2008

Unsteady incompressible Navier-Stokes simulations on deforming domains

Atefeh Shadpour
University of Windsor

UNSTEADY INCOMPRESSIBLE NAVIER-STOKES SIMULATIONS ON DEFORMING DOMAINS

by

ATEFEH SHADPOUR

A Thesis
Submitted to the Faculty of Graduate Studies
Through the Department of Mechanical, Automotive and Materials Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada

2008

Your file Votre référence
ISBN 978-0-494-82083-4
Our file Notre référence
ISBN 978-0-494-82083-4

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Mesh generation has been an important topic of research for the past four decades, primarily because it is one of the critical elements in the numerical simulation of fluid flows One of the main current issues in this regard is mesh generation and flow solution on domains with moving boundaries In this research a novel scheme has been proposed for mesh generation on domains with moving boundaries, with the location of boundary nodes known at any particular time A new set of linearized equations is derived based on a full nonlinear elliptic grid generation system The basic assumption in deriving these new equations is that each node experiences only a small amount of disturbance when the mesh moves from one time to the next Comparison with grids generated by the full elliptic system shows that this new method can generate high quality grids with significantly less computational cost

Inherently, the flow on such a domain will be unsteady The Navier-Stokes equations for unsteady 2D laminar incompressible flow are expressed in the primitive variables formulation A SIMPLE-like scheme is applied to link the pressure and velocity fields and ensure conservation of mass is satisfied The equations are discretized in a pure finite difference formulation and solved by implicitly marching in time The flow solver is validated against results in the literature for flow through a channel with a moving indentation along one wall

Dedicated to

My Country, Iran

and

My Family

Acknowledgements

My entire master thesis has come to completion through the steady and invaluable support and academic guidance of my supervisor, Dr. R. M. Barron, for which I am eternally grateful. Without him, this thesis could not have been fulfilled. I also wish to express my gratitude to the other members of my master thesis committee, Dr. G. W. Rankin and Dr. R. Carriveau, for their helpful comments.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

b	channel height
c	relative velocity
h_{max}	amplitude of the channel indentation
J	transformation Jacobian
I_o	orthogonality functional
I_{ML}	Modified Liao functional
I_s	smoothness functional
I_w	weighted area/volume functional
M	maximum number of nodes in the y direction
N	maximum number of nodes in the x direction
p	pressure
P,Q	control functions
Re	Reynolds number
S	source term
St	Strouhal number
t	time
t_f	fraction of time (t/T)
T	oscillation period
u,v	velocity components
U_{bulk}	bulk velocity
x,y	Cartesian coordinates

ξ, η	curvilinear coordinates
θ	skew angle
ν	kinematic viscosity
τ	artificial compressibility

Subscripts

i, j	coordinate indices
P	value at a calculation point
E	value at the East side of P
W	value at the West side of P
N	value at the North side of P
S	value at the South side of P
nb	value at neighbour nodes

Superscripts

$*$	iterated values
$'$	corrected values
n	value at time t_n
$n+1$	value at time t_{n+1}

CHAPTER 1 INTRODUCTION

1.1 Grid Generation

Mesh generation is an interdisciplinary area, and researchers from different disciplines, including mathematicians, computer scientists and engineers are working on developing improvements to the existing methodologies. One of the major applications of mesh generation is in computational fluid dynamics. Since the fluid flow equations will be discretized on the mesh that has been generated, the quality of the mesh generally plays a significant role in the accuracy of the results obtained from the numerical solution of the flow equations. One of the main issues of current interest is generating the grid on domains with moving boundaries, referred to as dynamic grid generation. The motivation for grid generation on domains with moving (deforming) boundaries is its application in many fluid mechanics problems such as free surface flow, simulating blood flow in carotid arteries, scour problems (Figure 1.1) and airfoil/wing shape optimization. In general, grid movement may be because of the change of the shape of the boundaries of the domain, or for the purpose of adapting the grid with the physical solution of the problem at hand.

Different methods for grid generation on domain with deforming boundaries have been introduced in the literature. The method proposed here applies an elliptic grid generator for generating the basic (initial) grid. The boundary points are then assumed to be perturbed by very small amounts. A new set of linear equations is obtained which, upon solution, give the x and y displacement of the interior nodes due to the motion of the boundaries. Since the displacement of the grid points is assumed to be very small, the second order terms in the equations can be ignored. For the purpose of avoiding entanglement of the grid in some regions, which is caused by the accumulation of errors after a few successive grids have been generated by the perturbed equations, a full elliptic grid generation can be applied once. The significant decrease in the number of grid generations by the full elliptic equations is the main advantage of this method, saving a noticeable amount of computational time.

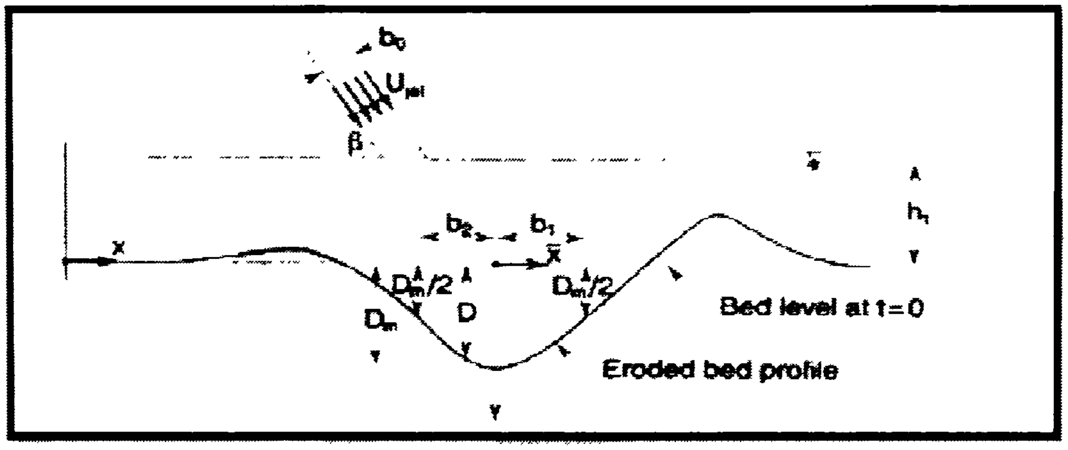

Figure 1.1: Scour by Water Surface Jet, Showing the Deformed Bed

1.2 Flow Solution

Navier-Stokes equations were formulated in the 19th century. They are one of the most useful sets of equations because they describe the physics of a large number of fluid flow phenomena. The Navier-Stokes equations are nonlinear partial differential equations.

They can be expressed in conservative or non-conservative, dimensional or non-dimensional form.

The governing equations for an incompressible flow may be expressed in primitive variable formulation or vorticity-streamfunction formulation. The primitive variable formulation uses velocity and pressure, while the vorticity-streamfunction formulation introduces vorticity and streamfunction to re-formulate the flow equations. The primitive variable formulation is a mixed elliptic-parabolic system of equations and there is no direct link between continuity and momentum equations. To resolve this limitation, two procedures exist, that is, there are two ways to link the continuity and momentum equations. The first method is to apply a Poisson equation for pressure and the second is to introduce artificial compressibility into the continuity equation. The advantage of the primitive variable formulation is that its extension to three dimensions is straightforward. In the vorticity-streamfunction formulation, the Navier-Stokes equations are decoupled into an elliptic and a parabolic equation. Extension of this method to three dimensions is not easy, since a simple streamfunction does not exist in three dimensions.

The focus of this research is on developing a solution algorithm for the unsteady two-dimensional incompressible equations in dimensional, non-conservative form, expressed in primitive variables. The basic idea proposed in this research is to implement a SIMPLE-type algorithm for the pressure field calculation, similar to that used in a finite volume method, whereas the discretized equations are developed as a purely finite difference formulation.

CHAPTER 2 LITERATURE REVIEW

2.1 Introduction

Computational fluid dynamics (CFD) is one of the branches of fluid mechanics that uses numerical methods and algorithms to solve and analyze problems that involve fluid flows. The equations to be solved in many CFD problems are the Navier-Stokes equations, which govern any single-phase fluid flow. The need for grid generation arises when the numerical solution of the Navier-Stokes equations is sought on a non-rectangular physical domain. In this case, the domain is discretized into cells or elements, and the flow equations are approximated on the discretized domain based on the applied numerical method, that is, the finite difference, finite volume, or finite element method.

Finite difference method is a numerical method for approximating the solutions to differential equations using finite difference formulae to approximate derivatives. Values of the flow variables are calculated at discrete nodes on a meshed geometry.

Finite volume method is similar to the finite difference method. But, instead of approximating derivatives, the flow equations are written in integral form and the integrals are approximated. Values are calculated on discrete volumes, surrounding each node, on a meshed geometry. In the finite volume method, volume integrals in a partial differential equation that contain a divergence term are converted to surface integrals, using the divergence theorem.

In the finite element method the solution approach is based on rendering the PDE into an approximating system of ordinary differential equations, by approximating the integrand in terms of some basis functions. These ODEs are then solved using standard techniques such as Euler's method, Runge-Kutta, etc.

The focus of the research in this thesis is on mesh generation for domains with deforming boundaries, and subsequently solving the unsteady Navier-Stokes equations using the finite differencing method.

2.2 Moving Grid Techniques

Grid (or mesh) generation has been around since the appearance of computational fluid dynamics. Grid generation can be viewed as distributing nodes on the physical domain and specifying the connectivity between these nodes. In structured grid generation this connectivity is accomplished by mapping the grid points from the computational domain onto the physical domain. With this definition Brackbill and Saltzman [1] noted that the differential properties of the mapping define the properties of the grid. They introduced three functionals that are measures of the accumulation of different grid properties; namely a measure of spacing between the grid lines (smoothness) $\boldsymbol{I_S}$, a measure of the orthogonality of the grid lines $\boldsymbol{I_O}$, and a measure of the area of the mesh cells $\boldsymbol{I_W}$. Minimizing any combination of these functional produces a grid with good qualities, such as smooth transition in the spacing between grid lines, reduced skewness and cell areas which are not too small. Applying the calculus of variations to minimize these functionals leads to a set of partial differential equations for the positions of the nodes.

If the solution domain has regions with high spatial activity, a fixed grid will be inefficient. In this situation, as the flow field evolves, the regions where the flow gradients are large may shift with time, and thus a good mesh at one time may not be appropriate at some later time. Meshing procedures which allow the nodes to adjust their location to respond to the evolution of the flow are referred to as adaptive techniques. In the realm of adaptive techniques for time-dependent PDEs, one can roughly distinguish between two classes of methods [2], *h*-refinement and *r*-refinement. In the *h*-refinement method the grid is adapted at discrete time levels and the partial differential equations are discretized on a grid which is kept fixed over the entire time level. The *r*-refinement methods have the advantage of moving continuously in time in such a way that the discretization of the PDEs and grid movement are coupled. The number of grid points is usually kept fixed.

Various major moving grid techniques will now be briefly introduced.

Method of Characteristics (MoC): One of the simplest choices is to let the grid move based on the characteristic equations of the PDEs that govern the flow. This method is primarily used for high speed compressible inviscid flows, since the governing equations are hyperbolic and therefore have a set of distinct characteristic curves which move with time and can be used to provide the grid system.

Equidistribution: One of the most widely used methods to move the grid in one dimension is to consider solution of the equation,

$$\frac{\partial}{\partial \xi}\left(\frac{\partial x}{\partial \xi} M\right) = 0 \tag{2.1}$$

where $M > 0$ is called a monitor or control function. Manipulating this equation gives a formula that describes equidistribution

$$\Delta X_{i-1} M_{i-1} = \Delta X_i M_i \; 1 \leq i \leq N-1 \tag{2.2}$$

Moving Finite Differences (MFD): This is an extension of the equidistribution method which generates smoother grid results.

Moving Finite Elements (MFE): This method is a two-dimensional moving grid technique based on the minimization of the PDE residuals, and is obtained by approximating the PDE solution with finite element basis functions.

Lagrangian-Eulerian Methods

A fundamentally important consideration in coping with problems with strong distortion of the continuum is determining the relationship between the deforming continuum and the mesh of the computing domain. Two classical descriptions of motion can be used, the Lagrangian approach and the Eulerian approach. Originally, Arbitrary Lagrangian-Eulerian (ALE) methods were developed by Noh [3], Franck and Lazarus [4], Trulio [5], Hirt et al. [6]. Donea et al. [7] surveyed this approach and the applications in fluid dynamics.

Lagrangian Algorithms:

In Lagrangian based algorithms, widely used in solid mechanics, each node of the computational grid follows the material particle during motion. In this approach the grid points are connected to the same material points at all times.

Eulerian Algorithms:

In algorithms based on the Eulerian approach, commonly used in fluid dynamics, the computational mesh is fixed and the continuum moves with respect to the grid.

Lagrangian-Eulerian Algorithms:

The ALE algorithms allow the nodes of the computational mesh to move as in the Lagrangian fashion or be fixed as in the Eulerian fashion or in some arbitrary way. The advantage of this approach is that greater distortions can be handled with more resolution. The fundamental ALE equation is

$$\frac{\partial f}{\partial t}|_X = \frac{\partial f}{\partial t}|_\chi + \frac{\partial f}{\partial x}.c = \frac{\partial f}{\partial t}|_\chi + c.\nabla f \qquad (2.3)$$

The function f represents a physical quantity for the particle X with the reference coordinate χ held fixed. and c is the relative speed between the material and the reference system.

2.3 Unsteady Incompressible Flow Equations

Generally. the incompressible Navier-Stokes equations are formulated in vorticity-streamfunction formulation or the primitive variables formulation. the variables being velocity components and pressure. Since there is no direct link between velocity and pressure in the continuity and momentum equations for incompressible flow. the major issue in solving the Navier-Stokes equations is pressure-velocity coupling. From a physical point of view. Navier-Stokes equations can be categorized as steady and unsteady. Numerical procedures are essentially the same for both categories. There are two points worth mentioning here. First. the "time" in steady problems is not a physical time, while time has a physical meaning in unsteady problems. Therefore. for unsteady flow simulations, the time step should match the actual physical time increment. and the maximum allowable time step may be determined by the desired accuracy or a stability criterion. If an unsteady solver is used to simulate steady flow problems. the maximum allowable time step is indicated only by the stability criterion. The process of marching in time to a steady-state is closely associated with the iterative solution of the steady flow equations. Secondly. it is important to note that for an unsteady problem the initial conditions are real physical conditions at time = 0, while for a steady problem it is just an initial guess.

2.3.1 Vorticity-Streamfunction Method

The main advantage of the vorticity-streamfunction formulation is that there is no need for pressure-velocity coupling, since the momentum equations are combined to eliminate the pressure gradient terms, yielding an equation for vorticity. Ghia et al. [8] proposed a direct method for the solution of unsteady incompressible flow equations in generalized curvilinear coordinates based on the vorticity-streamfunction formulation. The transport equation was solved by an alternating direction implicit method and the streamfunction equation was solved by direct block Gaussian elimination.

Ewing et al. [9] proposed a fourth-order equation for streamfunction and solved the equations by finite differencing on a uniform grid. A multilevel technique was applied for treating the ill-conditioned system of linear algebraic equations representing the fourth-order system.

There is a large body of literature using the vorticity-streamfunction formulation to test important issues that arise in CFD, such as the capability and limitations of new solution algorithms, effects of meshing, comparisons of different discretization techniques, implementation of boundary conditions, etc. However, extension of this formulation to 3D is extremely cumbersome, and hence it is not useful for practical application to realistic flow situations.

2.3.2 Artificial Compressibility Method

Artificial compressibility method was first introduced by Chorin [10] for steady flows, and extended to unsteady flows by Peyret [11]. This method is still very popular for the simulation of steady incompressible flows, and the evolution of the method has been outlined in [12]

The artificial compressibility scheme involves adding a time derivative of pressure to the continuity equation,

$$\frac{\partial p}{\partial t} + \frac{1}{\tau}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0 \qquad (2\ 4)$$

The quantity τ can be interpreted as artificial compressibility of the fluid The continuity equation along with the momentum equations are solved until convergence to the steady-state is reached, which implies that $\frac{\partial p}{\partial t}$ vanishes, which means divergence of velocity becomes zero Merkle and Athavale [13] extended the application of artificial compressibility method to three-dimensional unsteady calculations Tang and Sotiropoulos [14] introduced a fractional step artificial compressibility method for unsteady, three-dimensional, incompressible Navier-Stokes equations This approach applies the ideas from both the standard, dual-time stepping artificial compressibility iteration scheme and pressure-based fractional step formulations To obtain time-accurate solutions, pseudo-time derivatives of pressure and velocity fields are introduced into the continuity and momentum equations, respectively The resulting system is iterated until convergence is reached at each time step The CPU time per physical time step for artificial compressibility methods is considerably large This deficiency is somewhat overcome by applying fractional step methods In fractional step methods, an intermediate velocity field is calculated first This velocity field is projected into the divergence free space by solving a Poisson equation.

2.3.3 SIMPLE Method

SIMPLE stands for Semi-Implicit Method for Pressure Linked Equations, introduced by Patankar and Spalding [15] in 1972. In this method, a pressure field is guessed. The momentum equations are solved but the resulting velocity field does not satisfy the continuity equation due to the inaccurate pressure guess. The pressure and velocity fields are then corrected successively until a converged solution is obtained. The words semi-implicit in the name SIMPLE has been used to acknowledge the omission of the summation of velocity corrections of neighbouring points. This represents an indirect influence of the pressure correction on velocity.

Van Doormaal and Raithby [16] have introduced some enhancements for the SIMPLE method. As mentioned before, the summation of velocity corrections at neighbouring points is neglected from the velocity correction equations in the SIMPLE method. In an effort to enhance the SIMPLE method, Doormaal and Raithby suggest that this term can be subtracted from both sides of the equation and hence a new velocity correction equation is obtained. This method is called SIMPLE Consistent (SIMPLEC). Setting the pressure correction to be zero where pressure is specified is another recommendation made by them.

Raithby and Schneider [17] introduced a method of treating the velocity-pressure coupling for internal flows that are parabolic in one coordinate direction. They introduced the following equation, where subscript P denotes the point at which differencing is done,

$$Q = -\frac{\partial p}{\partial y} \text{ and } f_p = \frac{\partial v_P}{\partial Q}$$

The momentum equations are solved, which gives v_P^*, and consequently f_P is calculated. Then ΔQ, which is chosen in a way to make the total mass flow rate correct, is calculated, followed by a correction to the velocity field.

$$v_P = v_P^* + f_P \Delta Q$$

2.4 Unsteady Flow Solution on Domains with Moving Boundaries

Whilst many solutions of the unsteady Navier-Stokes equations have been described, there are only a limited number with moving boundaries [18]. Viecelli [19] applied the marker-and-cell method to solve free-surface problems. Peskin [20, 21] modeled cardiac flows. He replaced the boundary movement by a distribution of forces that satisfy the boundary conditions. Daly [22] used a mixed Lagrangian-Eulerian approach for studying the pressure distribution in flexible tubes.

