. They differ in the way in which the mesh points are locally organized. In the most general sense, this means that if the local organization of the grid points and the form of the grid cells do not depend on their position but are defined by a general rule, the mesh is considered as structured. When the connection of the neighboring grid nodes varies from point to point, the mesh is called unstructured. As a result, in the structured case the connectivity of the grid is implicitly taken into account, while the connectivity of unstructured grids must be explicitly described by an appropriate data structure procedure.

2.2 Structured Grids vs. Unstructured Grids

Structured grids have a set connectivity that defines the topology. All grid points have an index (*i, j*) associated with them. Their positions can be stored in a 2-D array for 2-D domains. The connectivity is straightforward, as every interior node (*i, j*) is connected to four neighbors. More specifically, two in the *i* direction (being *i-1*, *j* and *i+1, j*) and two in the *j* direction (which are *i, j-1* and *i, j+1*). This is shown graphically in Fig.2.1. From this figure, the structured nature of these grids is clear. Unstructured grids, on the other hand, have no inherent structure, that is, there is no set connectivity. Any given grid point is connected to an arbitrary number of

neighbors. This is shown in Fig. 2.2. The unordered nature of unstructured grids lead to a number of advantages and disadvantages compared to the structured approach. These are summarized below (Wilkinson, 1992).

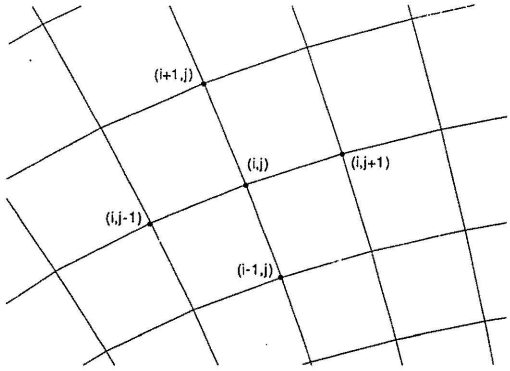


Fig. 2.1 Connectivity in a structured grid system.

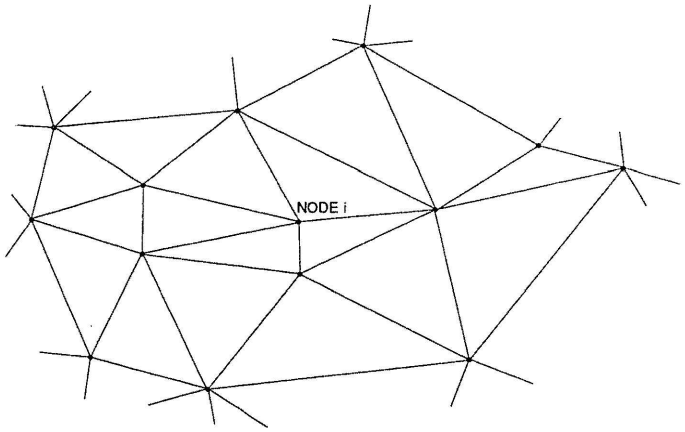


Fig. 2.2 Connectivity in an unstructured grid system.

2.2.1 Advantages of Unstructured Grids

a) **Ease of grid adaptation:** To add extra points in particular regions of interest in a structured grid, an extra line must be inserted in both grid directions. This introduces unwanted points elsewhere in the grid. With an unstructured grid, points can be added locally, thus minimizing the addition of superfluous points.

b) **Gridding complex geometries are much easier:** As there is no rigid data structure, almost any conceivable geometry can be accommodated by the data structure.

2.2.2 Disadvantages of Unstructured Grids

a) **The memory requirements are greater:** Due to the extra variables needed to perform the indirect addressing, memory requirements can be a factor of 4 to 4.5 times greater in single precision, but only around 2.5 times greater for double precision.

b) **Longer run times are required:** With the extra effort required for the indirect addressing and the extra cells and edges, CPU time increases about 4 to 4.5 fold (for the same number of points).

c) **Generating an unstructured grid is more involved than generating structured ones:** Although a grid can be generated by applying Delaunay triangulation to a structured grid, this method is limited mainly to single element geometries. The coding of specialized unstructured grid generators can be more complex than coding a structured gridder.

d) **The application of boundary conditions is not straight forward:** As the grid structure is arbitrary, extrapolation at boundaries becomes more difficult.

e) **Accuracy of unstructured schemes is slightly inferior to their structured counterparts:** This is partially due to the difficulty of developing accurate, but computationally inexpensive damping models. Although this is a point of contention in the literature, work to date suggests that the accuracy of unstructured solvers is inferior to structured solvers.

Thus, although unstructured grids require more memory and cpu time, they offer a

number of advantages over the structured grid. It is in this light that the unstructured grid is used in this thesis.

2.3 Methods for Unstructured Grid Generation

Unstructured grids can be obtained with cells of arbitrary shape, but are generally composed of tetrahedrons (triangles in two dimensions). There are fundamentally, three approaches to the generation of unstructured grids: octree methods, Delaunay procedures, and advancing-front techniques (Baker, 1990).

2.3.1 Octree Approach

In the octree approach the region is first covered by a regular Cartesian grid of cubic cells (squares in two dimensions). Then the cubes containing segments of the domain surface are recursively subdivided into eight cubes (four squares in two dimensions) until the desired resolution is reached. The cells intersecting the body surfaces are formed into irregular polygonal cells. The grid generated by this octree approach is not considered as the final one, but serves to simplify the geometry of the final grid, which is commonly composed of tetrahedral (or triangular) cells built from the polygonal cells and the remaining cubes.

The main drawback of the octree approach is the inability to match a prescribed boundary surface grid, so the grid on the surface is not constructed beforehand as desired but is derived from the irregular volume cells that intersect the surface. Another drawback of this grid is its rapid variation in size near the boundary. In addition, since each surface cell is generated the intersection of a hexahedron with the boundary there arise problems in controlling the variation of the surface cell size and shape.

2.3.2 Delaunay Approach

The Delaunay approach connects neighboring points (of some previously specified set of nodes in the region) to form tetrahedral cells in such a way that the circumsphere through the four vertices of a tetrahedral cell does not contain any other point. The points can be generated in two ways; they can be defined at the start

by some technique or they can be inserted within the tetrahedra as they are created, starting with very coarse elements connecting boundary points and continuing until the element size criteria are satisfied. In the latter case a new Delaunay triangulation is constructed at every step using usually Watson's and Rebay's incremental algorithms.

The major drawback of the Delaunay approach is that it requires the insertion of additional boundary nodes, since the boundary cells may not become the boundary segments of the Delaunay volume cells. Either Delaunay criterion must be mitigated near the boundaries or boundary points must be added as necessary to avert breakthrough of the boundary.

2.3.3 Advancing-Front Techniques

In these techniques the grid is generated by building cells progressively one at a time and marching from the boundary into the volume by successively connecting new points to points on the front until all previously unmeshed space is filled with grid cells. Some provision must be made to keep the marching front from intersecting.

To find suitable vertices for the new cells is very difficult task in this approach, since significant searches must be made to adjust the new cells to the existing elements. Commonly, the marching directions for the advancing front must take into account the surface normals and also the adjacent surface points. A particular difficulty of this method occurs in the closing stage of the procedure, when the front folds over itself and the final vertices of the empty space are replaced by tetrahedral. Serious attention must also be paid to the marching step size, depending on the size of the front faces as well as the shape of the unfilled domain that is left.

Unstructured grids, after they have been completed, are generally smoothed by a Laplacian-type or other smoother to enhance their qualitative properties.

2.4 DELAUNDO Software

In this thesis work, open-source program DELAUNDO will be used for unstructured grid generation. This program was written as a part of Ph.D. thesis by J.-D. Müller in 1996. It produces 2D unstructured triangular meshes with the Frontal Delaunay Method (Frod). The triangulation algorithm employed is Bowyer's/Watson's using the circumcircle criterion (Müller, 1996).

2.4.1 Advantages of DELAUNDO

DELAUNDO is specially designed for flow around airfoils. In coincidence, domain of this thesis is very similar to this application, i.e. flow around submerged bodies with any arbitrary shape and orientation. General structure of DELAUNDO suits the purpose of this thesis very well. Besides it provides the ability to change a large variety of parameters that influence the grid quality that enables one to generate a grid with high quality, and change or adapt the grid easily to the needs of each specific application.

Another advantage of it is that it has a good implemented, well structured algorithm with lots of subroutines that is responsible for a small part of the grid generation process, and for this reason, compared to other non-commercial codes; it is a very fast grid generation tool with providing more control over the grid parameters.

2.4.2 Input Output Control Over DELAUNDO

There are two input files for DELAUNDO, one of which includes the parameters to control grid generation, quality, orientation, size, etc.; and the other includes data to define the geometry, i.e. interior and outer boundary nodes. In this study, output of DELAUNDO is modified in this study in order to be read directly by the potential flow solver, and also for TECPLOT CFD visualization program. Detailed description of these files and grid generation parameters together with examples are included in Appendices.

CHAPTER 3

GOVERNING EQUATIONS AND SOLUTION ALGORITHM

3.1 Stream Function and Streamlines

Steady, incompressible, 2-D planar flow represents one of the simplest types of flow. There are two velocity components, u and v, when the 2-D flow in the x-y plane is considered. The continuity equation reduces to,

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{3.1}$$

The velocity components can be written in terms of the stream function $\Psi(x, y)$,

$$u = \frac{\partial \psi}{\partial y} \qquad \text{and} \qquad v = -\frac{\partial \psi}{\partial x} \tag{3.2}$$

The continuity equation (Eqn. 3.1) is identically satisfied by definition of the stream function. Thus, the 2-D planar flow problem is simplified to determination of only one unknown function, $\Psi(x, y)$, rather than the two velocity components.

Another particular advantage of using the stream function is the fact that, lines along which Ψ is constant are streamlines as shown in Fig. 3.1.

3.2 Numerical Solution for Stream Function in Potential Flow

Vorticity in 2-D two-dimensional planar flow can be written in terms of stream function and set to zero for irrotational flows,

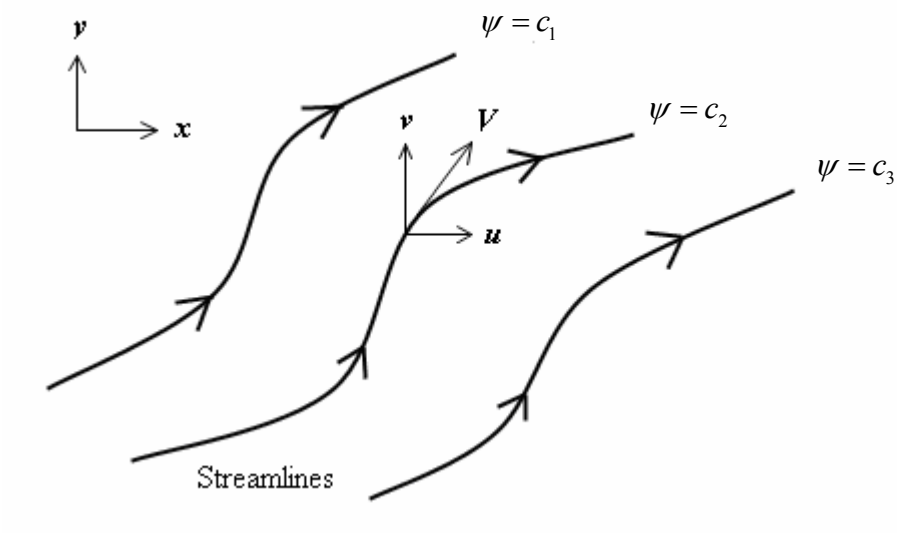


Fig. 3.1 Velocity components along a streamline in a flow field.

$$\zeta = -\left(\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2}\right) = 0 \quad \text{or} \quad \nabla^2 \psi = 0 \tag{3.3}$$

To facilitate an iterative solution of the above equation, a fictitious unsteady term is added,

$$\frac{\partial \psi}{\partial t} = \alpha \left(\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2}\right) \tag{3.4}$$

where α is the diffusivity parameter taken as unity. Mathematically Eqn. 3.4 can be classified as heat equation. Finite volume solution is formulated by integration over a control volume.

$$\int_{C.V.} \left(\frac{\partial \psi}{\partial t}\right) d\forall = \int_{C.V.} \left(\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2}\right) d\forall \tag{3.5}$$

Substituting $E = \dfrac{\partial \psi}{\partial x}$ and $F = \dfrac{\partial \psi}{\partial y}$,

$$\int_{C.V.} \left(\frac{\partial \psi}{\partial t}\right) dA = \int_{C.V.} \left(\frac{\partial E}{\partial x} + \frac{\partial F}{\partial y}\right) dA \tag{3.6}$$

where $d\forall = dA = dxdy$ assuming unit distance in z-direction. To evaluate the integrals of the spatial derivatives, Green's Theorem will be used.

$$\int_{C.V.}\left(\frac{\partial E}{\partial x}+\frac{\partial F}{\partial y}\right)dA = \oint_{C.S.} Edy - \oint_{C.S.} Fdx \tag{3.7}$$

Then, Eqn. 3.6 is integrated over the control volume and written as,

$$\frac{\partial \psi}{\partial t} A_{C.V.} = \oint_{C.S.} \left(Edy - Fdx\right) \tag{3.8}$$

where A_{cv} is the area of control volume in two dimensions. The integral equation is discretized for a control volume obtained from polygon of control points around an internal grid node (Fig. 3.2).

$$\psi^{n+1} = \psi^{n} + \frac{\Delta t}{A_{C.V.}}\left[\sum_{k=1}^{n}(E_k \Delta y_k - F_k \Delta x_k)\right] \tag{3.9}$$

Here k = 1…n indicates the control surfaces of the control volume. In the above equations, E_k and F_k stand for the x and y derivatives of the stream function over the k^{th} control surface. And the term $\left(E_k \Delta y_k - F_k \Delta x_k\right)$ is the net diffusive flux through the k^{th} control surface as seen in Fig. 3.2. The number of control surfaces for each control volume is equal to the number of triangles surrounding the grid point.

