Aerodynamic Flow Simulation Using the Vortex Cloud Method to Predict the Performance of Lifting Bodies

By

Julian D. T. Rimmer

B.Sc. (University of Manitoba) 2001

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Mechanical Engineering

at the

UNIVERSITY OF MANITOBA

Committee in charge:

Dr. R.W. Derksen, Chair
Dr. A. Gumel
Dr. M. Tachie

2006

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

Aerodynamic Flow Simulation Using the Vortex Cloud Method to Predict the Performance of Lifting Bodies

BY

Julian D.T. Rimmer

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

MASTER OF SCIENCE

Abstract

Aerodynamic Flow Simulation Using the Vortex Cloud Method to Predict the Performance of Lifting Bodies

by

Julian D. T. Rimmer

Master of Science in Mechanical Engineering

University of Manitoba

Dr. Robert W. Derksen, Chair

The purpose of this work is to develop and evaluate a vortex cloud method which simulates the flow over airfoil sections at low Reynolds numbers. The current state of the art consists of a good base of modern experimental data, but relies on modeling techniques based on and validated with high Reynolds number flow tools. The method continuously injects elements of vorticity within the flow filed and tracks their movement over time. The convective velocity is determined from the inviscid velocity component due to bodies within the flow field, interaction with other free vortices as well as a random component. The random component is based on the "random walk" model and is scaled to the Reynolds number. An assessment of the vortex cloud model is presented for a range of angles of attack and a range of airfoils along with the implementation of a graphical user interface that allows the user to watch the flow development.

Acknowledgements

I would like to thank my advisor, Rob Derksen, for suggesting the problem and for his advice, support and patience as I worked through it. Thanks to E.H. Price for financial, moral and scheduling support, without which completion of the project would not have been possible. Thanks go to my Dad for reviewing this report and providing excellent feedback as well as consultation on comma use. Thank you Jerry for taking the time to read this report so quickly and for having provided feedback. Finally, to Ainslie, thank you for the tremendous amount of support and patience you have shown towards me over the course of the project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

To date, the development of airfoils suitable for use under a prescribed set of application-specific parameters has been very time consuming and expensive. Prototyping, testing and comparison have been the designer's tools when it comes to airfoil optimization and aerodynamic design. For example, a designer might require an airfoil for use in a compressor with a high duty at a specific angular velocity or for an aircraft that is designed to fly at a given Mach number and altitude. Each of these cases has an optimal airfoil geometry that is specific to the application and, without the use of an automated testing and selection algorithm, it can take years to find an optimal shape, if one is found at all. Progress has been made over the past century in the development of procedures to predict the performance of airfoils under specific test conditions but the inherent complexities in the governing equations of fluid mechanics has proven to be intractable difficult and, at times, overwhelming.

Aircraft design is one of the most expensive and exhaustive design processes known to man. The design of the aerodynamic characteristics of an aircraft is both the first design step to be completed and the most important as the aerodynamic performance is what determines the remaining design details. In today's aerospace industry, computational fluid dynamics (CFD) is regularly used to predict the performance of aerospace structures. It is an unfortunate reality, however, that the procedure for selecting an optimal aerodynamic shape has progressed little from the days when a designer would test a profile in a wind tunnel and manually change the profile until a suitable candidate is found. The designer must still modify the geometry and rerun the solution with the new cross section, this often includes the requirement to generate a unique computational mesh, a time consuming and operator dependant task, for each iteration. In the end, the design is generally considered sufficient if it met the required airflow characteristics with little thought as to how the airfoil could be further optimized. An optimization scheme that considers the application and can select a suitable airfoil with minimal human intervention would be a great asset to the aerospace community.

The recent advancement in computational methods and computer speeds has been a catalyst for optimization schemes for all industries and has a particular relevance to the aerospace industry with the extremely high cost of development and prototyping.

Computational Fluid Dynamics (CFD) today is the product of decades of research that has developed ever-increasing efficiency in solving the

governing equations of fluid mechanics. Recent developments of relatively fast and accurate CFD methods have the capability to simulate viscous flows around complex geometries under a variety of conditions. These Eulerian methods are based on the numerical solution of Navier-Stokes equations and should have no difficulty with flow separation. These packages are of considerable value for the aerospace design community, giving the designer the ability to calculate consistently a variety of conditions more quickly than is possible in the fluid mechanics lab with a wind tunnel. Further progress in the form of coupled solvers, advanced turbulence and boundary models have allowed researchers and industry alike to achieve relevant and accurate solutions to real life problems. The main hindrance to the use of this type of tool for optimization and large scale solutions, particularly when particle tracking is of interest, is that the solvers that are used to analyze the problem are computationally exhaustive and are unsuitable for use in an iterative optimization process. A further limitation to the current methods is that, in general, any result that can be achieved in a reasonable time frame is steady state and gives little information about the formation of eddies or other macroaerodynamic flow characteristics. These macroaerodynamic phenomena play an important role in most simulations, particularly with high Reynolds number flows. If this additional level of resolution is required or particle tracking considered important then the designer must use the much more computationally intensive transient solutions and the complex turbulence models designed for such simulations. In order for these methods

to be of use in an optimization scheme, a reduction in the time required to solve a problem is needed, probably by at least an order of magnitude. An iterative optimization scheme requires two parts: an efficient solution algorithm which discretizes the problem and calculates the flow characteristics and an optimization algorithm which monitors the performance of the airfoils and makes geometric changes in an effort to optimize certain, user specified design parameters.

This work focuses on the first component of optimization to extend the substantial effort to date to find a faster way to predict the performance of an airfoil under certain application parameters. This work looks at extending a modern inviscid vortex panel method, from which aerodynamicists can obtain relatively quick solutions for aerodynamic problems, by implementing a relatively novel, yet relativey simple, numerical method, the vortex cloud model. The developers of this method have suggested that it is suitable to simulate viscous flows over complex geometries and is capable of predicting flow separation. The vortex cloud model comprises a vortex panel method coupled with a Lagrangian particle tracking algorithm. The particles in this case are free stream vortices, which are shed from the surface of the geometry under study. The important element of the vortex cloud method is the ability to model viscous effects by way of a diffusion model, named 'random walk', which prescribes the way that the free stream vortices interact and move through the flow field. This method has been used effectively to demonstrate Brownian motion and is considered useful in modeling viscosity in aerodynamic simulations. This model is advantageous

to this work because of the inherent computation efficiency of solving a largely inviscid problem which has the capability of inferring viscous effects from the movement of the free stream vortices.

The input parameters of the solver include the airfoil geometry and flow application characteristics such as altitude and velocity. The Algorithm also features tunable inputs such as the number of surface panels, the number of free stream vortices and the time step size. Three essential elements characterize the method: first, a method for receiving geometric information and discretizing the aerodynamic flow problem; second, a method for solving the flow regime, including the transport of vortices based on the local flow characteristics; and third a method to model the viscous diffusion of the free stream vortices, thereby introducing the effects of viscosity.

The aerodynamic calculations are performed using a vortex panel method flow solver; the surface vorticity and flow field are used to calculate the velocity of the free stream vortices. In order to model the viscosity, a random perturbation is applied to the particle position, as one would see in viscous flows. The surface vorticity is then recalculated from the flow field and is used to calculate aerodynamic forces. The free stream vortices are plotted at intervals throughout the analysis and give a visual indication of the developing flow field.

Implementation of the vortex cloud method offers the following advantages:

1. Vortex methods consider only the small portion of the flow field where vorticity is present

2. Vortex flows can easily accommodate complex geometries as they do not employ a computational grid
3. In the computation of the velocity field, mass conservation is satisfied directly
4. When vortex methods are used for external flow simulation in unbounded domains, boundary conditions can be applied such that the computational domain is not limited to a finite size.

The aerodynamic simulation code developed in this thesis will be used to produce a number of results. It will be used to demonstrate the effectiveness of the viscous diffusion as well as the effect of the tuning variables on the flow field. It will be used to demonstrate that this method provides a good visual representation of the flow field and will demonstrate the efficiency of the method by selectively omitting traditional solution optimization routines. The current work extends the state of the art by exploring the value of the vortex cloud for modeling low Reynolds number flows. A novel method of shedding vorticity is presented to increase computational stability and similarity to real flows. The vortex cloud method is also examined for its suitability for use as the solution tool for an aerodynamic optimization process.

Chapter 2

State of the Art

Vortex cloud methods have been used over the past 20 years in varying applications. As there has been limited work in the field, some discussion will deal with vortex cloud methods which are not directly applicable to the present work.

2.1 Background

Early efforts at simulating aerodynamic flows were limited by our inability to obtain analytical solutions to the Navier-Stokes equations to all but a few idealized examples. Real world problems required the development of more sophisticated design tools or a simplification of the Navier-Stokes model that would deliver accurate solutions to aerodynamic problems. The first such simplifications came in the form of inviscid models which allowed us to obtain reasonably accurate estimates of lift, pitching moments and induced drag for flow over streamline bodies at high Reynolds numbers. While these methods provide a good estimation of certain parameters, they are critically flawed in

their inability to estimate viscous drag and are, therefore, inadequate for obtaining the comprehensive performance information required to make educated design decisions.

Progress was made by approximating an airfoil as an equivalent flat plate. This approximation led to surprisingly accurate results in estimating minimum drag, a big advancement over previous models. Although, these results were promising, limitations were found when aerodynamicists used the model at higher angles of attack. Viscosity proved to be a continuing issue for airflow simulations and computational aero structure modeling by way of the flow separation observed at high angles of incidence.

The problem of modeling the viscous effects in aerodynamic simulation can be partially addressed through the use of Prandtl's boundary layer theory. Prandtl's boundary layer theory provides a better understanding of the flow characteristics in the near-wall regime. Researchers realized that this model, iteratively coupled with the simulation models of the time, would produce good results for flow over an airfoil. Once the boundary layer flow was solved, it could be used as a boundary condition for the free stream flow domain. Unfortunately, as the boundary layer model provided no consistent or universal method for dealing with flow separation, it gave way to the more sophisticated methods of today, although the procedure is regularly used as the foundation of modern aerodynamic simulation.

2.2 The Panel Method

The underlying solution method used in the current work is a vortex panel method (Martensen, 1959). This type of method has been around for many years with its origins stemming from elementary airflow analysis. Panel methods discretize the geometry under study into discrete line segments, or panels, in order to simplify it to a manageable state. The panel velocity is then used as an approximation of the velocity of the potential flow near the geometry. Pressure distribution and lift can be derived from the surface velocity distribution. While the panel method is less accurate than a full Navier-Stokes solver, it has the key advantage of not requiring a computational mesh. Complex solvers require that the flow field be discretized into volume elements whose properties are governed by the fundamental equations of fluid mechanics and their relationship with their neighboring elements. This group of elements, or mesh, is generated to describe the flow field. When a panel method is used, it is the geometric surface only that is discretized. It has much fewer computational points and is, therefore, faster and more suitable for optimization schemes than a solver employing a fully discretized flow field. The validity of the panel method, as well as the usefulness of the method for computing aerodynamic forces, has been demonstrated (Pfeiffer, 1989).

Kellog (1929) discussed the merits of modeling incompressible inviscid flows by using a boundary integral solution to Laplace's equation, $\nabla^2\phi = 0$, for the

velocity potential. In his book on potential theory, Kellog developed an integral equation representing uniform flow over a body. The method most commonly used by aerodynamicists to solve Laplace's equation is by singularities. Singularities are elementary flows which have infinite velocity at their center, whose flow functions satisfy Laplace's equation and may be superimposed to build flow fields. Kellog's work represents the earliest form of surface singularity model in which the body surface is replaced by a surface singularity distribution. Panel methods solve the flow field by solving the strength of the singularities over discrete sections of the geometry surface, or panels. The most common types of singularities used in these methods are the source, the doublet and the vortex. Generally, the singularity is spread over the panel but is typically solved at a specific point on the panel, the control point.

2.2.1 The Source Panel Method

The two common types of panel methods discussed here are the source panel method and the vortex panel method. The fundamentals of the source panel method can be traced back to the first half of the twentieth century. Source panel methods place a point source on each panel that describes the surface and contributes to the velocity of the potential flow on every other panel. Kellog's integral equation can then be written for each pair of panels in conjunction with the Neumann boundary condition, which states that the velocity normal to the panel surface is zero. The system of equations is then

solved with the source strength describing the potential flow around the geometry. The major limitation to the source panel method is that it is unsuitable for modeling flow around lifting bodies.

2.2.2 The Vortex Panel Method

In contrast with the source panel method, the vortex panel method describes the flow field using surface vorticity. Unlike a source distribution, a surface vorticity distribution provides a direct interpretation of the flow field by a fundamental similarity to the physical problem. In fact, the surface vorticity method is a good representation of the infinite Reynolds number flow of a real fluid. Martensen (1959) extended his boundary integral theory for airfoil cascades. Martensen represented the viscous boundary layer by a distribution of vorticity adjacent to the body's surface. The Dirichlet boundary condition is satisfied by imposing a zero tangential velocity on the panel surface; this also leads to a direct calculation of the surface vorticity and a solution of the potential flow field.

Jacob and Reigels (1963) first implemented this theory for numerical modeling with a computer, although several issues needed to be resolved before the method would garner acceptable solutions to physical problems. Wilkinson (1967) did much work to alleviate computational problems and his work has also led to the use of this method for modeling of airfoil cascades (Wilkinson, 1969). Further development of Martensen's method, done by

Nyiri and Baranyi (1983), among others, led to the surface vorticity method being a useful predictive tool for aerodynamic simulation.

Development of the vortex panel method continues today with recent investigations into the treatment of the trailing edge. Xu (1998) has demonstrated that the addition of a control point downstream of the trailing edge can help to deal with the different air velocities on the pressure and suction sides of an airfoil joining at a stagnation point.

Where difficulties arise is the surface vorticity method's inability to model boundary layer separation and for these problems a novel approach has been proposed in the vortex cloud model. A good summary of panel methods and their applications is available through two recent reviews by Hess (1990a & 1990b), the survey by Erickson (1990) and the book by Katz and Plotkin (1991).

2.3 Vortex Cloud Methods

Separated flows and wake analysis pose serious problems to traditional surface vorticity methods. Vortex cloud analysis attempts to model these phenomena by discretizing the vorticity inherent to them into small discrete vortex elements and observing the motion of these elements through the flow field. Often referred to as 'discrete vortex models', vortex cloud models use a Lagrangian method to track the vortex elements as they interact with the body surface as well as surrounding elements.

2.3.1 Vortex Dynamics and Vortex Clouds

Vortex dynamics have been studied by several groups but the first serious attempt at discrete vortex modeling can be attributed to Rosenhead (1931) who studied the Kelvin-Helmholtz instability of vortex sheets. Abernathy and Kronauer (1962) extended Rosenhead's discrete vortex model to demonstrate a progressive formation of a von Karman type vortex street as the outcome of an unstable sheet vortex configuration.

The circular cylinder has played a large role in the study of vortex dynamics. Gerrard (1967) used the reflection system, where a vortex, equal in strength and opposite in sign, is located inside the body surface, in one of the first attempts at simulating a flow field using vortex dynamics for the solution of the potential flow field due to a point vortex near a cylinder.

The vortex cloud model is compatible with the surface vorticity model and, arguably, it is a natural extension of it. The governing equations for the transport of the discrete vortices are analogous to the equations that solve the potential flow. Chorin (1973) suggested that, while the vortex cloud method had come under some criticism (Takami, 1964) and Moore (1971), the large errors observed were due to a fundamental flaw in the approximation of a vortex sheet or continuous vorticity by a point vortex with an unbounded potential flow. Chorin suggested that by smoothing the analysis, the point vortex potential flow becomes bounded and delivers useful results.

Recent work by Kuwahara (1973) and Sarpkaya (1975) employs this model to study separation around a flat plate set obliquely to a uniform flow. Clements (1973) explored the similar problem of vortex street development behind a rectangular body with a flat trailing face. One observes from looking at these works that Kuwahara (1973), Clements (1973) and Sarpkaya (1975), along with Katz (1981), and Cortelezzi *et al.* (1997), have chosen to model the continual shedding of vorticity from a sharp edge by introducing a new vortex element into the flow at the beginning of each time step. The strength, position, and velocity of the shed vortex elements are chosen, more or less arbitrarily, to satisfy the author's version of the unsteady Kutta condition. In the work of Krasny (1991) and Nitsche and Krasny (1994) the vortex elements are released at the edge of the body instead of being placed into the flow at some arbitrary location. As a result their comparison with the experimental work of Didden (1979), as presented in Nitsche and Krasny (1994), is promising. Jones (2003) provides a solution to the unsteady Kutta condition by using an alternate form of the conditions associated with the imposition of the steady Kutta condition in his study of flow around a moving flat plate.

The applications for vortex cloud models are quite broad. The work by Chan *et al.* (2000) investigated the behaviour of a premixed turbulent flame. The study used the vortex cloud model to solve the turbulent flow field and has shown that the computed field and the turbulence scalar statistics are in good agreement with the experimental results. Nakashima and Ono (2000) used the vortex cloud model to predict the thrust, energy consumption, and

propulsive efficiency of fish and cetaceans in water and found that the thrust decreases due to the increase in the lift force as the normalized propulsive speed increases when all the joints move in phase.

Vortex cloud methods have been a useful and predictive tool for aerodynamicists, providing insight into the evolution of jet and wake flows. There are limitations, however, in the inviscid approximation employed by the method. In many flows, viscous effects are responsible for the generation of vorticity at the body boundaries and an approximation of viscous effects, including diffusion, is necessary.

2.3.2 Hybrid Vortex Cloud Methods

Viscous effects have been modeled by fluid dynamicists for years in finite difference and finite element discretized solution methods. These methods typically solve the viscous Navier-Stokes equations using numerical approximations for the viscous terms. Traditionally, these have taken the shape of time-averaged Reynolds Average Navier Stokes (RANS) or Shear Stress Transport (SST) models. Recently, more computationally demanding schemes such as Large Eddy Simulation (LES) or Detached Eddy Simulation (DES) methodologies have been available (Frohlich and Rodi 2002).

In an effort to model the viscous effects of the boundary layer and, specifically, separation, there has been a significant effort to combine the efficient, inviscid methodology of the vortex cloud method with viscous Navier-Stokes solvers. This typically employs a grid in the near wall region

for viscous analysis coupled with a grid-free analysis in the unbounded flow field. One of the first successful attempts at this implementation was completed by Spalart *et al.* (1983) in the study of various flow problems, most of which had massive separation. Spalart's results, when compared with experimental results, demonstrate the reliability and the general accuracy of the hybrid method, with little dependence on empirical parameters. In Spalart's scheme, the vortex flow determines the instantaneous pressure distribution, while the boundary layer flow determines the separation points.

Many of the complex features of the flow past a circular cylinder, over a wide range of Reynolds numbers, are correctly reproduced. Spalart *et al.* (1983) also considered flow over and airfoil in dynamic stall; this represents the first application of the vortex cloud method to simulate flows over moving geometry. A NACA 0012 airfoil as studied, oscillating in pitch about the quarter chord position from 5° to 25° with Re = 2.6×10^6. Results were in good agreement with experimental data and predicted the phenomenon of lift coefficients being significantly higher than their steady state values at equivalent angle of attack, as seen in experiment (Carr *et al.* 1977).

The first blending of finite-difference and vortex methods was presented by Chen *et al.* (2002)

There has been significant development of a vortex in cell method where the vortex cloud method is used in conjunction with a computational grid which discretizes the entire flow field for the sake of a direct numerical solution of

viscous diffusion. This is outside of the scope of this discussion but a detailed account of the method can be found by Barber and Fonty (2005)

2.3.3 Viscous Diffusion in Vortex Clouds

The current work focuses on methods that are suitable for optimization schemes and, while these models have been used to varying levels of success, their demands on computational resources to present has been prohibitive. The design of a particle scheme to handle viscous effects that would directly model turbulence is compelled by the desire to maintain the Lagrangian character of the vortex cloud method.

In a transient or time-stepping scheme, a natural approach is to consider the inviscid and viscous parts of the equations separately. This is known as viscous splitting and is rooted in classical fluid mechanics. Viscous splitting for vortex methods was pioneered by Chorin (1973). Chorin developed an algorithm that split time steps into sub steps where:

1. Vortices move with local velocity – the inviscid portion
2. A diffusion algorithm is applied to the free vortices – the viscous portion

In the method introduced, known as random walk, particles undergo a Brownian-like motion and so simulate the effects of diffusion for high Reynolds number flows. The crux of the popular method is to subject all of the free vortex elements to a small perturbation of location, or small shift in their position. This produces a diffusion-like behaviour of the fluid under study and has been shown to emulate viscous effects and calculate drag

coefficients, for a high Reynolds number flow over a circular cylinder, which are in good agreement with experimental data (Chorin, 1973). Chorin (1978) extended his work to facilitate integration into hybrid algorithms and to improve on the generation of vorticity at boundaries. Chorin was aware that, for the random walk method to be useful as an alternative to Eulerian grid-type Navier Stokes solvers, it would have to be able to model the viscous boundary layer, while being suitable for larger scale phenomena such as von Karman vortex street development in downstream wakes.

Porthouse and Lewis (1981) developed a random walk technique as a natural extension of the vortex cloud modeling that was nearly identical to that of Chorin. The use of the vortex cloud method with random walk is often referred to as the 'random vortex method'. While Chorin's methodology has been widely adapted for modeling viscous flows in combination with vortex cloud methods, there was some skepticism about the validity of using viscous splitting to compute adequately the solutions of the Navier Stokes equations. Beale and Majda (1981) provide a rigorous proof that viscous splitting algorithms, which are the underlying design principle for the random vortex method, converge to solutions of the Navier Stokes equations at a rate which improves as the viscosity becomes smaller.