Ralph and Pedley [18] modeled the flow in a channel with a moving indentation using the vorticity-streamfunction formulation and the finite difference method. They incorporated the boundary movement into the flow equations by applying a time-dependent transformation in such a way that the computational domain remains a fixed rectangle. Since they applied an explicit scheme, they had to use small time steps to preserve stability.

In 1990, Demirdzic and Peric [23] simulated the same problem by integrating the governing equations for an arbitrary moving control volume, with pressure and Cartesian velocity components as dependent variables. Since they used fully implicit temporal differencing the method is stable at all time steps. SIMPLE algorithm was used for pressure-velocity coupling.

Strigberger [24] proposed a perturbation method to solve the Poisson equation, which is widely used in the computation of unsteady incompressible Navier-Stokes equations, on a moderately deforming grid.

Wu and Rath [25] proposed a finite difference solution of incompressible Navier-Stokes equations for flows with rotation and moving boundary on a nonstaggered grid, based on a SIMPLE-like method.

CHAPTER 3

DYNAMIC GRID GENERATOR

3.1 Goals of Grid Generation

Numerical grid generation arises from the need to discretize the partial differential equations of fluid dynamics in order to compute their solutions on physical regions with complex geometry [26].

A mesh is a discretization of a geometric domain into small simple shapes called elements or cells. Numerical grid generation, in terms of geometry, is categorized as structured or unstructured.

Structured Grids:

Structured grid generation has its roots in the U.S. with the work of Winslow and Crowley at Lawrence Livermore National Lab in the late 1960s and in Russia by Godunov and Prokopov at about the same time [26]. Structured numerical grid generation is an algorithm procedure that orderly distributes a finite number of computational nodes over a physical field in such a way that some coordinate (grid) lines are coincident with each segment of the boundary of the physical domain [27]. Structured grids can be generated algebraically or as the solution of PDEs [2].

A structured mesh is characterized by regular connectivity that can be expressed as a two- or three-dimensional array. This restricts the element choices to quadrilaterals in 2D or hexahedral in 3D. Some advantages of structured meshes that generally hold for most applications are simplicity, availability of code, and suitability for multigrid and finite difference methods. From a flow solution point of view, structured grids usually allow more effective grid clustering in regions of high flow gradients, such as boundary layers or shock regions, thereby improving the accuracy of the numerical solution. The main disadvantage is that it is often difficult to construct a good structured mesh in highly irregular domains. This problem is somewhat alleviated by using multiblock structured grids.

Unstructured Grids:

An unstructured mesh is characterized by irregular connectivity which is not readily expressed as a two- or three-dimensional array in computer memory. This allows for any possible element that a flow solver might be able to use. Compared to structured meshes, the storage requirements for an unstructured mesh can be substantially larger since the neighbouring connectivity must be explicitly stored. A hybrid mesh is a mesh that contains structured portions and unstructured portions and, as such, could be classified as a special case of a totally unstructured mesh.

The main advantages of unstructured meshes are that they conform to the flow domain more easily and allow element sizes to vary more dramatically.

3.2 Dynamic Grid Generation

One of the areas of research interest has been grid generation as a function of time, generally referred to as dynamic grid generation. In other words, grid points are allowed to move within the flow domain as the solution proceeds. Grid movement can be because of the change of the shape of the boundaries of the domain. or for the purpose of adapting the grid with the physical solution of the problem at hand. The first scenario usually occurs for unsteady flow problems, where forces are acting on the domain boundaries, causing them to move or deform. The second case occurs in problems in which the grid points adjust themselves in response to the development of the solution of the physical problem being solved on the grid, known as adaptive gridding. The goal is to concentrate the grid points in the regions where the gradients of the physical variables of the problem are large.

There are some considerations to be taken into account in dynamic mesh generation. A major issue is that the grid generation algorithm should be done in a way which ensures that no region will be void of points. For practical applications, it is also essential to keep the computational time spent on the grid generation as low as possible.

3.3 Grid Generation Techniques

The use of numerical techniques to generate curvilinear grids is considered to be one of the most important enabling technologies in computational fluid dynamics [28]. These techniques can be divided into three categories, conformal mapping system, algebraic systems, and partial differential equations systems. Conformal mappings are limited to 2D problems and require knowledge of complex variables, whereas algebraic and PDE methods are applicable in either 2D or 3D. Algebraic grid generation involves computing the nodal coordinates by interpolation of the boundary points. Different interpolation functions can be used for this purpose, the most popular method being transfinite interpolation. Partial differential equations systems can be elliptic, parabolic or hyperbolic. There are several important considerations when using partial differential equations to generate the grid system:

- Most numerical solution algorithms for PDEs must be applied on grids describing the physical domain of interest

- Accurate application of boundary conditions requires that grid lines coincide with physical boundaries This also tends to reduce the logic in computer codes applying the numerical techniques
- Solution accuracy and finite computational resources require grids to be concentrated in regions where there are high gradients of the flow variables

3.3.1 Transformation of Coordinates

The primary goal of transforming the physical coordinates to a simpler computational region is to remove the complexity of the shape of the physical region The computational domain is also referred to as the logical domain An effective approach to solving PDEs with complex boundary geometry and different scales of motion in the solution domain is to transform the physical domain and equations of motion to an idealized rectangular computational domain, where a flow solution algorithm can be applied In the terminology of transformations, the Jacobian of the transformation is required to be nonzero to ensure that the transformation is invertible (Figure 3 1) A rectangular grid system is first generated in the computational space and then mapped to physical space Transformations containing a point with zero Jacobian are called folded, avoiding a folded transformation is a major objective of grid generation algorithms Also, it is well known that the error in the approximations of the flow equations depends not only on the derivatives of the solution but also on the rate of change of grid spacing and the departure from orthogonality [29]

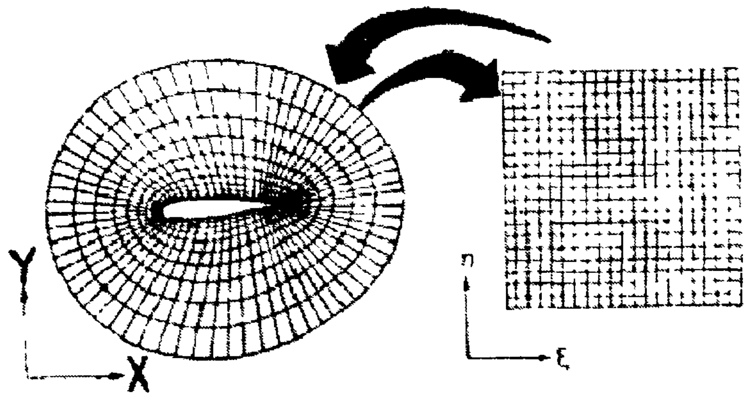

Figure 3.1 Grid System in 2D Physical and Computational Space

3.3.2 Elliptic Grid Generation

Elliptic grid generation, based on solving a system of elliptic PDEs, is the most commonly used method for generating a structured curvilinear grid on domains with known boundary. One of the advantages of an elliptic grid generator is that it will generate a smooth grid. Smoothness of the grid, which provides a measure of spacing between the grid lines, is an important factor when solving the flow equations on that domain. For 2D elliptic grid generation, suppose that ξ and η are the two coordinates which define the curvilinear coordinate system: x and y define the rectangular coordinate system. The goal is to generate a distribution of points such that ξ and η obtain their maximum and minimum values on the boundaries and change monotonically in the interior. This can be achieved by solving the elliptic Poisson equations

$$\xi_{xx} + \xi_{yy} = P(\xi,\eta)$$

$$\eta_{xx} + \eta_{yy} = Q(\xi,\eta) \qquad (3.1)$$

where $P(\xi,\eta)$ and $Q(\xi,\eta)$ are control functions, to be specified by the user in order to achieve some desirable characteristics for the resulting grid system, such as clustering around some point or line, or orthogonality near the boundaries.

Grid metrics are calculated as

$$g_{11} = x_\xi^2 + y_\xi^2$$

$$g_{12} = x_\xi x_\eta + y_\xi y_\eta$$

$$g_{22} = x_\eta^2 + y_\eta^2 \tag{3.2}$$

and the Jacobian is given by $J = x_\xi y_\eta - x_\eta y_\xi$

It is not convenient to solve Equations (3.1), since they are formulated in the physical space where the domain is irregular. Hence, these equations are transformed to the computational space, taking the form

$$g_{22}x_{\xi\xi} - 2g_{12}x_{\xi\eta} + g_{11}x_{\eta\eta} = -J^2(Px_\xi + Qx_\eta)$$

$$g_{22}y_{\xi\xi} - 2g_{12}y_{\xi\eta} + g_{11}y_{\eta\eta} = -J^2(Py_\xi + Qy_\eta) \tag{3.3}$$

These are two nonlinear coupled elliptic PDEs for functions $x(\xi,\eta)$ and $y(\xi,\eta)$.

The straight boundaries in the computational domain must be the images of the curved boundaries of the physical region. Discretization of this set of equations must conform to the boundaries of the region. The boundary conditions for Equations (3.3), which are solved on the rectangular computational domain, are the known boundary values in the physical domain, as illustrated in Figure 3.2.

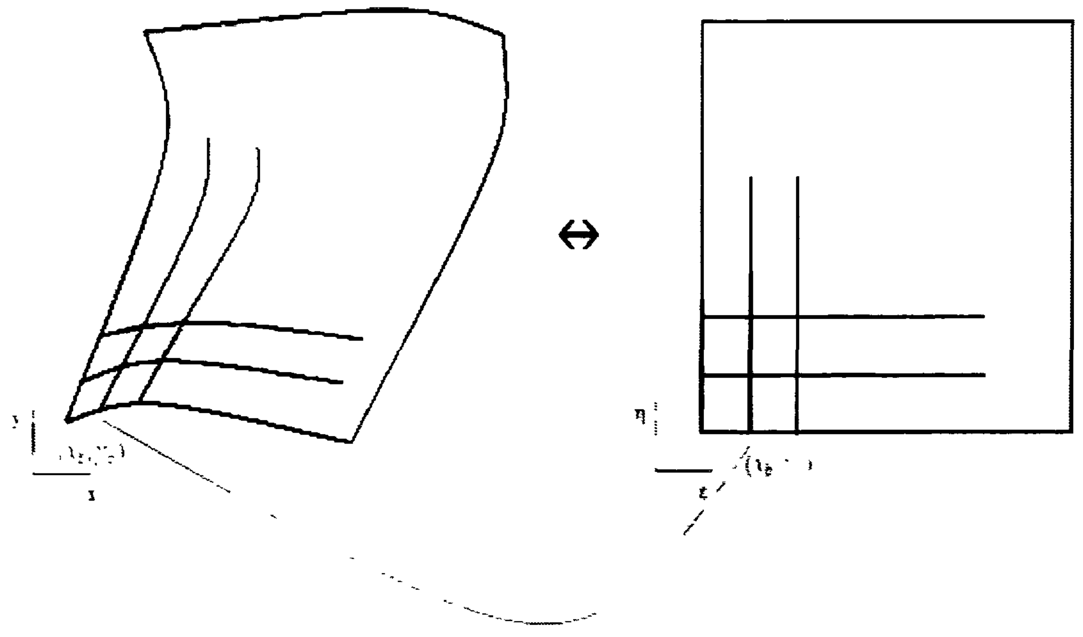

Figure 3.2: Boundary Conditions for Elliptic Grid Generation

3.3.3 Control Functions

Control functions are functions used to control the specific distribution of grid points in the domain. In the absence of control functions, i.e., when $P = Q = 0$, the generation system tends to produce the smoothest possible uniform grid, with a tendency for grid lines to concentrate over convex boundary regions and to spread over concave regions [2]. Generally speaking, the P function controls spacing in the ξ direction and the Q function controls spacing in the η direction. Various forms of these control functions have been suggested in the literature [2, 30].

3.4 Perturbation Method for Dynamic Grid Generation

For a static grid, the Cartesian coordinates x and y are functions of ξ and η. For a dynamic grid, x and y are also function of time t. In general, the grid movement could be caused by changes in the boundaries of the physical domain, or by requiring the grid to adapt to the development of the flow. In this section we devise a technique that is applicable to mesh generation on domains for which the boundaries of the domain are gradually deforming. The boundary shape is allowed to deform slightly over one time step Δt, from one time t_n to the next time t_{n+1}. Knowing the boundary nodes displacements from one time to the next, the aim is to obtain the location of the interior nodes at the new time. Elliptic grid generation method is used as the fundamental tool for the grid generation.

Assume that an interior point with coordinates (x, y) at time t_n moves to a new location $(x + x', y + y')$ at time t_{n-1}. Elliptic grid generation, Equations (3.3), is used as the basis to generate the grid at time t_{n-1}. However, the system of equations is simplified by assuming that x' and y' are small, resulting in a new system of linear equations which, upon solution, gives the amounts of perturbation of all interior nodes. The elliptic system (3.3) at time t_{n+1} takes the form

$$g_{22}^{n+1}(x_{\xi\xi} + x'_{\xi\xi}) - 2g_{12}^{n+1}(x_{\xi\eta} + x'_{\xi\eta}) + g_{11}^{n+1}(x_{\eta\eta} + x'_{\eta\eta})$$
$$= -(J^{n+1})^2(P^{n+1}(x_\xi + x'_\xi) + Q^{n+1}(x_\eta + x'_\eta)) \tag{3.4}$$

$$g_{22}^{n+1}(y_{\xi\xi} + y'_{\xi\xi}) - 2g_{12}^{n+1}(y_{\xi\eta} + y'_{\xi\eta}) + g_{11}^{n+1}(y_{\eta\eta} + y'_{\eta\eta})$$
$$= -(J^{n+1})^2(P^{n+1}\left(y_\xi + y'_\xi\right) + Q^{n+1}(y_\eta + y'_\eta)) \tag{3.5}$$

where superscript n+1 indicates the quantity is evaluated at time t_{n+1}. Using Equations (3.2) and neglecting quadratic terms in the small quantities x' and y', the grid metrics and the Jacobian of the transformation are approximated as follows:

$$g_{11}^{n+1} = \left(x_\xi + x'_\xi\right)^2 + \left(y_\xi + y'_\xi\right)^2 = x_\xi^2 + \cancel{x'^2_\xi} + 2x_\xi x'_\xi + y_\xi^2 + \cancel{y'^2_\xi} + 2y_\xi y'_\xi$$
$$\approx x_\xi^2 + y_\xi^2 + 2x_\xi x'_\xi + 2y_\xi y'_\xi \tag{3.6}$$

$$g_{12}^{n+1} = (x_\xi + x'_\xi)(x_\eta + x'_\eta) + (y_\xi + y'_\xi)(y_\eta + y'_\eta)$$

$$= x_\xi x_\eta + x_\xi x'_\eta + x'_\xi x_\eta + \cancel{x'_\xi x'_\eta} + y_\xi y_\eta + y_\xi y'_\eta + y'_\xi y_\eta + \cancel{y'_\xi y'_\eta}$$

$$\approx x_\xi x_\eta + y_\xi y_\eta + x_\xi x'_\eta + x'_\xi x_\eta + y_\xi y'_\eta + y'_\xi y_\eta \qquad (3.7)$$

$$g_{22}^{n+1} = (x_\eta + x'_\eta)^2 + (y_\eta + y'_\eta)^2 = x_\eta^2 + \cancel{x'^2_\eta} + 2x_\eta x'_\eta + y_\eta^2 + \cancel{y'^2_\eta} + 2y_\eta y'_\eta$$

$$\approx x_\eta^2 + y_\eta^2 + 2x_\eta x'_\eta + 2y_\eta y'_\eta \qquad (3.8)$$

$$(J^{n+1})^2 = (J^n + J')^2 \approx (J^n)^2 + \cancel{J'^2} + 2J^n J' \qquad (3.9)$$

The term J can be evaluated in terms of derivatives.

$$J^{n+1} = (x_\xi + x'_\xi)(y_\eta + y'_\eta) - (x_\eta + x'_\eta)(y_\xi + y'_\xi)$$

$$\approx \underbrace{x_\xi y_\eta - x_\eta y_\xi}_{J^n} + \underbrace{x_\xi y'_\eta + x'_\xi y_\eta - x_\eta y'_\xi - x'_\eta y_\xi}_{J'} \qquad (3.10)$$

Therefore,

$$J' = x_\xi y'_\eta + x'_\xi y_\eta - x_\eta y'_\xi - x'_\eta y_\xi \qquad (3.11)$$

Substitution of these grid metrics and the Jacobian into the elliptic grid generation system, Equation (3.4) results in the following equation for $\boldsymbol{x}$ at time t_{n-1}

L.H.S. =

$$(x_\eta^2 + y_\eta^2)x_{\xi\xi} + 2(x_\eta x'_\eta + y_\eta y'_\eta)x_{\xi\xi} + (x_\eta^2 + y_\eta^2)x'_{\xi\xi} - 2(x_\xi x_\eta + y_\xi y_\eta)x_{\xi\eta}$$

$$- 2(x_\xi x'_\eta + x_\eta x'_\xi + y_\xi y'_\eta + y_\eta y'_\xi)x_{\xi\eta} - 2(x_\xi x_\eta + y_\xi y_\eta)x'_{\xi\eta}$$

$$+ (x_\xi^2 + y_\xi^2)x_{\eta\eta} + 2(x_\xi x'_\xi + y_\xi y'_\xi)x_{\eta\eta} + (x_\xi^2 + y_\xi^2)x'_{\eta\eta}$$

$$= \boldsymbol{(x_\eta^2 + y_\eta^2)x_{\xi\xi} - 2(x_\xi x_\eta + y_\xi y_\eta)x_{\xi\eta} + (x_\xi^2 + y_\xi^2)x_{\eta\eta}} + 2(x_\eta x'_\eta + y_\eta y'_\eta)x_{\xi\xi} +$$

$$2(x_\xi x'_\xi + y_\xi y'_\xi)x_{\eta\eta} - 2(x_\xi x'_\eta + x_\eta x'_\xi + y_\xi y'_\eta + y_\eta y'_\xi)x_{\xi\eta} + (x_\eta^2 + y_\eta^2)x'_{\xi\xi} +$$

$$(x_\xi^2 + y_\xi^2)x'_{\eta\eta} - 2(x_\xi x_\eta + y_\xi y_\eta)x'_{\xi\eta} \qquad (3.12)$$

and R.H.S. =

$$((J^n)^2 + 2J^nJ')\left[(P^n + P')(x_\xi + x'_\xi) + (Q^n + Q')(x_\eta + x'_\eta)\right]$$

$$= -((J^n)^2 + 2J^nJ')\left[P^n x_\xi + P^n x'_\xi + P'x_\xi + \cancel{P'x'_\xi} + Q^n x_\eta + Q^n x'_\eta + Q'x_\eta + \cancel{x'_\eta Q'}\right]$$

$$\approx -(\boldsymbol{J^n})^2(\boldsymbol{P^n x_\xi + Q^n x_\eta}) - (J^n)^2(P^n x'_\xi + P'x_\xi + Q^n x'_\eta + Q\,x_\eta) + 2J^nJ'(P^n x_\xi + Q^n x_\eta) \qquad (3.13)$$

where we have written $P^{n+1} = P^n + P$ and $Q^{n+1} = Q^n + Q$

The bold terms on the L.H.S. and R.H.S. define the basic full elliptic grid generation equation, which has been satisfied at the previous time and therefore are canceled out. Hence, using (3.11) and collecting all the x' terms on the left, the equation which should be solved for $x\ (\xi,\eta)$ is,

$$(x_\eta^2 + y_\eta^2)x'_{\xi\xi} - 2(x_\xi x_\eta + y_\xi y_\eta)x'_{\xi\eta} + (x_\xi^2 + y_\xi^2)x'_{\eta\eta} + 2(x_\xi x_{\eta\eta} - x_\eta x_{\xi\eta})x'_\xi + 2(x_\eta x_{\xi\xi} - x_\xi x_{\xi\eta})x'_\eta + (J^n)^2(P^n x'_\xi + Q^n x'_\eta) + 2J^n(P^n x_\xi + Q^n x_\eta)(y_\eta x'_\xi - y_\xi x'_\eta) = -2y_\eta y'_\eta x_{\xi\xi} - 2y_\xi y'_\xi x_{\eta\eta} + 2y_\eta y'_\xi x_{\xi\eta} + 2y_\xi y'_\eta x_{\xi\eta} - (J^n)^2(x_\xi P + x_\eta Q') - 2J^n(P^n x_\xi + Q^n x_\eta)(x_\xi y'_\eta - x_\eta y'_\xi) \qquad (3.14)$$

The derivatives of x and y on the R.H.S. are known values from the grid at the previous time, and the derivatives of y' will be taken as known from the previous iteration during the iterative solution process. This equation is a linear elliptic PDE for the displacement of the x-coordinate of the grid nodes. Note that this equation also involves the known displacement of the y-coordinate, i.e. y', from the previous iteration.

