To the Graduate Council:

I am submitting herewith a dissertation written by Yonghu Wenren entitled "A New Computational Method for Computing Flow over Complex Aerodynamic Configurations and Its Application to Rotor/Body Computation Using Cartesian Grids" I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Aerospace Engineering.

John S. Steinhoff

John S. Steinhoff, Major Professor

We have read this dissertation
and recommend its acceptance:

Robert T Roach

Roy J. Schulz

VC Reddy

John E. Carruthers

Accepted for the Council:

Lawrence

Associate Vice Chancellor
and Dean of the Graduate School

A NEW COMPUTATIONAL METHOD FOR COMPUTING FLOW OVER COMPLEX AERODYNAMIC CONFIGURATIONS AND ITS APPLICATION TO ROTOR/BODY COMPUTATION USING CARTESIAN GRIDS

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Yonghu Wenren

May, 1997

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my advisor, Professor John Steinhoff, for his continual assistance, support and technical guidance during my graduate studies. Sincere thanks are extended to the committee members: Dr. K. C. Reddy, Dr. John Caruthers, Dr. Robert Roach and Dr. Roy Schulz for their helpful comments. Recognition is also given to Frank Caradonna of NASA Ames Research Center for his efforts in this respect. A special note of thanks is due Dr. Clin Wang of Flow Analysis, Inc. for his numerous and valuable comments. Appreciation is also expressed to my fellow student, Dr. David Underhill for introducing the "FISHPAK" subprogram.

Special thanks to my employer, Flow Analysis, Inc. (FAI) for the financial support. This work was partially supported by NASA SBIR contracts, No. NAS2–13489, No. NAS1–20358 and No. NAS2–14025, through FAI. I would like to express my special appreciation to Ms. Linda Engels for her assistance in preparing figures, to Ms. Mary Lo and other library staff for their assistance, to Mr. Friedhelm Decker of VW, Germany, for providing the Passat Limousine B3 automobile configuration data, to Dr. Henry Jones of NASA Langley Research Center for supplying the Apache–A Helicopter body configuration data, to Mr. John Riley and Mrs. Lorna Riley for checking spelling and grammar.

Finally, from the bottom of my heart, I would like to thank my wife, my children, my parents and all my friends. This work would not have been possible without their patience, support and encouragement.

ABSTRACT

A new numerical method for efficiently computing vortex-dominated flows over complex aerodynamic configurations is developed . This method uses only a fixed, uniform Cartesian grid, no body conforming grid is required. The complex geometry surface is described by a smooth scalar function "F", which is defined at each grid node. By using Vorticity Confinement, this method effectively confines the vorticity to a narrow region even on coarse computational grids and for low order discretization schemes. The flow both inside and outside the configuration is considered, although in aerodynamic applications, the internal flow is fictitious. The no–slip boundary condition is imposed on solid body surfaces by eliminating the flow inside the configuration. Unlike other general Cartesian methods, no specific logic is needed to determine the body surface in the present method. Also, the vorticity can be shed from smooth surface as well as surfaces with sharp corners.

Vorticity Confinement involves adding a simple term to the Navier–Stokes fluid dynamic equations. When discretized and solved, these modified equations admit convecting, concentrated vortices which maintain a fixed size and do not spread, even if there is numerical diffusion.

Numerical results are presented for flows around simple and complex configurations which were investigated with the present method. As an application of this method, preliminary numerical flow solutions of a combined helicopter blade and real helicopter body are presented.

The code developed in the present method uses a relatively coarse, fixed,

Cartesian grid and requires much less computing time compared to high order accurate flow solvers, while capturing the vorticity efficiently and accurately. The code will be very useful for engineering analysis and design of aircraft as well as automobile.

Contents

List of Figures

NOMENCLATURE

a	Weighting factor used in vorticity confinement
$d_{i,p}$	Distance from point i to point p
F	A smooth scalar function that defines the solid body surface
$\vec{n}$	An unit vector points away from the centroid of the vortex
p	Pressure
$\vec{q}$	Velocity which is defined at the grid node
$\vec{q}_\infty$	Free stream velocity
u, v, w	Velocity components in x, y, z direction respectively
$\vec{X}_{i,j,k}$	Coordinate of point (i, j, k) in Cartesian grid
Δt	Computational time step
ϵ'	A small number for avoiding numerical overflow
ε	Vorticity confinement parameter
η	A scalar field that has a local minimum on the centroid of the vortical region
μ	Diffusion coefficient
ρ	Density

ϕ	Velocity potential
ω	Relaxation factor
$\vec{\omega}$	Vorticity

Chapter 1

INTRODUCTION

1.1 Background

Computational Fluid Dynamics (CFD) is playing an important role in the design, analysis and support of aircraft, automobile, and ship, as well as other transportation systems. The value of CFD to industry is in its application. Two essential characteristics of a valuable CFD method are timeliness and the ability to handle complex geometry. Most aerodynamic configurations are geometrically complex. For example, the Apache Helicopter, shown in Fig. 1.1, has wing, tail, fuselage, sensor pod and etc. To aid in the design of such a complex configuration, the computational method must faithfully represent the dominant flow physics and the required geometric complexity and be capable of providing reliable solutions in a timely, affordable manner. Most of these flow are vortex–dominated flows.

Because of the importance of vortex–dominated flows, many researchers have attempted to find viable numerical methods for their prediction over the past several decades. In spite of these efforts, accurate and efficient methods still do not exist for the computation of general flows with strong, concentrated vortices. As stated in Reference [1]:

> Conventional computational methods for vortex–dominated flows can be grouped into two classes: Eulerian Method and Lagrangian

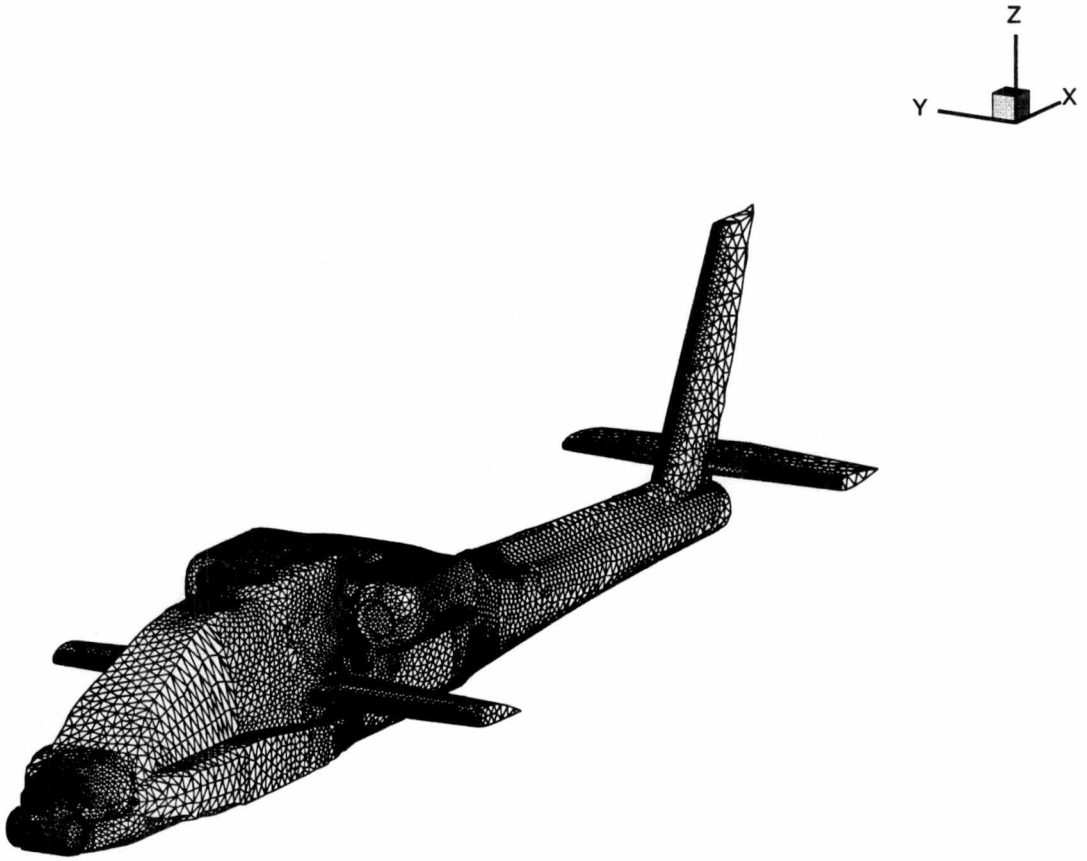


Figure 1.1: Apache Configuration with Mesh.

Method. In the Eulerian approach, body–fixed grids can be used effectively with discretized flow equations for flows at low Reynolds number where shed vorticity is spread over a rather wide region. However, for large Reynolds number flows, where this region is thin, that is, vorticity is concentrated, it is very difficult to include the number of grid points required for adequate resolution.

There are two conventional numerical techniques that have provided some improvement in treating thin vortical regions: high–order discretization and adaptive grid schemes. Examples of these approaches include the use of a fifth–order discretization of the Navier–Stokes equations to solve the problem of a vortex impinging on an airfoil, together with a fine, regular grid in the entire domain [2]. This requires a large amount of computing since the grid must be fairly fine, even though fairly efficient implicit solvers can be used since the grid is regular. In the other approach, adaptive grid schemes have been used for problems involving, for example, a vortex impinging on an airfoil [3], a steady vortex shed at the tip of a straight wing [4], and a vortex sheet shed from the leading–edge of a delta wing [5]. It can be shown that despite the adaptive embedding of extra grid points in the vortical region, these methods apparently are still not able to capture the shed vortex without having it spread over a region much larger than measured in experiments. In addition, the unstructured and moving grids require complex logic and bookkeeping. Without regular grids, conventional

and efficient implicit methods usually cannot be used.

In the many Lagrangian approaches, use is made of the fact that the internal structure of vortical regions, such as vortex sheets, are not important since the dimension of these structures are small compared to other dimensions of the problem. As long as the centroid surfaces of these vortex sheets are accurately computed, together with the total vorticity surrounding each point on the centroid surface, the overall flow solution will be accurate. An *ad hoc* structure can then be used instead of resorting to a detailed, high–resolution Navier–Stokes solver. Usually a "spreading" function is specified that, effectively, defines the internal structure of vortical regions treated with this approach. For these reasons, some of the most efficient methods for treating thin vortical flows currently involve Lagrangian marker–based schemes, where vorticity or circulation is assigned to individual markers which convect through the flow field. These methods, in the form of "Vortex Lattice" or "Vortex Blob" techniques for incompressible flows [6] and "Vortex Embedding" techniques for compressible flows [7], entail representation of vortex sheets or vortex filaments by surfaces or lines defined by markers.

Unfortunately, there are disadvantages to these Lagrangian methods. Since the vortical regions are defined by connected sets of markers, the topology of each region should be known beforehand so that suitable arrays of markers can be computationally defined. In general,

the vortical regions may interact with solid surfaces and their topology may change. This requires new specifications of the markers and re–connection. In addition, vortical regions cannot easily be made to merge in a natural way if they are defined by markers. Lagrangian methods which use large numbers of unconnected markers with overlapping structure also appear to require information on the locations of vortical regions for efficient allocation of markers.

A new computational technique, called *Vorticity Confinement*, developed by Steinhoff [8]–[9], is fundamentally different from the previously described methods in that it involves adding a term to the flow equations, and therefore modifying the basic equations. With the added term, the flow equations admit solutions with concentrated vortices that can convect without spreading, even if the basic equations have numerical diffusion. For this reason, simple, low order diffusive numerical schemes can be used to discretize and solve the modified flow equations, without resulting in vortices that spread as they convect. This approach is similar to shock capturing where the detailed internal dynamics of shocks are not computed, but rather a modified set of equations is solved which result in a shock spread over a few grid cells. The resulting internal structure satisfies conservation laws in integral form.

The Vorticity Confinement method has the flexibility and generality of standard Euler/Navier-Stokes methods, but treats vortical regions without numerical diffusion as they convect through the flow even in regions where the computational grid is relatively coarse. This method uses only a fixed, Eulerian grid with

no markers and no special logic concerning shed vortices: it consists, basically, of a velocity correction to be applied at each time step at grid points where velocity values are obtained by the basic solver. This correction is applied only in relatively coarse grid regions, as vortical fluid is convecting. As such, it is capable of being used in conjunction with conventional Euler/Navier-Stokes methods which can serve as the basic flow solver.

The Vorticity Confinement scheme with a simple "split velocity" formulation [10] has been used to solve a large number of flow problems including Delta Wing [11] and Interacting Vortex Ring cases in 3–D [9] and Blade– Vortex Interaction problems in 2–D. It has also been used with a compressible formulation for 3–D Wing and 2–D Blade–Vortex Interaction problems.

1.2 Objectives and Organization of the Present Study

1.2.1 Objectives

The main objective of the present research is to develop a new computational method that can be used to simulate flow with generated, convected, concentrated vorticity in the vicinity of complex aerodynamic configurations. This method uses only a fixed, uniform Cartesian grid. In this method, the solid body surface is described by a smooth scalar function "F", which is defined at each computational grid node. No specific logic to determine the body surface is required. The flow solution can be obtained by a conventional Euler/Navier–Stokes flow solver with the Vorticity Confinement method to capture the vorticity. By using a confinement function based on a smooth "F" function, vorticity can be shed

from smooth surfaces as well as surfaces with sharp corners, and can be confined within a few grid cells. By incorporating a moving embedding grid technique, this method can solve rotor/body flow problems.

1.2.2 Organization

The dissertation is organized into four main sections. First, the background of the present study is presented. Conventional methods for treating vortex dominated flow are briefly reviewed. Second, the present numerical methods used for the simulation of flow over complex configurations are described. Third, a thorough analysis of the simulation results is given. Comparison with experimental results is made whenever possible. Simulation results are presented graphically to illustrate the qualitative character of all key flow structures. Finally, conclusions and recommendations are presented relating to the present work.

Chapter 2

PHYSICAL PROBLEM AND NUMERICAL APPROACH

2.1 Physical Problem

The physical problems solved in the present study involve flows over a complex aerodynamic configuration. Although the numerical method employed here is applicable to both compressible and incompressible flows, only incompressible flow is considered in the present study. Viscous effects, including flow separation from solid body surfaces are simulated in the present computational scheme, as well as convection of free vorticity.

2.1.1 Governing Equation

The governing differential equations are the incompressible unsteady flow equations:

Continuity Equation:

$$\nabla \cdot \vec{q} = 0 \tag{2.1}$$

Momentum Equations:

$$\partial_t \vec{q} = -(\vec{q} \cdot \nabla)\vec{q} + \nabla p/\rho + \mu \nabla^2 \vec{q} \tag{2.2}$$

where t is time; p, ρ, and μ represent pressure, density, and the diffusion coefficient, respectively, and $\vec{q}$ is the velocity vector. For example, in Cartesian coordinates, $\vec{q}$

is given by

$$\vec{q} = u\vec{i} + v\vec{j} + w\vec{k}. \tag{2.3}$$

In the present study, a "split velocity" scheme [10] is employed, the momentum equations 2.2 are solved in two steps, velocity convection and velocity diffusion. The variables p, ρ, and μ do not explicitly appear in the flow solution.

2.1.2 Boundary Conditions

At the far field, the flow is equal to free stream, i.e:

$$\vec{q}|_B = \vec{q}_\infty$$

where B indicates the far field boundary and $\vec{q}_\infty$ is the freestream velocity.

At the solid body surface, the no–slip condition is imposed

$$\vec{q}|_s = 0$$

where s indicates the solid body surface.

2.2 Numerical Approach

To represent the complex configuration in the computational domain, a scalar function "F" is introduced. This function is defined at each grid node. The solid body surface is determined by $F = 0$. The flow, both inside and outside the complex configuration, is solved by a conventional Euler/Navier–Stokes flow solver with Vorticity Confinement on a fixed Cartesian grid. The boundary condition on the solid body surface is treated by applying a "confinement" method based on the

F function. With a moving embedding grid technique, this method can be used for solving rotor/body flows.

2.2.1 Salient Features of the Method

A significant feature of this method is that the flow is solved in a fixed Cartesian grid. No body conforming grid is required. The solid body surface is defined by a smooth scalar function–F, so that there are no other logic and bookkeeping required for determining the boundary region. Another advantage of using a Cartesian grid is that a conventional flow solver can be employed in this method. The governing equations can be easily discretized in the Cartesian grid. With the vorticity confinement scheme, vorticity can be constrained to thin regions in the flow field including those near the body surfaces. The velocity inside the body, where $F < 0$, is driven to zero during the solution process, and the no–slip condition is imposed on the solid surface, where $F = 0$. By using the confinement method based on the function F, vorticity can be separated from smooth as well as sharp surfaces, and, the total vorticity in simulated boundary layers is correct.

2.2.2 Numerical Procedure

1) As a preprocessor, a scalar function F is generated based on a given well–defined body surface geometry. This function F is defined at each grid node such that:

$$F \begin{cases} < 0 & \text{inside body} \\ = 0 & \text{on the body} \\ > 0 & \text{outside body} \end{cases}$$

2) The flow field is first solved by an existing flow solver. The primary or primitive (ρ, u, v, w) variables are only used in this computation. Low order accurate and diffusive schemes can be employed.

3) Vorticity confinement correction is added to the velocity field everywhere outside the solid body surface. It constrains the vorticity within $2 \sim 3$ grid cells.

4) For the region inside the solid body surface the modification scheme is to reduce the velocity magnitude to zero, while simultaneously convecting the vorticity gradually toward the solid surface during each computation. By doing this repeatedly in the computation, no special treatments of the solid boundary and the grid arrangement are necessary. The velocity inside the solid will become zero and vorticity generated by the solid/fluid interface is correctly accounted for in a near–surface, external layer on the simulated body in a robust way.

5) Finally, for incompressible flow, conservation of mass is ensured by solving a Poisson equation. After Steps 3 and 4, the velocity field has the property of confined vorticity (against numerical diffusion) in the vortical region outside the solids and ever-diminishing velocity inside the solids. This velocity field, however, may violate the mass conservation. To impose mass conservation, a Poisson equation derived from the continuity equation is solved. That is, a velocity–correction

potential is solved for from the equation

$$\nabla^2\phi = -\nabla \cdot \vec{q} \tag{2.4}$$

where $\vec{q}$ is the modified velocity vector obtained after Steps 3 and 4. The final corrected velocity for the new time level is then obtained by adding the vector $\nabla\phi$ to the velocity vector $\vec{q}$ obtained after Steps 3 and 4. Notice that the corrected velocity $\nabla\phi$ does not change the vorticity field that exists after Steps 3 and 4.

The detail numerical computation formulations for each step above are described in Section 2.4.

2.3 Vorticity Confinement Method

2.3.1 Description of Method

The Vorticity Confinement method is based on adding a simple term to the Navier–Stokes fluid dynamic equations. When discretized and solved, these modified equations admit convecting, concentrated vortices which maintain a fixed size and do not spread, even if there is numerical diffusion in the computed flow field. In many flows this is sufficient to provide accurate results. The main point is that the final results are often not sensitive to the exact computed vorticity distribution within the vortex core as long as it is not spread too much by the numerical scheme. The solution only degrades if vortices spread so much that their cores overlap or intersect solid surfaces; otherwise, it is accurate. (This is obviously the case, for example, for interacting axisymmetric vortices since the induced velocity outside a vortex core at a given point does not depend on the

distribution inside - as long as the core does not grow so large that it overlaps the given point.)

This modified scheme uses only vorticity and primitive variables on an Eulerian grid, but has greatly enhanced vortex capturing and convection capability: It allows the convection of highly concentrated vortices without numerical diffusion.

In this method, vorticity is represented on the computational grid as in standard methods, but is captured in a minimal number of grid cells (~ 2), with no numerical diffusion. As in conventional methods, only primitive variables and vorticity are used, with no auxiliary, geometric variables.

Thus, it can be seen that a standard Navier-Stokes solver, used together with the Vorticity Confinement technique, can provide very good results for cases where concentrated vortices are shed from the boundary layer and their effects on the creation of new vortices must be accurately accounted for.

2.3.2 Basic Formulation of Vorticity Confinement Method

The method involves adding a term to the momentum part of the continuum Euler/Navier-Stokes equations. The extra "Vorticity Confinement" term is local and simple to discretize. It is non-zero only within the vortical regions and does not change the total vorticity or mass within those regions. The basic technique is applicable to general compressible and incompressible flows. For conciseness, the "Vorticity Confinement" method is described here for incompressible flow only.

For 3D unsteady flows,

$$\nabla \cdot \vec{q} = 0 \tag{2.5}$$

$$\partial_t \vec{q} = -(\vec{q} \cdot \nabla)\vec{q} + \nabla p/\rho + \mu \nabla^2 \vec{q} + \varepsilon \vec{s} \tag{2.6}$$

where $\vec{q}$ is the velocity vector; t is time; p, ρ, and μ represent pressure, density, and the diffusion coefficient (representing numerical diffusion), respectively. The additional term, $\varepsilon\vec{s}$, is the "Confinement" term where the numerical coefficient ε controls the size of the convecting vortical regions. The "Confinement" term has the form:

$$\vec{s} = -\vec{n} \times \vec{\omega} \tag{2.7}$$

where

$$\vec{n} = \nabla\eta/|\nabla\eta| \tag{2.8}$$

and

$$\vec{\omega} = \nabla \times \vec{q} \tag{2.9}$$

is the vorticity vector. η is a scalar field variable that has a local minimum on the centroid of the vortical region:

$$\eta = -|\vec{\omega}| \tag{2.10}$$

For the confinement term, $\vec{n}$ is a unit vector pointing away from the centroid of the vortical region and the term serves to convect $\vec{\omega}$ back towards the centroid as it diffuses away. This convection increases the diffusion term and a steady-state form results when the two terms become balanced. It is noted that steady-state solutions exist for any positive value of ε. For the present types of flows, it appears

better to discretize the set of Equations 2.5 and 2.6 for problems which have thin, well-behaved vorticity distribution, even in the presence of the numerical diffusion, than to discretize the unmodified equations which only admit vorticity regions that continue to spread, if there is any numerical diffusion.

2.3.3 Salient Features of the Vorticity Confinement Method

An important feature of the vorticity confinement method is that the correction is non–zero only in the vortical regions. The correction of the velocity effectively convects vorticity back towards the local maximum of the vortical region and therefore the total vorticity is conserved. It can be shown that the correction also conserves mass and averaged momentum. Unlike artificial viscosity–like terms which are non–zero everywhere except near discontinuities, this correction vanishes outside of vortical regions. Another important feature concerns the total change induced by the correction in mass, δI_ρ, and velocity, $\delta \vec{I}_\omega$, integrated over the vortical regions. In general 3–D flows, because of the vanishing of the correction $\vec{s}$, outside the vortical regions,

$$\delta I_\rho = \varepsilon \int \nabla \cdot \vec{s} dv = 0$$

$$\delta \vec{I}_\omega = \varepsilon \int \nabla \times \vec{s} dv = 0$$

where the integration is done over the vortical regions. Another important quantity that can be conserved with the method is momentum. Consider a thin vortical "line" that is slowly varying along its length, and a 2–D section of this line, and

write for the change in momentum there:

$$\delta \vec{I}_M = \varepsilon \int \vec{s} dA$$

where the integral is over the 2–D section. In this case:

$$\vec{s} = \vec{\omega} \times \frac{\nabla \omega}{|\nabla \omega|} \times \vec{k}$$

where ω is the value of $\vec{\omega}$ in the direction of a vortex line, $\vec{k}$, and $\nabla\omega$ and $\vec{s}$ are in the 2–D plane of the section. Then

$$\delta \vec{I}_M = \varepsilon \rho \vec{J} \times \vec{k}$$

where

$$\vec{J} = \int \omega \frac{\nabla \omega}{|\nabla \omega|} dA$$

In general, $\vec{J}$ will not be zero. However, for the class of ω distributions that have two axes of symmetry (such as elliptical distributions) $\vec{J}$ will vanish due to symmetry. The confinement term is intended to be used where thin vortical regions are convected over relatively long distances and where the velocity (except for that due to the vortex) varies slightly on a length scale of the vortex diameter. In that case the viscous terms are expected , either due to the basic numerical convection process or added explicitly, to symmetrize the ω distribution, since any strong, concentrated vortex will be spinning rapidly. As a result, the time averaged values of $\vec{J}$ and hence $\delta \vec{I}_M$ are expected to be small.

2.3.4 Closed–Form Solution: Axisymmetric Vortex

The results of applying of the "Vorticity Confinement" method can be illustrated by a computation of a simple 2D flow, where an isolated axisymmetric

vortex moves in a uniform flow. If $\vec{r}$ is a position vector defined with respect to the center of the vortex, then Equations 2.8 and 2.10 will give

$$\vec{n} = \vec{r}/r \tag{2.11}$$

where $r = |\vec{r}|$. The velocity of the flow field can be represented by

$$\vec{q} = \vec{q}_\infty + T(r,t)\vec{e}_\theta \tag{2.12}$$

where $\vec{q}_\infty$ is the uniform velocity, T is the azimuthal velocity magnitude contributed by the vortex, and $\vec{e}_\theta$ is a unit vector in the azimuthal direction. Substituting this into the modified momentum equation, 2.6, in a frame convecting with $\vec{q}$, one has

$$\partial_t \vec{q} = \mu \nabla^2 \vec{q} - \varepsilon \vec{n} \times \vec{\omega} \tag{2.13}$$

If $\varepsilon = 0$, the solution is

$$T = [T_0/r](1 - e^{-r^2/2\mu t}) \tag{2.14}$$

where T_0 is a constant. This, of course, results in a continually spreading vortical region with radius proportional to $\sqrt{2\mu t}$ and no non-trivial steady solution.

When $\varepsilon > 0$, the steady solution of Equation (9) with $\partial_t \vec{q} = 0$ is

$$T = [T_0/r][1 - (1 + r/a)e^{-r/a}] \tag{2.15}$$

where $a = \mu/\varepsilon$ is the length scale. The existence of the steady solution shows that the added confinement term effectively balances the diffusion by convecting the vorticity back toward the vortex center.

2.3.5 Confinement Term

The confinement term is added explicitly at each time step. A formulation that has a bias towards smaller values of $\eta(-|\nabla\omega|)$ is used. The correction then transports vorticity towards regions of small η, while conserving total vorticity. This has proven to be a robust scheme. If a grid cell with velocity defined at the nodes is considered , then the box-type central differencing used to compute the curl results in an ω defined at the cell center. Then average values of η at the nodes are computed, and a unit vector pointing away from the centroid of the vortex are evaluated at the cell center:

$$\vec{n} = \frac{\nabla\eta}{|\nabla\eta|}$$

The correction to be added to the velocity is then simply

$$\delta\vec{q}_2 = -\varepsilon\Delta t a_l \vec{n} \times \vec{\omega} \tag{2.16}$$

where ε is a relaxation factor (which can depend on grid cell dimensions) and a_l are weights computed at cell nodes (labeled l) to enforce the biasing. A simple form of weighting function is used:

$$a_l' = min(0, \eta_l - \bar{\eta})$$

where

$$a_l = \frac{a_l'}{\Sigma_l a_l'}$$

and where $\bar{\eta}$ is the averaged value of η, over the cell nodes.

2.4 Cartesian Grid Method

A scalar function F is first defined on the entire grid to describe the complex configurations. It is convenient to define a function that is zero at the solid surface, positive outside the solid, and negative inside the solid. This function can be defined analytically if the solids are of simple shape, or defined numerically if the solids are complex. The whole region is solved by the following numerical steps, where an initial uniform flow is assumed:

1. Velocity convection and diffusion: the velocity at every grid node, except the far boundary, is solved by an existing 3–D incompressible Navier-Stokes flow solver;

2. Velocity correction using vorticity confinement: the velocity within the exterior region, where $F > 0$, and vortical, is modified by use of the vorticity confinement where $\vec{n}$ is based on Equation 2.8;

3. Body surface boundary condition: the velocity within the interior region, where $F < 0$, is first multiplied by $1./(1-F)^2$. The velocity is then modified by the confinement but with the normal vector, $\vec{n}$, based on $\eta = |F|$

4. Mass conservation: the velocity field is updated and corrected by enforcing the continuity condition of equation 2.1.

The details of the numerical computational procedures are described in the following sections.

2.4.1 F Function

A scalar function F is defined on the fixed Cartesian grid. Several numerical methods have been used to obtain distributions of this function that define solid body surfaces. Numerical methods used in this study are summarized below.

a) A simple analytical geometry body surface.

The F function can be obtained from an analytical function for a simple body geometry. For example, for an ellipsoid in a 3–D Cartesian coordinates, (x, y, z). The following can be written for the function F in such case:

$$F = \alpha(x - x_0)^2 + \beta[(y - y_0)^2 + (z - z_0)^2] - a^2 \tag{2.17}$$

where x_0, y_0, and z_0 are the center of ellipsoid, α, β, and a the parameters to determine the shape of ellipsoid. The ellipsoid surface is defined by $F = 0$. The outside region, $F > 0$, is where the flow physically exists.

b) Body surface is given by a number of points.

An iterative method is employed in this case. Initially, a positive number, eg. 1, is assigned to F at outer boundary while a negative value, eg. -1 is set at a point inside the body surface. (These F values are fixed during the iteration procedure). F value of zero is initially assigned in elsewhere. At each iteration, the F value at each given point is calculated, then the F values at the neighbor grid points are redistributed to ensure the F value at the given points to be zero.

A simple iteration method is used here for computing the F value at a

grid node for the next iteration.

$$\begin{aligned} F^{n+1}_{i,j,k} = \; & \omega F^{n}_{i,j,k} + \tfrac{(1-\omega)}{6}(F^{n}_{i+1,j,k} + F^{n}_{i-1,j,k} + \\ & F^{n}_{i,j+1,k} + F^{n}_{i,j-1,k} + F^{n}_{i,j,k+1} + F^{n}_{i,j,k-1}) \end{aligned} \tag{2.18}$$

where ω is the relaxation factor.

