MODELING OF STEADY CONJUGATE HEAT TRANSFER PROCESS ON AN AIRFOIL SHAPED OBJECT IMMERSED IN LAMINAR FLUID FLOW

BY

MIODRAG JOJIC

A Thesis

Submitted to the Faculty of Graduate Studies

In Partial Fulfillment of the Requirements for the Degree

of

MASTER OF SCIENCE

Department of Mechanical and Industrial Engineering

University of Manitoba

Winnipeg, Manitoba

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

MODELING OF STEADY CONJUGATE HEAT TRANSFER PROCESS ON AN AIRFOIL SHAPED OBJECT IMMERSED IN LAMINAR FLUID FLOW

BY

MIODRAG JOJIC

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

ACKNOWLEDGMENTS

It is my pleasure to express my deepest gratitude to my supervisor Dr. Rob W. Derksen for his mentorship, guidance, and support while pursuing this degree. His impressive knowledge, valuable technical advice, and encouragement throughout the course of my research studies helped me enormously. He has left a profound, lifelong influence on me not only as a wonderful faculty advisor but also as a good person. His patience and generous assistance is greatly appreciated.

I sincerely thank professors Dr. Mark Tachie and Dr. Abba Gumel for being my committee members.

I am also grateful to The Boeing Canada – Winnipeg for their financial support of this research. The company support, encouragement, and flexibility allowed me to accomplish my goals.

Finally, I would like to thank my mother Danica and my sisters, Brankica and Slavica, for their encouragement, sincere care and constant support during my studies. Their unconditional love truly helped me to overcome all the difficulties that I encountered.

I owe a special debt of gratitude to my sister Brankica without whose assistance and understanding I could not have done this thesis.

August, 2003

CONTENTS

List of Figures

List of Tables

Abstract

Modeling of Steady Conjugate Heat Transfer Process on an Airfoil Shaped Object Immersed in Laminar Fluid Flow

In a practical heat transfer process, interaction between heat transfer by forced convection and conduction through the bodies immersed in the flow is shown to be very important. The modeling of a steady conjugate heat transfer process between an aerofoil shaped object and a forced stream is explored within this thesis.

The solution to the problem is composed of the following primary task areas, each requiring mathematical definition and one or more computer subroutines along with necessary interfacing architecture suitable to digital computations. These are:

- Potential flow solution
- Boundary layer solution
- Heat transfer by conduction within a body
- An iterative procedure that combines solution of fluid flow and conduction analysis

A two-dimensional incompressible potential flow past an aerofoil shaped object is analyzed using the surface vorticity boundary integral method. An implicit finite difference method was applied to obtain detailed boundary-layer characteristics which were provided as output through solving the boundary-layer equations. Complete viscid characterization of the airfoil was obtained through utilization of an iterative technique to combine boundary-layer and potential flow solutions. The results of both, the potential solution and complete viscid solution, were compared with classical exact and experimental solutions in terms of the velocity distribution around an airfoil and the drag coefficient.

Since the convection transfer is determined by boundary layers that develop on the surface, the convection heat transfer coefficient and heat flux were calculated from the

wall temperature gradient $\delta T/\delta y|_{y=0}$ (where T is the temperature on the wall and y is the direction normal to the wall) which was obtained through the implicit finite difference procedure for solving boundary layer energy equations.

The steady-state heat conduction problem was treated through the application of a boundary element method. Satisfaction of the convection type boundary conditions produced a system of equations that was solved in order to find the boundary unknowns, the temperature on the surface of the immersed body. The solution of the conjugate heat transfer process over a surface of the body, as a converged surface temperature, was obtained through an iterative procedure combining heat transfer by conduction across the body and forced convection.

Comprehensive presentation of the converged surface temperature, heat flux on the surface, and domain of applicability of the coupled procedure in terms of conductivities ratio coefficient CR are provided. The results of the study indicate that the conjugate heat transfer program can provide valuable insight into all aspects of fluid flow, the convective heat transfer, and the body heat conduction analysis on an airfoil shaped object in a fluid flow.

A natural framework is provided in which the existing program capabilities can readily be extended into turbulent boundary layer flow or compressive flow problem areas for optional shape bodies and various angles of incidence.

1 Introduction

1.1 Problem Statement

The present work deals with the analysis of the conjugate heat transfer on an airfoil shaped object, with a cooling duct within the interior of the solid, immersed in a forced convection laminar boundary layer. The object of the study is interaction, applicability, and coupling of different numerical analysis methods in conjugate heat transfer analysis. Additional goal would be a reliable numeric procedure producing correct solutions for the problem analyzed. Procedure would establish methodology and a direction for analysis of a more complex and generally curved object and emphasize issues that would have to be resolved if such an object is exposed to conjugate heat transfer within a fluid flow different from the one analyzed here. The steady-state numerical analysis applied would show the evolution of the surface temperature of a body washed by forced laminar boundary layer. One of the relevant phenomena in this case is the appropriate modeling of a potential flow that would produce the inviscid velocity distribution around a body. As it will be shown, the results of the surface vorticity method of the potential flow analysis have a huge influence on the solution of velocity boundary layer analysis because the inviscid velocity distribution is used as a boundary condition in boundary layer modeling. While the boundary layer equations are solved by an implicit finite difference marching technique, the two-dimensional heat conduction problem is solved through an application of the boundary element method. Accordingly, the governing partial differential equations defining the thermal boundary layer of flowing fluid and the boundary integral equations, from which the surface temperature boundary values can be

found, are solved simultaneously satisfying the heat flux and the temperature at the interface.

1.2 Literature Review

Conventional heat transfer analyses related to similar applications are based on the assumption of a uniform surface temperature or the surface heat flux. There are several works that analyzed the effect of wall heat conduction and convective heat transfer or conjugate heat transfer process over surfaces.

Luikov (1974) and Payvar (1977) analyzed the conjugate problem where the lower surface of a horizontal flat plate is maintained at uniform temperature, while at the upper surface of the plate, heat is convected to a laminar boundary layer. Luikov (1974) developed two approximate solutions, one based on a differential analysis with low Prandtl number and the second based on an integral analysis with assumed polynomial forms for the velocity and temperature profiles and suggested certain formulae for the calculation of local Nusselt number. Jilani at all (2002) presented conjugate forced convection-conduction heat transfer analysis of a heat generating vertical cylinder. They used a finite difference scheme to resolve both the steady two-dimensional conduction equation for heat generating cylinder and steady two-dimensional laminar boundary layer equations for the flowing fluid. They presented results for a wide range of conduction-convection, heat generating parameters, and length to diameter ratios for a specific fluid having Prandtl number 0.005. The transient conjugate heat transfer process between two laminar counterflowing forced streams separated by a wall with finite thermal conductivity was analyzed by Trevino at all (1997). They used asymptotic analysis and presented a classification of the solutions of the problem in terms of two main parameters: the ratio of thickness to the height of the wall and the ratio of thermal resistance of one of the boundary layers to the thermal resistance of the wall. Nakayama and Park (1996) analyzed conjugate heat transfer from a single surface-mounted block to forced convective airflow in a parallel-plate channel experimentally and analytically. The experimental data of temperature and heat transfer coefficient were used in setting the boundary conditions for numerical analysis of conductance for different heat flow paths.

The heat conduction analysis code was then used to find the heat transfer capability of various block-support/floor combinations. Krishna (1996) studied conjugate heat transfer in a finned heat sink with power transistors. He analyzed conjugate fluid-heat transfer problem using finite element NISA/3D-FLUID family of programs to obtain temperature distribution in the sink and the fluid. Velocity vectors in the fluid were also obtained. Short (2003) obtained numerical model of conjugate heat transfer in a cast pin fin coldwalls for electronic system cooling system. The three-dimensional numerical solution utilizes SIMPLE algorithm. The conjugate heat transfer problem is resolved with constant heat flux applied to a heated wall while the top wall bounding the flow is insulated. Sparrow and Chyu (1982) analyzed conjugate heat transfer for a vertical plate fin in a forced convection laminar boundary layer flow. Heat transfer in the fin was resolved as one-dimensional conduction. The results they obtained from numerical solution were compared with those from conventional methods.

Coupling of two-dimensional heat conduction in the solid and the analysis of forced convective fluid flow were not analyzed in many of those studies. In addition to that, only Jilani at all (2002) and Krishna (1996) presented problems with heat source or heat sink within a solid. The assumption that there is no heat generation or absorption is certainly good for some but not for all applications. For instance, the problem of conjugate heat transfer on a turbine blade with no internal cooling ducts is not as general as the problem of conjugate heat transfer on a turbine blade with a heat sink or a heat source within the body of the blade. The advantage of having a procedure resolving the problems with heat generation or absorption within a solid is that the procedure could probably produce a solution for the problems without heat sink or source just by changing the input to the procedure without changing the procedure or code itself. Such a general procedure would take into account radial heat conduction in addition to axial heat conduction. Additionally, most of the analyses that can be found in the literature are limited to either horizontal, vertical flat plates, or other objects not changing the curvature of a surface along the fluid stream, which significantly complicates fluid flow analysis and verification of its results.

As already stated within the Section 1.1, the present work deals with the analysis of the two-dimensional conjugate heat transfer on an airfoil shaped object with relatively complex geometry compared to horizontal or vertical flat plates that were analyzed in the above studies. The analysis performed in this study is more general and complex than the analysis along the longitudinal axis of a heat generating vertical cylinder as presented by Jilani at all (2002). Furthermore, there is a heat sink or a heat source within the body of the airfoil shaped object. Coupling and usage of different numerical methods in the conjugate heat transfer analysis distinguish this study even further from the other studies in the same field.

Since the subject of the work presented in this study is the conjugate heat transfer process on a curved object immersed in the laminar boundary layer fluid flow, the studies referenced in this section were done within the same filed, the laminar boundary layer analysis. The conjugate heat transfer process on objects immersed in the turbulent boundary layer and numerical models of the turbulent boundary layer were not considered. However, the procedure and the computer code created in this work are the groundwork to issues such as the application and implementation of numerical solvers to the turbulent boundary layer and to the transition region in between laminar and turbulent boundary layers.

1.3 Motivation

The analysis of interaction between conductive and convective heat transfer, on the bodies immersed in the flow, is very important due to the existence of these simultaneous effects in the practical heat transfer process. Therefore, the analysis of heat transfer process to which we refer as conjugate heat transfer requires coupling of convection in the fluid and conduction through a solid. The condition that facilitates the coupling is the continuity in temperature and heat flux at the fluid-surface interface. One of the classical examples of conjugate heat transfer process, that exist or is applicable to many engineering devices, is a heat exchanger. The design and effectiveness of these devices, in which the conduction in a solid tube, a fin, or a plate greatly depends on the convection

in fluid flowing over it, would be a field where results of conjugate heat transfer would be highly applicable. The key of coupling the conduction within a solid and the convection in the fluid surrounding the solid is that they have to be simultaneously analyzed. The result of the conduction analysis becomes the boundary condition for the convection and vice versa.

Another good application of a conjugate heat transfer analysis would be any element or a piece of a surface from which the internally generated heat must be dissipated and carried away by a stream of coolant or air passing over it. There are numerous examples of such elements among electronic devices. A computer CPU would be one that everybody is familiar with. Such analysis would be useful for the device level analysis of electronic assemblies in redesigning heat sinks, choosing optimum heat sink configurations, and dimensions for a particular application, and for optimal component placement.

The conjugate heat transfer process is a dominant phenomenon in the cure process of the manufacturing of epoxy based matrix composite structures with reinforcing fibers such as those found in aerospace industry. The cure typically takes place in an autoclave, that is a large pressure vessel with a convective heat transfer environment. Composite parts are of various shapes, as simple as a flat plate to a very complex three-dimensional curvature surfaces. As a part of a research project into curing the composite parts and the epoxy resins, the temperature on the surface of a part and within the part would be required. This temperature is sometimes a function of time and sometimes is not. Among the other aspects of the cure process, mechanical properties of the composite material and the quality of a part are predominantly determined by the curing process or more precisely by temperature distribution and the rate of heat absorption within the part. This would necessitate a detailed conjugate heat transfer analysis of a composite part washed by a forced flow. The following relevant assumption would have to be utilized in this analysis:

- flow is two-dimensional, and could be a transient or a steady-state problem in laminar or turbulent flow
- pressure drop could be present along the length of the panel

- temperature outside the boundary layer is equivalent on both sides of the panel and tool, but heat transfer will vary pending on boundary layer conditions and geometry of the part
- nitrogen within autoclave could be considered an ideal gas and conductivity and viscosity are not functions of pressure
- the presence of either laminar or turbulent boundary layer, or both of them with a transition region could be assumed
- composite parts lay up tools are not at uniform temperature since each of them has a significant support structure

Due to required complex and time demanding analysis, accompanied with experimental results needed to verify the results, such a huge task will not be attempted. The more suitable conjugate heat transfer analysis of internally cooled turbine blade, employing the same methods and principles, will be conducted.

In particular, the design, cooling, and performance of an airfoil offers an excellent opportunity to analyze these complex physical phenomena. Airfoils have relatively complex geometry demanding potential flow analysis whose solution can be readily verified due to existence of numerous experimental data on airfoil performance characteristics. Airfoil cooling is an area where significant research work has been performed due to aerospace industry's high demand for its application, Ref. 9. Previous work provides knowledge to build upon, simultaneously with empirical and experimental data that new work can be compared. Due to the high demand for airfoil shaped and other simpler geometry details cooling, the methodology, knowledge, and results of the work on such a problem could find wide application on many variations on analyzed geometries.

The design of an airfoil cooling system would be an iterative process with initial aerodynamic design including airfoil contours. The arrangement of internal cooling ducts, their geometries, cooling flows and temperatures would be also given. Configuration would be changing, based on conjugate heat transfer analysis, to determine the life and

cooling penalties and it would have to be done several times, modifying the design each time. To determine material temperature distribution, which has strong influence on life of the part, three major aspects of the problem have to be considered:

- the heat load to the surface from the fluid stream, which will be by convection through a viscous boundary layer
- the heat conduction and absorption within the part
- the convective cooling of the part by the coolant within interior cooling ducts

In the following sections, the requirements for calculating heat loads and performing conjugate heat transfer analysis on an airfoil are discussed. They will be discussed in a way that somebody new to the field might see what information, analysis, and methods are available and more importantly what analysis or additional aspects would be needed in order to execute a successful design or to perform the conjugate heat transfer analysis of a complex or simple geometry part.

1.4 Assumptions and Overall Structure

For an airfoil shaped object with an inner cooling duct immersed in a hot fluid stream, the heat transfer mechanism involves the following two phenomena that need to be considered:

- Convection from the fluid stream to the surface of the airfoil
- Conduction inside the airfoil body

Velocity and temperature distribution within a thin layer near the airfoil surface are governed by the boundary layer equations. Temperature distribution within the body of the airfoil is described by two-dimensional steady-state heat conduction equation.

Before we get into the analysis, certain assumptions have to be made:

- the flow is incompressible
- the thermophysical properties of the fluid are constant (k, μ, etc.)
- the flow is two-dimensional, laminar and steady
- the body forces are negligible and there is no energy generation inside the flow
- the boundary layer approximations are applied
- the thermal conductivity of the body does not change with the temperature
- the body material is homogeneous and isotropic
- the heat generation or cooling is uniform throughout the body
- the analysis will be performed for zero angle of attack $\alpha=0$

The fluid velocity at the surface of the airfoil is zero due to no slip condition. Thus, in the vicinity of the surface, the heat transfer within the fluid can be described by *Fourier's law* of heat conduction. The boundary condition for the conjugate heat transfer on the body surface is that the heat flux and the temperature are continuous.