A similar equation applies for $y'(\xi,\eta)$.

$$(x_\eta^2 + y_\eta^2)y'_{\xi\xi} - 2(x_\xi x_\eta + y_\xi y_\eta)y'_{\xi\eta} + (x_\xi^2 + y_\xi^2)y'_{\eta\eta} + 2(y_\xi y_{\eta\eta} - y_\eta y_{\xi\eta})y'_\xi + 2(y_\eta y_{\xi\xi} - y_\xi y_{\xi\eta})y'_\eta + (J^n)^2\left(P^n y'_\xi + Q^n y'_\eta\right) + 2J^n(P^n y_\xi + Q^n y_\eta)\left(x_\xi y'_\eta -\right.$$

$$x_\eta y'_\xi) = -2x_\eta x'_\eta y_{\xi\xi} - 2x_\xi x'_\xi y_{\eta\eta} + 2x_\eta x'_\xi y_{\xi\eta} + 2x_\xi x'_\eta y_{\xi\eta} - (J^n)^2(y_\xi P' + y_\eta Q') -$$

$$2J^n(P^n y_\xi + Q^n y_\eta)(y_\xi x'_\eta - y_\eta x'_\xi) \qquad (3.15)$$

The x and y derivatives on the R.H.S. are known values from the grid at the previous time and the x' derivatives are known from the previous iteration.

The amount of perturbation of the boundaries, which defines the boundary conditions on x' and y', is known prior to the solution of Equations (3.14) and (3.15). Figure 3.3 depicts this concept.

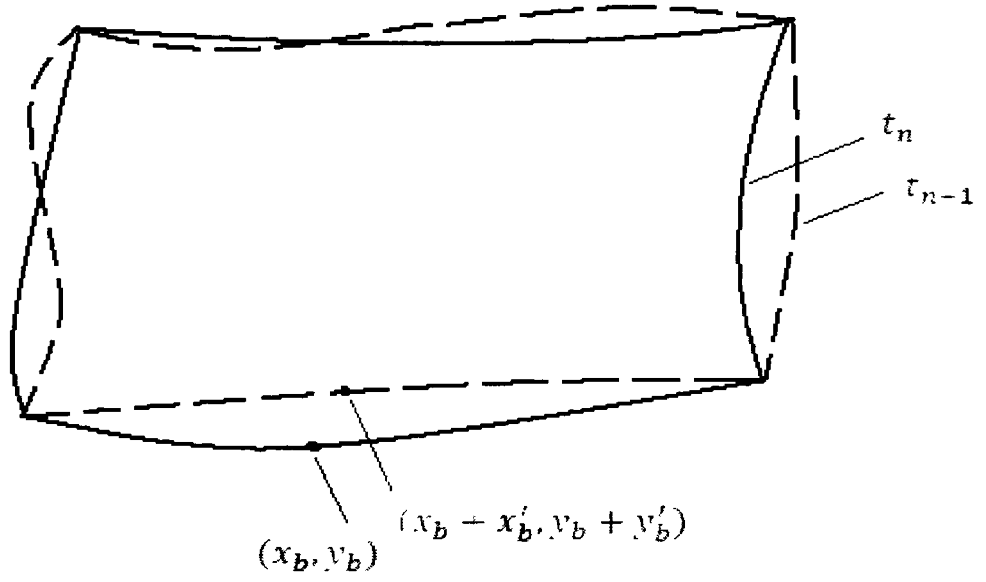

Figure 3.3: Illustration of Boundary Conditions for Domains with Moving Boundaries

Hence, we use an iterative procedure to solve for x' and y' Second order central differencing is applied for all derivatives of x' on the L.H.S. The resulting finite difference equation at (i, j) will take the form,

$$\left(\frac{x_\xi^2+y_\xi^2}{\Delta\eta^2} + \frac{2x_\eta x_{\xi\xi}}{2\Delta\eta} - \frac{2x_\xi x_{\xi\eta}}{2\Delta\eta} + \frac{J^2 Q}{2\Delta\eta} - \frac{2J(Px_\xi+Qx_\eta)y_\xi}{2\Delta\eta}\right) x'_{i,j+1} + \left(\frac{x_\xi^2+y_\xi^2}{\Delta\eta^2} - \frac{2x_\eta x_{\xi\xi}}{2\Delta\eta} + \frac{2x_\xi x_{\xi\eta}}{2\Delta\eta} -\right.$$

$$\frac{J^2Q}{2\Delta\eta} + \frac{2J(Px_\xi+Qx_\eta)y_\xi}{2\Delta\eta}\Bigg) x'_{i,j-1} + \left(\frac{x_\eta^2+y_\eta^2}{\Delta\xi^2} + \frac{2x_\xi x_{\eta\eta}}{2\Delta\xi} - \frac{2x_\eta x_{\xi\eta}}{2\Delta\xi} + \frac{J^2P}{2\Delta\xi} + \frac{2J(Px_\xi+Qx_\eta)y_\eta}{2\Delta\xi}\right) x'_{i+1,j} +$$

$$\left(\frac{x_\eta^2+y_\eta^2}{\Delta\xi^2} - \frac{2x_\xi x_{\eta\eta}}{2\Delta\xi} + \frac{2x_\eta x_{\xi\eta}}{2\Delta\xi} - \frac{J^2P}{2\Delta\xi} - \frac{2J(Px_\xi+Qx_\eta)y_\eta}{2\Delta\xi}\right) x'_{i-1,j} + \left(-2\frac{x_\xi^2+y_\xi^2}{\Delta\eta^2} - 2\frac{x_\eta^2+y_\eta^2}{\Delta\xi^2}\right) x_{i,j} +$$

$$\left(-2\frac{x_\xi x_\eta+y_\xi y_\eta}{4\Delta\xi\Delta\eta}\right) x'_{i+1,j+1} + \left(2\frac{x_\xi x_\eta+y_\xi y_\eta}{4\Delta\xi\Delta\eta}\right) x'_{i+1,j-1} + \left(2\frac{x_\xi x_\eta+y_\xi y_\eta}{4\Delta\xi\Delta\eta}\right) x'_{i-1,j+1} +$$

$$\left(-2\frac{x_\xi x_\eta+y_\xi y_\eta}{4\Delta\xi\Delta\eta}\right) x'_{i-1,j-1} = -2y_\eta {y'}_\eta x_{\xi\xi} - 2y_\xi {y'}_\xi x_{\eta\eta} + 2y_\eta {y'}_\xi x_{\xi\eta} + 2y_\xi {y'}_\eta x_{\xi\eta} -$$

$$J^2(x_\xi P' + x_\eta Q') - 2J(Px_\xi + Qx_\eta)(x_\xi {y'}_\eta - x_\eta {y'}_\xi) \qquad (3.16)$$

where the coefficients are known from the previous grid (grid at the previous time) and evaluated at (i,j). Point-Successive-Over-Relaxation method is applied for solving the above equation. Solving the Equation (3.16) iteratively will give the amount of displacement of interior nodes, $x'_{i,j}$.

A similar equation applies for $y'_{i,j}$. central differencing is applied for the derivatives of y' on the L.H.S. The resulting finite difference equation will take the form

$$\left(\frac{x_\xi^2+y_\xi^2}{\Delta\eta^2} + \frac{2y_\eta y_{\xi\xi}}{2\Delta\eta} - \frac{2y_\xi y_{\xi\eta}}{2\Delta\eta} + \frac{J^2Q}{2\Delta\eta} - \frac{2J(Py_\xi+Qy_\eta)x_\xi}{2\Delta\eta}\right) y'_{i,j+1} + \left(\frac{x_\xi^2+y_\xi^2}{\Delta\eta^2} - \frac{2y_\eta y_{\xi\xi}}{2\Delta\eta} + \frac{2y_\xi y_{\xi\eta}}{2\Delta\eta} +\right.$$

$$\left. -\frac{J^2Q}{2\Delta\eta} + \frac{2J(Py_\xi+Qy_\eta)x_\xi}{2\Delta\eta}\right) y'_{i,j-1} +$$

$$\left(\frac{x_\eta^2+y_\eta^2}{\Delta\xi^2} + \frac{2y_\xi y_{\eta\eta}}{2\Delta\xi} - \frac{2y_\eta y_{\xi\eta}}{2\Delta\xi} + + \frac{J^2P}{2\Delta\xi} + \frac{2J(Py_\xi+Qy_\eta)x_\eta}{2\Delta\xi}\right) y_{i+1,j} + \left(\frac{x_\eta^2+y_\eta^2}{\Delta\xi^2} - \frac{2y_\xi y_{\eta\eta}}{2\Delta\xi} + \frac{2y_\eta y_{\xi\eta}}{2\Delta\xi} -\right.$$

$$\left.\frac{J^2P}{2\Delta\xi} - \frac{2J(Px_\xi+Qx_\eta)x_\eta}{2\Delta\xi}\right) y'_{i-1,j} + \left(-2\frac{x_\xi^2+y_\xi^2}{\Delta\eta^2} - 2\frac{x_\eta^2+y_\eta^2}{\Delta\xi^2}\right) y'_{i,j} + \left(-2\frac{x_\xi x_\eta+y_\xi y_\eta}{4\Delta\xi\Delta\eta}\right) y'_{i+1,j+1} +$$

$$\left(2\frac{x_\xi x_\eta+y_\xi y_\eta}{4\Delta\xi\Delta\eta}\right) y'_{i+1,j-1} + \left(2\frac{x_\xi x_\eta+y_\xi y_\eta}{4\Delta\xi\Delta\eta}\right) y'_{i-1,j+1} + \left(-2\frac{x_\xi x_\eta+y_\xi y_\eta}{4\Delta\xi\Delta\eta}\right) y'_{i-1,j-1} =$$

$$-2y_\eta {y'}_\eta x_{\xi\xi} - 2y_\xi {y'}_\xi x_{\eta\eta} + 2y_\eta {y'}_\xi x_{\xi\eta} + 2y_\xi {y'}_\eta x_{\xi\eta} - (J^n)^2(x_\xi P' + x_\eta Q') -$$

$$2J(P^n x_\xi + Q^n x_\eta)(x_\xi {y'}_\eta - x_\eta {y'}_\xi)$$

(3.17)

Point-Successive-Over-Relaxation is applied for solving Equation (3.17) for $y'_{i,j}$.

The solution algorithm is to solve for $x'_{i,j}$ and $y'_{i,j}$ respectively point by point until the whole domain is swept once. This procedure, sweeping across the domain point-by-point, continues until convergence.

Whenever a grid point is about to fall outside of the domain, the grid is regenerated by full elliptic grid generation once and then the perturbed method is continued.

3.5 Results

The perturbation method described in section 3.4 is applied on different geometries and the grid qualities are compared to the grid quality of the grid generated by the full elliptic grid generation system. Note that in the following cases T represents the period of oscillation.

Case I: Grid Generation on a Domain with a Moving Indentation ($0 \leq t \leq T$)

Figures 3.5 to 3.8 demonstrate that the grids generated by the linear perturbation method and the full elliptic equations are almost identical. The overall quality of these grids will be discussed in section 3.5.1.

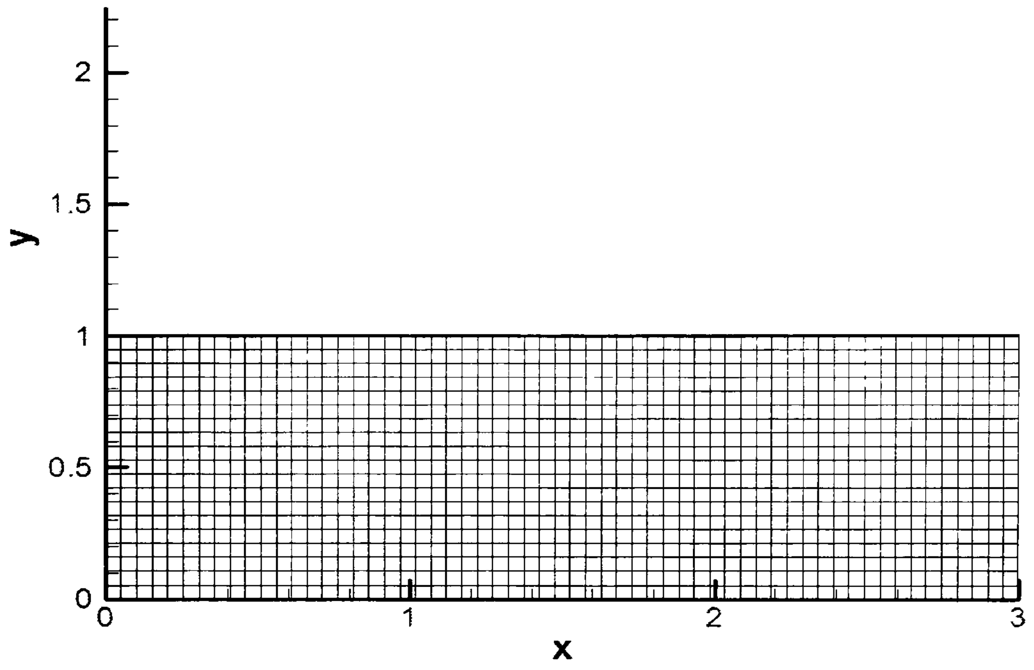

Figure 3.4: Grid Generated by Full Elliptic System at $t = 0$ (Initial Grid)