At each given surface point p, the value of F is computed by the interpolation function:

$$F_p = I(F, \vec{X}_p) \tag{2.19}$$

where $\vec{X}_p$ is the location of the surface point p and $I(Q, \vec{X})$ is an interpolation function for calculating the value of Q field at location $\vec{X}$. A standard bilinear interpolation is used here. For a uniform Cartesian grid in 3–D, $dx = dy = dz = 1$, the function $I(Q, \vec{X})$ has the following form:

$$\begin{aligned} I(Q, \vec{X}) = \; & (1-r_x)(1-r_y)(1-r_z)Q_{i,j,k} + r_x(1-r_y)(1-r_z)Q_{i+1,j,k} + \\ & (1-r_x)r_y(1-r_z)Q_{i,j+1,k} + (1-r_x)(1-r_y)r_zQ_{i,j,k+1} + \\ & r_xr_y(1-r_z)Q_{i+1,j+1,k} + r_x(1-r_y)r_zQ_{i,j,k+1} + \\ & (1-r_x)r_yr_zQ_{i,j+1,k+1} + r_xr_yr_zQ_{i+1,j+1,k+1} \end{aligned} \tag{2.20}$$

where i, j, k are the index of the grid cell, in which the point $\vec{X}$ is located, in x, y, z direction respectively, and

$$\vec{X} = \{x, y, z\}$$

$$r_x = x - x_{i,j,k}$$

$$r_y = y - y_{i,j,k}$$

$$r_z = z - z_{i,j,k}.$$

Then the value of F at the neighbor grid nodes of p is redistributed by a distance weighting procedure

$$F_l = F_l - F_p W_l \tag{2.21}$$

where the weighting factor

$$W_l = \frac{1}{d_{p,l}} / \sum_{l=1}^{8} \frac{1}{d_{p,l}}$$

where $d_{p,l} = |\vec{X}_l - \vec{X}_p|$ is the distance from grid node l to surface point p.

As the result, although the F values at the given points are not exactly zero, the F function varies smoothly from negative to positive F–regions, and the points $F = 0$ define the body surface.

c) The body surface is given by a number of panels.

In most cases, the complex geometry will be given by a number of connected panels. In such a case, the F function can be defined as a distance function. The F value at each grid node is defined as the normal distance from that grid node to the closest panel with an assigned positive sign outside the body, and a negative sign inside the body.

To simplify the computation, for each grid node, the normal distance from that grid node to the closest panel is approximately equal to the minimum distance from that grid node to the center of all panels.

For a panel with n nodes, the center of this panel is obtained by:

$$\vec{X}_c = \frac{1}{n}(\sum_{m=1}^{n} \vec{X}_m). \tag{2.22}$$

To compute the F function at each grid node, i, the distances from that

grid node to all panel centers are calculated first:

$$\begin{aligned} d_{i,p} &= |\vec{X}_{p_c} - \vec{X}_i| \\ &= \sqrt{(x_{p_c} - x_i)^2 + (y_{p_c} - y_i)^2 + (z_{p_c} - z_i)^2} \end{aligned} \tag{2.23}$$

where $\vec{X}_{p_c}$ is the coordinate of the center point of panel p. Then the distance from the grid node $\vec{X}_i$ to the center of closest panel is

$$d_i = min(d_{i,1}, d_{i,2}, \ldots, d_{i,p}, \ldots, d_{i,np})$$

where np is the total number of panels.

To determine the sign of the F value, the normal direction of the closest panel p, $\vec{N}_p$, is calculated.

$$\vec{N}_p = \vec{AB} \times \vec{AC} \tag{2.24}$$

$$(\vec{NP})_x = (y_B - y_A)(z_C - z_A) - (y_C - y_A)(z_B - z_A)$$

$$(\vec{NP})_y = (z_B - z_A)(x_C - x_A) - (z_C - z_A)(x_B - x_A)$$

$$(\vec{NP})_z = (x_B - x_A)(y_C - y_A) - (x_C - x_A)(y_B - y_A)$$

where $\vec{AB}, \vec{AC}$ are the sides of panel p shown in Fig 2.1.

The sign of an F value is calculated as:

$$S = sign(\vec{N}_P \cdot \vec{PH}) \tag{2.25}$$

where $\vec{PH}$ is a vector from the center of closest panel p to the grid node i, and

$$(\vec{PH})_x = (x_i - x_{p_c})$$

$$(\vec{PH})_y = (y_i - y_{p_c})$$

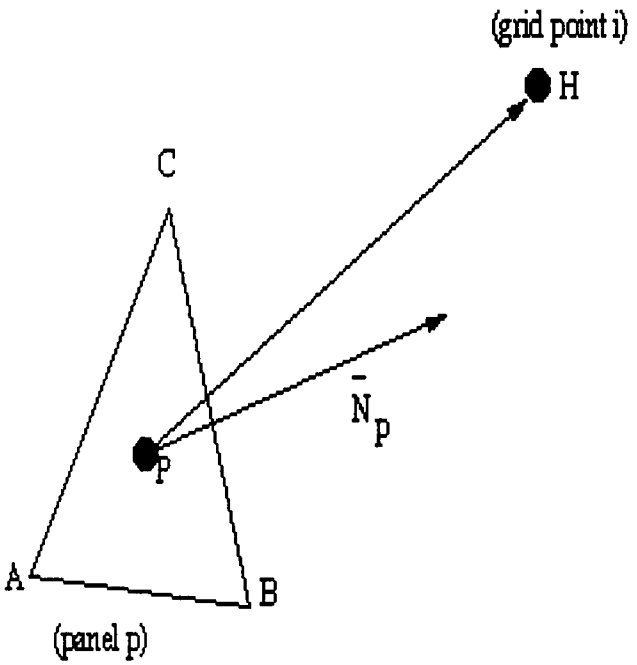


Figure 2.1: Distance from Grid Node i to Surface Panel p.

$$(\vec{PH})_z = (z_i - z_{p_c})$$

$$\vec{NP} \cdot \vec{PH} = (\vec{NP})_x(\vec{PH})_x + (\vec{NP})_y(\vec{PH})_y + (\vec{NP})_z(\vec{PH})_z.$$

Then, the F value at the grid node i is determined by

$$F_i = (S)d_i.$$

Finally, the values of the F function are normalized by dividing all F values by the minimum value of F, $F = F/F_{min}$, so that the minimum F which appears in the center of the solid body is equal to -1.

2.4.2 Velocity Convection and Diffusion

"Convected" velocities are first computed on the Cartesian grid,

$$\vec{q}_1^{n+1} = C\vec{q}^n \tag{2.26}$$

where C is the convection procedure, $\vec{q}_1^{n+1}$ is the velocity field after convection, $\vec{q}^n$ is the velocity field in previous time step. In the continuum limit (for small time step Δt), 2.26 can be written:

$$\vec{q}_1^{n+1} = \vec{q}^n - \Delta t(\vec{q}^n \cdot \nabla)\vec{q}^n \tag{2.27}$$

Any existing accurate numerical convection routine could be employed here. A simple, first order in time and space, robust explicit scheme was used in the present study.

First, the location of a flow particle is found by a one step backward Euler method, in 3-D Cartesian grid system, for each inner grid node i, j, k, it can be

formulated as:

$$\vec{X}_{i,j,k_{imag}} = \vec{X}_{i,j,k} - \Delta t \vec{q}^{n}_{i,j,k} \tag{2.28}$$

for $1 < i < imax$, $1 < j < jmax$, $1 < k < kmax$, where Δt is the time step, $\vec{q}^{n}_{i,j,k}$ velocity at the grid node i,j,k, $\vec{X}_{i,j,k}$ coordinate of grid node i,j,k. $imax, jmax, kmax$ are the total number of grid nodes in i,j,k direction respectively.

Then, the velocity field at next time level at node i, j, k, is interpolated at point $\vec{X}_{i,j,k_{imag}}$ from previous velocity field:

$$\vec{q}^{n+1'}_{1_{i,j,k}} = I(\vec{q}^{n}, \vec{X}_{i,j,k_{imag}}) \tag{2.29}$$

where I is the interpolation function described in Section 2.4.1.

This numerical procedure is equivalent to a first order upwind scheme if the time step Δt is small such that the CFL number less that 1. The advantage of using the present method is that the flow is stable even if the time step was relatively large.

A velocity diffusion step is applied to velocity field after velocity convection, to simulate the physical viscous effects whish are not computed in the present study. For each grid node,

$$\begin{aligned} \vec{q}^{n+1}_{1_{i,j,k}} = \; & \alpha \vec{q}^{n+1'}_{1_{i,j,k}} + \tfrac{1}{6}(1-\alpha)[\vec{q}^{n+1'}_{1_{i+1,j,k}} + \vec{q}^{n+1'}_{1_{i-1,j,k}} + \\ & \vec{q}^{n+1'}_{1_{i,j+1,k}} + \vec{q}^{n+1'}_{1_{i,j-1,k}} + \vec{q}^{n+1'}_{1_{i,j,k+1}} + \vec{q}^{n+1'}_{1_{i,j,k-1}}] \end{aligned} \tag{2.30}$$

where $0 \leq \alpha \leq 1$ is the smoothing factor.

2.4.3 Velocity Correction Using Vorticity Confinement

To overcome the effects of numerical diffusion and capture the vorticity, vorticity confinement is employed after the calculated velocity convection. A velocity correction, $\delta\vec{q}_2$, is made on the grid so that, at each grid node, vorticity is confined:

$$\vec{q}_2^{n+1} = \vec{q}_1^{n+1} + \delta\vec{q}_2^{n+1} \tag{2.31}$$

where $\vec{q}_1^{n+1}$ is the velocity field after convection.

The function of this correction, as described in Section 2.3.1, is to convect the vorticity toward vortex center. Fig. 2.2 shows the functions used in this correction.

First, the vorticity field is computed at each cell center:

$$\vec{\omega} = \nabla \times \vec{q}_1^{n+1} \tag{2.32}$$

where

$$\vec{q}_1^{n+1} = u_1^{n+1}\vec{i} + v_1^{n+1}\vec{j} + w_1^{n+1}\vec{k}. \tag{2.33}$$

Then the vorticity components in each direction are:

$$\omega_x = \frac{\partial w_1^{n+1}}{\partial y} - \frac{\partial v_1^{n+1}}{\partial z}$$

$$\omega_y = \frac{\partial u_1^{n+1}}{\partial z} - \frac{\partial w_1^{n+1}}{\partial x}$$

$$\omega_z = \frac{\partial v_1^{n+1}}{\partial x} - \frac{\partial u_1^{n+1}}{\partial y}$$

Numerically, in a uniform 3-D Cartesian coordinate system, $\Delta x = \Delta y =$

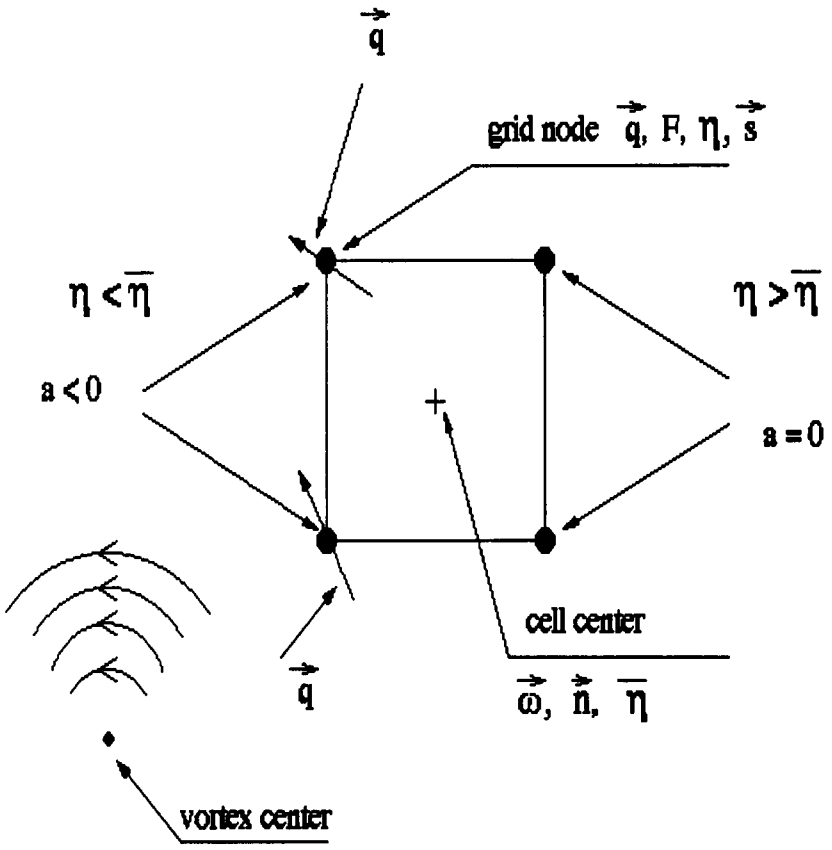


Figure 2.2: Vorticity Confinement Grid and Functions.

$\Delta z = 1$, these can be discretized as:

$$\begin{aligned}
\omega_{x_{i,j,k}} = \; & \tfrac{1}{4}(w^{n+1}_{1_{i,j+1,k}} - w^{n+1}_{1_{i,j,k}} + w^{n+1}_{1_{i+1,j+1,k}} - w^{n+1}_{1_{i+1,j,k}} + \\
& w^{n+1}_{1_{i,j+1,k+1}} - w^{n+1}_{1_{i,j,k+1}} + w^{n+1}_{1_{i+1,j+1,k+1}} - w^{n+1}_{1_{i+1,j,k+1}}) - \\
& \tfrac{1}{4}(v^{n+1}_{1_{i,j,k+1}} - v^{n+1}_{1_{i,j,k}} + v^{n+1}_{1_{i+1,j,k+1}} - v^{n+1}_{1_{i+1,j,k}} + \\
& v^{n+1}_{1_{i+1,j,k+1}} - v^{n+1}_{1_{i+1,j,k}} + v^{n+1}_{1_{i+1,j+1,k+1}} - v^{n+1}_{1_{i+1,j+1,k}})
\end{aligned}$$

$$\begin{aligned}
\omega_{y_{i,j,k}} = \; & \tfrac{1}{4}(u^{n+1}_{1_{i,j,k+1}} - u^{n+1}_{1_{i,j,k}} + u^{n+1}_{1_{i+1,j,k+1}} - u^{n+1}_{1_{i+1,j,k}} + \\
& u^{n+1}_{1_{i+1,j,k+1}} - u^{n+1}_{1_{i+1,j,k}} + u^{n+1}_{1_{i+1,j+1,k+1}} - u^{n+1}_{1_{i+1,j+1,k}}) - \\
& \tfrac{1}{4}(w^{n+1}_{1_{i+1,j,k}} - w^{n+1}_{1_{i,j,k}} + w^{n+1}_{1_{i+1,j+1,k}} - w^{n+1}_{1_{i,j+1,k}} + \\
& w^{n+1}_{1_{i+1,j,k+1}} - w^{n+1}_{1_{i,j,k+1}} + w^{n+1}_{1_{i+1,j+1,k+1}} - w^{n+1}_{1_{i,j+1,k+1}})
\end{aligned}$$

$$\begin{aligned}
\omega_{z_{i,j,k}} = \; & \tfrac{1}{4}(v^{n+1}_{1_{i+1,j,k}} - v^{n+1}_{1_{i,j,k}} + v^{n+1}_{1_{i+1,j+1,k}} - v^{n+1}_{1_{i,j+1,k}} + \\
& v^{n+1}_{1_{i+1,j,k+1}} - v^{n+1}_{1_{i,j,k+1}} + v^{n+1}_{1_{i+1,j+1,k+1}} - v^{n+1}_{1_{i,j+1,k+1}}) - \\
& \tfrac{1}{4}(u^{n+1}_{1_{i,j+1,k}} - u^{n+1}_{1_{i,j,k}} + u^{n+1}_{1_{i+1,j+1,k}} - u^{n+1}_{1_{i+1,j,k}} + \\
& u^{n+1}_{1_{i,j+1,k+1}} - u^{n+1}_{1_{i,j,k+1}} + u^{n+1}_{1_{i+1,j+1,k+1}} - u^{n+1}_{1_{i+1,j,k+1}}).
\end{aligned}$$

where ω_x, ω_y, ω_z are the components of vorticity in i, j, k direction respectively.

The vorticity magnitude is also calculated at each cell center by:

$$|\vec{\omega}| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \tag{2.34}$$

or, in discretization form,

$$|\vec{\omega}|_{i,j,k} = \sqrt{\omega^2_{x_{i,j,k}} + \omega^2_{y_{i,j,k}} + \omega^2_{z_{i,j,k}}}$$

Then, the scalar field η, which has a local minimum on the centroid of the vortical region, is computed at each grid node by equation 2.10:

$$\eta = -|\vec{\omega}|$$

$$\eta_{i,j,k} = -\tfrac{1}{8}(|\vec{\omega}|_{i,j,k} + |\vec{\omega}|_{i,j-1,k} + |\vec{\omega}|_{i,j,k-1} + |\vec{\omega}|_{i,j-1,k-1} + |\vec{\omega}|_{i-1,j,k} + |\vec{\omega}|_{i-1,j-1,k} + |\vec{\omega}|_{i-1,j,k-1} + |\vec{\omega}|_{i-1,j-1,k-1}).$$

The unit vector $\vec{n}$, which points away from the centroid of the vertical region, is computed at the cell center by equation 2.8:

$$\vec{n} = \nabla\eta/|\nabla\eta|$$

where

$$\nabla\eta = \frac{\partial\eta}{\partial x}\vec{i} + \frac{\partial\eta}{\partial y}\vec{j} + \frac{\partial\eta}{\partial z}\vec{k} \tag{2.35}$$

and

$$|\nabla\eta| = \sqrt{(\frac{\partial\eta}{\partial x})^2 + (\frac{\partial\eta}{\partial y})^2 + (\frac{\partial\eta}{\partial z})^2} \tag{2.36}$$

$$(\tfrac{\partial\eta}{\partial x})_{i,j,k} = \tfrac{1}{4}(\eta_{i+1,j,k} - \eta_{i,j,k} + \eta_{i+1,j+1,k} - \eta_{i,j+1,k} + \eta_{i+1,j,k+1} - \eta_{i,j,k+1} + \eta_{i+1,j+1,k+1} - \eta_{i,j+1,k+1})$$

$$(\tfrac{\partial\eta}{\partial y})_{i,j,k} = \tfrac{1}{4}(\eta_{i,j+1,k} - \eta_{i,j,k} + \eta_{i+1,j+1,k} - \eta_{i+1,j,k} + \eta_{i,j+1,k+1} - \eta_{i,j,k+1} + \eta_{i+1,j+1,k+1} - \eta_{i+1,j,k+1})$$

$$(\tfrac{\partial\eta}{\partial z})_{i,j,k} = \tfrac{1}{4}(\eta_{i,j,k+1} - \eta_{i,j,k} + \eta_{i+1,j,k+1} - \eta_{i+1,j,k} + \eta_{i,j+1,k+1} - \eta_{i+1,j,k} + \eta_{i+1,j+1,k+1} - \eta_{i+1,j+1,k})$$

so that the mesh values of $|\nabla\eta|$ are computed from:

$$(|\nabla\eta|)_{i,j,k} = \sqrt{(\frac{\partial\eta}{\partial x})^2_{i,j,k} + (\frac{\partial\eta}{\partial y})^2_{i,j,k} + (\frac{\partial\eta}{\partial z})^2_{i,j,k}}$$

Hence, the numerical evaluations of the unit vector components on the mesh are given by

$$n_{x_{i,j,k}} = \frac{(\frac{\partial \eta}{\partial x})_{i,j,k}}{(|\nabla \eta|)_{i,j,k} + \epsilon'}$$

$$n_{y_{i,j,k}} = \frac{(\frac{\partial \eta}{\partial y})_{i,j,k}}{(|\nabla \eta|)_{i,j,k} + \epsilon'}$$

$$n_{z_{i,j,k}} = \frac{(\frac{\partial \eta}{\partial z})_{i,j,k}}{(|\nabla \eta|)_{i,j,k} + \epsilon'}$$

where ϵ' is a small number for avoiding numerical overflow. By using equation 2.7, the confinement term, $\vec{s}$, is evaluated at the grid node.

The components of $\vec{s}$ are

$$s_{x_{i,j,k}} = n_{z_{i,j,k}} \omega_{y_{i,j,k}} - n_{y_{i,j,k}} \omega_{z_{i,j,k}}$$

$$s_{y_{i,j,k}} = n_{x_{i,j,k}} \omega_{z_{i,j,k}} - n_{z_{i,j,k}} \omega_{x_{i,j,k}}$$

$$s_{z_{i,j,k}} = n_{y_{i,j,k}} \omega_{x_{i,j,k}} - n_{x_{i,j,k}} \omega_{y_{i,j,k}}$$

so that

$$\vec{s}_{i,j,k} = (s_{x_{i,j,k}}, s_{y_{i,j,k}}, s_{z_{i,j,k}})$$

The value of the F function at the cell center is averaged from F at the neighbor grid nodes:

$$\begin{aligned} F_{C_{i,j,k}} = \ & \tfrac{1}{8}(F_{i,j,k} + F_{i,j+1,k} + F_{i,j,k+1} + F_{i,j+1,k+1} + \\ & F_{i+1,j,k} + F_{i+1,j+1,k} + F_{i+1,j,k+1} + F_{i+1,j+1,k+1}) \end{aligned}$$

For each cell center, if $F_C > 0$, the velocity correction that confines vorticity is calculated by the equation 2.16:

$$\delta \vec{q}_2^{n+1} = -\varepsilon \Delta t a_l \vec{n} \times \vec{\omega}$$

The average value of η at cell center is calculated by:

$$\begin{aligned}\bar{\eta}_{i,j,k} = \ & \tfrac{1}{8}(\eta_{i,j,k} + \eta_{i,j+1,k} + \eta_{i,j,k+1} + \eta_{i,j+1,k+1} + \\ & \eta_{i+1,j,k} + \eta_{i,j+1,k} + \eta_{i,j,k+1} + \eta_{i,j+1,k+1})\end{aligned} \tag{2.37}$$

and weighting factors are

$$\begin{aligned}a'_1 &= min(0, \eta_{i+1,j+1,k+1} - \bar{\eta}_{i,j,k}) \\ a'_2 &= min(0, \eta_{i+1,j+1,k} - \bar{\eta}_{i,j,k}) \\ a'_3 &= min(0, \eta_{i+1,j,k} - \bar{\eta}_{i,j,k}) \\ a'_4 &= min(0, \eta_{i+1,j,k+1} - \bar{\eta}_{i,j,k}) \\ a'_5 &= min(0, \eta_{i,j+1,k+1} - \bar{\eta}_{i,j,k}) \\ a'_6 &= min(0, \eta_{i,j+1,k} - \bar{\eta}_{i,j,k}) \\ a'_7 &= min(0, \eta_{i,j,k} - \bar{\eta}_{i,j,k}) \\ a'_8 &= min(0, \eta_{i,j,k+1} - \bar{\eta}_{i,j,k})\end{aligned}$$

so that with

$$a'_{sum} = \sum_{l=1}^{8} a'_l$$

the weighting factors are normalized to

$$a_l = \frac{a'_l}{a'_{sum} + \epsilon'} \qquad for\ l = 1, 8 \tag{2.38}$$

where, again, ϵ' is a small constant added to avoid overflow.

Then the velocity correction at this numerical step, for the grid nodes around each cell center, is:

$$\vec{q}^{\,n+1}_{2_{i+1,j+1,k+1}} = \vec{q}^{\,n+1}_{1_{i+1,j+1,k+1}} - \varepsilon \Delta t a_1 \vec{s}_{i+1,j+1,k+1}$$

$$
\begin{aligned}
\vec{q}^{n+1}_{2_{i+1,j+1,k}} &= \vec{q}^{n+1}_{1_{i+1,j+1,k}} - \varepsilon\Delta t a_2 \vec{s}_{i+1,j+1,k} \\
\vec{q}^{n+1}_{2_{i+1,j,k}} &= \vec{q}^{n+1}_{1_{i+1,j,k}} - \varepsilon\Delta t a_3 \vec{s}_{i+1,j,k} \\
\vec{q}^{n+1}_{2_{i+1,j,k+1}} &= \vec{q}^{n+1}_{1_{i+1,j,k+1}} - \varepsilon\Delta t a_4 \vec{s}_{i+1,j,k+1} \\
\vec{q}^{n+1}_{2_{i,j+1,k+1}} &= \vec{q}^{n+1}_{1_{i,j+1,k+1}} - \varepsilon\Delta t a_5 \vec{s}_{i,j+1,k+1} \\
\vec{q}^{n+1}_{2_{i,j+1,k}} &= \vec{q}^{n+1}_{1_{i,j+1,k}} - \varepsilon\Delta t a_6 \vec{s}_{i,j+1,k} \\
\vec{q}^{n+1}_{2_{i,j,k}} &= \vec{q}^{n+1}_{1_{i,j,k}} - \varepsilon\Delta t a_7 \vec{s}_{i,j,k} \\
\vec{q}^{n+1}_{2_{i,j,k+1}} &= \vec{q}^{n+1}_{1_{i,j,k+1}} - \varepsilon\Delta t a_8 \vec{s}_{i,j,k+1}
\end{aligned}
$$

where ε is the relaxation factor, Δt the computational time step.

Notice that this correction is explicitly performed for each grid cell, hence velocity at some grid nodes may have more than one correction in the same time step.

2.4.4 Body Surface Boundary Condition

The flow field is solved in the whole computational domain, including inside the body where $F < 0$. At each computational step, the velocity inside the body is corrected by using the F function confinement to convect the vorticity toward solid body surface.

First, the velocity is multiplied by a factor σ:

$$\vec{q}^{n+1'}_3 = \sigma \vec{q}^{n+1}_2 \tag{2.39}$$

where

$$\sigma = \begin{cases} \frac{1}{(1-F)^2} & for F < 0 \\ 1 & F \geq 0 \end{cases}$$

$\vec{q}_2^{n+1}$ is the velocity field after the vorticity confinement correction has been made. As the result, the velocity inside the solid body is gradually reduced.

Then, the correction is applied by:

$$\vec{q}_3^{n+1} = \vec{q}_3^{n+1'} + \delta\vec{q}_3^{n+1} \tag{2.40}$$

where

$$\delta\vec{q}_3^{n+1} = \varepsilon_F \vec{n}_F \times \vec{\omega}$$

$$\vec{n}_F = \frac{\nabla F}{|\nabla F|}$$

$$\varepsilon_F = \begin{cases} \varepsilon & for F \leq 0 \\ 0 & F > 0 \end{cases}$$

ε is the confinement parameter same as described in Section 2.4.3 and $\vec{\omega} = \nabla \times \vec{q}_3^{n+1'}$ the vorticity after the vorticity confinement step has been conducted.

The numerical computations in this correction step are similar to the vorticity confinement step described in Section 2.4.3.

Notice that this correction is only carried out inside the body surface, where $F \leq 0$. As such, the vorticity generated inside body surface is convected toward the body surface.

2.4.5 Mass Conservation

A potential is solved on the Cartesian grid so that the sum of the gradient of the potential and the convected velocity with corrections enforces mass conservation.

$$\vec{q}^{n+1} = \vec{q}_3^{n+1} + \nabla\phi^{n+1} \tag{2.41}$$

The potential, $\nabla\phi^{n+1}$, satisfies the Poisson equation

$$\nabla^2\phi^{n+1} = -\nabla\cdot\vec{q}_3^{n+1} \tag{2.42}$$

where

$$\nabla\cdot\vec{q}_3^{n+1} = \frac{\partial u_3^{n+1}}{\partial x} + \frac{\partial v_3^{n+1}}{\partial y} + \frac{\partial w_3^{n+1}}{\partial z} \tag{2.43}$$

The right-hand-side of the Poisson equation 2.42 is evaluated at the cell center:

$$\begin{aligned}(-\nabla\cdot\vec{q}_3^{n+1})_{i,j,k} = \; & -\tfrac{1}{4}(u^{n+1}_{3_{i+1,j,k}} - u^{n+1}_{3_{i,j,k}} + u^{n+1}_{3_{i+1,j+1,k}} - u^{n+1}_{3_{i,j+1,k}} + \\ & u^{n+1}_{3_{i+1,j,k+1}} - u^{n+1}_{3_{i,j,k+1}} + u^{n+1}_{3_{i+1,j+1,k+1}} - u^{n+1}_{3_{i,j+1,k+1}}) \\ & -\tfrac{1}{4}(v^{n+1}_{3_{i,j+1,k}} - v^{n+1}_{3_{i,j,k}} + v^{n+1}_{3_{i+1,j+1,k}} - v^{n+1}_{3_{i+1,j,k}} + \\ & v^{n+1}_{3_{i,j+1,k+1}} - v^{n+1}_{3_{i,j,k+1}} + v^{n+1}_{3_{i+1,j+1,k+1}} - v^{n+1}_{3_{i+1,j,k+1}}) \\ & -\tfrac{1}{4}(w^{n+1}_{3_{i,j,k+1}} - w^{n+1}_{3_{i,j,k}} + w^{n+1}_{3_{i+1,j,k+1}} - w^{n+1}_{3_{i+1,j,k}} + \\ & w^{n+1}_{3_{i+1,j,k+1}} - w^{n+1}_{3_{i+1,j,k}} + w^{n+1}_{3_{i+1,j+1,k+1}} - w^{n+1}_{3_{i+1,j+1,k}})\end{aligned} \tag{2.44}$$

Any existing Poisson solver can be used for this step. A Cartesian grid Poisson solver, consisting of FORTRAN subprograms HW3CRT and HWSCRT of "FISHPAK" [12], was used in the present study for 3-D and 2-D Poisson equation solutions respectively. The Neumann boundary condition, $\frac{\partial\phi}{\partial n} = 0$, was imposed on each boundary of computational domain.

2.4.6 Initial Flow Field and Far Field Boundary Conditions

Initially, the flow field is set to uniform flow in the computations of flow over a complex configuration .