In order to calculate the fluxes at the control surfaces, first derivatives E_k and F_k must be evaluated at the control surfaces. For the control surface k shown in Fig. 3.3 between the points i and i +1, derivatives at the mid-point of this control surface denoted by E_k and F_k, are simply the average of the E and F at i and i +1.

$$E_k = \frac{E_i + E_{i+1}}{2} \qquad \text{and} \qquad F_k = \frac{F_i + F_{i+1}}{2} \tag{3.10}$$

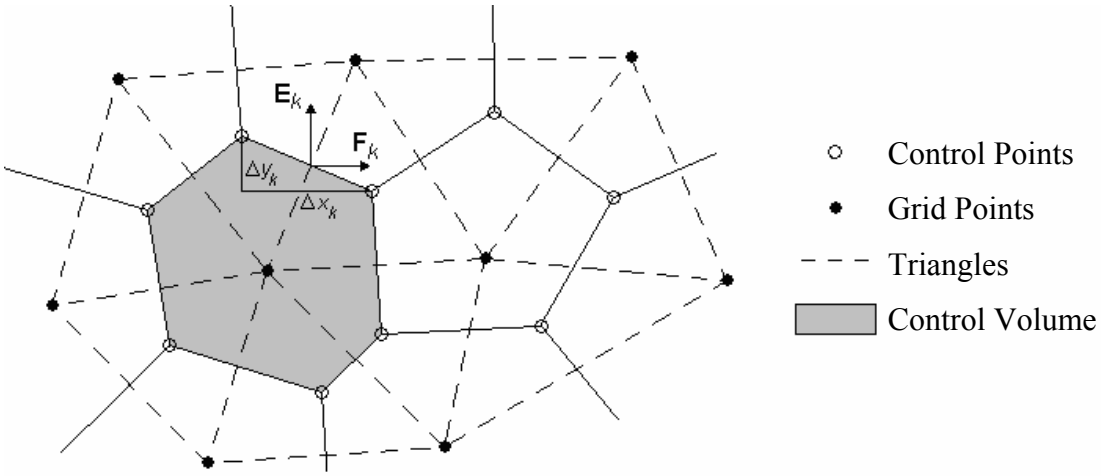


Fig. 3.2 Grid system and the control volume.

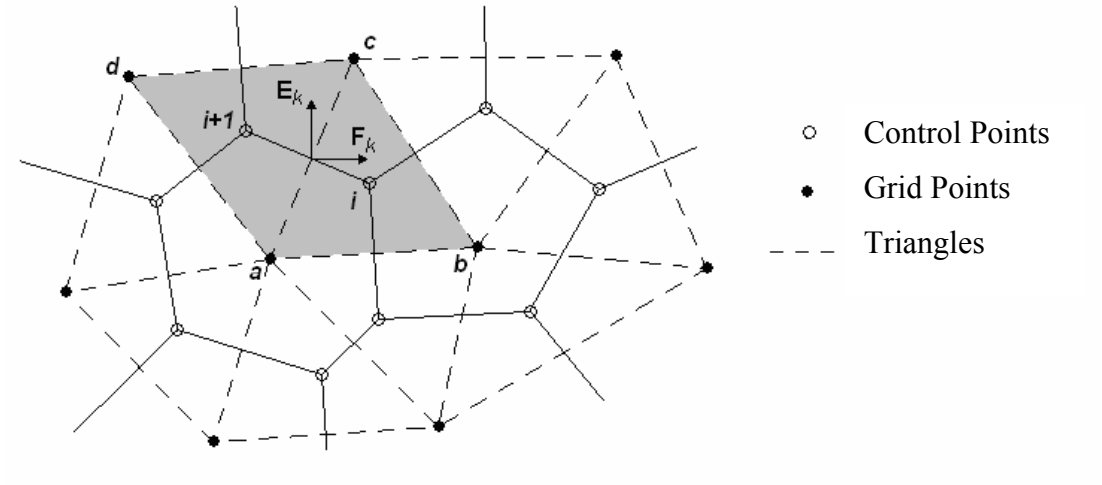


Fig. 3.3 Grid nodes and control points used in calculation of fluxes at control surfaces.

In order to evaluate the derivatives at centroids of triangles, integral averaging approach is used.

$$E_i = \int_{A_{abc}} \frac{\frac{\partial \psi}{\partial x} dA}{A_{abc}} = \frac{(\psi_a + \psi_b)(y_b - y_a) + (\psi_b + \psi_c)(y_c - y_b) + (\psi_c + \psi_a)(y_a - y_c)}{2A_{abc}} \quad (3.11)$$

$$F_i = \int_{A_{abc}} \frac{\frac{\partial \psi}{\partial y} dA}{A_{abc}} = \frac{(\psi_a + \psi_b)(x_b - x_a) + (\psi_b + \psi_c)(x_c - x_b) + (\psi_c + \psi_a)(x_a - x_c)}{2A_{abc}} \quad (3.12)$$

3.3 Boundary Conditions

There are interior and outer boundaries to describe the computational domain (Fig. 3.4). Outer boundaries are inflow, outflow and free-stream boundaries. Interior boundaries are wall boundaries of the immersed bodies. The outer boundaries of the potential flow domain are assumed to be far enough from the immersed bodies so that they are not influenced from the flow around the immersed bodies.

3.3.1 Inflow, Outflow and Free-Stream Boundaries

At the inflow and outflow sections, the flow is assumed to be uniform at maximum velocity.

$$U = U_{\mathrm{m}} \tag{3.13}$$

Then, the stream function values at the inflow and outflow boundaries are found by integration. The free-stream boundaries at the sides of the domain are fixed by defining straight streamlines. The streamline at y =0 is specified as

$$\psi_{(y=0)} = 0 \tag{3.14}$$

Then the inflow and outflow boundary values are obtained by integration

$$\psi_{(x=0,y)} = \psi_{(x=L_P,y)} = \int_0^{L_P} u dy = y U_m \tag{3.15}$$

Finally,

$$\psi_{(x,y=L_P)} = L_P U_m \tag{3.16}$$

3.3.2 Boundary Conditions on Solid Surfaces

The interior boundaries are solid boundaries across which there is no flow. Therefore interior boundaries are also streamlines with a constant stream function value. Although, for the free-stream boundaries, the stream function values are implemented as Dirichlet type boundary conditions; for the interior boundaries,

stream function values have to be calculated from the solution of the governing equation (Eqn. 3.9) at the boundary points.

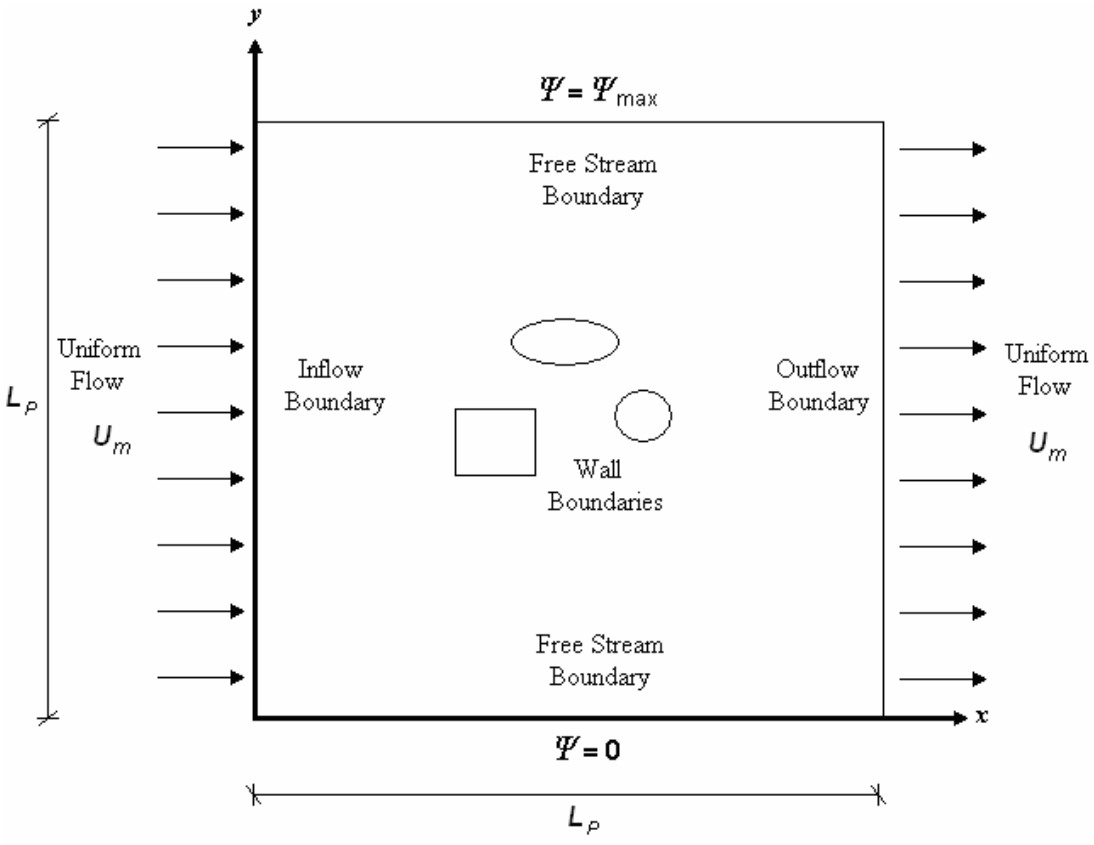
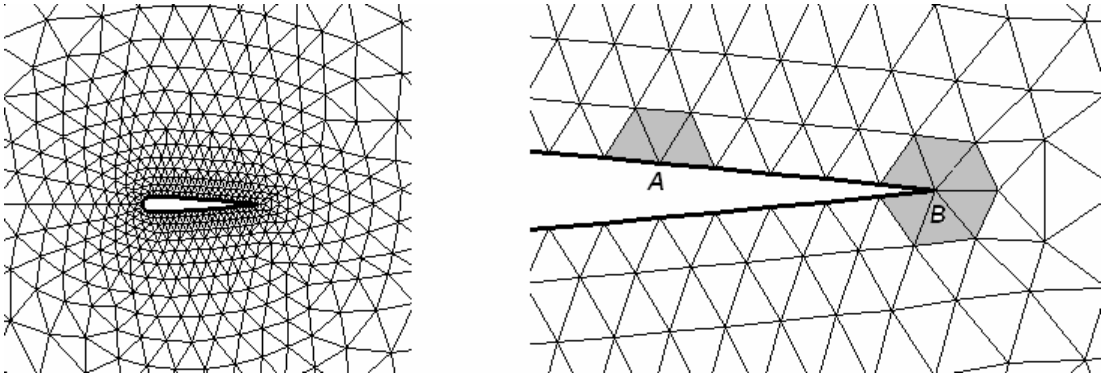


Fig. 3.4 Boundaries of the computational domain and the boundary conditions.

The finite volume scheme applied to internal control volumes is also employed to calculate the stream function values at the grid points on the solid boundary. The only difference is that a constant stream function value along the solid boundary is enforced. When the solution converges, all of the points on the same solid boundary have the same stream function value to form a streamline. Control volumes that are bounded by the centroids of the surrounding triangles and the solid boundary (Fig. 3.5.(b)) are considered. Depending on shape of the boundary curve different number of triangles may be contained in the control volume. The fluxes at the control surfaces are evaluated by area averaging as in the internal control volumes. The fluxes for points on the solid boundary (points d and e in Fig.3.6) are simply set to zero to imply the no-slip condition.



(a) (b)

Fig. 3.5 Triangles near wall boundaries a) Unstructured grid system for an airfoil, b) Typical boundary control volumes.

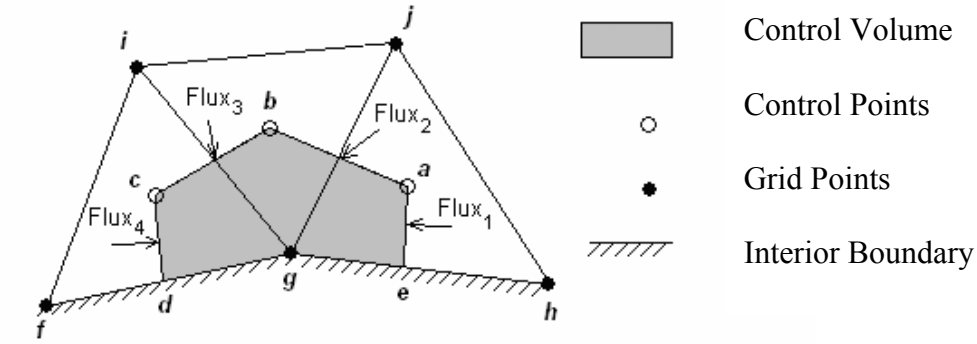


Fig. 3.6 Control volume on the solid boundary.

3.4 Stability Criteria and Convergence

Eqn 3.4 is a model heat equation. Diffusion number must be set to d_f = ¼ to obtain the allowable time step for iterations to keep stability of computations. Diffusion number limitation gives,

$$\Delta t = \frac{A_{C.V.}}{4\alpha} \tag{3.17}$$

where α = 1. The last term on the right side of eqn. 3.9 is normalized by the ψ_m and used as a dimensionless convergence parameter. Convergence is assumed when this dimensionless value is less than 10^{-5}. Computation time is approximately 15 minutes with a computer having 256MB RAM and 2.4 GHz CPU.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Determination of Potential Flow Domain Parameters

Potential flow solution is used to locate the bounding streamlines of the viscous flow domain around solid bodies. The streamline pattern obtained from potential flow solution should be independent of the dimensions of the potential flow domain and the grid system used in the solution. The parameters used to specify the potential flow domain are the domain size L_p, the minimum mesh size on the solid boundary Δ_{min}, the maximum mesh size on the outer boundaries Δ_{max} and the diameter, D, of a circular body located at the center of the domain. When the immersed body is not circular or when there are more than one immersed bodies, D is defined as the diameter of the circumscribing cylinder.

L_p must be large enough so that the development of the streamline patterns around the enclosed solid bodies are not affected from the inflow, outflow and free-stream boundary conditions. In this study, the potential flow domain is assumed to be a square of dimensions L_p x L_p (Fig. 3.4). To start the potential flow solution, L_p, Δ_{min} and Δ_{max} can be related to D. To investigate the appropriate relationships between domain parameters and D, a series of numerical experiments are performed.

Potential flow around a circular cylinder is considered as a test case to obtain the relationships between computational parameters. The cylinder diameter is selected as D = 1 m and the free-stream velocity is U_m = 1 m/s. Possible location of the boundary streamline for viscous flow domain around a circular cylinder is assumed at $y = 5D$ from the symmetry axis. Point $x = 0$, $y = 5D$ (Fig. 4.1) is fixed as the test point to compare the computed stream function values to the analytical solution. The dimensionless stream function value at the test point is defined as

$$\psi^{+}_{5D} = \frac{\psi(x = 0, y = 5D)}{U_m D} \tag{4.1}$$

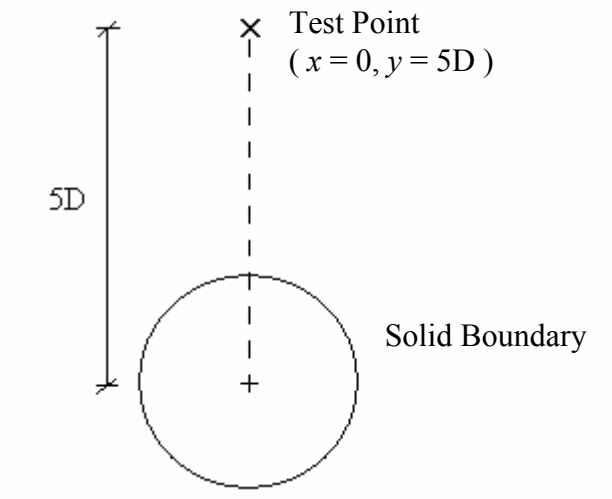


Fig. 4.1 Location of the test point where the stream function values are compared.