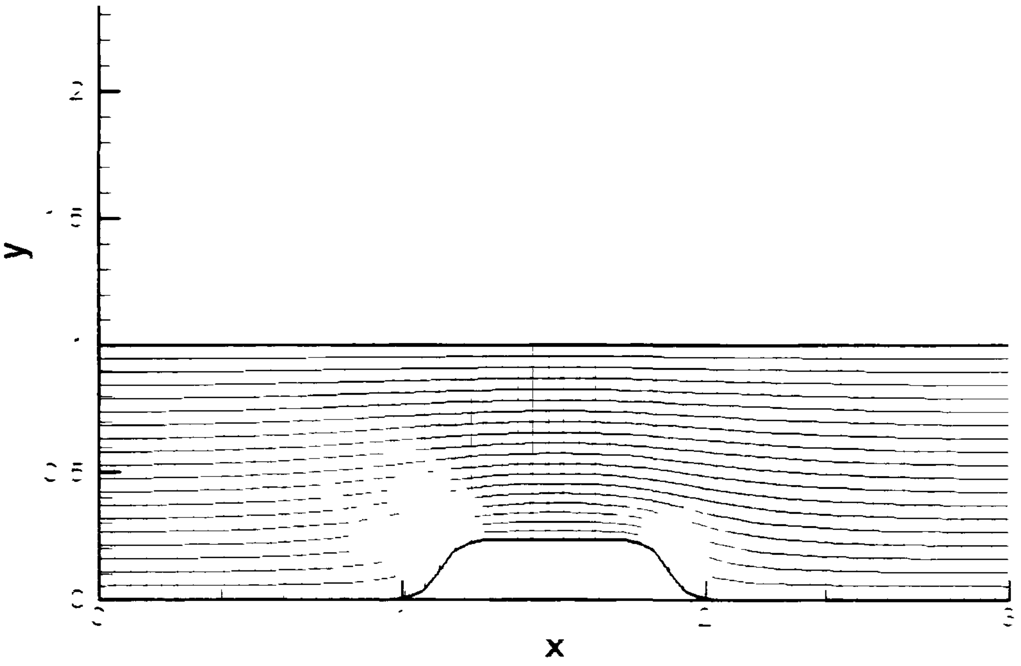

Figure 3.5(a): Grid Generated by Perturbation Method at $t = \frac{T}{4}$

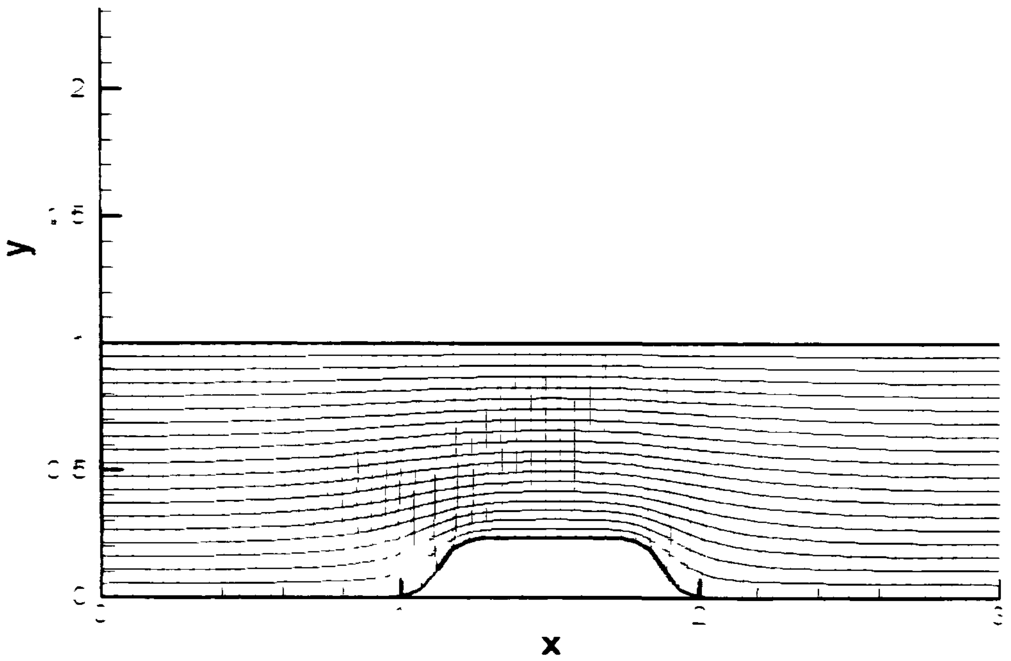

Figure 3.5(b): Grid Generated by Full Elliptic System at $t = \frac{T}{4}$

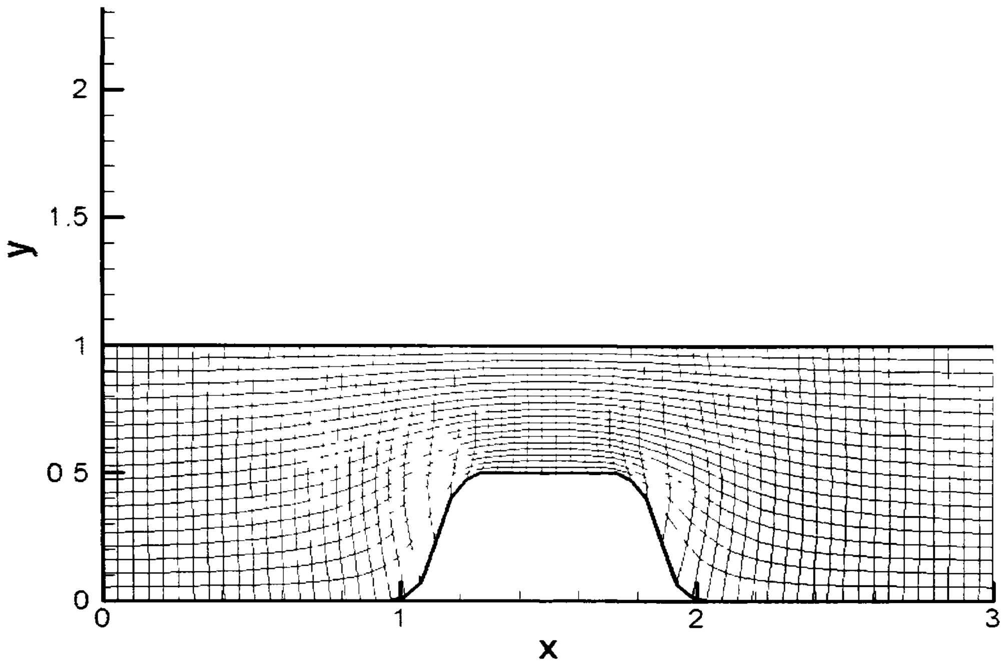

Figure 3.6(a): Grid Generated by Perturbation Method at $t = \frac{T}{2}$

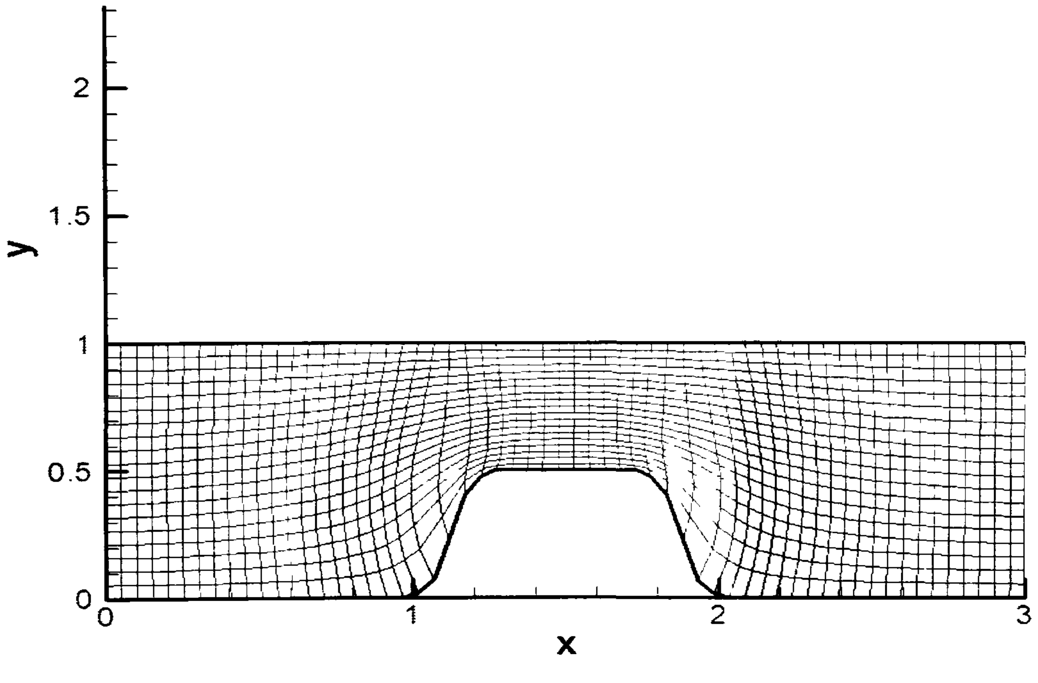

Figure 3.6(b): Grid Generated by Full Elliptic System at $t = \frac{T}{2}$

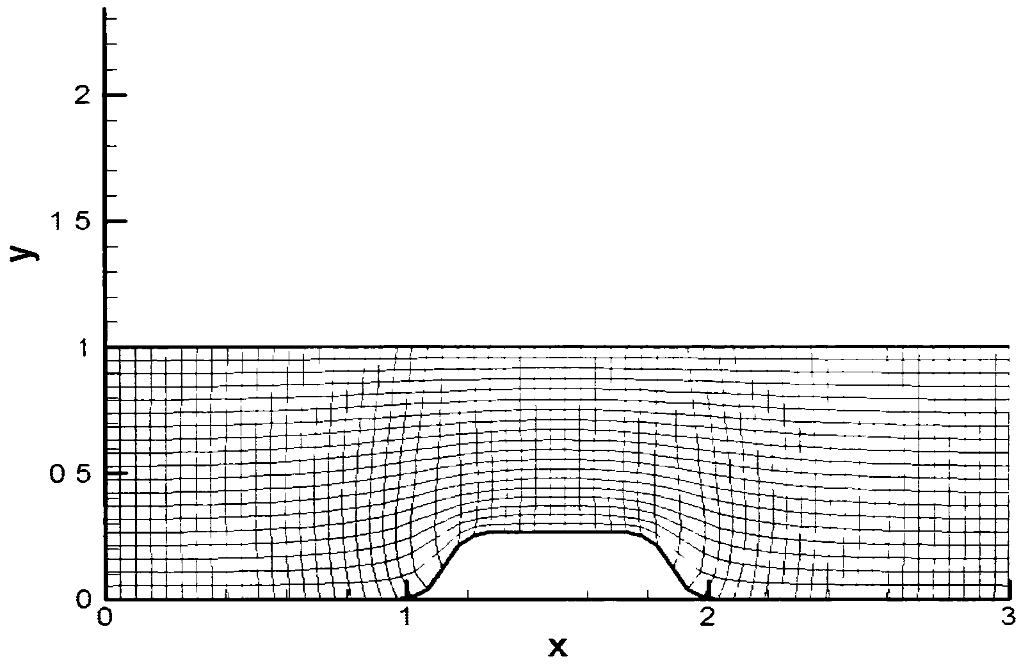

Figure 3.7(a): Grid Generated by Perturbation Method at $t = \frac{3T}{4}$

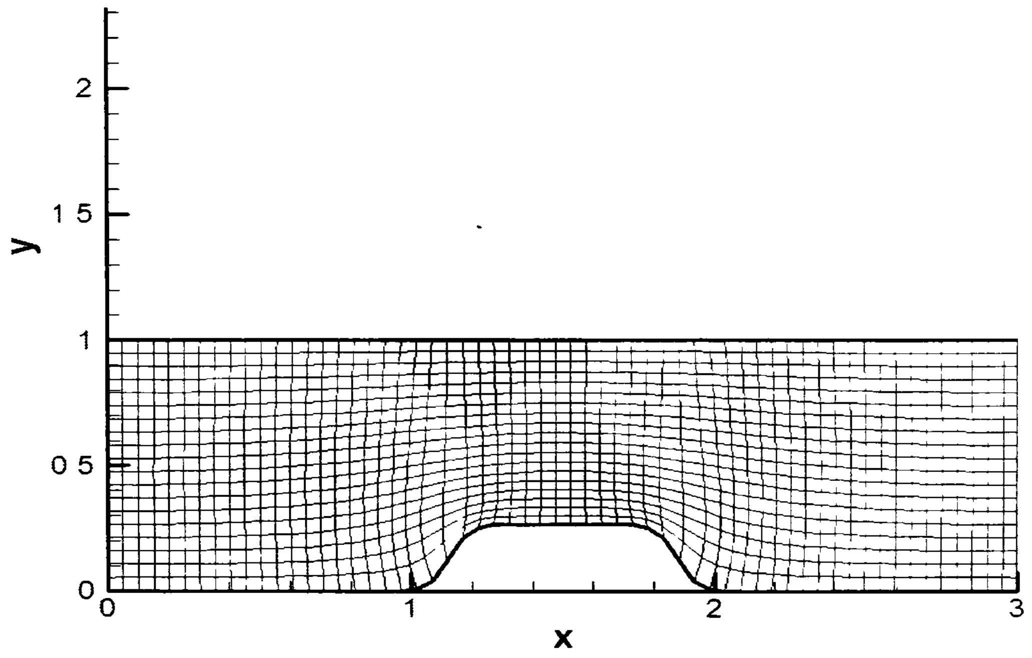

Figure 3.7(b): Grid Generated by Full Elliptic System at $t = \frac{3T}{4}$

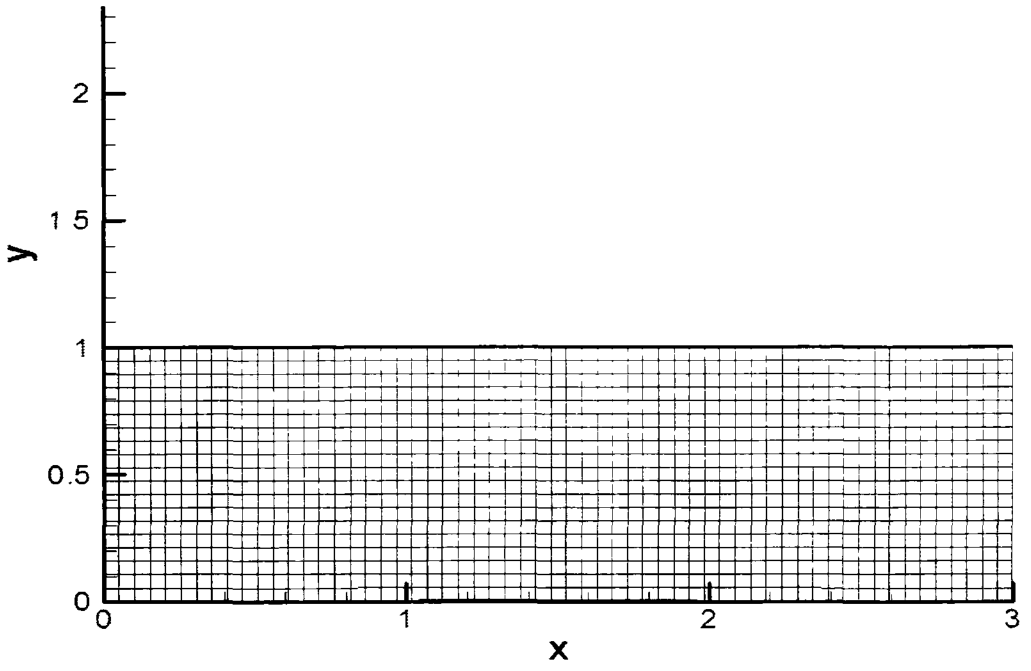

Figure 3.8(a): Grid Generated by Perturbation Method at $t = T$

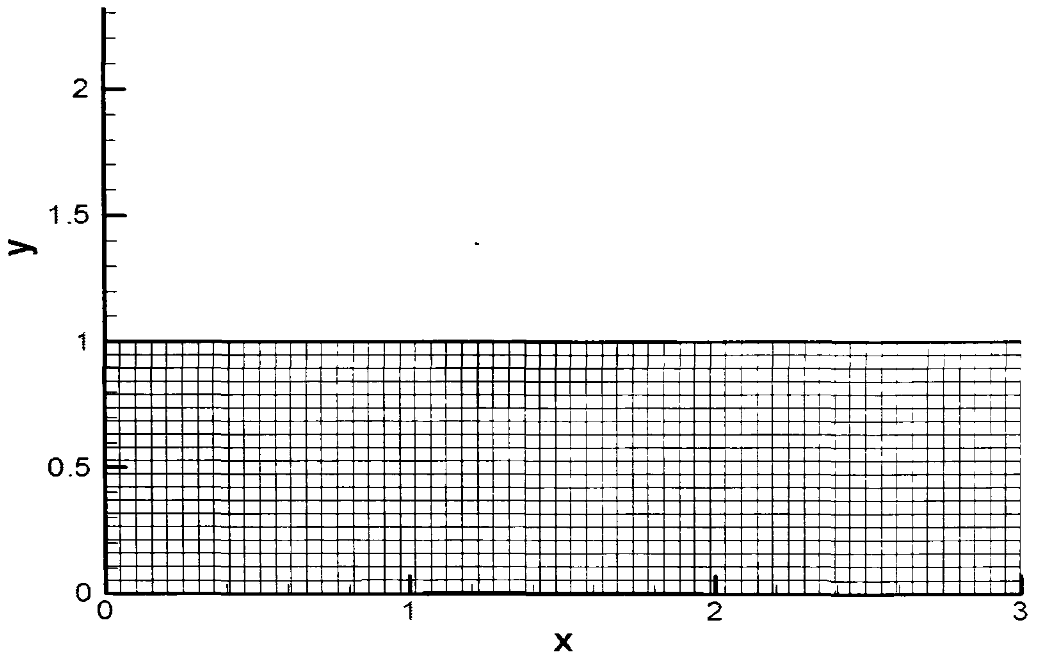

Figure 3.8(b): Grid Generated by Full Elliptic System at $t = T$

Case II: Grid Generation on a Domain with a Moving Sine Wave ($0 \leq t \leq T$)

In this case a sine wave moves along the lower boundary and, as seen from Figures 3.10 to 3.13, there is very little difference between the grids generated by the two methods.

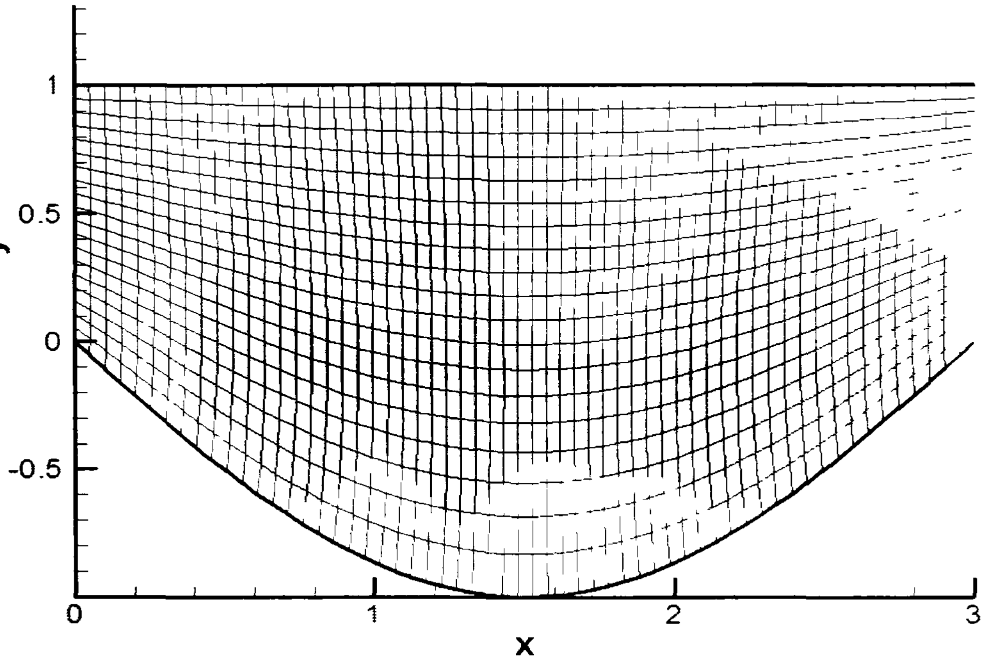

Figure 3.9: Grid Generated by Full Elliptic System at t = 0 (Initial Grid)