Assume the freestream is in the i direction in 3-D Cartesian system. The velocity, $\vec{q} = (u, v, w)$, at time $t = 0$ is set to:

$$\vec{q}^0 = V_\infty\vec{i}$$

which numerically is, for each grid node in the computational domain:

$$u^0_{i,j,k} = V_\infty$$

$$v^0_{i,j,k} = 0$$

$$w^0_{i,j,k} = 0$$

where u, v, w are the velocity components in i, j, k direction, respectively.

The far field boundary condition is explicitly imposed after velocity convection and corrections, at every time step. At the upstream boundary, the velocity is set to the free stream velocity:

$$\vec{q}^n_{1,j,k} = \vec{q}^{n-1}_{1,j,k} = \vec{q}^0_{1,j,k} = V_\infty \vec{i}.$$

The velocity at other boundaries of computational domain are linearly extrapolated from the inner domain:

$$\begin{aligned}
\vec{q}^n_{imax,j,k} &= 2\vec{q}^n_{imax-1,j,k} - \vec{q}^n_{imax-2,j,k} \\
\vec{q}^n_{i,1,k} &= 2\vec{q}^n_{i,2,k} - \vec{q}^n_{i,3,k} \\
\vec{q}^n_{i,jmax,k} &= 2\vec{q}^n_{i,jmax-1,k} - \vec{q}^n_{i,jmax-2,k} \\
\vec{q}^n_{i,j,1} &= 2\vec{q}^n_{i,j,2} - \vec{q}^n_{i,j,3} \\
\vec{q}^n_{i,j,kmax} &= 2\vec{q}^n_{i,j,kmax-1} - \vec{q}^n_{i,j,kmax-2}
\end{aligned}$$

2.4.7 Pressure Computation

The pressure coefficient C_p is derived from the modified momentum equations 2.6 and the numerical procedures for solving that equation.

Starting with the inviscid modified momentum equations,

$$\partial_t \vec{q} = -(\vec{q} \cdot \nabla)\vec{q} + \nabla p/\rho + \varepsilon \vec{s} \tag{2.45}$$

and the numerical relationship of equation 2.41 where

$$\vec{q}^{n+1} = \vec{q}_3^{n+1} + \nabla \phi^{n+1} \tag{2.46}$$

and, where $\vec{q}_3^{n+1}$ is the velocity field after velocity covection, diffusion and correction, ie.

$$\vec{q}_3^{n+1} = \vec{q}^n - \Delta t((\vec{q}^n \cdot \nabla)\vec{q}^n + \varepsilon \vec{s}) \tag{2.47}$$

then equationo 2.46 can be written as:

$$\vec{q}^{n+1} = +\vec{q}^n - \Delta t((\vec{q}^n \cdot \nabla)\vec{q}^n + \varepsilon \vec{s}) + \nabla \phi^{n+1} \tag{2.48}$$

with equation 2.45 discretized as shown below,

$$\vec{q}^{n+1} = \vec{q}^n - \Delta t((\vec{q}^n \cdot \nabla)\vec{q}^n + \nabla p/\rho + \varepsilon \vec{s}) \tag{2.49}$$

then by combineing equations 2.48 and 2.49, the relation between the pressure, p, and potential, ϕ can be derived:

$$\nabla p = -\frac{\rho}{\Delta t} \nabla \phi^{n+1} \tag{2.50}$$

Then:

$$p = -\frac{\rho}{\Delta t}(\phi^{n+1}) + C \tag{2.51}$$

where C is a constant. To determine C, assuming $\phi = 0$ at far field boundary, then $C = p_\infty$. Hence

$$p - p_\infty = -\frac{\rho}{\Delta t}(\phi^{n+1}) \tag{2.52}$$

By definition:

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty V_\infty^2} \tag{2.53}$$

The C_p for present method has the form:

$$C_p = -\frac{2\phi}{\Delta t V_\infty^2} \tag{2.54}$$

Notice that, for incompressible flow, $\rho_\infty = \rho$.

2.5 Helicopter Rotor and Body Flow

A major application of the present method was to compute rotor/body flows. A rotor/body flow can be solved by using a moving embedding grid technique together with the present method. The rotor solution is first obtained from a separate rotor solver, the Transonic Unsteady Rotor Navier–Stokes(TURNS). This solution is then interpolated into the body grid which is the fixed Cartesian grid. Finally, the rotor/body solution is obtained by the present method on the body grid.

2.5.1 The Computational Grid

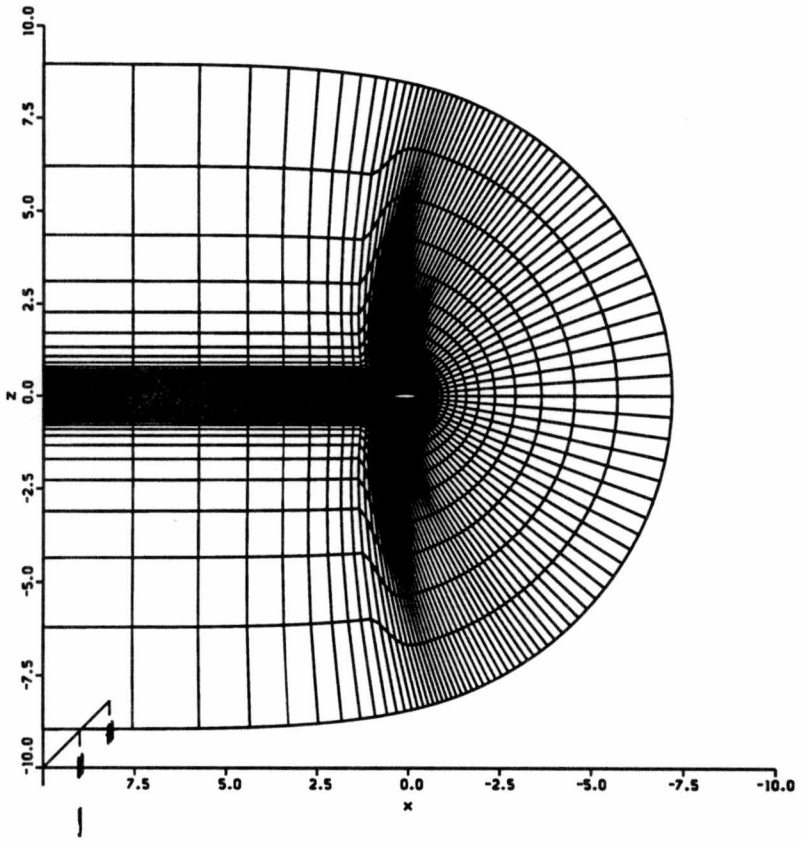
Two grid systems are used in rotor/body flow computation. A C-H type grid is used for the rotor solution and a Cartesian background grid embeds the entire flow field and body. The approach used for geometry decomposition and domain discretization was a combination of a background Cartesian grid and a topologically simple, body–fitted grid. The background Cartesian grid is a simple way to efficiently discretize the overall domain and resolve the dominant flow features away from the rotor. A body–fitted grid was used to resolve the geometric detail of the configuration and the viscous dominated regions of the surrounding flow field in rotor solution computations.

A C–H grid is used for rotor computation in the TURNS code. In order to obtain a better vorticity distribution, the C–H grid behind the blade tip has been modified. At each rotor cross section in the span wise direction, the grid was made uniform instead of being stretched. Fig. 2.3 shows the C–H grid that has been using for a fixed wing computation. Fig. 2.3(a) shows a cross section view of whole grid. Fig. 2.3(b) shows the grid near the wing surface.

2.5.2 Blade Solution

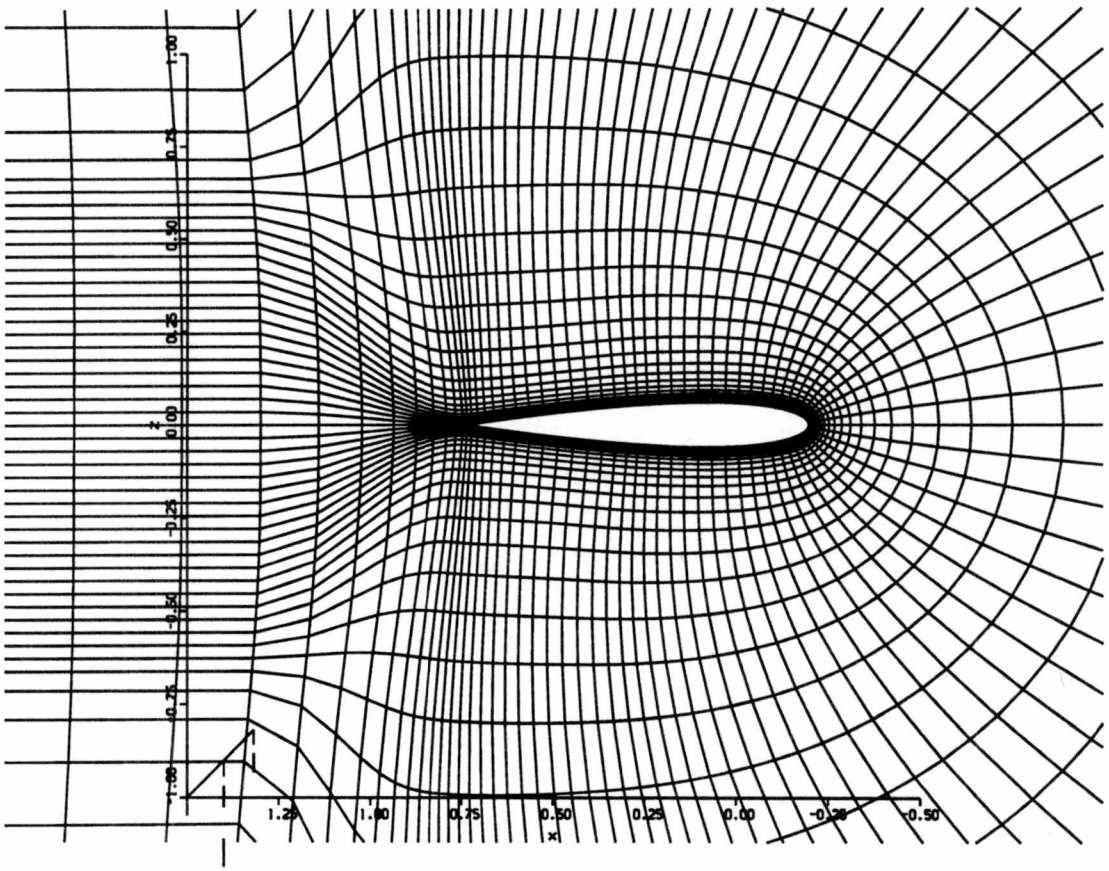
In the present study, the blade solution was supplied by Mr. John Bridgeman [1] using a compressible Euler/Navier–Stokes flow solver TURNS.

The governing differential equations solved in TURNS code are the thin layer Navier–Stokes equations. These can be written in conservation law form in



2.3(a)

Figure 2.3: C–H Grid Used for Wing Solution. 2.3(a): Cross Section near the Wing Tip. 2.3(b): Grid near the Wing Surface



2.3(b)

Figure 2.3 (continued)

a generalized coordinate system as follows:

$$\partial_\tau \hat{Q} + \partial_\xi \hat{E} + \partial_\eta \hat{F} + \partial_\zeta \hat{G} = \frac{1}{Re} \partial_\zeta \hat{S} \tag{2.55}$$

where, $\hat{Q} = J^{-1}[\rho, \rho u, \rho v, \rho w, e]^T$ is the column vector of the conserved variables, density, momentum components, and energy, scaled by the transformation Jacobian and, $\hat{E}$, $\hat{F}$,$\hat{G}$,$\hat{S}$ are the scaled inviscid and viscous flux vectors, Re the Reynolds number [13].

The vorticity contours of the blade solution are shown in Fig. 2.4. The isosurface corresponds to a vorticity level at 30% of the maximum tip vorticity magnitude. The figure shows that vorticity is highly concentrated in the rotor tip wake within two chord lengths down stream which is the region in which the velocity will be interpolated to the background Cartesian grid for rotor/body computations.

2.5.3 Interpolation

Flow around a combined rotor and body configuration was computed by a procedure that utilizes both a rotor solver code and a body solver code. First, a flow solution for a rotor blade at a given angle of attack was obtained from a solution provided by the rotor solver code, TURNS. The solution was then transferred onto the body grid, and the subsequent body flow was computed by the current method. There are basically two grid systems involved in the combined solution: the blade grid and the body grid. The blade grid is fine around and near the blade, where the body grid is relatively coarse in this domain. To be able to transfer the blade solution to the body grid, a simple yet effective interpolation

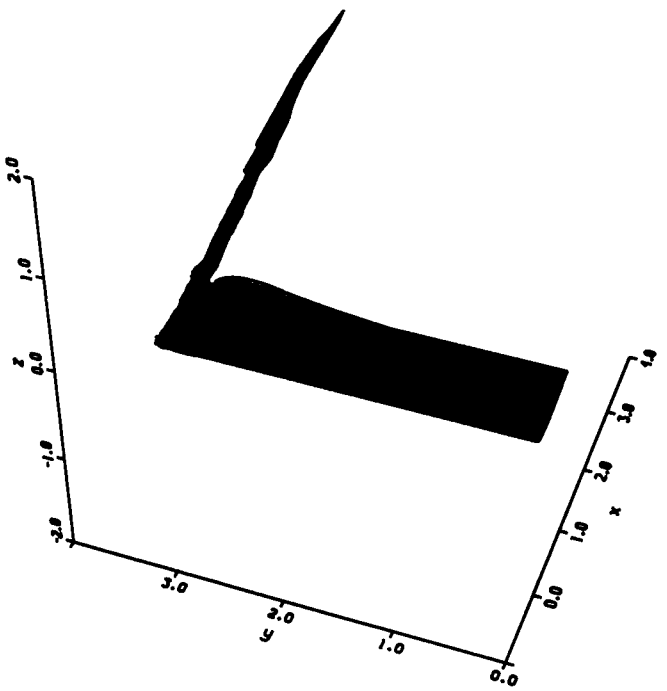


Figure 2.4: Isosurface of Vorticity Contour of Wing Solution

procedure was required for the present computations.

Within the body grid system, where the blade is located, two flow domains were defined. These two domains are centered on the blade and are designated as the *inner*, and the *transition* domains. The inner domain is surrounded by the transition domain. That is, the flow solution on the inner domain is identical to the blade solution. The flow solution on the transition domain is blended from both the blade and the body solutions. Outside the transition domain, the body solution is sought. It is noted that as the blade rotates above the body, the two domains also rotate within the body grid. On the points of the body grid located in the inner and transition domains, the velocity is first obtained by interpolation from the blade solution. The interpolation process is accomplished by a simple bi–linear interpolation scheme. In the transition domain, the interpolated velocity is then blended with the existing body solution. The transition domain velocity field is smoothly varied from the blade solution at the inner domain to the outside existing body solution. In the present computation, blending is accomplished by using a weighting function which is dependent on the distance from the inner domain. Notice that the inner domain is large enough to include the blade and its near flow field. In the computation of the body solution, the flow field in the inner domain does not directly enter into the computation procedure for the body flow.

Hence, for each grid point on body grid at each time step,

$$\vec{q}_{body}^{new} = \vec{q}_{blade}^{old} f + \vec{q}_{body}^{old}(1 - f) \tag{2.56}$$

with weighting function:

$$f = \begin{cases} 1 & \text{for } r < r1 \\ \cos((\pi/2)(r - r1)/(r2 - r1)) & \text{for } r1 \leq r \leq r2 \\ 0 & \text{for } r > r2 \end{cases}$$

where $r1$ is the radius of the inner domain, $r2$ is the radius of transition domain, r is the distance of body grid point to the center of blade.

2.5.4 Combined Solution

The combined solution is computed on the background Cartesian grid. The helicopter body is defined by the F function. The blade solution which was obtained by TURNS code is first read in, then interpolated on to the background grid. At selected time intervals, the blade solution is rotated through a defined number of degrees, then the convection step is performed. As described in Section 2.4, the correction using vorticity confinement is added to convected velocity field where $F > 0$, and the boundary condition on solid body surface is imposed by the correction based on the F function. Finally, a potential is solved to generate correction velocities that enforce mass conservation.

Chapter 3

COMPUTATIONAL RESULTS AND DISCUSSIONS

3.1 Vorticity Confinement

3.1.1 Non-Diffusing Axisymmetric Vortex

To demonstrate the Vorticity Confinement method, an axisymmetric vortex was treated and discretized on a 32×32 Cartesian grid. Numerical diffusion was simulated by averaging the Cartesian components of velocity at each of a sequence of computational steps using:

$$\vec{q}_{i,j}^{n+1} = \frac{1}{8}[\vec{q}_{i+1,j}^{n} + \vec{q}_{i-1,j}^{n} + \vec{q}_{i,j+1}^{n} + \vec{q}_{i,j-1}^{n} + 4\vec{q}_{i,j}^{n}] \tag{3.1}$$

Then, at each step, a correction was applied as described in Section 2.3.1. This was based on computed vorticity values, $\vec{\omega}^n = \vec{\nabla} \times \vec{q}^n$ and (constant) $\vec{n} = -\vec{r}/r$:

$$\vec{q}^{n+1} = \vec{q}^{n+1} - \varepsilon \vec{n} \times \vec{\omega}^n \tag{3.2}$$

To illustrate the effects of the correction, an initial vorticity distribution was set as constant within a certain (small) radius, and zero outside. This form, of course, will evolve due to the applied diffusion and corrections and reach a (different) steady state. In Fig. 3.1, initial vorticity contours of .4 and .8 of the maximum value are plotted, together with the grid lines. Vorticity is plotted in Fig. 3.2 along a horizontal line through the center. It is also plotted after 50 and 100 diffusion cycles, with no corrections. The vertical velocity is also plotted in

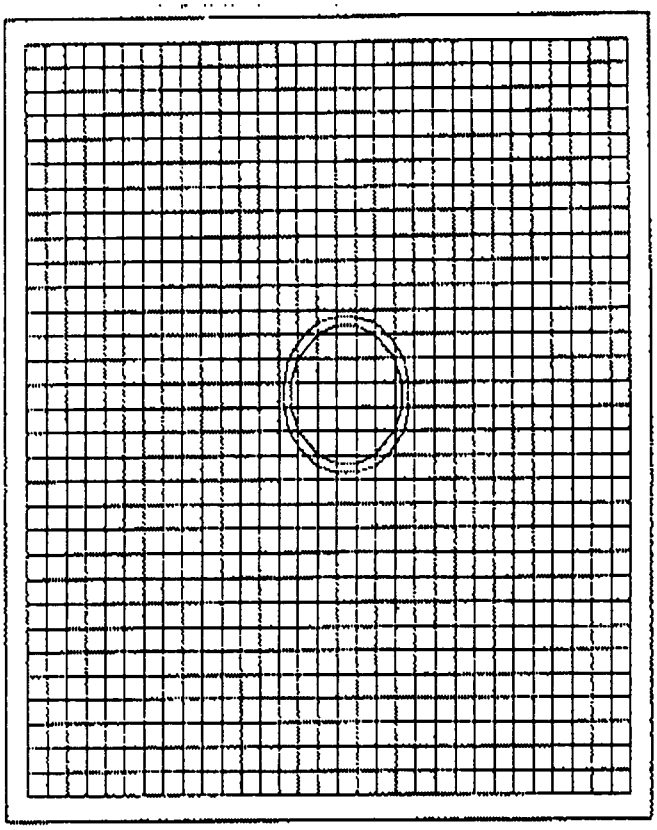


Figure 3.1: Initial Vorticity Contours.

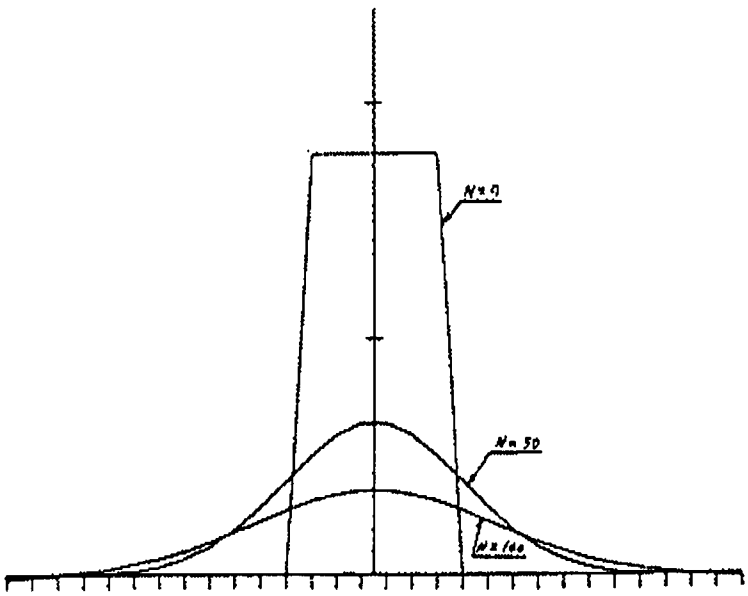


Figure 3.2: Vorticity Diffusion Without Confinement.

Fig. 3.3 along thesame line for the same stages. The strong effects of numerical diffusion can clearly be seen. This is expected, since the vortex is spread over only three cells. In Fig. 3.4, vorticity at 0, 50, 100, 150, and 200 cycles are plotted, but with one correction applied after each diffusion step. It can be seen that the vortex quickly attains a steady–state form, as predicted by the closed–form solution. The width of this form varies inversely with the constant, ε. Vertical velocity is plotted in Fig. 3.5 for the same stages. Since the velocity at a point away from the center does not decrease in time, it can be seen that total vorticity, or circulation, is conserved and does not decrease. It can be seen that with vorticity confinement, the steady state form represents a highly concentrated, non–diffusing vortex, even on a coarse grid.

3.1.2 Crow Instability Calculation

For this demonstration, the flow is initialized with a counter-rotating pair of line vortices. The distance between the vortices is b_o and the initial downward migration speed of the pair is V_o, which is approximately equal to $\Gamma_o/2\pi b_o$ where Γ_o is the initial circulation magnitude about one of the vortices. The approximate time required for the vortices to travel downward a distance b_o is thus b_o/V_o, which is denoted as T_o. The downward migration of the vortices has been computed and tracked in the present study, and the results are compared with observations. The data of Tomassian [14], Sarpkaya [15], Liu and Srnsky [16], and Delisi and Greene [17] are used for this comparison.

The initial parallelism of the line vortices is slightly perturbed in order

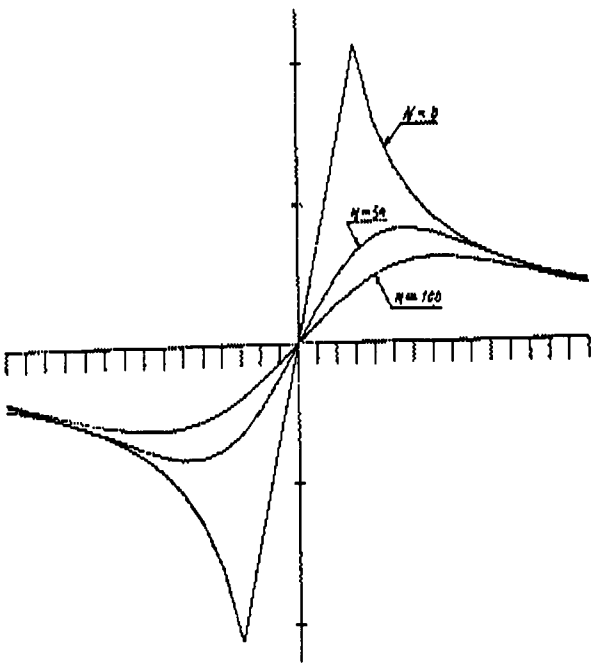


Figure 3.3: Velocity Diffusion Without Confinement.

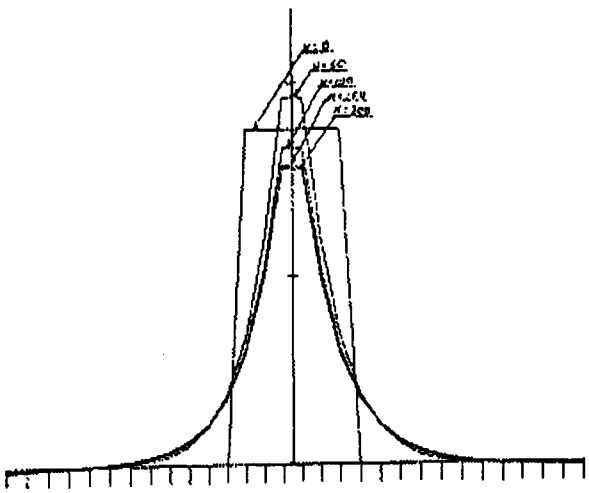


Figure 3.4: Vorticity Diffusion With Confinement.

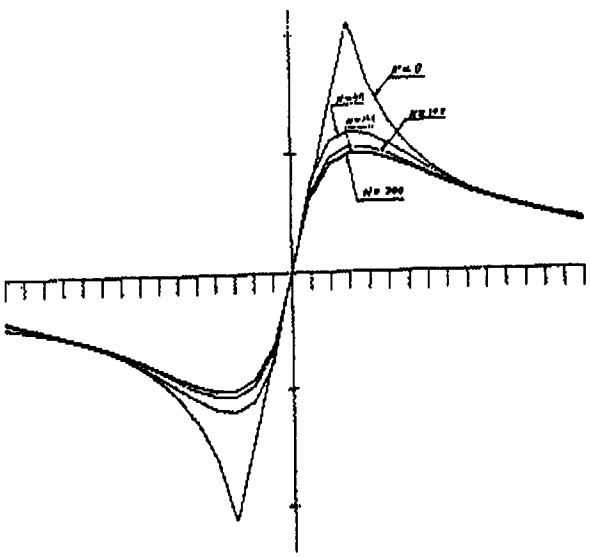


Figure 3.5: Velocity Diffusion With Confinement.

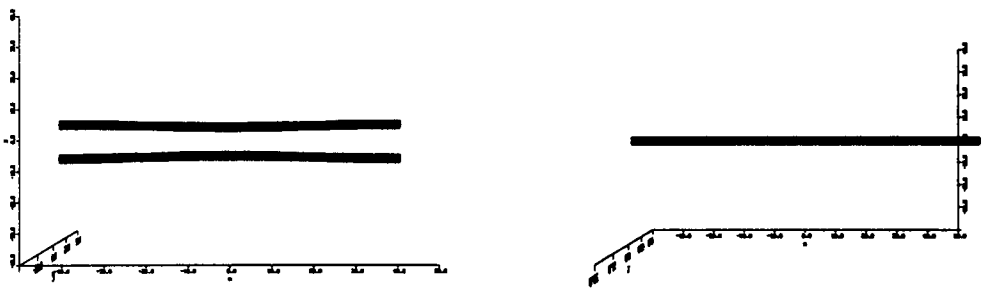
to promote the onset of linking instability (also known as Crow instability). The perturbation is sinusoidal and anti-symmetric with the peak-to-peak amplitude of the perturbation equal to one per cent of b_o. The perturbation wavelength is chosen to be approximately $8.6b_o$, which according to Crow [18] is the wavelength at which maximum perturbation growth should occur. Again the computed downward migration is to be compared with observations, the data reported by Delisi, et al [19]. According to this published data, the time at which vortex linking occurs is approximately $10T_o$.

A wavelength of $8b_o$ was used, which was slightly more convenient than the $8.6b_o$ value described above. The core diameter for this case was specified to be 0.25 b_o. The computational grid, as in the first case, was equally spaced with $96 \times 64 \times 64$ cells with 96 in the axial direction. Periodic conditions were used on all boundaries, so that as the pair descended through the lower boundary they reappeared at the upper one. It was determined that the perturbations resulting from the finite size of the computational domain were very small compared to the mutual interaction, and did not affect the development of the Crow instability.