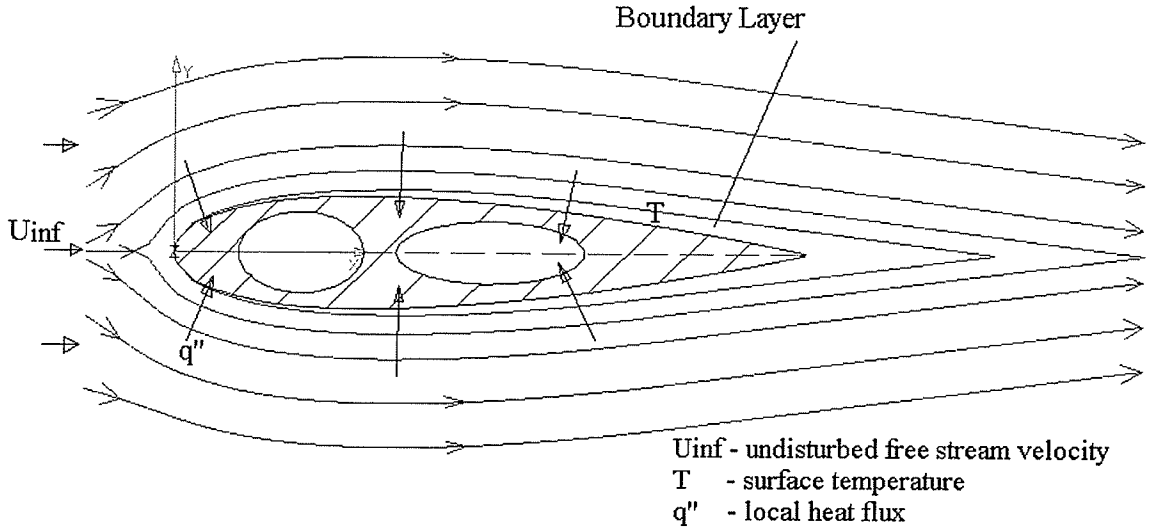

Figure 1-1 An airfoil with cooling ducts immersed in a fluid flow

The procedure starts with resolving the boundary layer equations. After a solution for velocities u (the streamvise direction) and v (the direction normal to the surface) is obtained, temperature distribution within boundary layer can be calculated using the assumed temperature as a boundary condition on the body surface.

From the temperature field, heat flux is determined and used as a boundary condition on the surface for conduction analysis within a body, resulting in improved surface temperature. The resulting surface temperature becomes the boundary condition for a new thermal boundary layer solution. Iterations continue until the temperature on the body surface is converged.

2 The Inviscid Flow Solver

2.1 Introduction

This chapter introduces the surface vorticity method of potential flow analysis. The introduction is followed by a discussion of major coding concerns. At the end of the chapter, an evaluation of the panel method subroutine created in the study is performed.

2.2 The Surface Vorticity Method

The inviscid, two-dimensional potential flow analysis adopted here is the vortex element boundary integral method. It is relatively simple to code and is economical to implement. A detailed description of the method and more about its applicability can be found in the book used as a fundamental panel method reference, "Turbomachinery Performance Analysis" R.I. Lewis (1996).

2.2.1 The physical basis for surface vorticity modeling

The physical model underlying the surface vorticity method of potential flow analysis is based on the concept of existence of a boundary/shear layer attached to any solid surface. The fluid velocity is reduced from the value v_s at the outer edge of the boundary layer, to zero on the wall due to shear or vorticity within the boundary layer. If $Re \rightarrow \infty$ (remember that Re is the Reynolds number defined as the ratio of inertial and viscous

forces in a flow), the boundary layer transforms into an infinitely thin vorticity sheet of strength $\gamma(s)$ per unit length at point s, which represents the bounding surface of any body in an inviscid 'potential' flow. Since the velocity jumps from zero below the vorticity sheet to v_s just outside the vorticity sheet, vorticity sheet will travel along the surface with velocity $v_c=1/2\ v_s$ indicating that vorticity is supplied from upstream. The boundary condition at the body surface is that velocity v_{si} just inside the vorticity sheet and parallel to the surface is to be zero. This leads us to the conclusion that the local vortex sheet strength γ (s) is exactly equal to the surface velocity v_s in potential flow past a body. That statement can be expressed as:

$$\gamma\,(s) = v_s \tag{2.1}$$

The above physical model establishes a foundation to the surface vorticity modeling of potential flows often referred to as Martensen method (1959).

2.2.2 Martensen's boundary integral equation

We apply the above principles to the flow past a two-dimensional body that is immersed in a uniform stream W_∞ inclined to the x-axis at an angle α_∞, Fig.2-1. The vorticity sheet of strength $\gamma(s)$ is measured clockwise around the body perimeter from the leading edge in the case of an aerofoil. The velocity dq_{mn} at some surface location s_m induced by the vortex element $\gamma(s_n)\ ds_n$ at location s_n will be normal to the radial vector r_{mn} and will be of magnitude

$$dq_{mn} = [\ \gamma(s_n)\ ds_n\ ] / [2\pi\ r_{mn}] \tag{2.2}$$

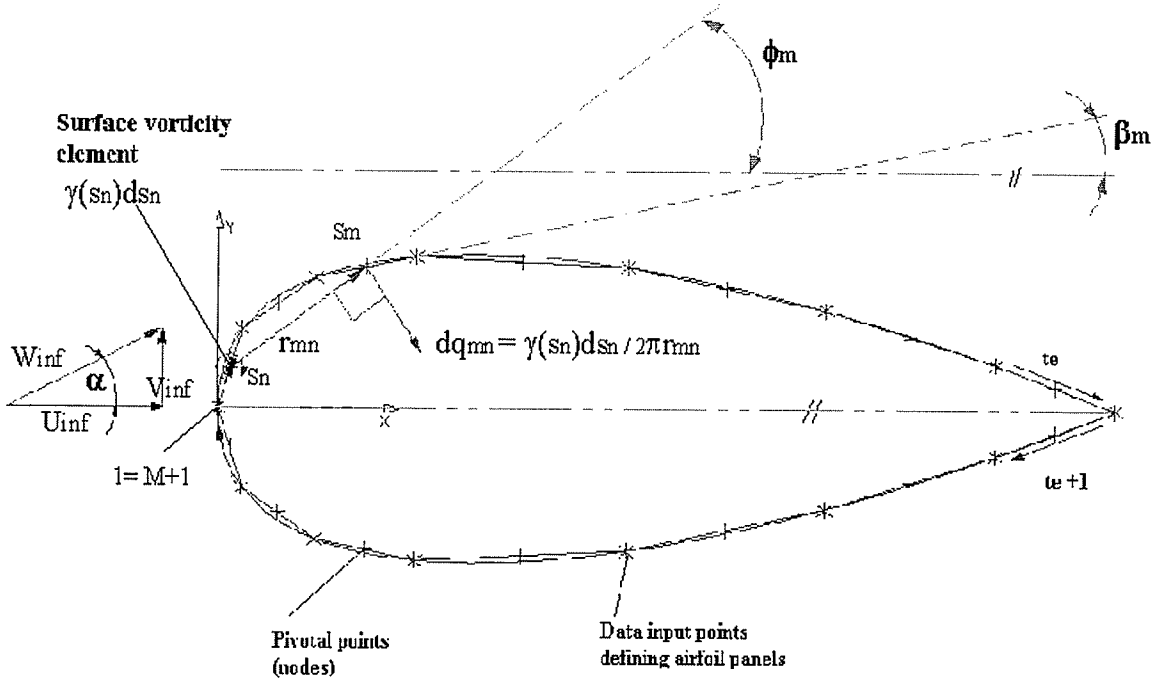

Figure 2-1 Surface vorticity model for flow past an airfoil

The surface boundary condition, from which the boundary integral equation will be derived, will be stated after dq_{mn} is resolved parallel to the body surface at s_m and points s_m and s_n are linked through the coupling coefficient $k(s_m, s_n)$. The self-convection velocity v_{cm}, parallel to s_m, will be included into the boundary integral equation after:

- dq_{mn} is resolved parallel to the body surface at s_m
- points s_m and s_n are linked through the coupling coefficient $k(s_m, s_n)$, and
- the integration is performed to account for the entire sheet.

Since the integration is performed through the center of the sheet, $\gamma(s_n)$ /2 has to be subtracted to obtain the velocity just inside the sheet. After combining the above contributions to v_{si} with the components of the uniform stream resolved parallel to the surface at s_m, the boundary condition at s_m or the boundary integral equation will be expressed as

$$-\tfrac{1}{2}\gamma(s_m) + \oint k(s_m, s_n)\,\gamma(s_n)\,ds_n + W_\infty\,(\cos\alpha_\infty\,\cos\beta_m + \sin\alpha_\infty\,\sin\beta_m) = 0 \qquad (2.3)$$

2.2.3 Application of Martensen's boundary integral equation

The body surface will be divided into M straight line elements of length Δs_n with pivotal points (x_n, y_n) located at the center of each panel. There will be a finite vortex element of strength $\gamma(s_n)\,\Delta s_n$ on each element that transforms integral Eq. (2.3) into a linear equation

$$\sum_{n=1}^{M} K(s_m, s_n)\,\gamma(s_n) = -\,U_\infty \cos\beta_m - V_\infty \sin\beta_m \qquad (2.4)$$

where the coupling coefficients $K(s_m, s_n)$ are given as,

$$K(s_m, s_n) = k(s_m, s_n)\,\Delta s_n =$$

$$= (\Delta s_n / 2\pi)\,\{[(y_m - y_n)\cos\beta_m - (x_m - x_n)\sin\beta_m]/[(x_m - x_n)^2 + (y_m - y_n)^2]\} \qquad (2.5)$$

Equation (2.4) will be written for each pivotal point (x_m, y_m) resulting in a set of M linear equations for the M initially unknown surface vorticity values $\gamma(s_1), \gamma(s_2), \gamma(s_3), \ldots, \gamma(s_M)$. The matrix form of Eq. (2.4) is

$$\mathbf{[\,K_{m,n}\,]\,\{\gamma(s)\} = \{W(s)\}} \qquad (2.6)$$

where $[\,K_{m,n}\,]$ is the matrix of the coupling coefficients, $\{\gamma(s)\}$ is the vector of unknown surface vorticity values, and $\{W(s)\}$ is the vector of values defined through the components of the uniform stream resolved parallel to the surface at s_m.

In order to obtain a solution of the above equation when applied to lifting airfoils, trailing edge Kutta-Joukowski condition needs to be imposed and coupling coefficients on the back diagonal have to be corrected. The procedure for back-diagonal correction is the one

advocated by Jacob and Riegeles (1963) that enforces zero internal circulation expressed as

$$\sum_{n=1}^{M} K_{mn}\, \Delta s_m = 0 \tag{2.7}$$

that will make the matrix singular and insoluble.

The matrix will be non-singular if we impose the bound circulation Γ around the airfoil given by the total net vorticity on the body surface

$$\Gamma = \int \nu_s \, ds = \int \gamma(s)\, ds = \sum_{n=_1}^{M} \gamma(s_n)\, \Delta s_n \tag{2.8}$$

Eq. (2.8) will be added to every equation of the matrix. So, the m-th equation will be

$$\sum_{n=1}^{M} [K(s_m, s_n) + \Delta s_n\,]\, \gamma(s_n) = -\, U_\infty \cos \beta_m - \, V_\infty \sin \beta_m \, + \, \Gamma \tag{2.9}$$

The bound vorticity Γ for airfoil is strongly controlled by the flow in the trailing edge region. Fluid flows from left to right on both upper and lower surfaces. The local surface velocity and associated vorticity, $\nu s = \gamma(sn)$ are defined as positive if aligned in clockwise direction. Therefore, the Kutta-Joukowski trailing edge condition is applied through forcing the equal and opposite vorticity strengths on the two panels that form the trailing edge,

$$\gamma(s_{te}) = -\, \gamma(s_{te+1}) \tag{2.10}$$

Then we replace the bound vorticity Γ in Eq. (2.9) by a unit bound vortex,

$$\sum_{n=1}^{M} [K(s_m, s_n) + \Delta s_n]\, \gamma(s_n) = - (U_\infty \cos \beta_m + V_\infty \sin \beta_m) + 1 \qquad (2.11)$$

In the code provided, the above equation is broken into separate equations for W_∞ and Γ,

$$\sum_{n=1}^{M} [K(s_m, s_n) + \Delta s_n]\, \gamma_1(s_n) = - (U_\infty \cos \beta_m + V_\infty \sin \beta_m) + 1 \qquad (2.12)$$

$$\sum_{n=1}^{M} [K(s_m, s_n) + \Delta s_n]\, \gamma_2(s_n) = 1 \qquad (2.13)$$

After separate solution for $\gamma_1(s)$ and $\gamma_2(s)$ are obtained, the final solution for any bound vortex is calculated by using the following expression

$$\nu_{sn} = \gamma(s_n) = \gamma_1(s_n) + \Gamma\, \gamma_2(s_n) \qquad (2.14)$$

The aerofoil bound vorticity is obtained after substitution of the above relation into the Eq. (2.10)

$$\Gamma = - \frac{\gamma_1(s_{te}) + \gamma_1(s_{te+1})}{\gamma_2(s_{te}) + \gamma_2(s_{te+1})} \qquad (2.15)$$

In addition to everything said above, it has to be mentioned that the 'self-inducing' coupling coefficient $K(s_m, s_m)$, which represents the velocity induced parallel to the surface at s_m by element Δs_m itself, as shown by Lewis (1991, page 23) may be expressed with good approximation by

$$K(s_m, s_m) = - \tfrac{1}{2} - [(\beta_{m+1} - \beta_{m-1})/ 8\, \pi]$$

2.3 Coding Issues

As stated in the previous section, the simulation of a flow past an airfoil involves two major issues namely: back diagonal correction and making the coupling coefficient matrix non-singular by imposing the bound circulation Γ.

When it comes to coding, implementation of the recommended Jacob and Riegels procedure (from Ref. 3) translates into that the back diagonal coefficients have to be calculated separately from the rest of the coefficients in the matrix through a different expression. After zero internal circulation C=0 is enforced for each column as suggested by Eq. (2.7), the back diagonal coefficient in a column *n* will be calculated as function of the coefficients in that column that have been already calculated. It is not very hard to implement and does not represent any significant computing effort.

Application of the bound circulation Γ in order to make the matrix non-singular is a problem resolved by adding the Eq. (2.8) to every equation of the matrix followed by imposing the Kutta-Joukowski trailing edge condition. As explained, separate solutions for W_∞ and a unit bound vortex Γ=1 are obtained. Once the separate solutions are computed the airfoil bound vorticity and the final solution are determined. Obtaining the separate solutions does not represent a new computing requirement since the same coupling coefficient matrix is used.

The procedure employed in resolving the system of equations is the matrix inversion method. Through the numerous executions of the main code, it has been noticed that the resolving the system of equations is taking most of the execution time. If the number of the panels around the body is increased by attaching a wake body, the computing time increases more than several times the total number the panels. Among all of the subroutines of the main code, the panel subroutine has been shown to be the slowest. It is much slower than the subroutine employed to resolve heat conduction within a body. The conduction procedure where a similar system of equations of the same order has to be resolved will be discussed in the Chapter 4.

2.4 Evaluation of The Code

Since it represents one of the crucial segments of the main code, a great effort was devoted to the evaluation of the solution generated by the vortex element subroutine. The results of each separate procedure within the subroutine, such as back diagonal correction or matrix inversion, had been verified before the subroutine was tested as a separate program.

2.4.1 Uniform stream past a circular cylinder application

The subroutine was first employed in obtaining the solution for a uniform stream U_∞ past a circular cylinder. The solution obtained by the surface vorticity method with 22 elements on the cylinder surface agreed with the exact solution to within less than 0.1%. The results will not be discussed in detail since it is rather trivial case but the solution is shown and compared with the exact solution on the chart below.