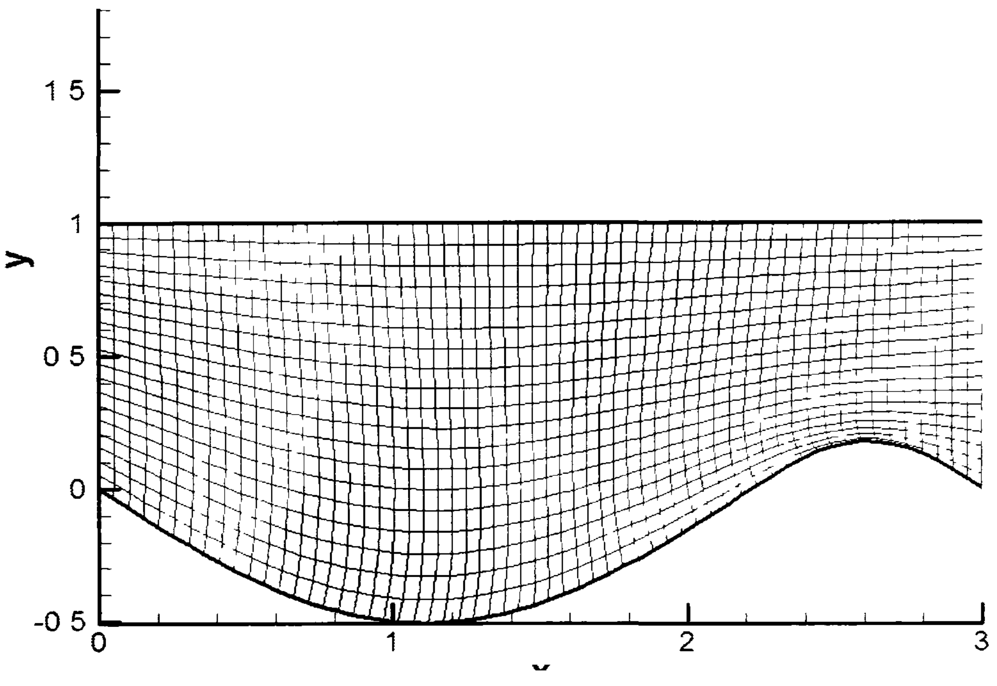

Figure 3.10(a): Grid Generated by Perturbation Method at $t = \frac{T}{4}$

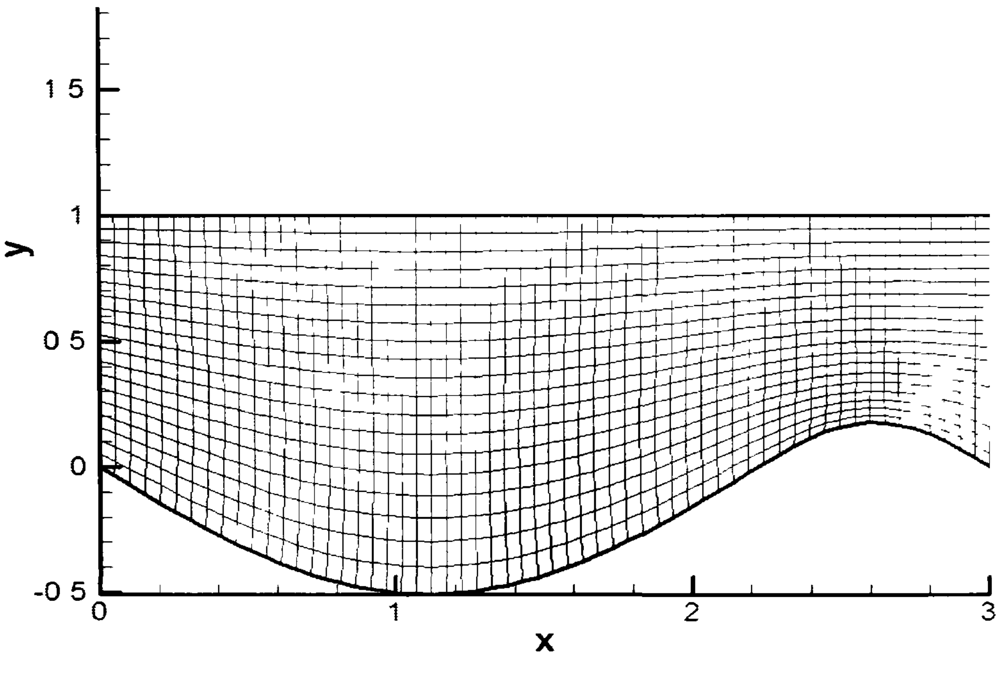

Figure 3.10(b): Grid Generated by Full Elliptic System at $t = \frac{T}{4}$

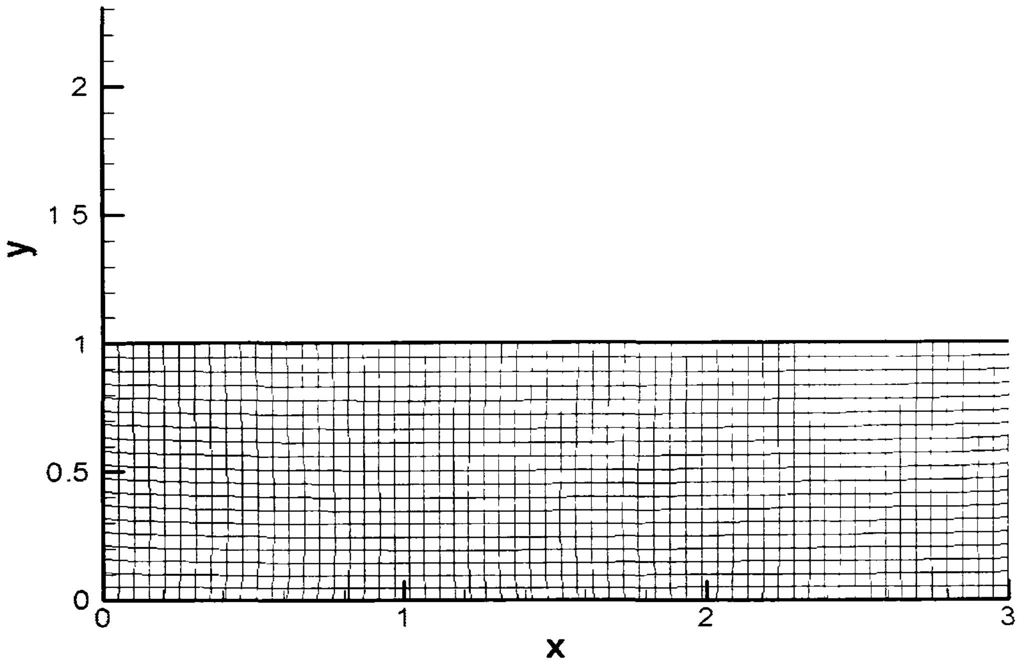

Figure 3.11(a): Grid Generated by Perturbation Method at $t = \frac{T}{2}$

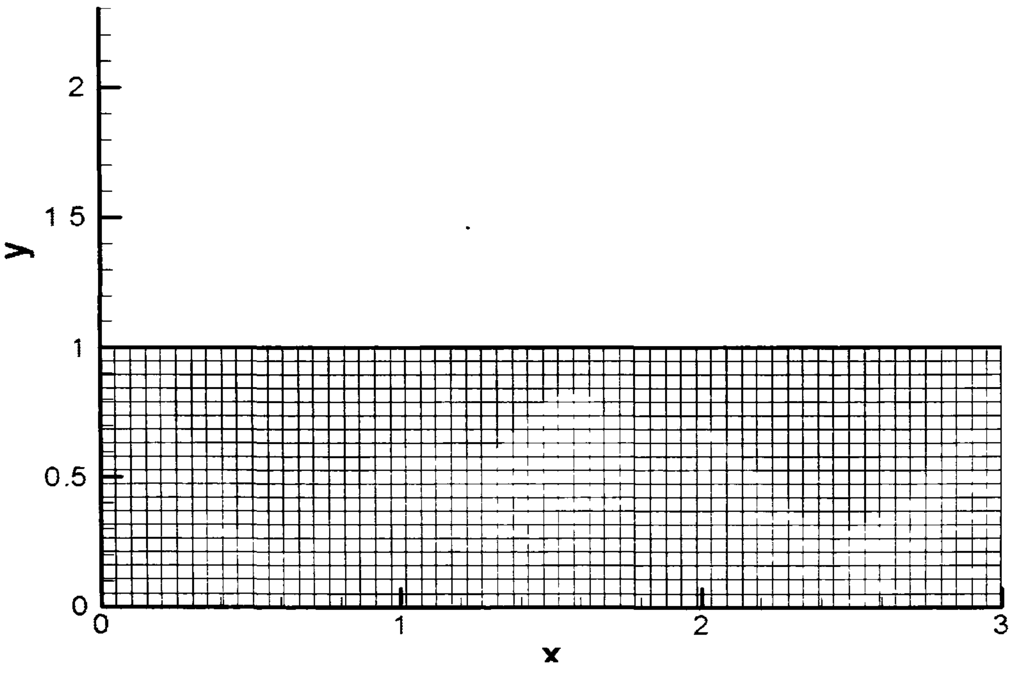

Figure 3.11(b): Grid Generated by Full Elliptic System at $t = \frac{T}{2}$

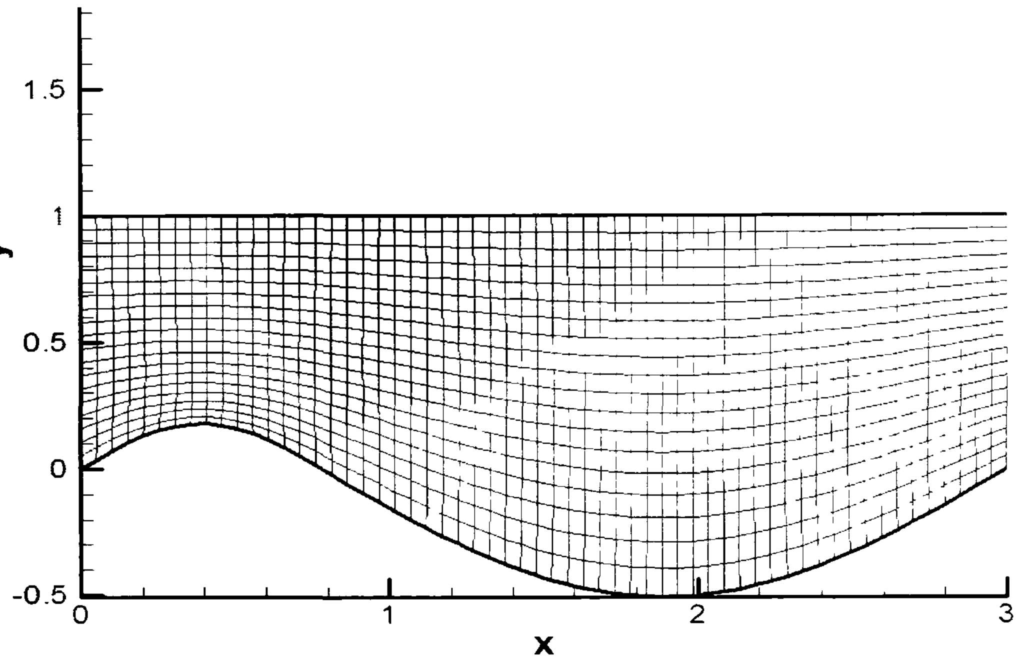

Figure 3.12(a): Grid Generated by Perturbation Method at $t = \frac{3T}{4}$

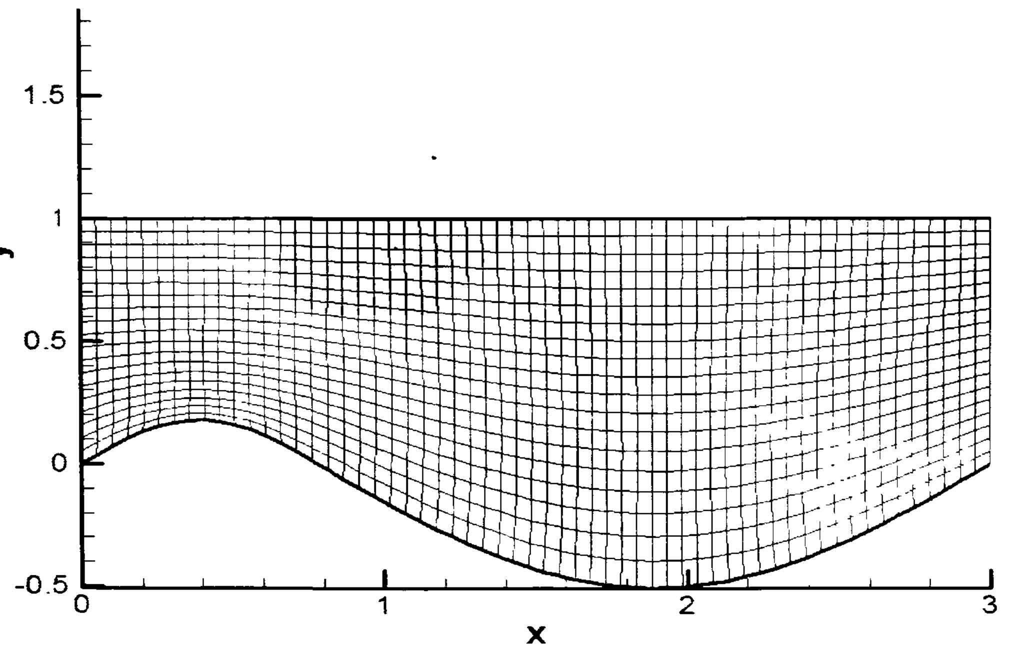

Figure 3.12(b): Grid Generated by Full Elliptic System at $t = \frac{3T}{4}$

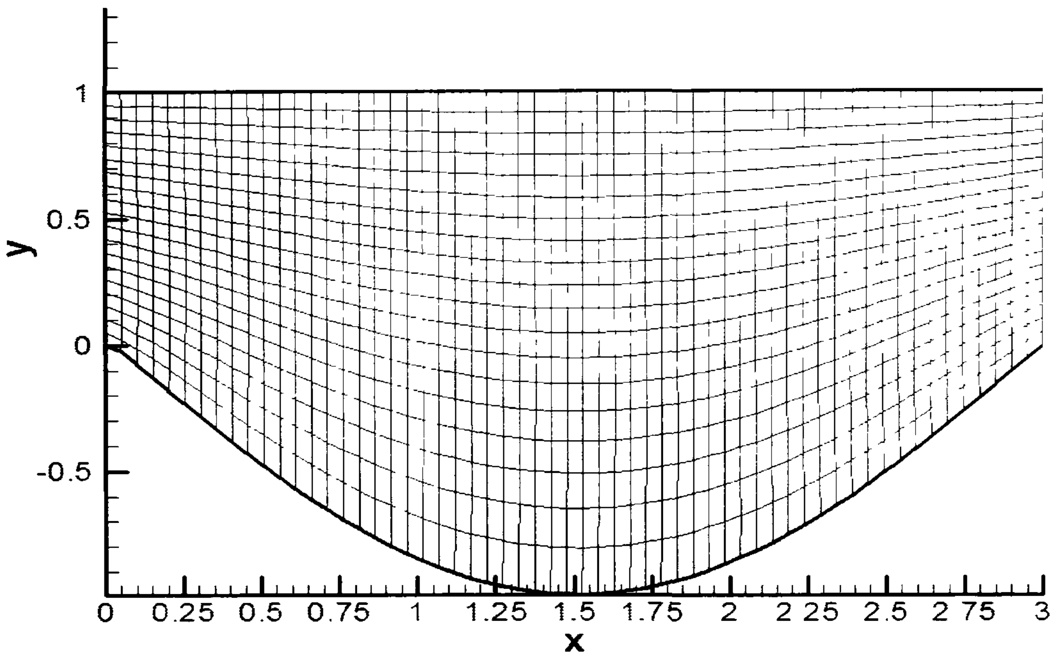

Figure 3.13(a): Grid Generated by Perturbation Method at $t = T$

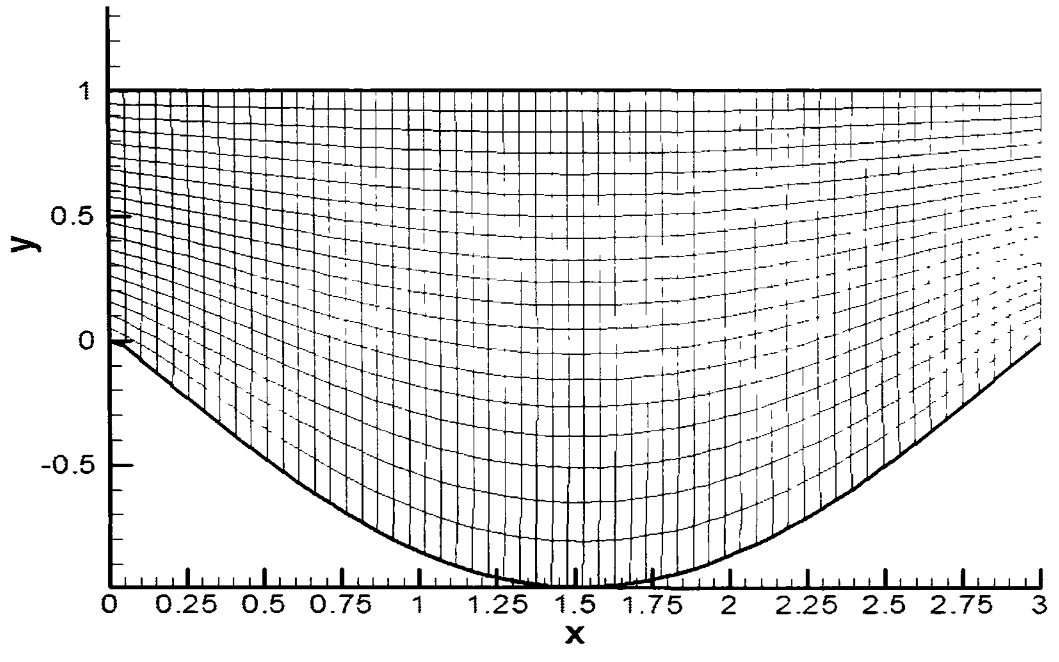

Figure 3.13(b): Grid Generated Full by Elliptic System at $t = T$

3.5.1 Grid Quality

A. Global Functionals

The orthogonality functional (I_o) and the Modified Liao functional (I_{ML}), defined in Equations (3.18) and (3.19) respectively, are introduced as a measure of the quality of the grid generated by the perturbation method and the full elliptic system. The results for *Case II* at time $t = \frac{T}{4}$ are shown in Table 3.1. For both functionals, the minimum value obtained by the perturbation method is only slightly higher than that obtained from the full elliptic generator.

$$I_o = \int_0^1 \int_0^1 \frac{g_{12}^2}{g_{11}g_{22}} d\xi d\eta \qquad (3.18)$$

$$I_{ML} = \int_0^1 \int_0^1 (\frac{g_{11}+g_{22}}{J})^2 d\xi d\eta \qquad (3.19)$$

	Orhogonality Functional	Modified Liao Functional
Perturbation Method	45.161493	16.168057
Elliptic Method	44.391483	15.998572

Table 3.1: Comparison of Grid Functionals at $t = \frac{T}{4}$ *(Case II)*

B. Skew Angle

The angle created by ξ and η lines crossing each other is called the skew angle and can be calculated at any grid node from

$$\theta = cos^{-1}(\frac{g_{12}}{\sqrt{g_{11}g_{22}}})$$

The skew angle at some typical points are recorded in Table 3.2.

Coordinates	Perturbation Method	Elliptic Method
i = 20, j=2	89.371836	96.873012
i = 40, j=8	72.936906	72.830267
i = 20, j=5	94.994802	95.190263
i = 40, j=15	80.323625	83.58687

Table 3.2: Comparison of Grid Skew Angles (deg) at $t = \frac{T}{4}$ *(Case II)*

Considering these tables, it can be seen that the grid generated by the perturbation method not only preserves the qualities of the grid generated by the full elliptic system but improves these qualities at some points. It should also be remembered that the main advantage of introducing this new grid generator is to achieve a reduction in computational time. For instance, in these examples, the computational time consumed for generating the grid with the perturbation method was 1 5 of the time required for generating the same grid with the full elliptic system.