Lewis (1991) did extensive work on the application of the random vortex method to modeling flow fields around lifting bodies. In his thorough discussion on random vortex methods, Lewis presents applications to bluff body flows as well as to lifting bodies such as airfoils and cascades, Lewis addresses the issue raised by Porthouse and Lewis (1981) and Spalart *et al.*

(1983) and deals with the seeming inability of the random vortex method to cope with boundary layer stability and the appearance of premature stall. Lewis develops a hybrid model that stabilizes the upper surface of the airfoil by enforcing potential flow and the lower surface being represented by a full random vortex cloud model. Predicted flow pattern and lift and drag data were presented for a NACA 0012 airfoil at 5° angle of incidence and $Re = 10^{6}$ and are compared to experimental results by Miley (1982). The predicted lift coefficient is in reasonable agreement with the data; the drag coefficient is, not unexpectedly, under predicted due to the virtual elimination of form drag and associated separations. The predicted surface pressure distribution is in very good agreement with a potential flow surface vorticity solution. Lewis' seminal work also describes vortex cloud methods from first principles to deal with shear layers, boundary layers, periodic wakes, bluff-body flows, and cascades. Due to the sophistication of Lewis' models, the current work will be fundamentally based on his airfoil simulation structure.

Borthwick and Barber (1992) describe a Biot-Savart vortex cloud model for simulating the flow patterns which occur when a single high-velocity inflow jet is used to stir the fluid within a circular container. Borthwick mapped the circular perimeter of the container onto a rectangle by means of a Schwarz-Christoffel transformation. A potential flow solution is then obtained and this is transformed to give the potential flow inside the circle. Discrete vortices are added at the inlet of the physical system in order to model the inflow shear layers. Velocity components resulting from the discrete vortices and their images in the walls of the cylinder are superimposed on the uniform

potential flow solution. Viscous effects are incorporated through the use of the random walk method. From the results it is shown that the discrete vortex method does predict qualitatively the important features of jet-forced reservoir flow.

Pereira (2003) presented an implementation and initial test of a sub-grid scale model coupled with the random vortex cloud method under the same test conditions as Lewis. Pereira demonstrated that the random vortex cloud method with turbulence modeling can improve results obtained by use of the vortex cloud method alone although the cost of the turbulence modeling for marginal benefit still needs to be evaluated.

Vamos *et al.* (2003), in an effort to increase the computation speed and to reduce the required memory of random vortex cloud methods, derived a 'global random walk' method in which the free vortices at a given site are simultaneously scattered following the binomial Bernoulli repartition. Vamos found that the computation time is reduced three orders of magnitude with respect to individual random walk methods, while obtaining good simulation of transport in unbounded domains, using normal size grids.

2.3.4 Other Modeling Considerations

As vortex particles are allowed to freely convect, 'gaps' may appear in areas where the flow strain is large and the model loses resolution of the flow field as a result. One solution to this issue is the application of a 'remeshing' or 'redistribution' scheme, where the free vortices are periodically re-located

onto a regular lattice or grid, and their circulation weights are interpolated to the new locations.

The addition of remeshing to vortex cloud methods made long-time unsteady calculations possible (Koumoutsakos, 1993) and has been embraced by various investigators who have produced remarkable results (Cottet *et al.*, 1999) and (Ploumhans and Winckelmans, 2000). Conversely, it can be argued that the grid-free nature of vortex methods is undermined by the requirements of remeshing.

In an effort to decrease the convergence time of the random vortex cloud method, Barba *et al.* (2005) introduced an alternate method of modeling viscous effects, known as the core spreading method. In this method, the vortex core size is adjusted in the spatial adaption. Numerical experiments demonstrate considerable increase in accuracy in comparison with standard remeshing schemes used with vortex methods.

An alternate method of modeling the viscosity in a vortex cloud method is one where the vortex strength is adjusted according to a viscous relation. There has been little work extending on this thought and it, therefore, will not be examined.

Chapter 3

Problem Formulation and Results

The vortex cloud solver contains three elements. A vortex panel method receives the geometrical information and computes the surface vorticity. The surface vorticity is used to shed free vortices into the flow field where the convection algorithm computes their movement through the computational domain. A graphical user interface was developed to allow the user to input simulation parameters easily as well as to watch the developing flow field.

3.1 Vortex Panel Method

An inviscid potential flow solver, Martensen's surface vorticity method, is used to model the flow over the simulation geometry. There is a large body of work in the area of panel method theory and its application to aerodynamic simulation is well understood. An extended discussion of the

theoretical background of panel methods can be found in many aerospace and fluid mechanics textbooks and therefore only a brief summary of the relevant theory and equations is given here. Special consideration will be given to unique algorithms developed in the course of the present work.

3.1.1 The Vortex Panel Method – Theoretical Background

In all real flows around a body, there is a boundary viscous shear layer, the boundary layer, adjacent to the body surface. Outside of this layer, the fluid will have a finite velocity, v_s. It is the friction between the surface and the airflow that causes the velocity parallel to the body surface to be zero; this is known as the Dirichlet boundary condition. The large velocity gradient in the highly viscous layer causes substantial vorticity. As viscosity approaches zero, and the Reynolds number approaches infinity, the layer becomes infinitely thin and becomes a vortex sheet. It is therefore analogous and physically representative of the airflow to model this layer by replacing the body surface with a vortex sheet, $\gamma(s)$, where $\gamma(s)$ is the strength per unit length at location s on the body surface. If we apply the Dirichlet boundary condition to the problem to ensure zero velocity on the surface, we can say that the velocity immediately outside of the viscous layer is equal to the surface vorticity at that point.

$$v_s = \gamma(s) \tag{3.1}$$

We can also infer that the velocity at which the surface vorticity is convecting downstream to satisfy Dirichlet is given by:

$$v_{conv} = \frac{1}{2}\gamma(s) \tag{3.2}$$

For the two-dimensional body in Figure 1, the velocity dq_{mn} induced at m by incremental vortex element n of strength $\Gamma_n = \gamma(s_n)ds_n$ at location s_n is given by the Biot-Savart law in a simplified form, namely:

$$dq_{mn} = \frac{\gamma(s_n)ds_n}{2\pi r_{mn}} \tag{3.3}$$

Where r_{mn} is the straight line distance from point n to point m.

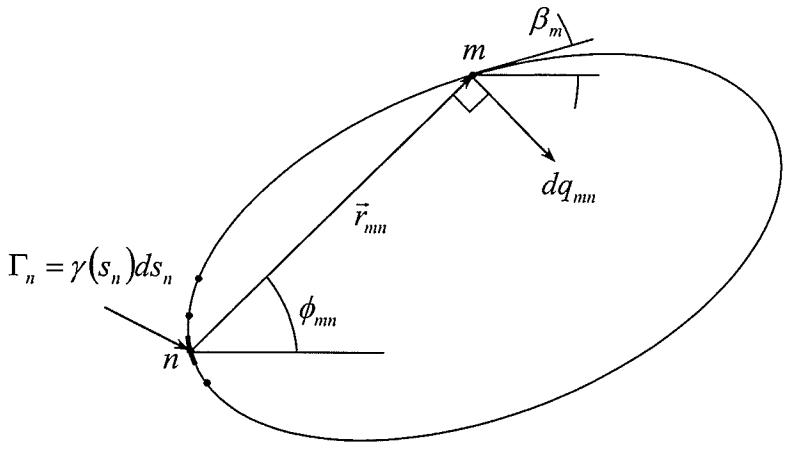


Figure 1) Velocity induced by vortex element at s_n

For computational simplicity, it is beneficial to write the x and y components of velocity dq_{mn}:

$$du_{mn} = \frac{\gamma(s_n)ds_n}{2\pi r_{mn}}\sin\phi_{mn} = (y_m - y_n)\frac{\gamma(s_n)ds_n}{2\pi r_{mn}^2} \tag{3.4}$$

$$dv_{mn} = \frac{\gamma(s_n)ds_n}{2\pi r_{mn}} \cos\phi_{mn} = (x_m - x_n)\frac{\gamma(s_n)ds_n}{2\pi r_{mn}^2} \tag{3.5}$$

The induced velocity dv_{smn} is given by:

$$dv_{smn} = \frac{1}{2\pi}\left[\frac{(y_m - y_n)\cos\beta_m - (x_m - x_n)\sin\beta_m}{(x_m - x_n)^2 + (y_m - y_n)^2}\right]\gamma(s_n)ds_n$$

The Dirichlet boundary condition can then be satisfied by Martensen's boundary integral equation for plane two dimensional flow (Lewis, 1991):

$$-\frac{1}{2}\gamma(s_n) + \oint k(s_m, s_n)\gamma(s_n)ds_n + W_\infty(\cos\alpha\cos\beta_m + \sin\alpha\sin\beta_m) = 0 \tag{3.6}$$

Where $-\frac{1}{2}\gamma(s_n)$ is the self induced vorticity at n and represents the surface velocity for non-convected surface vorticity and $W_\infty(\cos\alpha\cos\beta_m + \sin\alpha\sin\beta_m)$ is the free stream velocity, parallel to the surface at s_n $k(s_m, s_n)$ is the coupling coefficient and is given by:

$$k(s_m, s_n) = \frac{1}{2\pi}\left[\frac{(y_m - y_n)\cos\beta_m - (x_m - x_n)\sin\beta_m}{(x_m - x_n)^2 + (y_m - y_n)^2}\right] \tag{3.7}$$

The body geometry is divided into a finite number of discrete elements of vorticity, or panels, of length the center of which is known as the control or pivotal point (x_n, y_n) as seen in Figure 2.

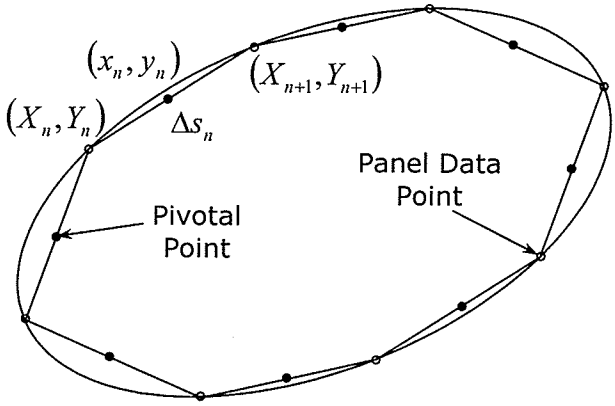


Figure 2) Discretization of a body surface

In Figure 2, the body geometry is represented by a series of short straight lines of length:

$$\Delta s_n = \sqrt{(X_{n+1} - X_n)^2 + (Y_{n+1} - Y_n)^2} \tag{3.8}$$

The pivotal points, traditionally located at the center of each panel are given by:

$$\left.\begin{aligned} x_n &= \frac{1}{2}(X_n + X_{n+1}) \\ y_n &= \frac{1}{2}(Y_n + Y_{n+1}) \end{aligned}\right\} \tag{3.9}$$

If, for simplicity, we assume that the self-induced vorticity is zero, integrating (3.6) and rearranging we arrive at:

$$\sum_{n=1}^{M} K(s_m, s_n)\gamma(s_n) = -U_\infty \cos\beta_m - V_\infty \sin\beta_m \tag{3.10}$$

Equation (3.10) is the fundamental equation of the vortex panel method. We expand (3.10) to develop a system of equations with M equations and

M unknowns of surface vorticity $\gamma(s_n)$ by writing it for each of the pivotal points. $K(s_m, s_n)$ is known as the coupling coefficient and is expressed by:

$$K(s_m, s_n) = \frac{\Delta s_n}{2\pi}\left[\frac{(y_m - y_n)\cos\beta_m - (x_m - x_n)\sin\beta_m}{(x_m - x_n)^2 + (y_m - y_n)^2}\right] \qquad (3.11)$$

Representation of the surface by a series of straight line elements causes the self-induced surface vorticity on m to be zero. While this is a reasonable approximation, Lewis (1991) has developed an approximation for the self inducing velocity due to surface curvature. As a result:

$$K(s_m, s_m) = -\frac{1}{2} + K'(s_m, s_m) = -\frac{1}{2} - \frac{\Delta\beta_m}{4\pi} = -\frac{1}{2} - \frac{1}{8\pi}(\beta_{m+1} + \beta_{m-1}) \qquad (3.12)$$

Where $\frac{1}{2}(\beta_{m+1} + \beta_{m-1})$ is considered to be a reasonable estimate of the change in slope $\Delta\beta$ on m.

3.1.2 The Vortex Panel Method – Implementation

The theory presented in section 3.1.1 was used in development of the current work. To this point, there are five major sections to the program as outlined in Figure 3:

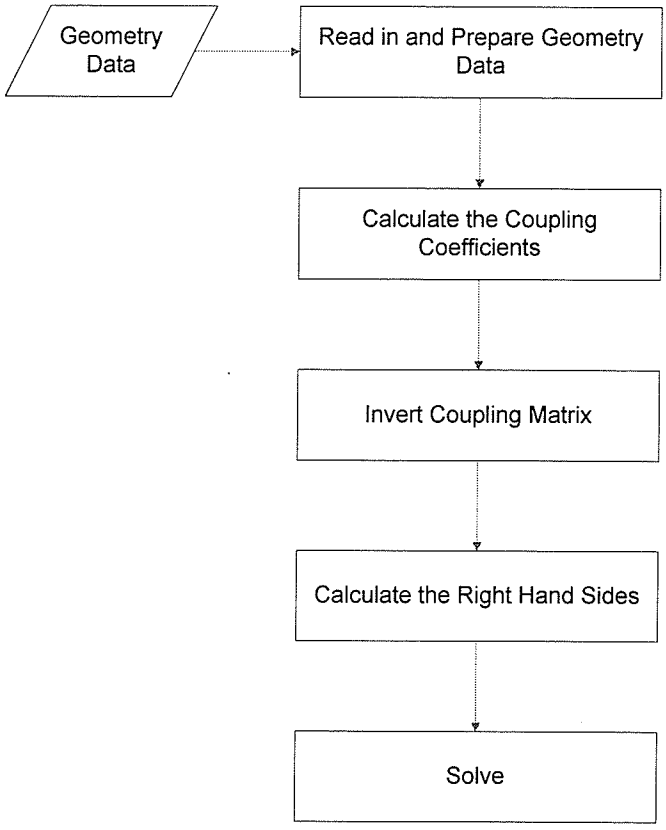


Figure 3) Flow chart of a vortex panel program

3.1.2.1 Input Data and Data Preparation

Data is read from a geometry file which describes the body surface in a series of data coordinates moving clockwise from the leading edge. An example data set, describing the surface of a circular cylinder is shown in Table 1.

Whether the number of data points is odd or even is not important, so long as they are sufficient in number to satisfy the user's required accuracy in describing the body surface. The final point, however, should be equal to the first point to ensure that the body is closed. The beginning of the data file contains information describing the flow field. The angle of attack, free

stream velocity, number of time steps and the number of surface panels, which must be even, are all obtained from the data file. This was done to speed up the development process and a more user-friendly manner of inputting data was developed, discussed in section 3.3.

X	Y
0.000000	0.000000
0.048943	0.309017
0.190983	0.587785
0.412214	0.809017
0.690982	0.951056
1.000000	1.000000
1.309015	0.951056
1.587784	0.809017
1.809016	0.587785
1.951056	0.309017
2.000000	0.000000
1.951058	-0.309017
1.809016	-0.587785
1.587784	-0.809017
1.309015	-0.951056
1.000000	-1.000000
0.690982	-0.951056
0.412214	-0.809017
0.190983	-0.587785
0.048943	-0.309017
0.000000	0.000000

Table 1) Data set for a circular cylinder

This data is then prepared for the simulation. This preparation includes:

i. Calculation of the panel end- and pivotal-points

ii. Calculation of the sine and cosine of the panel slope as well as the panel length

iii. Calculation of the change in slope for each panel

The panel end points are derived from the input data set and are spaced around the surface of the body according to a cosine distribution:

$$(x_n) = \frac{c}{2} + c\cos(\pi + nd\theta) \tag{3.13}$$

Where c is the chord length, $d\theta = \frac{2\pi}{N}$ and the $\frac{c}{2}$ term ensures that the simulation profile starts at $(0,0)$. This equation will concentrate panels, for elliptical horizontal profiles, near the leading and trailing edge where it is desirable to have increased resolution. The code implementation of this, for an arbitrary number of panels, `pDoc->i_npanels` is:

```
DTHETA = (2*pDoc->m_PI)/pDoc->i_npanels;

while ((THETA/DTHETA)<pDoc->i_npanels)
{

      pDoc->fl_points[X][D_X] = (pDoc->fl_chordl/2) +
            ((pDoc->fl_chordl/2) * cos(pDoc->m_PI+THETA));

      THETA = THETA + DTHETA;


      if (fabs(pDoc->fl_points[X][D_X])<MINVALUE)
            pDoc->fl_points[X][D_X]=0;

      X++;
}
```

The code segment here determines the location of the points on the x-axis only. The y-values are determined by using a linear interpolation scheme on the original profile data and the newly computed x-values:

```
for (FN=0; FN<pDoc->m_NFILE; FX++)
{
      for (N=0; N<pDoc->i_npanels; X++)
      {

            XA=pDoc->fl_panelpts[FN][D_X]; // original data
            YA=pDoc->fl_panelpts[FN][D_Y]; // original data

            if (FX==pDoc->m_NFILE)
            {

                  XB=pDoc->fl_panelpts[1][D_X];
```

```
                        YB=pDoc->fl_panelpts[1][D_Y];

                    }

                    else
                    {

                        XB=pDoc->fl_panelpts[FN+1][D_X];
                        YB=pDoc->fl_panelpts[FN+1][D_Y];

                    }

                    if (N<=(pDoc->i_npanels/2))
                    {

                        if (pDoc->fl_points[N][D_X] > XA &&
                            pDoc->fl_points[N][D_X] < XB)
                        {

                            XI=pDoc->fl_points[N][D_X];
                            pDoc->fl_points[N][D_Y] =
                                Inter(XA, XB, YA, YB, XI);

                        }
                    }

                    else
                    {

                        if (pDoc->fl_points[N][D_X] < XA &&
                            pDoc->fl_points[N][D_X] > XB)
                        {

                            XI=pDoc->fl_points[N][D_X];
                            pDoc->fl_points[N][D_Y] =
                                Inter(XA, XB, YA, YB, XI);

                        }
                    }
                }
            }COUNT++;
```

This routine searches for the computed x-values of the panels `pDoc->fl_points[N][D_X]`, which falls between two consecutive points from the profile geometry, `pDoc->fl_panelpts[FN][D_X]` and `pDoc->fl_panelpts[FN+1][D_X]`, and then interpolates the y-value of the panel end point using the function `Inter()`. This procedure is iterated a few times to ensure that

all of the panel points are captured correctly. This routine is only required to be called once in the simulation and ensuring that the points are computed correctly in the initial set up is well worth the minor cost in processing time. The pivotal points are calculated as the average of the panel end points:

```
X1=pDoc->fl_points[N][D_X];
Y1=pDoc->fl_points[N][D_Y];
X2=pDoc->fl_points[N+1][D_X];
Y2=pDoc->fl_points[N+1][D_Y];
if (N==pDoc->i_npanels-1)
{
      X2=pDoc->fl_points[0][D_X];
      Y2=pDoc->fl_points[0][D_Y];
}
pDoc->fl_points[N][D_X]=(X1+X2)/2;
pDoc->fl_points[N][D_Y]=(Y1+Y2)/2;
```

Finally, the change in slope, $\Delta\beta$, on each panel was calculated according to (3.12).

3.1.2.2 Coupling Coefficients

The coupling coefficients $K(s_m, s_n)$ and $K(s_m, s_m)$ were calculated according to (3.11) and (3.12), respectively.

3.1.2.3 Matrix Inversion

The program employs a Gauss-Jordan inversion scheme. The program Invert_Test() was written and tested by importing a matrix into the program, inverting it and then crossing it with the original. The identity matrix was found to be the cross product for all test cases ranging in size from a 3 x 3 to a 10 x 10.

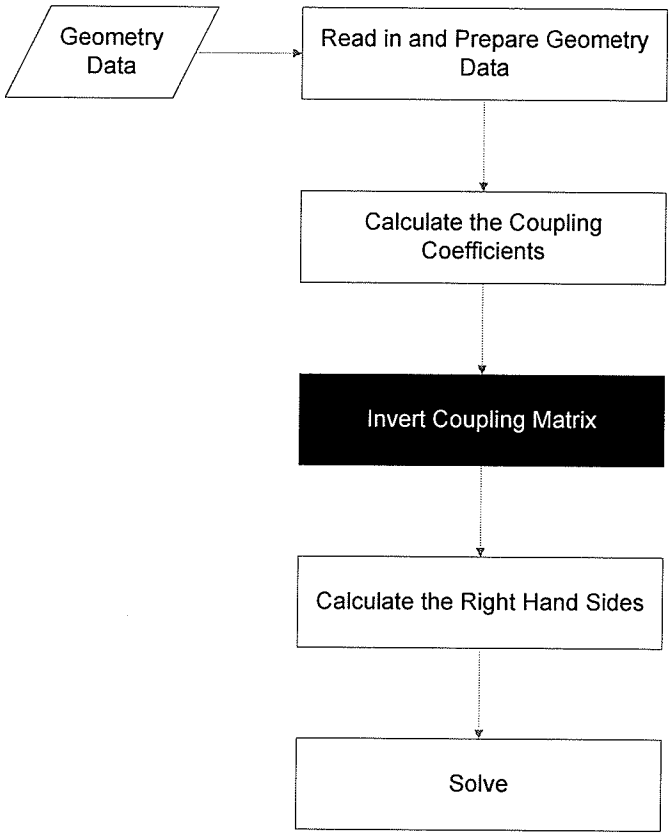


Figure 4) Invert coupling matrix was developed as a standalone program Invert_Test()

3.1.2.4 Right Hand Sides

The calculation of the right hand sides is a direct calculation of (3.10).

```
pDoc->fl_uinf = m_wInf * cos(m_alpha);
pDoc->fl_vinf = m_wInf * sin(m_alpha);

TEMP1 = -1 * pDoc->fl_cosine[S] * (pDoc->fl_uinf);
TEMP2 = -1 * pDoc->fl_sine[S] * (pDoc->fl_vinf);

pDoc->fl_rhs[S] = TEMP1 + TEMP2;
```

3.1.2.5 Solver

The system of equations (3.6) is solved by multiplying the inverted matrix by the right hand side vector, `pDoc->fl_rhs`. The resulting vector is the normalized surface vorticity which, when multiplied by the panel length gives the surface velocity.