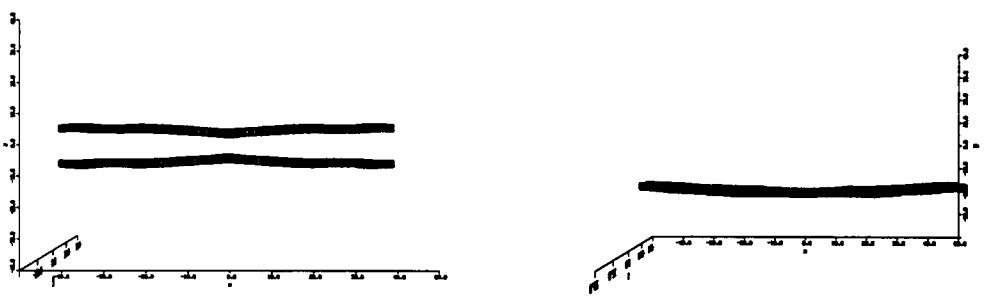
Results are presented in Figs. 3.6(a)–3.6(i) for an isosurface of constant vorticity magnitude corresponding to 30% of the maximum value of the initial state. Each figure shows a top view and a side view. Figure 3.6(e), which corresponds to $5.33T_o$, clearly shows the onset of linking. This agrees closely with the value of $5T_o$ obtained by Robins [19], using standard numerical methods, for a $0.20b_o$ core diameter and a $7.6b_o$ wavelength, and is within the range 5–9 b_o observed in the laboratory. (The wide range observed in laboratory experiments is

Top View

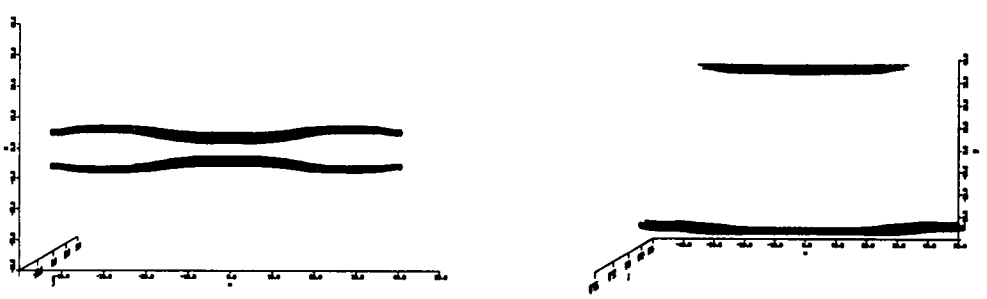
Side View



3.6(a)



3.6(b)

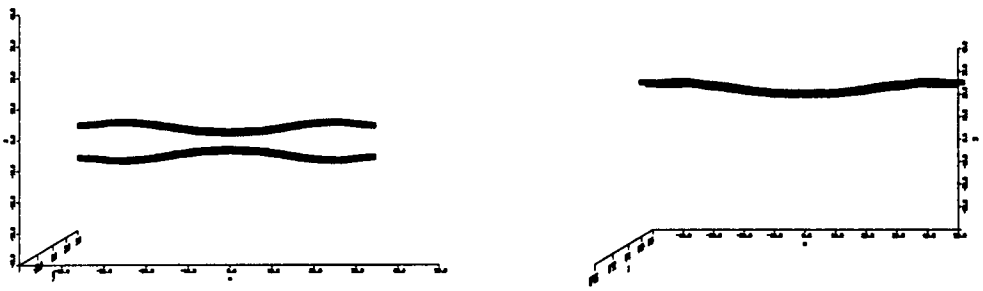


3.6(c)

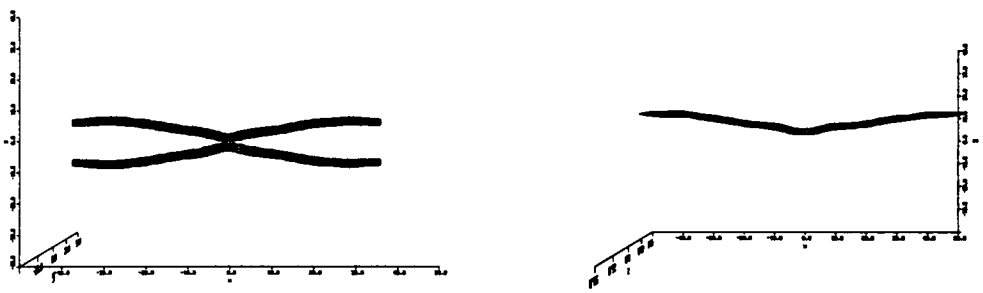
Figure 3.6: Isosurfaces of Constant Vorticity Showing Crow Instability Computation Using Vorticity Confinement.

Top View

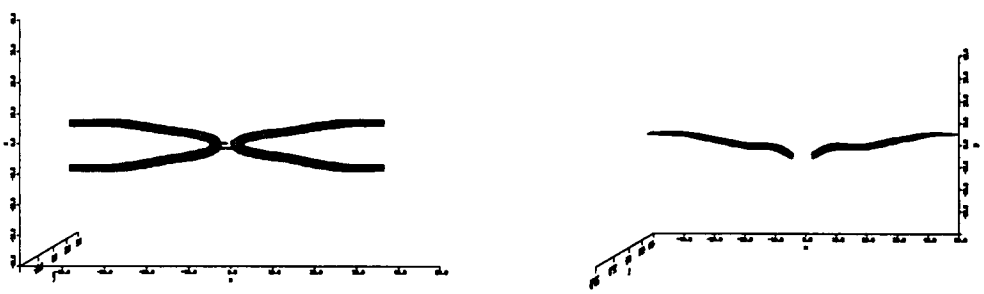
Side View



3.6(d)



3.6(e)

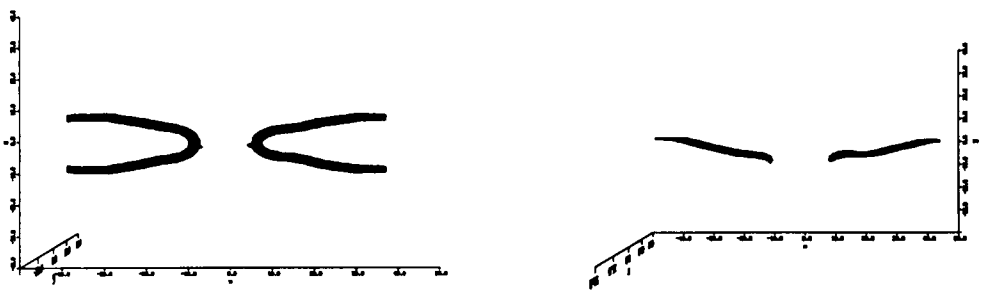


3.6(f)

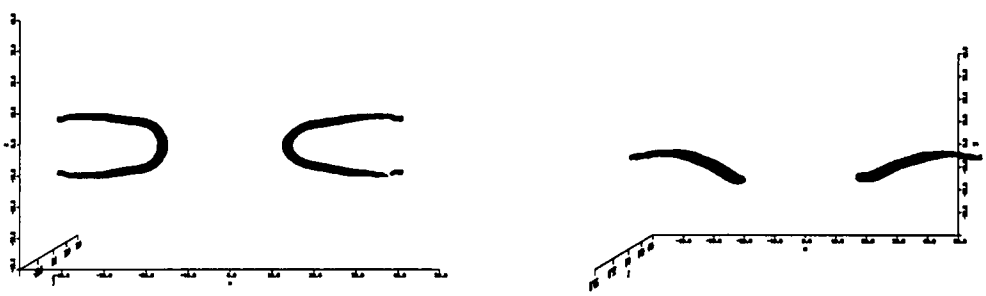
Figure 3.6 (continued)

Top View

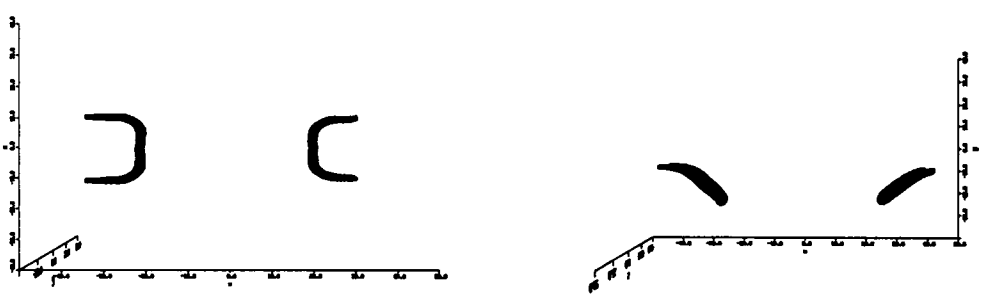
Side View



3.6(g)



3.6(h)



3.6(i)

Figure 3.6 (continued)

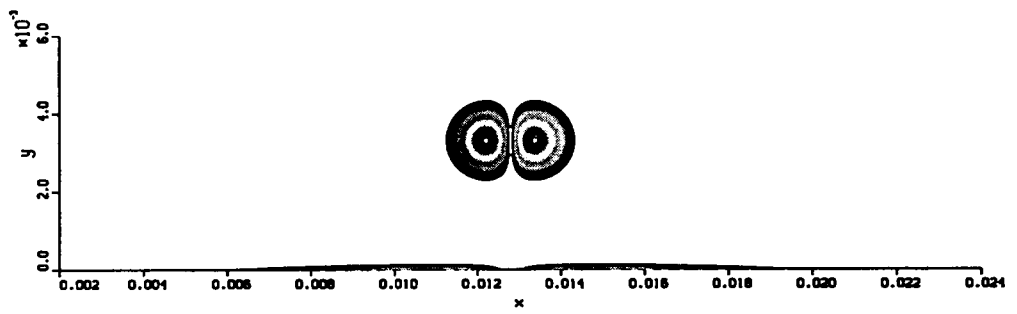
due to the vortices having more than one wavelength.) After linking, the vortices clearly form rings with no sign of numerical dissipation, as expected for confinement method. This corresponds to the physical case, where viscous dissipation effects do not cause any significant spreading over this time scale.

3.1.3 Computation of the Impingement of Concentrated Vortices on Surfaces

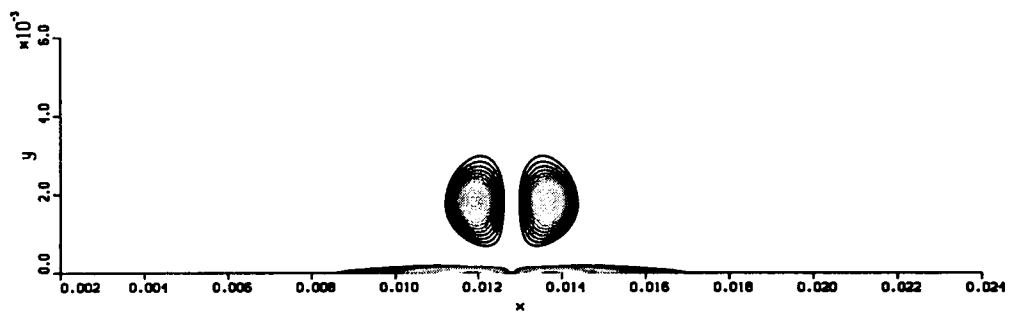
This case was used to study the interaction of descending vortices with the ground. Cases (for different vortex starting heights) described by Barker and Crow [20] and Delisi, et al [21], are simulated and compared. The numerical results are qualitatively compared with these observed data sets.

The first simulated case, corresponding to the 34 cm case of Delisi et. al. [21] involved a pair of opposite signed vortices with circulation of 250 cm^2/sec and $0.2b_o$ core radius, with an initial separation of 10 cm, and height above ground plane of 34 cm. The Reynolds number was 25000. The computational results and their comparison with experiment were all very good.

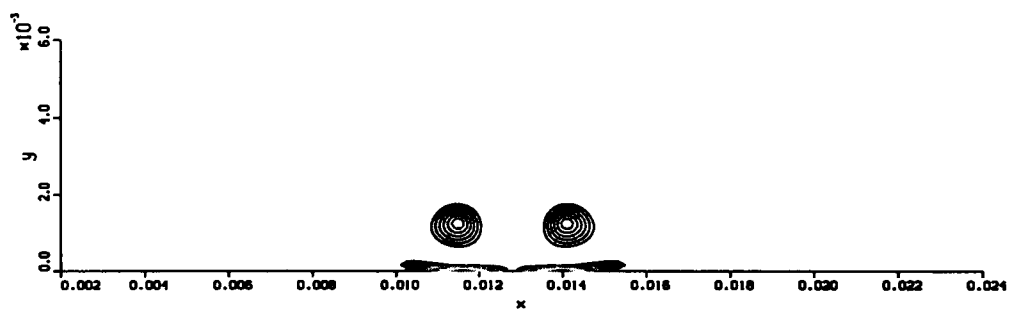
In Figs. 3.7(a)–3.7(f) vorticity contours are plotted for the numerical simulation. The ejection of surface boundary layer vorticity can clearly be seen. A visualization of the experimental results is shown in Fig. 3.8 (from Delisi, et al [21]). The trajectories of the centers of the computed vortex are shown in this figure with solid circles. It can be seen that agreement of the computed simulation with experiment is very good. Of particular importance is the distance of closest approach to the ground. The experimental value was 12 cm as was the computed value. (The asymmetry in the experimental data is apparently due to measurement



3.7(a)

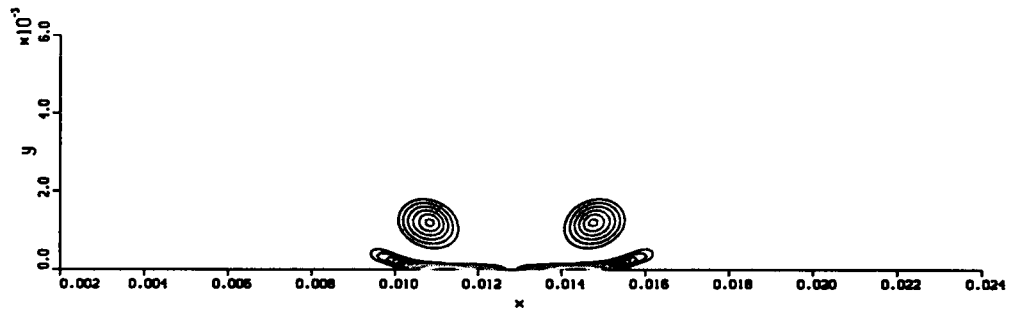


3.7(b)

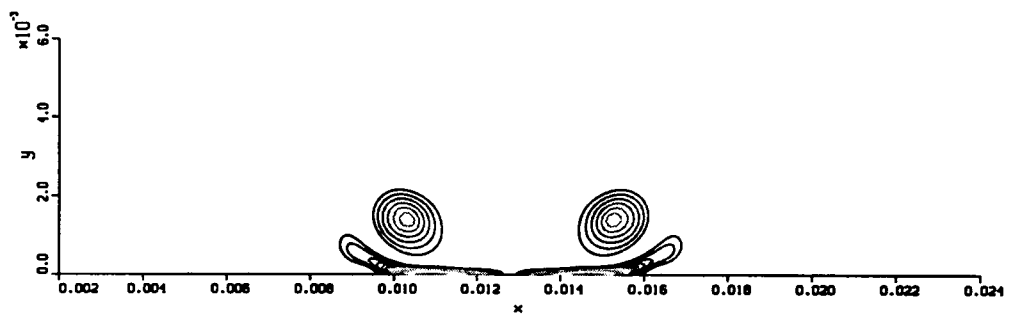


3.7(c)

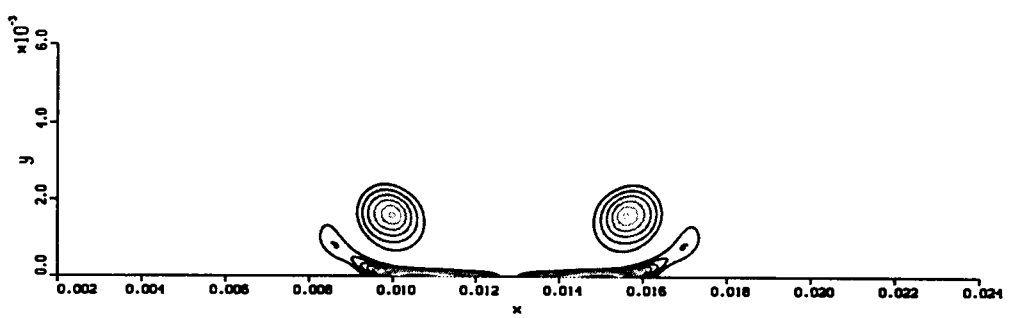
Figure 3.7: Vorticity Contours for Vortices with Core Radius of $0.20b_0$ Interacting with the Ground.



3.7(d)



3.7(e)



3.7(f)

Figure 3.7 (continued)

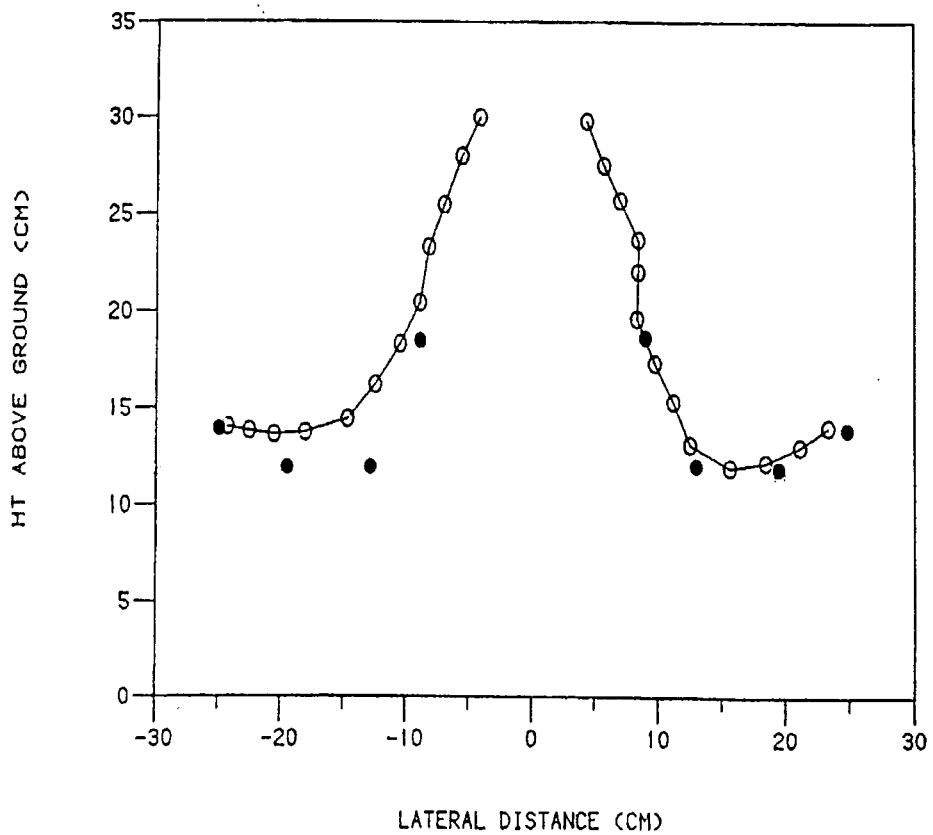


Figure 3.8: Laboratory Data of Delisi, et al , Open Circles, Compared with Predicted Locations of Vortices from Fig. 3.7, Plotted as Solid Circles.

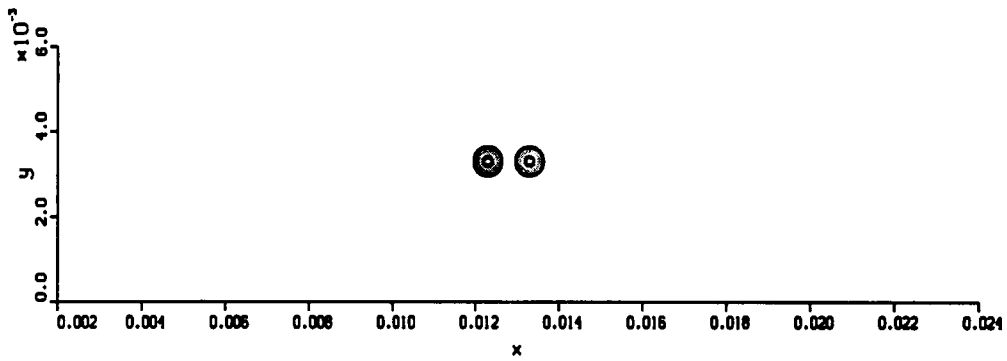
error. The computation was symmetric).

For comparison, the same case as above was simulated, but with core radius of $0.12b_o$. The vorticity contours in the simulation at the same times as above are depicted in Figs. 3.9(a)–3.9(f). It can be seen that separate, secondary (and tertiary) vortices are ejected from the boundary. These orbit around the primary vortices, altering their trajectories. This is seen in related experiments on normally impinging vortex rings [22], as shown in Fig. 3.10. The effects of the smaller core radius can clearly be seen.

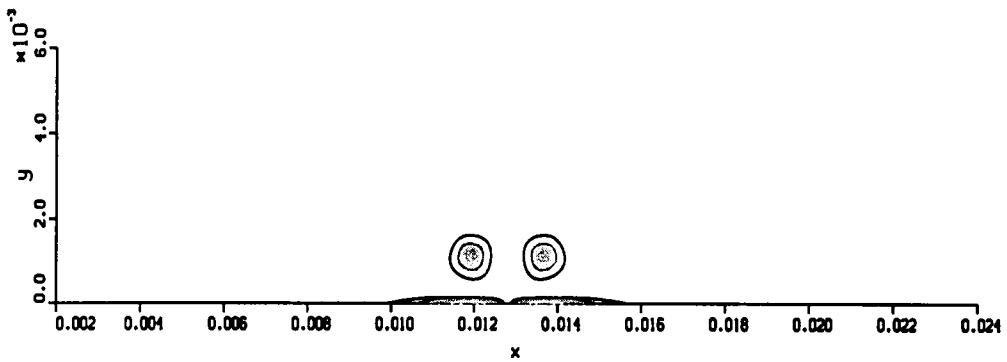
3.1.4 Effect of Shear on Vortex Filament Evolution

To study the effects of a shear layer on vortex convection, an unperturbed counter–rotating pair of line vortices were allowed to migrate in a flow field having sufficient shear to annihilate the vortex rotating in a sense opposite to that of the shear. Qualitative comparisons are made with the calculation of Bilanin, et al [23].

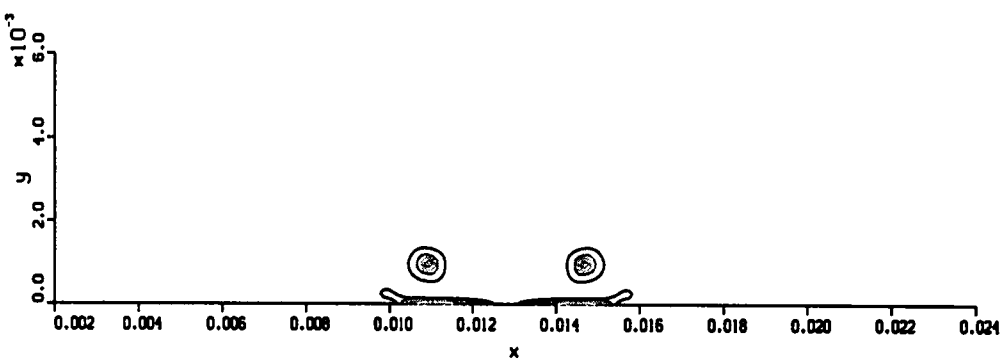
The computed sequence of vorticity contours in a 2–D plane are depicted in Figs. 3.11(a)–3.11(h). (Note that the vortices are swept by the current out the left side and back in the right side of the computational box since the side boundary conditions are periodic.) It can be seen that the vortex having the same rotational sense as the shear vorticity grows and the opposite one decays. The parameters of the simulation were: $\Gamma_o = 28.3cm^2/sec$, $b_o = 7.5cm$, core radius $0.2b_o$, and the shear, $dU/dz = 0.08sec^{-1}$. These were chosen to be the same as that of the experiment of Liu and Srnsky [16]. A sequence of photos from the experiment is presented in Fig. 3.12, which shows dye entrained by the vortices as they evolve.



3.9(a)

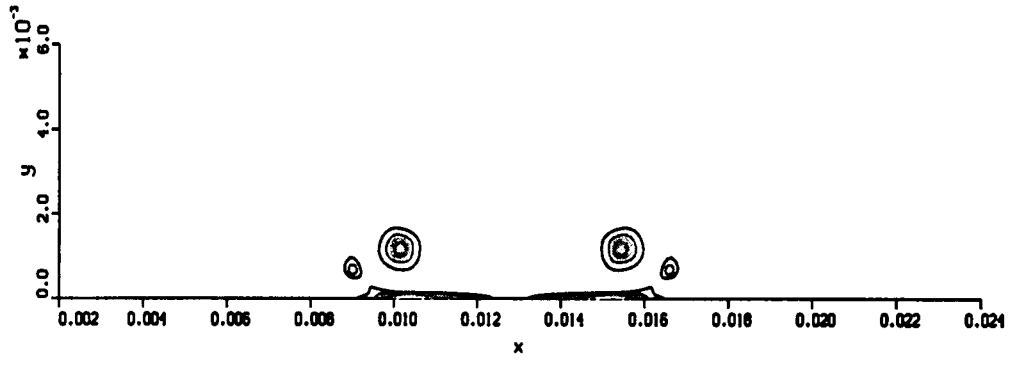


3.9(b)

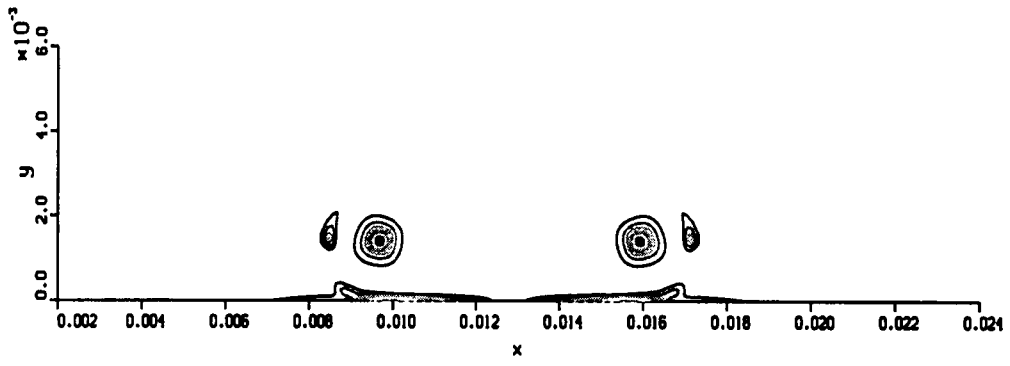


3.9(c)

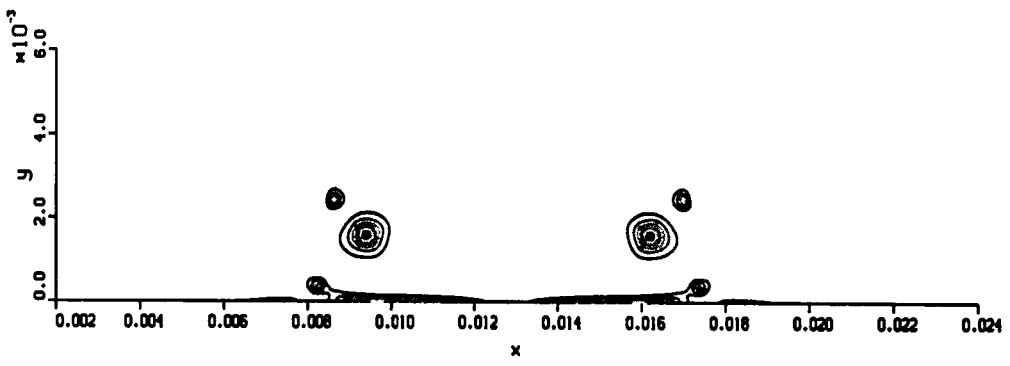
Figure 3.9: Vorticity Contours for Vortices with Core Radius of $0.120b_0$ Interacting with the Ground.



3.9(d)



3.9(e)



3.9(f)

Figure 3.9 (continued)

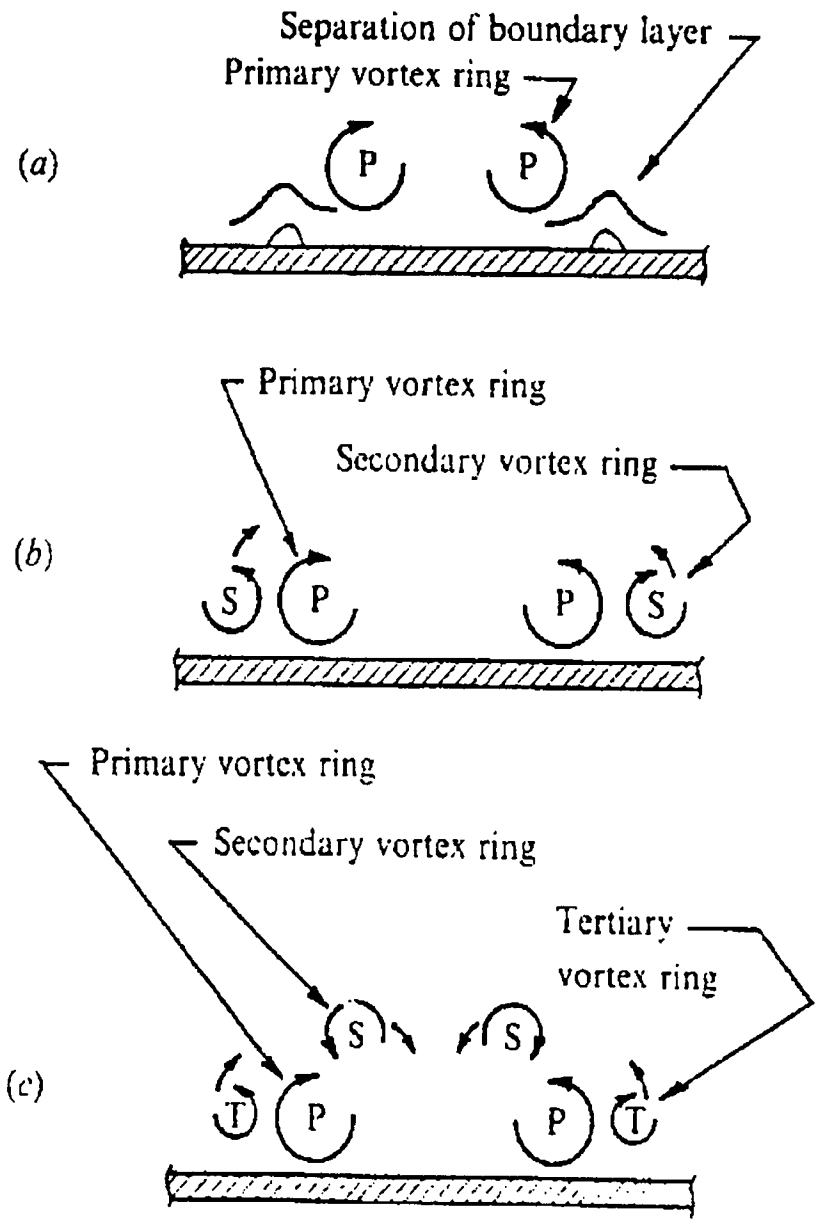
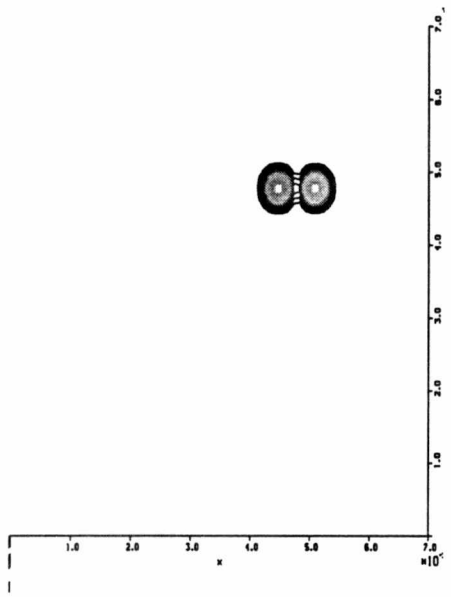
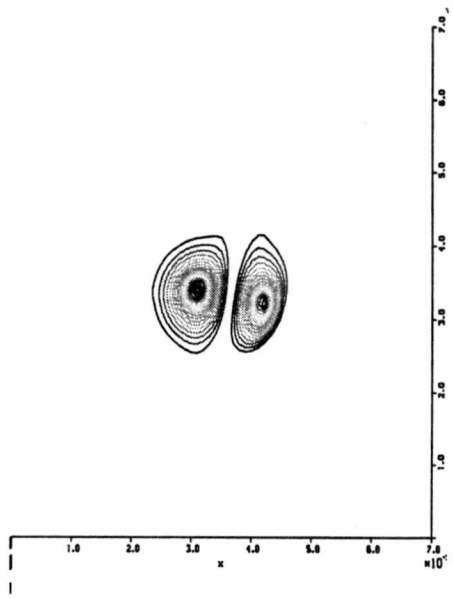


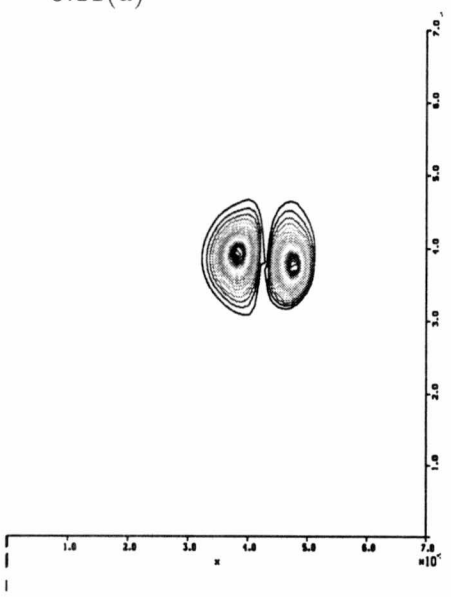
Figure 3.10: Schematic Diagram of Vortices Interacting with Ground, from Walker, Walker, et al .