CHAPTER 4 FLOW SOLVER

4.1 Introduction

This chapter concentrates on solving the two-dimensional, unsteady incompressible Navier-Stokes equations in general curvilinear coordinates and non-conservative form. The continuity and momentum equations are written in dimensional form and discretized in computational curvilinear space. The velocity field is marched in time using the momentum equations, which are solved implicitly. To ensure conservation of mass, a SIMPLE-like approach for unsteady flows is used to derive a pressure correction equation, which also gives velocity corrections and the pressure field at the next time. This approach is an extension of the method introduced by Zogheib and Barron for steady flows [31].

4.2 Flow Equations

The governing unsteady flow equations in general curvilinear coordinates (ξ, η) and non-conservative form are

Continuity equation:

$$\xi_x u_\xi + \eta_x u_\eta + \xi_y v_\xi + \eta_y v_\eta = 0 \tag{4.1}$$

u-momentum equation:

$$u_t + (\xi_t + u\xi_x + v\xi_y)u_\xi + (\eta_t + u\eta_x + v\eta_y)u_\eta$$

$$= -\frac{\xi_x}{\rho}p_\xi - \frac{\eta_x}{\rho}p_\eta$$

$$+ \upsilon\{(\xi_x^2 + \xi_y^2)u_{\xi\xi} + (\eta_x^2 + \eta_y^2)u_{\eta\eta} + 2(\xi_x\eta_x + \xi_y\eta_y)u_{\xi\eta}$$

$$+ (\xi_{xx} + \xi_{yy})u_\xi + (\eta_{xx} + \eta_{yy})u_\eta\}$$

(4.2)

v-momentum equation:

$$v_t + (\xi_t + u\xi_x + v\xi_y)v_\xi + (\eta_t + u\eta_x + v\eta_y)v_\eta$$

$$= -\frac{\xi_y}{\rho}p_\xi - \frac{\eta_y}{\rho}p_\eta + \upsilon\{(\xi_x^2 + \xi_y^2)v_{\xi\xi} + (\eta_x^2 + \eta_y^2)v_{\eta\eta}$$

$$+ 2(\xi_x\eta_x + \xi_y\eta_y)v_{\xi\eta} + (\xi_{xx} + \xi_{yy})v_\xi + (\eta_{xx} + \eta_{yy})v_\eta\}$$

(4.3)

where u and v are the velocity components in the x and y directions respectively. ρ is the density, p is the pressure and υ is kinematic viscosity. The quantities ξ_t, ξ_x, ξ_y. η_t. η_x and η_y are metrics of the transformation which are given by

$$\xi_x = \frac{y_\eta}{J}$$

$$\eta_x = -\frac{y_\xi}{J}$$

$$\xi_y = -\frac{x_\eta}{J}$$

$$\eta_y = \frac{x_\xi}{J}$$

$$\xi_t = \frac{-x_\tau y_\eta + y_\tau x_\eta}{J}$$

$$\eta_t = \frac{x_\tau y_\xi - y_\tau x_\xi}{J}$$

where J is the Jacobian of the transformation from Cartesian coordinates to curvilinear coordinates, ie.,

$$J = \frac{\partial(x,y)}{\partial(\xi,\eta)} = \frac{1}{\frac{\partial(\xi,\eta)}{\partial(x,y)}}$$

4.3 Discretization of the Momentum Equations

A staggered grid with N points in the ξ direction and M points in the η direction is applied for discretizing the momentum and continuity equations, with $\Delta\xi = \Delta\eta = 1$. The u and v values are stored at $i-1,j$ and $i,j+1$ locations respectively, and p is stored at i,j as shown in Figure 4.1. Upwinding is used for convective terms and diffusion terms are second order central differenced. Pressure gradients are approximated by second order central differencing. For calculation of grid metrics, central differencing with spacing $\Delta\xi$ and $\Delta\eta$ between the nodes is used at interior nodes. The grid metrics on the West and South boundaries are second order forward differenced and the grid metrics on the East and North boundaries are second order backward differenced; refer to [32] for details of differencing various terms.

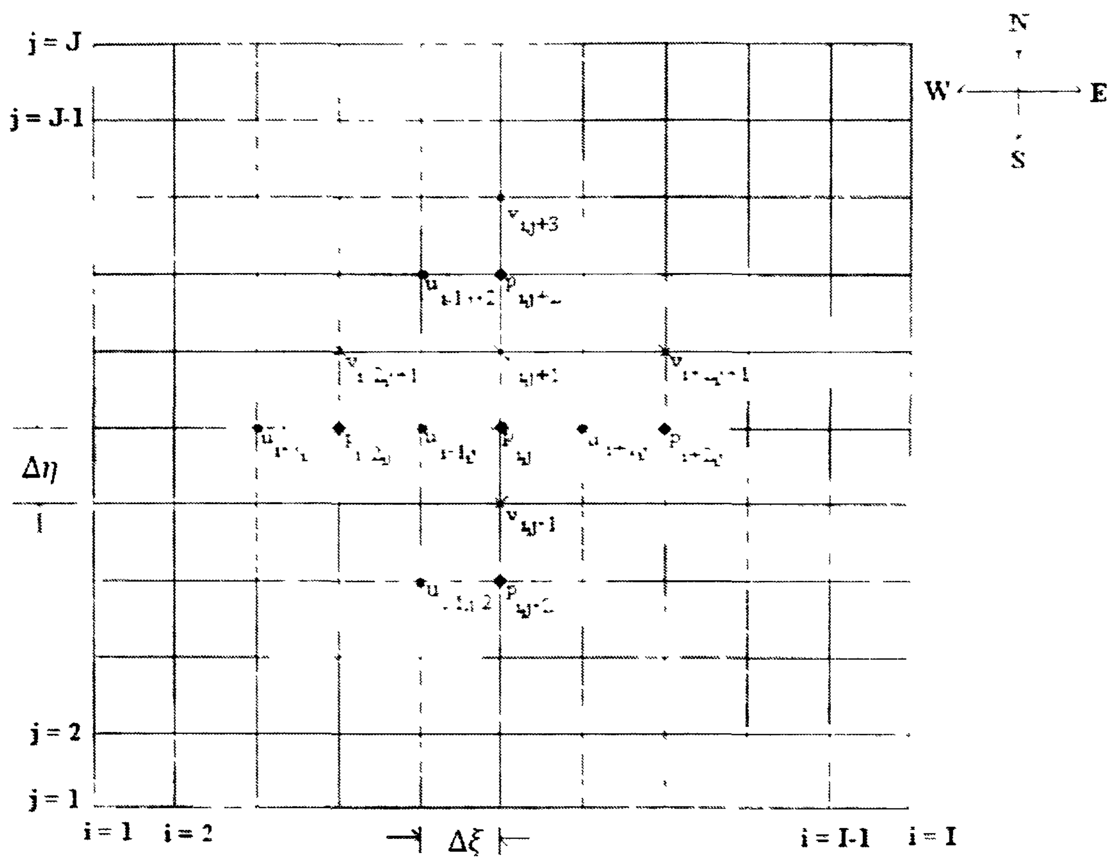

Figure 4.1: Staggered Grid with Notations and Storage Locations

If Δt is the time between two successive grid geometries at times t_n and t_{n+1}. during which the grid nodes change their positions from (x^n, y^n) to (x^{n+1}, y^{n+1}). then the following first order approximations are used in the expressions for ξ_t and η_t,

$$x_t \approx \frac{x^{n+1} - x^n}{\Delta t}$$

$$y_t \approx \frac{y^{n+1} - y^n}{\Delta t}$$

4.3.1 Discretized Equations at Interior Nodes

Using the Implicit Euler scheme for the time march, the discrete $\boldsymbol{u}$- and $\boldsymbol{v}$- momentum equations at interior nodes are written as

$$a_P u_{i-1,j} + a_N u_{i-1,j+2} + a_S u_{i-1,j-2} + a_W u_{i-3,j} + a_E u_{i+1j} = \xi_x \frac{p_{i-2,j} - p_{i,j}}{2\rho} + S_u \tag{4.4}$$

$$b_P v_{i,j+1} + b_N v_{i,j+3} + b_S v_{i,j-1} + b_W v_{i-2,j+1} + b_E v_{i+2,j+1} = \eta_y \frac{p_{i,j} - p_{i,j+2}}{2\rho} + S_v \tag{4.5}$$

Denoting $U = u\xi_x + v\xi_y$ and $V = u\eta_x + v\eta_y$ and with $\Delta\xi = \Delta\eta = 1$, the coefficients take the form

$$a_P = \frac{1}{\Delta t} + \frac{\xi_t}{2} + \frac{\eta_t}{2} + \frac{|U_{i-1,j}|}{2} + \frac{|V_{i-1,j}|}{2} + \frac{\upsilon}{2J^2} g_{11} - \frac{\upsilon}{2J^2} g_{22}$$

$$a_N = \min(0, \frac{V_{i-1,j}}{|V_{i-1,j}|}) \frac{\eta_t}{2} + \frac{\min(0, V_{i-1,j})}{2} - \frac{\upsilon}{4J^2} g_{11} - \frac{\upsilon}{4} \nabla^2 \eta$$

$$a_S = -\max(0, \frac{V_{i-1,j}}{|V_{i-1,j}|}) \frac{\eta_t}{2} - \frac{\max(0, V_{i-1,j})}{2} - \frac{\upsilon}{4J^2} g_{11} - \frac{\upsilon}{4} \nabla^2 \eta$$

$$a_W = -\max(0, \frac{U_{i-1,j}}{|U_{i-1,j}|}) \frac{\xi_t}{2} - \frac{\max(0, U_{i-1,j})}{2} - \frac{\upsilon}{4J^2} g_{22} - \frac{\upsilon}{4} \nabla^2 \xi$$

$$a_E = \min(0, \frac{U_{i-1,j}}{|U_{i-1,j}|}) \frac{\xi_t}{2} + \frac{\min(0, U_{i-1,j})}{2} - \frac{\upsilon}{4J^2} g_{22} - \frac{\upsilon}{4} \nabla^2 \xi$$

and

$$S_u = \frac{u^n_{i-1,j}}{\Delta t} + \frac{y_\xi}{8\rho J}\left(p_{i,j-2} - p_{i,j-2} - p_{i-2,j-2} - p_{i-2,j-2}\right)$$
$$- \frac{\upsilon}{8J^2} g_{12}\left(u_{i-1,j-2} - u_{i-3,j-2} + u_{i-3,j-2} - u_{i-1,j-2}\right)$$

The coefficients in the v-equation are the same as in the u-equation, but all variables and metrics are evaluated at the *v*-nodes *i,j*+1 instead of the *u*-nodes *i*-1,*j* and

$$S_v = \frac{v^n_{i,j+1}}{\Delta t} + \frac{x_\eta}{8\rho J}\left(p_{i+2,j+2} + p_{i+2,j} - p_{i-2,j} - p_{i-2,j-2}\right)$$

$$-\frac{\upsilon}{8J^2} g_{12}\left(v_{i+2,j+3} - v_{i-2,j+3} - v_{i+2,j-1} + v_{i-2,j-1}\right)$$

These equations are an extension to unsteady flow of those derived by Barron and Zogheib [33], the first terms representing extra terms due to the time derivative terms in the momentum equations. All the terms in the Equations (4.4) and (4.5) are evaluated at time t_{n+1} except those with superscript n which are evaluated at time t_n. For the purpose of linearization the velocity terms in the coefficient and the pressure terms are evaluated at the previous iteration within time t_{n+1}.

4.3.2 Discretized Equations at the Boundaries

Discretizing equations at the boundaries depends on the problem to be solved and the type of boundary conditions to be applied. The main problem considered in this chapter is a channel with a moving indentation on the South boundary. The channel is rectangular with fully developed parabolic flow throughout the channel at the initial time. As time proceeds the boundary moves up until the maximum amplitude is reached and then it comes back to its initial position. The oscillation period is the time required for the channel to pass one whole cycle. Dirichlet boundary conditions, which means zero velocity components (ie., no-slip), are applied along the North and South boundaries. On the East boundary, an outflow condition is applied in the form

$$\frac{\partial u}{\partial \xi} = 0, \qquad \frac{\partial v}{\partial \xi} = 0$$

In order to approximate pressure gradients at near-boundary nodes, one-sided differencing is used since the pressure values along the boundaries are not known. Along the East boundary the condition $\frac{\partial v}{\partial \xi} = 0$ is applied at $i = N$ and $\frac{\partial u}{\partial \xi} = 0$ at $i = N - 1$. Refer to [32] for details on discretizing the momentum equations near the boundaries.

4.4 SIMPLE-like Velocity-Pressure Coupling

If $u^*$and $v^*$are the discrete velocity fields that result from the solution of momentum equations (4.4) and (4.5) corresponding to some pressure field p^*, they do not necessarily satisfy the continuity equation. Hence, the velocity and pressure fields are corrected as

$$u = u^* + u'$$

$$v = v^* + v'$$

$$p = p^* + p' \qquad (4.6)$$

where primes denote corrections. Subtracting the corrected and the uncorrected momentum equations will result in equations for velocity and pressure corrections

$$a_{i-1,j} u'_{i-1,j} = \sum a_{nb} u'_{nb} + \frac{\xi_x}{\rho} \frac{p'_{i-2,j} - p'_{i,j}}{2} + S'_u \qquad (4.7)$$

$$b_{i,j+1} v'_{i,j+1} = \sum b_{nb} v'_{nb} + \frac{\eta_y}{\rho} \frac{p'_{i,j} - p'_{i,j+2}}{2} + S'_v \qquad (4.8)$$

As in the SIMPLE algorithm, these equations are approximated as

$$a_{i-1,j} u'_{i-1,j} \approx \frac{\xi_x}{\rho} \frac{p'_{i-2,j} - p'_{i,j}}{2}$$

$$b_{i,j+1} v'_{i,j+1} \approx \frac{\eta_y}{\rho} \frac{p'_{i,j} - p'_{i,j+2}}{2}$$

Substituting in Equations (4.6) gives

$$u_{i-1,j} = u^*_{i-1,j} + \xi_x \frac{p'_{i-2,j} - p'_{i,j}}{2\rho a_{i-1,j}} \qquad (4.9)$$

$$v_{i-1,j} = v^{*}_{i-1,j} + \eta_y \frac{p'_{i\,j} - p'_{i\,j+2}}{2\rho b_{i\,j+1}} \tag{4.10}$$

A similar approach has been suggested by Chung [34].

4.4.1 Pressure Correction Equation

The form of the pressure correction equation is the same as the steady flow case, since there is no time dependent term in the continuity equation. Substitution of corrected velocity fields into the continuity equation (4.1) results in an equation for correcting the pressure field,

$$\xi_x \frac{\left\{u^{*}_{i+1,j} + \frac{\xi_x}{\rho}\frac{p'_{i,j} - p'_{i+2,j}}{2a_{i+1,j}}\right\} - \left\{u^{*}_{i-1,j} + \frac{\xi_x}{\rho}\frac{p'_{i-2,j} - p'_{i,j}}{2a_{i-1,j}}\right\}}{2} + \eta_y \frac{\left\{v^{*}_{i\,j+1} + \frac{\eta_y}{\rho}\frac{p'_{i,j} - p'_{i\,j+2}}{2b_{i,j+1}}\right\} - \left\{v^{*}_{i,j-1} + \frac{\eta_y}{\rho}\frac{p'_{i,j-2} - p'_{i,j}}{2b_{i,j-1}}\right\}}{2} =$$

$$-\eta_x u_\eta|_{i,j} - \xi_y v_\xi|_{i,j}$$

Rearranging terms gives the pressure correction equation at interior nodes

$$C_P p'_{i,j} + C_E p'_{i+2,j} + C_W p'_{i-2,j} + C_N p'_{i,j+2} + C_S p'_{i,j-2}$$
$$= \xi_x \frac{u^{*}_{i-1,j} - u^{*}_{i+1,j}}{2} - \eta_y \frac{v^{*}_{i,j+1} - v^{*}_{i,j-1}}{2} + S_p \tag{4.11}$$

where

$$S_p = -\eta_x \left(\frac{u_{i+1,j+2} + u_{i-1,j+2} - u_{i+1,j-2} - u_{i-1,j-2}}{8}\right)$$
$$- \xi_y \left(\frac{v_{i+2,j+1} + v_{i+2,j-1} - v_{i-2,j+1} - v_{i-2,j-1}}{8}\right)$$

$$C_P = \frac{\xi_x^2}{4\rho a_{i+1,j}} + \frac{\xi_x^2}{4\rho a_{i-1,j}} + \frac{\eta_y^2}{4\rho b_{i,j+1}} + \frac{\eta_y^2}{4\rho b_{i,j-1}}$$

$$C_E = -\frac{\xi_x^2}{4\rho a_{i+1,j}}$$

$$C_W = -\frac{\xi_x^2}{4\rho a_{i-1,j}}$$

$$C_N = -\frac{\eta_y^2}{4\rho b_{i,j+1}}$$

$$C_S = -\frac{\eta_y^2}{4\rho b_{i,j-1}}$$

4.4.2 Pressure Correction Equation at the Boundaries

The velocity components are known at the inlet (West boundary). Since North and South boundaries are walls the velocity components are set to be zero along these boundaries. When Dirichlet conditions on velocity are known at a boundary, these values are used in the continuity equation adjacent to that boundary before (4.9) and (4.10) are substituted to obtain the pressure correction equation. In the discretized pressure correction equation at the East boundary the link to the outlet boundary side is suppressed by setting $C_E = 0$. For a more detailed treatment of pressure equations near the boundaries, refer to [32].

4.4.3 Solution Algorithm

Once all the necessary equations have been derived, the overall solution algorithm can be described by the following steps:

1. Set $n = 0$.

2. Read the mesh data files produced by the mesh generator.

3. Calculate the grid metrics at the new time.

4. Set the initial conditions on the pressure and velocity fields over the domain at time $t = t_n$.

5. Solve the u-momentum equation at all u-nodes.

6. Solve the v-momentum equation at all v-nodes.

7. Solve the pressure correction equation at all p-nodes.

8. Correct the pressure and velocity fields.

9. Continue through steps 5-8 until the results (u, v, p) meet a specified convergence criteria.

10. Set the converged results to be the initial conditions for the next time.

11. If n is less than the specified number of time steps, increase n to $n + 1$ and go back to step 2.

4.5 Results and Discussions

In this section the algorithm developed in this chapter is applied to two test problems. The first is developing flow in a rectangular channel; the second problem is fully developed flow in a channel with a moving indentation along the wall. The second problem is of great interest in the literature as a benchmark problem for the validation of unsteady codes. It has been experimentally studied by Pedley and Stephanoff [35], and numerically studied by Ralph and Pedley [18] using a vorticity-streamfunction formulation. Thereafter, Demirdzic and Peric [23] solved this problem with a finite volume approach and their results are in good agreement with those of Ralph and Pedley [18] and the experimental results of Pedley and Stephanoff [35].