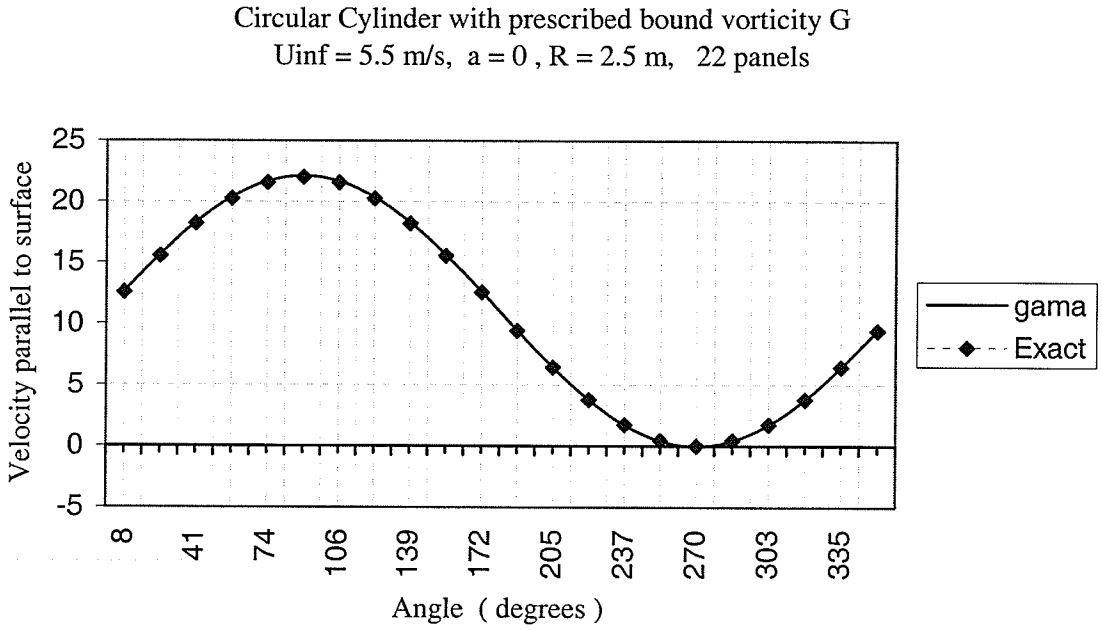

Figure 2-2 Velocity distribution around a circular cylinder

2.4.2 Uniform stream past an airfoil application

The method and the subroutine went through a real test when they were employed in resolving the flow around a real airfoil with plenty of experimental data on airfoil performance characteristics. One of these well known airfoils NACA 0012 provided a shape to the body analyzed in the main code. Usually, the potential flow around an airfoil is characterized by pressure coefficient, however this time it will not be the case, due to the available data for the velocity distribution around the airfoil, "Theory of Wing Sections, Including a Summary of Airfoil Data", Abbot, I.H./ Doenhoff, A.E. The velocity distribution for NACA 0012 is given as the ratio of the velocity on the surface to the upstream velocity U_∞. In the analysis, a total of 46 elements around the airfoil were used. On the chart below, an excellent agreement with the experimental data is shown. At some locations along the chord the solution matched the experimental data to less than 0.01 %.

The most critical zones, when it comes to accuracy of the solution, are the leading and the trailing edge of the airfoil. The panels at those two regions have to be much smaller than panela somewhere half way along the chord that are usually 10% of the length of the chord. In this specific case, y-coordinate of the airfoil was not specified for 0.5% of the chord, which forced the calculation of the coordinate by assuming the airfoil leading edge follows the second order parabola. After that additional panel was provided, the compliance with the experimental results at the leading edge was much better. The region at the trailing edge or more precisely the last 5% of the chord was divided into 5 panels in order to move the spike in the pressure coefficient distribution chart closer to the end of the airfoil. Accuracy of the solution would certainly improve if the number of panels around the body approximating the surface increased. Due to already more than satisfactory solution obtained with 46 panels and the effect additional panels would have on the program execution time, such effort was not attempted.

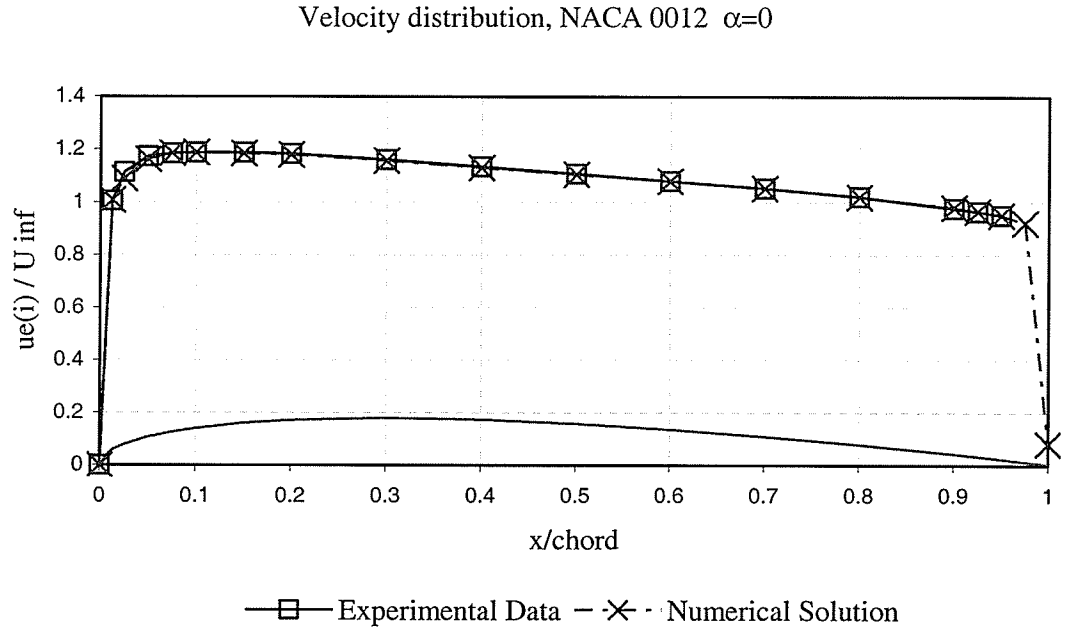

Figure 2-3 Velocity distribution, NACA 0012

A more challenging problem was obtaining the same pressure and friction drag coefficients as provided by the literature. The part of the problem was that the flow analyzed in this study is laminar. Even for a well-known and well-studied NACA 0012, airfoil characteristics are not readily available for low *Re* flows. The drag coefficient at low *Re* was available for NACA 0009 but the airfoil coordinates provided were not such that could facilitate application of uniform grid for the boundary layer solver. Additionally, there were no experimental data for velocity distribution over NACA 0009 available. Those issues led to analysis of NACA0012. The drag coefficients for NACA 0009 were used to evaluate the results for NACA0012, since the performance of airfoils is related to their geometry. Both airfoils are uncambered and differ in their thickness distribution only by scale factors. The airfoils have about the same Cd_{min} that is different by few percents only, implying that NACA0012 should have higher Cd_{min} than NACA 0009 for about 1-3%, as can be observed on Fig. 2-4 from Ref.7. The solutions for pressure and friction drag coefficients were obtained well within those boundaries, which verified the application of the vortex element method as a solver for a potential flow.

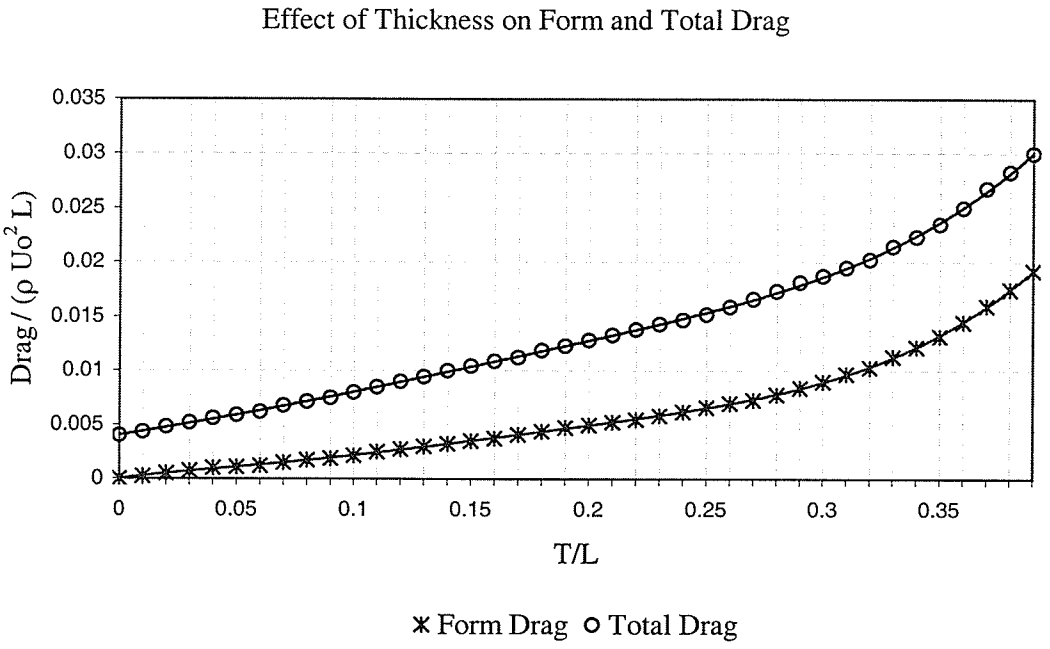

Figure 2.4 Effect of thickness on form and total drag

A separate procedure, attaching a wake body to an airfoil at the point of separation, was used in the main program. After a wake body is attached to original airfoil geometry, inviscid flow analysis is performed on the new body called "pseudobody". The velocity distribution around the pseudobody after several iterations, including the boundary layer calculations, is such that the fluid stream will not separate from the surface. Aerodynamic characteristics are then obtained based on velocity distribution for the pseudobody. By doing this, viscous effects are taken into consideration and fairly accurate values for aerodynamic characteristics are computed. Even though it takes a fair amount of time and skill to implement the procedure in the code, the procedure and the methodology will not be discussed in great detail within this study. The routine does not have influence on the major subject of the study, the conjugate heat transfer. It was used to calculate airfoil characteristics and evaluate surface vorticity method.

The panel method subroutine produces a solution for different angles of attack. Due to the coupling with other procedures, such as an attachment of the pseudobody and the boundary layer calculations, that feature stays unemployed.

3 The Boundary Layer Solver - Finite Difference

3.1 Introduction

Chapter 3 gives a short introduction to the significance of boundary layers and the finite difference method. The Crank-Nicolson method from Ref. 2 and Ref. 8, which is an implicit finite difference method, and its application in the code are presented. Major coding issues encountered in the study, code evaluation, and a few examples are discussed.

3.2 Significance of the Boundary Layers

3.2.1 The Velocity Boundary Layer

The concept of the boundary layer is associated with the presence of the shear stress τ acting in planes parallel to the fluid velocity (see Fig. 3-1). As it is known, the shear stress τ is a consequence of the fluid viscosity. Due to viscosity, the surface of an object will retard the motion of particles in the adjoining fluid layer. Those particles will be slower than the particles in the next layer and they will try to slow down those in the next layer. The phenomenon is present through the fluid due to viscosity but at some distance away from the surface, it becomes negligible compared to its magnitude in the vicinity of the surface. That region where velocity gradients and shear stresses are large is called the boundary layer. The velocity component of the fluid in the streamvise direction denoted

by u, that is equal to 0 at the surface, increases across the boundary layer until it reaches the value of the free stream u_∞. This is how we get one of the main characteristics of each boundary layer, its thickness δ defined as the value of y for which $u=0.99U_\infty$. That value of u is denoted u_e. The above concept led to the idea that the fluid flow is characterized by two distinct regions: a thin fluid layer in which velocity gradients and shear stresses are large, and a region outside the boundary layer in which velocity gradients and shear stresses are negligible. Analysis of the fluid flow in this study is divided accordingly. Since there is variation of x component velocity along y across the boundary layer, its more specific name is the velocity boundary layer (see Ref. 2,4, and 5). The importance of the velocity boundary layer to problems involving convection transport will be seen more clearly in the section describing application of the finite difference method to equations modeling energy transfer within boundary layer.

If we know the velocity gradient at the surface, we can obtain the surface shear stress in case of a Newtonian fluid (from Ref. 2)

$$\tau_s = \mu \, (\delta u/\delta y)|_{y=0} \tag{3.1}$$

where μ is the dynamic viscosity and y direction perpendicular to the surface.

Since surface frictional effects are closely related to the surface shear stress, the local friction coefficient (see Ref.2), from which the surface frictional drag may be determined, is found by employing the following formulation

$$C_f = \tau_s / (\tfrac{1}{2} \rho \, u_e^2) \tag{3.2}$$

3.2.2 The Thermal Boundary Layer

If the temperatures of the fluid free stream and surface temperature are different, thermal boundary layer must develop (see Ref. 4).

In case of a flow over an isothermal flat plate, there will be a uniform temperature profile at the leading edge $T_{x=0} = T_\infty$. At other locations along the plate, temperature gradients in

direction y normal to the surface, will develop in the fluid due to energy exchange among the particles of the fluid. Energy exchange starts at the wall, where particles that come into contact with the plate reach thermal equilibrium with the surface temperature, and propagates through the fluid. That region in the fluid, where temperature gradients are present, is called the thermal boundary layer and its thickness is defined as the value of y where $[(T_s-T)/(T_s - T_\infty)] = 0.99$. T, T_s, and T_∞ are the temperature within boundary layer, the temperature on the surface, and the temperature of the free stream respectively. Since no fluid motion occurs very close to the surface, energy transfer takes place only by conduction. Therefore, at any distance x from the leading edge, the local heat flux may be obtained by applying *Fourier's law* to the fluid at y=0

$$HF_x = - k_f \, \delta T/\delta y|_{y=0} \tag{3.3}$$

where k_f is the thermal conductivity of the fluid and $\delta T/\delta y|_{y=0}$ is the temperature gradient on the surface in direction y normal to the surface.

By knowing that the local heat flux expressed by the *Newton's law* of cooling,

$$HF_x = h\,(T_s - T_\infty) \tag{3.4}$$

is equal to that defined trough Eq. (3.3), we can obtain the convection heat transfer coefficient h, in terms of conditions in the thermal boundary layer

$$h = (- k_f \, \delta T/\delta y \,|_{y=0}) / (T_s - T_\infty) \tag{3.5}$$

The above relation could be read as, the rate of heat transfer across the boundary layer is defined by the wall temperature gradient $\delta T/\delta y \,|_{y=0}$, determined by conditions in the thermal boundary layer.

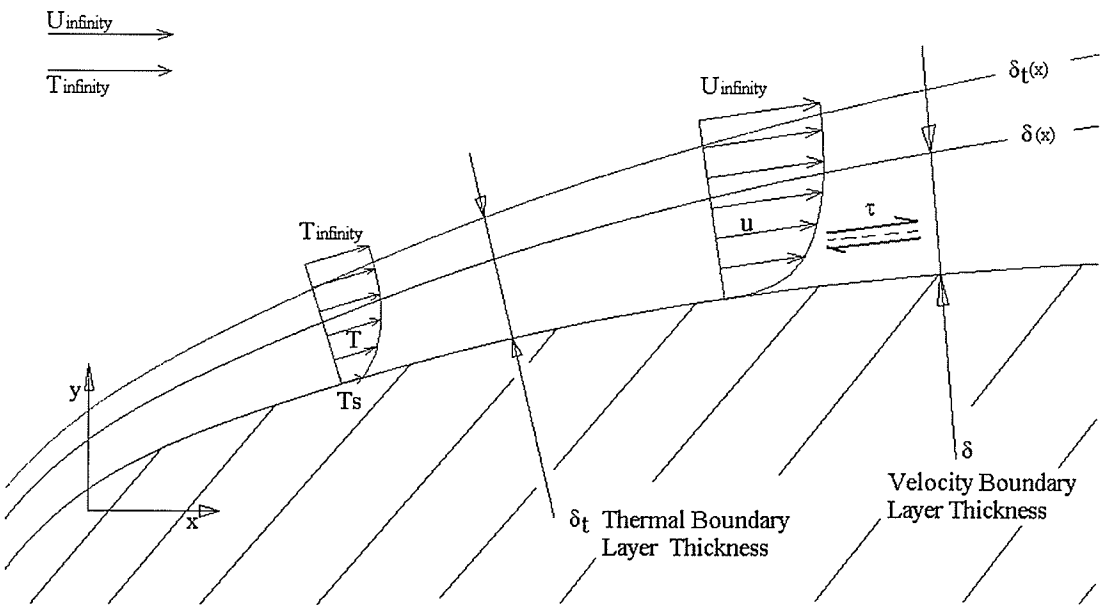

Figure 3-1 Velocity and thermal boundary layers on an airfoil

3.3 The Finite Difference Method

Solving the boundary layer equations requires the solution of differential equations, which means that boundary layer problems are not arithmetic problems. The key to development and wide presence of numerical methods are powerful computers that can do arithmetic calculations very rapidly on a very large scale. These facts led to the main idea of finite difference, to reduce the solution of the differential equations to an essentially arithmetic problem. This is done by creating a series of algebraic equations between the values of the dependent variables at various specific values of the independent variables that are separated by small finite differences (e.g. dx and dy in x and y directions respectively), instead of solving the differential equations. The process will be explained through the brief description of the finite difference implicit method applied in the code in resolving boundary layer equations. The region of interest or domain is divided by a small but finite spacing (dx and dy) into a grid. Even though the region should be curved due to curvature of the airfoil, the mesh will be two-dimensional rectangular mesh over a flat plate. The effects of the curvature will be taken into

consideration through boundary conditions specified at the edge of the boundary layer that are the results of the inviscid flow analysis. The points of interest are the intersections of grid lines, called node points, where the values of the dependent variables are calculated. The nodes or locations where the values are calculated are identified by using indexes n,m where n identifies a location in the x-direction and m in y-direction, $u(x,y) = u_{n,m}$. This notation will be used here, in the description of the method, rather than (i,j) indices used in the code.