3.1.2.6 Validation

A validation of the vortex panel program Flow() was done using a circle as the test profile. The geometry was positioned such that the leading edge was located on the origin. The surface was represented by 18 panels and follows the profile expressed by:

$$x = a(1 - \cos\phi)$$
$$y = a\sin\phi$$

Figure 5 shows the position of the circular profile in the domain as well as the calculated panel end and pivotal points. From the figure it is evident that the pivotal points calculated by the data preparation routine lie on the surface of the body and are evenly distributed on the surface, as would be expected of a cosine distribution on a circular profile.

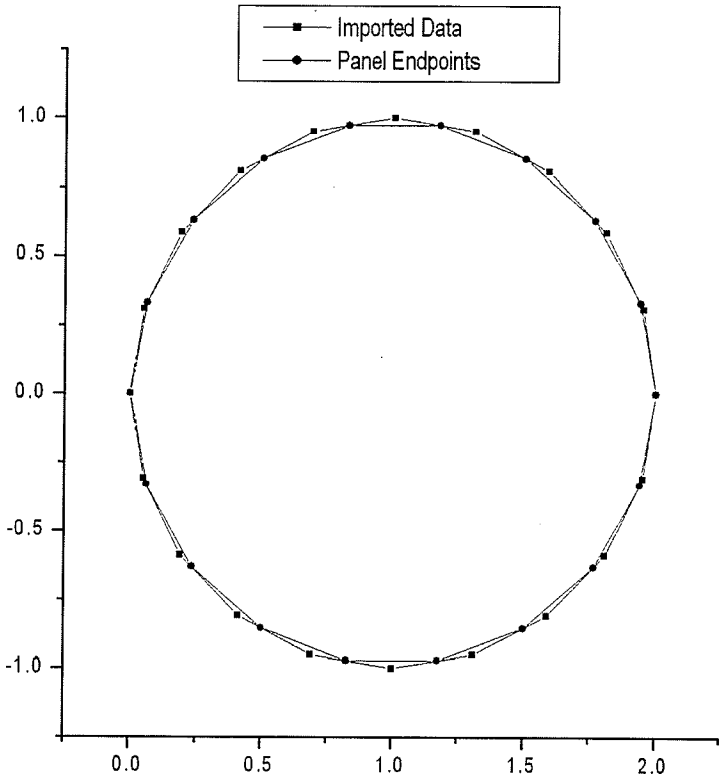


Figure 5) Comparison of data set and computed panel endpoints

Batchelor (1970) states the exact solution for the surface velocity due to a uniform stream U_∞ as:

$$\upsilon_s = 2U_\infty \sin\phi$$

The surface velocity is compared for a circle of radius $a = 1$, with its diameter along the line y=0 shown in Figure 5. Starting at $(0,0)$, the panels are numbered clockwise around the circle. Very good agreement is noticed in Table 2 between Batchelor's exact solution and the output from the Flow() program, indicating that the program is a useful tool for predicting surface velocity. The percent relative error was found by subtracting the calculated

velocity from the exact solution and then dividing by the exact solution for each panel.

PANEL	EXACT	Visiflow	% Error (Rel.)
1	0.347296	0.349116	0.52%
2	1.000000	1.00095	0.10%
3	1.532089	1.53817	0.40%
4	1.879385	1.88944	0.54%
5	2.000000	2.01155	0.58%
6	1.879385	1.88947	0.54%
7	1.532089	1.53825	0.40%
8	1.000000	1.00116	0.12%
9	0.347296	0.347868	0.16%
10	-0.347296	-0.347868	0.16%
11	-1.000000	-1.00117	0.12%
12	-1.532089	-1.53824	0.40%
13	-1.879385	-1.88947	0.54%
14	-2.000000	-2.01155	0.58%
15	-1.879385	-1.88944	0.54%
16	-1.532089	-1.53817	0.40%
17	-1.000000	-1.00094	0.09%
18	-0.347296	-0.349115	0.52%

Table 2) Comparison of surface velocity predicted by Flow() and the exact solution

3.1.3 Vortex Panel Method – Considerations for Airfoils

To this point, a vortex panel method has been developed which is applicable to arbitrary geometric profiles. In order to apply the vortex panel method to airfoils, there are some additional considerations that must be addressed. Airfoils generate lift and are generally quite thin. These can lead to computational errors due to the proximity of opposing panels.

The airfoil used for the majority of the code development was 12 percent thick NACA 0012 airfoil. Geometric data was obtained from the work of Selig *et al.* (1996). Figure 6 is the airfoil profile for a NACA 0012 section.

Figure 6) Airfoil profile for a NACA 0012 section

In order to examine lifting flows over a surface we apply the *Kutta-Joukowski theorem* which states that the lift per unit span on a two-dimensional body is directly proportional to the circulation around the body, or:

$$L' = \rho_\infty V_\infty \Gamma \tag{3.14}$$

We can determine the bound circulation by taking the line integral over the body surface:

$$\Gamma = \oint \gamma(s) ds = \sum_{n=1}^{N} \gamma(s_n) \Delta s_n \tag{3.15}$$

Implementation of (3.13) has the unfortunate effect of increasing the number of unknowns to N+1 with only N independent equations. Exploration of methods to deal with this complexity will be discussed in section 3.2.1.2.

3.2.1.1 Back Diagonal Correction

From (3.11) it is noted that as $(x_m - x_n)^2 + (y_m - y_n)^2$ approaches 0, $K(s_m, s_n) \to \infty$. For thin profiles, such as airfoils, the panel n, on the opposite side of m, that will have the largest coupling coefficient, $K(s_m, s_n)$, will be the panel N-m+1, and therefore, the largest contribution to the surface vorticity.

The coupling coefficients for panel 9 of a NACA 0012 airfoil represented by 28 panels are shown in Figure 7:

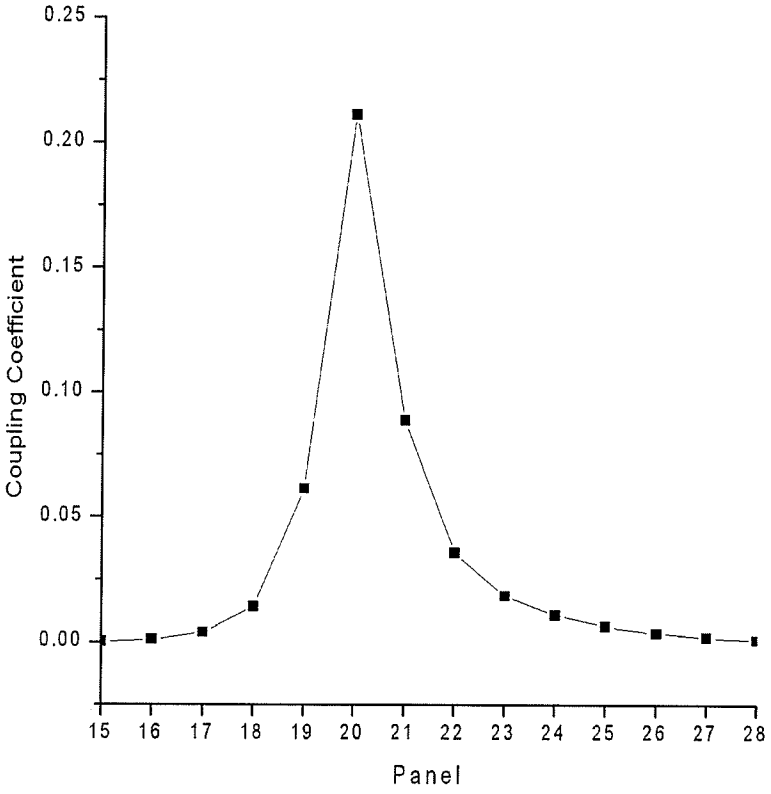


Figure 7) Coupling coefficients for panel 9 of 28

The considerably larger value of the coupling coefficient for the 20th panel is evident and corresponds with the N-m +1 location, i.e. 28-9+1=20.

Jacob and Riegels (1963) recommended a treatment to correct this disproportionate influence by observing that, according to Kelvin's theorem, the net circulation around the profile interior due to the surface vorticity of

element m, $\gamma(s_m)\Delta s_m$, should be zero. Implementation of this observation takes the form of:

$$K(s_{opp}, s_m) = -\frac{1}{\Delta s_{opp}} \sum_{\substack{n=1 \\ n \neq opp}}^{N} K(s_n, s_m)\Delta s_n \qquad (3.16)$$

where $opp = M - n + 1$. Equation (3.14) sets the opp element's coupling coefficient to the negative sum of the other elements, for a panel n. It is interesting to note that as n increases in value from 1 to M, opp decreases in value from M to 1. This allows a computational simplification in that only the back diagonal of the coupling coefficient matrix is affected.

Once the back diagonal correction has been applied, the matrix has the characteristic of any one equation being the minus of the sum of the other equations and is therefore singular and non-invertable. This difficulty will be addressed in the following section.

3.2.1.2 The Kutta Condition

In order to increase computation accuracy at the trailing edge, Wilkinson (1967) suggested that the implementation of the Kutta condition should take the form of an assurance that the static pressure, and therefore the surface vorticity, at the two trailing edge segments on the upper and lower surfaces should have the same magnitude. This is as opposed to ensuring a stagnation point at the trailing edge implemented by a prescribed bound

circulation. In order to ensure that the surface pressure approaches an equal and opposite surface vorticity, the following constraint is imposed:

$$\gamma(s_{te}) = -\gamma(s_{te+1}) \tag{3.17}$$

The surface vorticity on the lower surface is negative in order to ensure smooth flow leaving the trailing edge.

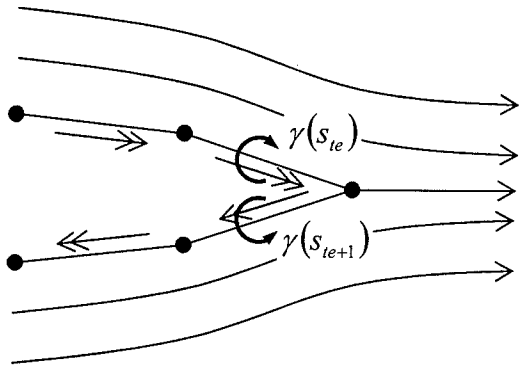


Figure 8) Trailing edge flow

Wilkinson also noted that (3.15) eliminates one of the unknowns from the system of equations given by (3.10) and if (3.10) is written for an element n, we arrive at:

$$K(n,\ 1)\gamma(s_1) + \ldots + \left(K(n,\ te) - K(n,\ te+1)\right)\gamma(s_{te}) + \ldots + K(n,\ N)\gamma(s_N) = rhs_n \tag{3.18}$$

where columns te and $te+1$ have been combined.

As the number of columns and unknown vorticity values has been reduced by one, it is essential that the equations are reduced by an equivalent number. There are three logical options that are considered for accomplishing this:

1. Simply eliminate the superfluous $(te+1)^{th}$ equation
2. Add the two equations that describe the trailing edge elements

3. Subtract these equations instead

Lewis (1991) suggests that the method of reducing the number of equations by 1 that provides the best results is to subtract the two equations that represent the trailing edge elements. The logic in this can be seen in Figure 8, where the induced velocity on the inner surface is represented by the double arrows and flows positively clockwise around the profile. The effect of the equation subtraction is the equivalent of determining the average downstream flow in the region of the trailing edge.

The implementation of this requires that the $(te+1)^{th}$ coupling coefficient is subtracted form that of the te^{th} element and that the right hand side vector is reduced in size accordingly. Whereas the matrix created during the back diagonal correction was singular, the current procedure eliminates one of the equations and the matrix ceases to be ill-conditioned.

For programming simplification, the implementation of the Kutta condition was split into two parts, one for the coupling coefficients and one for the right hand sides. This allows the coupling coefficients to be treated completely independently, an advantage that will become clear when we discuss the vortex cloud method

3.2.1.4 Solution

There are some additional considerations for the solution of the flow field and induced body forces when airfoils are considered. The surface pressure distribution is given by

$$C_p = 1 - \left(\frac{\gamma(s)}{W_\infty} \right)^2 \tag{3.19}$$

and is typically plotted against x/l where x is the x-coordinate of the profile. The lift coefficient can be calculated using the standard definition and (3.13):

$$C_L = \frac{L}{\frac{1}{2}\rho W_\infty^2 l} = \frac{2\Gamma}{Wl} \tag{3.20}$$

Calculating the bound vorticity from (3.15) and substituting into (3.20) arrives at:

$$C_L = \frac{2}{Wl} \sum_{n=1}^{N} \gamma(s_n) \Delta s_n \tag{3.21}$$

3.2.1.5 Implementation

The program Flow() was extended to include the considerations for airfoils discussed in the previous sections and an updated flow chart is given in Figure 10. The code implementation of the back diagonal correction and equation (3.16) is:

```
for(I=0;I<pDoc->i_npanels;I++)
{
      SUM=0.0;

      for(J=0;J<pDoc->i_npanels;J++)
      {

            if (J!=(pDoc->i_npanels-I-1))
            {

                  SUM+=pDoc->fl_coupcoeff[I][J]*pDoc->fl_dels[J];

            }

      }

      pDoc->fl_coupcoeff[I][(pDoc->i_npanels-I-1)]=
            -1*SUM/pDoc->fl_dels[pDoc->i_npanels-I-1];
```

}

The effect of the back diagonal is shown in Figure 9 where the net circulation due to panel 9 is shown for a NACA 0012 airfoil represented by 28 panels. The back diagonal correction has the effect of ensuring zero net circulation around the geometry interior by adjusting the contribution from panel 20.

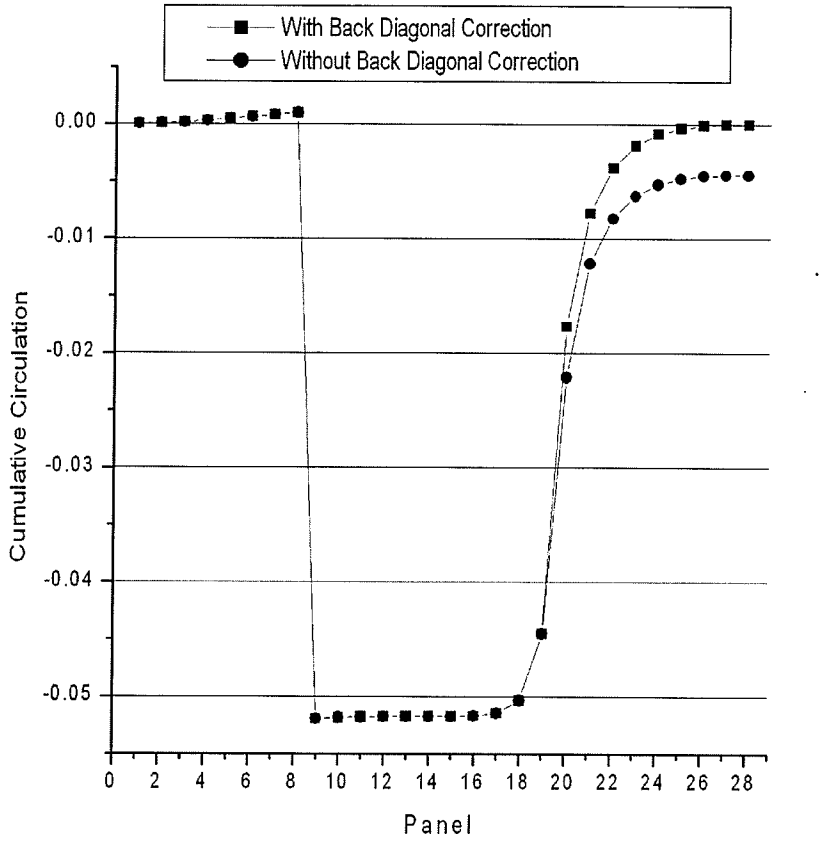


Figure 9) The effect of the back diagonal correction routine on the net circulation due to panel 9.

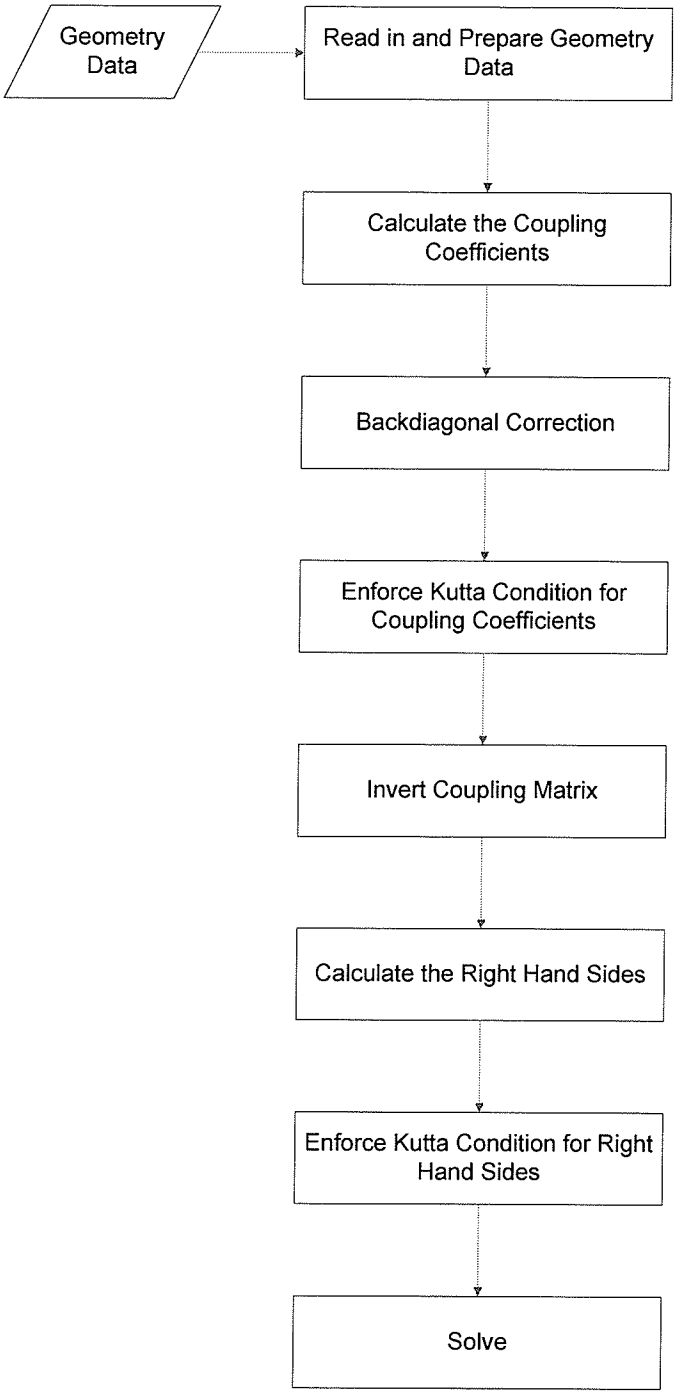


Figure 10) Flow chart of vortex panel program Flow () for airfoils

The code implementation for the Kutta condition's adjustment of the coupling coefficients is as follows:

```
for(J=pDoc->m_TE;J<=pDoc->i_npanels-1;J++)
{
      for(I=0;I<=pDoc->i_npanels-1;I++)
      {
            if(J>pDoc->m_TE) COEFF[I][J] = COEFF[I][J+1];
            else COEFF[I][J] = COEFF[I][J] - COEFF[I][J+1];
      }
}
```

In this algorithm, all of the coupling coefficients that are past the trailing edge (except for $te+1$) are moved to the location n-1. The trailing edge coefficients are subtracted, i.e. $K_{te} = K_{te} - K_{te+1}$, as indicated in (3.18).

The code implementation for the Kutta condition's right hand adjustment sides is as follows and is largely the equivalent as the implementation of the coupling coefficients:

```
for (I=pDoc->m_TE+1;I<=pDoc->i_npanels-1;I++)
{
      if (I>pDoc->m_TE+1)     pDoc->fl_rhs[I-1]=pDoc->fl_rhs[I];
      else pDoc->fl_rhs[I-1]=pDoc->fl_rhs[I-1]-pDoc->fl_rhs[I];
}
pDoc->i_npanels=pDoc->i_npanels-1;
```

3.2.1.6 Validation

The full flow solver, Flow(), with the above algorithms implemented shown in the flow chart Figure 10, was validated using the NACA 0012 airfoil previously discussed. The surface was represented by 30 panels using a cosine distribution. The angle of attack was varied from 0 to 10 degrees. Figure 11 shows the coefficient of pressure for several angles of attack.