4.5.1 Test Case 1: Developing Flow in a Rectangular Channel

The purpose of this example is to test the capability of the unsteady code to solve steady flow problems by considering the time as iteration, and marching to the steady-state. The channel is covered with 371×41 grid lines in the x and y directions respectively. The channel length is 27.85 cm and channel height is 1 cm. The inlet flow has a uniform profile. the media is water at 20°C and Reynolds number based on inlet velocity and channel height is 50.

The results show that the unsteady solver is capable of generating the steady flow by marching in time until the steady-state is reached. The tolerances for convergence have been set at 10^{-10}, and typical residuals for the u-momentum, v-momentum and continuity equations are of the order of at least 10^{-8}, 10^{-9} and 10^{-10} respectively. Figure 4.2 compares the non-dimensional centreline velocity with the data available from the steady solver of Zogheib and Barron [31]. The present method predicts $x = 0.301$ as the point where centreline velocity reaches 99% of the maximum non-dimensional centreline velocity (ie.,1.5). This development length is the same as that reported by Zogheib and Barron [31].

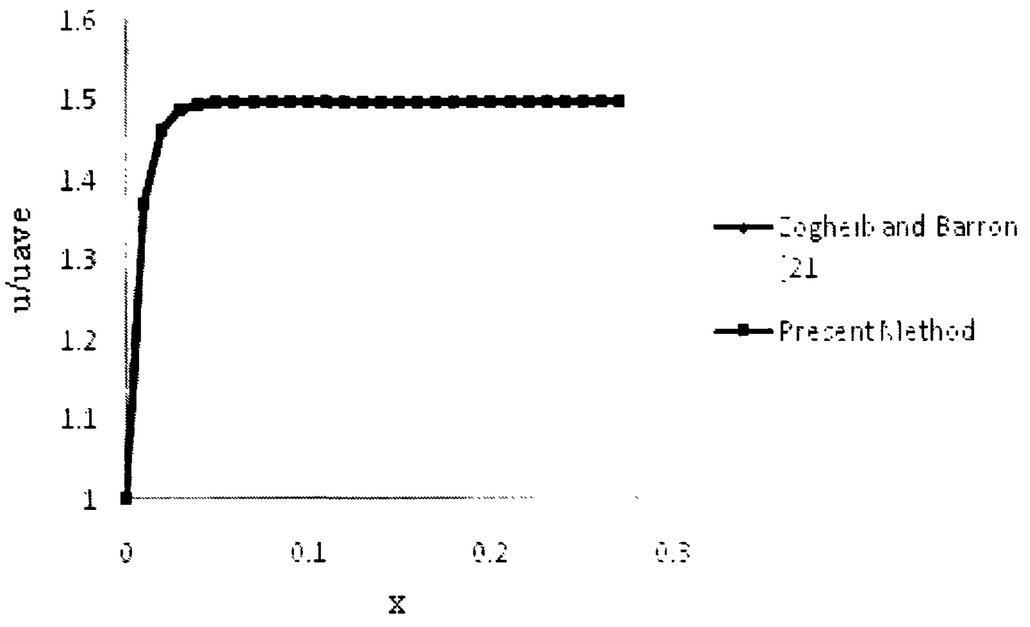

Figure 4.2: Comparison of Non-Dimensional Centreline Velocities at Different Locations from the Flow Entrance

4.5.2 Test Case 2: Flow in a Channel with a Moving Indentation

The channel shown in Figure 4.3 covered with 371×41 grid lines in the ξ and η directions respectively (Figure 4.4a). The un-indented channel has length of 27.85 cm ($l_1 = 9.85\ cm,\ l_2 = 18cm$) and height of $b = 1\ cm$. The inlet flow has a fixed parabolic profile, the media is air at 20°C. The geometry of the channel changes with time (see. for example, Figure 4.4b) and the grid is generated by the method developed in Chapter 3, as illustrated in Figures 3.4-3.8.

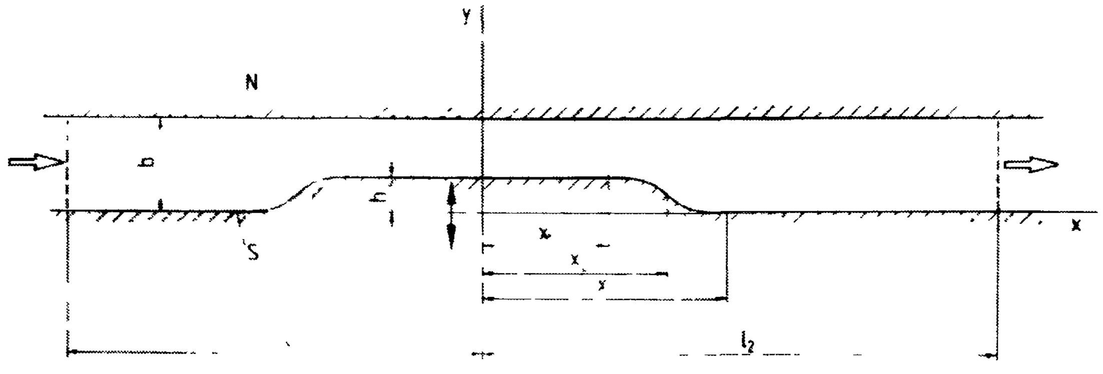

Figure 4.3: Geometry of Test Case 2

The height of the bottom (indented) wall is given by [23]

$$y(x) = \begin{cases} h & for\ 0 < x < x_1, \\ 0.5h(1 - \tanh[a(x - x_2)]\,) & for\ x_1 < x < x_2, \\ 0 & for\ x > x_3, \end{cases} \tag{4.12}$$

where $a = 4.14, x_1 = 4b, x_3 = 6.5b, x_2 = 0.5(x_1 + x_3)$ and

$$h = 0.5h_{max}[1 - \cos(2\pi t_f)]$$

Here b is the channel height, T is the oscillation period and $h_{max} = 0.38b$ specifies the maximum blockage of the channel at $t_f = 0.5$. where $t_f = \frac{t}{T}$ is introduced as the time fraction relative to the oscillation period.

The Reynolds number and Strouhal number based on the channel height b. bulk velocity U_{bulk} and oscillation period T are defined as

$$Re = \frac{U_{bulk}b}{\upsilon}$$

$$St = \frac{b}{TU_{bulk}}$$

In order to compare directly with the results of Demirdzic and Peric [23], the simulations are carried out for $Re = 507$ and $St = 0.037$. For the purpose of marching in time, the time step in the unsteady flow solver is taken to be $\Delta t_f = \frac{T}{50}$.

The inlet velocity profile is

$$u = 6U_{bulk}\frac{y}{b}(1 - \frac{y}{b}) \tag{4.13}$$

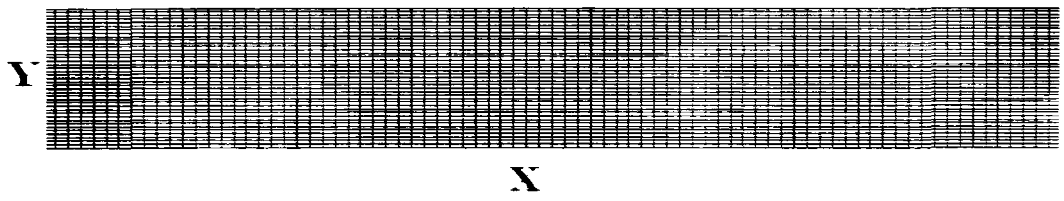

Figure 4.4(a): Mesh for Flow in a Channel with a Moving Indentation at $t_f = 0$

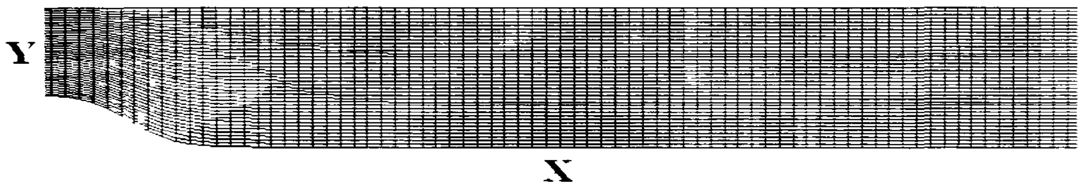

Figure 4.4(b): Mesh for Flow in a Channel with a Moving Indentation at $t_f = 0.5$

The velocity vectors (Figures 4.5(a)-4.5(i)) and streamlines (Figures 4.6(a)-4.6(i)) are compared at various time steps with the numerical simulation results available from Demirdzic and Peric [23]. In each figure the upper graph depicts results from Demirdzic and Peric's simulation and the lower graph is the current research results. Since the section before the indentation is not affected very much by the indentation, it is simply a converging channel with rather uninteresting flow features. it is not shown in these figures.

There are a few conclusions which can be made by observing the figures and numerical data obtained from the solver. The maximum velocity occurs at $t_f = 0.4$. The first separation occurs behind the indentation between $t_f = 0.2$ and $t_f = 0.3$. Between $t_f = 0.35$ and 0.4 another vortex appears on the opposite wall. At about $t_f = 0.45$ the third vortex appears on the bottom wall. The vortex building process continues until there are three of them on each wall. After $t_f = 0.7$. they start to become weaker until $t_f = 0.9$ when almost no eddies exist. The strength of the eddies predicted by the present method is a little weaker. therefore the smaller eddies are not captured. However. the maximum speeds at various times are compared in Table 4.1. and show good agreement with previous results.

t_f	Demirdzic and Peric [23]	Present Method
0.2	2.20	2.19
0.3	2.53	2.48
0.4	2.64	2.61
0.5	2.38	2.33
0.6	1.99	1.93
0.7	1.73	1.54
0.8	1.56	1.51
0.9	1.50	1.50
1.0	1.51	1.50

Table 4.1: Comparison of Maximum Velocities at Various Times (m/s)

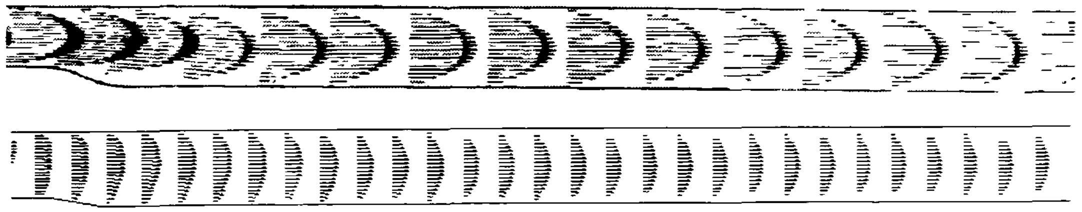
Figure 4.5 (a): Velocity Vectors Downstream of the Indentation ($t_f = 0.2$)

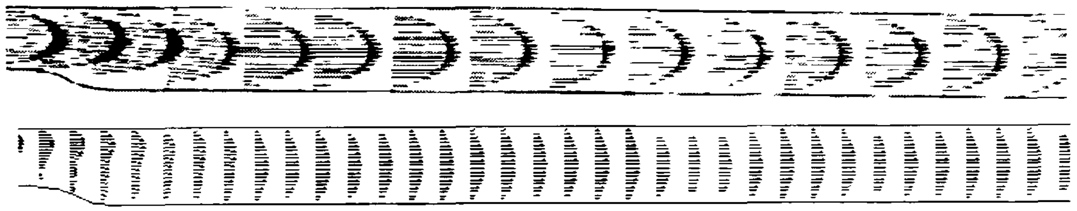
Figure 4.5 (b): Velocity Vectors Downstream of the Indentation ($t_f = 0.3$)

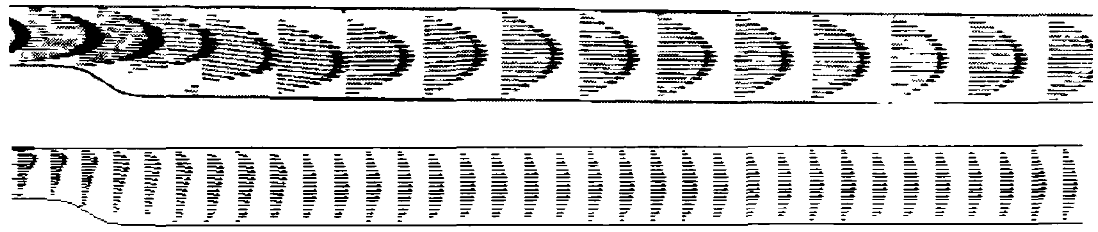
Figure 4.5 (c): Velocity Vectors Downstream of the Indentation ($t_f = 0.4$)

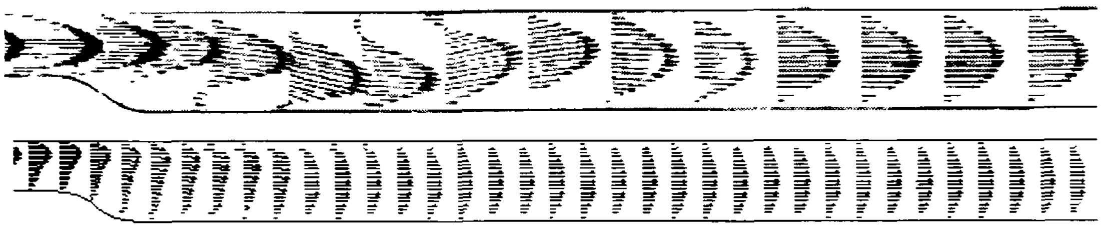
Figure 4.5 (d): Velocity Vectors Downstream of the Indentation ($t_f = 0.5$)