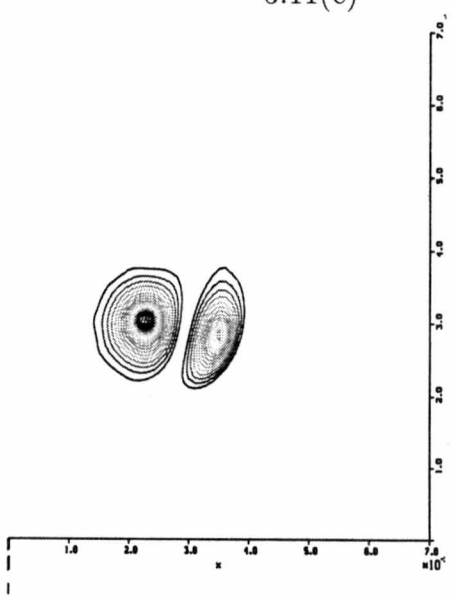
3.11(a)



3.11(c)

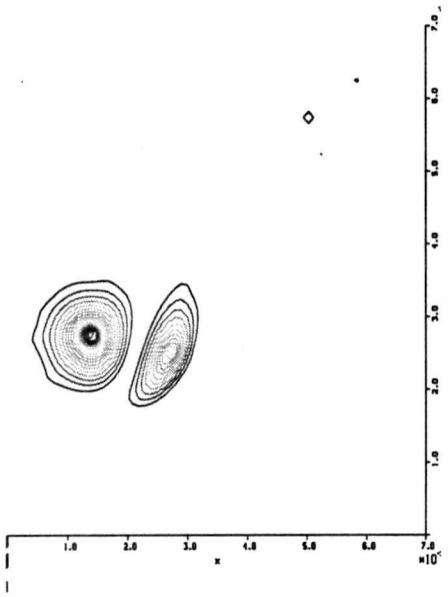


3.11(b)

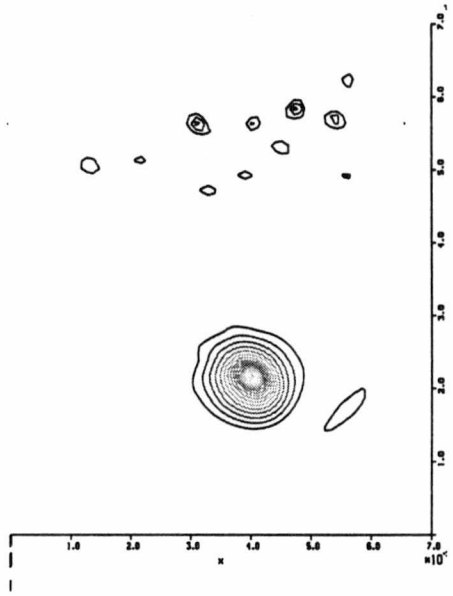


3.11(d)

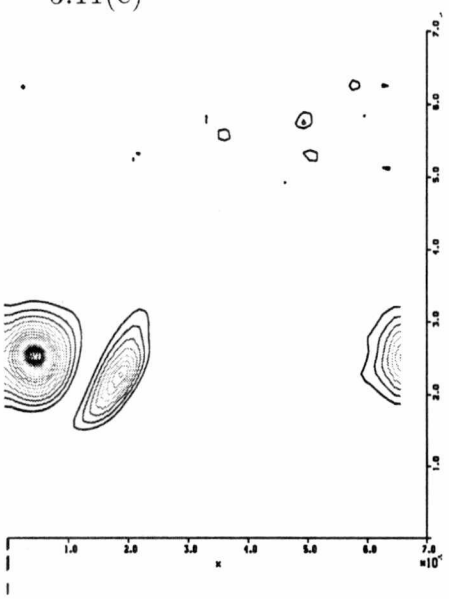
Figure 3.11: Vorticity Contours Showing the Effect of Shear on Evolving Vortices.



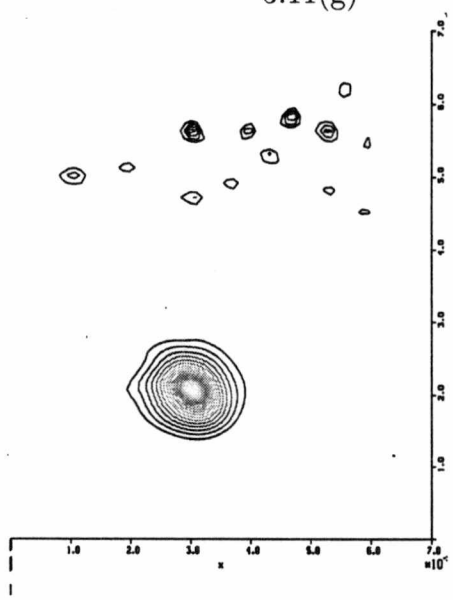
3.11(e)



3.11(g)



3.11(f)



3.11(h)

Figure 3.11 (continued)

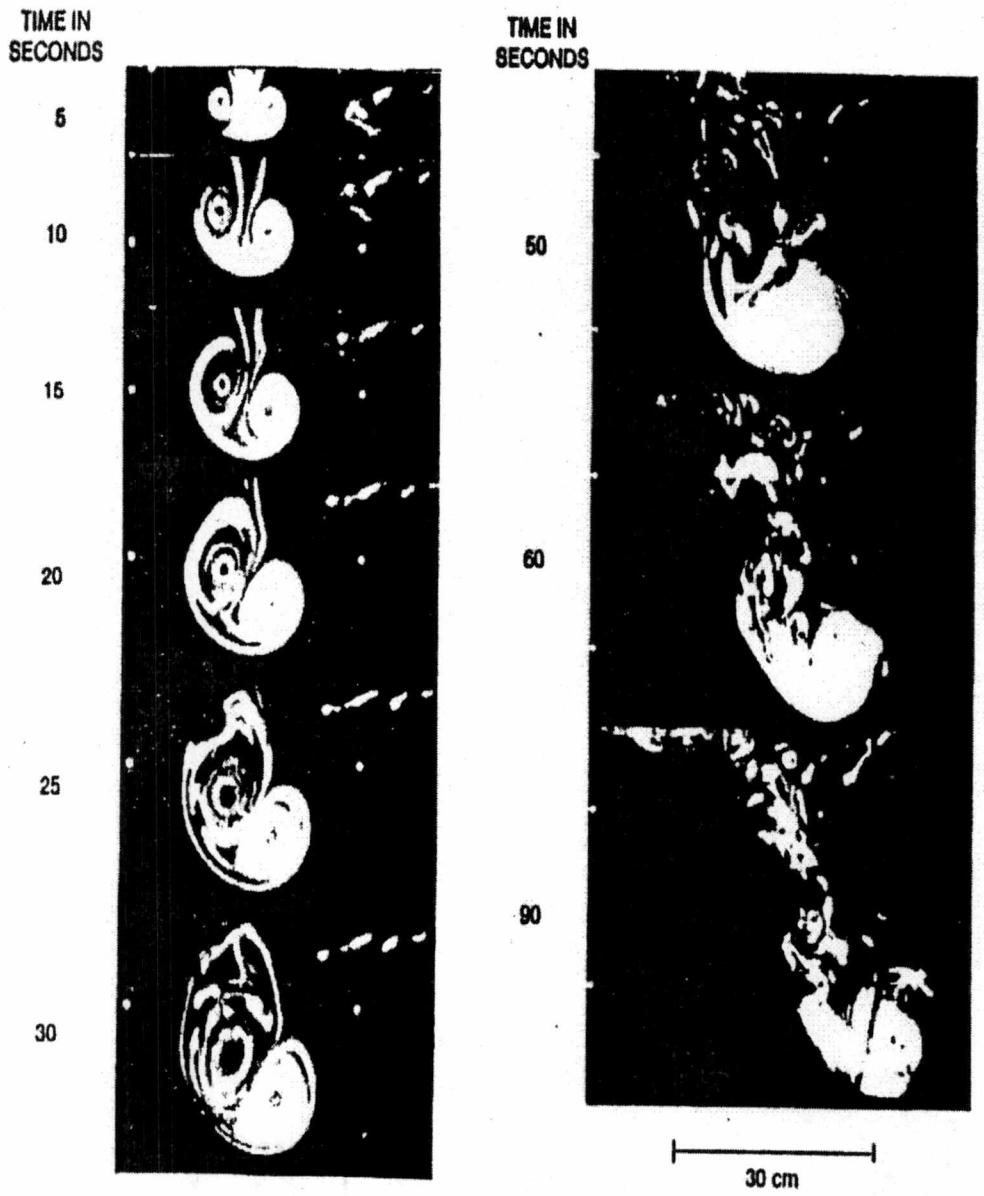


Figure 3.12: Laboratory Visualization of Vortices in Shear from Liu and Srnsky.

Qualitative agreement between the computation and the flow visualization can be seen. Note that the rotational sense of the shear in the calculation is opposite to that in the experiment. In addition, the results of Bilanin's calculation, referred to above, have been compared and are in close agreement with this computation. A 3–D isovorticity contour plot of the computation taken at the same time as the data of Fig. 3.11(e) is shown in Fig. 3.13.

3.2 Cartesian Method

3.2.1 Flow Over the Circular Cylinder

As a basic 2–D test case, the flow over a circular cylinder has been computed. The computational grid was 192×64. A circular cylinder with radius equal to the spacing between 16 grid cells was located at $i = 33$. Figs. 3.14– 3.16 show 3 different computed cases made with different values of the confinement parameter, which simulates different Reynolds numbers. The vortices shed from the cylinder surface form a Kármán Vortex Street. Fig. 3.17 is a schematic diagram of flow regimes for a cylinder by Panton [24](page 388). In a qualitative comparison, the results of a computational case with $\varepsilon = 0$ are similar to case (c) in the diagram, which is for $4 < Re < 40$. The computational case with $\varepsilon = 0.1$ is similar to case (d) in the diagram, which is for $40 < Re < 60 - 100$. Finally, the computational case with $\varepsilon = 0.3$ is similar to case (e) in the diagram, which is for $60 - 100 < Re < 200$.

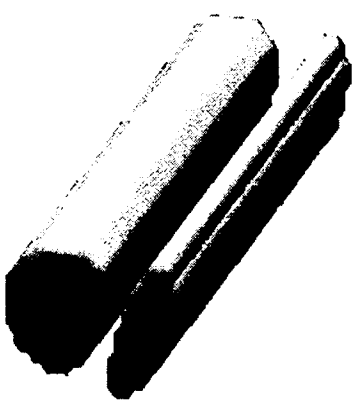
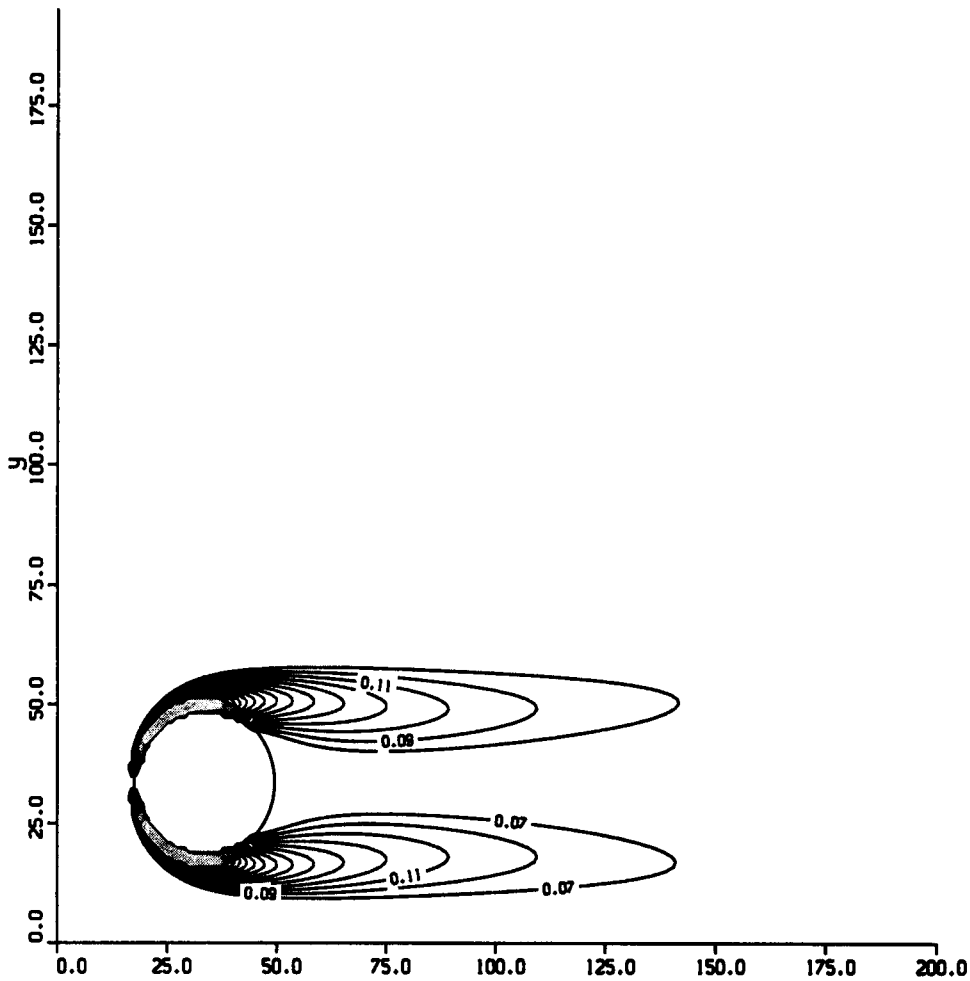
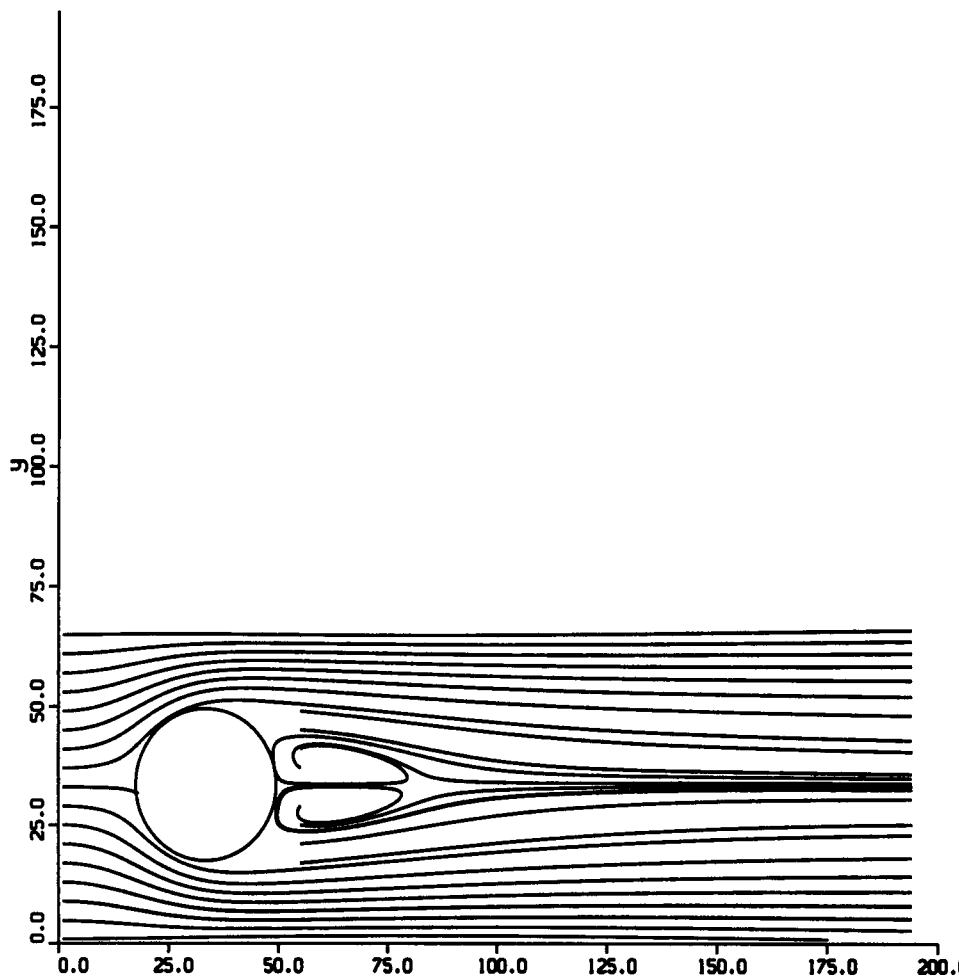


Figure 3.13: 3–D View of Vortices in Shear Corresponding to Fig. 3.11(e).



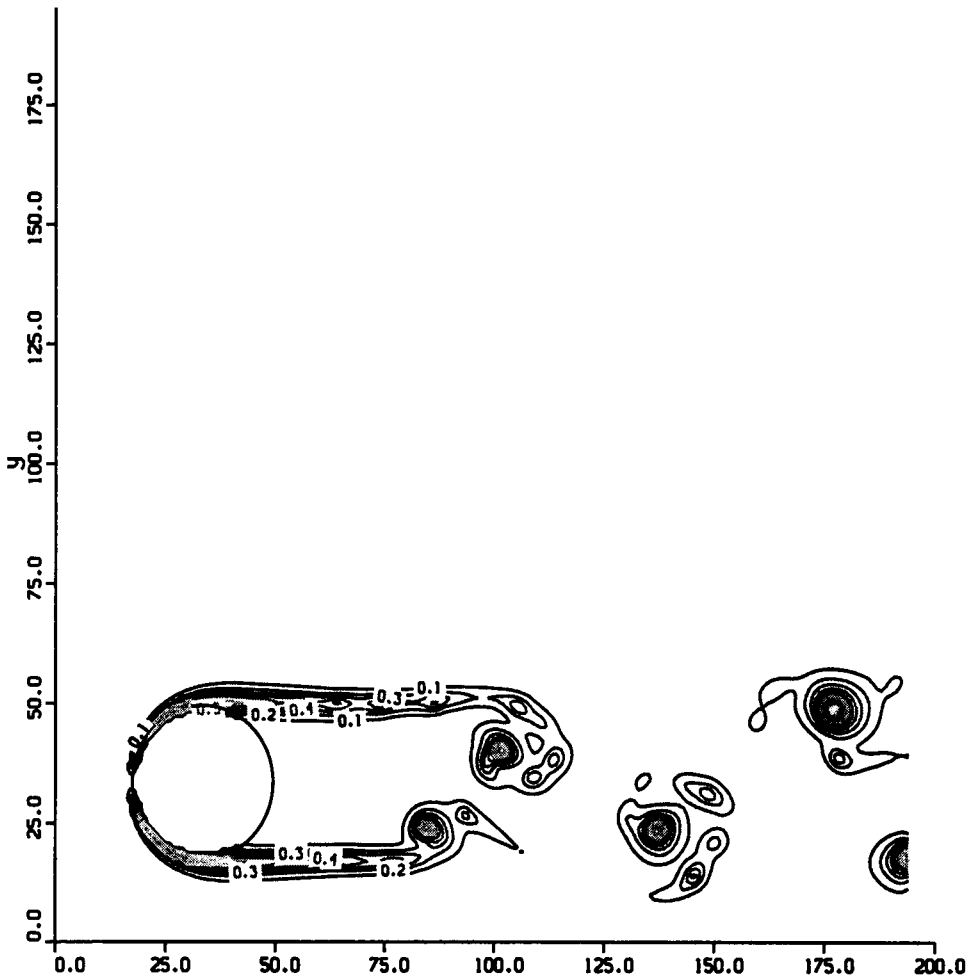
3.14(a)

Figure 3.14: (a) Vorticity Contour and (b) Stream Lines for Flow over Cylinder ($\varepsilon = 0$).



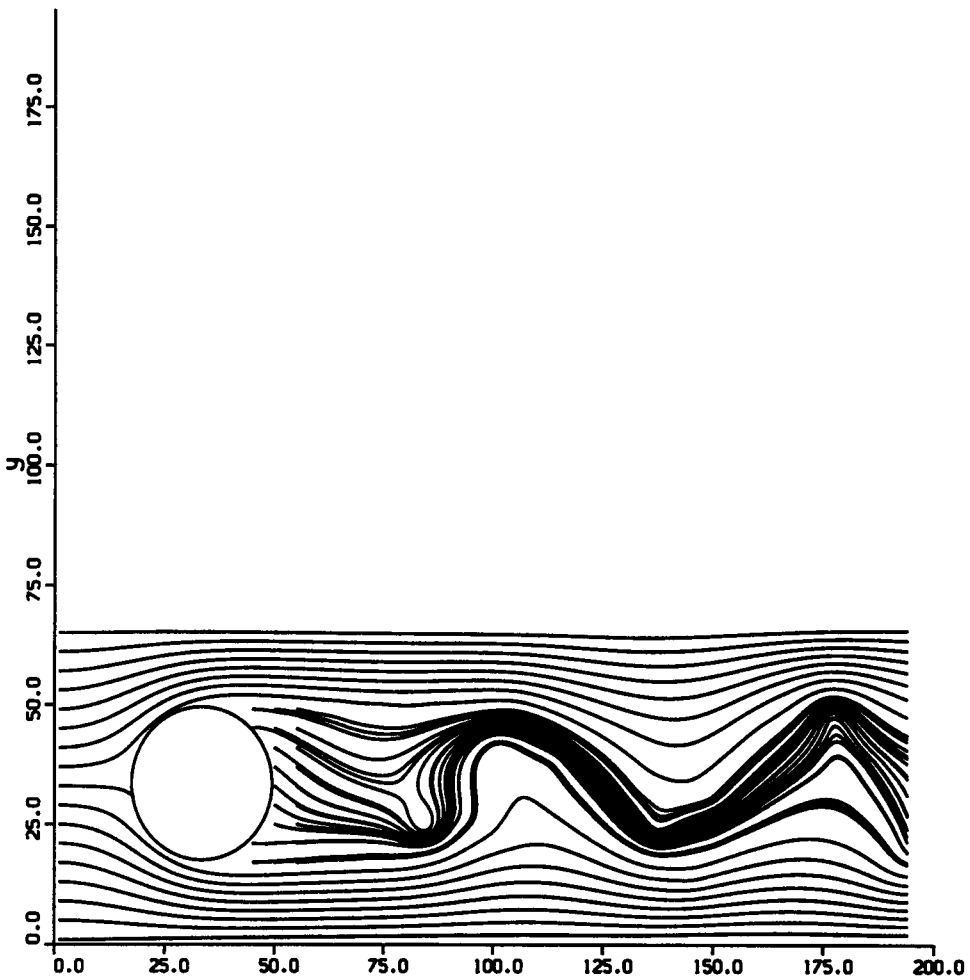
3.14(b)

Figure 3.14 (continued)



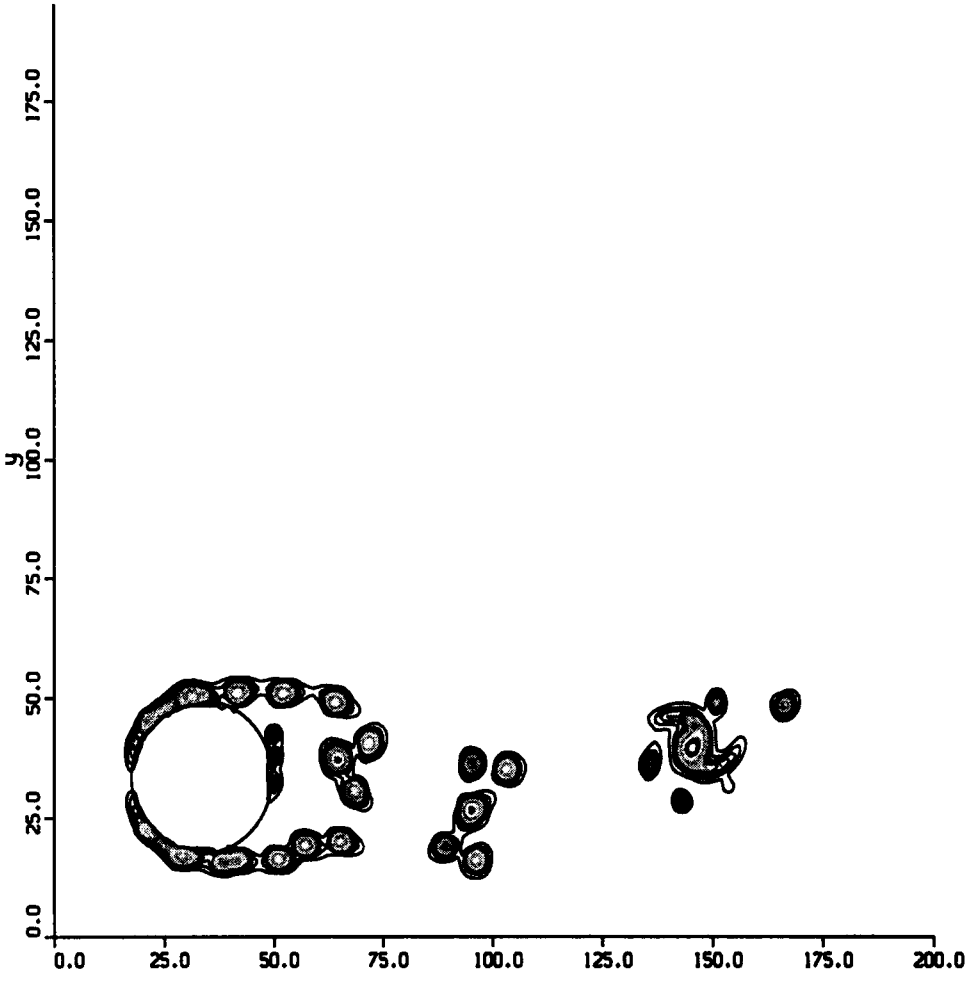
3.15(a)

Figure 3.15: (a) Vorticity Contour and (b) Stream Lines for Flow over Cylinder ($\varepsilon = 0.1$).



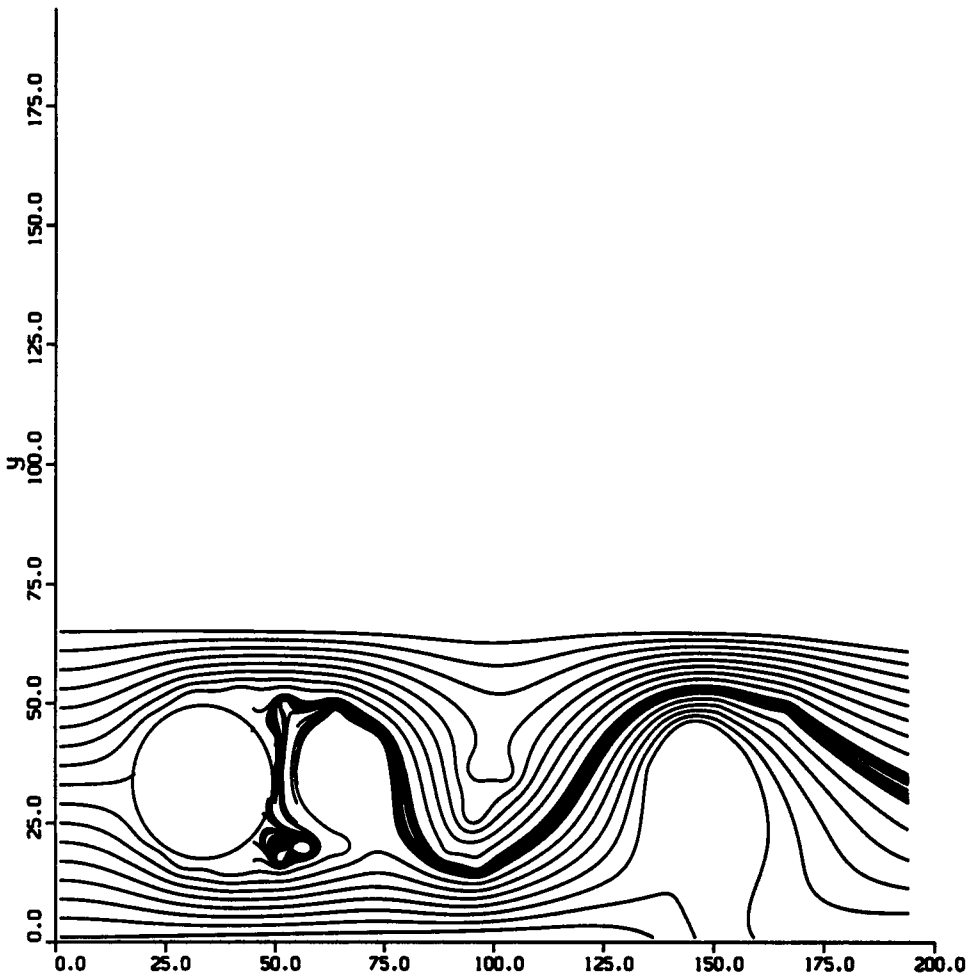
3.15(b)

Figure 3.15 (continued)



3.16(a)

Figure 3.16: (a) Vorticity Contour and (b) Stream Lines for Flow over Cylinder ($\varepsilon = 0.3$).