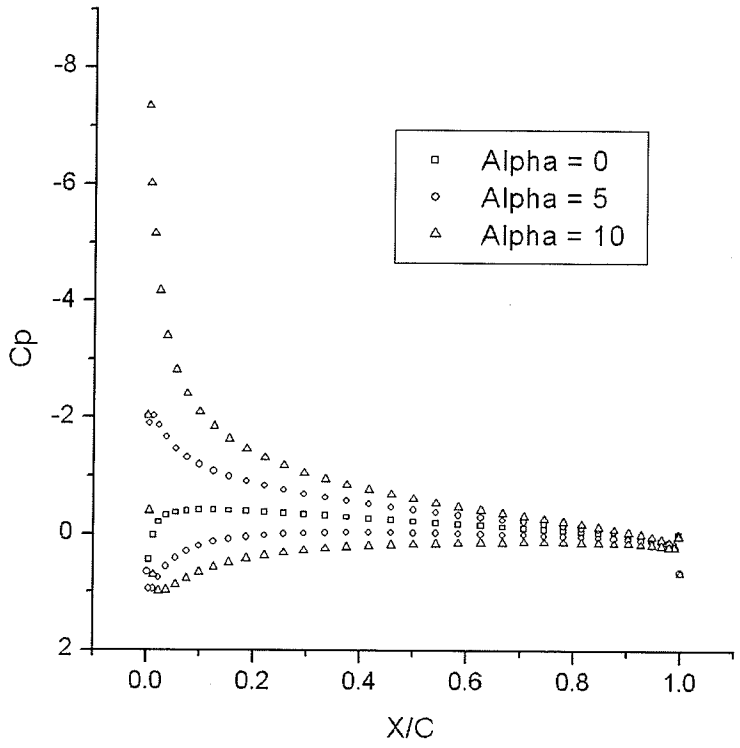


Figure 11) Coefficient of pressure for a NACA 0012 Airfoil

Table 2 shows the panel vorticity, the bounded circulation and the lift coefficient calculated for 0, 5 and 10 degrees. With 30 surface elements, the trailing edge is located at the 15^{th} and 16^{th} elements. Due to the implementation of the Kutta Condition, it is noticed that the values at this location are equal in magnitude and opposite in sign, even for the non-symmetric flows of 5 and 10 degrees angle of attack.

The results in Table 3 were compared with the work of Abbott and von Doenhoff (1959) and the predicted lift coefficients are in good agreement with the empirical data as seen in Table 4.

	Panel Vorticity		
	Angle of Attack		
Element	0	5	10
1	11.9381	22.1485	32.1904
2	11.1252	16.7939	22.3347
3	11.7985	15.3578	18.8002
4	11.8870	14.3760	16.7556
5	11.7761	13.6221	15.3645
6	11.5785	12.9925	14.3077
7	11.3277	12.4285	13.4347
8	11.0570	11.9192	12.6906
9	10.7905	11.4639	12.0500
10	10.5267	11.0452	11.4796
11	10.2571	10.6443	10.9505
12	9.9670	10.2396	10.4343
13	9.6365	9.8060	9.9009
14	9.4100	9.4830	9.4837
15	8.0544	8.0237	7.9320
16	-8.0544	-8.0237	-7.9320
17	-9.4100	-9.2655	-9.0505
18	-9.6365	-9.3937	-9.0794
19	-9.9670	-9.6185	-9.1968
20	-10.2571	-9.7918	-9.2520
21	-10.5267	-9.9281	-9.2540
22	-10.7905	-10.0351	-9.2032
23	-11.0570	-10.1107	-9.0874
24	-11.3277	-10.1407	-8.8765
25	-11.5785	-10.0763	-8.4974
26	-11.7761	-9.8404	-7.8298
27	-11.8870	-9.3076	-6.6573
28	-11.7985	-8.1495	-4.4383
29	-11.1252	-5.3719	0.4223
30	-11.9381	-1.6367	8.6771
Predicted Circulation	0.0000	3.0022	5.9815
CL=	0.0000	0.6004	1.1963

Table 3) Surface vorticity, bound circulation and lift coefficient for a NACA 0012 Airfoil

Angle of Attack	Flow()	Abbott & v. Doenhoff
0	0.00	0
5	0.600	0.58
10	1.20	1.18

Table 4) Comparison of the coefficient of lift calculated with program Flow() to the empirical data

3.2 The Vortex Cloud Method

3.2.1 Vortex Cloud Method – Theoretical Background

The vortex cloud, or discrete vortex, method comprises the surface vorticity method, described in 3.1, modified to interact with free vortices which move through the flow domain. The free vortices are convected through several processes:

1. Interaction with the free stream velocity, W_∞
2. Interaction with the other free stream vortices
3. Random displacements or permutations of velocity
4. Interaction with the body surface

The first interaction is quite simple to model; the free vortex is assigned the same velocity as the free stream velocity. The vortex will then move through the flow field at a constant velocity. The other three mechanisms are quite complex as their values are largely dependant on the local flow field and their proximity to the body surface.

3.2.2 Vortex Convection

Consider the vortex pair of equal strength shown in Figure 12. If there is no influence from an external system, these vortices will move in a direction perpendicular to the straight line connecting them and will rotate about a circle centered about the midpoint of this line.

Using a simplified form of (3.3),

$$\upsilon_d = \frac{\Gamma}{2\pi r} \tag{3.22}$$

the drift velocity for each free vortex can be calculated due to the strength of the other.

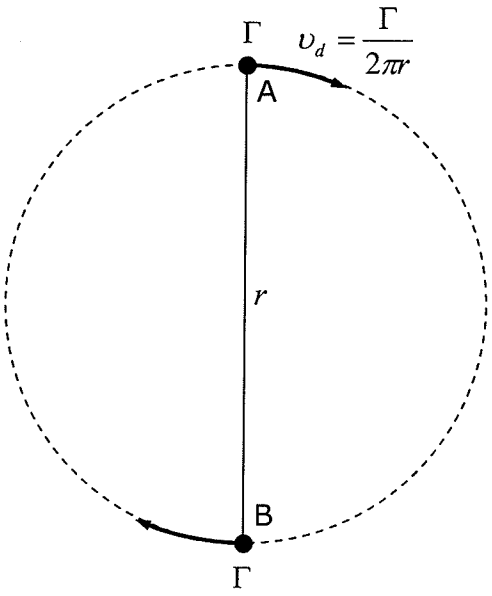


Figure 12) Exact vortex convective motion

If the simple system shown in Figure 12 is modeled using a time stepping routine with the time elapsed between steps of Δt, the displacement that vortex A will undergo is equivalent to

$$d = \frac{\Gamma}{2\pi r}\Delta t \tag{3.23}$$

Over several time steps, vortex A will experience a series of induced displacements and will pass through (x_1, y_1), (x_2, y_2), (x_3, y_3) etc. This displacement normal to the chord AB over time will lead to a spiraling motion, outward from the circle. Furthermore, if an attempt is made to retrace the path of motion by applying a negative time step $-\Delta t$, the motion

is irreversible as is evident in Figure 13. Also evident is that the larger the time step, Δt, and thus the induced displacement, d_n, the larger the error due to the spiral motion will be.

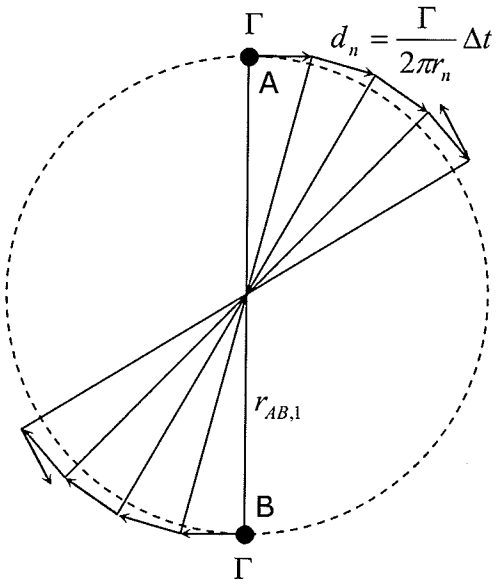


Figure 13) Spiral path and irreversibility due to forward differencing

Interestingly, this spiral motion can be considered to be similar, if not analogous, to the effects of diffusion. We would expect that, if subjected to a Brownian-type motion coupled with the induced velocity, the vortex A would tend to move perpendicular to the chord AB and away from B, effectively a spiral motion. It is for this reason that numerical error is commonly referred to as "numerical viscosity". More discussion on this will be presented in the section dealing with random walk.

3.2.3 Modified Euler Method

3.2.3.1 Theoretical Background

In order to manage the errors evident in Figure 13, and to obtain a method of convection that is reversible, a modified Euler method was employed. The unsigned velocity induced on one vortex, m, by another, n, described by (3.22) can be written, for unit strength Γ_n

$$\left.\begin{aligned} U_{mn} &= \frac{1}{2\pi}\left(\frac{y_m - y_n}{(x_m - x_n)^2 + (y_m - y_n)^2}\right) \\ V_{mn} &= \frac{1}{2\pi}\left(\frac{x_m - x_n}{(x_m - x_n)^2 + (y_m - y_n)^2}\right) \end{aligned}\right\} \tag{3.24}$$

If vortex m is in a cloud of N vortices, then the velocity induced by the cloud on m is the sum of the velocity induced by the free vortices n.

$$\left.\begin{aligned} u_{mn} &= \sum_{\substack{n=1 \\ n\neq m}}^{N} \Gamma_n U_{mn} \\ v_{mn} &= \sum_{\substack{n=1 \\ n\neq m}}^{N} \Gamma_n V_{mn} \end{aligned}\right\} \tag{3.25}$$

Using the forward differencing method, the vortex m would convect according to:

$$\left.\begin{aligned} x_{m,2} &= x_{m,1} + u_{m,1}dt \\ y_{m,2} &= y_{m,1} + v_{m,1}dt \end{aligned}\right\} \tag{3.26}$$

Once all of the vortices have been convected, we can now recalculate the convection step for the new position for each vortex. If, instead of moving

the vortex to point $(x_{m,3}, y_{m,3})$, the vectors $u_{m,1}dt$ and $u_{m,2}dt$, and $v_{m,1}dt$ and $v_{m,2}dt$ are averaged, a much better solution for the actual motion of vortex m is obtained. This is the fundamental element of the modified Euler method and is physically described by Figure 14.

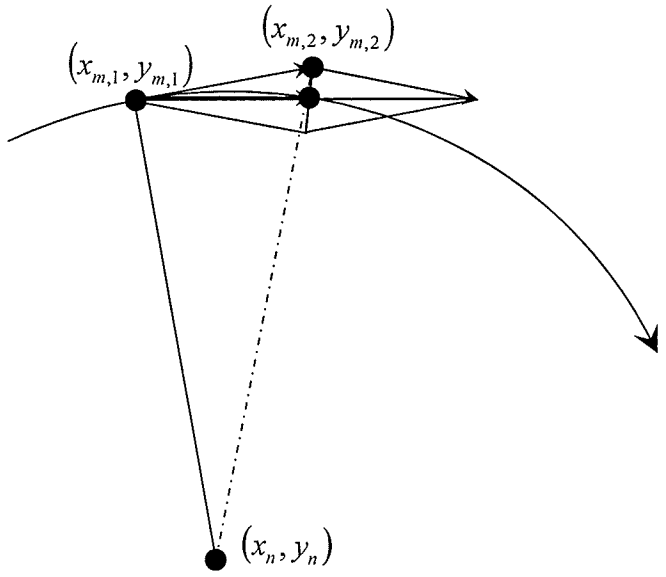


Figure 14) Modified Euler estimate of vortex motion

In this method, the effective displacement is taken to be the average of the displacements for the first and second steps.

$$\left.\begin{aligned} x_{m+1} &= x_m + \frac{(u_{m+1} + u'_{m+1})dt}{2} \\ y_{m+1} &= y_m + \frac{(v_{m+1} + v'_{m+1})dt}{2} \end{aligned}\right\} \qquad (3.27)$$

3.2.3.2 *Implementation*

Program Vortices() was written to explore vortex convection and examine methods to increase the accuracy of the numerical simulation. The program has the capacity to model many vortices and stores the vortex information, x-location, y-location, x-velocity, y-velocity, vortex strength and a switch variable that would enable the "turning on and off" of the vortex. This last variable will become useful in the program written for the vortex cloud model. The flow diagram for the program is shown in Figure 15 and 16, 15 being for the forward differencing method and 16 being for the central differencing method.

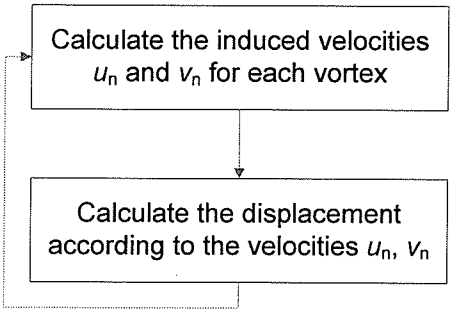


Figure 15) Flow chart for forward difference method

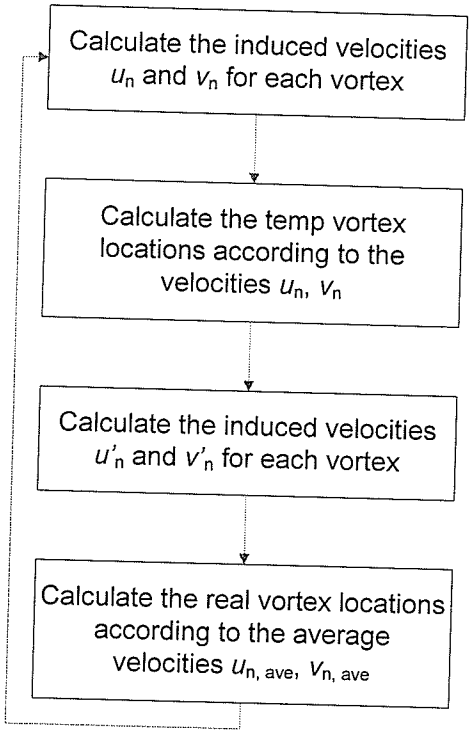


Figure 16) Flow chart modified Euler method

The program calculates the induced velocity on each vortex by every other vortex; it then stores this information in a temp array and uses this data for the intermediate time step. The program then calculates the true velocity and position of each vortex based on the average of the first and intermediate steps, i.e. `VORTECES[S][U]=(TEMP[S][U]+UI)/2+UFSV` for the x-velocity. Although this calculation includes the addition of the free stream velocity term, its value was zero in the work presented in this section. A shortened section of the code is presented for N vortices:

```
for(S=NI;S<=N_VORTECES;S++)
        {
                for(I=NI;I<=N_VORTECES;I++)
                {
                        if(I!=S)
                        {
                                ...
```

```
                        GAMA=VORTECES[I][GAMMA];
                        W=-1*GAMA/(2*PI*D);
                        UI+=sin(PHI)*W;
                        VI+=cos(PHI)*W;

                }
        }

        TEMP[S][U]=UI;
        TEMP[S][V]=VI;

        TEMP[S][X] = VORTECES[S][X]+(TEMP[S][U]*DTIME);
        TEMP[S][Y] = VORTECES[S][Y]+(TEMP[S][V]*DTIME);

}

for(S=NI;S<=N_VORTECES;S++)
{
        for(I=NI;I<=N_VORTECES;I++)
        {
                if(I!=S)
                {

                        …
                        GAMA=VORTECES[I][GAMMA];
                        W=-1*GAMA/(2*PI*D);
                        UI+=(y2-y1)*W;
                        VI+=(x2-x1)*W;
                }
        }

        VORTECES[S][U]=(TEMP[S][U]+UI)/2+UFSV;
        VORTECES[S][V]=(TEMP[S][V]+VI)/2+VFSV;

        VORTECES[S][X]=VORTECES[S][X]+(VORTECES[S][U]*DTIME);
        VORTECES[S][Y]=VORTECES[S][Y]+(VORTECES[S][V]*DTIME);
}
```

3.2.3.1 Validation

Two vortices were chosen for the validation of the modified Euler method and were arranged in a similar fashion to Figure 12 with vortex 1 located at $(1,0)$ and vortex 2 located at $(-1,0)$. Each vortex has a vorticity of unit strength. The path of the induced motion of each vortex for the forward difference method is shown in Figure 17. The spiraling effect of the method is evident as is expected.

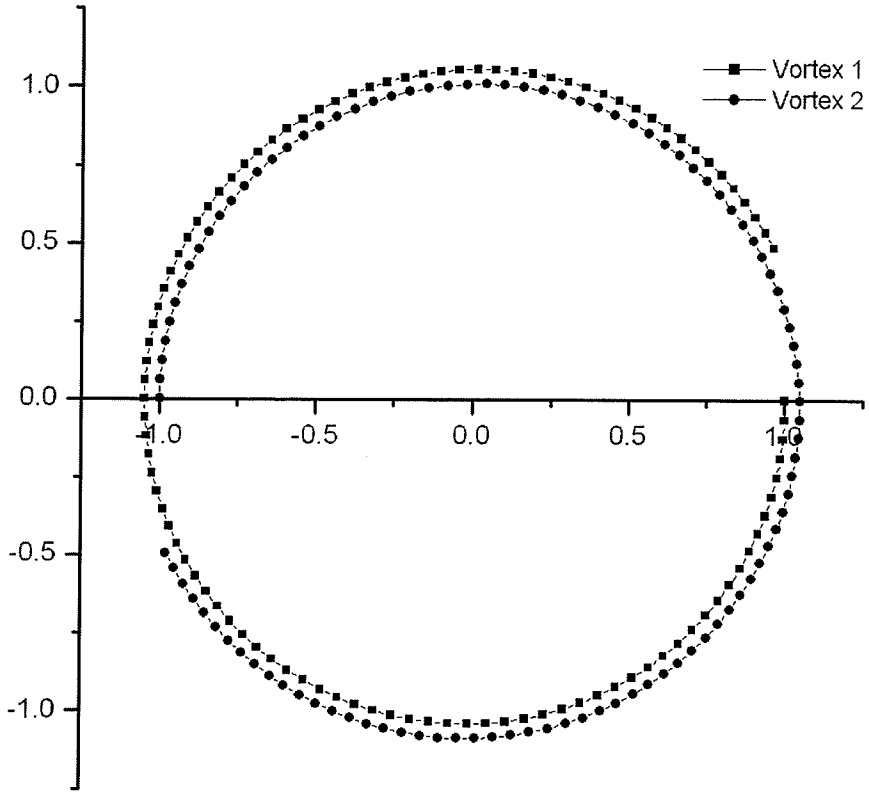


Figure 17) Spiral effect of the forward difference method

The results for the central differencing method are presented in Figure 18. Here, only one vortex is plotted but the quality of the path prediction is clearly shown. The time step was equal for both test cases.

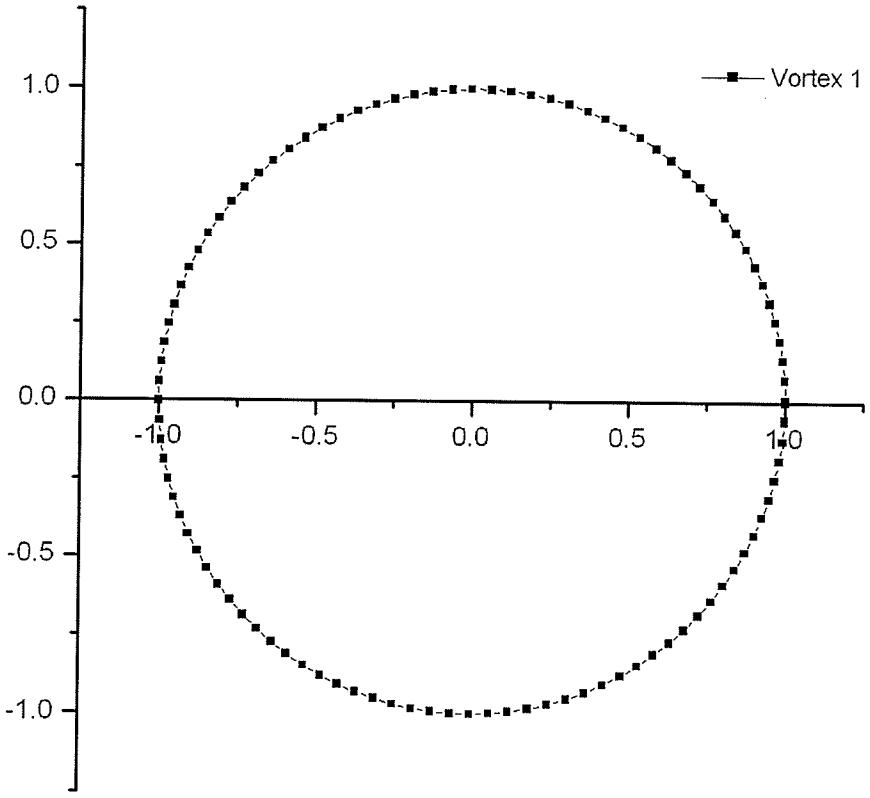


Figure 18) Results from the implementation of the modified Euler method

A logarithmic chart showing the error for each method is shown in Figure 19. The error is defined as the difference between the final position of vortex 1 (initially at $(1,0)$) and the location $(1,0)$ after the number of time steps required to drift the path defined by 2π radians. The time step was calculated based on the initial velocity of the vortex, known from the simple calculation using (3.21) and was based on 100 steps to complete the path.