3.3.1 The Velocity Boundary Layer Solver

The finite difference implicit methods are based on using unknown information at the downstream station that we are stepping toward, rather than known information at the station where we are.

The differential equations to be treated are the planar, incompressible forms of the continuity equation,

$$\delta u/\delta x + \delta v/\delta y = 0 \tag{3.6}$$

and the momentum equation without the unsteady term,

$$u\,\delta u/\delta x + v\,\delta u/\delta y = (-1/\rho)\,\delta p/\delta x + \nu\,(\delta^2 u/\delta y^2) \tag{3.7}$$

The parabolic momentum equation is solved with an implicit "downstream-marching" procedure, and the continuity equation is viewed as an auxiliary equation for determining v(x,y). First term in the momentum equation will be approximated by using forward difference approximation

$$\delta u/\delta x\big|_{n,m} \approx (u_{n+1,m} - u_{n,m}) / \Delta x \tag{3.8}$$

For the other derivatives in the momentum equation, Crank-Nicolson method of approximation will be applied:

$$\delta^2u/\delta y^2 \approx \tfrac{1}{2}\{[(u_{n+1,m+1}-2u_{n+1,m}+u_{n+1,m-1})/\Delta y^2]+[(u_{n,m+1}-2u_{n,m}+u_{n,m-1})/\Delta y^2]\} \quad (3.9)$$

$$\delta u/\delta y \approx \tfrac{1}{2}\{[(u_{n+1,m+1} - u_{n+1,m-1})/2\Delta y]+[(u_{n,m+1} - u_{n,m-1})/2\Delta y]\} \quad (3.10)$$

The Crank-Nicolson method (see Ref. 2. and 8) consists of taking the average of the approximations for $\delta^2u/\delta y^2$ evaluated at n and n+1 giving $\delta^2u/\delta y^2$ evaluated at n + ½, halfway across the strip over which one is stepping. The method has a relatively simple formulation and the order of approximation of $O(\Delta y)^2$, which means improved accuracy and smaller truncation errors. It is unconditionally stable but involves more computations per step than the simple implicit method.

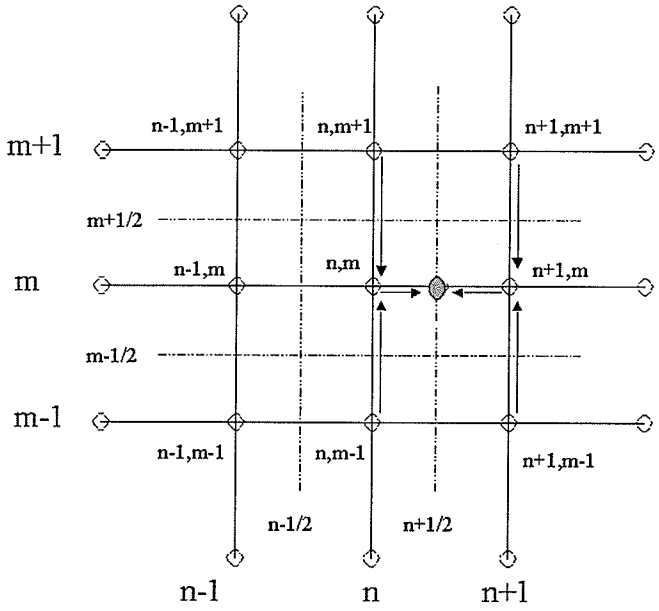

Figure 3-2 Finite difference grid, x-momentum equation terms approximation

The pressure gradient is specified from the inviscid solution through:

$$\delta p/\delta x = - \rho\, u\, \delta u/\delta x \quad (3.11)$$

$$-(1/\rho)\, \delta p/\delta x = u\, \delta u/\delta x \quad (3.12)$$

$$-(1/\rho)\, (p_{n+1} - p_n)/\Delta x \approx ue_n\, [(ue_{n+1} - ue_n)/\Delta x] \quad (3.13)$$

The terms aa and bb that include viscosity of the fluid and the step size in both directions are defined as

$$aa = 1/(4\Delta y) \qquad bb = \nu/(2\Delta y^2) \tag{3.14}$$

After the above equations are substituted into the momentum equation and the equation is rearranged, the implicit equation for calculating the streamwise velocity $u_{n+1,m}$ after a downstream step in terms of known information at the station before the step, is

$$A \cdot u_{n+1,m-1} + B \cdot u_{n+1,m} + C \cdot u_{n+1,m+1} = D \tag{3.15}$$

where coefficients A,B,C, and D are given as

$A = -[v_{n,m} \cdot aa + bb]$

$B = [(u_{n,m}/\Delta x) + 2 \cdot bb]$

$C = [v_{n,m} \cdot aa - bb]$

$D = (ue_n/\Delta x) \cdot [ue_{n+1} - ue_n] + u_{n,m}^2 - v_{n,m} \cdot aa \cdot [u_{n,m+1} - u_{n,m-1}] + bb \cdot [u_{n,m+1} - 2u_{n,m} + u_{n,m-1}]$

From the Eq.(3.15), we can calculate new velocity at one grid point downstream (n+1,m) in terms of known values from the previous step and unknown values from the current step at points (n+1,m-1) and (n+1,m+1). After the equation (3.15) is formed for each unknown, the system will be closed. Since the equation for each unknown involves only two other unknowns, the matrix is tridiagonal. As a convenient method of direct solution for the tridiagonal system, Thomas algorithm (Ref. 8) was employed within the code. Since we have a relation for one unknown in terms of some of the other unknowns, Eq. (3.15) is an implicit finite difference formulation. The boundaries of the computation region are the wall with no-slip condition at y=0 and the outer edge of the viscous region with velocity u_e at a distance far from the surface. The velocity u_e depends on free-stream velocity U_∞ and the shape of the body. The boundary and the initial conditions enter into the calculations easily. As can be noticed in the code, the no-slip condition is used directly in the calculation of all points that are one grid spacing up from the wall or

at m=2. On the other side, the outer boundary condition is used directly to calculate the values at all points that are one grid spacing down from the top boundary, m=M-1. The initial conditions are defined at n=1 and were directly used to obtain values at n=2.

The important fact to discuss here is that the differential momentum equation is nonlinear, however, the finite difference form of the momentum equation is linear. That was accomplished by evaluating the u in the coefficient of $\delta u/\delta x$ at n, where it is known and not at n+1 location where it is not known. The same was applied to the v multiplied by $\delta u/\delta y$ term.

Another obstacle in modeling the boundary layer equations besides having linear approximation of nonlinear momentum equation is the coupling between incompressible finite difference forms of the continuity and the momentum equations.

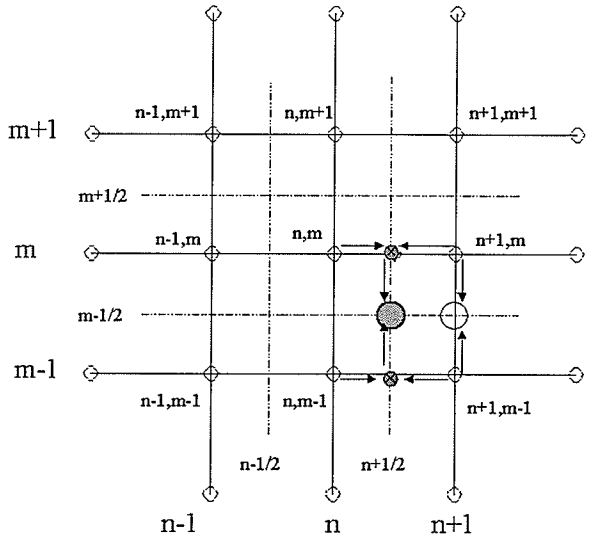

Figure 3-3 Finite difference grid, the continuity equation terms approximation

The second term in the continuity equation $\delta v/\delta y$ is approximated by backward difference discretization at the downstream station n+1,

$$\delta v/\delta y \mid_{n+1,m-\frac{1}{2}} \approx (v_{n+1,m} - v_{n+1,m-1})/\Delta y \qquad (3.16)$$

Approximation could be regarded as a central difference at a location that is halfway between two grid points (n+1,m) and (n+1,m-1) which is the point (n+1,m- ½), shown as a circle in the Fig. 3-3.

Term $\delta u/\delta x$ of the continuity equation will be approximated at point (n+ ½,m- ½) as the average of the $\delta u/\delta x$ central difference approximations at locations n+ ½ on levels m and m-1, as

$$\delta u/\delta x \mid_{n+½,\, m-½} = ½ \left[\delta u/\delta x \mid_{n+½,\, m} + \delta u/\delta x \mid_{n+½,\, m-1}\right]$$

$$\approx ½ \left[(u_{n+1,m} - u_{n,m})/\Delta x + (u_{n+1,m-1} - u_{n,m-1})/\Delta x\right] \qquad (3.17)$$

After substitution of the above approximations into the continuity equation and regrouping, velocity $v_{n+1,m}$ can be determined starting at m=2, using the boundary condition for v at the wall m=1, and proceeding up to m=M through

$$v_{n+1,m} = v_{n+1,m-1} - ½ (\Delta y/\Delta x) \left[(u_{n+1,m} - u_{n,m}) + (u_{n+1,m-1} - u_{n,m-1})\right] \qquad (3.18)$$

3.3.2 The Boundary layer Characteristics

Once an accurate solution for the velocity components is obtained, various other quantities of interest can be calculated. Certainly, the most important is the shear stress τ_s given by

$$\tau_s = \mu(\delta u/\delta y)_{y=0} \approx [\mu/(2\Delta y)] (4u_{n,2} - u_{n,3}) \qquad (3.19)$$

Eq. (3.19) comes from forward difference approximation Eq.(3.20) with the order of approximation of $O(\Delta x)^2$ at $u_{n+1,m}$ where m=1 or at the wall $u_{n+1,m} = 0$,

$$\delta u/\delta y \mid_{n+1,m} = (1/2\Delta y) (-3u_{n+1,m} + 4u_{n+1,m+1} - u_{n+1,m+2}) \qquad (3.20)$$

The skin friction coefficient is calculated by substituting Eq. (3.19) into Eq. (3.2) to get

$$C_f \approx [\nu/(\Delta y\, u_e^2)]\, (4u_{n+1,m+1} - u_{n+1,m+2}) \qquad (3.21)$$

The boundary layer thickness δ, can be found by searching the velocity $u_{n,m}$ for the value of m, thus y, corresponding to $u_{n,m} = 0.99\, u_e$. The integral thicknesses, δ^* and θ were found through integration of velocity profiles as specified by the following two equations:

$$\theta = \int [1 - (u/u_e)](u/u_e)\, dy \qquad (3.22)$$

$$\delta^* = \int [1 - (u/u_e)]\, dy \qquad (3.23)$$

The foregoing method is simple and quite successful. Even with the step-size restrictions described, very accurate calculations can be made in a very short time for planar, incompressible, constant-property flows.

3.3.3 The Thermal Boundary Layer Solver

The desired form of the energy equation (as shown in Ref. 2) is the exact boundary layer equation written in terms of the static temperature

$$\rho\, Cp\, [\, \delta T/\delta t + u\, \delta T/\delta x + v\, \delta T/\delta y\,] = u\, \delta p/\delta x + \delta p/\delta t - \delta q_y/\delta y + \tau\, \delta u/\delta y \qquad (3.24)$$

The above equation is valid for both laminar and turbulent flow, because the shear stress τ and the heat transfer rate q_y, have not been modeled.

For laminar flow of many common fluids, a suitable approximation is *Fourier's law*

$$q_y = -\, k\, \delta T/\delta y \qquad (3.25)$$

where $k = k(T)$ is a physical property of the fluid composition and temperature. Note that q_y does not depend on the motion of the fluid. Using this expression and the Newtonian expression for shear in laminar flow, we obtain the energy equation,

$$\rho Cp[\ \delta T/\delta t + u\delta T/\delta x + v\delta T/\delta y\ ] = \delta(k\delta T/\delta y)/\delta y + \mu(\delta u/\delta y)^2 + u\delta p/\delta x + \delta p/\delta t \qquad (3.26)$$

For a steady, constant property flow and incompressible fluid we can write

$$\rho Cp\ [\ u\ \delta T/\delta x + v\ \delta T/\delta y\ ] = k\ \delta^2 T/\delta y^2 + \mu(\delta u/\delta y)^2 \qquad (3.27)$$

The finite difference form of the energy equation will be developed the same way as the finite difference form of the differential momentum equation.

The first differential term in the equation $\delta T/\delta x$ will be approximated by using central difference around (n+½,m) with a grid step $\Delta x/2$ along x-axis, with the order of accuracy $O(\Delta x)^2$ as

$$\delta T/\delta x \approx (\ T_{n+1,m} - T_{n,m})/2(\Delta x/2) \qquad (3.28)$$

For the other derivatives in the energy equation, the derivatives will be approximated at both points (n+1,m) and (n,m) using the central difference. The approximation used in the equation will be the average of the two giving derivatives evaluated at (n+½,m), halfway across the strip over which one is stepping. As mentioned before the approximation is called Crank-Nicolson method and will be applied as follows

$$\delta T/\delta y \approx ½\{[(T_{n+1,m+1} - T_{n+1,m-1})/(2\Delta y)]+[(T_{n,m+1} - T_{n,m-1})/(2\Delta y)]\} \qquad (3.29)$$

$$\delta^2 T/\delta y^2 \approx ½\{[(T_{n+1,m+1} - 2T_{n+1,m} + T_{n+1,m-1})/\Delta y^2]+[(T_{n,m+1} - 2T_{n,m} + T_{n,m-1})/\Delta y^2]\} \qquad (3.30)$$

$$\delta u/\delta y \approx ½\{[(u_{n+1,m+1} - u_{n+1,m-1})/2\Delta y]+[(u_{n,m+1} - u_{n,m-1})/2\Delta y]\} \qquad (3.31)$$

After the equations (3.28), (3.29), (3.30), and (3.31) are substituted into the equation (3.27), regrouping, and if we use terms defined as

$A = (1/ 2\Delta x)\ [u_{n+1,m} + u_{n,m}]$

$B = (1/ 8\Delta y)\ [v_{n+1,m} + v_{n,m}]$

$C = (1/ 2\Delta y^2)$

there will be

$$\rho C_v A(T_{n+1,m}-T_{n,m}) + \rho C_v B(T_{n+1,m+1} - T_{n+1,m-1}) + \rho C_v B(T_{n,m+1}-T_{n,m-1}) =$$
$$k C T_{n+1,m+1} - 2k C T_{n+1,m} + k C T_{n+1,m-1} + k C T_{n,m+1} - 2k C T_{n,m} + k C T_{n,m-1} +$$
$$+ \mu[(1/ 4dy)(u_{n+1,m+1} - u_{n+1,m-1} + u_{n,m+1} - u_{n,m-1})]^2 \qquad (3.32)$$

In order for the above equation to be written in the form used in the code, the following constants will be defined

$NX1 = \rho C_v B$

$NX2 = \rho C_v A$

$NX3 = k C$

$NX4 = 1/ 4dy$

$NX5 = \mu\ [NX4(u_{n+1,m+1} - u_{n+1,m-1} + u_{n,m+1} - u_{n,m-1})]^2$

and the coefficients of the equation will be

$A1 = -(NX1+NX3)$

$B1 = NX2+2\ NX3$

$C1 = NX1 - NX3$

$D1 = (NX3 - NX1)\ T_{n,m+1} + (NX2 - 2\ NX3)\ T_{n,m} + (NX3 + NX1)\ T_{n,m-1} + NX5$

so that the final equation can be written in the form

$$A1 \cdot T_{n+1,m-1} + B1 \cdot T_{n+1,m} + C1 \cdot T_{n+1,m+1} = D1 \qquad (3.33)$$

After one sweep through the domain we will have the temperature distribution within the boundary layer.