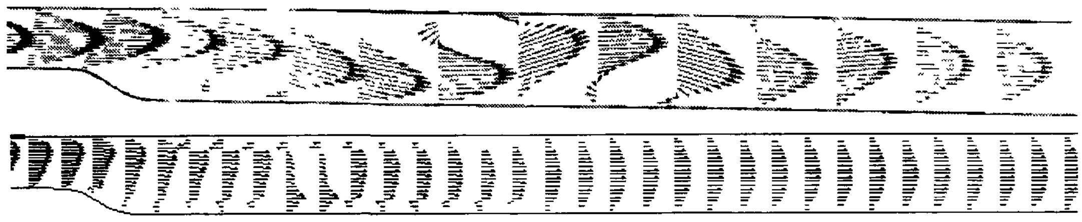

Figure 4.5 (e): Velocity Vectors Downstream of the Indentation $(t_f = 0.6)$

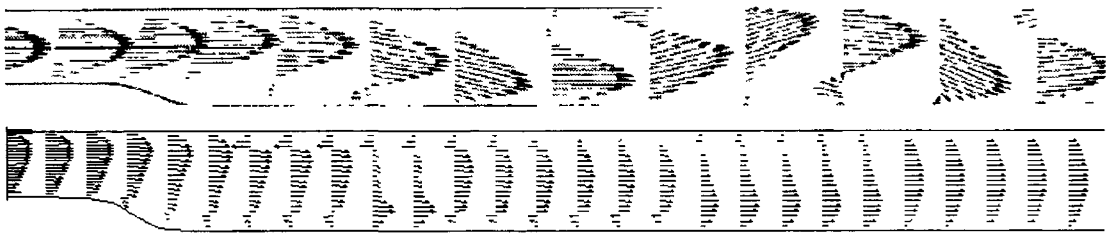

Figure 4.5 (f): Velocity Vectors Downstream of the Indentation $(t_f = 0.7)$

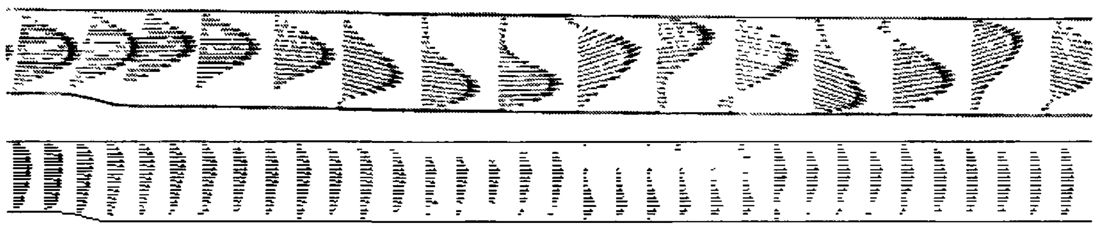

Figure 4.5 (g): Velocity Vectors Downstream of the Indentation $(t_f = 0.8)$

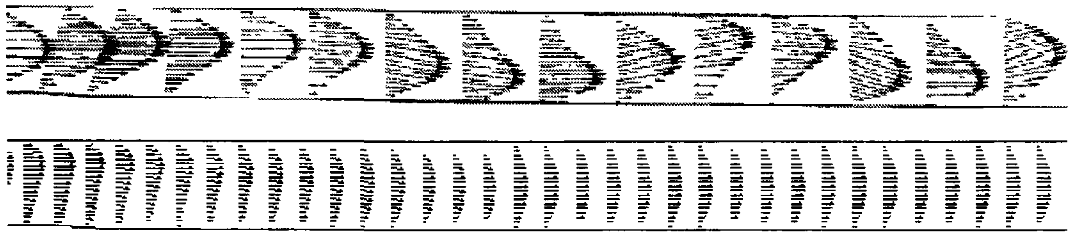

Figure 4.5 (h): Velocity Vectors Downstream of the Indentation $(t_f = 0.9)$

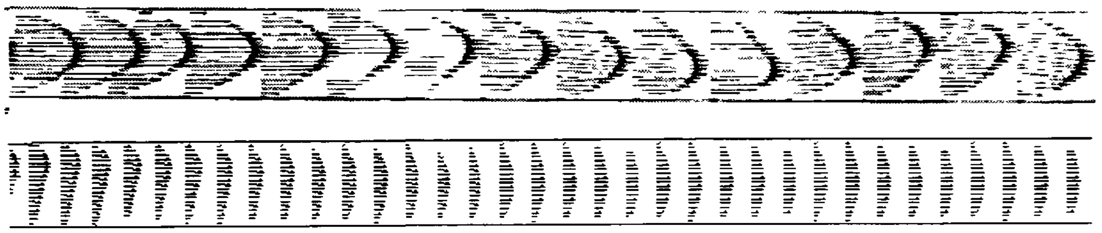

Figure 4.5 (i): Velocity Vectors Downstream of the Indentation $(t_f = 1.0)$

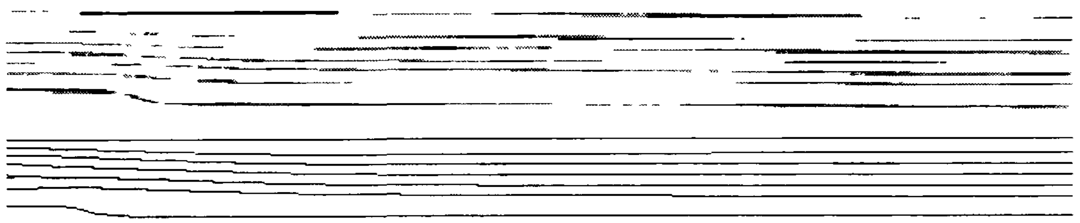

Figure 4.6 (a): Streamlines Downstream of the Indentation $(t_f = 0.2)$

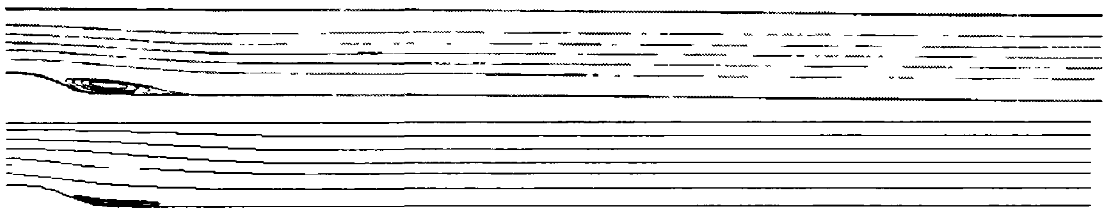

Figure 4.6 (b): Streamlines Downstream of the Indentation $(t_f = 0.3)$

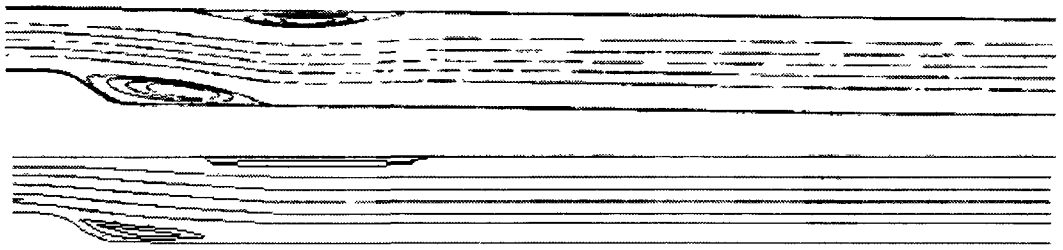

Figure 4.6 (c): Streamlines Downstream of the Indentation $(t_f = 0.4)$

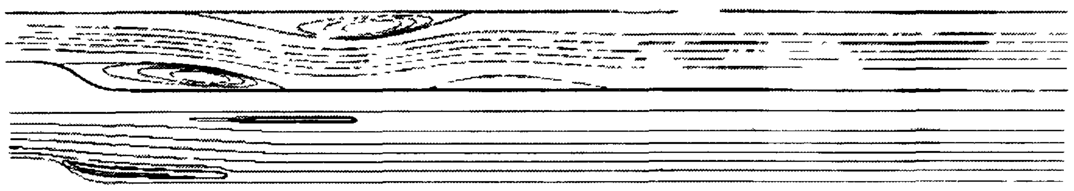

Figure 4.6 (d): Streamlines Downstream of the Indentation $(t_f = 0.5)$

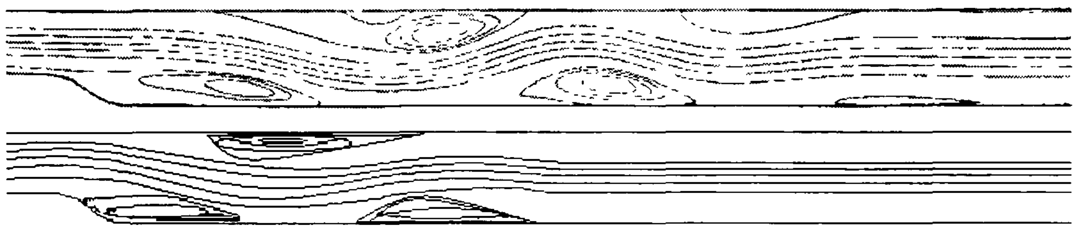

Figure 4.6 (e): Streamlines Downstream of the Indentation $(t_f = 0.6)$

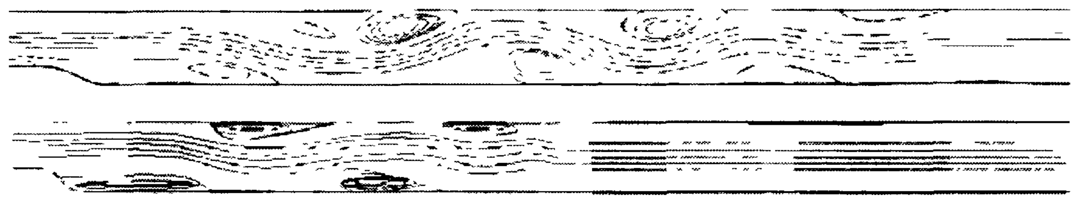

Figure 4.6 (f): Streamlines Downstream of the Indentation $(t_f = 0.7)$

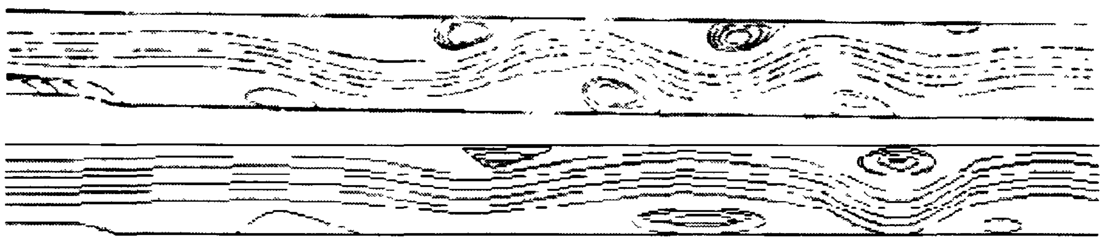

Figure 4.6 (g): Streamlines Downstream of the Indentation $(t_f = 0.8)$

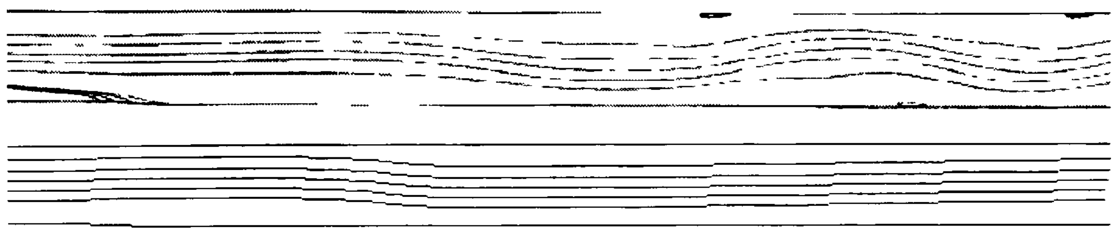

Figure 4.6 (h): Streamlines Downstream of the Indentation $(t_f = 0.9)$

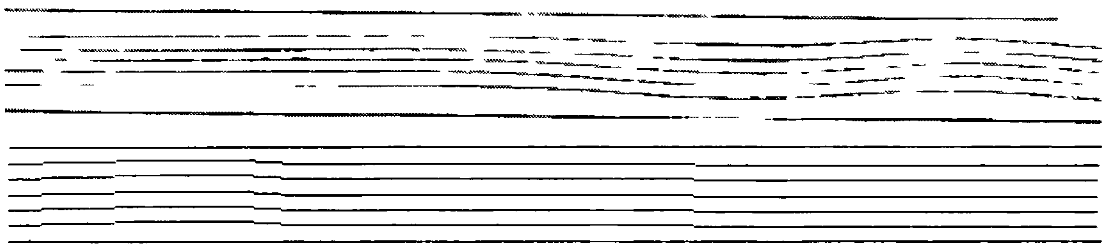

Figure 4.6 (i): Streamlines Downstream of the Indentation $(t_f = 1.0)$

CHAPTER 5 CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

A novel method for dynamic grid generation on two-dimensional domains with moving boundaries, with the boundary nodes coordinates prescribed as a function of time, has been developed. The full elliptic grid generation system is applied once to generate the initial grid on the domain, and a perturbed system of linear equations is obtained from the full elliptic equations by assuming a small amount of disturbance at each node. Solving the perturbed equations at subsequent times, based on the known amount of perturbation on the boundaries, results in the amount of perturbation at all interior nodes. The basic advantage of applying this method, rather than solving the full elliptic system for all times, is the significant reduction in the amount of time consumed for grid generation. Based on our limited tests, we can achieve about 80% reduction in the computational time using the grid perturbation method. This could be especially significant on domains with fine grids or for 3D domains, for which the computational time considerably increases. In order to avoid grid entanglement, the full elliptic system can be applied as needed after some time steps.

In Chapter 4 the finite difference formulation introduced by Zogheib and Barron [31] for steady flows has been extended to unsteady flows. The idea is to apply a SIMPLE-like scheme commonly introduced in finite volume formulations to link the velocity and pressure, when the momentum and continuity equations are discretized in a pure finite difference formulation. Hence, a pressure correction equation is solved for obtaining the amount of pressure correction at each grid point on the domain. The velocity and pressure fields are then corrected, and the momentum equations and pressure correction equation are solved again using the updated values from the previous solution. This procedure is resumed at each time step until the results are obtained for that time. The velocity and pressure fields of one time constitute the initial conditions for the next time step.

The grid generator and flow solver are applied to two test cases. In the first case, the unsteady solver is used to solve a steady flow problem. In the second, the flow through a channel with a moving boundary is simulated.

5.2 Recommendations

The method proposed for dynamic grid generation has shown great capability for efficiently generating good quality meshes in 2D. This approach can easily be extended to surface meshing and 3D volume meshing, where the computational savings will be even more significant. In the present formulation, the number of boundary nodes and interior nodes must remain the same throughout the calculation process, ie., for all time. This is related to the fact that the corners of the domain are not allowed to move. There is nothing inherent in the approach which forces this restriction, so it should be possible to allow movement of the corner nodes as well as insertion and deletion of grid lines as the domain deforms.

Furthermore, the control functions introducing in the perturbed system of equations are simply taken to be zero in the present work, since zero perturbation to the control functions seemed to give the best results in terms of less likelihood of grid entanglement. A considerable amount of investigation can be done on the behavior of the perturbed equations as these control functions change. The improvements which may be realized by choosing appropriate control functions include reducing the need for occasional application of the full elliptic system and improving the grid quality.

The results from the current flow solver can be improved in several ways. The time step applied in this research is four times larger compared to the time step applied by Demirdzic and Peric [23]. Taking a smaller time step can return more accurate results. The number of grid points can also be increased, which would reduce any truncation errors due to the finite difference approximations. A grid with clustering near the boundaries would generate more accurate results, since a finer grid would capture more points in the boundary layers, ie., in the regions with high flow gradients.

The flow solver itself could also be improved. For example, the first order upwinding scheme for the convective terms could be replaced by higher order schemes, such as third order upwinding, or through the use of Pade approximations. Higher order accurate time march methods, such as a second order Crank-Nicolson or fourth order Runge-Kutta scheme could be implemented. Improved velocity-pressure coupling procedures such as SIMPLEC or PISO could be used to accelerate the convergence.

Having established, in this thesis, the capability of the proposed approach to solve the unsteady incompressible Navier-Stokes equations, the above suggestions are promising avenues for future research.

REFERENCES

[1] Brackbill, J.U., Saltzman, J.S. Adaptive Zoning for Singular Problems in Two Dimensions. Journal of Computational Physics, Vol. 46, pp. 342-368, 1982

[2] Thompson, J.F., Soni, B.K., Weatherill. N.P. *Handbook of Grid Generation.* Editors, CRC Press, 1999

[3] Noh, W.F CEL: A Time-dependent Two-space Dimensional Coupled Eulerian-Lagrangian Code. *Methods in Computational Physics, Vol. 3: Fundamental Methods in Hydrodynamics*, pp. 117-179, 1964

[4] Franck, R.M., Lazarus, R.B. Mixed Eulerian Lagrangian Method. *Methods in Computational Physics, Vol. 3: Fundamental Methods in Hydrodynamics.* pp. 47-67, 1964

[5] Trulio, J.G. Theory and Structure of AFTON Codes, Report AFWL-TR-66-19, Air Force Weapons Laboratory: Kirtland Air Force Base, 1966

[6] Hirt, C.W., Amsden, A.A., Cook J.L. An Arbitrary Lagrangian-Eulerian Computing Method for All Flow Speeds. Journal of Computational Physics, Vol. 14, pp. 227-253, 1974, Reprinted in Journal of Computational Physics, Vol. 135, pp. 203-216. 1997

[7] Donea, J., Huerta, A., Ponthot, J.-Ph., Rodriguez-Ferren, A. Arbitrary Lagrangian-Eulerian Methods. Encyclopedia of Computational Mechanics, Vol. 1: Fundamentals. Eds. Stein, E., de Borst, R., Hughes, T.J.R. Chapter 14. John Wiley & Sons, 2004

[8] Ghia, K.N., Osswald, G.A., Ghia, U. A Direct Method for the Solution of Unsteady Two-Dimensional Incompressible Navier-Stokes Equations. Proceedings of the Symposium on Numerical and Physical Aspects of Aerodynamic Flows (2nd), California State Univ. Long Beach, AD-A134 151, 1983

[9] Ewing, R.E., Iliev, O.P., Margenov, S.D., Vassilevski, P.S. Numerical Study of Three Multilevel Preconditioners for Solving 2D Unsteady Navier-Stokes Equations. Computer Methods in Applied Mechanics and Engineering, Vol. 121, pp. 177-186, 1995

[10] Chorin, A.J. A Numerical Method for Solving Incompressible Viscous Flow Problems. Journal of Computational Physics, Vol. 2, pp. 12-26, 1967

[11] Peyret, R. Unsteady Evolution of a Horizontal Jet in a Stratified Fluid. Journal of Fluid Mechanics, Vol. 78, pp. 49-63. 1976

[12] Venkateswaran, S., Merkle, C.L. Evolution of Artificial Compressibility Methods in CFD. *Numerical Simulations of Incompressible Flows,* Hafez M.M. Editor, World Scientific Publishing Co., Singapore, pp. 35-49, 2003

[13] Merkle, C.L., Athavale, M. Time-Accurate Unsteady Incompressible Flow Algorithms Based on Artificial Compressibility. AIAA Paper-87-1137. 1987

[14] Tang, H.S., Sotiropoulos, F Fractional Step Artificial Compressibility Schemes for the Unsteady Incompressible Navier-Stokes Equations. Computers and Fluids, Vol. 36, pp. 974-986, 2007

[15] Patankar, S.V. *Numerical Heat Transfer and Fluid Flow.* Hemisphere Publishing Corporation, 1980

[16] Van Doormaal, J.P., Raithby, G.D. Enhancements of the Simple Method for Predicting Incompressible Fluid Flows. Numerical Heat Transfer, Vol. 7, pp. 147-163, 1984

[17] Raithby, G.D., Schneider, G.E. Numerical Solution of Problems in Incompressible Fluid Flow: Treatment of the Velocity-Pressure Coupling. Numerical Heat Transfer, Vol. 2, pp. 417-440, 1979

[18] Ralph, M.E.. Pedley, T.J. Flow in a Channel with a Moving Indentation. Journal of Fluid Mechanics, Vol. 190, pp. 87-112, 1988

[19] Viecelli, J.A. A Computing Method for Incompressible Flows Bounded by Moving Walls. Journal of Computational Physics, Vol. 8, pp. 119-143. 1971

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

[21] Peskin, C.S. Numerical Analysis of Blood Flow in the Heart. Journal of Computational Physics. Vol. 25. pp. 220-252, 1977

[22] Daly. B.J. A Numerical Study of Pulsatile Flow through Constricted Arteries. Proceedings of 4th International Conference on Numerical Methods in Fluid Dynamics (ed. R.D. Richtmeyer). Lecture Notes in Physics. Vol. 35, pp. 117-124. 1974

[23] Demirdzic, I.. Peric. M. Finite Volume Method for Prediction of Fluid Flow in Arbitrary Shaped Domains with Moving Boundaries. International Journal for Numerical Methods in Fluids, Vol. 10. pp. 771-790. 1990

[24] Strigberger, J. Numerical Perturbation Method for Approximate Solution of Poisson's Equation on a Moderately Deforming Grid. International Journal for Numerical Methods in Fluids, Vol. 9. pp. 599-607. 1989

[25] Wu, J., Rath, H.J. Finite Difference Method of Incompressible Flows with Rotation and Moving Boundary in a Non-Staggered Grid. Numerical Heat Transfer. Part B. Vol. 26, pp. 189-206, 1994

[26] Knupp, P., Steinberg, S. *Fundamentals of Grid Generation*, CRC Press, 1994

[27] Zhang, H., Karim Moallemi. M. MAGG - A Multizone Adaptive Grid Generation Technique for Simulation of Moving and Free Boundary Problems. Numerical Heat

Transfer, Part B. Vol. 27, pp. 255-276, 1995

[28] Anderson, J.D. *Computational Fluid Dynamics: the Basics with Applications.* McGraw-Hill Inc., New York, 1995

[29] Mastin, C.W., Thompson, J.F Transformation of Three-Dimensional Regions onto Rectangular Regions by Elliptic Systems. Numerical Mathematics, Vol. 29, pp. 397-407, 1978

[30] Barron, R.M. Improvements to Grid Quality and Cost of Multiblock Structured Grid Generation. Proceedings of the 4th Annual Conference of the CFD Society of Canada. CFD96, pp. 303-309, Ottawa, Canada, 1996

[31] Zogheib, B., Barron, R.M. Velocity-Pressure Coupling in Finite Difference Formulations for the Navier-Stokes Equations. International Journal for Numerical Methods in Fluids (under review)

[32] Zogheib, B. Velocity-Pressure Coupling in Finite Difference Formulations for the Navier-Stokes Equations. PhD Dissertation. University of Windsor. 2006

[33] Barron, R.M., Zogheib, B. A Finite Difference Calculation Procedure for the Navier-Stokes Equations on a Staggered Curvilinear Grid (to be published)

[34] Chung, T.J. *Computational Fluid Dynamics.* Cambridge University Press, 2002

[35] Pedley, T.J., Stephanoff. K.D. Flow Along a Channel with a Time-Dependent Indentation in One Wall: The Generation of Vorticity Waves. Journal of Fluid Mechanics, Vol. 160, pp. 337-67, 1985

VITA AUCTORIS

Atefeh Shadpour was born in September 1982 in Shiraz, Iran. She received her BSc in Mechanical Engineering with specialization in thermo-fluids from Amirkabir University of Technology, Tehran, Iran in December 2004. She is currently a candidate for the Master of Applied Science degree in Mechanical Engineering at the University of Windsor with specialization in Fluid Mechanics, and expects to graduate in November 2008.