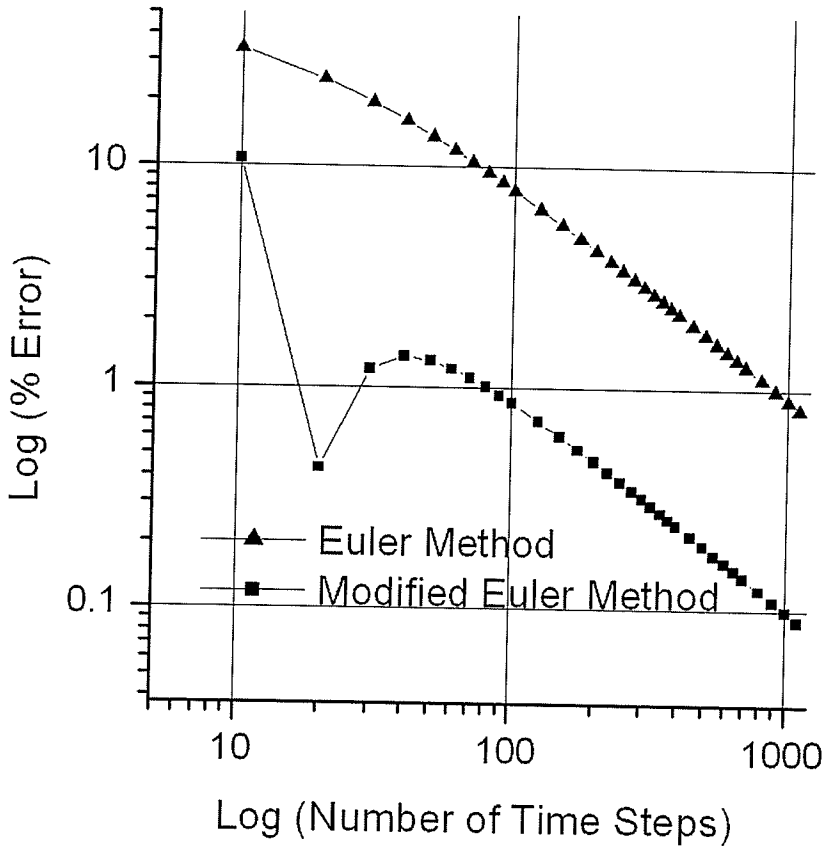


Figure 19) Error associated with the forward and modified Euler methods for a circular path and varying the time steps to travel 2π radians

The results shown in Figures 18 and 19 demonstrate that the vortex convection model which employs the modified Euler method tracks very well and is representative of natural flows and will therefore be used in the vortex cloud model. Although both methods are of order Δt, the modified Euler method is demonstrably more accurate, by an order of magnitude, than the accuracy of the forward differencing scheme for the equivalent number and size of time step. This affords a couple of opportunities for implementation:

1. The method can take advantage of the increased accuracy
2. The size of the time step may be increased to achieve a reasonable level of accuracy while reducing the computational requirement

In the development of the Flow() program, there was not investigation into the second option.

3.2.4 Viscous Diffusion

The third process for vortex displacement described in 3.2.1 is random displacements of velocity. This is the implementation of viscous diffusion which will be discussed here.

3.2.4.1 Theoretical Background

The "random walk" model, first proposed by Chorin (1973), has been used with good success for the simulation of viscous diffusion in vortex cloud models. The basic principle is to subject each element in the cloud to a scaled perturbation of location or velocity that will model diffusion due to viscosity. Such motion can be described by the two-dimensional Navier Stokes equations.

$$\frac{\partial \omega}{\partial t} + (\mathrm{q} \cdot \nabla)\omega = \nu \nabla^2 \omega \qquad (3.28)$$

where q is the velocity vector and ∇^2 is the Laplacian operator. The second term, $(\mathrm{q} \cdot \nabla)\omega$, describes the convection or vorticity and the third term, $\nu \nabla^2 \omega$, describes the diffusion of vorticity. Writing (3.28) in an alternative

dimensionless form, normalized by length and velocity scales, l and W_∞, respectively

$$\frac{\partial \overline{\omega}}{\partial \bar{t}} + (\overline{\mathrm{q}} \cdot \nabla)\overline{\omega} = \frac{\nu}{lW_\infty}\nabla^2\omega = \frac{1}{\mathrm{Re}}\nabla^2\omega \tag{3.29}$$

the influence of the Reynolds number is apparent. For high Reynolds number flows, the diffusion term has little effect on the flow field. At low Reynolds number flows, equation (3.28) reduces to:

$$\frac{\partial \omega}{\partial t} = \nu\nabla^2\omega \tag{3.30}$$

The solution of (3.30) for polar coordinates is given by Batchelor (1970),

$$\omega(r,t) = \frac{\Gamma}{4\pi\nu t} e^{\left(\frac{-r^2}{4\nu t}\right)} \tag{3.31}$$

Porthouse and Lewis (1981) have developed a random walk scheme based on this solution. They argue that, if Γ is replaced by N discrete vortices of strength Γ / N initially located at the origin, there will be n number of discrete vortices in the area $r\Delta r\Delta\theta$ after time t. The total amount of vorticity, p, in this area can be expressed by:

$$p = \frac{n}{N} = \frac{1}{4\pi\nu t} e^{\left(\frac{-r^2}{4\nu t}\right)} r\Delta r\Delta\theta \tag{3.32}$$

Equation (3.32) can also be interpreted to define the probability p that a vortex will be in the area $r\Delta r\Delta\theta$ at time t.

The symmetry of the radial system led to the conclusion that the probability that a vortex will land within any arc $\Delta\theta$ is equal and is independent of the probability that the vortex will fall into the range between r and $r+\Delta r$. The

angular portion of the random displacement is then assigned the simple value of

$$\theta' = 2\pi L \tag{3.33}$$

where L is a random number between 0.0 and 1.0.

Porthouse and Lewis (1981) argue that the radial scattering of the vortices is best found by first integrating (3.32) between $\theta - 0$ and 2π and then integrating between $r = 0$ and r:

$$p(r) = 1 - e^{\frac{-r^2}{4\nu t}} \tag{3.34}$$

Equation (3.34) is a form of the standard normal curve from statistical theory,

$$p(x)dx = \frac{1}{\sqrt{2\pi}} e^{\frac{-z^2}{2}} dz$$

and $p(r)$ is equally likely to fall anywhere between 0.0 and 1.0. P_i is then defined as a random number between 0.0 and 1.0 that will be used as a seed to the radial displacement. If (3.34) is solved for the r_i^{th} element it will describe the radial portion of the random displacement

$$r' = \sqrt{4\nu\Delta t \ln(1/P_i)} \tag{3.35}$$

The x and y components of displacement are then given by:

$$\left.\begin{aligned} x' &= r'\cos\theta' \\ y' &= r'\sin\theta' \end{aligned}\right\} \tag{3.36}$$

The method proposed by Porthouse and Lewis (1981) is attractive because it scales the random displacement in accordance to the system viscosity, ν, as

well as to the Reynolds number by way of the time step, Δt and ν, as will be discussed.

Because the current work employs a modified Euler method to track the vortex motion, it must be determined at which step the random component will be applied. It is considered to be overkill to apply this during the intermediary time step as well as after the total velocity has been calculated and it is therefore only applied after the central differencing step has been completed. Besides computational simplicity, this also has the secondary benefit of reducing the chances of randomly displacing a vortex into the body surface during the central differencing step and thereby reducing unnecessary occasions to treat this event. This will be further discussed in section 3.2.7. Adding (3.36) to (3.27):

$$\left.\begin{aligned} x_{m+1} &= x_m + \frac{(u_{m+1} + u'_{m+1})dt}{2} + x' \\ y_{m+1} &= y_m + \frac{(v_{m+1} + v'_{m+1})dt}{2} + y' \end{aligned}\right\} \tag{3.37}$$

3.2.4.2 *Implementation*

The Vortex() program, discussed in 3.2.3.1 was also used to determine the merit of the random walk component of vortex convection as a suitable model for representing Brownian motion. The program was extended in order to track N vortices through vortex convection and includes the random walk according as seen in the program flow chart, Figure 20.

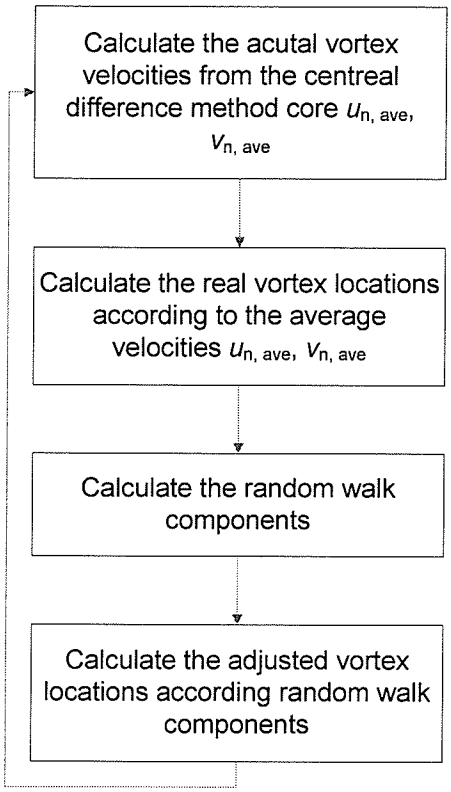


Figure 20) Flow chart for Vortices() with random walk

The implementation of the random walk is quite simple. The following code generates the random numbers and applies the positional shift to the calculated x- and y-locations for each free vortex.

```
L=random();
K=random();
L = 2*pDoc->m_PI*L;
K = pow(4*(pDoc->m_viscosity/pDoc->m_density)*
      pDoc->DTIME*log(1/K), .5);

pDoc->fl_vorteces[S][D_X] = pDoc->fl_vorteces[S][D_X]+
      (pDoc->fl_vorteces[S][D_U]*pDoc->DTIME)+(K*cos(L));

pDoc->fl_vorteces[S][D_Y] = pDoc->fl_vorteces[S][D_Y]+
      (pDoc->fl_vorteces[S][D_V]*pDoc->DTIME)+(K*sin(L));
```

3.2.4.3 Validation

As the entire premise of the random walk method rests on the ability to create quality random numbers, the first element to be validated was the function `random()`. The following code was added to the Vortices() program in order to determine whether the function would return a reasonable distribution of random numbers:

```
int bin[10]={0,0,0,0,0,0,0,0,0,0};

for (I=1;I<=1000;I++)
{
      K=random();

      bin[((int)(K*10))]++;
}
```

The algorithm creates 1000 random numbers and sorts them into bins according to their value. Table 4 shows that the output from this routine is indeed reasonable as every bin has approximately the same number of random numbers in it, as would be expected of a random distribution:

Bin	Range	Number Collected
1	0.0-0.1	111
2	0.1-0.2	97
3	0.2-0.3	86
4	0.3-0.4	89
5	0.4-0.5	98
6	0.5-0.6	115
7	0.6-0.7	93
8	0.7-0.8	93
9	0.8-0.9	105
10	0.9-1.0	113

Table 5) Range of random numbers created by random()

The method employed to validate the motion is similar to that used to determine the quality of the random numbers. A series of bins, shown in Figure 21, was created and the program tracked the location of each vortex at each time step.

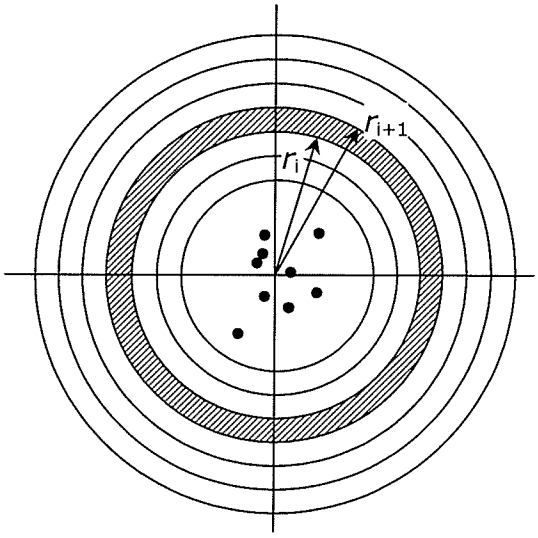


Figure 21) Bins with initial random vortex positions

N vortices, of unit strength, were created and their initial location was determined using the function `random()`, i.e. their initial x- and y-locations were random and between 0 and 1 in magnitude. The number of vortices which fall into concentric annular "bins", where bin i will be the annular region described between r_i and r_{i+1}, is tracked and plotted in Figure 22 for the 200th time step. One would expect that the random walk treatment applied in the program Vortices() would cause them naturally to drift apart. If a reasonable representation of Brownian motion is achieved, the resultant position of the vortices due to drift should follow a normal or Gaussian

distribution. Figure 22 shows the results from the Binning test as well as a trendline following a normal distribution and demonstrates that the results appear to be Gaussian.

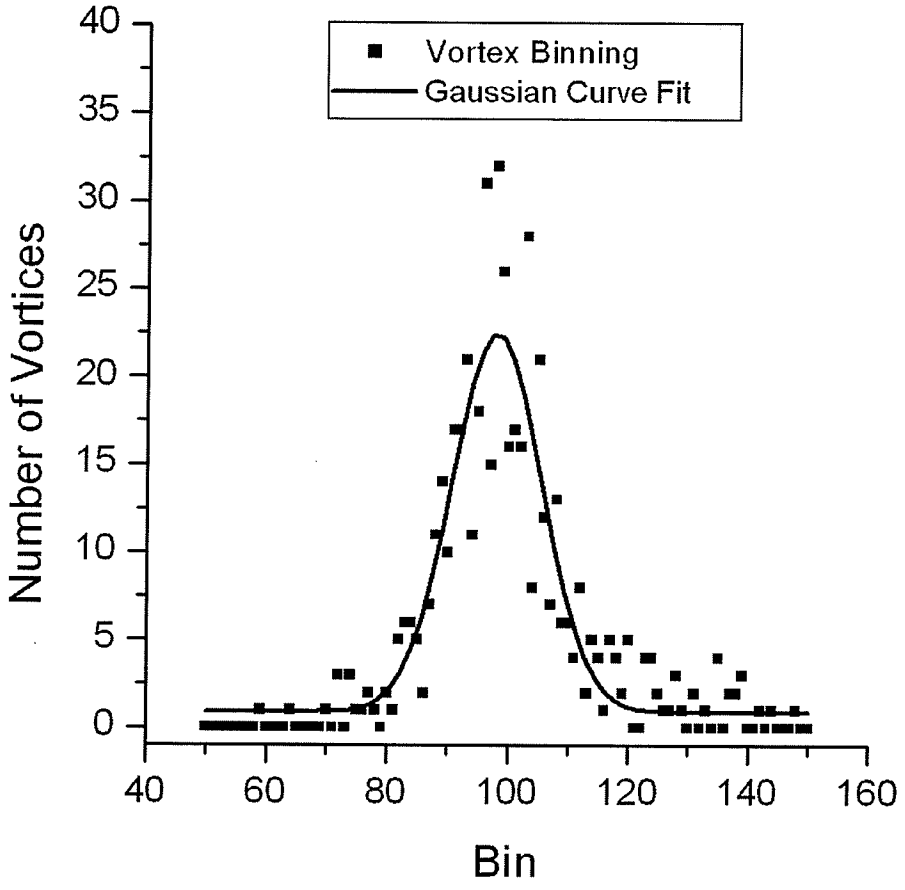


Figure 22) Distribution of vortices after 200 iterations due to drift

3.2.5 Panel-Vortex Interaction

As stated in 3.2.1, vortex convection, in a vortex cloud model, is dependant on four factors. To this point, the first two have been discussed and the third

source of induced velocity causing vortex convection, is the interaction between the free vortices and the body surface. The vortex panel method discretizes the body surface into a series of finite elements each with a surface vorticity that is representative of the surface velocity. This surface vorticity interacts with the free vortices in much the same way as was presented in section 3.2.2. The relationship between the surface vorticity $\gamma(S_p)\Delta S$ of a panel, p, and the induced velocity of the free vortex is:

$$\left.\begin{aligned} u_{m,p} &= \frac{1}{2\pi}\sum_{p=1}^{N}\gamma(S_p)\Delta S_p\left(\frac{y_m - y_p}{r_{mp}^{\ 2}}\right) \\ v_{m,p} &= \frac{-1}{2\pi}\sum_{p=1}^{N}\gamma(S_p)\Delta S_p\left(\frac{x_m - x_p}{r_{mp}^{\ 2}}\right) \end{aligned}\right\} \tag{3.38}$$

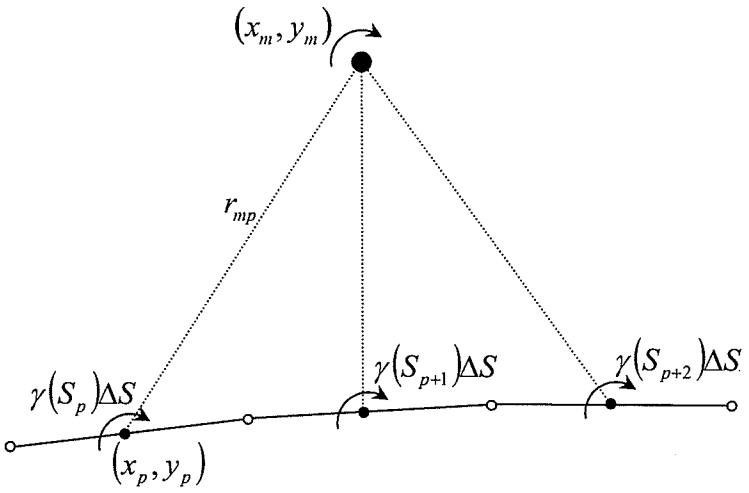


Figure 23) Vortex interaction with the body surface

The convective contribution from the body surface must be taking into consideration during the central differencing method. Adding (3.26) to (3.23) and (3.24) gives

$$\left.\begin{aligned} u_m &= U_\infty + \frac{1}{2\pi}\sum_{n=1}^{M}\gamma(S_n)\Delta S_n\left(\frac{y_m - y_n}{r_{mn}^{\ 2}}\right) + \frac{1}{2\pi}\sum_{n=1}^{M}\Gamma_n\left(\frac{y_m - y_n}{r_{mn}^{\ 2}}\right) \\ v_m &= V_\infty + \frac{1}{2\pi}\sum_{n=1}^{M}\gamma(S_n)\Delta S_n\left(\frac{x_m - x_n}{r_{mn}^{\ 2}}\right) + \frac{1}{2\pi}\sum_{n=1}^{M}\Gamma_n\left(\frac{x_m - x_n}{r_{mn}^{\ 2}}\right) \end{aligned}\right\} \qquad (3.39)$$

and

$$\left.\begin{aligned} u'_{m+1} &= U_\infty + \frac{1}{2\pi}\sum_{n=1}^{M}\gamma(S_n)\Delta S_n\left(\frac{y'_{m+1} - y_n}{r'^{\ 2}_{m+1,n}}\right) + \frac{1}{2\pi}\sum_{n=1}^{M}\Gamma_n\left(\frac{y'_{m+1} - y'_n}{r'^{\ 2}_{m+1,n}}\right) \\ v'_{m+1} &= V_\infty + \frac{1}{2\pi}\sum_{n=1}^{M}\gamma(S_n)\Delta S_n\left(\frac{x'_{m+1} - x_n}{r'^{\ 2}_{m+1,n}}\right) + \frac{1}{2\pi}\sum_{n=1}^{M}\Gamma_n\left(\frac{x'_{m+1} - x'_n}{r'^{\ 2}_{m+1,n}}\right) \end{aligned}\right\} \qquad (3.40)$$

Lewis (1991) presents the boundary integral equation for a body immersed in a vortex cloud as:

$$-\frac{1}{2}\gamma(s_n) + \frac{1}{2\pi}\oint k(s_m, s_n)\gamma(s_n)ds_n + W_\infty \mathrm{d}s + \frac{1}{2\pi}\sum_{j=1}^{Z} L(m, j)\Gamma_j = 0 \qquad (3.41)$$

The first three terms are equivalent to the original system of equations written for the vortex panel method. With the body immersed in cloud of free vortices, there is significant velocity induced on the body surface by these vortices. The fourth term in (3.41) ensures that the Dirichlet boundary condition is maintained and its numerical form is:

$$\sum_{n=1}^{M} K(s_m, s_n)\gamma(s_n) = -U_\infty \cos\beta_m - V_\infty \sin\beta_m - \sum_{j=1}^{Z} \frac{\Gamma_j}{2\pi r_{mj}} \cos\phi_{mj} \cos\beta_m - \sum_{j=1}^{Z} \frac{\Gamma_j}{2\pi r_{mj}} \sin\phi_{mj} \sin\beta_m \quad (3.42)$$

where $r_{mj} = \sqrt{(x_m - x_j)^2 + (y_m - y_j)^2}$.

3.2.6 Sub-Panels

One of the major concerns of vortex–surface interaction is the risk of a disproportionate influence of a local pivotal point due to vortex proximity, coarse discretization, or both. Figure 24 gives a good physical description of the problem.

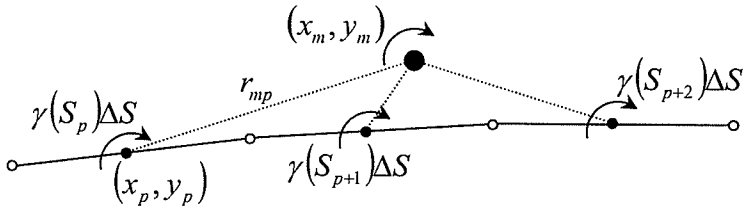


Figure 24) Influence of surface vorticity in close proximity

The issue is that the average surface vorticity for panel p+1, $\gamma(s_{p+1})\Delta s_{p+1}$, is applied at the control point, rather than spread over the panel surface. As the free vortex interacts with this discrete point there is the unnatural effect of disproportionately large influence caused by the surface vorticity whereas the panel vorticity should be averaged over the entire panel, causing an even, distributed influence as the vortex passes near the surface. The application of positive vorticity at the control point will cause vortex m in

Figure 24 to have a large velocity towards the body surface when it is near but ahead of the pivotal point and away from the surface when it is near but behind the pivotal point. This leads to the type of vortex motion around a NACA 0012 airfoil, seen in Figure 25.