3.16(b)

Figure 3.16 (continued)

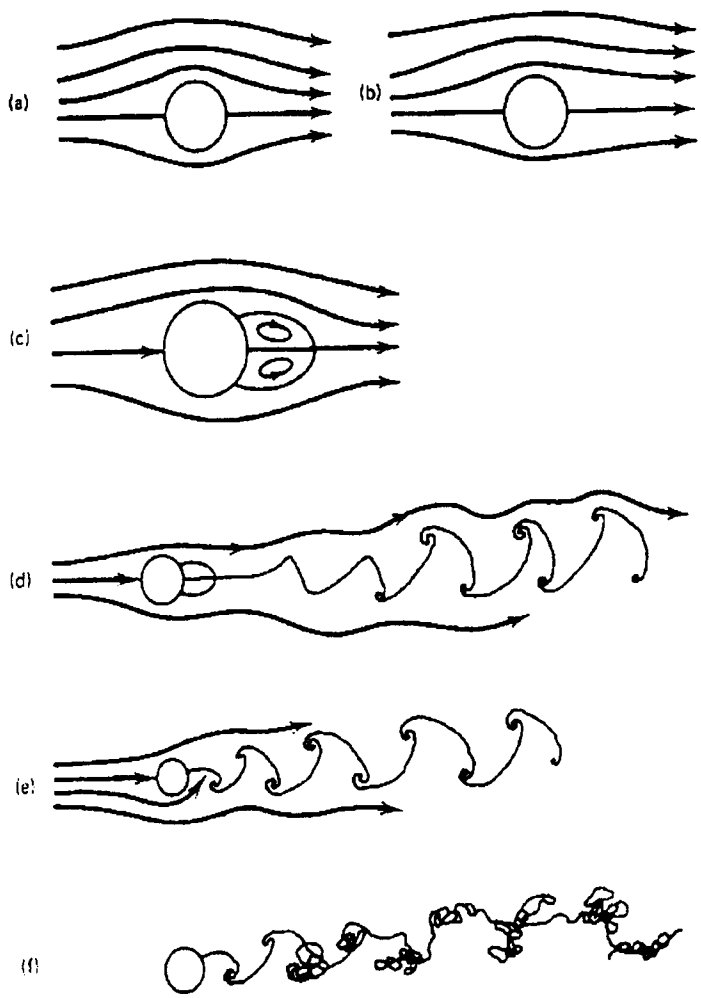


Figure 3.17: Flow Regimes for a Cylinder:

(a) $Re = 0$, Symmetrical; (b) $0 < Re < 4$; (c) $4 < Re < 40$, Attached Vortices; (d) $40 < Re < 60 - 100$, Kármán Vortex Street; (e) $60 - 100 < Re < 200$, Alternate Shedding; (f) $200 < Re < 400$, Vortices Unstable to Spanwise Bending.

3.2.2 Flow Over an Automobile

The flow over a generic automobile has been considered for the first 3-D computational case. The computation was carried out on a Cartesian grid with dimension of $128 \times 64 \times 64$, indexes i, j, and k, corresponding to x, y, and z directions, respectively. The surfaces of the automobile were generated by orienting two rectangular boxes. The F function is first set as shown in Fig. 3.18:

$$F^0 = \begin{cases} -1 & \text{inside the boxes} \\ 1 & \text{elsewhere} \end{cases}$$

To obtain the round shape of the body surface, the F function was smoothed by averaging the F values at the neighborhood points for several cycles. A simple iteration method was used for smoothing the F function:

$$F^1_{i,j,k} = \omega F^0_{i,j,k} + \frac{(1-\omega)}{6}(F^0_{i+1,j,k} + F^0_{i-1,j,k} + F^0_{i,j+1,k} + F^0_{i,j-1,k} + F^0_{i,j,k+1} + F^0_{i,j,k-1})$$

where ω is a relaxation factor.

The F function then was modified by convection in the x direction to simulate the car shape. The convecting velocity, for each grid node, was set to

$$\vec{q}_{c_{i,j,k}} = cz^2_{i,j,k}(1 - x_{i,j,k})\vec{i}$$

where c is a constant.

The one step Euler backward scheme, which was described in Section 2.4.2, was used for convecting the F function with velocity $\vec{q}_c$. Notice that the convection is only performed in x, or i, direction. First the location of the flow particle is found by:

$$x_{i,j,k_{ima}} = x_{i,j,k} - u_{c_{i,j,k}}$$

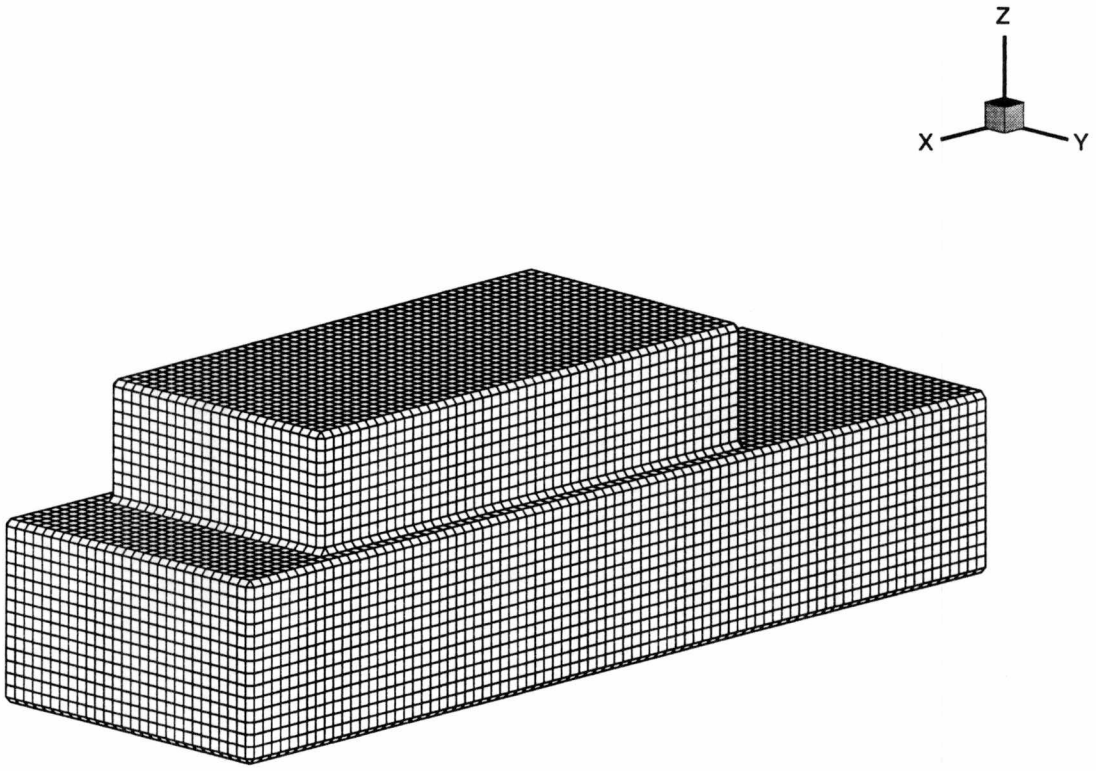


Figure 3.18: Initial F Function for Generic Car.

where u_c was the convecting velocity ($\vec{q}_c$) component in x direction. Then, the F function was obtained by the following interpolation steps:

$$i_{ima} = \frac{x_{i,j,k_{ima}}}{dx} + 1$$

$$r_x = x_{i,j,k_{ima}} - (i_{ima} - 1)dx$$

$$F^{n+1}_{i,j,k} = (\frac{r_x}{dx})F^n_{i_{ima},j,k} + (1 - \frac{r_x}{dx})F^n_{i_{ima}+1,j,k}$$

where $dx = 1$ for the uniform Cartesian grid.

The final F function for computing flow over the generic automobile is shown in Fig. 3.19.

The flow field was initially set to free stream velocity $V_0\vec{i}$ in the entire computational domain. A convergent solution was obtained in 80 time steps. Fig. 3.20 shows the vorticity contour at the mid plane of the generic automobile. The concentrated vortices generated from body surface can clearly be seen. Fig. 3.21 shows the streamlines along the car mid plane. Figs. 3.22– 3.23 show the velocity vectors near the body. The flow separates from the car surfaces as expected in the physical phenomena. The pressure distribution on the car surface is shown in Fig. 3.24. High pressure areas occurred at front of the car and the window, while the low pressure areas were at the back of the car.

The flow over a real automobile body was also computed in the present study. The geometry data, which contains 36864 panels and 37992 node points for a half body, is that of a Passat Limousine B3, which was provided by Volkswagen Automobile Company of Germany. The F function was generated by the numerical method (c) described in Section 2.4.1. The geometry configuration with surface

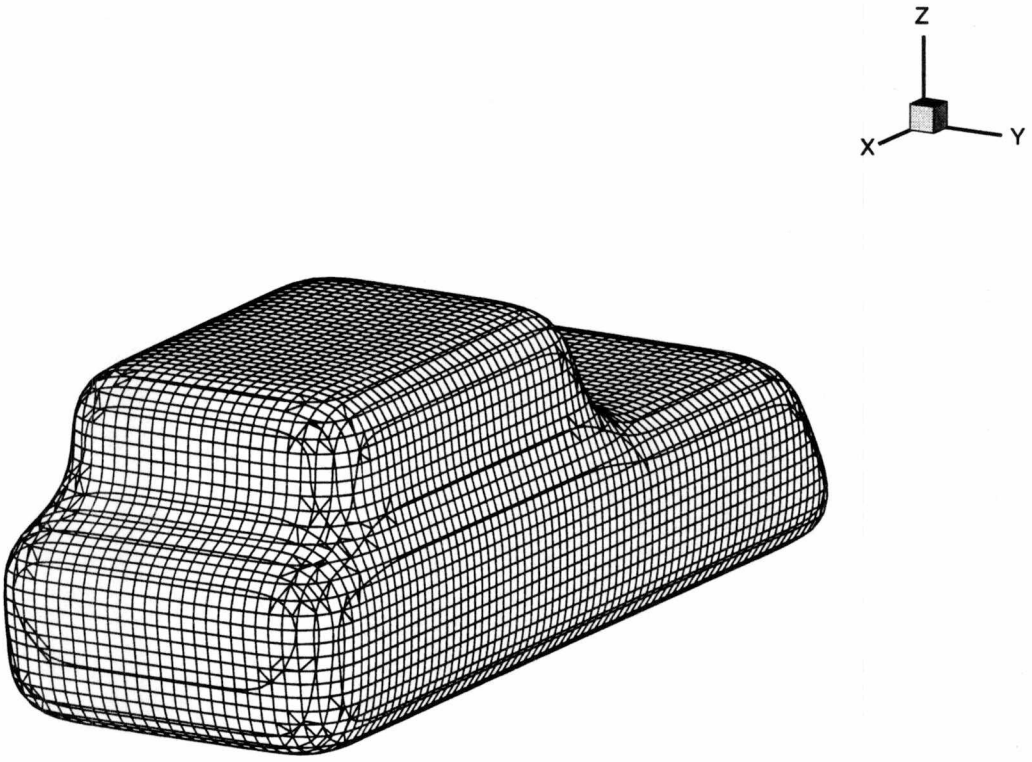


Figure 3.19: Final F Function for Generic Car.

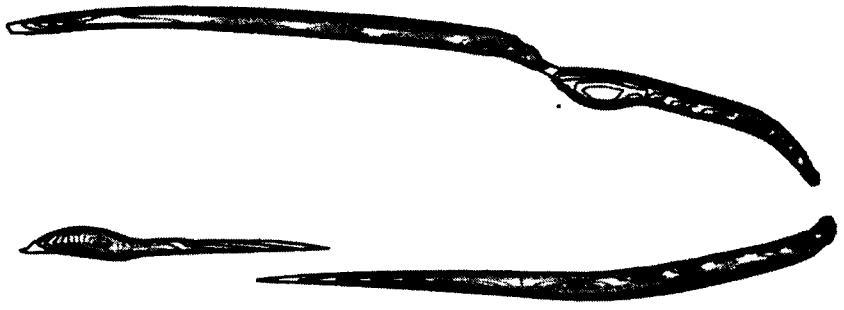


Figure 3.20: Vorticity Contour at Mid Plane.

Figure 3.21: Particle Trace.

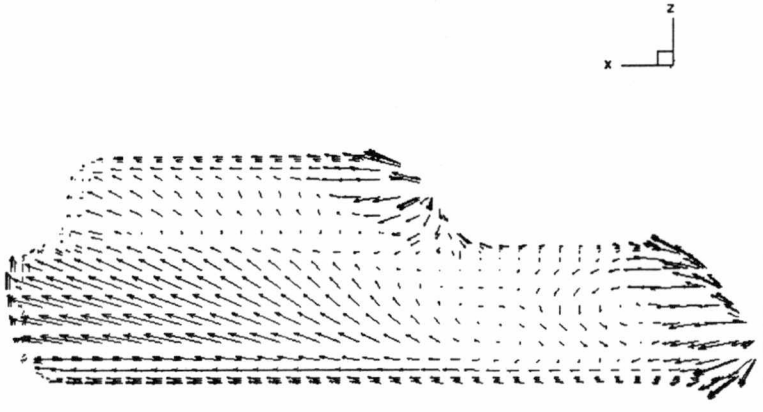


Figure 3.22: Velocity Vector Near the Surface (Side View).

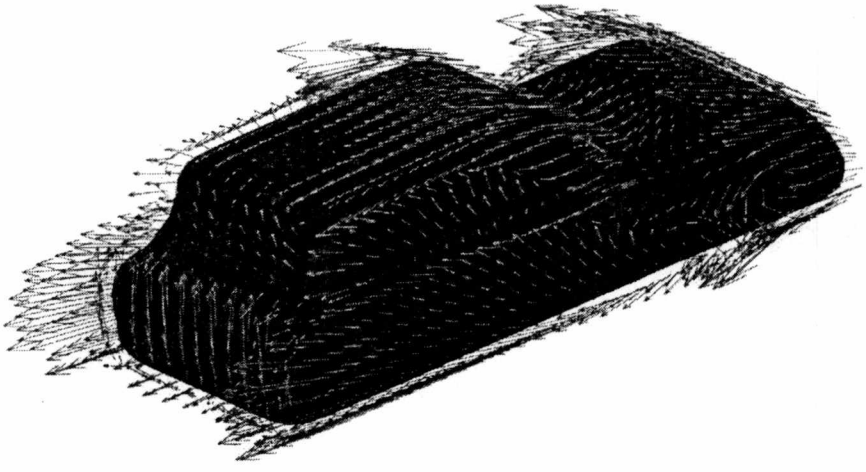


Figure 3.23: Velocity Vector near the Surface (3–D View).

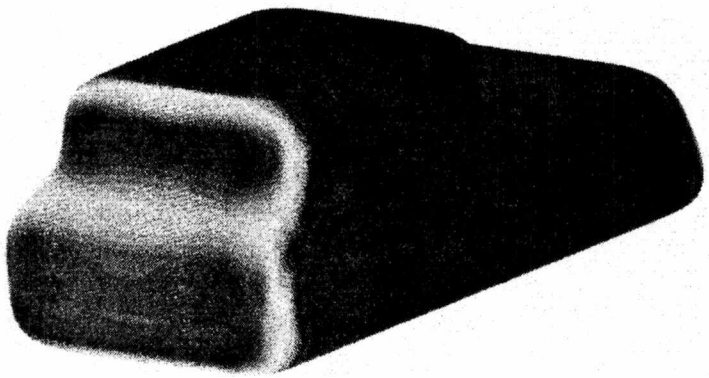


Figure 3.24: Pressure Distribution on the Generic Car Surface.

mesh was shown in Fig. 3.25. The isosurface of $F = 0$ is shown in Fig. 3.26. Fig. 3.27 shows the $F = 0$ contour line at the mid plane of the car within the computational grid. Fig. 3.28 shows the contour lines of F function in the same plane. Fig. 3.29 shows the $F = 0$ contour line at the mid plane in axial direction of the car with computational grid and Fig. 3.30 shows the contour lines of F function in that plane. The computational grid for flow over the Passat Limousine B3 was $128 \times 48 \times 48$, the body was 92 cells long in i direction, 33 cells in width in j direction, and 24 cells high in k direction. Due to the coarse grid, the F function was not as smooth in the roof and hood areas as in the other regions.

Fig. 3.31 shows the computed pressure coefficient (C_p) distribution on the body surface. The pressure value shown is interpolated to the physical body surface, not the computational body surface which is defined by $F = 0$, from the

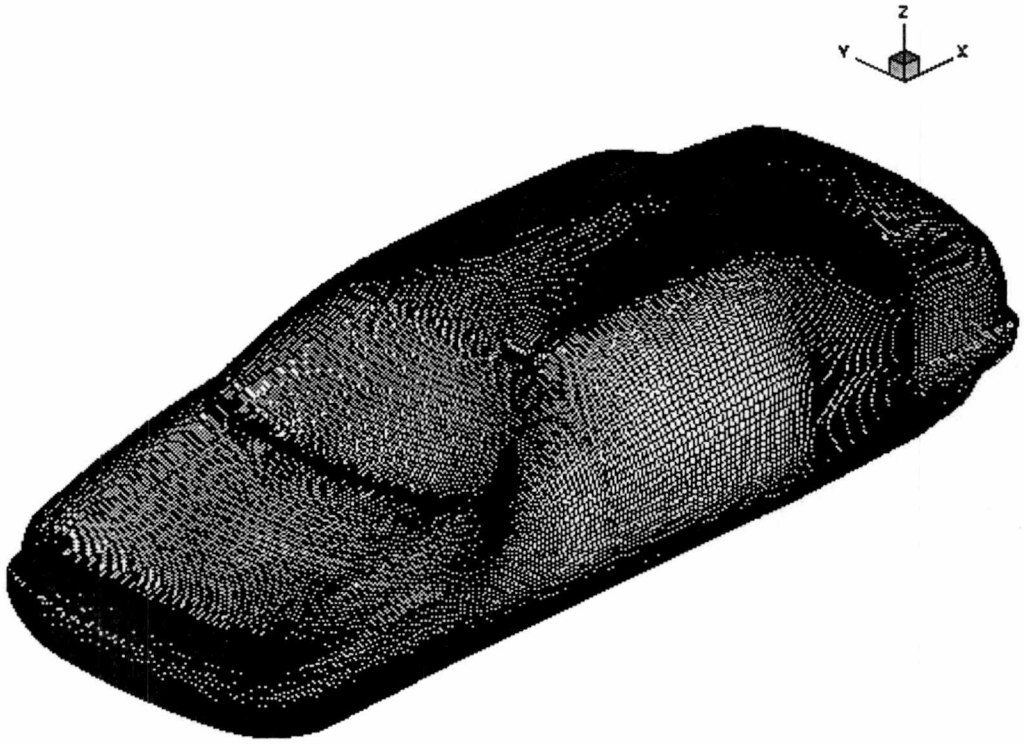


Figure 3.25: Passat Limousine B3 Configuration with Mesh.

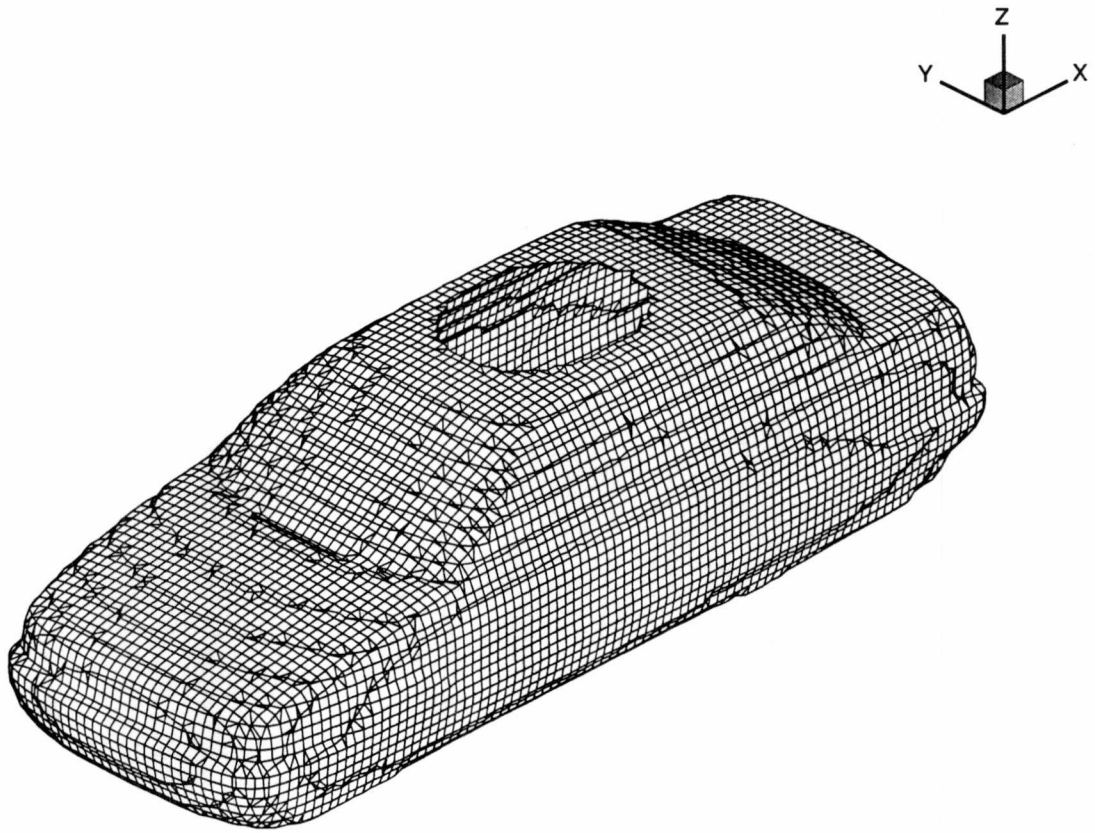


Figure 3.26: Isosurface of $F = 0$.

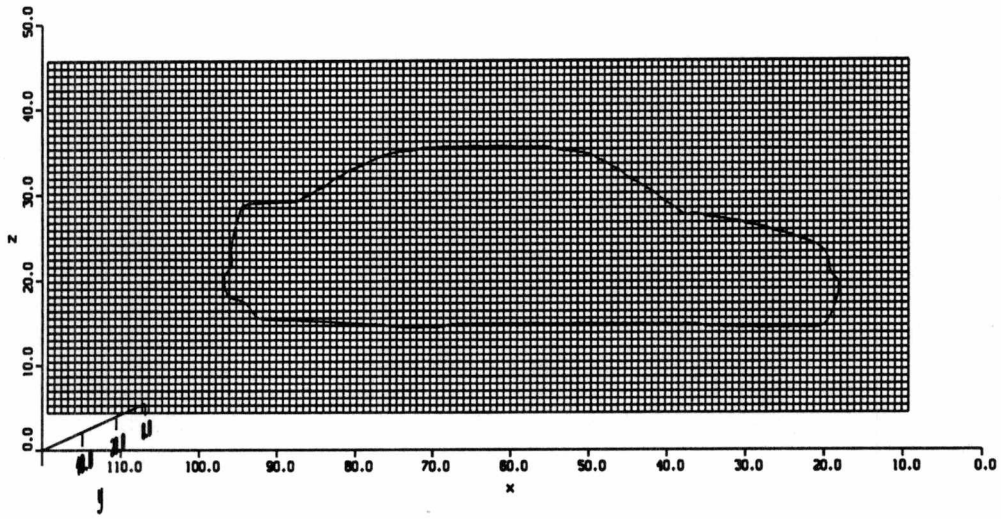


Figure 3.27: $F = 0$ Contour at Mid Plane with Grid.

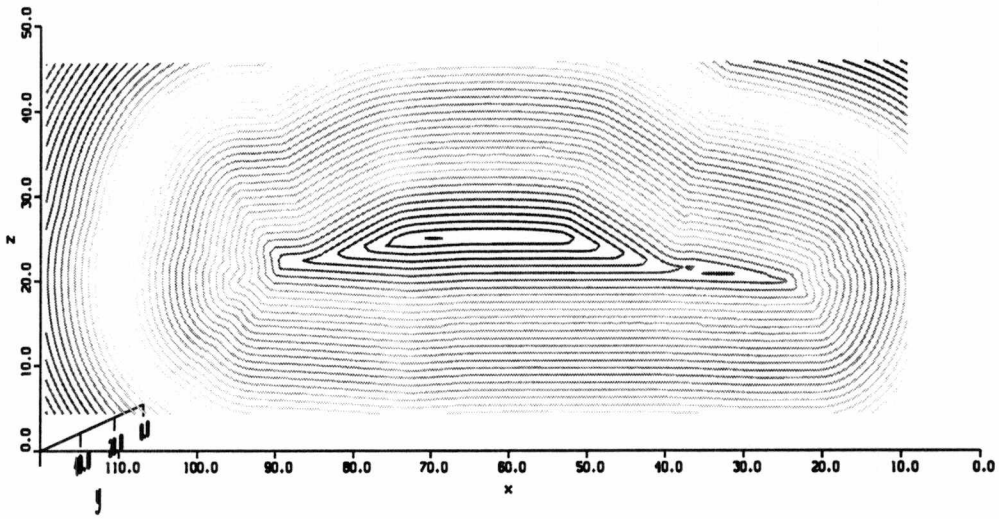


Figure 3.28: Contour Lines of F Function at Mid Plane.

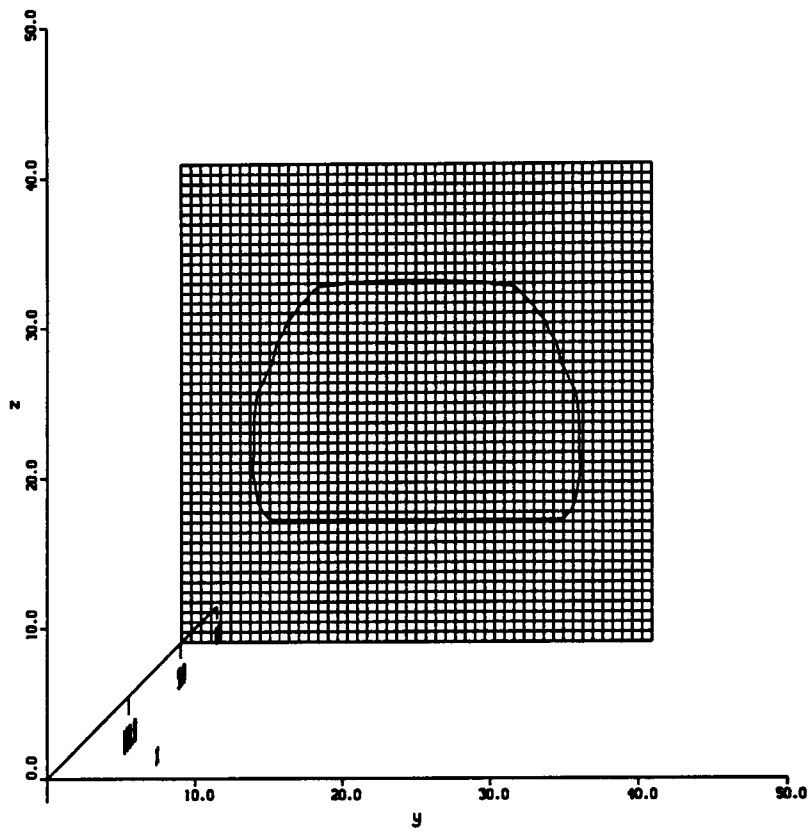


Figure 3.29: $F = 0$ Contour at Cross Section with Grid.

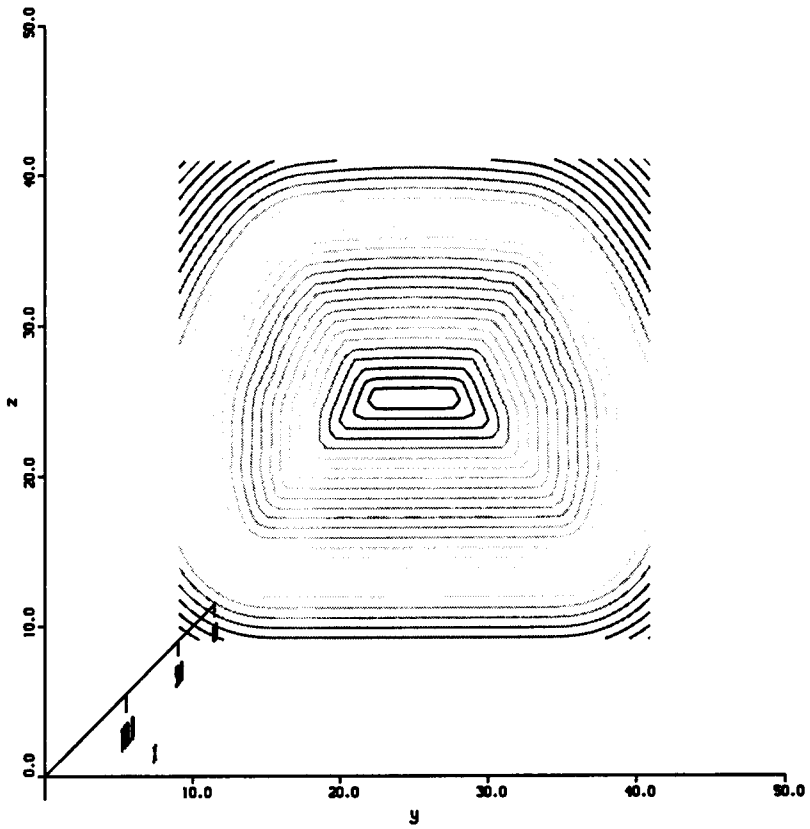


Figure 3.30: Contour Lines of F Function at Cross Section.