Analytical solution (Munson, 1998) gives ψ^{+}_{5D} = 4.95. Computational values for variable potential flow domain length L_p are given in Fig. 4.2. The computed stream function values approach the analytical solution asymptotically for $L_p/D > 30$. The potential flow domain size is fixed as,

$$L_p = 50D \tag{4.2}$$

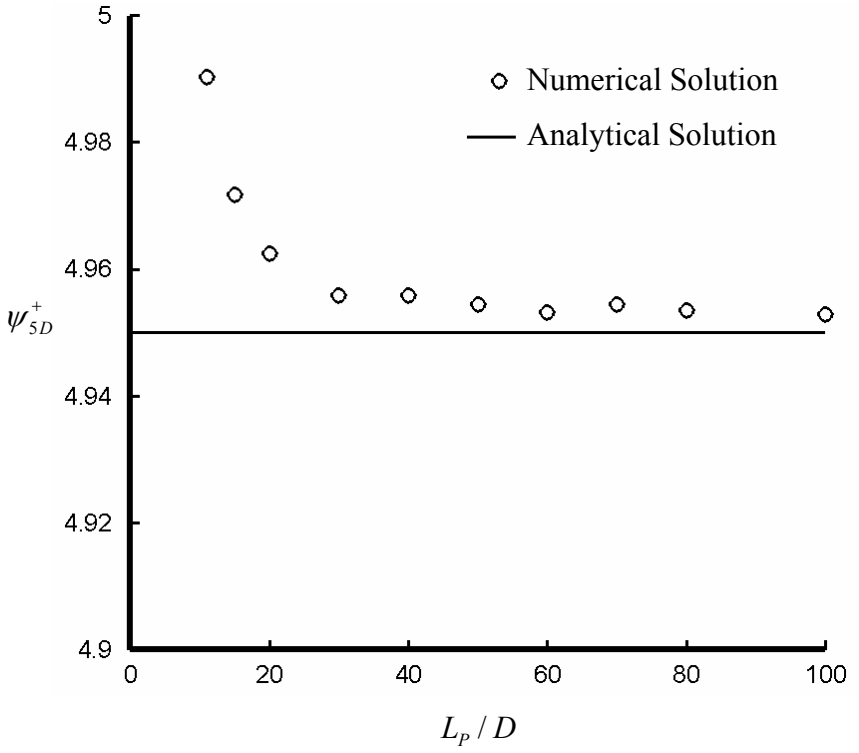


Fig. 4.2 Test point stream function value as function of L_P / D ($\Delta_{\max} / D = 1$ and $\Delta_{\min} / D = 0.05$).

The reference point stream function values are computed for variable Δ_{min} used on the solid boundaries. Computational results are shown in Fig. 4.3 in comparison to the analytical solution. The minimum mesh size is fixed as,

$$\Delta_{min} = 0.05D \tag{4.3}$$

A similar study is also carried out for the maximum mesh size used on the outer boundaries of the potential flow domain, Δ_{max}, and the results are shown in Fig. 4.4. The maximum mesh size is fixed as,

$$\Delta_{max} = D \tag{4.4}$$

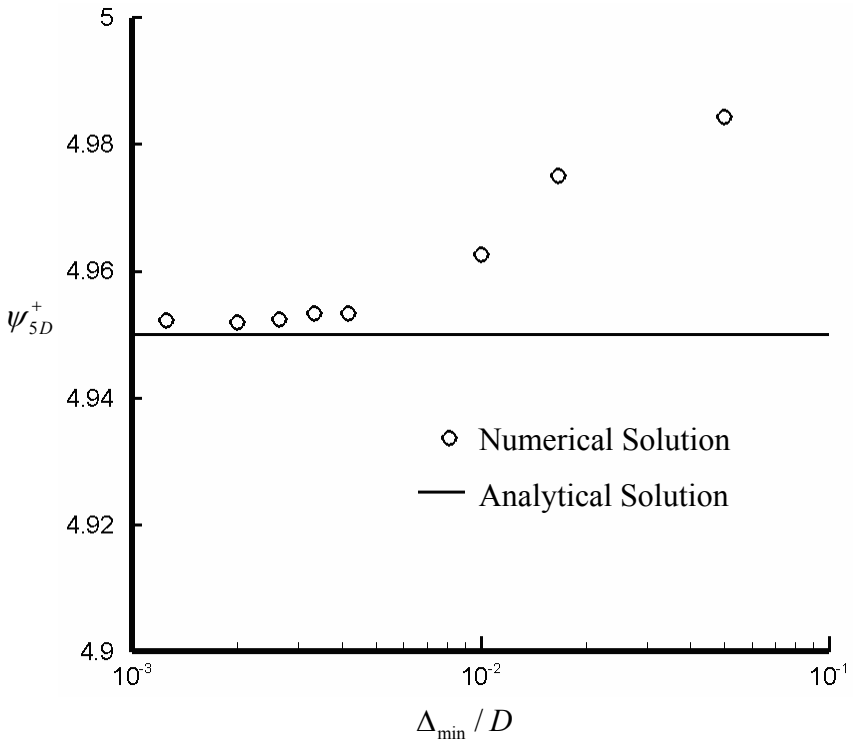


Fig. 4.3 Test point stream function value as function of $\Delta_{\min} / D$ ($L_P / D = 50$ and $\Delta_{\max} / D = 1$).

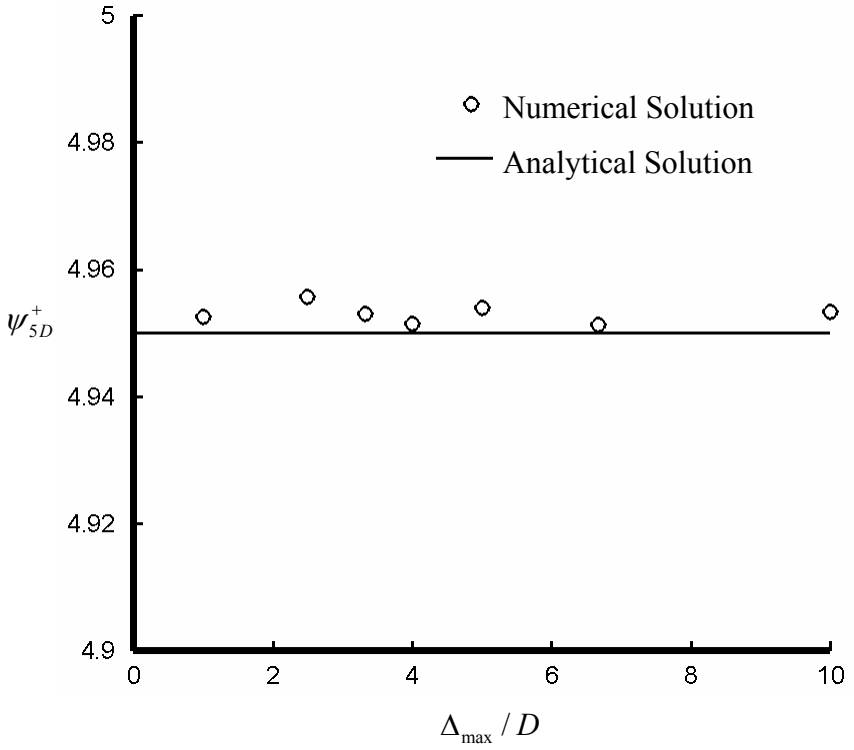


Fig. 4.4 Test point stream function value as function of $\Delta_{\max} / D$ ($L_P / D = 50$ and $\Delta_{\min} / D = 0.05$).

4.2 Deformation Modulus

Flow field around an immersed body can be divided into two regions, the viscous flow region surrounding the body and the inviscid potential flow region surrounding the viscous flow region (Fig. 4.5). The transition between the two flow regions is so gradual that definition of any dividing line in between can be restrictive for the viscous region. The dividing streamline required for computational purposes may be placed at different distances from the solid body depending on purpose of analysis and accuracy requirements. However, a quantitative measure of deformation characteristics can be specified to aid locating the bounding streamlines of the viscous domain. Maps of rate of deformation contours obtained from potential flow solution are used to identify possible borders of the viscous flow around the solid body. A dimensionless parameter is defined to construct the deformation rate contours,

$$D_M = \frac{\left|\frac{\partial u}{\partial y}\right| D}{U_m} \tag{4.5}$$

where D_M is named as deformation modulus. The streamlines around a cylindrical body are shown in Fig. 4.6. Contourlines of the deformation modulus are shown in Fig. 4.7.

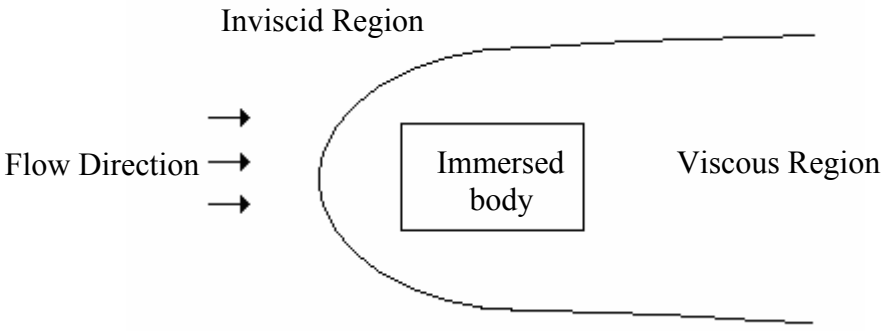


Fig. 4.5 Viscous and inviscid flow regions around a solid body.

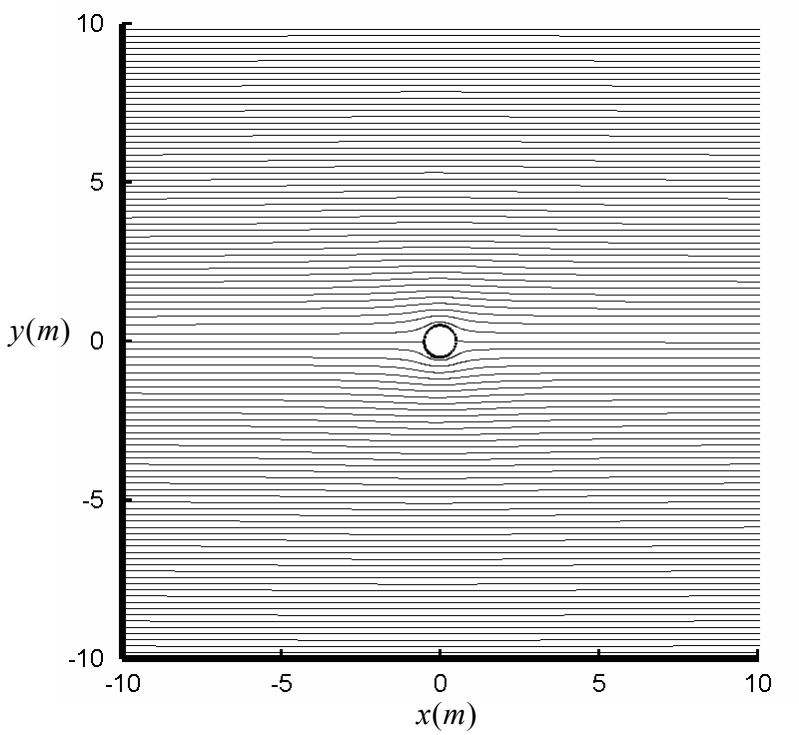


Fig. 4.6 Potential flow streamlines around a circular cylinder.

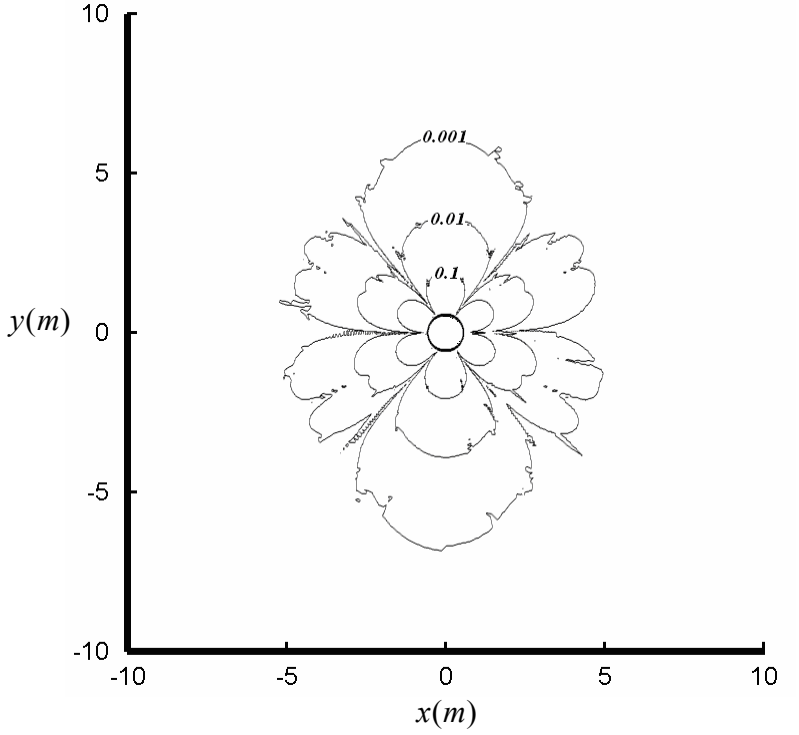


Fig. 4.7 Contourlines of deformation modulus around a circular cylinder.

Distribution of deformation modulus shows variations around the solid body. However, in general, it is possible to consider a continuous decrease in the deformation modulus with increasing radial distance from the cylinder. The main flow direction is parallel to x-axis, therefore the bounding streamlines can be located by evaluating the variation of D_M along y-axis at $x = 0$ (Fig. 4.8). Determination of the appropriate value of D_M to locate the boundary streamlines depends on availability of computing power to accomplish a numerical solution in the viscous domain. For more accurate and complete solutions, the viscous flow domain should be as wide as possible. However, such a large domain would require more computational power. Therefore, to obtain quick and economical solutions, it is possible to fix the boundary streamlines close to solid body and impose the velocity and pressure along the boundary computed from potential solution as Dirichlet boundary conditions rather than imposing free-stream boundary conditions.