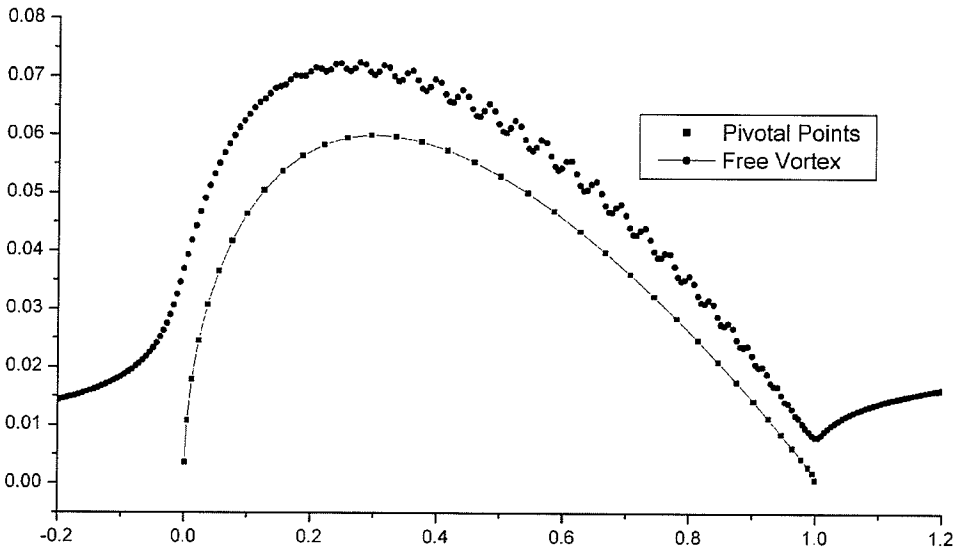


Figure 25) Vortex path demonstrating unnatural pathline caused by surface vorticity applied at the pivotal point.

Figure 25 is stretched in the vertical direction to give a better representation of the vortex motion. It is also interesting to note that the irregular motion is only present along the back side of the airfoil, starting just forward of the quarter chord. This is due to cosine distribution of panel points, where the panels are much coarser along the back side of the airfoil. Although the leading edge panels appear larger in the figure, they are actually quite small

and this reduces the near wall effect. The vortex shown in Figure 25 is shown to oscillate as much as 8.3% at the 50% chord position.

A common method to alleviate this problem is by using sub-panels, where the panels that are less than a predetermined distance from a free vortex, (Lewis (1991) suggests $d \leq 2.0\Delta s_m$ as a reasonable distance), are divided into a series of smaller panels each with a temporary control point where surface vorticity is applied.

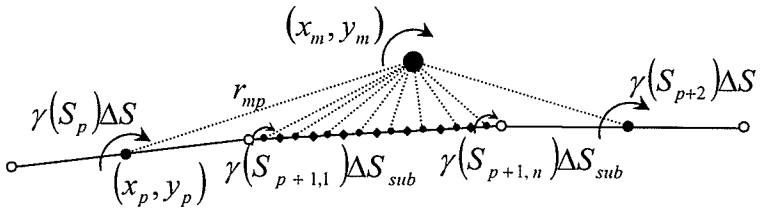


Figure 26) The use of sub-elements reduces the near wall effects of coarse discretization

The number of sub-elements will have a large impact on the computational requirement and it is therefore prudent to make the number a binary-friendly number (2, 4, 8...) to reduce the computational impact. Lewis (1991) suggests that a minimum of four sub-elements is required for general use.

The implementation of sub-panels requires that, when each distance, r_{mp}, is calculated, a test is performed to determine whether or not the local panel qualifies for subdivision. If it does, then the panel in question must be flagged for subdivision both in the convection process as well as in the

calculation of the right hand sides. The panel right hand sides would then be equal to the average of the right hand sides of the sub-panels.

Sub-paneling was considered in the development of the current work but the complexities of tracking which panels need subdivision, additional programming complexity and validation did not merit the incremental decrease in computational resources, especially considering the speed of modern computers. This element of program was explored and discarded in favour of increasing the number of surface panels. The effect is essentially the same and there is the double benefit of simplicity in programming and standardization in the algorithms. Figure 27 depicts the effect of increasing the number of surface panels from 74 to 174.

The path of the vortex over 174 panels is much more realistic than the path of the vortex over 74 panels and the maximum oscillation has been reduced to 0.16% at 37% chord. Even though a slight wavering in the path is evident along the back side of the airfoil in the case of 174 panels, it is a good representation of an actual streamline and is considered to be more than sufficient for this algorithm. It is important to note that, as the user may specify the number of surface panels, they would have to be aware of this omission in order to specify a number of panels for a reasonable accuracy. Expermimentation during the course of the developing the algorithm has shown that 120 panels provide a good numerical compromise between accuracy and speed, some variability in the path over the geometry is acceptable as it only represents ¼ of the interactions causing motion and the free vortices are being randomly displaced.

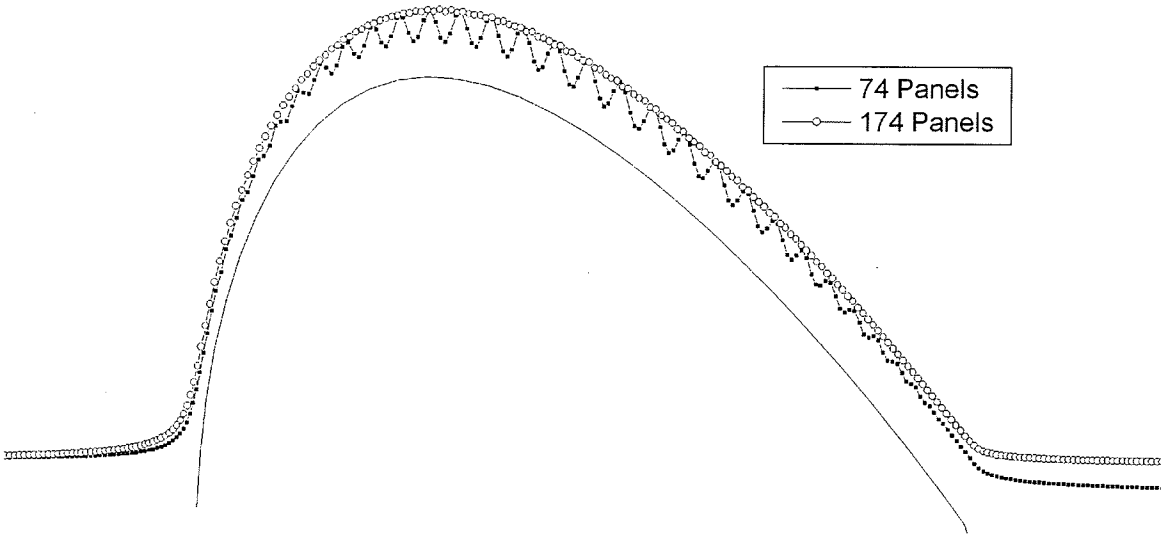


Figure 27) Effect of increasing the number of surface panels on the vortex pathline

3.2.7 Absorption and Destruction of Vorticity

There are two events that require that the free vortices be removed from the simulation domain:

1. The vortex enters the body surface
2. The vortex moves so far from the body that it no longer has an effect on the analysis

In the first case, as the path of free vortices under convection routinely comes near the body surface, it is possible for this vortex to pass through the body surface due to a random movement or because of interaction with the body itself. In this event, the vortex is no longer a viable contributor to the flow field. There are several theories on how to deal with this:

1. Porthouse (1983) recommends that these be retained for the consequent convection iteration and then destroyed.
2. Chorin (1978) recommends that these vortices should be bounced back into the flow.
3. Lewis (1991) argues that the vortex entering the body should be absorbed into the surface vorticity, which can then be shed during a future time step.

It is computationally exhaustive to continue tracking the free vortices as they move far downstream from the body. These no longer contribute to the domain under investigation and can be removed from the simulation. However, an additional statement is required to ensure conservation of vorticity and is expressed as:

$$\sum_{n-1}^{N}\gamma(s_n)\Delta s_n + \sum_{j=1}^{Z}\Gamma_j - \Gamma_{circ} = 0 \tag{3.43}$$

Where Γ_{circ} is the cumulative strength of all destroyed vorticity and is initially 0. Equation (3.44) must be added to the Martensen equations in order to ensure that the correct amount of vorticity is shed at each time step:

$$\sum_{n=1}^{M}\left(K(s_m,s_n)+\Delta s_n\right)\gamma(s_n) = -U_\infty\cos\beta_m - V_\infty\sin\beta_m - \sum_{j=1}^{Z}\frac{\Gamma_j}{2\pi r_{mj}}\cos\phi_{mj}\cos\beta_m - \sum_{j=1}^{Z}\frac{\Gamma_j}{2\pi r_{mj}}\sin\phi_{mj}\sin\beta_m - \sum_{j=1}^{Z}\Gamma_j + \Gamma_{circ} \tag{3.44}$$

3.2.7.1 Implementation

The current work employs Lewis' absorption scheme for vorticity entering the body surface because of the natural manner in which this method handles

vorticity, the numerical simplicity in adding vorticity to vorticity and for computational convenience. Also there is little overhead required to implement such a scheme. A sample of the code for this algorithm is given for vortices that are above the line y=0.

```
if (pDoc->fl_vorteces[S][D_X]>pDoc->m_XDATA[I] &&
      pDoc->fl_vorteces[S][D_X]< pDoc->m_XDATA[I+1])
{
      if (pDoc->fl_vorteces[S][D_Y]>0 &&
            pDoc->fl_vorteces[S][D_Y]< pDoc->m_YDATA[I])
      {

            pDoc->fl_vorteces[S][D_SWITCH]=0;
            pDoc->fl_points[I][D_GAMMA] +=
                  pDoc->fl_vorteces[S][D_GAMMA];
            pDoc->fl_vorteces[S][D_X]=0.0;
            pDoc->fl_vorteces[S][D_Y]=0.0;
            pDoc->fl_vorteces[S][D_U]=0.0;
            pDoc->fl_vorteces[S][D_V]=0.0;
            pDoc->fl_vorteces[S][D_GAMMA]=0.0;
            pDoc->VORTINFLOW--;

      }
}
```

where `pDoc->VORTINFLOW` is the number of vortices in the flow field at any instant. The first statement is setting the switch for the deleted vortex element to 0. This will cause the program to ignore this vortex in all subsequent calculations until a housekeeping routine is called to allow these dead vortices to be replaced by new ones.

Where the free vortices travel so far downstream as to be no longer computationally significant, a similar coding is used except that the free vorticity that is being destroyed is added to the accumulating circulation variable `pDoc->CIRC`. The variable `OOR` is user defined and represents the distance downstream after which the free vortices will be destroyed.

```
if (pDoc->fl_vorteces[S][D_X]>OOR)
{

      pDoc->fl_vorteces[S][D_SWITCH]=0;
      pDoc->CIRC += pDoc->fl_vorteces[S][D_GAMMA];
      pDoc->fl_vorteces[S][D_X]=0.0;
      pDoc->fl_vorteces[S][D_Y]=0.0;
      pDoc->fl_vorteces[S][D_U]=0.0;
      pDoc->fl_vorteces[S][D_V]=0.0;
      pDoc->fl_vorteces[S][D_GAMMA]=0.0;
      pDoc->VORTINFLOW--;
      return true;

}
```

The implementation of the coupling coefficients and the right hand side additions according to (3.44) are simple and will therefore not be presented.

3.2.8 Vortex Shedding

3.2.8.1 Theoretical Background

The vortex cloud is formed by vorticity being shed from the body surface. In a full vortex cloud model, the surface vorticity, $\gamma(s_m)\Delta s_m$, calculated in the potential flow analysis is shed as a free vortex into the flow field. Lewis presents two methods for delivering these free vortices into the flow field:

1. by random walk
2. by initial offset.

If the random walk method is used, Lewis recommends that the vortex strength of the free vortices should be double the surface vorticity. This is due to the statistical reality that, if random walks are applied to a free vortex on the body surface, half will enter the surface and half will enter the flow field. This doubling of the surface vorticity is an attempt to conserve the vorticity at each time step.

The initial offset method throws the free vortex a certain distance from the panel surface, dramatically increasing the chance that the vortex will enter the flow field. This is shown in Figure 28.

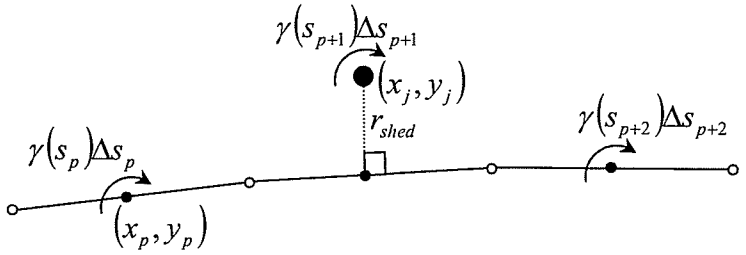


Figure 28) Vortex shedding by offset method

Porthouse (1983) recommends that this offset distance is defined by:

$$r_{shed} = \sqrt{\frac{4}{3}\nu\Delta t} \tag{3.45}$$

There is some difficulty with using this value at high Reynolds number flows as the value of r_{shed} becomes infinitely small due to the decreasing time step. This will be further discussed in the following sections.

3.2.8.2 Implementation

Vortex shedding was implemented using (3.45), ensuring that the distance, r_{shed}, is normal to and centered on the local panel. The number of vortices shed in any given time step is equal to the difference between the maximum number of free vortices in the flow, `pDoc->i_nvorteces`, and the total number of vortices in the flow at the instant that vorticity is being shed, `pDoc->VORTINFLOW`, up to a maximum number, `VORADD`. Both `pDoc-`

>i_nvorteces and VORADD are user specified. The panels to shed vorticity are randomly selected using the random() function. There is also a tuning variable, OFFSETFACTOR, that allows the offset distance to be adjusted by the user if required.

The variable PERC was added to the algorithm as a way to adjust the percentage of vorticity shed from a given surface. The addition of this variable brings some coding difficulties due to the requirement of conservation of vorticity. The reason for the variable is to promote program stability and to reduce the surface vorticity fluctuations that are present if the entire Martensen vorticity is shed into the flow field. As demonstrated in the next section, a further advantage to shedding only a portion of the surface vorticity is that the influence of the surface on the free stream vortices is maintained. The method of ensuring that the total simulation vorticity is conserved is to sum the shed vorticity, VORTOT, and compare this with the circulation variable, pDoc->CIRC. If pDoc->CIRC is not reduced to an acceptable vorticity, VORLIM, that will be carried forward to the next time step, the program continues to shed vorticity until this condition is met. This condition overrides the users limit on vorticity shed in a given time step. The code implementation of this routine is:

```
pDoc->VORTDIFF = pDoc->i_nvorteces-pDoc->VORTINFLOW;

if (pDoc->VORTDIFF > VORADD ) pDoc->VORTDIFF = VORADD

for(I=1;I<=pDoc->VORTDIFF;I++)
{

      RNDNUM = random();
      RNDNUM = RNDNUM*pDoc->i_npanels;
      PANEL = (int)RNDNUM;
```

```
        int OFFSETFACTOR = 10;

        GMA = pDoc->fl_points[PANEL][D_GAMMA];

        Rshed = OFFSETFACTOR*sqrt(4*(pDoc->m_viscosity/
              pDoc->m_density)*pDoc->DTIME/3);

        X1=pDoc->m_XDATA[PANEL];
        Y1=pDoc->m_YDATA[PANEL];
        X2=pDoc->m_XDATA[PANEL+1];
        Y2=pDoc->m_YDATA[PANEL+1];

        X1=X2-X1;
        Y1=Y2-Y1;

        PANELLENGTH= sqrt(pow(X1,2)+pow(Y1,2));

        X2=X2-(X1/2);           // find center of panel
        Y2=Y2-(Y1/2);

        pDoc->fl_vorteces[VN][D_Y]=Y2-fabs(X*X1/PANELLENGTH);
        pDoc->fl_vorteces[VN][D_X]=X2+fabs(X*Y1/PANELLENGTH);

        pDoc->VORTINFLOW++;
        pDoc->fl_vorteces[VN][D_SWITCH]=1;
        pDoc->fl_vorteces[VN][D_GAMMA]=PERC * GMA;
        VORTOT += pDoc->fl_vorteces[VN][D_GAMMA]
        pDoc->fl_vorteces[VN][D_U]=0;
        pDoc->fl_vorteces[VN][D_V]=0;
        pDoc->fl_points[PANEL][D_GAMMA]=(1-PERC) * GMA;
        if fabs((VORTOT - pDoc->CIRC) > VORLIM) I--;
}

pDoc->CIRC -= VORTOT;
```

Figure 29 shows 30 vortices shed from a NACA 0012 airfoil as calculated by the function addvort():

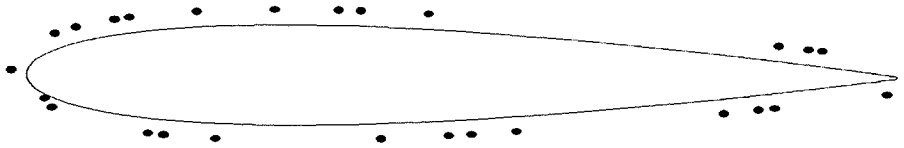


Figure 29) Free vortices shed from the surface of a NACA 0012 airfoil

3.2.9 Full Vortex Cloud Simulation

The algorithms presented in the last three sections are the final steps required to implement a full vortex cloud model. The program Flowa() was written to encompass all of these algorithms. The program flow chart indicating the adopted sequence for solution of the system of equations and vortex tracking is shown in Figure 30.

The time interval, Δt, is scaled to the Reynolds number by way of the free stream velocity. For the Flowa() program, the time step was determined using

$$\Delta t = \frac{2l}{aW_\infty} \tag{3.46}$$

where a is a user-defined variable. In the current case, a value of $a = 20$ was used. As W_∞ increases, Δt decreases, and therefore the vorticity shedding distance, r_{shed}, also decreases. The variable a may be used to increase this distance at high Reynolds number flow, so long as the numerical scheme retains resolution and validity.

Another addition to the program was a small calculation that helped to reduce the pure numerical dependence for the determination of the velocity of the free vortices. As a vortex comes very near another, (3.3) suggests that the induced velocity on each vortex approaches infinity. This is discordant with natural vortex motion and therefore

$$\Gamma_{j,calc} = \frac{\Delta s_{\min}}{2\pi}\Gamma_j \tag{3.47}$$

was applied for vortices within $\Delta s_{\min}/2\pi$ of each other. The following code was added to the vortex convection routines and the calculation of the right hand sides:

```
if (RmjSQ < MGCheck)
{
      GMA = GMA*(RmjSQ/MGCheck);
}
```

where `RmjSQ` is the straight line distance between vortices.

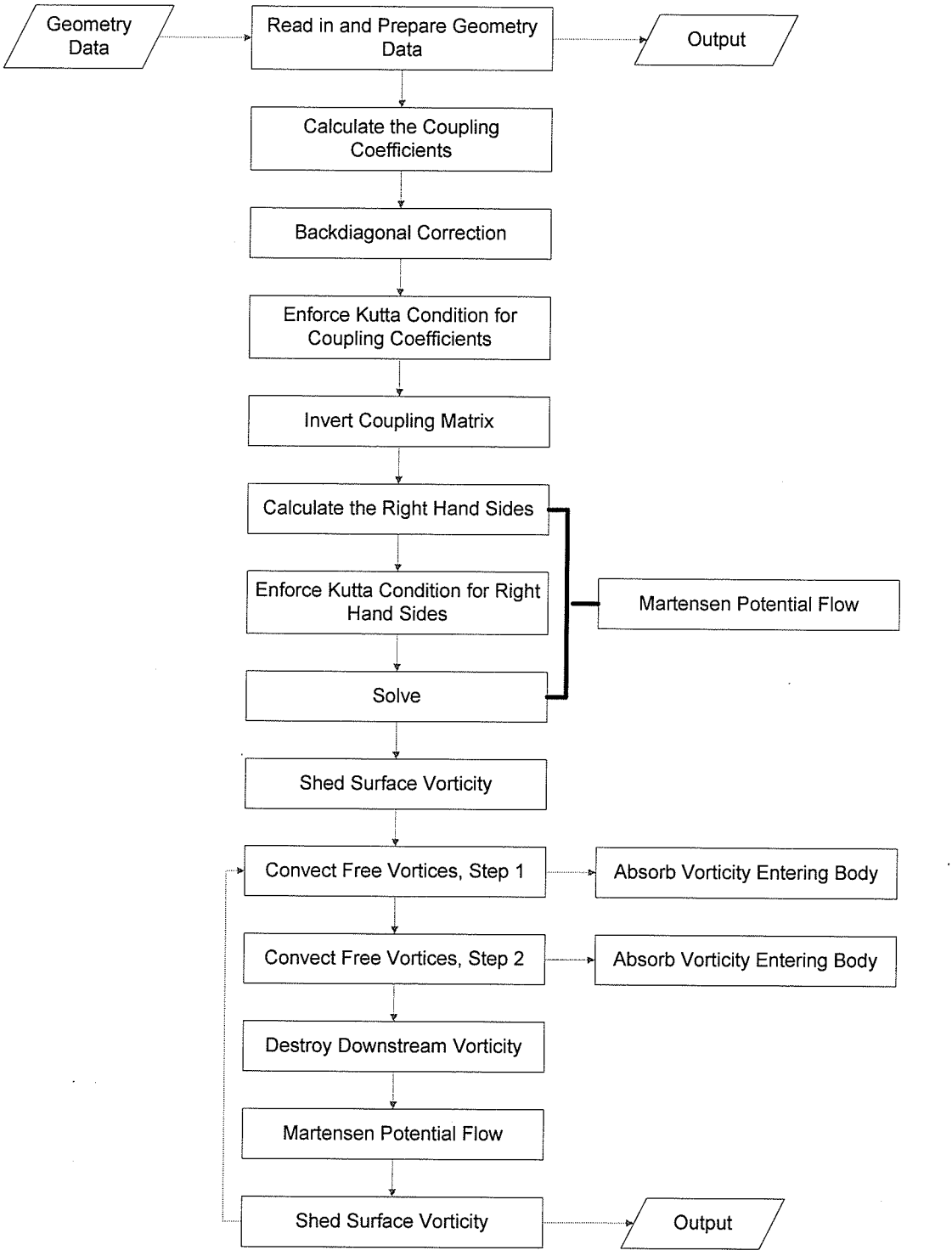


Figure 30) Flow chart for program Flowa()

3.3 Visual Vortex Cloud Program

The Flowa() program, although a full vortex cloud simulation program, was unable to show graphically the developing flow field without a large amount of post-processing by the user. It was for this reason that the Visiflow() program was written. This windows based program accepts user input and shows the developing flow field on the screen.