3.4 Coding Issues

At the beginning of the study, a code for solving the boundary layer equations based on the explicit method of Wu (1961) was developed. The explicit, finite difference form of the momentum equation is not difficult to develop by analogy with the explicit treatment of the model equation. Even with its limitations, such as step-size restrictions, the method is quite successful. Very accurate calculations can be made in a very short time for a planar, incompressible, constant-property flow. Results of the numeric explicit procedure are not shown in this study since they are almost identical to the results shown in Section 3.5 obtained by the implicit procedure VelocityBL.for. However, it has to be mentioned that the results of the numeric explicit procedure were compared with the benchmark solutions for various problems such as Howard problem, and Falkner-Skan wedge flows given in Schetz (1993). The solution of the explicit procedure together with flat plate laminar boundary layer Blasius solution, Falknar-Skan solution for flows with pressure gradient and Howarth problem were used to verify the solution of the more complex implicit Crank-Nicolson finite difference procedure employed within the main conjugate heat transfer code.

Within the explicit formulation, there is only one grid point at n+1 location where the value of the unknown function is calculated based on the values from the previous step. The implicit formulation requires solving the system of equations with unknowns along the y-axis at n+1 location, which means that the values of the function will be calculated simultaneously at all grid points at n+1 location. This implies that the procedure for solving the boundary layer equations depends on the formulation used to approximate the differential equations. The explicit formulation is simple to code but its application is limited to a small step size due to stability of the solution. Even though the implicit formulation is more complicated to code, the usage of larger steps in both directions could make the implicit procedure more efficient than the explicit one. However, the steps Δx and Δy cannot be too large since the accuracy of the solution will be reduced. If the accuracy of the solution is a primary concern, the more efficient procedure to use might be the explicit formulation. Since the stability analysis cannot not be rigorously

performed for a non-linear system, results from the stability analysis for an idealized liner model equation system influenced the choice of technique used. The main reason why the implicit Crack-Nicolson method was used is that it is believed to be unconditionally stable for any ratio $Q= \nu \Delta x/(Ue\ \Delta y^2)$ and that the choice of step seizes will not cause significant obstacles in the study.

The coordinates, for the explicit and implicit procedures, were chosen such that the leading edge represents the origin of a coordinate system with coordinates (0,0) where the indices are n=m=1. First node at x = 0 is denoted by n=1. At the last station $x = x_{length}$, n will have a value of n = N. For the transverse coordinate, m=1 at y=0 and m=M at $y=y_{max}$. Our computation region is limited by N station in x-direction that is the trailing edge and by M station in y direction that is somewhere in the flow. The edge of the boundary layer will be defined by a number smaller than M.

In this study, attention was not devoted to optimization and computational efficiency. That is the reason why the step size in both directions Δx, Δy were constant through the marching steps and through the iterations. The improvements could be made in terms of grid size optimization like different steps in both directions through the marching steps pending on the region analyzed. Due to the fact that boundary layer thickness has to increase as the flow moves away from the leading edge, the steps Δx, Δy close to the leading edge would have to be very small and increasing down stream. That would imply that a number of grid points would have to be changing through iterations that would complicate the code even further. Since the finite difference subroutine was just one of the subroutines in the main code, there were more important issues that should be looked at. The same step size Δx and Δy were used for both velocity and thermal boundary layer solvers. The grid mesh, defined by the step size, was rectangular of the order of magnitude $\Delta x \approx 10\Delta y$.

The step size in the x direction could not be chosen to be any number. The body shape to be analyzed is an airfoil. In order to verify the inviscid flow calculations and the calculation of the pressure and friction drag and coefficients, an airfoil which already had

known characteristics, had to be analyzed. The coordinates x and y defining the airfoil geometry, for almost any airfoil, were specified at the same locations. The locations are not a constant distance apart from each other and usually are closer to each other at the leading edge at the order of 1.25% of the chord. As we are approaching the trailing edge, the distance would increase to 10% of the chord. Certainly those coordinates, used to define the length as well as the start and the end of each panel for the potential flow and conduction analysis, could not be used to define the grid for the finite difference calculations due to their size. However, since they define the airfoil and since the boundary layer analysis has to be coupled with the other two numerical methods, they had to be the nodes of the finite difference grid. There is an interpolation subroutine INTERPOL.for creating the finite difference grid points between x coordinates of the airfoil panels for finite difference calculations. The approach taken was that distances between x-coordinates of the points defining the airfoil will be divided by a suitable x direction step size required by finite difference, which is a common denominator for the lengths between panel's x coordinates. This is how the maximum step size in x direction that could be used was defined.

The same is not applicable to the value of Δy step size in y direction. Due to large gradients of velocity and temperature near the body surface Δy has to be kept quite small in that region. The choice had to be made between uniform and non-uniform grid. Implementing non-uniform grid in y direction would not be that easy because of the central differencing employed for several terms in y direction. Therefore, even though the uniform Δy across the whole boundary layer increases the number of simultaneous equations to be solved, the uniform grid was the choice in this study.

The choice of Δx and Δy values was not driven by the upper limits only as explained above. The other very important influence came from the quality of the resulting solution or in some cases the values for which the solution could be obtained. After a number of sets were tested, with different step sizes in both directions, for different kinematic viscosities and two airfoils 0010-35 and 0012, no definitive determination on the best grid to be used could be achieved. There were a few attempts to establish a relation between quality of the solution (or at least a trend) and certain quantities that define stability, for explicit and some implicit methods, but unfortunately without success. At

the end, the set that could produce satisfactory solution was adopted and used in further analysis.

The stability question is a bit more complex to explain and a detailed explanation will not be presented here. The stability of the solution (or the approximation) of the linear model parabolic equation could be analyzed mathematically, as shown within "Boundary Layer Analysis" by Joseph A. Schetz, and would be a good example to understand the issue.

Other causes besides the stability, that could lead to obtaining a good solution in certain cases and no solution in others, could be associated to round-off and truncation errors. Round-off errors come from the finite number of digits that must be handled in any computer. Clearly, each number must be rounded off at the last digit that can be retained. Truncation errors are consequence of disregarding most of the terms in series expansion when approximating derivatives. Errors could dominate the solution if they accumulate a little at each step that would lead to an unstable procedure, which probably happened for most of the input sets tried. Opposite to that is when the errors tend to cancel each other and the total error by the end of the problem remains small, and the procedure is called stable. Some of this effect is present within solutions for boundary layer characteristics, displacement and momentum thickness calculated for an airfoil and can be noticed on charts shown in Chapter 5, Figure 5-2. Even if the solution is not stable at the first few steps, the errors are kind of "washed" down the stream and solution becomes stable.

Boundary layer characteristics such as local skin friction coefficient, boundary layer thickness, momentum thickness, displacement thickness, and point of separation were calculated within subroutine "VelocityBL2.for". For some of the sets executed, solutions for boundary layer characteristics would not stabilize along x direction and the sets would be considered unable to produce a solution. Remember that the boundary layer characteristics are calculated from the formulas where the ratio between u (x-component velocity within the boundary layer) and u_e (velocity at the edge of the boundary layer) or u/u_e, is the dominant term. Solution for the skin friction coefficient was shown not to be that hard to obtain, for flat plate examples and in some cases for airfoils, through Eq.

(3.21) employed. In almost any execution of the code (some of them are shown in Table 3.3) or set of Δx, Δy, ν, and U_∞ tried, the solution followed the trend or behavior from the literature. Only at the second step in x-direction, the values were not aligned with the curve. Other than that, the skin friction curve was almost always smooth and the skin friction coefficient values were almost always steadily decreasing as marching towards the trailing edge.

The issue creating a major obstacle within the boundary layer solution, which has a huge influence on coupling between potential flow and boundary layer analysis, is the determination of the point of separation. The key is the criterion determining how close to a zero value the skin friction coefficient has to be in order for a location to be regarded as the point of separation. Several criteria were tried, basically different minimum values that could not produce reliable solution in terms of defining the point of separation within 10 percent of the chord through the pressure loop iterations. That problem led to the idea of changing the formula Eq. (3.21) for the skin friction calculation that comes from forward difference approximation with the order of approximation of O $(\Delta x)^2$ at $u_{n+1,m}$ where m=1 or at the wall $u_{n+1,m} = 0$ to the expression

$$Cf \approx [\nu/(\Delta y\, U_e^2)]\, (u_{n+1,3} - u_{n+1,1}) \tag{3.34}$$

The Eq. (3.34) is a linear approximation or the forward difference approximation of $\delta u/\delta y$ with the order of approximation of O (Δx). Surprisingly, the lower order of approximation produced better results and was used in further analysis. The skin friction coefficient curve became smoother along the chord, especially as it was approaching the point of separation and fluctuation. With the new expression used, the point of separation through iterations was kept to within 5% of the chord with the criterion defining the location as that $Cf<0.0001$.

Influence of the potential solution to the boundary layer solution and determination of point of separation will be discussed in great detail within Chapter 5.

3.5 Evaluation of The Velocity Boundary Layer Solver

Before the subroutine VelocityBL.for that resolves boundary layer equations and gives the solution for velocity distribution within the boundary layer was inserted into the main code and applied to an airfoil, its solution had to be verified. The evaluation was done through comparison with the following benchmark solutions for laminar flow. The results of VelocityBL.for were first compared with the exact solution for flow over a flat plate that was originally produced by Blasius. Looking at those results and comparing them with the Blasius solution in the below table, it was concluded the results are in good agreement.

ν (m^2/s)	$U\infty$ (m/s)	X (m)	Solution Source	δ	δ^*	θ	Cf
0.0002	10	1	Blasius Solution	0.0224	0.00771	0.00297	0.00297
			VelocityBL.for	0.023	0.00773	0.00297	0.00296

Table 3-1 Comparison VelocityBL.for vs. Blasius Solution

The grid used to obtain the values in the table was uniform in both directions with the same step size dx = dy = 0.001.

Next step in evaluation was the comparison of numerical solutions produced by VelocityBL.for and the implicit procedure, given by J.A. Schetz in Ref. 2, for resolving the so-called Howarth problem ($u_e = U_o - Ax$). The Howarth problem gives a flow that is nonsimilar, the adverse pressure gradient flow leading to separation, which would be a severe test. This case was chosen since similar adverse pressure gradient will exist on an airfoil. Characteristics of the boundary layer obtained by two solutions are shown on comparison charts. The analyzed flow was laminar with kinematic viscosity $\nu = 0.0002$ m^2/s, at $U_\infty = 10$ m/s over a surface that is a flat plate from the leading edge to x=1 m. At that station, a ramp begins to produce an inviscid velocity distribution $u_e(x)= 10.5-x/2$ m/s. The boundary layer development was calculated over the surface from x=1 to x=2 (m).

Two solutions have been shown to be in good agreement but they are not identical. The solution shown by the Schetz was obtained or calculated by starting from station x=1 m. The values from Blasius solution at x=1 m were used to obtain the "initial" conditions for calculations. Since initial profiles were also required, Schetz adopted the cubic velocity profile for u,

$$u/ue = 3/2\ (y/\delta) - \tfrac{1}{2}\ (y/\delta)^3$$

and v = 0 as adequate approximations to the exact Blasius solution, which would require tabular input for the profiles.

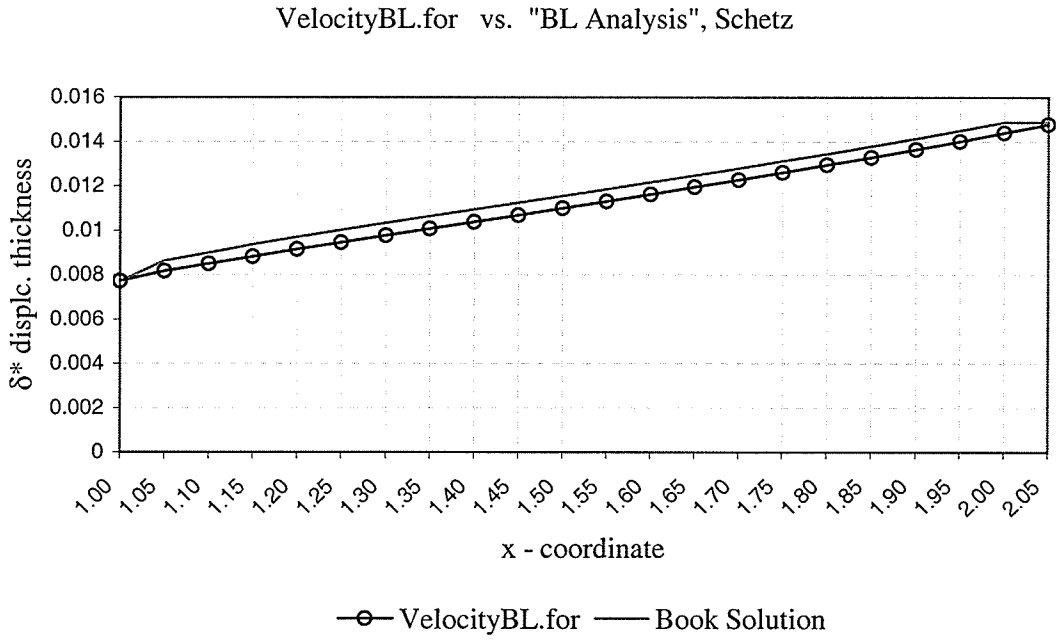

Figure 3-4 VelocityBL.for vs. "BL Analysis" by Schetz, displacement thickness δ*

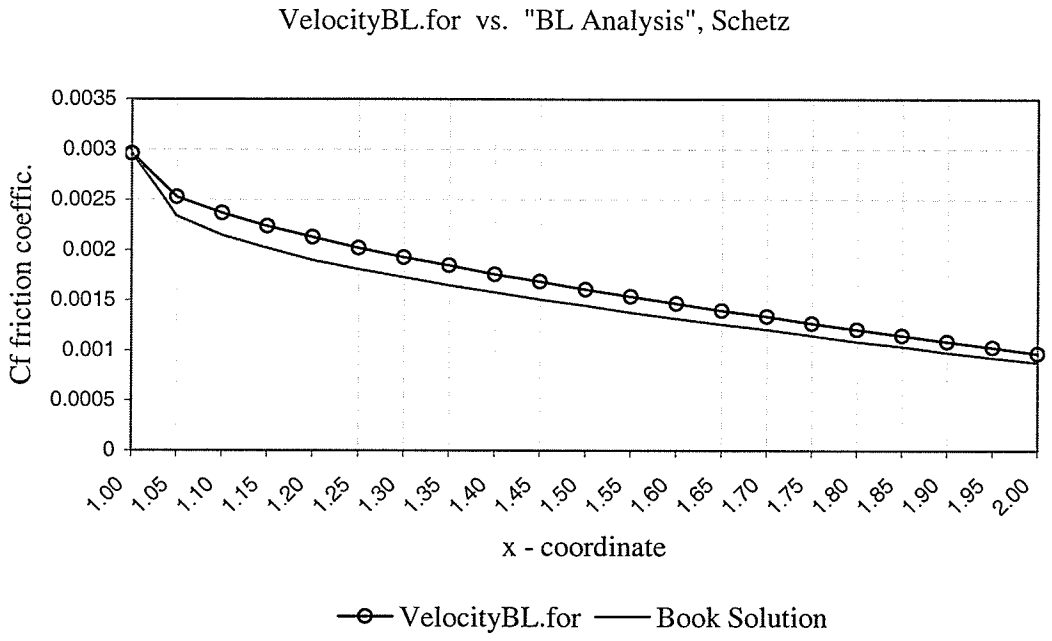

Figure 3-5 VelocityBL.for vs. "BL Analysis" by Schetz, Skin friction coefficient Cf