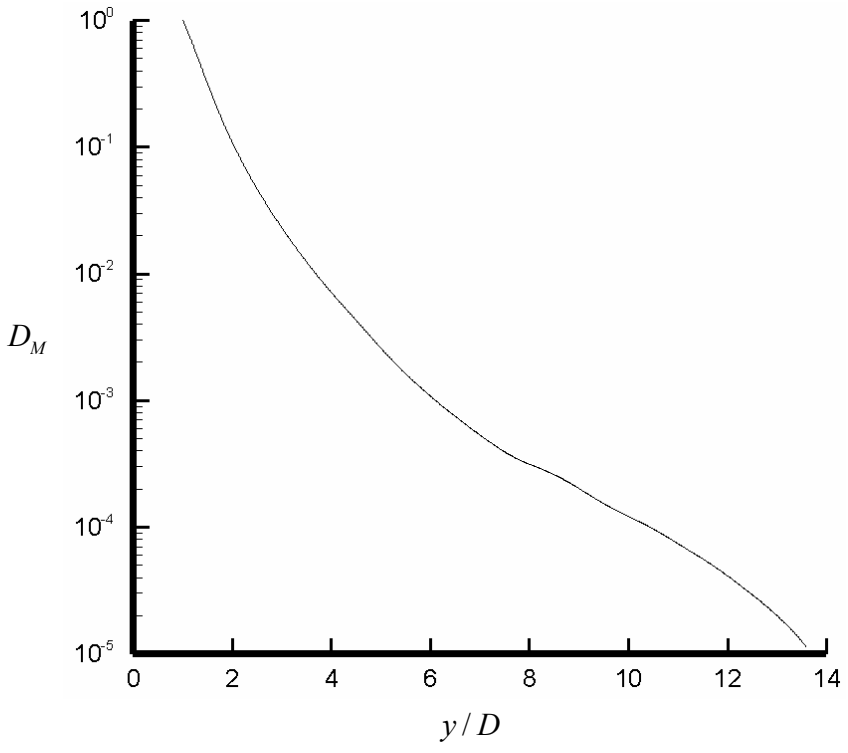


Fig. 4.8 Deformation modulus as function of distance from the cylinder (D = 1m, U_m = 1m/s, x = 0).

4.3 Evaluation of Boundary Data for the Viscous Domain

After potential flow solution, the limiting value of the deformation modulus is set and the corresponding point along y-axis is determined. The next step is to extract the streamline passing through that point. To do this, coordinates of the points along the streamline are interpolated from the computed stream function as illustrated in Fig. 4.9. In this figure points A, B, C and D are positioned with constant Δs distance along the streamline. Coordinates of points describing the streamline are interpolated from the computational grid data.

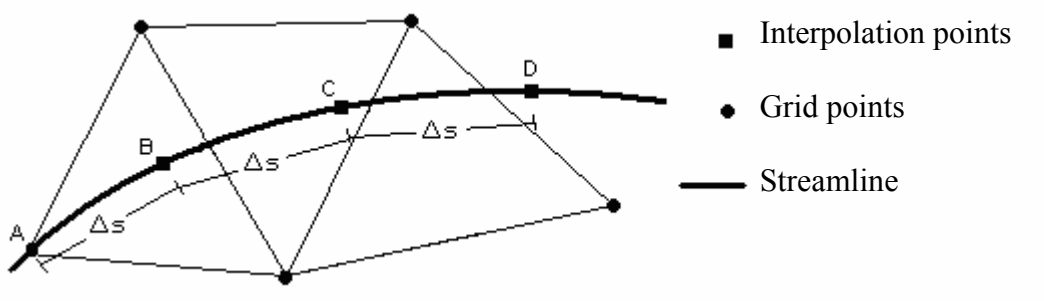


Fig. 4.9 Interpolation of boundary streamline from potential flow solution.

Velocity components, u and v along the streamline are also interpolated from the potential solution. Pressure along the boundary streamline is calculated by using the Bernoulli equation. As the total head is constant in potential flow, the pressure is computed from:

$$p = P_{atm} + \frac{\rho}{2}\left[U_m^2 - \left(u^2 + v^2\right)\right] \tag{4.6}$$

4.4 Examples

A possible choice of viscous flow domain for a circle with $D = 1$ is shown in Fig. 4.10. The boundary streamlines for viscous flow are obtained by setting $D_M = 0.01$. The domain boundaries and a typical computational grid for the viscous flow region are shown in Fig. 4.11.

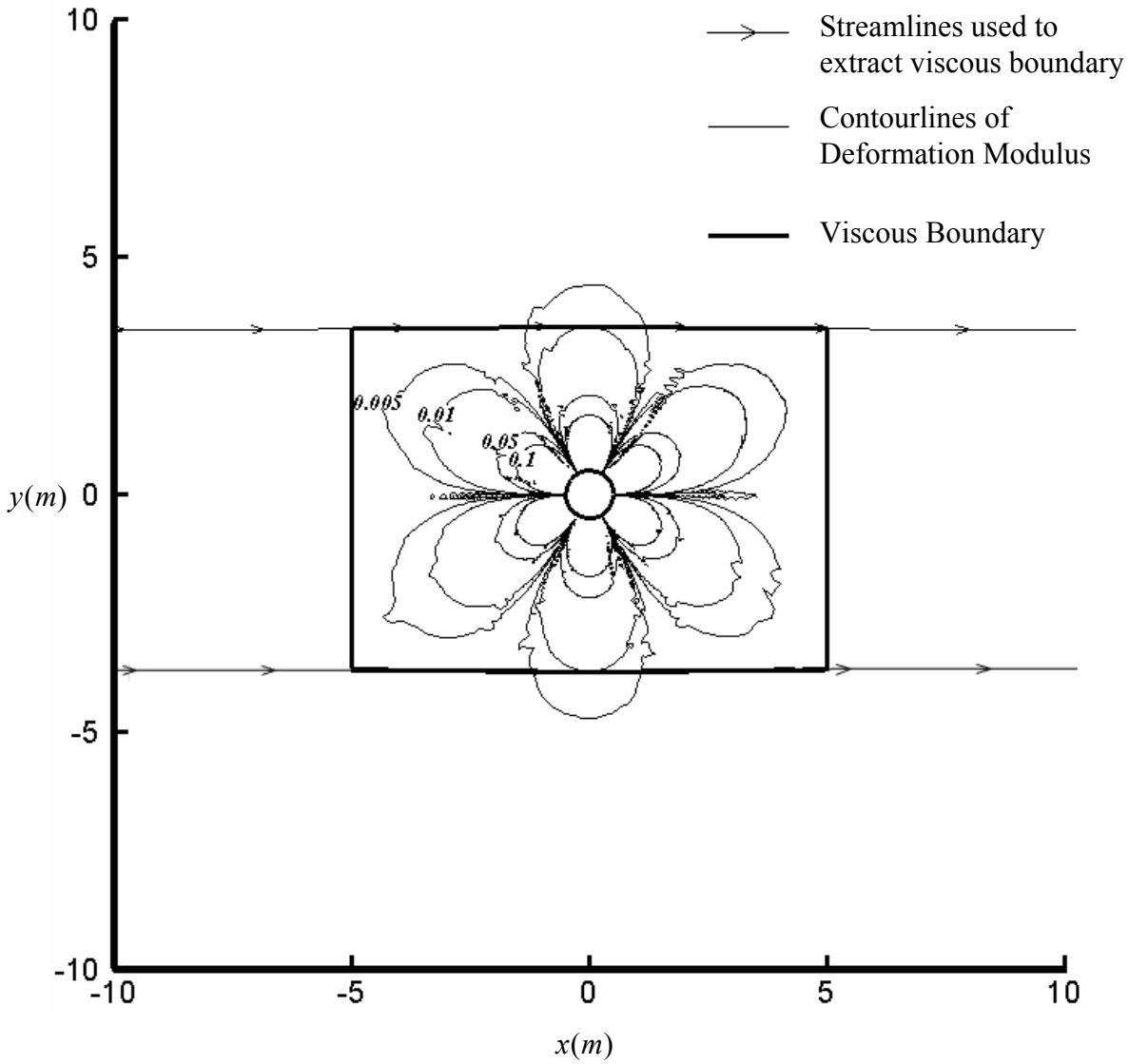


Fig. 4.10 Contours of deformation modulus and the bounding streamlines.

Another example study of flow domain enclosing three different bodies; one ellipse, one circle and a rectangle, is shown in Fig. 4.12. Streamlines for the potential flow are shown in Fig. 4.13. Extraction of viscous boundary for $D_M = 0.05$ can be seen in Fig. 4.14(a). Final computational grid prepared for the viscous flow solution is shown in Fig. 4.14(b).

Potential flow solution with different grid resolutions is shown in Fig. 4.15, Fig. 4.16 and Fig. 4.17 to illustrate the sensitivity of the solution to mesh size. As it can be seen from the figures, grid size does not have a crucial effect on streamlines.

Streamlines very close to the solid boundaries are more sensitive to the grid resolution.

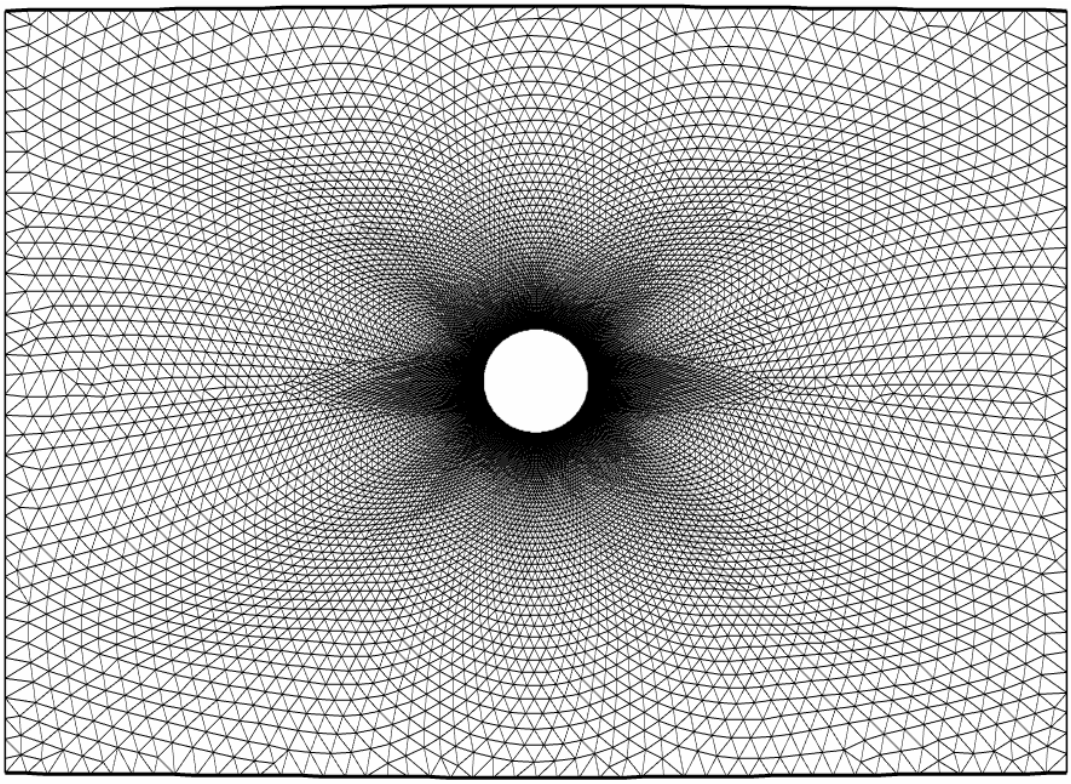


Fig. 4.11 Unstructured grid system for viscous region.

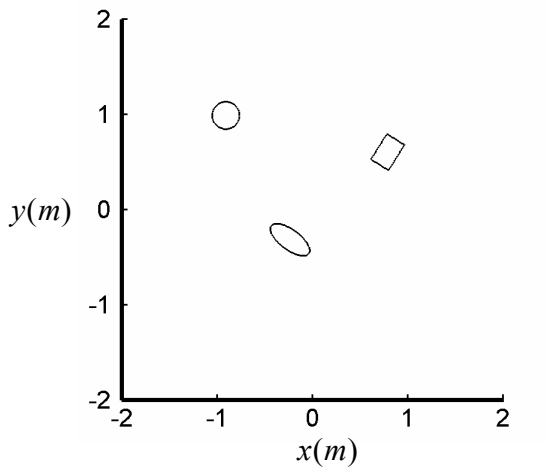


Fig. 4.12 Computational domain with three objects.