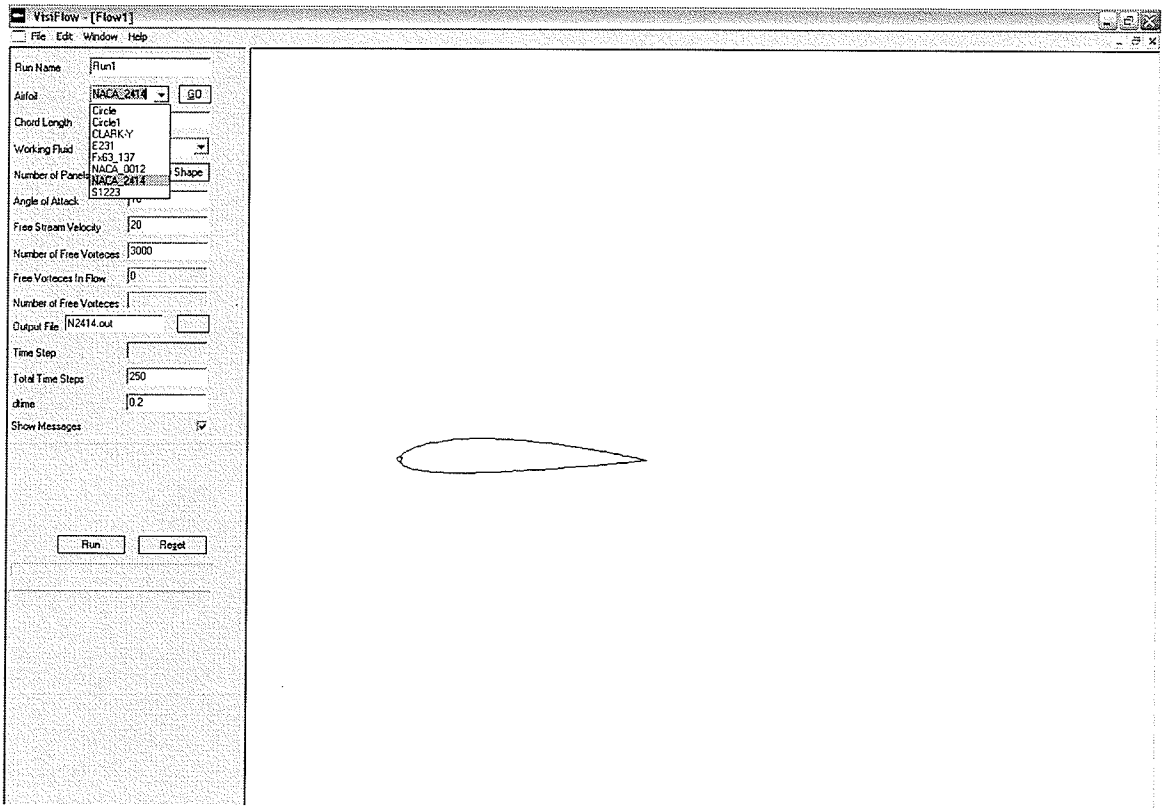


Figure 31) Screen shot of program Visiflow()

Run information is input by the user into the pane on the left hand side. The user may select the airfoil as well as the flow parameters such as the chord length and the number of panels that will describe the body surface. The

fluid used for the simulation as well as the number of free vortices and time steps is also selectable here.

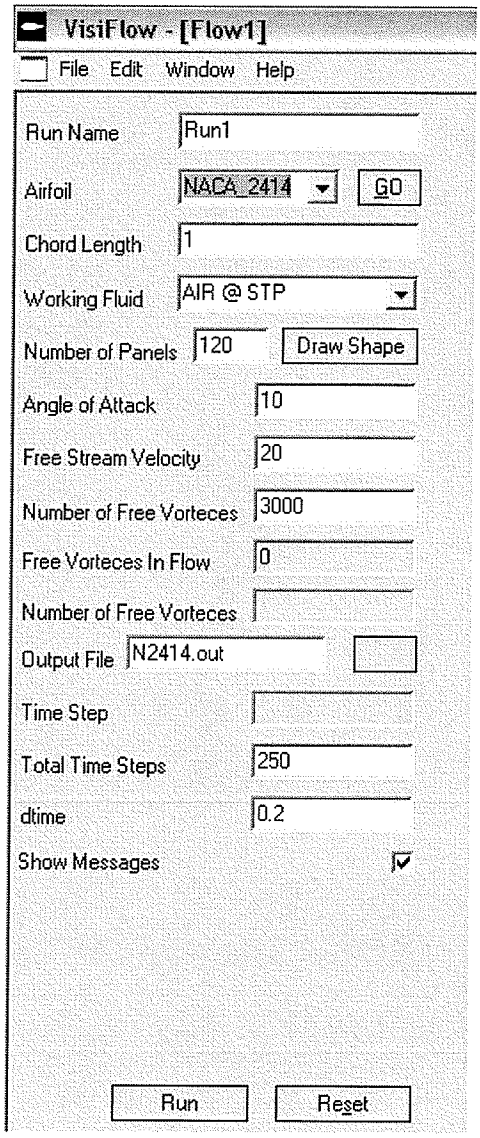


Figure 32) Input pane to set up flow simulation

The Visiflow() program was built on the main code from the Flowa() program with some modifications that were required in order to interface with the user, to accept input from the screen as well as graphically to show the flow field, as seen in Figure 32. The free vortices are represented by circles and

triangles, with the circles denoting a positive rotation, as seen in Figures 33 and 34.

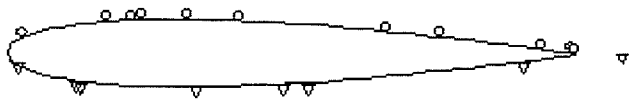


Figure 33) Vorticity shed from body surface, Visiflow() at 0°angle of attack

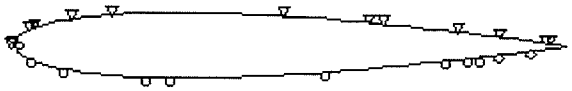


Figure 34) Vorticity shed from body surface, Visiflow() at 180°angle of attack

In the case of zero angle of attack, the vorticity shed from the underside of the airfoil is negative, as expected. When the flow field is reversed, and the flow comes from the left over the airfoil, the vorticity shed is opposite in sign. An updated flow chart for Visiflow() is shown in Figure 35.

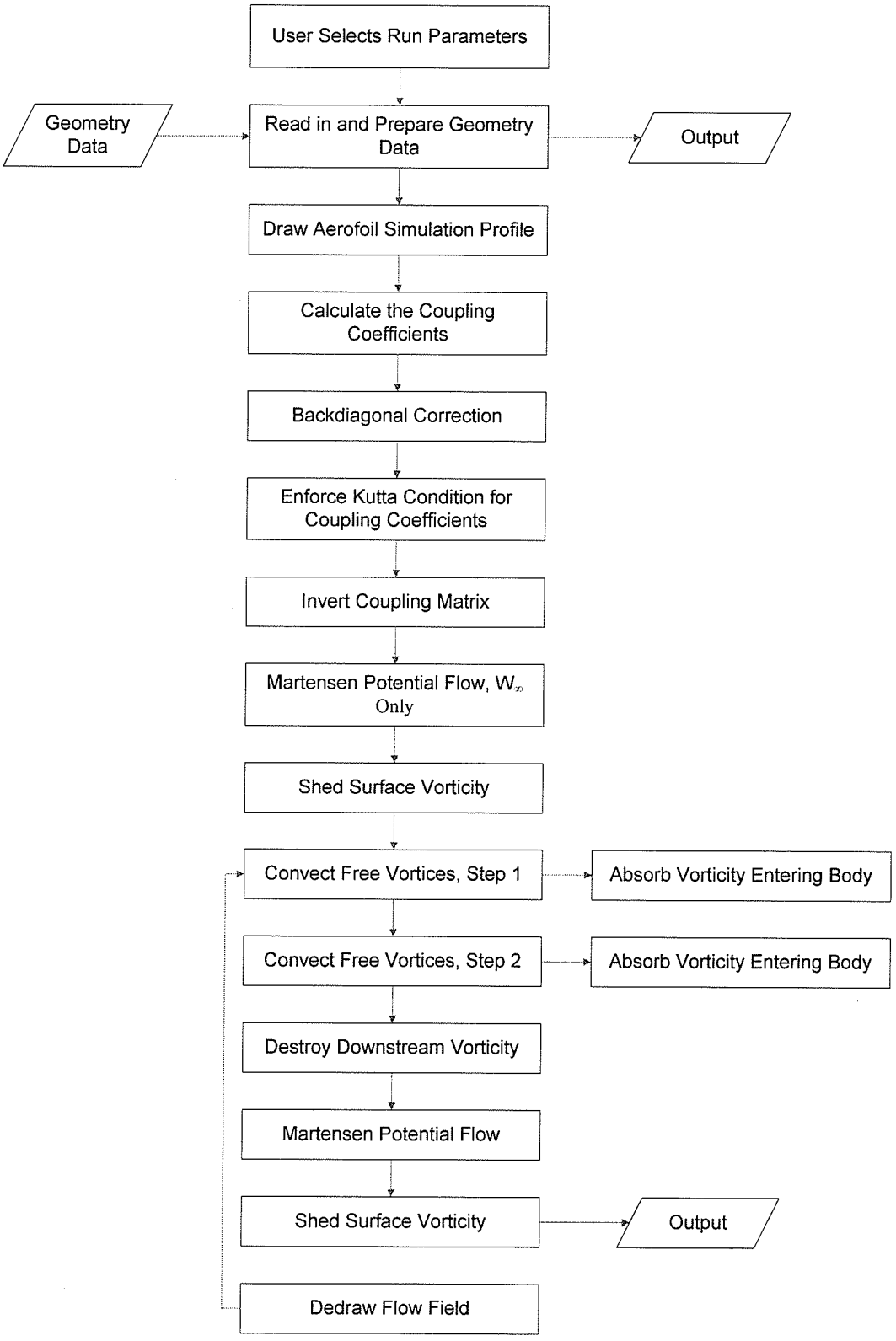
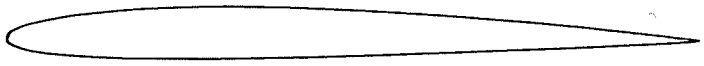


Figure 35) Flow chart for program Visiflow()

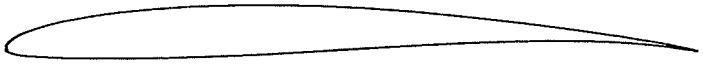
3.4.1 Test Cases

The test cases chosen were based on the availability of high quality experimental data from Selig *et al.* (1996). Selig's work details the results of a carefully constructed experimental program to produce a database of modern, low-Reynolds number airfoils. The measurements were exhaustively examined for their accuracy; for example drag was measured by the more accurate wake traverse method. This work filled a gap in previously existing databases by focusing in on low-Reynolds number and is of great use in the design of remote controlled aircraft and small unmanned aerial vehicles.

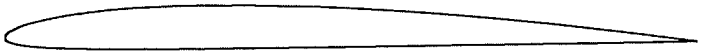
The current work focused on a series of airfoils, the NACA 2414, the FX63-137 and the CLARK-Y airfoils shown in Figure 36. These airfoils were selected for this study as a complete data set could be found in Selig's work. The data specified the ideal profile shape and the deviation of the experimental model from the ideal.



a) NACA2414



b) FX63-137



c) CLARK-Y

Figure 36) Airfoil profiles selected for this study, airfoil data from Selig *et al.* (1996)

3.4.2 Results

A variety of test simulations were run using 120 panels to model each of the selected airfoils. Example flow patterns are shown for the NACA2414 airfoil

with 250 free vortices in Figure 37 and for with 500 free vortices in Figure 38. The simulations were done for the airfoil at zero angle of attack and a Reynolds number of approximately 150,000. The figures show similar behaviour and, as expected, the free vorticity gathers into clusters in the wake of the airfoil. The results show that vortices shed from the lower surface are grouped to the lower half of the wake near the airfoil and, as we move downstream, diffuse across the wake due to interaction with other vortices and the application of random walk.

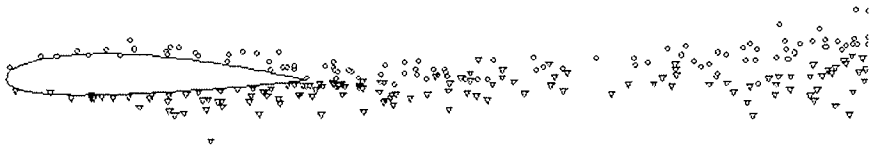


Figure 37) Vortex Cloud Model simulation of flow over a NACA 2414 airfoil at Re = 149,503, with 250 free vortices shed into the flow field.

Of concern is the number of vortices that need to be shed in order to obtain an accurate representation of the flow field.

Figure 38) Vortex Cloud Model simulation of flow over a NACA 2414 airfoil at Re = 149,503, with 500 free vortices shed into the flow field.

As is evident in Figures 37 and 38, there is good similarity of the results for both 250 and 500 shed vortices demonstrating that a sufficient number of vortices were being shed to obtain an accurate representation and a reasonable resolution of the bulk flow.

Figures 39 to 41 show the developed flow field for each of the airfoils with a Reynolds number of 200,000. In each case the airfoils are set to have zero angle of attack and 500 free vortices are active. The flow fields demonstrate several properties that indicate that the results are of good quality.

1. There are no free vortices in the body; the elimination routine is functioning as desired. The inclusion of these vortices can lead to erroneous right hand side calculations and bizarre vortex dynamics.
2. The downstream wake is directed slightly downwards which is expected for each of these non-symmetric airfoils. Also, for the FX63-137 airfoil, this behaviour is increased. As the zero lift angle for this

airfoil is greater than the other two, the results are representative of the real case.

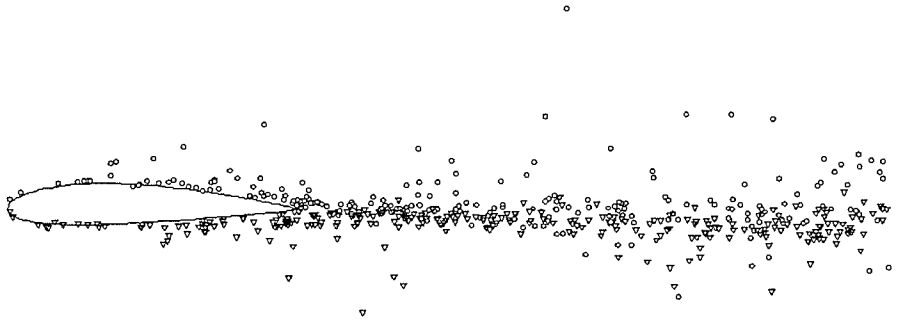


Figure 39) Vortex Cloud Model simulation of flow over a NACA2414 airfoil at Re = 199,337, with 500 free vortices shed into the flow field

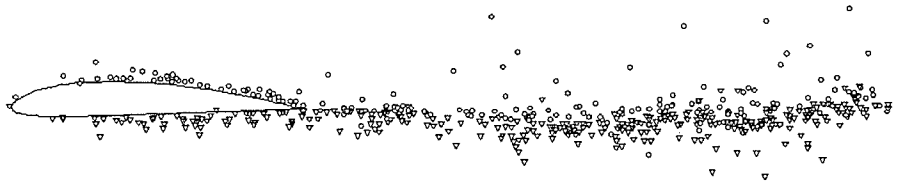


Figure 40) Vortex Cloud Model simulation of flow over a CLARK-Y airfoil at Re = 199,337, with 500 free vortices shed into the flow field

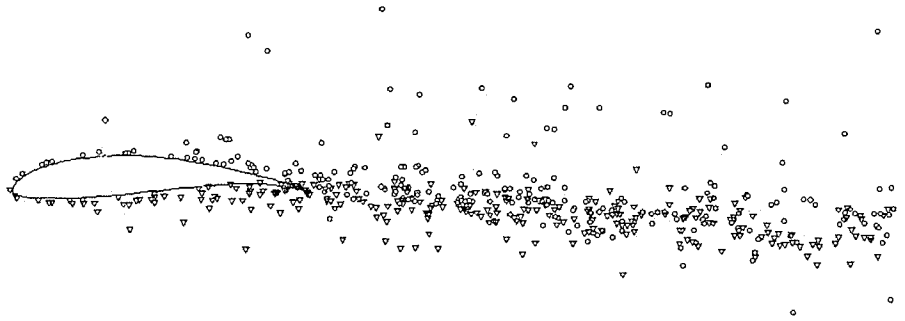


Figure 41) Vortex Cloud Model simulation of flow over a FX63-137 airfoil at Re = 199,337, with 500 free vortices shed into the flow field

In each of the above figures, there appears to be no separation of the flow from the body surface and a reasonable picture of the flow field is presented. Simulations of flow over the NACA 2414 airfoil were made for a range of angles of attack, α, and are shown in Figure 42. The Visiflow() program displays the airfoil in its horizontal orientation and the free stream velocity is inclined upward as α is increased. There is no indication of flow separation over the airfoil at $\alpha = 0^\circ$ angle of attack. When $\alpha = 5^\circ$ the wake deflected slightly upward in the direction of the free stream velocity and, as with the case of $\alpha = 0^\circ$, no separation is evident. These angles of attack are within the linear portion of the lift curve and the flow would not be expected to have significant separation regions. When the angle of attack is increased into the nonlinear portion of the lift curve, near the maximum lift region of the lift curve, say $\alpha = 10^\circ$, we would expect to see the presence of a significant portion of separated flow on the suction surface near the trailing edge. The

simulation represents this situation reasonably well with a separation zone that is approximately 40% of the chord length. Finally, the flow over the airfoil in the deep stall range, represented by $\alpha = 15^{\circ}$, clearly shows a separated zone that extends over nearly the entire suction surface, approximately 80 – 90 % of the chord length. In general, the Visiflow() output indicates that the predicted flow appears to give a good qualitative representation of the actual flow.

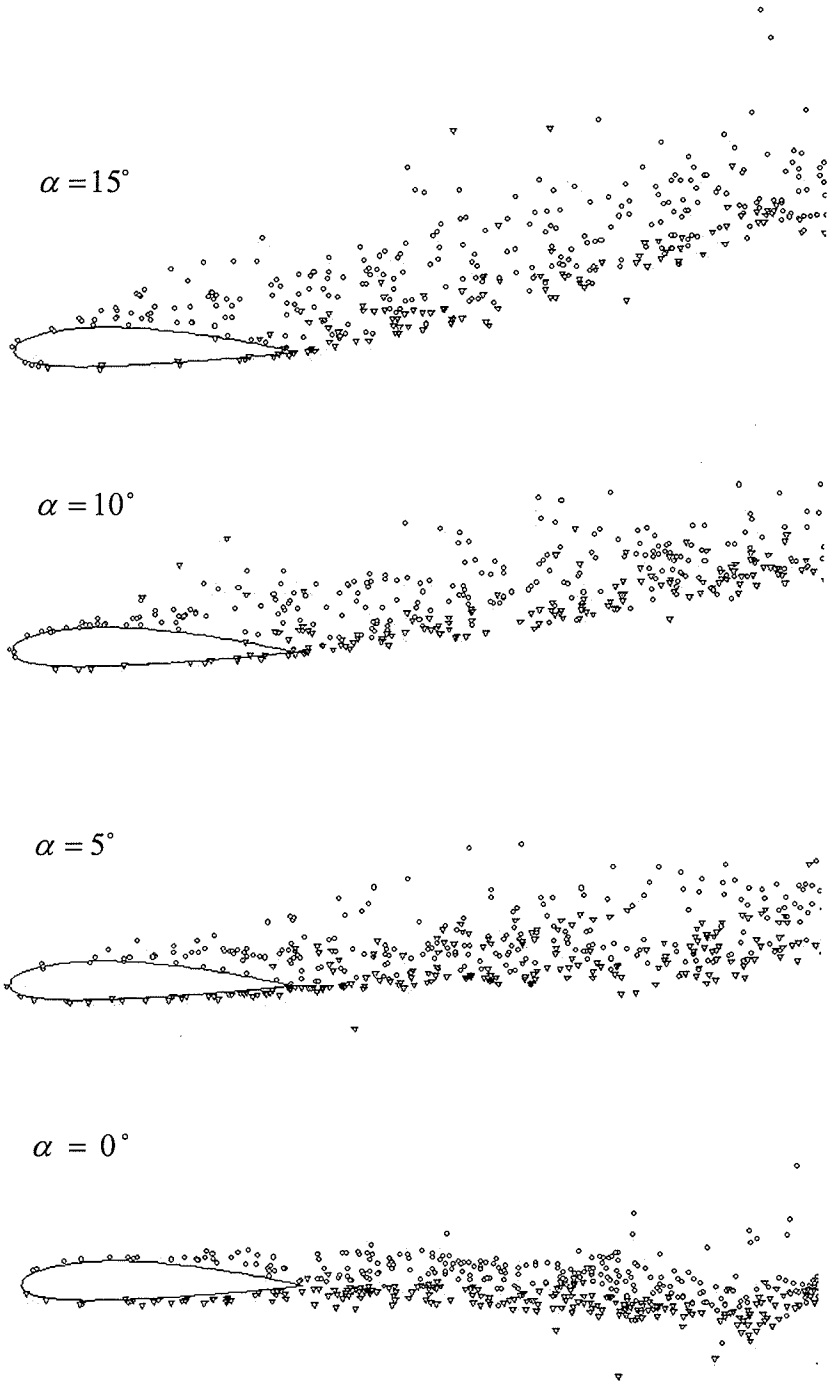


Figure 42) Vortex Cloud Model simulation of flow over a NACA 2414 airfoil at Re = 149,503, 500 vortices shed, for α = 0°, 5°, 10° and 15°.