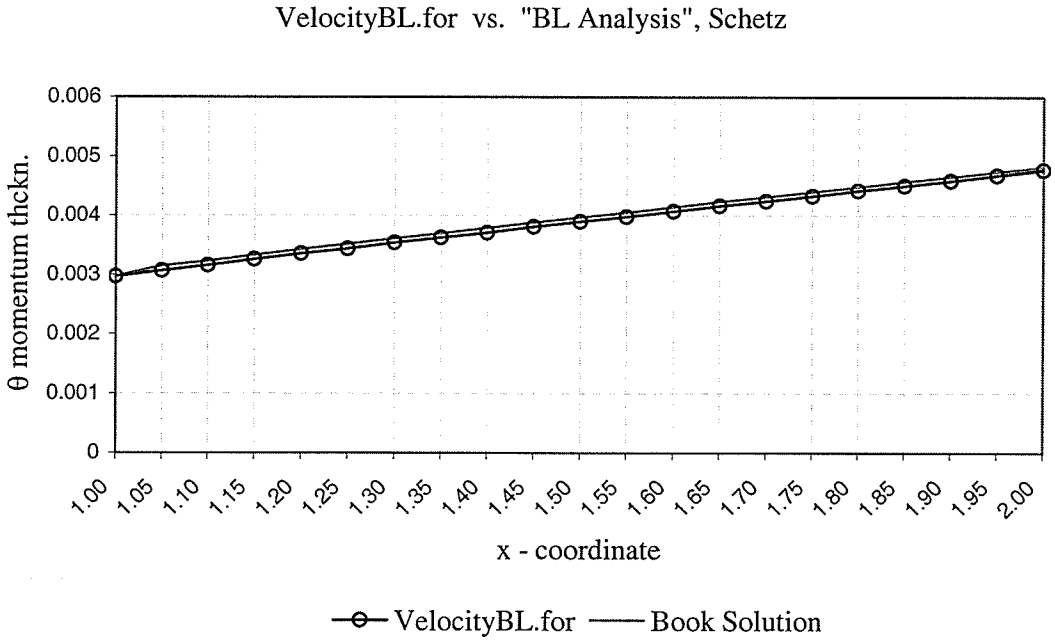

Figure 3-6 VelocityBL.for vs. "BL Analysis" by Schetz, momentum thickness θ

m	β	Solution Source	Cf $Re^{0.5}$	(δ*/x) $Re^{0.5}$	(θ*/x) $Re^{0.5}$	(δ/x) $Re^{0.5}$
1	1	"BL Analysis" Schetz	2.465	0.648	0.292	2.4
		VelocityBL.for	2.735	0.722	0.316	2.735
1/3	½	"BL Analysis" Schetz	1.515	0.985	0.429	3.4
		VelocityBL.for	1.571	1.021	0.438	3.729
0.111	0.2	"BL Analysis" Schetz	1.025	1.320	0.548	4.2
		VelocityBL.for	1.036	1.334	0.547	4.226
0	0	"BL Analysis" Schetz	0.664	1.721	0.664	5.0
		VelocityBL.for	0.661	1.730	0.665	5.221
-0.024	-0.05	"BL Analysis" Schetz	0.559	1.879	0.701	5.4
		VelocityBL.for	0.550	1.9	0.707	5.469

Table 3-2 Comparison VelocityBL.for vs. Hartree as shown by Schetz

On the other side, VelocityBL.for performs a calculation from the beginning of the plate at the station x = 0 m as it is a flat plate until station x=1. From the ramp at x=1 m the inviscid velocity distribution was calculated through the formula provided. Thus, VelocityBL.for used the results of its own calculation, u and v velocity distribution within the boundary layer from the previous step, as an input at station x=1. Other than the "initial" conditions for calculations, there are no other differences in approach to the problem. The codes used by Schetz and VelocityBL.for are based on the same implicit procedure. The grid used by VelocityBL.for to obtain the values shown on the Figure 3-4 to Figure 3-6 for the Howarth problem and the results for the Falkner-Skan problem in the Table 3.2, was uniform in both directions with the same step size dx = dy = 0.001.

The final evaluation of VelocityBL.for was done by comparing its results with the ones produced by similar solutions with the pressure gradient. If we consider the boundary layer development on an airfoil, we can notice four distinct regions: stagnation point region, region from stagnation point to about maximum thickness with favorable pressure gradient $dp/dx < 0$, zone around maximum thickness $dp/dx \approx 0$, and section with adverse pressure gradient $dp/dx > 0$. The special case, covering all four distinct regions, was

derived by Falkner and Skan (1930). It corresponds to the inviscid flow over a wedge with an opening angle

$$\beta\pi = 2\,m\,\pi\ /\ (m+1)$$

where the inviscid velocity distribution would be calculated by using

$$ue\,(x) = U_0\ x^{\,m}$$

The solution obtained by Hartree (1937) from Ref.2, is compared to the solution produced by VelocityBL.for in the above Table 3-2. The case $\beta = 0$ is the flat plate problem. The values of $\beta < 0$ represent the adverse pressure gradients. The cases with $\beta > 0$ represent favorable gradients, and $\beta = m = 1$ is the case of planar flow at a stagnation point. Even though the velocity profiles over wedges of various angles were not compared to velocity profiles generated by VelocityBL.for, through comparison of overall boundary layer characteristics it was concluded that the results were fairly close and that the code could be applied to a flow over an airfoil.

3.6 Evaluation of The Thermal Boundary Layer Solver

Before sharing the results generated by the solver, it is worth noting that the convection heat transfer coefficient and heat flux strongly depend on the velocity field within the boundary layer. In this case it comes to a numerical solution for the velocity boundary layer.

If we take a look at Eq. (3.6) and Eq. (3.7), which describe the incompressible constant property laminar boundary layer flow, we can notice that Eq. (3.6) and Eq. (3.7) are uncoupled from Eq. (3.27). The methodology of resolving Eq. (3.6) and Eq. (3.7) for velocity fields u(x,y) and v(x,y) to the exclusion of Eq. (3.27) was applied in this study. From knowing the velocity fields we can calculate other velocity boundary layer characteristics. In the energy equation Eq. (3.27), the temperature and velocity fields are coupled. Thus u(x,y) and v(x,y) must be known before Eq. (3.27) may be solved for T(x,y). Using one numerical solution to calculate another implies that all assumptions, errors, and inaccuracies from the previous solution will be inherited in the later one. The fact that helps in resolving the temperature filed within the boundary layer is that the energy equation Eq. (3.27) is not non-linear as the x-momentum equation. This led to an assumption that the inaccuracies due to non-linearity will not multiply. Basically, all the issues that had to be dealt with in resolving u,v fields, appeared in the application of the thermal boundary layer solver. The considerations that were discussed in the previous section such as grid size, the point of separation, and order of approximations in the process of linearization, will not be discussed here again.

As it was shown in the previous section, there are few benchmark solutions for velocity filed that the result can be compared to, which is not the case for the thermal boundary layer. Therefore, the solution for the thermal boundary layer has been much harder to evaluate than the solution for the velocity boundary layer. Although, there are few expressions offered by different authors for different thermal boundary layer parameters, mainly local Nusselt number, expressions from the Ref. 4 Chapter 7 will be used in this evaluation. The expressions are derived through assumed similarity solution for T in the

boundary layer based on knowledge of conditions in the velocity boundary layer obtained through the Blasius similarity solution for steady, incompressible, laminar flow with constant properties and negligible viscous dissipation.

Local Nusselt number, as shown in Ref. 4, is of the form:

$$Nu_x = h_x\,(x/k) = 0.332\,(Re_x)^{0.5}\,Pr^{1/3} \tag{3.35}$$

The average Nusselt number is defined as,

$$Nu_{AVRG} = 2\,Nu_x \tag{3.36}$$

The average heat transfer coefficient would be then

$$h_{AVRG} = Nu_{AVRG}\,(k/L) \tag{3.37}$$

And total heat transfer rate along the length per unit width

$$Q = h_{AVRG}\,L\,(T_s - T_\infty) \tag{3.38}$$

where k is thermal conductivity, Re_x – local Reynolds number, Pr - Prandtl number, L-length of region, T_s and T_∞ – temperature of the surface and fluid stream respectively.

The results obtained with different Δx, Δy steps for a case of a laminar flow over a flat plate are shown in the Table 3-3. At the end of the table in the last column, the Blasisus and similarity solution for thermal boundary layer characteristics are shown.

	Run1	Run2	Run3	Run4	Run5	Run6	Theory
Step size - y direction dy	0.001	0.001	0.0001	0.0001	0.00001	0.00001	/
Step size - x direction dx	0.0025	0.001	0.0025	0.001	0.001	0.0001	/
Number of grid points y direction M	400	400	600	700	3000	4000	/
Heat transfer rate over plate Q	1589.48	1589.85	1976.15	1995.89	2044.28	2048.70	2344.58
Average convection coefficient H aver	11.65	11.65	14.48	14.62	14.98	15.01	17.18
Average Nusselt number Nu aver	159.95	159.99	198.86	200.85	205.72	206.16	235.94
i - node	201	501	201	501	501	5001	/
coordinate x	0.5	0.5	0.5	0.5	0.5	0.5	0.5
velocity BL thickness δ	0.0070	0.0070	0.0062	0.0062	0.0061	0.0061	0.0062
displacement thickness δ^*	0.00217	0.00217	0.00215	0.00214	0.00212	0.00213	0.00214
momentum thickness θ	0.00082	0.00082	0.00083	0.00082	0.00082	0.00082	0.00082
friction coefficient Cf	0.00162	0.00162	0.00163	0.00164	-0.01598	0.00165	0.00165
thermal BL thickness δt	0.0110	0.0110	0.0100	0.0099	0.0104	0.0099	/

Table 3-3 Finite difference grid size effect on final solution of the solver

The above calculations were performed for flow defined with:

Laminar kinematic viscosity	NI	0.00003084
Free stream velocity	UNinf	10
Density	ρ	0.7992
Air thermal conductivity	cndt	0.0364
Specific heat at constant volume	Cv	727.98
Temperature outside the boundary layer	Tg	300
Temperature of the body on the surface	Tw	27

Table 3-4 Parameters defining the fluid flow for results shown in Table 3-3

From the above results, we can see that the closest the solvers "Tempearture.for" and "Heatflux.for" could get to the solutions from the theory was less than 12 % for thermal boundary layer characteristics. The boundary layer parameters could be calculated with fairly high accuracy even with the coarser grids. It is interesting that the solvers do not give a solution for Cf in Run5 even with a very fine step Δy. Based on results shown in the table and few other solutions for different flow parameters, Δx and Δy steps from Run4 were adopted and used in further calculations. They produced the thermal boundary

layer characteristics that are within 15 % in agreement with the theory and almost identical solution for velocity boundary layer parameters, as the above discussed Blasius solution.

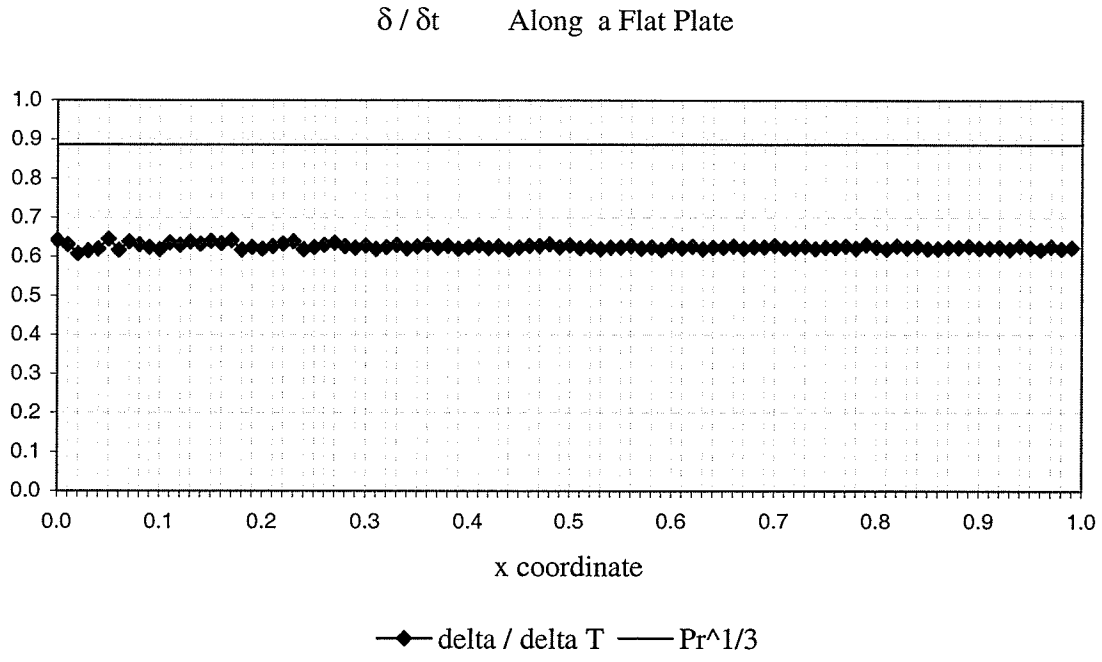

Figure 3-7 Velocity over thermal boundary layer thickness ratio along a flat plate

In this evaluation, more attention was devoted to the behavior of the solution and its relation to velocity boundary layer parameters, such as the ratio between velocity and thermal boundary layer thickness. According to the Ref. 4, the ratio should be $\delta/\delta t \approx Pr^{1/3}$. Although that ratio was not matched, results could be considered satisfactory regarding the difficulties with determining both boundary layer thicknesses, the numerical procedure itself, and the fact that the ratio used for comparison represents the order of solution not the exact number since something like that does not exist.

The displayed graphs characterize the boundary layer flow defined by following parameters:

The BonLay comp region length x direction (m)	L=	0.10000E+01
Number of grid points in the x direction	N=	1001
Step size - x direction :	dx=	0.001
The computation region height y direction (m)	H=	0.0699
Number of grid points in the y direction	M=	700
Step size - y direction :	dy=	0.0001
Free stream velocity (m/s)	UNinf=	0.10000E+02
Fluid laminar kinematic viscosity (m^2/s)	NI =	0.24540E-04
Density of the fluid (kg/m^3)	ro =	0.91320E+00
Fluid specific heat at constant volume (J/kg K)	Cv =	0.72285E+03
Fluid thermal conductivity (W/m K)	cndt =	0.32500E-01
Temperature outside the boundary layer (deg. C)	Tg =	0.20000E+03
Guessed temperature of the blade surface (deg. C)	Tw =	0.20000E+02
Prandtl number	Pr =	0.693

Table 3-5 Parameters defining the fluid flow for results shown on graphs

As shown in the charts below, all the parameters vary along the length of the plate smoothly. Curves shown on the charts follow the behavior suggested by the expressions from the literature. It can be noticed that the average and the local convection coefficient decrease along the plate. The local Nusselt number increases along the plate and closely follows the curve defined through Eq. (3.35).