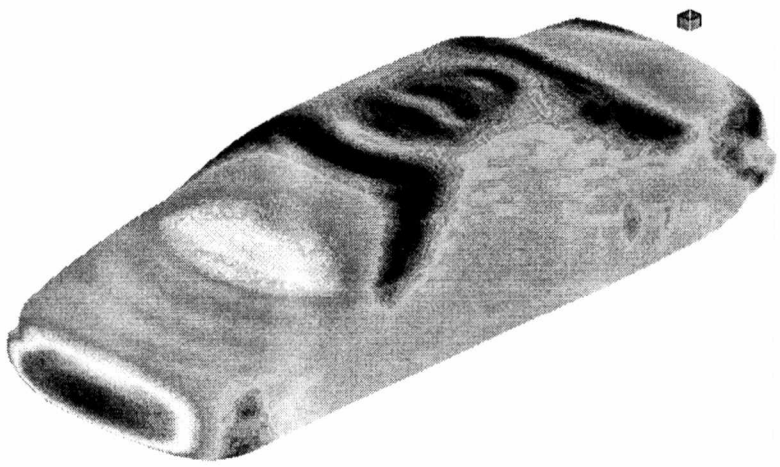


Figure 3.31: Pressure Distribution on Passat Limousine B3 Surface.

computational domain. The highest pressure (stagnation point) appeared at the front of car, as it should. Fig. 3.32 shows the C_p distribution in the mid plane. Fig. 3.33 shows the velocity vectors near the car surface. The velocity is interpolated to $F = s$ (where s is a small positive value, eg. 0.1) which corresponded to regions outside the body surface. Flow separation zones from front, side, roof, and back of the car are visualized. In the back of the car, the rear wake vortex is well generated.

3.2.3 Realistic Helicopter (Apache) Body in Forward Flight (Rotorless)

The geometry of a realistic helicopter, an Apache–A was provided by NASA Langley Research Center. The body surface consisted of 29562 panels with 14779 node points. The configuration of the Apache helicopter body is much more

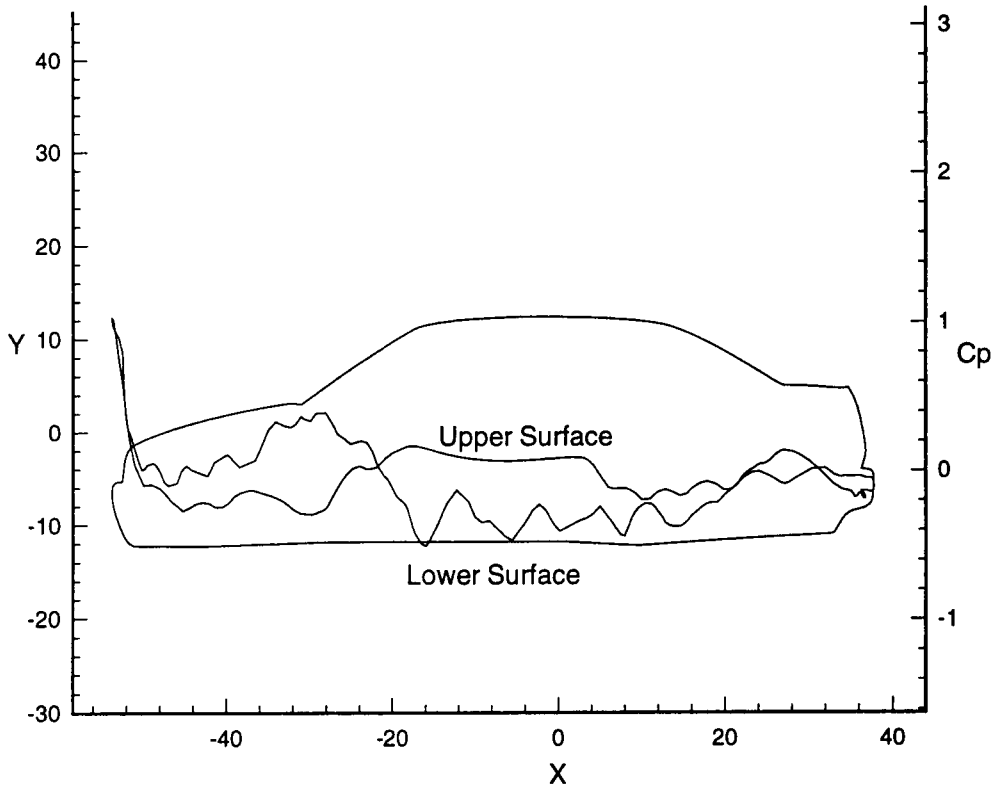
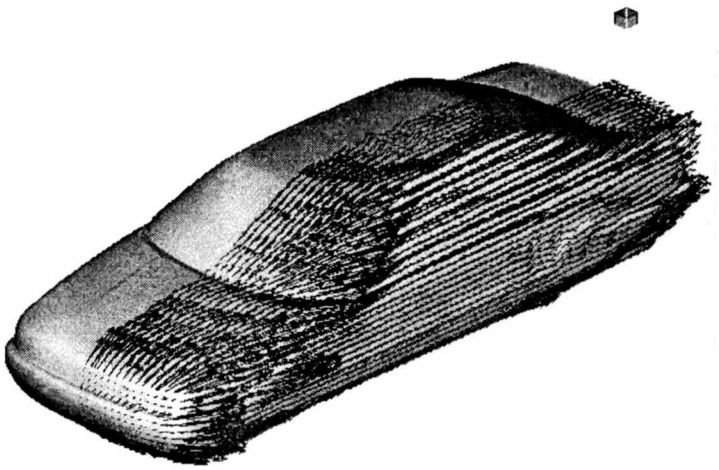
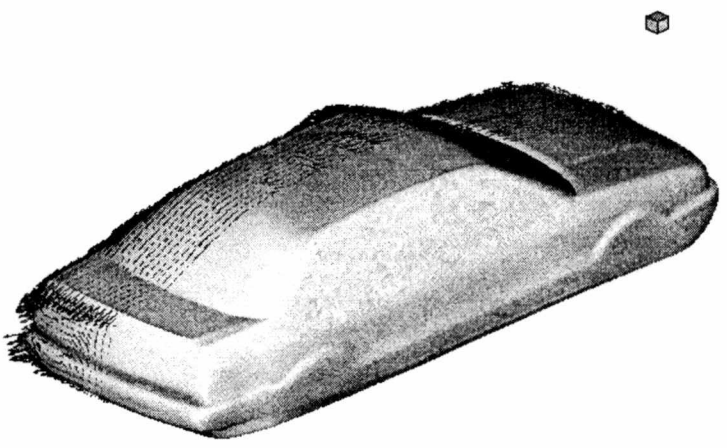


Figure 3.32: Mid Plane Pressure Distribution.



3.33(a)



3.33(b)

Figure 3.33: Velocity Vector near the Passat Limousine B3 Surface (a) Left–Front View (b)Right–Back View.

complicated than the automobile previously computed. It contains wings, tail and fuselage in its configuration. The numerical configuration of the Apache helicopter body with mesh points is shown in Fig. 1.1.

The F function was generated by method (c) described in Section 2.4.1. Because of the complexity of the body geometry, the simple logic previously used to determine the F distribution to define the car could not be satisfied in certain helicopter regions. Extra work was done to check the F function, especially the sign of the F, plane by plane in slices through the helicopter. For example, in the region close to the tail, since the panels on the tail were not regularly ordered, it was difficult to use a simple logic to determine the normal distance from a grid node to the closest panel with the correct sign. The isosurface of $F = 0$ is shown in Fig. 3.34 and is close to the body shape of the real Apache. Fig. 3.35 shows the $F = 0$ contour line at the mid plane of the Apache within the computational grid. Fig. 3.36 shows the contour lines of F function in the same plane. Fig. 3.37 shows the $F = 0$ contour line at the cross plane in axial direction of the Apache, $i = 44$, with computational grid and Fig. 3.38 shows the contour lines of F function in that plane.

The computational grid for flow over the Apache helicopter body was $128 \times 128 \times 64$, the body being 82 cells long in i direction, 24 cells in width in j direction, and 20 cells high in k direction. Three cases have been investigated for the Apache helicopter body in forward flight without a rotor included.

A zero angle of attack case was first computed. Fig. 3.39 shows the surface pressure distribution on the body surface of Apache–A in forward flight

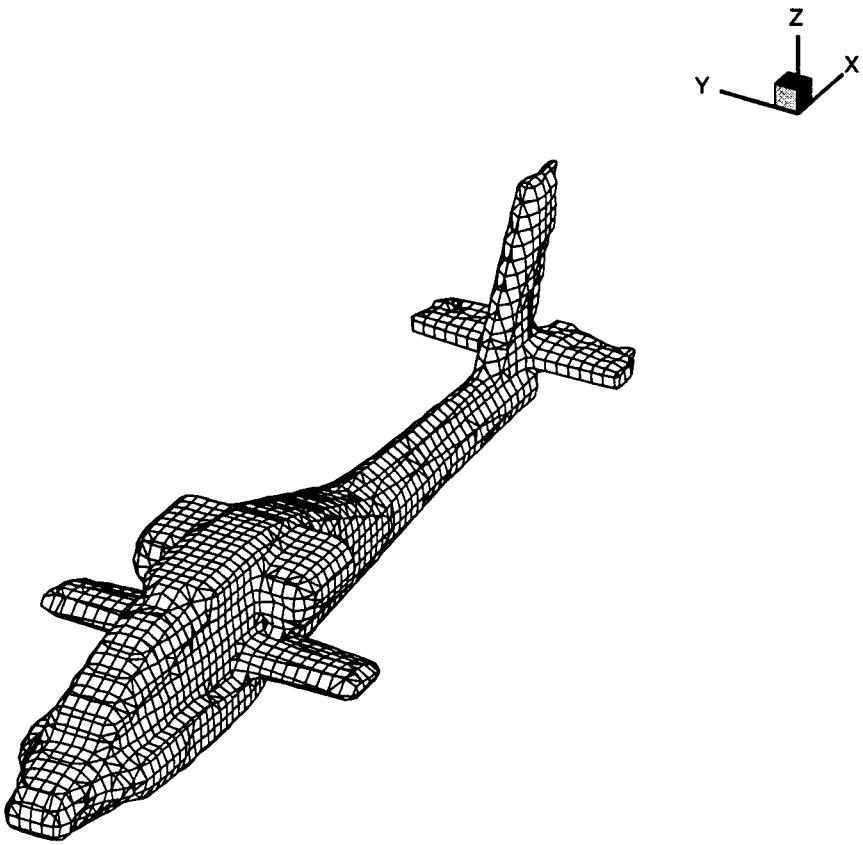


Figure 3.34: Isosurface of $F = 0$.

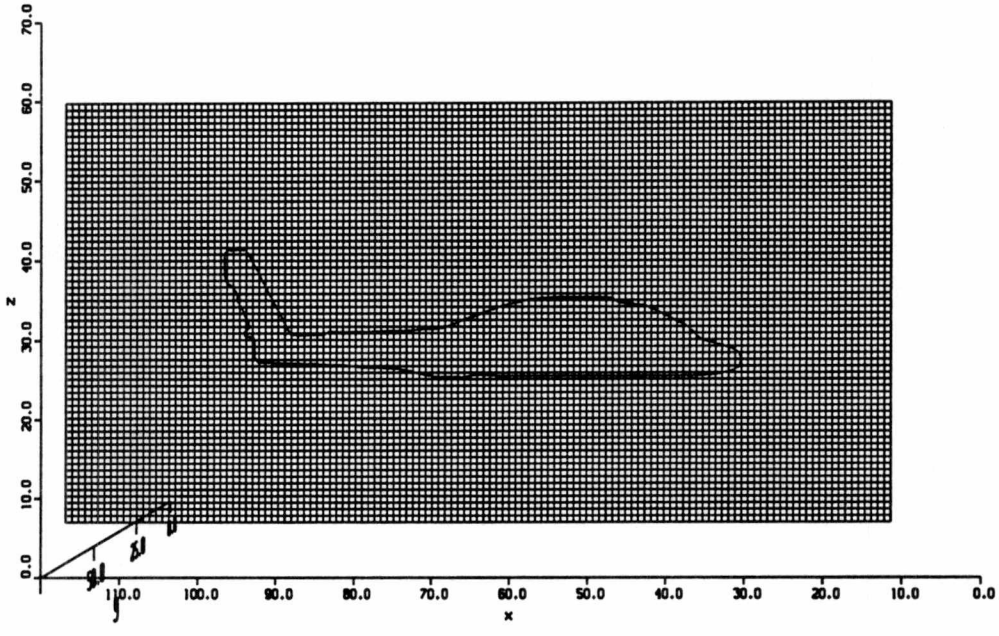


Figure 3.35: $F = 0$ Contour at Mid Plane with Grid.

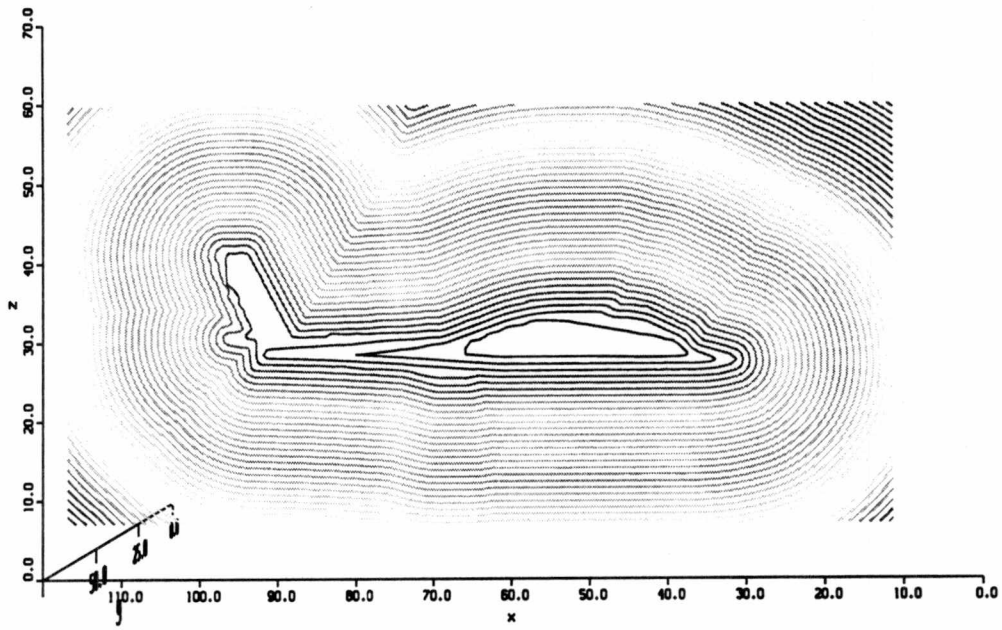


Figure 3.36: Contour Lines of F Function at Mid Plane.

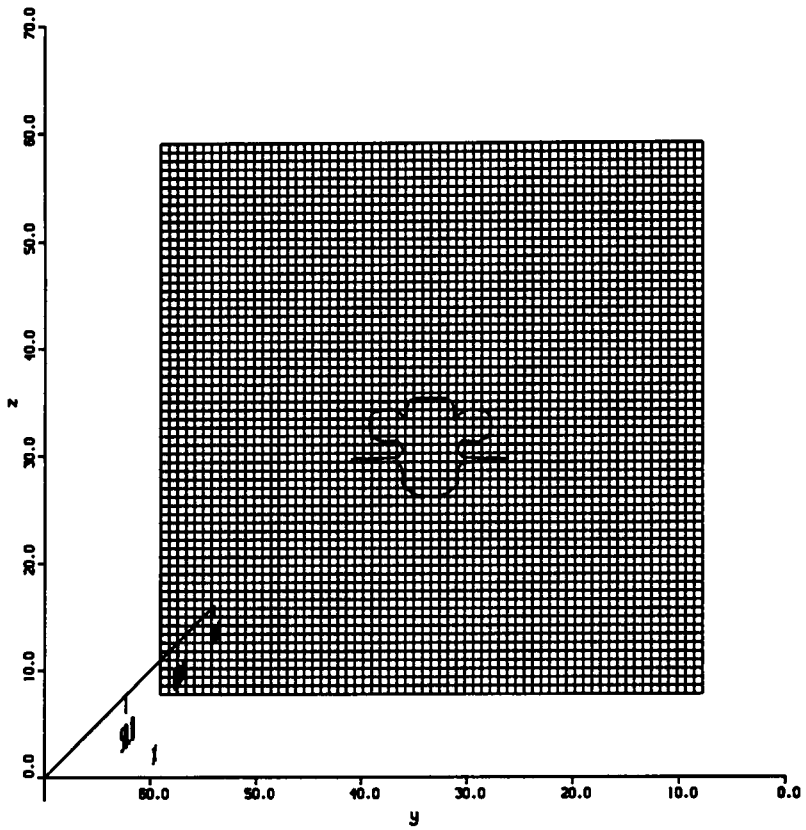


Figure 3.37: $F = 0$ Contour at Cross Section with Grid.

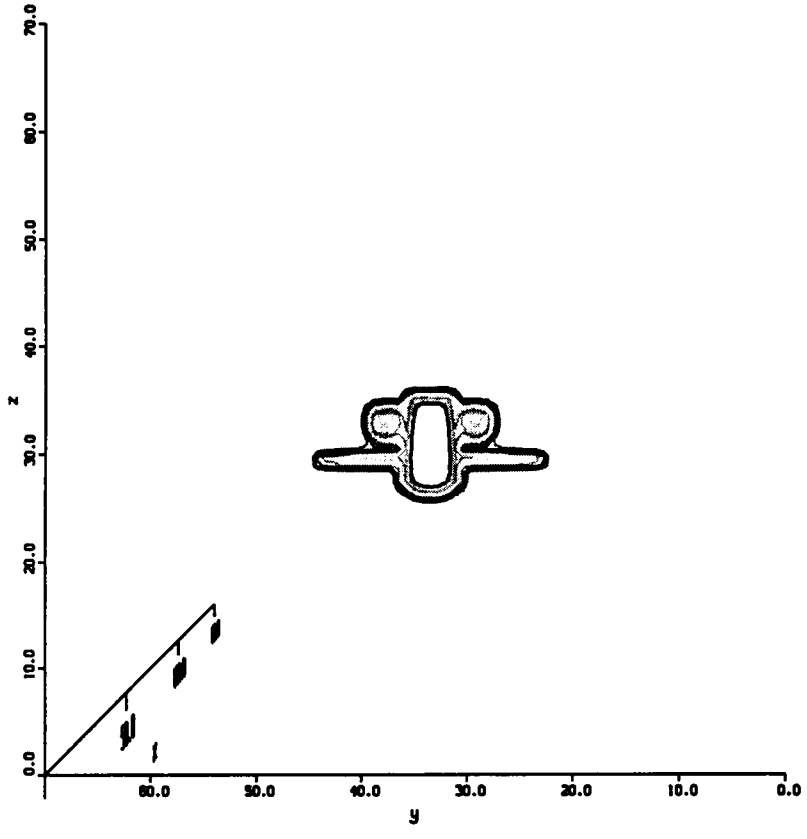


Figure 3.38: Contour Lines Close to $F = 0$ at Cross Section.

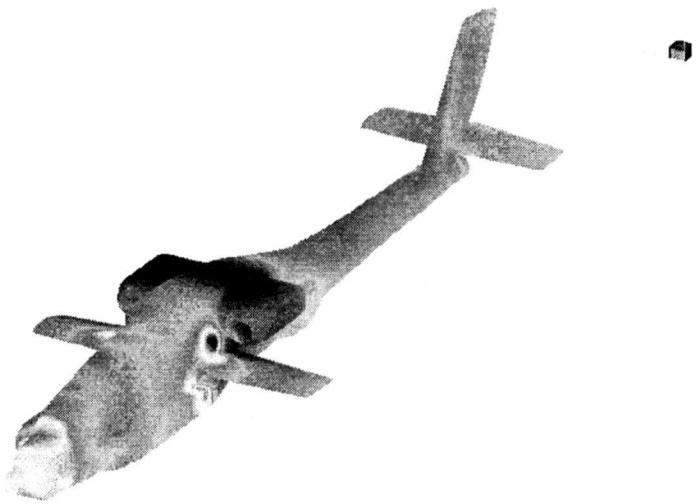


Figure 3.39: Pressure Distribution on Apache–A Surface with $\alpha = 0°$.

with no angle of attack. Fig. 3.40 is the C_p distribution in the mid plane of the configuration. Both figures that show the stagnation point is in the front of the body. Fig. 3.41 shows the vorticity contours of the whole flow field. For the purpose of obtaining a better view, the vorticity on the body surface (for $F < 0.1$) was cut off. It can be seen that the vortices shed from the fan shell impact the tail, which causes a high pressure region in that area.

The second computational case is Apache–A in forward flight with 20° angle of attack, again rotorless. Fig. 3.42 shows the surface pressure distribution for this case. The figure shows that the stagnation point is located lower on the surface. The vortices generated by the body surface are well captured, and travel downstream to the end of the computational domain at 20° angle. The C_p

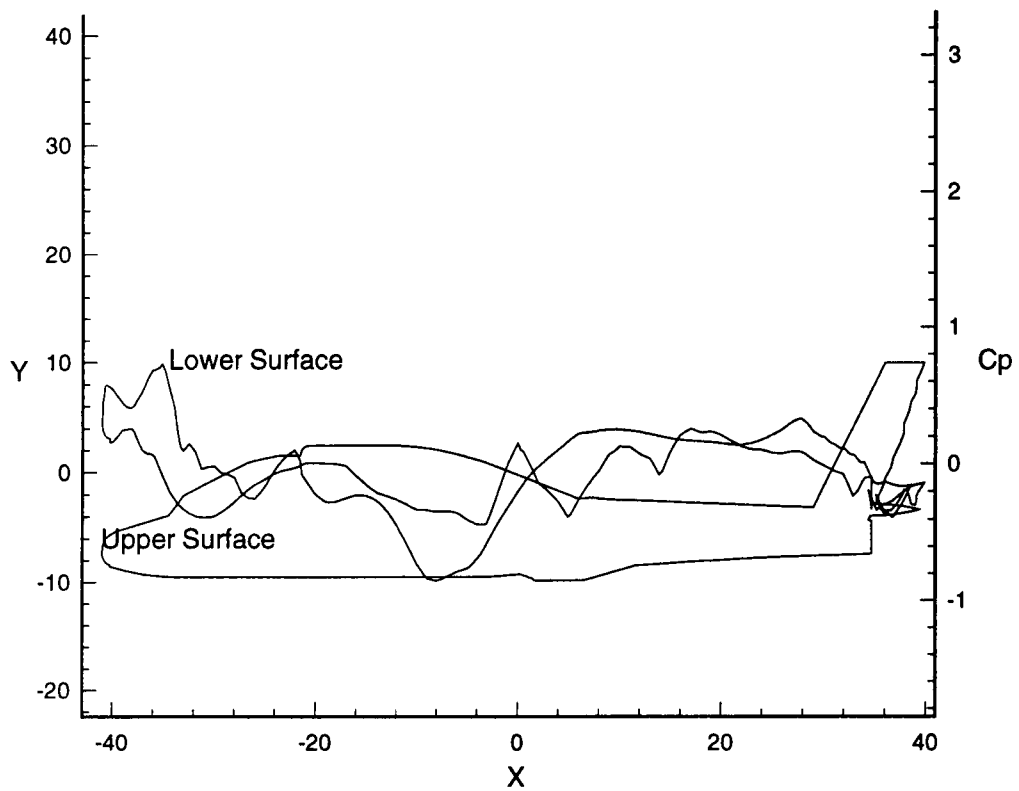
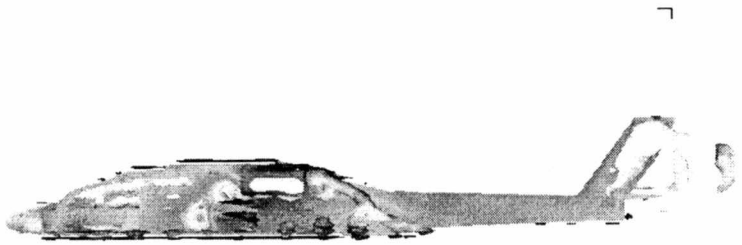
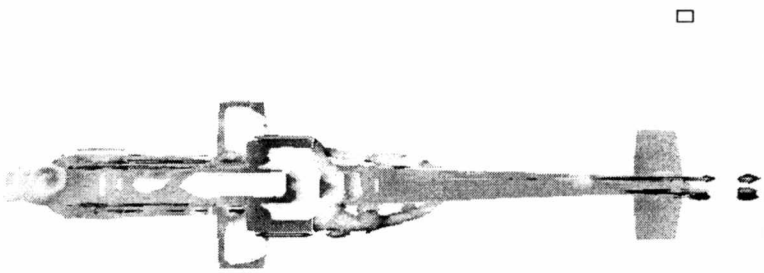


Figure 3.40: Mid Plane Pressure Distribution $\alpha = 0°$.

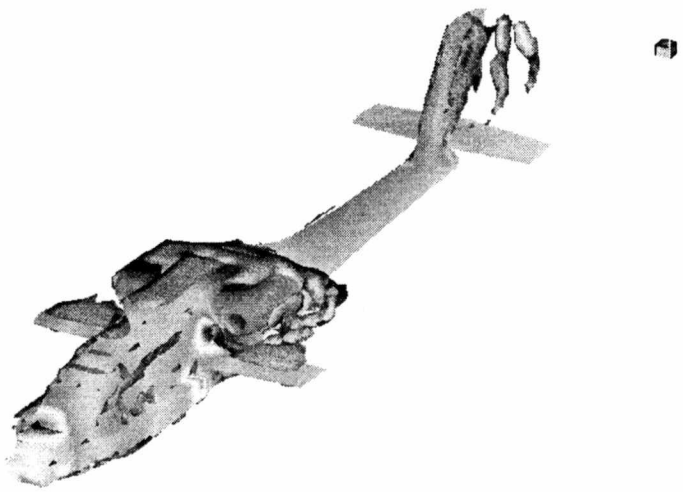


3.41(a)



3.41(b)

Figure 3.41: Isosurface Vorticity Contour on Apache–A with $\alpha = 0°$ (a) Side View (b) Top View (c) Perspective View.



3.41(c)

Figure 3.41 (continued)

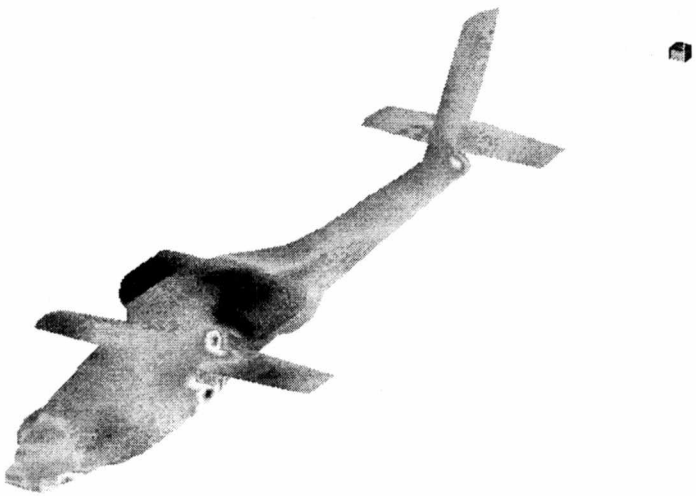


Figure 3.42: Pressure Distribution on Apache–A Surface with $\alpha = 20°$.

distribution in the mid plane is shown in Fig. 3.43. Fig. 3.44 shows the vorticity contour in the flow field.

In the third case, a modified Apache–A body, denoted Apache–D, is treated in forward flight with 20° angle of attack, also rotorless. This helicopter configuration has a bigger pod on one side of the fuselage. The F function was modified by stretching the F value at the bigger pod location. The isosurface of $F = 0$ is shown in Fig. 3.45. Fig. 3.46 shows the pressure distribution on the body surface. Fig. 3.47 shows the vorticity contours in the flow field. This computed solution is different from that of the second case. The vorticity generated by the bigger pod travels under the wing and hits the rear fuselage, which has been confirmed by experiment.