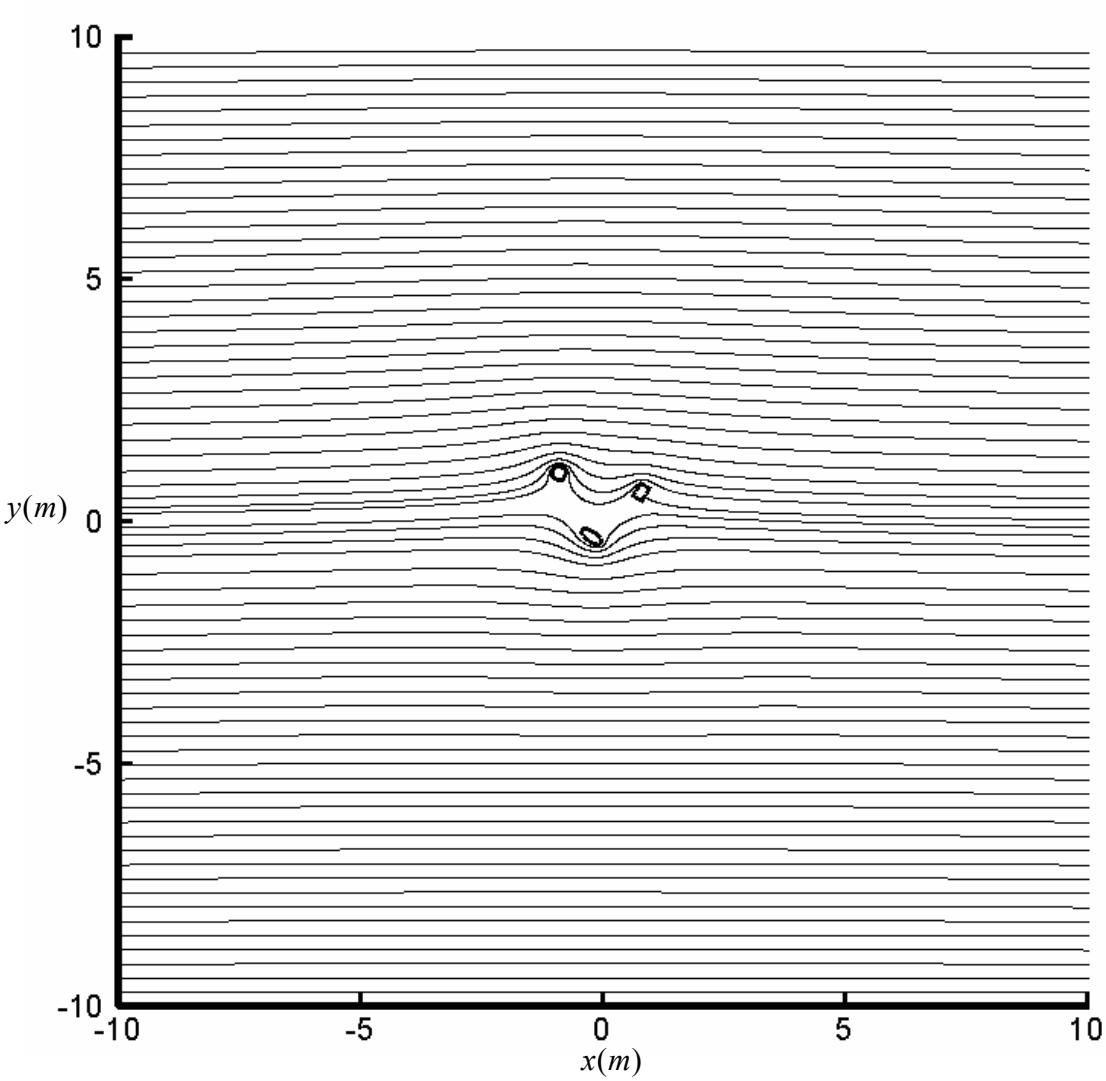
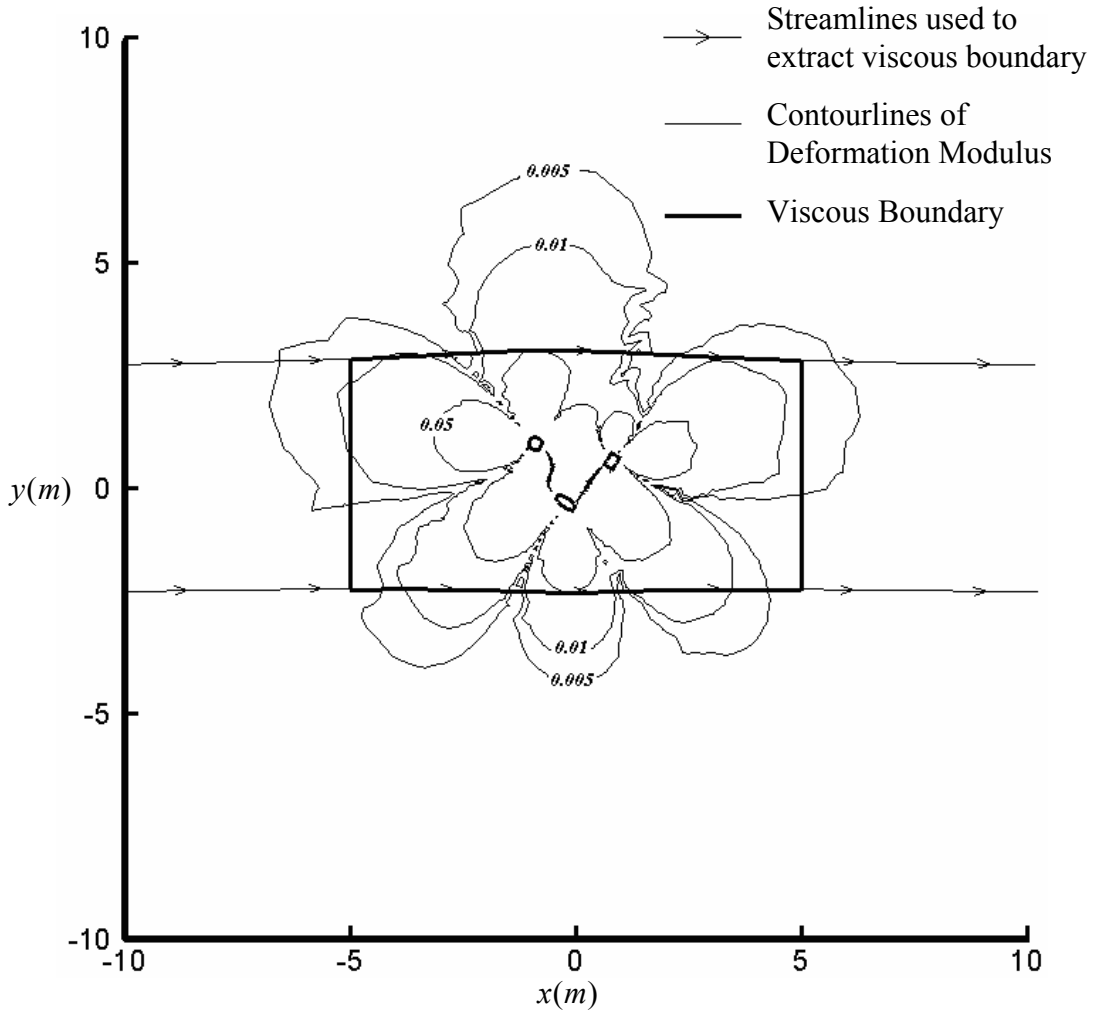
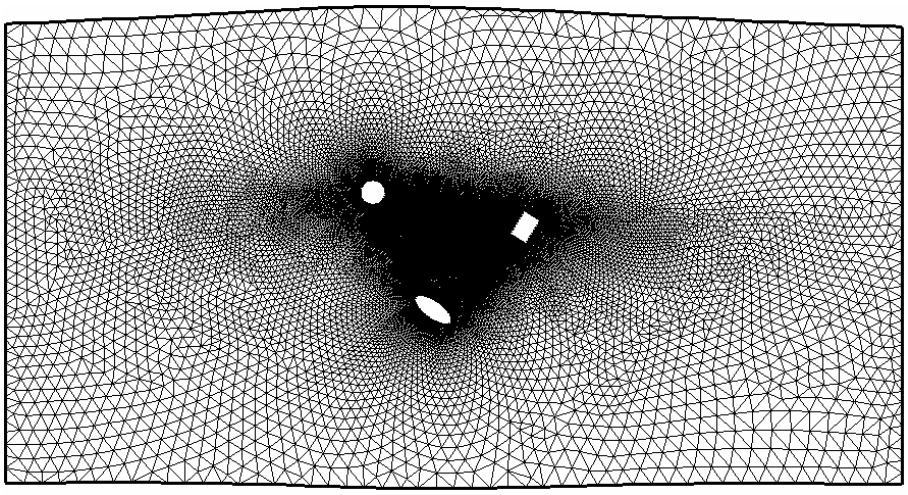


Fig. 4.13 Streamlines for potential flow around three immersed bodies.

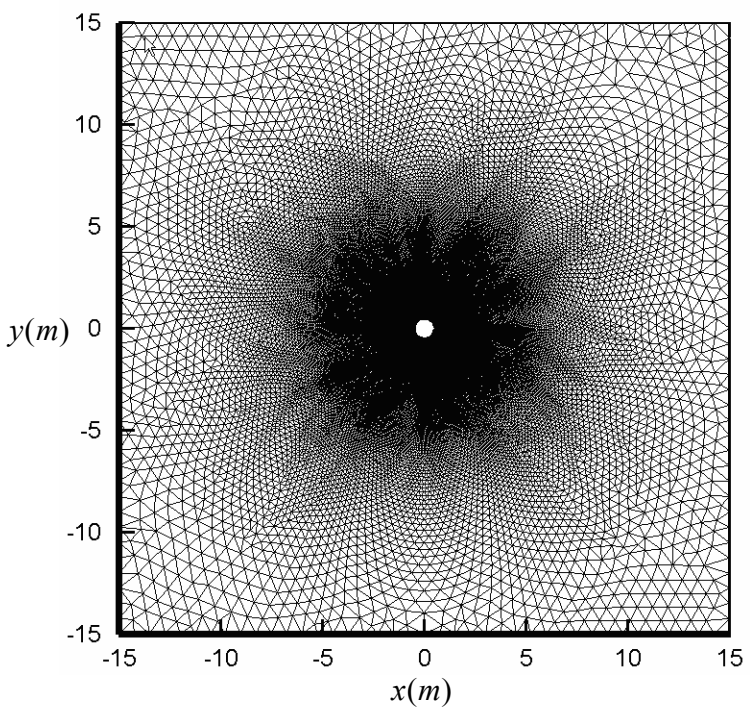


(a)

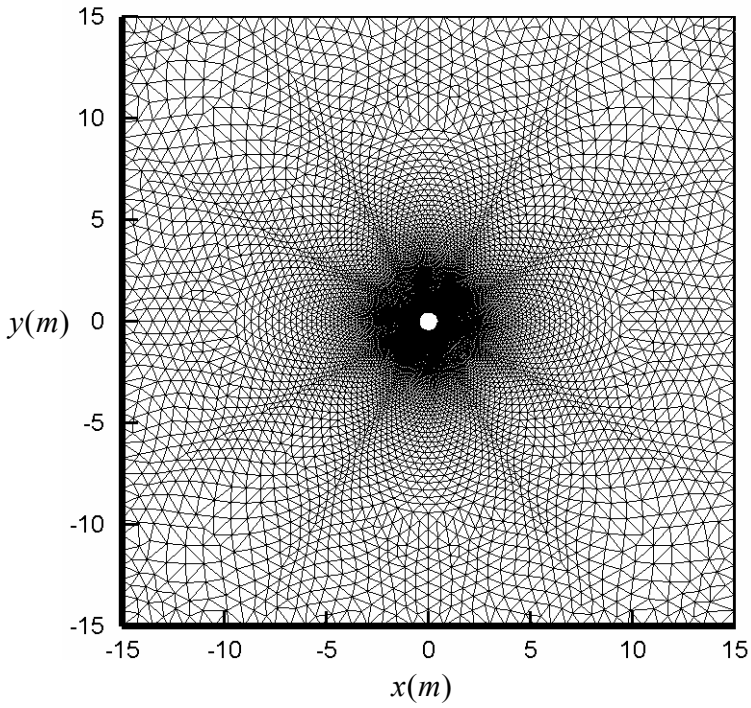


(b)

Fig. 4.14 a) Viscous boundary for $D_M = 0.05$ for flow around three immersed bodies
b) Unstructured grid for viscous flow solution.

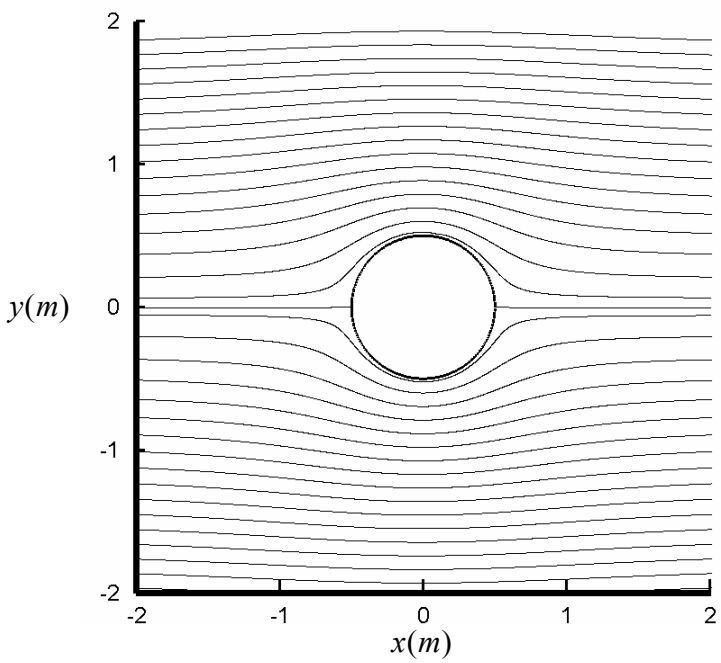


(a)

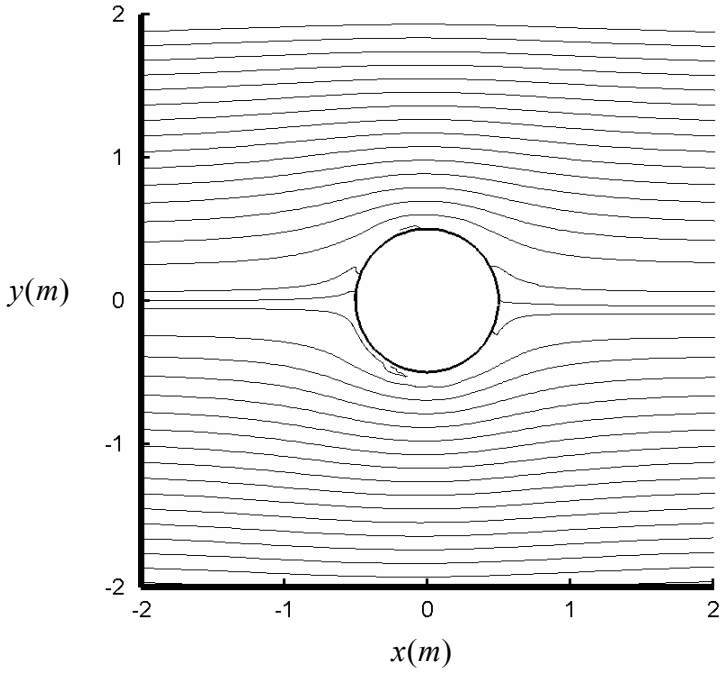


(b)

Fig. 4.15 Unstructured Grids different for different number of grid nodes, a) 46181 nodes b) 7004 nodes.