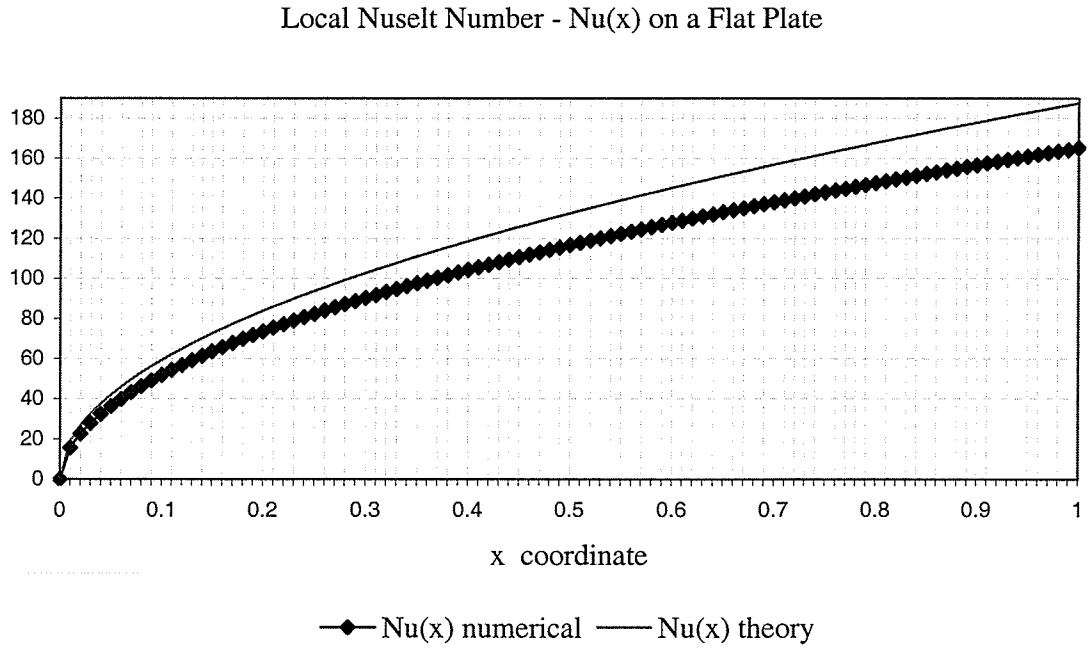

Figure 3-8 Local Nuselt number variation along a flat plate

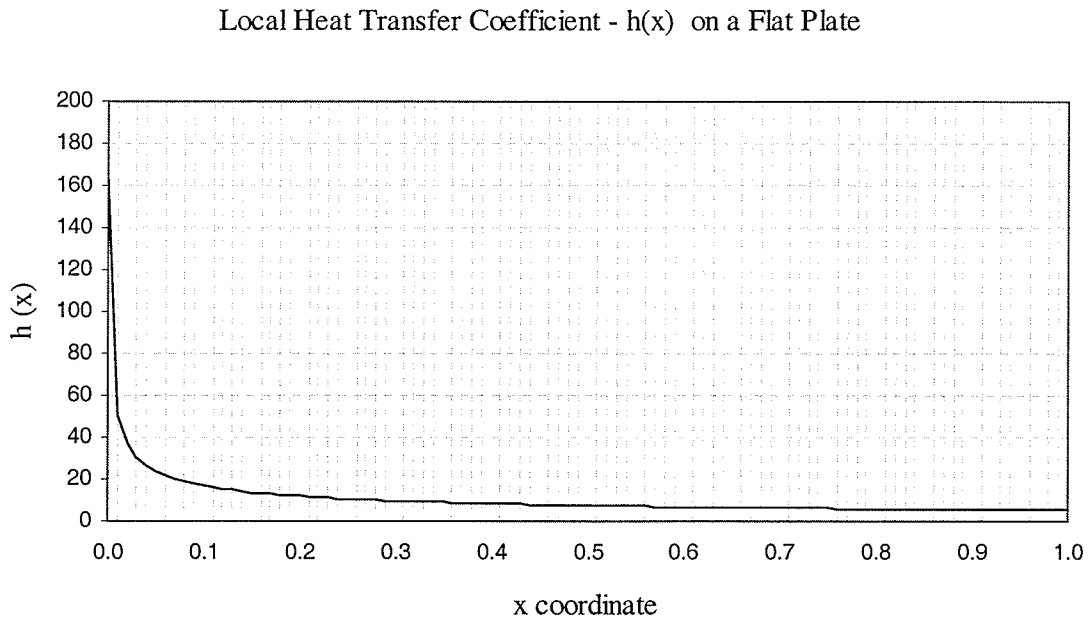

Figure 3-9 Local convection heat transfer coefficient variation along a flat plate

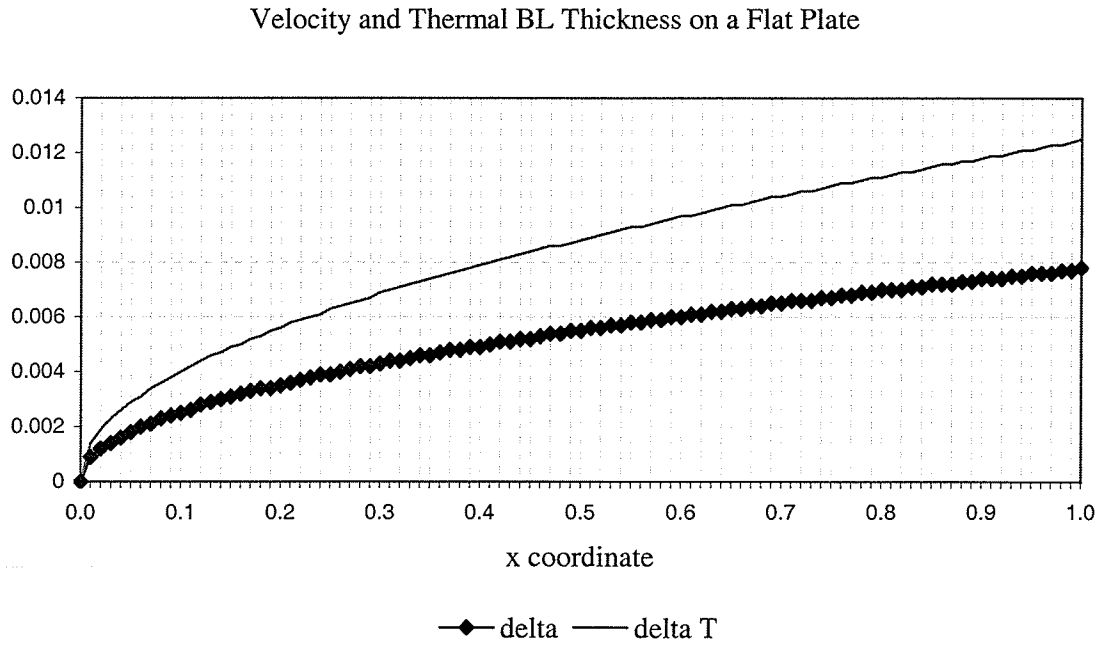

Figure 3-10 Velocity and thermal boundary layer thickness along a flat plate

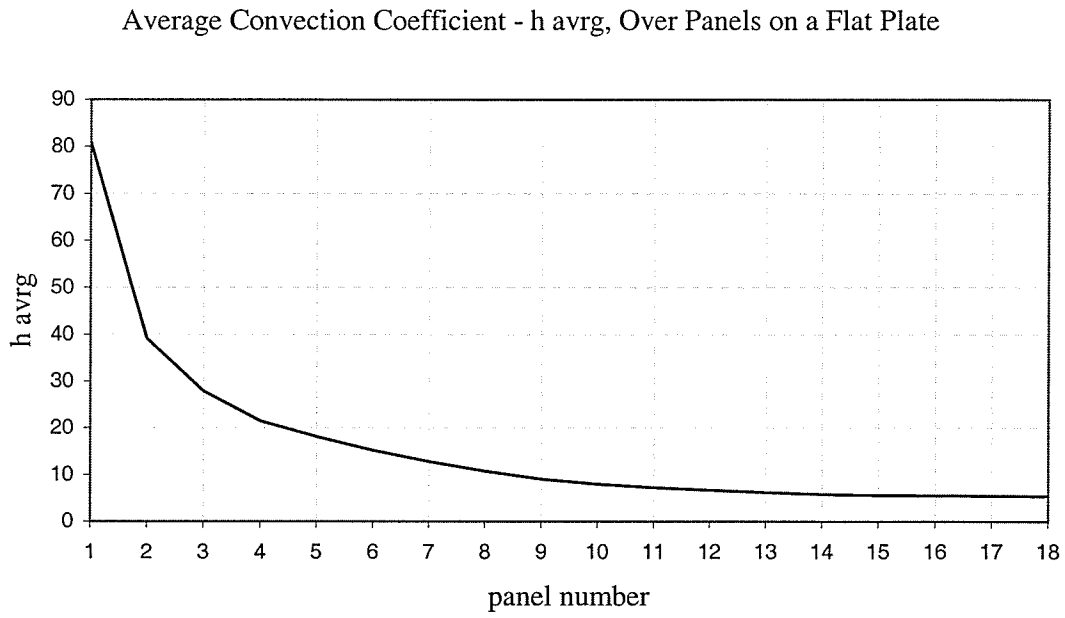

Figure 3-11 Convection coefficient averaged per panel along a flat plate

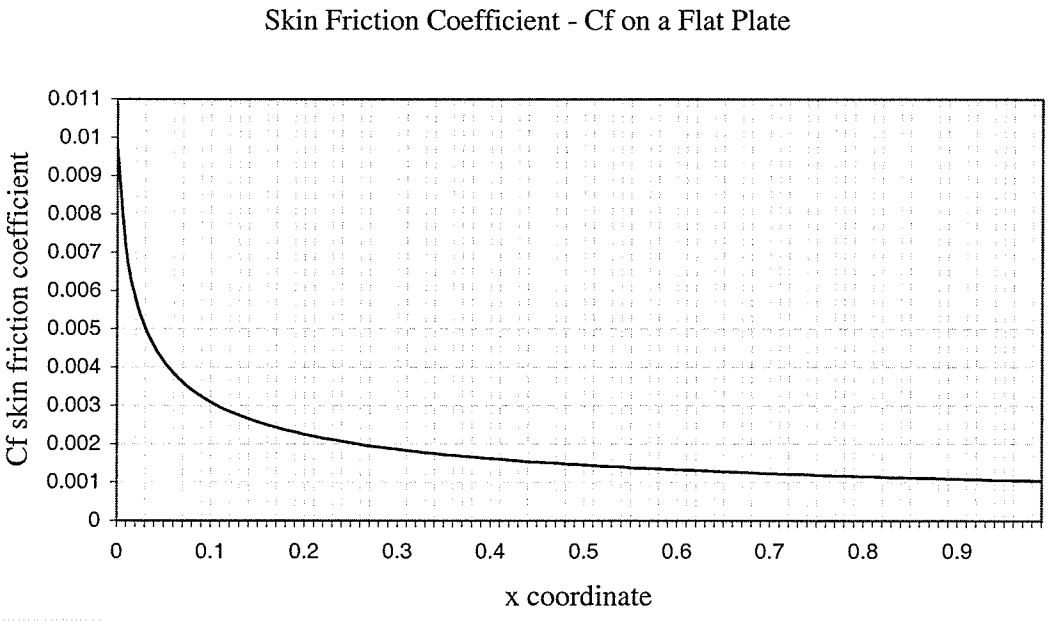

Figure 3-12 Skin friction coefficient variation along a flat plate

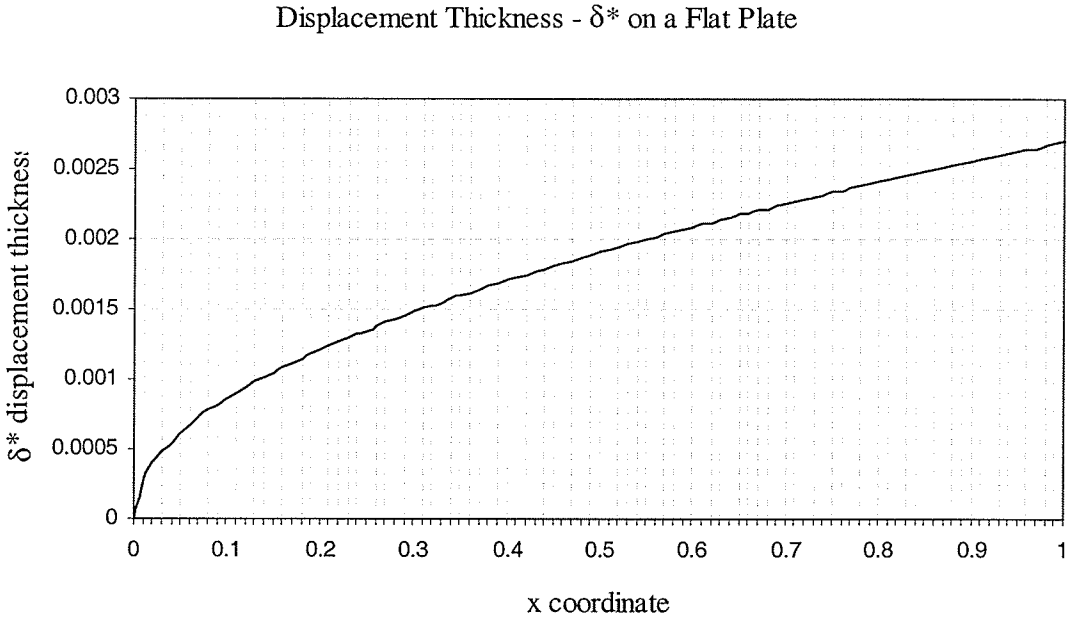

Figure 3-13 Displacement thickness variation along a flat plate

Since the solutions obtained through "Tempearture.for" and "Heatflux.for" for a flat plate have shown a good agreement with the solutions suggested through the theory, it was concluded they could be applied to an airfoil shaped body.

4 The Conduction Solver

4.1 Introduction

This chapter briefly presents the boundary element method formulation used to create the heat conduction solver. The encountered coding issues are brought to the attention of the reader. The code evaluation and test results are demonstrated.

4.2 The Boundary Elements Method

The direct formulation in potential problems can be traced to Jaswon which as early as 1963 with Symm presented a numerical technique to solve Fredholm boundary integral equations, which consisted of discretizing the boundary into a series of small segments and assuming a constant source density within each segment. The first book on boundary elements "The Boundary Element Method for Engineers" by C.A. Brebbia, was published in 1978 but the name BEM was first used in 1977 in two papers by Brebbia and Dominuqez. Until then, boundary integral solutions were almost exclusively the domain of mathematicians and physicists.

The direct formulation is the main subject of the book "Boundary Elements An Introductory Course" by C.A. Brebbia and J. Dominguez. I used it as a main BEM reference and it was the knowledge source for the work performed within the thesis.

It could be said that the boundary element method belongs to the same family of numerical techniques such as finite elements, finite differences, or the vortex method applied for the potential flow analysis in this study. The methods could be seen related

through application of weighted residual or variational type techniques. Only the formulation of boundary elements for potential problems will be discussed here. The numerical implementation of the method and its application used in the code will be described.

4.2.1 Basic Integral Equation

Before the boundary element method is described for potential problems such as steady-state conduction, it should be explained how the starting boundary integral equation required by the method can be obtained. The notation used in this chapter is the same as the one used by Brebbia in Ref. 6. Since the method of weighted residuals provides a very powerful approximate solution procedure that is applicable to a wide variety of problems such as a finite element method, it will be applied here to obtain the desired expression of the boundary integral equation. The problem analyzed in the thesis is the steady–state heat conduction in a homogeneous isotropic solid body, which is like the other potential problems governed by the Laplace equation in a Ω domain. (The potential function is denoted as "u" and its derivative as "q".)

$$\nabla^2 u = 0 \text{ in } \Omega \tag{4.1}$$

Boundary Γ consists of two regions $\Gamma 1$ and $\Gamma 2$ such that $\Gamma = \Gamma 1 + \Gamma 2$. Conditions present on the boundary are as follows

(i) Essential Conditions of the type $u = u'$ on $\Gamma 1$ (4.2)

(ii) Natural Conditions such as $q = \delta u/\delta n = q'$ on $\Gamma 2$

where n is the normal to the boundary Γ and the u'and q' are not the derivatives but known values as shown in Eq. (4.2).

Assume there is an approximate solution of Eq. (4.1) that satisfies the boundary conditions in Eq. (4.2). If such a solution is substituted into Eq. (4.1), it will not

necessarily satisfy the equation exactly, that is, there will be some residual error R. Since there are residuals R, it can be written,

$$R = \nabla^2 u \neq 0$$

$$R1 = u - u' \neq 0$$

$$R2 = q - q' \neq 0 \qquad (4.3)$$

Where u and q are approximate values.

The idea behind weighted residual technique is to make R as small as possible by setting its weighted residual to zero for a certain weighting function *w* such that

$$\int_\Omega R\, w\, d\Omega = 0 \text{ in } \Omega$$

If the weighting function is u*, with derivatives on the boundary $q^* = \delta u^*/\delta n$, there will be

$$\int_\Omega R\, u^*\, d\Omega = \int_{\Gamma 2} R_2 u^*\, d\Gamma - \int_{\Gamma 1} R_1 q^*\, d\Gamma \qquad (4.4)$$

or

$$\int_\Omega (\nabla^2 u)\, u^*\, d\Omega = \int_{\Gamma 2} (q-q')u^*\, d\Gamma - \int_{\Gamma 1} (u-u')q^*\, d\Gamma \qquad (4.5)$$

After integrating the left hand side twice by parts, one obtains

$$\int_\Omega (\nabla^2 u^*)\, u\, d\Omega = -\int_{\Gamma 2} q' u^*\, d\Gamma - \int_{\Gamma 1} q u^*\, d\Gamma + \int_{\Gamma 2} u q^*\, d\Gamma + \int_{\Gamma 1} u' q^*\, d\Gamma \qquad (4.6)$$

The Eq. (4.6) will be transformed into a boundary integral equation by using a special type of weighting function u* called the fundamental solution that represents the field

generated by a concentrated unit charge acting at a point 'i' and in the case of steady-state conduction it would be a concentrated heat source.