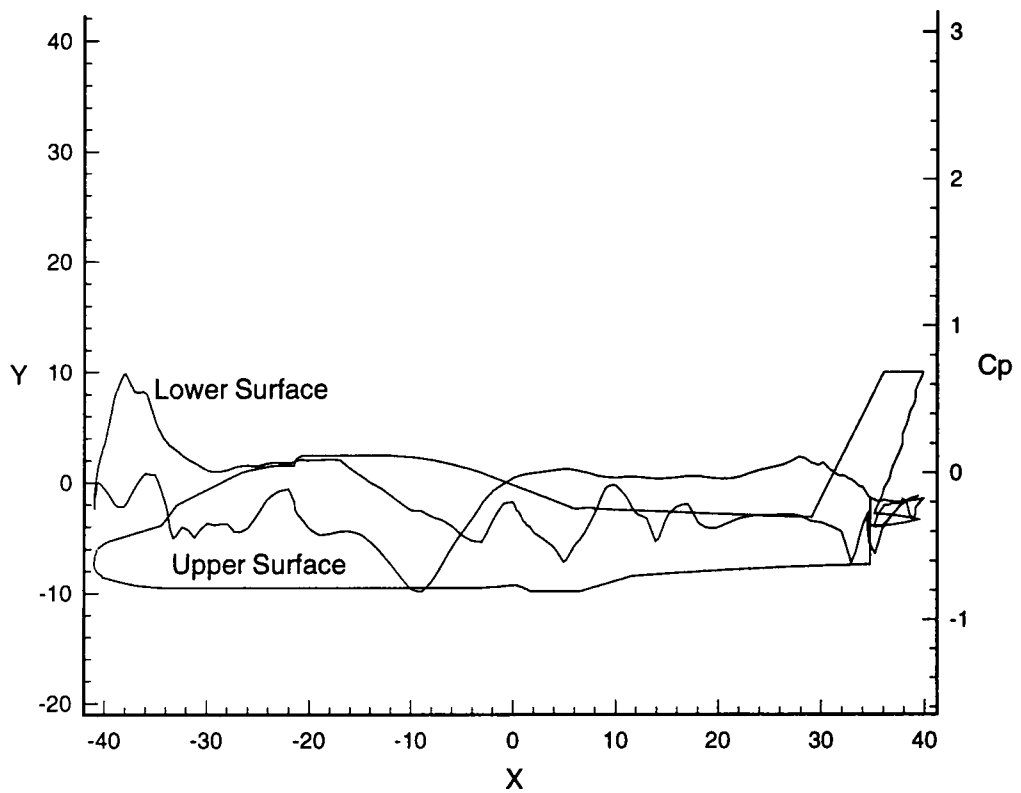
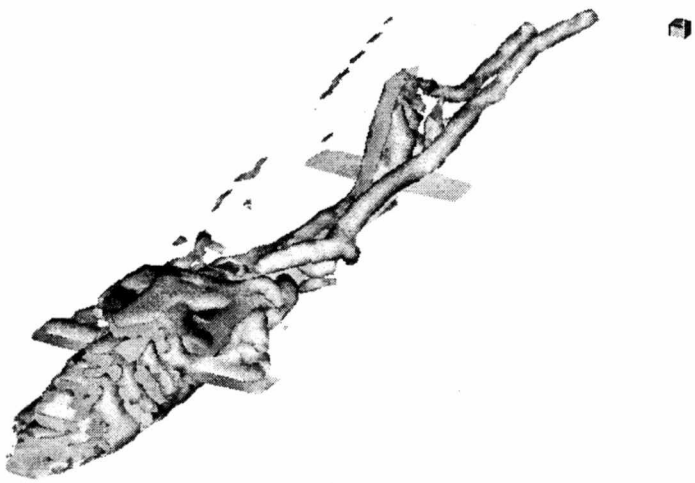


Figure 3.43: Mid Plane Pressure Distribution $\alpha = 20°$.

3.44(a)

3.44(b)

Figure 3.44: Isosurface Vorticity Contour on Apache–A with $\alpha = 20°$ (a) Side View (b) Top View (c) Perspective View.



3.44(c)

Figure 3.44 (continued)

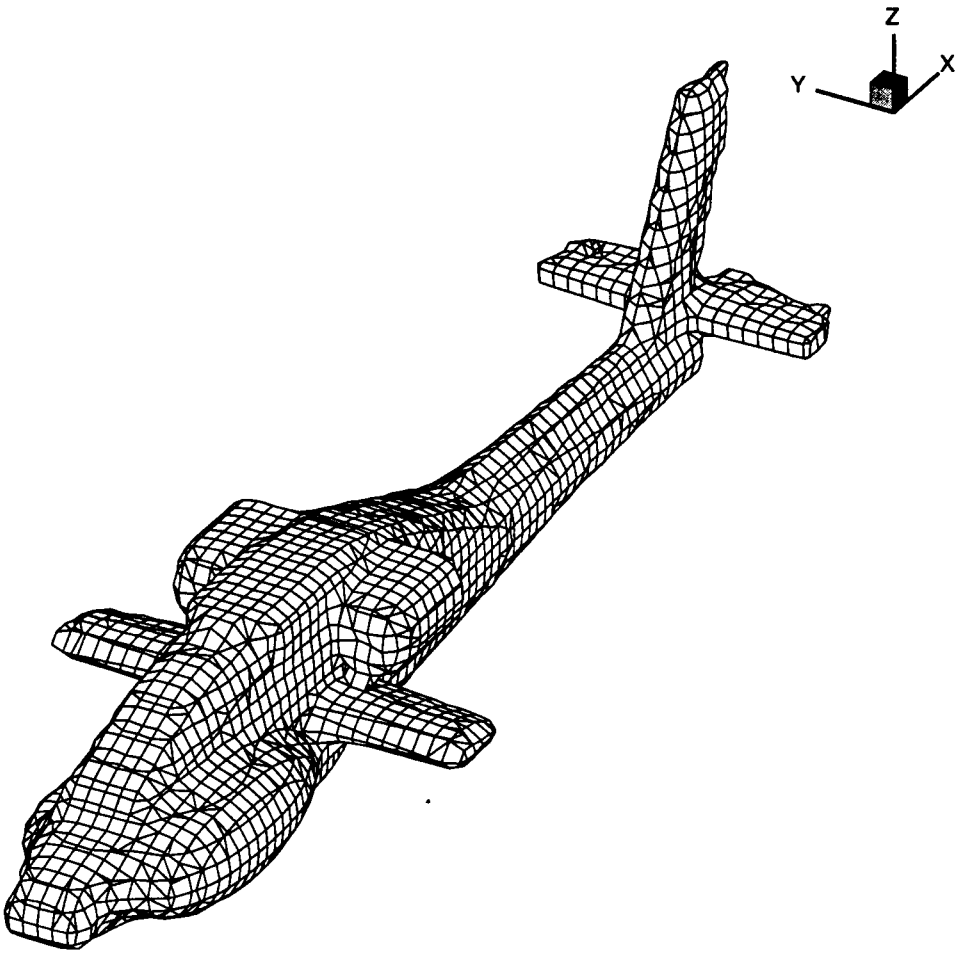


Figure 3.45: Isosurface of $F = 0$.

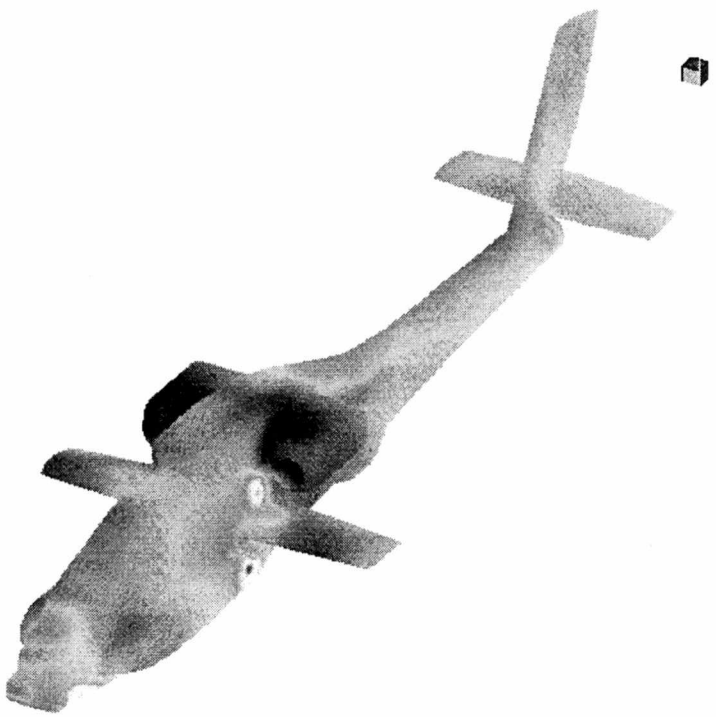
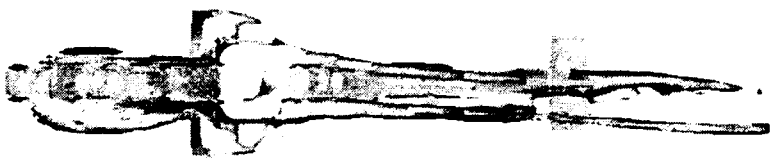


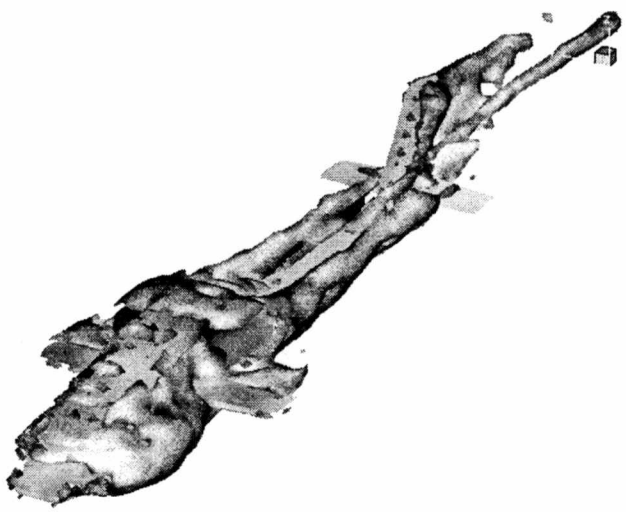
Figure 3.46: Pressure Distribution on Apache–D Surface.

3.47(a)



3.47(b)

Figure 3.47: Isosurface Vorticity Contour on Apache–D (a) Side View (b) Top View (c) Perspective View.



3.47(c)

Figure 3.47 (continued)

3.3 Rotor/Body Flow Computation

3.3.1 Generic Helicopter with One Rotor Blade in Hover

The first computational study of a generic helicopter body, and a single rotor blade, in a hover flight mode was carried out in the present study. The generic helicopter body was generated by two ellipsoids connected together on their major axes. The F function for the body was generated by method(a) in Section 2.4.1. The isosurface of $F = 0$ is shown in Fig. 3.48

The body grid for this computation was $128 \times 128 \times 64$ in the i, j, k direction. The generic helicopter body was located in the middle of the computational domain. The rotor blade was simulated rotating in a plane 16 cells above the body center line. The rotor blade was simulated with 5 cells in chord, and 45 cells along the span from hub to tip. The largest cross-section radius of the body was about 8 cells. The rotor blade was assumed to rotate at a constant speed which resulted in an angular displacement of about 6 degrees per time step. To reduce computing time, an interpolation region was defined by a domain of size of $40 \times 40 \times 20$ behind the blade tip. Since the rotor blade solution is a fixed wing solution obtained with the TURNS code, a small downward freestream velocity was initially added everywhere to the flow field to simulate the down flow induced by the rotor blade in hover.

A global solution at 180 computational time steps is shown in Fig. 3.49. The tip vortices traveled downstream in a helical path about 160 chord lengths. In the same plot, also shown are the interactions of tip vortices with the body. The

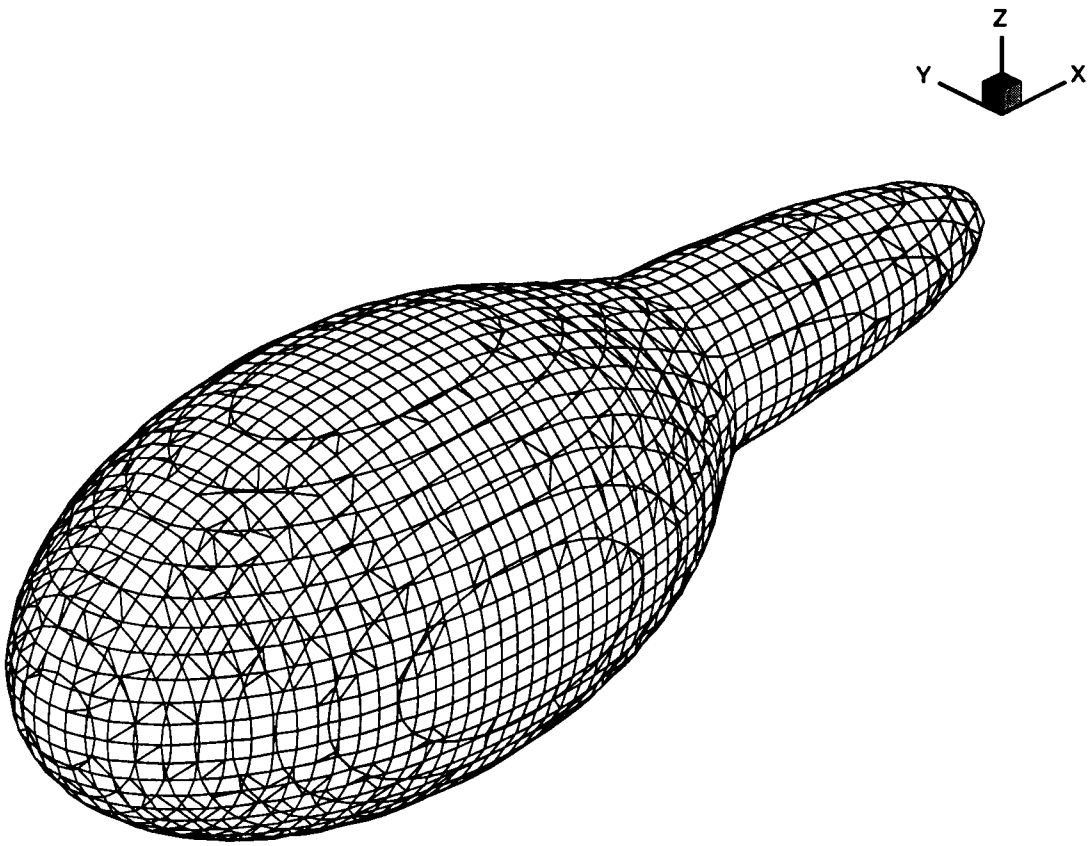


Figure 3.48: Isosurface of $F = 0$.

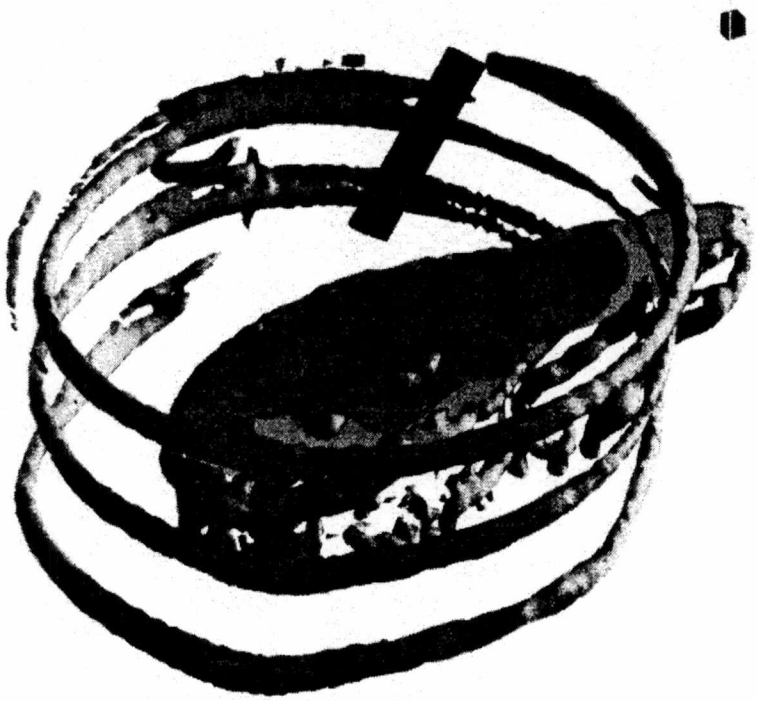


Figure 3.49: Generic Helicopter with One Rotor Blade in Hover.

vortices were then shed from the body after their interaction. It can be seen that the surface pressure changed in the interaction region.

3.3.2 Apache with Four Rotor Blades in Hover

The last case discussed is a simulation of an Apache helicopter with four rotating rotor blades in hover which was computationally simulated. First, however, four free blades were computed rotating freely in the flow field with no helicopter fuselage present. The results of this four free–blade simulation are shown in Fig. 3.50. The Cartesian grid dimension utilized was $128 \times 128 \times 64$ in the i, j, k direction. The blades were located at 3/4 of the height of the domain from the bottom. The solution shows vorticity levels at 30% of the maximum vorticity magnitude generated in the solution. The tip vortex from each blade traveled one complete revolution (56 chord lengths). The computation was performed using a CRAY–YMP supercomputer. The solution took about 1.25 hours of CPU time.

With the experience gained in computing the four isolated, rotating blades, the next computational case attempted was the real helicopter (Apache–A) body with four rotor blades in hover. The computational domain was again $128 \times 128 \times 64$. The blades were treated as rotating in 8 cells above the body. Fig. 3.51 shows the isosurface of the tip vortices contour at level 30% of the maximum computed vorticity magnitude. The computed surface pressure distributions on the helicopter body surface are also shown in the same figure. The interaction of tip vortices and the body surface can be seen clearly.

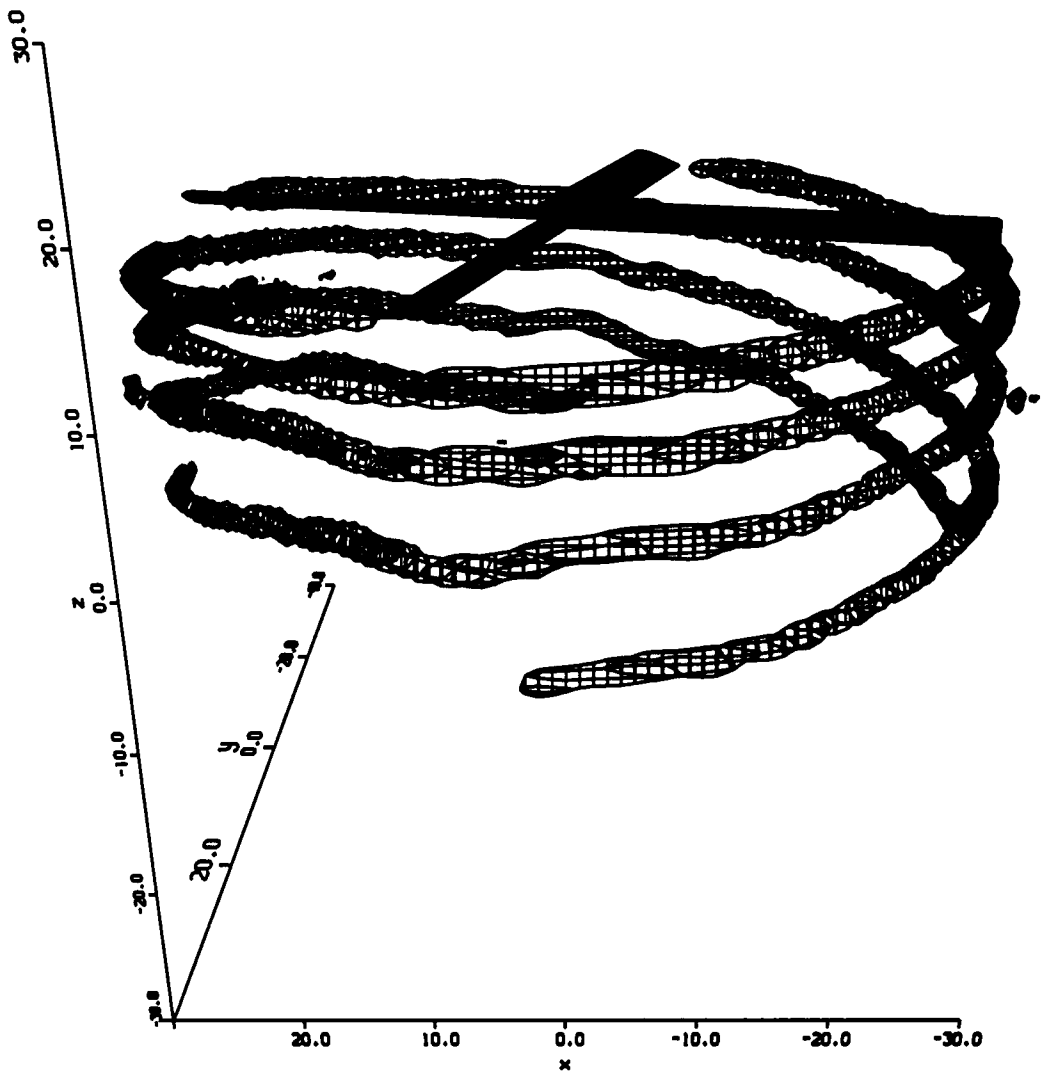


Figure 3.50: Tip Vortices Shed From Four Rotor Blades.

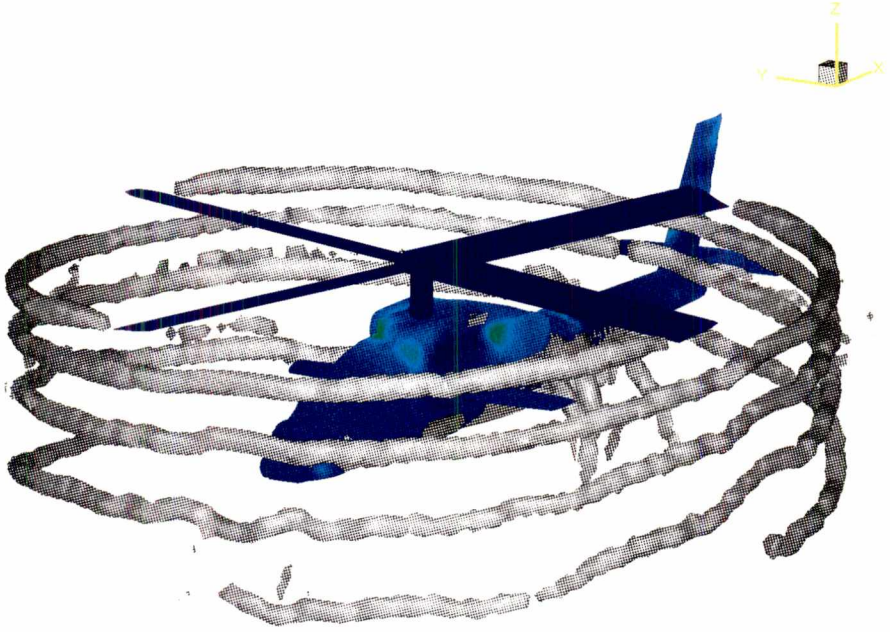


Figure 3.51: Apache Helicopter with Four Rotor Blades in Hover.

Chapter 4

CONCLUSION AND REMARKS

A new computational method for computing flows with concentrated vorticity in a simple Cartesian grid over complex aerodynamic configuration has been developed. Several examples have been presented which demonstrate the following capabilities of the new numerical approach. The new approach can:

1. efficiently analyze arbitrary configurations in incompressible flow.

2. reduce the set up time to solve a complex geometry problem by generating "F" function distributions that require only $1 \sim 2$ days.

3. generate and track the flow separation from solid body surfaces using a relatively coarse modeling grid.

4. confine concentrated vorticity to within a few grid cells.

5. overcome the effects of numerical diffusion in the flow domain solver on vortex convection.

6. be used to analyze general rotorcraft flow.

The computer code developed in the present study can be used as a tool in the aircraft industry, as well as in automobile industry, for design and analysis of new products. Typically, a new geometry configuration might only require a

weeks effort to get a flow solution. These solutions may be also used to assist the scientists and engineers before a test by improving the conceptual design.

Future work recommended is:

1. Improve the "F" function generator. As discovered in the computation of flow over a Passat Limousine B3, the flow solution was sensitive to the F function distributions.

2. Investigate the relation between the confinement parameter ε and the Reynolds number Re.

3. Do quantitative analysis for the computational solutions. Compare the numerical results with experimental data as data becomes available.

4. Include a turbulence model for the flow. The flow behind automobiles and in aircraft wakes usually develops large–scale turbulence that could not be modeled in the present version of the computational method.

BIBLIOGRAPHY

Bibliography

[1] Wang, C., Bridgeman, J., Steinhoff, J., and Wenren, Y., "The Application of Computational Vorticity Confinement to Helicopter Rotor and Body Flow." Proceedings, 49th American Helicopter Society Meeting, St. Louis, MO,, May 1993.

[2] Rai, M. M., "Navier-Stokes Simulations of Blade-Vortex Interaction Using High-Order Accurate Upwind Schemes," AIAA Paper, no. 87-0543, 1987.

[3] Oden, J. T., Strouboulis, T., and Devloo, P., "Moving-Grid Finite Element Algorithm for Supersonic Flow Interaction Between Moving Bodies," Computational Methods in Applied Mechanical Engineering, vol. 59, 1986.

[4] Strawn, R., "Wing Tip Vortex Calculation with an Unstructured Adaptive-Grid Euler-Solver." presented at the *47th AHS Forum of the American Helicopter Society*, Phoenix, AZ, May 1991.

[5] Moore, D. W., "A Numerical Study of the Roll-Up of a Finite Vortex Sheet," Journal of Fluid Mechanics, vol. 63, pp. 225–235, Apr. 1974.

[6] Hoeijmakers, W. M. and Vaatstra, W., "A Higher Order Panel Method Applied to Vortex Sheet Roll-Up," AIAA Journal, vol. 21, pp. 516–523, Apr. 1983.

[7] Steinhoff, J. and Ramachandran, K., "A Vortex Embedding Method for Free-Wake Analysis of Helicopter Rotor Blades in Hover," Vertica, vol. 13, no. 2, pp. 133–141, 1989.

[8] Steinhoff, J., Wenren, Y., Mersch, T., and Senge, H., "Computational Vorticity Capturing: Application to Helicopter Rotor Flow," AIAA Paper, no. 92-0056, 1992.

[9] Steinhoff, J. S. and Underhill, D., "Modification of the Euler Equations for Vorticity Confinement: Application to the Computation of Interacting Vortex Rings," Physics of Fluids, vol. 6, pp. 2738–2744, Aug. 1994.

[10] Kim, J. and Moin, P., "Application of a Fractional-Step Method to Incompressible Navier-Stokes Equations," Journal of Computational Physics, vol. 59, no. 2, p. 308, 1985.

[11] Steinhoff, J., Mersch, T., and Decker, F., "Computation of Incompressible Flow Over Delta Wings Using Vorticity Confinement," AIAA Paper, no. 94-0646, 1994.

[12] Adams, J., Swarztrauber, P., and Sweet, R., "FISHPAK: A Package of FORTRAN Subprograms for the Solution of Separable Elliptic Partial Differential Equations." FORTRAN Netlib, Nov. 1988.

[13] Srinivasan, G. R., Baeder, J. D., Obayashi, S., and McCroskey, W. J., "Flowfield of a Lifting Rotor in Hover: Navier–Stokes Simulation," AIAA Journal, vol. 30, pp. 2371–2378, Oct. 1992.

[14] Tomassian, J. D., *The Motion of a Vortex Pair in a Stratified Medium.* PhD thesis, University of California, Los Angeles, CA, 1979.

[15] Sarpkaya, T., "Trailing Vortices in Homogeneous and Density-Stratified Media," Journal of Fluid Mechanics, vol. 136, pp. 85–109, 1983.

[16] Liu, H. T. and Srnsky, R. A., "Laboratory Investigation of Atmospheric Effects on Vortex Wakes," Flow Research Report 497, Flow Research, Inc., Kent, WA, 1990.

[17] Delisi, D. P. and Greene, G. C., "Measurements and Implication of Vortex Motion Using Two Flow-Visualization Techniques," Journal of Aircraft, vol. 27, pp. 968–971, 1990.

[18] Crow, S. C., "Stability Theory for a Pair of Trailing Vortices," AIAA Journal, vol. 8, pp. 2172–2179, 1970.

[19] Delisi, D. P., Robins, R. E., and Altman, D. B., "Laboratory and Numerical Studies of Vortex Evolution in Ideal and Realistic Environments." Proceedings of the Aircraft Wake Vortices Conference, Washington, D.C., Oct. 1991. DOT/FAA/SD-92/1.1, 1992.

[20] Barker, S. J. and Crow, S. C., "The Motion of Two-Dimensional Vortex Pairs in a Ground Effect," Journal Fluid Mechanics, vol. 82, pp. 659–671, 1977.

[21] Delisi, D. P., Robins, R. E., and Fraser, R. B., "The Effects of Stratification and Wind Shear on the Evolution of Aircraft Wake Vortices Near the Ground:

Phase I Results," NWRA report NWRA-87-r006, Northwest Research Associates, Inc., Bellevue, WA, 1987.

[22] Walker, J. D. A., Smith, C. R., Cerra, A. W., and Doligalski, T. L., "The Impact of a Vortex Ring on a Wall," Journal of Fluid Mechanics, vol. 181, pp. 99–140, 1987.

[23] Bilanin, A. J., Teske, M. E., and Hirsh, J. E., "Neutral Atmospheric Effects on the Dissipation of Aircraft Wake Vortices," AIAA Journal, vol. 16, pp. 956–961, 1978.

[24] Panton, R. L., *Incompressible Flow.* A Wiley-Interscience Publication, 1933.

VITA

Yonghu Wenren was born on June 7, 1962 in Yong An City, Fujian, People's Republic of China. His parents are Fang Wenren and Zhiwei Huang of Shanghai, China. He was graduated from high school in July, 1978. The following September, he entered Zhejiang University, HangZhou, China, and in August, 1982 he graduated with a Bachelor of Science degree in Fluid Mechanics. He entered Northrop University, Los Angles, CA, in September, 1986 and received a Master of Science degree with a major in Aerospace Engineering in August, 1989. He entered University of Tennessee Space Institute, Tullahoma, TN, in August, 1989; meanwhile, he was employed by Flow Analysis, Inc. He received Ph. D. degree in May, 1997 with a major in Aerospace Engineering.

Mr. Wenren was married on August 6, 1986 to Hongwei Li of Fuzhou, China. They have two children, daughter Larissa and son Jesse.