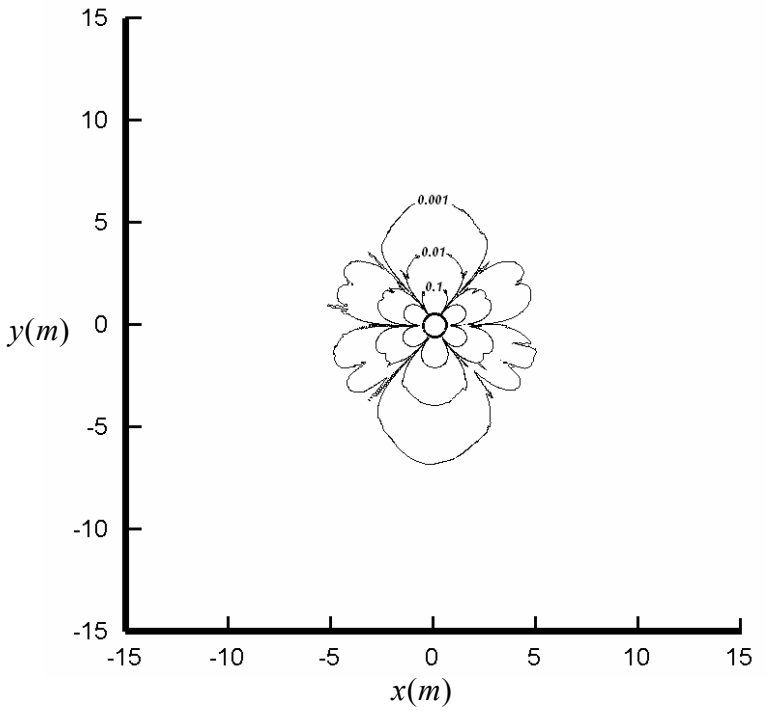


(a)

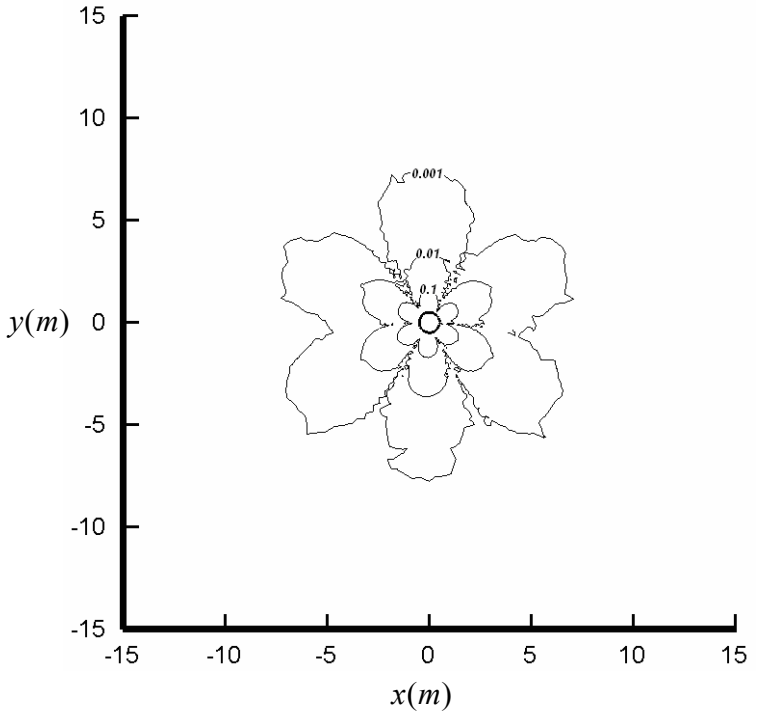


(b)

Fig. 4.16 Streamlines for different number of grid nodes,
a) 46181 nodes b) 7004 nodes.



(a)



(b)

Fig. 4.17 Deformation modulus using different number of grid nodes, sa) 46181 nodes b) 7004 nodes.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

A grid generation computer code is adopted for generation of unstructured triangular grid systems for domains involving immersed bodies of any shape at arbitrary orientations such as a group of tall buildings in horizontal plane. Finite volume method is used in the solution of Laplace equation for the stream function. The followings are the main contributions and conclusions derived from this study.

1. The minimum dimensions of the potential flow domain and the appropriate mesh size distributions required for a grid independent determination of the viscous flow boundaries are recommended as given by Equations 4.2, 4.3 and 4.4.
2. A deformation modulus is introduced by Eqn. 4.5 as a probe parameter to monitor locating the viscous flow boundaries. Appropriate choice of the value of the modulus to set the boundaries is dependent on the problem studied, accuracy requirements and computational power available.
3. The computer code can be used as a preprocessor for viscous flow computations to specify the computational boundaries for a smaller domain to reduce computational costs. While reducing the domain size any errenous restriction on the flow development is eliminated by importing Dirichlet type boundary conditions for the velocity and pressure fields from the potential flow solution.
4. User manual is prepared and presented in Appendices for use of the codes by graduate students in fluid mechanics for educational purposes.

REFERENCES

Anderson J.D. "*Computational Fluid Dynamics The Basics with Applications*" Int Ed., McGraw Hill Singapore, 1995

Chapman, D.R. "*Computational Aerodynamics Development and Outlook*", AIAA Journal, Vol.17, No.12, 1979, pp.1293-1313

Kütler, P. "A *Perspective of Theoretical and Applied CFD*", AIAA Journal, Vol.23, No.4, 1985, pp.328-341

Doğru, K. "*Parallel Processing of Two-Dimensional Euler Equations for Compressible Flows*", M.Sc. Thesis, Middle East Technical University, May, 2000.

Hirsch, C. "*Numerical Computation of Internal and External Flows*", Vol.1,2. Jonh Wiley & Sons, 1989-1990.

Telçeker, N., Mutaf, Ö. "*Aerodinamik Katsayıların Euler Çözücüleri ile Bulunması Konusunda Literatür Araştırması*", TÜBİTAK-SAGE Report, 92/04-4/1.0, 1992

Wilkinson, R. Andrew "*Numerical Solution of the Euler Equations for Transonic Airfoil Flows Using unstructured Grids*" M.Sc. Thesis, University of Toronto, January 1992

Baker, T., "*Computational Fluid Dynamics: On Unstructured Grids and Solvers*", Lecture Notes for Von Karman Institute for Fluid Dynamics Lecture Series: Computational Fluid Dynamics Lecture Series 1990-03, Brussels, Belgium, March 1990.

Munson, Young, Okishi, "*Fundamentals of Fluid Mechanics*" Third Edition, 1998

J.-D. Müller, ``*On Triangles and Flow*", PhD Thesis, The University of Michigan, 1996.

Aydın, İsmail, "*CE 580 Computational Fluid Dynamics Lecture Notes*", Middle East Technical University, 2004

APPENDIX A

USER MANUAL FOR DELAUNDO

DELAUNDO has two input files. First one is a control file with ".CTR" extension, which includes parameters that control orientation and number of unstructured grids. Second one is the geometry file with ".PTS" extension, and it includes the geometry data, i.e. coordinates of boundary points. These two files must be in the same directory with "DELAUNDO.exe".

A.1 Control File (CTR file)

A control file determines how DELAUNDO will treat the nodes read from the PTS file. In a CTR file, information is addressed by six-letter capitalized keywords. First, keyword is written and then in the next line its value is entered. DELAUNDO uses default values for parameters that are not included in control file. Keywords and their options are:

HELPME

The help menu will be called up at the beginning of program execution.

XAMPLE

The example menu will be called up and allows user to choose one of the examples.

RELEAS

It gives information about the current release.

VERBOS

This sets the verbosity. '0' gives no output at all, except for warnings and fatal error messages, '5' will give all important data in the progress of grid generation. Some of these are, time spent for each part of the triangulation process, number of elements of triangulation and etc. Default value is 3.

ALLPAR

If it is set to 't' or 'y', user will be prompted for all parameters that apply to current selections. Default is 'n' or 'f'.

INFILE

File name of the PTS file. The default is 'delaundo.pts'.

INFORM

The input file can be formatted by setting INFORM to 'y' or 't'; and unformatted by setting INFORM to 'n' or 'f'. Default is 't'.

NODEUS

Can be set to 't' or 'y'; this will make DELAUNDO use a given set of internal nodes. Note that only either of *NODEUS* or *NODECO* may be specified due to coding constraints. Default is 'f' or 'n'.

NODECO

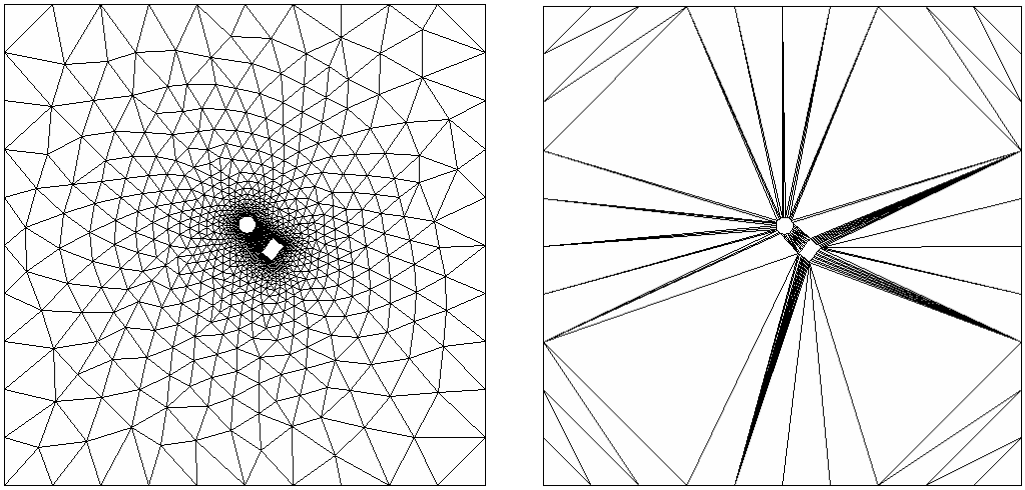
Can be set to 't' or 'y'; this will make DELAUNDO construct a set of internal nodes with the Frontal Delaunay method (Fig.A.1). Default is 't' or 'y'.

ASKROW

Can be set to 't' or 'y'; this will make DELAUNDO prompt the user for more rows to be constructed, once the current counts have been exceeded. Otherwise, the process will output the grid at the current stage. Default is 'f' or 'n'.

ANTICO

Can be set to 't' or 'y'; this will make DELAUNDO use the "anti-connectivity-information" specified in the PTS file. Default is 'f' or 'n'.



(a) (b)

Fig. A.1 Preview of initial grid by *NODECO,* a) *NODECO* off, b) *NODECO* on.

ALLDIS

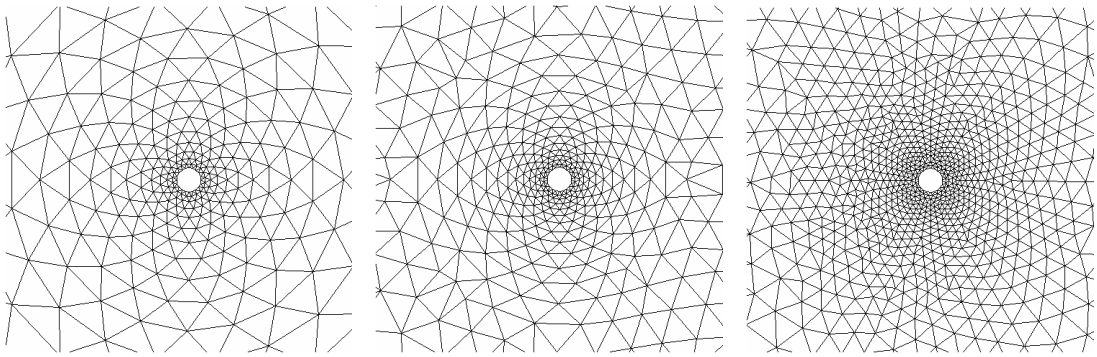
Can be set to 't' or 'y'; this will make DELAUNDO remove all edges in the background grid that connect non-consecutive boundary nodes, even if they reside on the same boundary segment. Use it with discretion as this might lead to many extra nodes needed for the disconnection of close surfaces.

SPCRAT

This floating point value specifies the ratio between the spacing gradients at the points of highest and lowest spacing Values higher than one will cause DELAUNDO to interpolate with a power law to extend the regions of fine spacing further into the domain (Fig.A.2). Default is 1.

DTOLER

This floating point value specifies the fraction of the background mesh size that is being used as a minimum distance between nodes. Default value is .65.



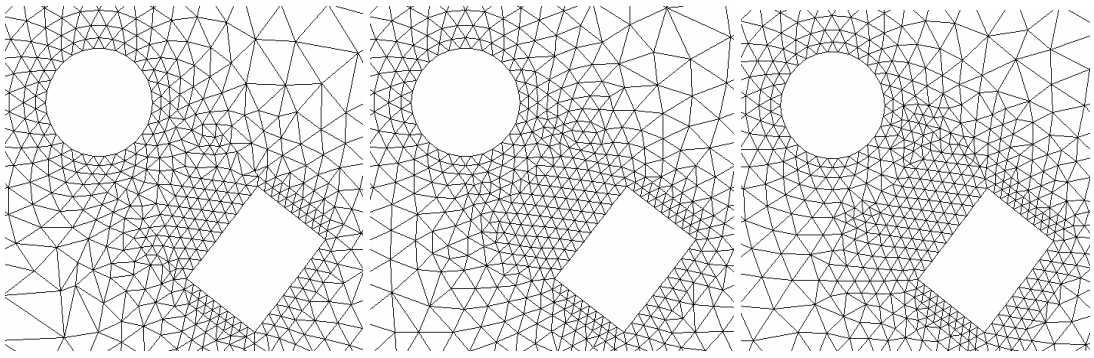
(a) (b) (c)

Fig. A.2 Effect of *SPCRAT* on unstructured grid generation. a) *SPCRAT* = 0.4, 373 nodes, b) *SPCRAT* = 1, 516 nodes, c) *SPCRAT* = 4.0, 1029 nodes.

QTOLER

This floating point value specifies the minimum fraction of the maximum side length that the smaller sides must have in order to make the triangle acceptable. Default value is .65

This parameter is another critical parameter that influences the grid quality. By setting different values to *QTOLER* between 0.5 and 1.0, different grid systems can be obtained (Fig.A.3). The result is more obvious at the mid-sections of different inner boundaries.



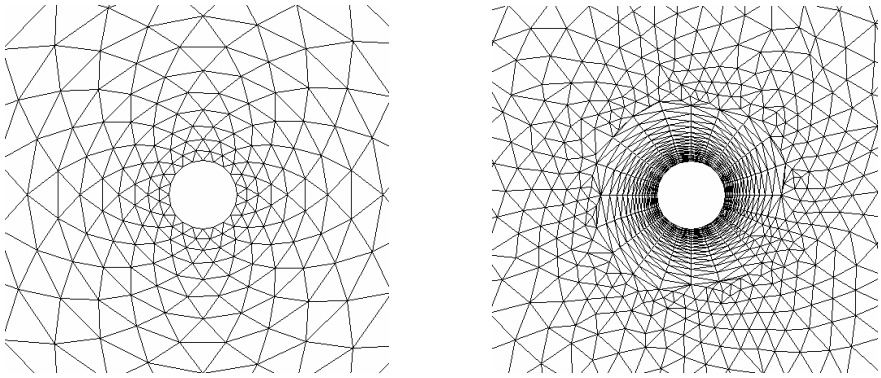
(a) (b) (c)

Fig. A.3 Effect of *QTOLER* on unstructured grid generation. a) *QTOLER* = 0.5, b) *QTOLER* = 0.80, c) *QTOLER* = 0.95.

STRETC

Can be set to 't' or 'y'; this will make DELAUNDO construct a layer of wedge type stretched triangles in form of rectangular boxes around frontal surfaces. Default is 'f' or 'n'.

Actually, this option enables to form a structured grid region around the inner boundaries, but the grids are still triangular. Using this option creates hybrid grid systems made up of both structured and unstructured triangular grids can be generated (Fig.A.4). Setting a stretched layer around the inner boundary can be useful in order to demonstrate a viscous region where smaller grid is required in order to achieve a better solution.