The results in Figure 42 represent typical results from the Visiflow() program and it was deemed superfluous to include the same results for each test case. Use of the program can easily recreate the results. To demonstrate this, Figure 43 shows the CLARK-Y airfoil at 15° angle of attack. There is some separation noticed on the suction side of the airfoil as the vorticity is spread from the typically narrow wake. There is a noticeable streamline coming around the pressure side of the airfoil and resisting the free stream velocity for nearly half a chord length downstream before angling upwards.

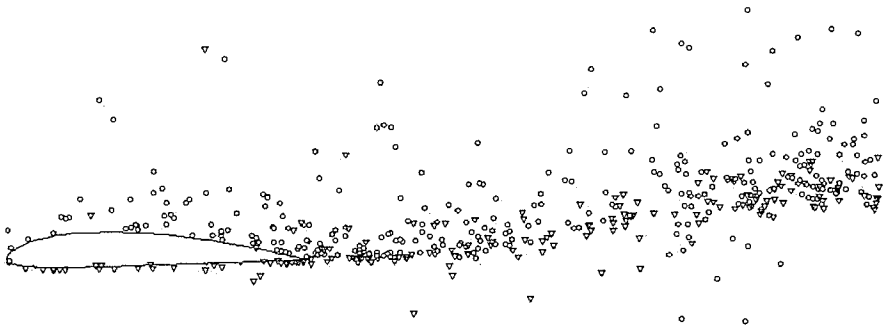


Figure 43) Vortex Cloud Model simulation of flow over a CLARK-Y airfoil at Re = 199,337, 500 vortices shed for α = 15°.

Chapter 4

Discussion and Conclusions

In this work, a vortex cloud aerodynamic flow simulation software package has been developed which graphically demonstrates the developing flow field over a body. The current work has seen the development of a vortex panel method extended to include a vortex cloud model as well as the vortex creation, interaction and destruction algorithms that are required. The algorithms developed here provide a mechanism for the observation of fluid dynamics over a body profile for low Reynolds number flows. The suitability of the following algorithms will be discussed: the calculation of aerodynamic forces, the method of vortex shedding and the visual representation of the flow.

The vortex convection algorithm, with the implementation of the modified Euler method, was shown to provide excellent prediction of vortex dynamics. Further exploration of this algorithm demonstrated the modified Euler method affords a 65% reduction in computational requirement for the equivalent accuracy when compared to the forward difference method.

The random walk algorithm and the program Vortices() showed the validity of simulated viscosity being able to reproduce vortex movement that is similar to Brownian motion. The omission of the sub-paneling routine from the current program is seen as an advantage. As previously discussed, the programming simplicity and continuity in the solver algorithm maintained by not implementing sub-paneling are two of the features that make the vortex cloud model attractive. In fact, with the speed of modern computers, sub-paneling is becoming more and more a technology of the past.

Our assessment is that the vortex cloud model is capable of simulating the flow over an airfoil at low-Reynolds numbers. However, there are several areas that need to be addressed before the method is suitable for use in an optimization routine. There is some concern about the calculation of aerodynamic forces. In the current work, the attempts made in estimating the lift and drag forces were thwarted by numerical instabilities. There are several potential causes for this difficulty:

1. The vortex cloud method is, by definition, a random process that requires averaging. It may be that a large number of time steps are required before the averages reach their asymptotic values.
2. The integration method was based on a simple trapezoid rule where a more accurate method may be required. This method did, however, generate reasonable results for a strictly inviscid estimate.
3. The act of shedding vorticity, particularly if the entire surface vorticity is shed, will cause discontinuity in the predicted, instantaneous,

surface pressure distribution. As a result, it may not be sufficiently accurate to allow for the direct computation of these quantities.

To help in the calculation of these forces, a control volume approach, where a momentum analysis is performed fore and aft of the body profile, may be more suitable for the estimation of these forces.

Furthermore, there are several issues with the vortex cloud model which need to be addressed. The vortex shedding method that was recommended by Lewis did not work as described. In that form the vorticity was wildly flung around the flow field. This necessitated a 75% reduction of shed vortex strength and the implementation of a routine that limits the induced velocity for a given vortex in order for the free vorticity to evolve through the domain in a reasonable manner. This indicates that a more formal, scientifically based method for the determination of the shed vorticity needs to be established.

Due to the difficulty realized in the calculation of the aerodynamic forces, lift and drag measurements are not available. Thus, the validity of force estimation from the random vortex displacement model could not be assessed.

The graphical user interface is an invaluable tool in the analysis of fluid flow. Real-time graphics are available to the user at any interval of the simulation whereas with traditional CFD programs post-processing is required after the entire flow field has converged. The transient nature of the algorithm is another advantage of traditional CFD codes. Eulerian flow solvers require that the flow field is fully meshed and it is typical for several iterations to be

completed in order to converge the state at each time step for a transient solution. The visual interface employed in Visiflow() allows the user to observe the more minute detail in the transient flow model as well as the more holistic characteristics such as flow separation.

The general flow characteristics predicted by the vortex cloud model were in general agreement with natural flows, the observation and prediction of flow separation being one of the key results from this work. The results shown here are also generally consistent with Lewis's (1991), there is good prediction of vortex dynamics and the wake development is graphically well represented.

The vortex cloud model does have some significant advantages as a flow simulation tool. The grid free nature of the Lagrangian vortex tracking method allows a grid free approach. A significant amount of mesh generation tool development is eliminated. The lack of computational mesh also reduces the discretization procedure from full mesh generation to simple surface panel generation. This has real advantages when used in an optimization scheme. It allows for quick geometrical changes with no change to the fluid domain. As the optimization process typically requires the simulation of a large number of candidate airfoils, it will benefit greatly from the reduction of simulation effort.

The application of the method, aerodynamic forces not withstanding, was simple to implement into a vortex panel method. The model is also relatively fast, making it quite competitive with other methods. The vortex cloud model has merit in that it is a quick tool that gives, at least visually, a good

representation of the flow field for low Reynolds number flows and is suitable, if not ideal, for use in an aerodynamic optimization scheme.

Chapter 5

Recommendations

Future development is required in several areas before this algorithm can be implemented into a full scale optimization program. The issue most critical to the advancement of this routine is the calculation of aerodynamic forces. Without a validated method to predict the performance of the body under investigation, there are few applications where this algorithm will find use. The program Visiflow() does require some extension before it can be used as a stand-alone, autonomous program. First, there is currently no check for aerodynamic convergence. Although watching the flow field develop is interesting, the aerodynamic forces calculated while the flow is developing will be unstable and unusable for averaging. It is necessary to start the averaging of forces once the code determines that the domain is sufficiently stable. Convergence for a transient solver is difficult to determine and therefore the author recommends the use of the Strouhal number in order to determine when the simulation has become sufficiently stable to commence the performance analysis. The Strouhal number can identify the flow

periodicity and provides a good time scale for averaging. The width of this time scale can also be used to generate output graphics automatically at predetermined, user-defined intervals. For example, if the user would like twenty pictures generated for a simulation, the time scale divided by this number will tell the program at which time steps it must generate this output.

There is also some real opportunity for improvement in the algorithm that sheds vorticity. In the current state, this algorithm has a methodology of questionable stability. If shed vorticity is not opposite in sign to the accumulated circulation, it is possible for this routine to cause the program to stall in the function as it tries to satisfy an inappropriate condition. Because the random() function used to select the panel to shed vorticity has an average output of 0.50 there is a 50/50 chance that the shed vorticity will be opposite in sign from the accumulated circulation. Because of this, the accumulated vorticity should be approximately equal to the average free vorticity, or 0.0. The current implementation sheds only a portion of the total surface vorticity and therefore the risk of stall is small but present. A more suitable method of retaining a portion of surface vorticity while ensuring zero net circulation is a scaled approach. It is possible to implement a scaled vortex shedding process where the computer can target the net circulation for shed vorticity and allow for more than the preset amount of surface vorticity to be shed at any instant. This will compromise the user-defined limit on shed vorticity but will ensure that the maximum number of free vortices shed during the time step is enforced. Alternatively,

the order of solution might be adjusted so that, if all of the surface vorticity is shed and the noticed instabilities can be resolved, the surface vorticity is regenerated before the convection processes, i.e. the calculation of the right hand sides is completed before the convection routine. This would effectively eliminate the discontinuity in surface velocity that arises from shedding the entire surface vorticity.

The process of matrix inversion is a known source of numerical instability and some exploration should be made into alternative solution methods. Although this would mean that the matrix would have to be solved during each time step as opposed to the current work where the matrix is inverted only once, the speed of modern computers make this type of code optimization less of a consideration than in the past.

The graphical user interface could be extended to provide additional information to the user, such as plotting the pressure distribution at each logical interval or providing additional controls for the simulation.

There is also some opportunity for optimization of the current program as Visiflow() is the amalgamation of four other programs. Algorithms may be able to be simplified and the memory allocation for the program should be examined. In the current implementation there was no investigation into leveraging the increased accuracy of the modified Euler method for a longer time step. This might reduce computational time as fewer calculations would be required for the equivalent movement. The time step used in the analysis also is somewhat arbitrary and does not consider all of the implications of its size. Effort should be made to have each section of the code consider the

code consider the effect of specifying a global factor. An example of this would be the timescale calculation. If it is scaled to the velocity, the program should also consider limits in order to ensure a user-specified accuracy or resolution.

Bibliography

ABBOTT, I. and VON DOENHOFF, A. E., **Theory of Wing Sections.** *Dover Publications*, New York, NY, 1959.

ABERNATHY, F. H. and KRONAUER, R. E., The **Formation of Vortex Streets.** *J. Fluid Mech*, Vol. 13, 1962, pp. 1-20.

ANDERSON, J.D., **Fundamentals of Aerodynamics.** *McGraw-Hill*, New York, NY, 2001.

BARBER, R. W. and FONTY, A., **Numerical Solution of Low Reynolds Number Flows Using a Lagrangian Discrete Vortex Method.** http://www.cse.clrc.ac.uk/ceg/misc/vortex/vortex.shtml, 2005.

BARBA, L. A., LEONARD, A. and ALLEN, C. B., **Vortex Method with Meshless Spatial Adaption for Accurate Simulation of Viscous, Unsteady Vortical Flows.** *Int. J. Numer. Meth. Fluids* Vol. 47, 2005, pp. 841–848.

BASUKI, J. and GRAHAM, J. M. R., **Discrete Vortex Computation of Separated Airfoil Flow.** *AIAA Journal*, Vol.25 1987 pp. 1409-1410.

BATCHELOR, G. K., **An Introduction to Fluid Dynamics.** *Cambridge University Press*, 1970.

BEALE, J. T. and MAJDA, A., **Rates of Convergence for Viscous Splitting of the Navier-Stokes Equations.** *Mathematics of Computation*, Vol. 37, No. 156, 1981, pp. 243-259.

BORTHWICK, A. G. L. and BARBER, R. W., **Numerical Simulation of Jet-Forced Flow in a Circular Reservoir Using Discrete and Random Vortex Methods.** *International Journal for Numerical Methods in Fluids* Vol. 14, No. 12, 1992, pp. 1453-1472.

CARR, L. W., McALISTER, K. W. and McCROSBAY, W. J., **Analysis of the Development of Dynamic Stall Based on Oscillating Airfoil Experiments.** *NASA* TN D-8382, 1977.

CHAN, C. K., LAU, K. S. and ZHANG, B. L., **Simulation of a Premixed Turbulent Flame with the Discrete Vortex Method.** *International Journal for Numerical Methods in Engineering,* Vol. 48, 2000 pp. 613-627.

CHEN, B., GUO, L. J. and YANG, X.G., **Numerical Simulation of Flow Around Circular Cylinder by Discrete Vortex Method.** *Progress in Natural Science*, Vol. 12, 2002, pp. 964-969.

CHORIN, A. J. and BERNARD, P. S., **Discretization of a Vortex Sheet, with an Example of Roll-Up**. *J. Computational Physics* Vol. 13, No. 3, 1973.

CHORIN, A. J., **Numerical Study of Slightly Viscous Flows**. *J. Fluid Mech.* Vol. 57, 1973, pp. 485.

CHORIN, A. J., **Vortex Sheet Approximation of Boundary Layers**. *J. Computational Physics.* Vol. 27, No. 3, 1978.

CLEMENTS, R. R., **An Inviscid Model of Two-Dimensional Vortex Shedding.** *J. Fluid Mech.* Vol. 57, 1973, pp. 321–336.

CORTELEZZI, L., CHEN, Y. C. and CHANG, H. L., **Nonlinear Feedback Control of the Wake Past a Plate: From a Low-Order Model to a High-Order Model**. *Phys. Fluids,* Vol. 9, 1997, pp. 2009–2021.

COTTET, G. H. and KOUMOUTSAKOS, P. D., **Vortex Methods. Theory and Practice.** *Cambridge University Press*, New York, N.Y., 2000.

COTTET, G. H., OULD SALIHI, M. L. and EL HAMRAOUI M., **Multi-purpose Regridding in Vortex Methods.** *ESAIM Proceedings, Giovannini A et al. (eds),* Vol. 7. Societe de Mathematiques Appliques et Industrielles, 1999, pp. 94–103.

DERKSEN, R. W. and RIMMER, J., **Aerodynamic Flow Simulation**, *WIT Transactions on Engineering Sciences*, Vol. 52, 2006.

DIDDEN, N., **On the Formation of Vortex Rings: Rolling-up and Production of Circulation**. *Z. Angew. Math. Phys*. 30, 101 1979.

ERICKSON, L.L., **Panel Methods — An Introduction.** *NASA TP-2995*, 1990.

FROHLICH, J. and RODI, W., **Introduction to Large Eddy Simulation of Turbulent Flows.** *Closure Strategies for Turbulent and Transitional Flows, B. Launder and N. Sandham, eds., Cambridge University Press, Cambridge*, 2002, pp. 267–298.

GERRARD, J. H., **Numerical Computation of the Magnitude and Frequency of the Lift on a Circular Cylinder.** *Phil Trans. Roy. Soc. A.*, Vol. 261, 1967, pp. 137-162.

HESS, J. L., **Panel Methods in Computational Fluid Dynamics.** *Annual Review of Fluid Mechanics,* Vol. 22, 1990, pp. 255-274.

HESS, J.L., **Linear Potential Schemes.** *Applied Computational Aerodynamics, P.A. Henne, ed., AIAA*, Washington, 1990, pp. 21-36.

JACOB, K. and RIEGELS, F. W., **The Calculation of the Pressure Distributions over Aerofoil Sections of Finite Thickness with and without Flaps and Slats.** *Z. Flugwiss,* Vol. 11, No. 9, 1963, pp. 357-367.

JONES, M.A., **The Separated Flow of an Inviscid Fluid Around a Moving Flat Plate.** *J. Fluid Mech.,* Vol. 496, 2003, pp. 405–441.

KAO, H.C., **A Note on Trapping Moving Vortices.** *Glenn Research Center*, Cleveland, Ohio, July 2000.

KATZ, J., **A Discrete Vortex Method for the Non-Steady Separated Flow Over an Airfoil.** *J. Fluid Mech.* Vol. 102, 1981, pp. 315–328.

KATZ, J., and PLOTKIN, A., **Low-Speed Aerodynamics From Wing Theory to Panel Methods.** *McGraw-Hill, Inc.*, New York, NY, 1991.

KELLOG, O., **Foundations of Potential Theory.** *Frederick Ungar Publishing Corp.*, New York, NY, 1929.

KOUMOUTSAKOS, P., **Simulation of Unsteady Separated Flows using Vortex Methods.** *Ph.D. Thesis, California Institute of Technology*, 1993.

KRASNY, R., **Vortex sheet computations: roll-up, wakes, separation.** *Lectures in Applied Mathematics. Am. Math. Soc.* Vol. 28, 1991, pp. 385–401.

KUWAHARA, K., **Numerical Study of Flow Past an Inclined Flat Plate by an Inviscid Model**. *J. Phys. Soc. Japan* Vol. 35, 1973, pp. 1545.

LIU L., JI, F., FAN J., and CEN K., **Recent Development of Vortex Method in Incompressible Viscous Bluff Body Flows.** *Journal of Zhejiang University SCIENCE* Vol. 6A No. 4, 2005, pp. 283–288.

LEWIS, R.I., **Vortex Element Methods for Fluid Dynamic Analysis of Engineering Systems.** *Cambridge University Press*, New York, N.Y. 1991.

MARTENSEN, E., **Berechnung der Druckverteilung an Gitterprofilen in ebener Potentialstromung mit einer Fredholmschen Integralgleichung.** *Arch. Rat. Mech., Anal. 3*, 1959, pp.235-270.

MILEY, S. J., **A Catalogue of Low Reynolds Number Airfoil Data for Wind Turbine Applications.** *Department of Aerospace Engineering, Texas A&M University, College Station, Texas,* RFP-3387, UC-60, 1982.

MOORE, D. W., **The Discrete Vortex Approximation of a Finite Vortex Sheet.** *California Institute of Technology Report* AFOSR-1804-69. 1971.

NAKASHIMA, M. and ONO, K., **Numerical Study of the Thrust, Energy Consumption, and Propulsive Efficiency of a Three Joint Bending Propulsion Mechanism.** *Journal of Fluids Engineering* Vol. 122, 2000, pp. 614-618.

NITSCHE, M. and KRASNY, R., **A numerical study of vortex ring formation at the edge of a circular tube.** *J. Fluid Mech.* Vol. 97, 1994, pp. 239–255.

NYIRI, A. and BARANYI, L., **Numerical Method for Calculating the Flow around a Cascade of Aerofoils.** *Proc. Of VII International J.S.M.E. Symposium of Fluid Machinery and Fluidics,* Tokyo, Vol. 2, 1983.

PEREIRA, L. A. A., HIRATA, M. H. and SILVEIRA NETO, A., **Vortex Method with Turbulence Sub-Grid Scale Modelling.** *J. Braz. Soc. Mech. Sci. & Eng*, Vol. 25, No. 2, 2003, pp.140-146.

PFEIFFER, N. J. and ZICKUHR, T. D., **Validation of Computational Aerodynamics Applied to General Aviation Configurations.** *Proc. of 7th Applied Aerodynamics Conference*, AIAA-1989-2169, 1989.

PORTHOUSE, D. T. C. and LEWIS, R. I., **Simulation of Viscous Diffusion for Extension of the Surface Vorticity Method to Boundary and Separated Flows.** *J. Mech. Eng. Science, I. Mech. E.* Vol. 23, No. 3, 1981, pp.157-167.

PORTHOUSE, D. T. C., **Numerical Simulation of Aerofoil and Bluff Body Flows by Vortex Dynamics.** *Ph.D. Thesis, Univ. of Newcastle up Tyne*, 1983.

PLOUMHANS, P. and WINCKELMANS G. S., **Vortex Methods for High-Resolution Simulations of Viscous Flow Past Bluff Bodies of General Geometry.** *Journal of Computational Physics,* No. 165, 2000, pp. 354–406.

ROSENHEAD, L., The **Formation of Vortices from a Surface of Discontinuity.** *Proc. Roy. Soc. A.*, Vol, 134, 1931, pp. 170-192.

SARAPKAYA, T., **An Inviscid Model of Two-Dimensional Vortex Shedding for Transient and Asymptotically Steady Separated Flow over an Inclined Plate.** *J. Fluid Mech.* Vol. 68, 1975, pp. 109–128.

SELIG, M. S., LYON, C. A., GIGUERE, P., NINHAM, C. P. and GUGLIELMO, J. J., **Summary of Low Speeds Airfoil Data**. *Soartech Publications, Virginia Beach, U.S.A.*, Vol. 2, 1996.

SPALART, P. R., LEONARD, A. and BAGANOFF, D., **Numerical Simulation of Separated Flows.** *NASA Technical Memorandum 84328*, 1983.

TAKAMI, H., **Numerical Experiment with Discrete Vortex Approximation, with Reference to the Rolling Up of a Vortex Sheet.** *Dept. of Aero. And Astro., Stanford University*, Report SUDAER 202, 1964.

VAMOS, C. and SUCIU, N., **Global Random Walk Simulations of Diffusion.** *Scientific Computing, Validated Numerics, Interval Methods,* Plenum Publishing, New York, NY, 2001, pp. 343-354.

VAMOS, C., SUCIU, N. and VEREECKEN, H., **Global Random Walk Simulations of Diffusion.** *Journal Comp. Phys* Vol. 186, 2003, pp. 527-544.

WILKINSON, D.H., **A Numerical Solution of the Analysis and Design Problems for the Flow Past One or More Aerofoils or Cascades.** *A.E.C., R&M.* No. 3545, 1967.

WILKINSON, D.H., **The Analysis and Design of Blade Shape for Radial, Mixed and Axial Turbomachines with Incompressible Flow**. *M.E.L. Report No. W/M(3F), English Electric Co.,* Whetstone Leister, 1969.

XU, C., **Surface Vorticity Modeling of Flow Around Airfoils.** *Computational Mechanics*, Vol. 21, No. 6, 1998, pp. 526 – 532.