(a) (b)

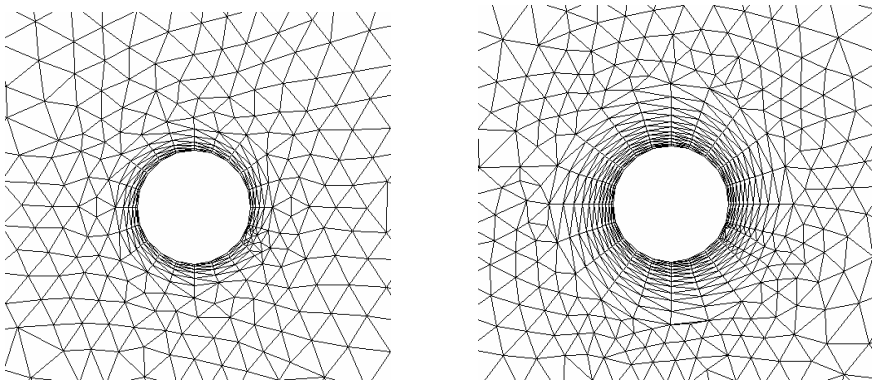
Fig. A.4 Effect of *STRECT* on unstructured grid generation,
a) *STRECT* off, b) *STRECT* on.

BTOLER

This floating point value specifies the minimum fraction of the background mesh size that is being used as a minimum distance between nodes in the background grid. Default value is 2.

DELTAS

This floating point value specifies the thickness of the stretched layer in the scale of the other points (Fig.A.5). IF *STRETC* is specified as 't' or 'y', this parameter is required.

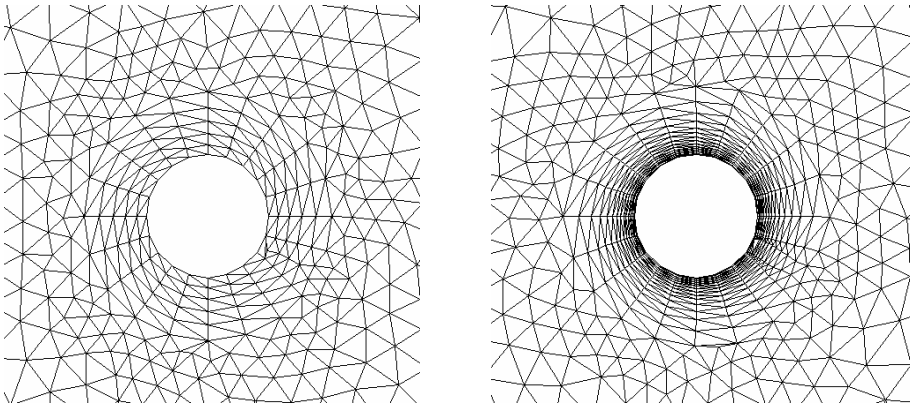


(a) (b)

Fig. A.5 Effect of *DELTAS* on unstructured grid generation,
a) *DELTAS* = 0.1, b) *DELTAS* = 0.2.

MAXASP

This floating point value specifies the maximum aspect ratio in the stretched layer. IF *STRETC* is specified as 't' or 'y', this parameter is required. Difference between the *DELTAS* and the Aspect Ratio is that *DELTAS* specifies the width of the stretched region, but the aspect ratio is the ratio between the width and the height of the triangles that forms the stretched region (Fig.A.6). In order to have a good grid quality, triangles must not have small angles at the corners. From the grid quality point of view, it may be necessary to use a smaller aspect ratio. But decreasing it results in less grid points inside the stretched region.



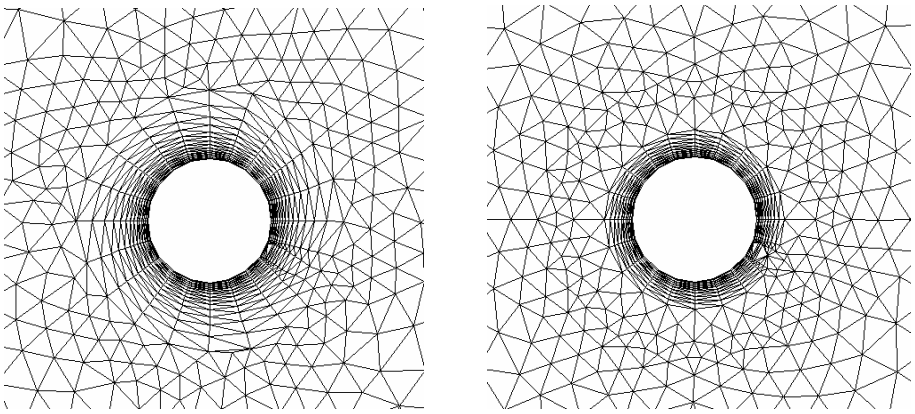
(a) (b)

Fig. A.6 Effect of *MAXASP* on unstructured grid generation,
a) *MAXASP* = 2.0, b) *MAXASP* = 20.

MVISRO

This integer value specifies how many stretched, viscous rows are to be built (Fig.A.7). If *ASKROW* is set to 't' or 'y', the user will be prompted for more rows. (Limiting the number of rows is to be considered a debugging tool.) Default is 30000.

It is important not to define a value for this parameter in order for the number of rows to conform to the stretched layer thickness defined by *DELTAS*.



(a) (b)

Fig. A.7 Effect of *MVISRO* on unstructured grid generation, a) *MVISRO* = (Default) 30,000, b) *MVISRO* = (Default) 10.

ISMOOT

This integer value specifies how many stretched rows of cells will be opened for isotropic re-triangulation once the stretched process has terminated. 0 does not allow any re-triangulation,

a) **1:** allows re-triangulation of the outermost cells,
b) **2:** allows re-triangulation of the neighbors of the outermost cells as well.

Note that whenever stretched layers from different geometry segments impinge on each other, the "sew-up" becomes more gentle with *ISMOOT* = 2. Default is 2.

MISORO

This integer value specifies how many isotropic rows are to be built. If *ASKROW* is set to 't' or 'y', the user will be prompted for more rows. (Limiting the number of rows is to be considered a debugging tool.) Default is 30000.

LAPLAC

If set to 't' or 'y', a Laplacian filter is applied to smoothen the mesh. Default is 'f'. Note that the applying the Laplacian to a stretched mesh will most likely fail due to mesh overlap.

RSWEEP

This integer value specifies the number of relaxation sweeps for the Laplacian. Default is 10.

LPBETA

This floating-point value specifies how much the weight of the grid points increases with increasing vertex degree. 0 means fixed weights, 1 is maximum variance. Default is 5.

RELFAC

This floating point value specifies the relaxation factor for the Laplacian. The stability limits are [0,1], default is 0.75

MLEVEL

This integer value specifies the number of multi-grid levels that are to be produced. The maximum permissible depth is 10, default is 1.

FLATSW

Set to 't' or 'y', this will make DELAUNDO swap diagonals in the final mesh in order to minimize the maximum angles. (Note that although the implementation is rather

efficient to prevent any N^2 worst case, it is supposed to be used only to correct cases where the stretched process has stopped before reaching isotropy and the isotropic process created flat cells.) Default is 'f','n'.

ANGMAX

This floating point value specifies the maximum tolerable cell angle before *FLATSW* kicks in (Fig.A.8). Note that this does not guarantee that all angles are below that value. A diagonal switch is not carried out if the maximum angle in the quadrilateral were to increase. So the optimum value must be found by trial and error. Default is 120.

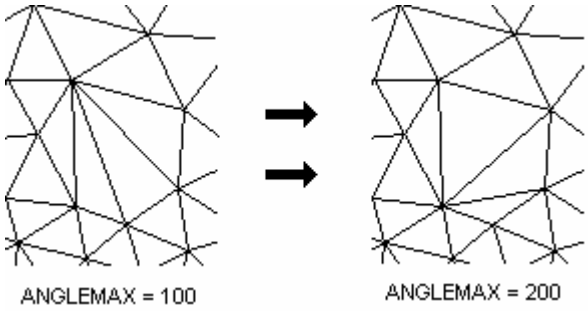


Fig. A.8 Effect of *ANGMAX* on the final grid connectivity.

MCYCSW

This integer value specifies how many swapping cycles will be executed. Default is 10.

OUTFIL

This string is the output file name. The default is 'delaundo.dpl'.

OUTTYP

This character determines the output type. Possible types are:

a) **'t':** the triangulation with the cells outside the domain removed.
b) **'q':** the triangulation of t, but with 6 nodes per triangle for quadratic elements.

c) **'h':** the convex hull with all the triangles. This is useful for creating a grid of scattered nodes for interpolation purposes or for debugging in case a boundary enforcement check fails.

d) **'b':** the background grid with the automatically added background nodes. Nodal values given are the node spacing as rho (1. state quantity) and the stretching magnitude multiplied by the spacing as rho*u (2. state quantity).

e) **'i':** the initial triangulation of the set of given nodes for debugging purposes. The default is 't'.

OUTFOR

This character determines the output format for special programs. It should be used only for DPLOT program; other programs are for other previous projects in which DELAUNDO was used recently. Possible formats are:

a) **'g':** a VKB .gri file with grid.

b) **'o':** a VKB .out file with grid and solution.

c) **'d':** a DPLOT file.

d) **'u':** a.ucd file.

The default is 'd'. Output of original DELAUNDO is modified so that besides the above mentioned output files, a TECPLOT file is also created, as described in Appendix B.

TITLE1,...,TITLE4

Here are the titles for the 'g' and 'o' output formats, as specified for *OUTFOR* parameter. Titles 2-4 apply only to 'o'. Default is ' '.

DOLOGF

Set to 't' or 'y', this will make DELAUNDO to write a log file. Default is 'f' or 'n'.

LOGFIL

This string is the name of the log file. Default is 'delaundo.log'.

ENDDAT

Any information after this keyword will be ignored.

A.2 Geometry Data File (PTS file)

A PTS file contains the geometry data, including the coordinates of initial boundary grid nodes. The geometry is split up in segments that are given by ordered sets of boundary nodes, preferably given with the domain to the left. Segments are connected to other segments at their ends, possibly themselves. The nodes where segments connect are listed for both segments.

The information in the formatted PTS files are read via, the familiar six-letter capitalized keywords. The information pertaining to a keyword follows the line[s] after the keyword.

The following keywords are known to DELAUNDO:

NEWBND

This opens a new boundary segment and closes the previous one (if there was). Any information pertaining to one boundary must be given before the next *NEWBND*, *INDEXY* or *ENDDAT* statement.

%

Comment sign. A line beginning with '%' is ignored.

NAMEBN

Each segment can receive a name which is an integer between 1 and 20. Segments are addressed with this name. If the name is omitted, the name defaults to the number of the boundary taken from the position in the PTS file, if this name is not yet taken. Otherwise, the first open name starting from 1 is selected. Note that, it is a bad practice to omit the name unless there are self-connected boundaries that carry no

anti connectivity information. In addition, user might run into trouble with a name that is chosen for a boundary that coincides with a name given by the program to a boundary that is listed earlier in the PTS file.

NRBNDE

The number of nodes found in the PTS file for this boundary is compared to this integer. A warning is issued if they do not coincide.

NFRSBN

The name of the boundary connected to the first node of this boundary. If *NFRSBN* is omitted, the boundary is supposed to be linked to itself. *NFRSBN* is a little tricky for wake-type boundaries that are connected to a solid surface. By convention, for a 'counter clockwise' wake, i.e. *ITYPBN* = +4, *NFRSBN* is the boundary to the left of the juncture, as viewed from the wake. In this case, *NLSTBN* = 0. Note that, for reasons of keeping a simple data structure, wake-type boundaries may not be connected to other wake-type boundaries. They may only be left open, *NFRSBN* = 0, or connected to a solid surface at a junction of boundary segments at one of their ends.

NLSTBN

The name of the boundary connected to the last node of this boundary. If *NLSTBN* is omitted, the boundary is supposed to be linked to itself. In the case of *ITYPBN* = -4, *NLSTBN* is the boundary to the left of the juncture, as viewed from the wake. The limitations on the connectivity of wake-type segments are unchanged, naturally.

ITYPBN

It is an integer specifying the type of the boundary (Fig.A.9). Possible types are:

a) **1:** a frontal, enforced boundary. (wall boundary)
b) **2:** a non-frontal, enforced boundary. (outer boundary.)

c) **3:** a non-frontal, non-enforced set of nodes. (a set of interior nodes to be used when also constructing interior nodes. Note that as there is no nodal overlap due to connected segments, all nodes are used.)
d) **4:** a frontal, non-enforced boundary. (a wake. Note that as there is no nodal overlap due to connected segments, all nodes are used.)
e) **'9':** a boundary in the background grid. (can be used to change the spacing distribution.)

All boundaries with positive type have the domain to the left, the ones with negative type to the right. Default is 1.

MINRES

The minimum required resolution for this segment. The segment will not be coarser beyond this value. Default is 2.

ANTICO

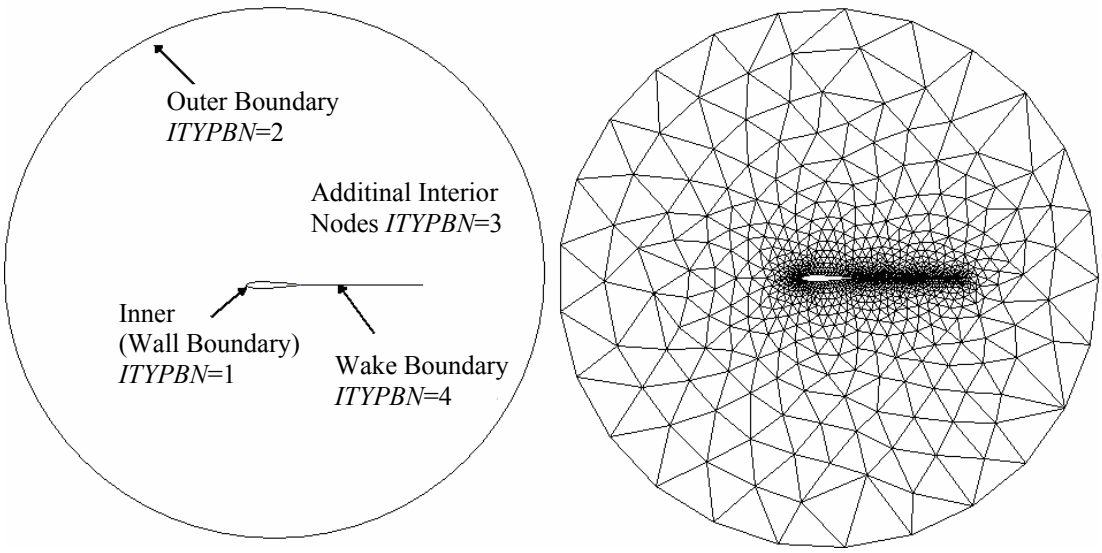
It sets the set of names of segments this one must not be connected to. Connection between mutual *ANTICO* boundaries will be removed by insertion of nodes with a spacing value extrapolated from the surfaces with an average gradient. This information is used only if *ANTICO* is set in the CTR file.

BNDEXY

The set of boundary nodes as *x,y* pairs.

NRINDE

The number of internal nodes found in the PTS file is compared to this integer. A warning is issued if they do not coincide. Note that *NRINDE* and *NRBNDE* are equivalent.



(a) (b)

Fig. A.9 Basic Boundary types defined by *ITYPBN*, a) Boundaries of the domain, b) Unstructured grid for flow around an airfoil and the wake behind it.

INDEXY

The set of boundary nodes as *x,y* pairs. An *INDEXY* statement opens a type 3 boundary that is a set of non-enforced foreground vertices. It is equivalent to specifying *ITYPBN* = 3 or *NFRSBN* = 0 or NLSTBN = 0. Every new *INDEXY* statement will open a new type 3 segment. The fact that all arrays pertaining to boundary vertices are only dimensioned for a small subset of all the vertices, which should not cause any trouble since these arrays are only invoked for the frontal process. That is, if there is large number of internal vertices, user should list them at the end after the boundaries with frontal character.

ENDDAT

Anything after an *ENDDAT* statement will be ignored.

APPENDIX B

USER MANUAL FOR POTENTIAL FLOW SOLVER

For the potential solution program, two input files are required. First one is "GBOUND" file which includes the boundary information, and second one is "TECH.DAT" file which includes unstructured grid data file.

B.1 GBOUND File

In this file, information about the boundary grids is stored. It includes the following data:

NINTB		
I	*NSINT(I)*	*NEINT(I)*
I+1	*NSINT(I+1)*	*NEINT(I+1)*
.	.	.
.	.	.
NIS	*NIE*	
NLS	*NLE*	
NOS	*NOE*	
NUS	*NUE*	

NINTB is the total number of interior boundaries.

I is the index for each interior boundary. Each row contains the starting and end points of each interior boundary. Note that, total number of the rows starting with the index number *I* is equal to *NINTB*.

NSINT(I)

Index of the starting node of the I^{th} interiour boundary.

NEINT(I)

It is the index of the end node of the I^{th} interiour boundary.

NIS

It is the index of the starting node of the inner boundary.

NIE

It is the index of the end node of the inner boundary.

NLS

It is the index of the starting node of the lower boundary.

NLE

It is the index of the end node of the lower boundary.

NOS

It is the index of the starting node of the outer boundary.

NOE

It is the index of the end node of the outer boundary.

NUS

It is the index of the starting node of the upper boundary.

NUE

It is the index of the end node of the upper boundary.

A sample data corresponding to the above notation is defined in Fig.B.1. Sign convention considered for the boundaries except the interior boundary is counter clockwise starting from the upper-right corner, which is *NUS*. Sign convention considered for the interior points is clockwise.

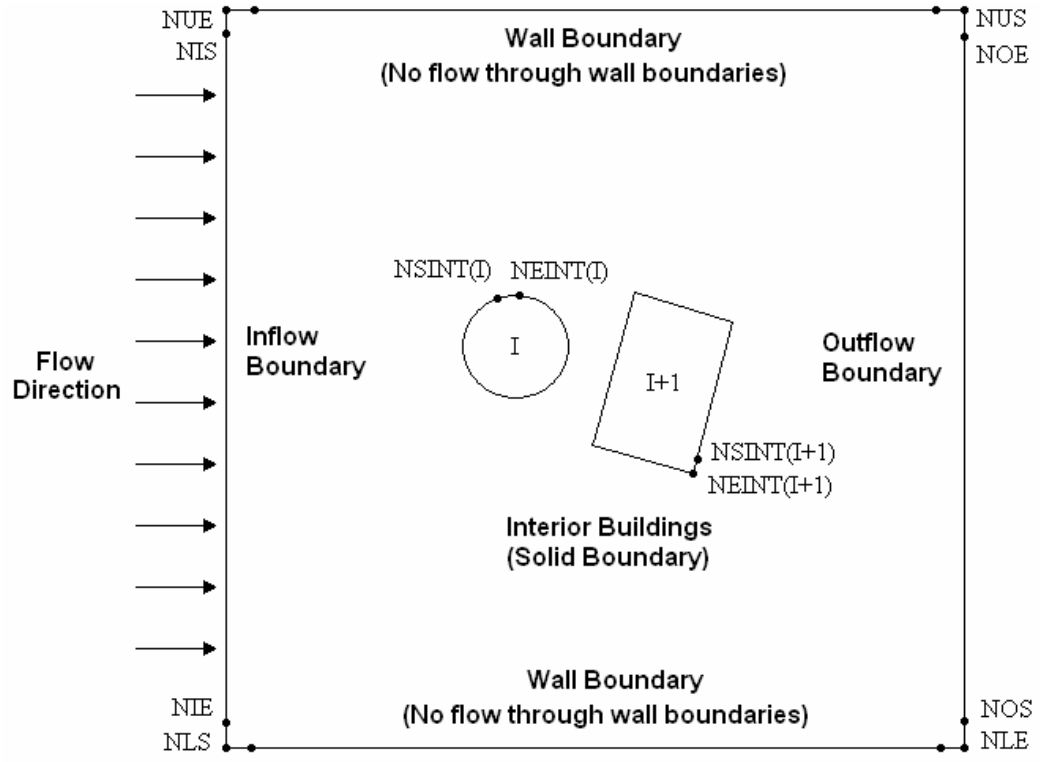


Fig. B.1 Notation to define the boundary nodes in the GBOUND file.

B.2 TECH.DAT File

This is the file where the unstructured grids and their connectivity data (triangulation) is stored. The output file with the same name generated by DELAUNDO can be used. Its format is also compatible with TECPLOT visualization program, so that it can be loaded as data file by TECPLOT. Data structure of this file is as follows:

VARIABLES="*X*","*Y*"…….....**1**

ZONE T="P_1", *F*=FEPOINT, *N*=…, *E*=…, *ET*=TRIANGLE.....**2**

X(1)	*Y(1)*	
X(2)	*Y(2)*	
.	.	
.	.	
X(i)	*Y(i)*	
.	.	
.	.	
X(N)	*Y(N)*	

.....**3**

T(1,1)	*T(1,2)*	*T(1,3)*
T(2,1)	*T(2,2)*	*T(3,3)*
.	.	.
.	.	.
T(j,1)	*T(j,2)*	*T(j,3)*
.	.	.
.	.	.
T(E,1)	*T(E,2)*	*T(E,3)*

.....**4**

1. This line implies that only *X* and *Y* variables are stored in this data file, which are indeed the *X* and *Y* coordinates of each grid point.

2. This part is where some data for TECPLOT is stored as:

***ZONE*:**Any name can be entered here it specifies the block name in TECPLOT. You can load different blocks into one TECPLOT file. Using different names is used to identify each block.

***F*:** This value is the data file format for TECPLOT. It is "FEPOINT" by default. It means that A data file format for finite-element zones in which the node data is listed by point-by-point. All the variable values of the first point are listed first, then the variable values of the second point, and so forth.

***N*:** This is the total number of grid points

***E*:** This is the total number of triangles.

3. These lines store the *X* and *Y* coordinates of each grid point. It starts with *X(1)* and *Y(1)* in the first line which are the *X* and *Y* coordinates of the 1st grid point and ends up with *X(N)* and *Y(N)*, in which *N* is the total number of grid points as mentioned above.

In general:

X(i) *Y(i)*

where $i = 1...N$ where *N* is the total no. of grid points

X(i) = *x* coordinate of i^{th} grid point

Y(i) = *y* coordinate of i^{th} grid point

4. This part is where connectivity information (triangulation) is stored. It mentions which grid points are connected to form each triangle with index of *j*. It is denoted as *T(1,1), T(1,2)* and *T(1,3)* which are the index numbers *i*, mentioned as in part 3. They are three grid nodes, which are connected to form the triangle with index number *j*.

In general:

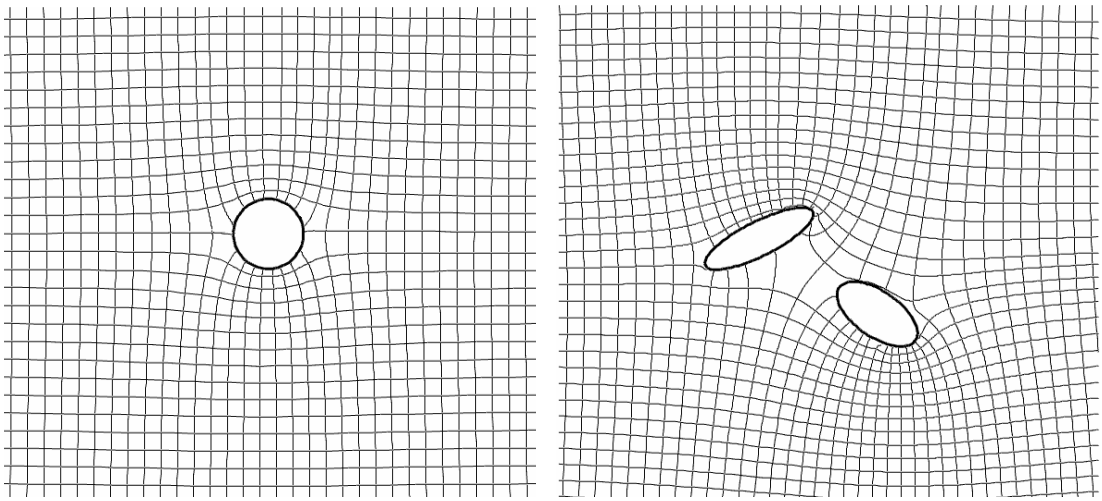
T(j,1) *T(j,2)* *T(j,3)*

where $j = 1...E$ where *E* is the total number of triangles.

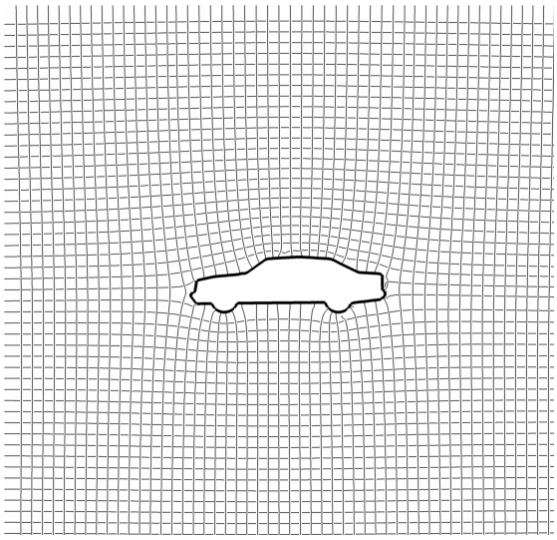
T(j,1)= index number *i*, specified in part 3 as 1st node of j^{th} triangle

T(j,2)= index number *i*, specified in part 3 as 2nd node of j^{th} triangle

T(j,3)= index number *i*, specified in part 3 as 3rd node of j^{th} triangle



(a) (b)



(c)

Fig. B.2 Formation of flow net a) a cylinder, b) two ellipses, c) a car.

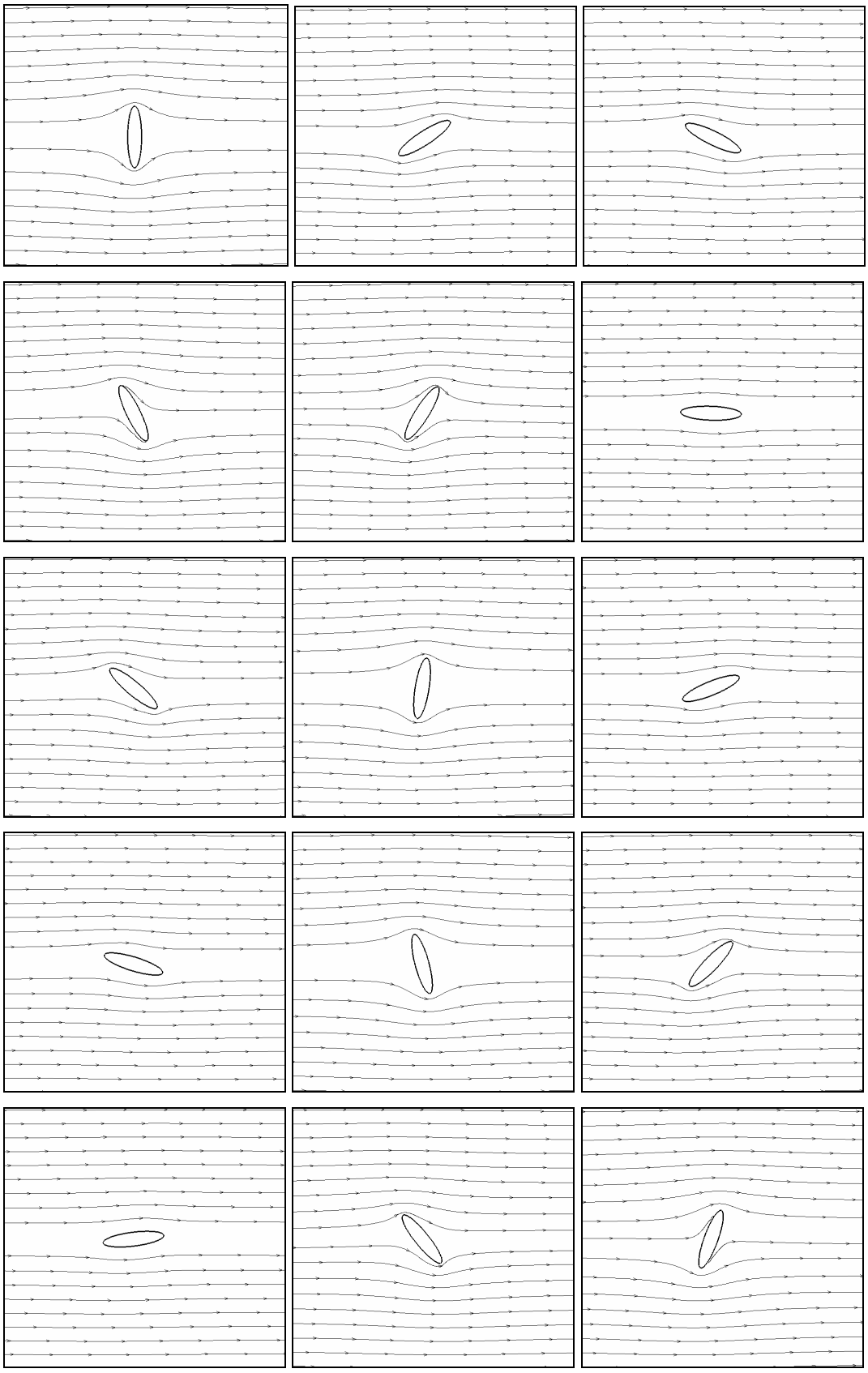


Fig. B.3 Potential flow around a rotating elliptical slender body.