The effect of this charge is propagated from i to infinity without any consideration of boundary conditions. Because of this, the solution can be written

$$\nabla^2 u^* + \Delta^i = 0 \tag{4.7}$$

The Dirac delta function Δ^i tends to infinity at the xi and is equal to zero anywhere else but its integral is equal to one. Another characteristic is that the integral of Dirac delta function multiplied by any other function is equal to the value of that function at a point xi

$$\int_\Omega u(\nabla^2 u^*)\, d\Omega = \int_\Omega u(-\Delta^i)\, d\Omega = -u^i \tag{4.8}$$

Eq. (4.6) becomes

$$u^i + \int_{\Gamma 2} uq^*\, d\Gamma + \int_{\Gamma 1} u'q^*\, d\Gamma = \int_{\Gamma 2} q'u^*\, d\Gamma + \int_{\Gamma 1} qu^*\, d\Gamma \tag{4.9}$$

Since the fundamental solution u* represents the field generated by a concentrated unit charge acting at a point 'i', u* and q* are the weighting function and its derivative at that point. For any other point xi, there will be another integral equation.

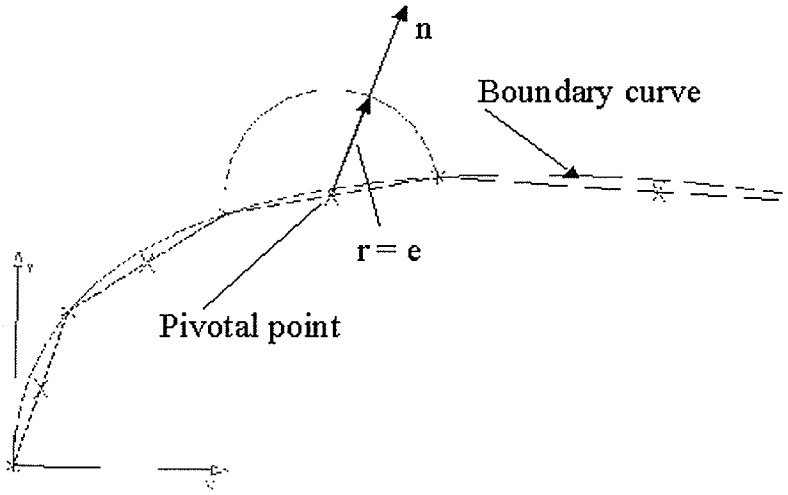

Figure 4-1 Semicircle around a pivotal point

Assume the boundary curve is not smooth but there is a bump like region or semicircle on the boundary that enlarges the domain. If the radius tends to zero, the point "i" considered to be at the center of our semicircle becomes a boundary point and the boundary becomes smooth. Before doing this, the Eq. (4.9) was a general form applicable to any point within Ω domain. If the solution of Eq. (4.7) for two-dimensional isotropic region

$$u^* = (1/2\pi) \ln (1/r) \tag{4.10}$$

is applied into the Eq. (4.9) and if the radius is taken to zero the boundary integral equation for two dimensional problems will be :

$$\tfrac{1}{2}\, u^i + \int_\Gamma u\, q^*\, d\Gamma = \int_\Gamma q u^*\, d\Gamma \tag{4.11}$$

where the integrals are in the sense of Cauchy Principal Value.

4.2.2 The Elements and Matrix Equation

Assume the boundary of a body is divided into M panels as it was when the vortex element boundary integral method was applied and as it is shown on the Figure 2.1. Another assumption would be that the values of u and q are constant over each panel. Therefore, the pivotal points at the center of our elements are the locations that we will obtain a solution for. Based on all of that, the boundary integral equation discretized for a given point 'i' will be

$$\tfrac{1}{2}\, u^i + \sum_{j=1}^{N} \left(\int_{\Gamma j} u\, q^*\, d\Gamma\right) u^j = \sum_{j=1}^{N} \left(\int_{\Gamma j} q u^*\, d\Gamma\right) q^j \tag{4.12}$$

The above equation relates the "i" point where fundamental solution u*,q* is applied to the "j" pivotal point on every other panel through the integrals carried out over the panels. Unknown constant values u and q are taken out of the integrals. After we apply

fundamental solution through the Eq. (4.12) to each node, the system equations is formed and boundary values can be found.

To make writing of equations easier and relations more understandable, panel integrals could be called

$$H'^{ij} = \int_{\Gamma j} q^* \, d\Gamma \qquad \text{and} \qquad G^{ij} = \int_{\Gamma j} u^* \, d\Gamma \tag{4.13}$$

Then the set of equations would be written as

$$\sum_{j=1}^{N} H^{ij} \, u^j = \sum_{j=1}^{N} G^{ij} \, q^j \tag{4.14}$$

Or in a matrix form

$$\mathbf{HU = GQ} \tag{4.15}$$

Where H and G are two NxN matrices and U, Q are vectors of length N.

We should remember here that our boundary conditions were defined in Eq. (4.2) such that we can have known either value of potential or its derivative at a pivotal point. Therefore, there will be only N unknowns in the system that are a mixture of potential and its derivative. After matrices are rearranged such that all unknowns are on one side and all known values are on the other side the equation will be:

$$\mathbf{AX = F} \tag{4.16}$$

Where X is a vector of u's and q's boundary values that are not known.

By resolving the Eq. (4.16) we will know all the values of potential or its derivative on the boundary. By knowing the boundary values, we can calculate u and q at any internal point "i" by using Eq. (4.9) written as

$$u^i + = \int_\Gamma q u^* \, d\Gamma - \int_\Gamma u \, q^* \, d\Gamma \tag{4.17}$$

Where values u* and q* are those corresponding to that particular point. The values u and q are already calculated boundary values. The same notation could be used here which would lead to

$$u^i = \sum_{j=1}^{N} G^{ij} q^j - \sum_{j=1}^{N} H'^{ij} u^j \qquad (4.18)$$

The derivative of potential at a point "i" or internal flux in any direction k can be obtained easily if we derive Eq. (4.17)

$$(q_{xk})^i = (\delta u/\delta x_k)^i = \int_\Gamma q\,(\delta u^*/\delta x_k)^i \, d\Gamma - \int_\Gamma u\,(\delta q^*/\delta x_k)^i \, d\Gamma$$

Integrals in the above expressions are usually obtained through numerical integration.

A relatively simple FORTRAN subroutine used within the main code for solving steady-state heat conduction in a homogeneous isotropic solid body is basically an application of the above presented theory. The boundary values (u's and q's) produced by the subroutine are constant over each panel since the constant elements were used.

The code used was taken from the " Boundary Elements An Introductory Course" C.A. Brebbia and J. Dominguez, however it was modified to fit within and to produce additional variables required by the main convection-conduction coupling code. The boundary element code input and output files had to be changed accordingly from the files shown in the book.

4.3 Coding Issues

Compared to the finite difference method explained within Chapter 3., it could be said that BEM has more complex mathematical foundation, which makes its numerical implementation much easier. The above described boundary element theory should not be that complex to apply but there are few issues that everybody should be aware of before writing a code.

Attention should be devoted to the weighting function. In this specific case, it was the fundamental solution applied at a particular pivotal point or position of charge. The weighting function has to have a relationship from a point of application to a panel on the boundary. The integrals to be resolved in the equations are carried out over a panel. Values of potential and its derivative are taken out of integrals since they have been assumed to be constant over a panel. The only function left under an integral will be the fundamental solution or its derivative. Therefore, the influence of a panel to a point of charge and vice versa is established through a fundamental solution and its integration over a panel on the boundary.

A good practice would be to change coordinates over a panel to a homogeneous ξ coordinate and limits of integration from -1 to 1 in order to facilitate integration. Numerical integration formulae used in the code is Gaussian quadrature for cases $i \neq j$. For elements $i=j$ a more accurate integration might be required such as higher-order integration rules or a special formula. In this particular application for a case of constant elements Hij and Gij integrals were computed analytically. Thus, a key in creating a reliable BEM code would be an appropriate and accurate numerical integration technique.

Another important issue comes from implementing boundary conditions that contain values of u and q over boundary. As it was shown earlier, that will result in N unknowns in the system as mixture of potential and its derivative. Therefore, system Eq. (4.15) has to be reshuffled by moving columns of H and G from one side to another in order to

implement boundary conditions. After it has been done, we'll have system Eq. (4.16) which is not difficult to solve.

Routine SLNPD suggested in the book was used within BEM subroutine to resolve system of equations. Within subroutine PANEL.for solving the potential flow, system of equations like the one in Eq. (4.16) was resolved by the matrix inversion method. Through the application of the code, SLNPD methodology was shown to be faster than the matrix inversion method.

Other than these few issues, numerical implementation of BEM is pretty straightforward. The BEM formulation explained and used within this study is based on "constant" elements. There are several other BEM formulations that are not discussed here but certainly would be valuable options and subjects of study for somebody who wants to learn more about the technique and its application.

4.4 Evaluation of the code

4.4.1 Square Domain Example

The first example considered in the evaluation of the BEM code and its results was a closed square domain shown in Figure 4-2. The boundary was divided into 12 constant elements, with three on each side of the square. Boundary conditions specified through an input file were the known temperature values on the vertical walls and the known heat flux on horizontal planes of the domain. The required solution of the problem was obtaining the unknown temperature and heat flux values on the boundary and at five internal points.

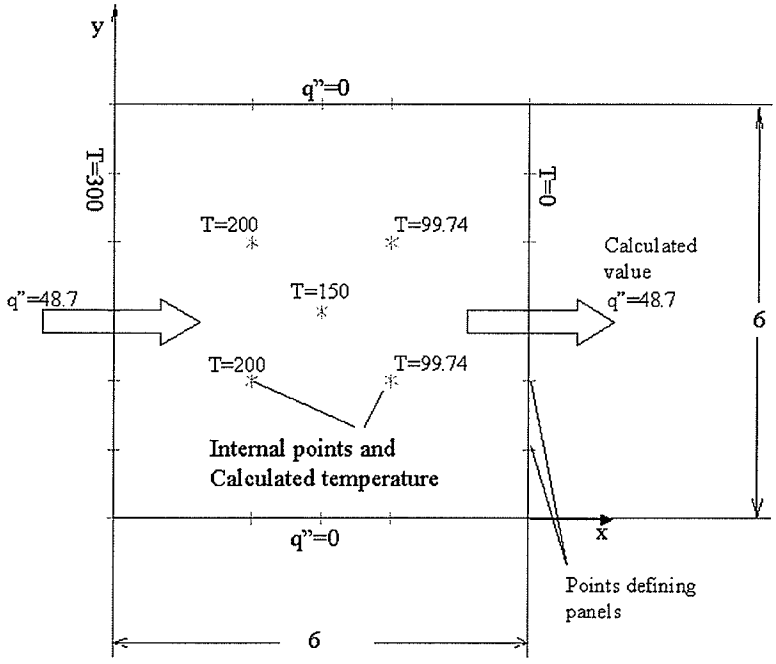

Figure 4-2 Square domain considered with calculated values shown

The square domain, the boundary conditions and results are all shown on the same sketch in Figure 4-2. The first thing that comes to mind is that very few elements were used and feasibility of obtaining valuable results with such a coarse mesh. For boundary conditions chosen, it is not difficult to evaluate results due to existence of the conduction rate equation or Fourier's law that implies that direction of heat flux is normal to a surface of

constant temperature. From Fourier's law we can calculate heat flux and temperature within the domain that would represent the exact solution.

Considering the simplicity of the method and very few elements used, the excellent agreement with the exact solution could be noticed. The heat flux on vertical walls was quite close to the exact solution of 50. The values calculated at the internal points as a weighted average of boundary values were even closer to the exact solution as seen in the figure.

4.4.2 Rectangular Tube Example

Second case considered was a rectangular tube or a rectangular domain with a rectangular inner boundary. This time the mesh was finer and all surfaces including the internal ones were specified to be at a constant temperature, see Fig. 4-3.

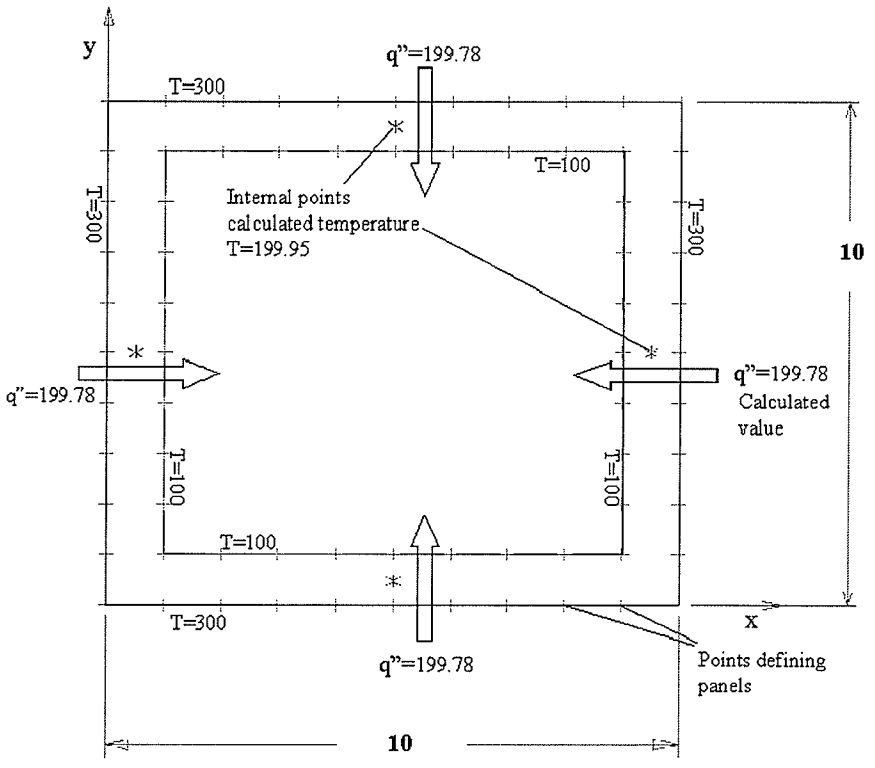

Figure 4-3 Rectangular tube domain considered with calculated values shown

The internal points, where temperature and temperature derivative were calculated, were specified to be in the middle of each wall. Such a choice was made in order to stay away from the corner zones were multidimensional effects of heat transfer have to be considered. In the middle of the walls, the temperature gradient is significant in only one direction and Fourier's law could be applied. The exact solution obtained through Fourier's law is again used to evaluate results produced by the code.

In the Fig. 4-3, we can see that results in the middle of the walls are in accordance with the exact solution that varies linearly. The accuracy is better than in the previous case of the square domain, which is probably due to finer mesh.

The code used produces a solution for problems that have more than one surface, with internal and external boundaries. It was used in calculations for both examples shown here since the input to the code can easily be changed to specify only one boundary over a domain.

5 Integrating the Elements – The Complete Code

5.1 Introduction

This chapter presents integration of the numerical methods used in the analysis of conjugate heat transfer phenomena on a model of a cooled airfoil shaped object. Major coding issues that have to be resolved are discussed. The final results of the integrated numerical methods are presented.

5.2 Applied Methodology and Coding Issues

The sequence of the solution could already be noticed through the order in which the applied numerical methods were presented in this study. Although progression in obtaining the final results could be easily followed in the main code given in the Appendix, there are a few issues that should be discussed.

The first step would be obtaining the inviscid flow velocity distribution around the object. Since the issues associated with the implementation of the surface vorticity method were discussed in Chapter 2, they will be mentioned here very briefly. When it comes to the coupling of solutions of the potential flow and the boundary layer solvers, the most important concern determining the quality of the solution is number of the panels and their size. The velocity distribution around a body will be more accurate if we employ more panels of a smaller size on the surface, but that is not something that has to be done.

The approach that was taken here is to have much more smaller panels in the leading and the trailing edge zones of the airfoil due to the curvature of the surface, and longer panels around the half way of the chord. Airfoil aerodynamic characteristics given in the literature are usually accompanied with the data points defining the contour of the airfoil. Those points defining the airfoil were used as points defining the panels. Since the velocity distribution did not match the literature data in the leading and trailing edge zones, a few panels had to be added. The solution is obtained at a point located at the center of each panel.

To provide an input for the boundary layer solver, which is an implicit finite difference method employed, the potential flow solution has to be specified at the finite difference grid locations. At the same time, the panel method grid nodes have to be the nodes of the boundary layer solver. A separate subroutine INTERPOL.FOR is employed in specifying the boundary conditions to the finite difference grid, which are the values interpolated between values of the potential flow velocities at points determining the start and the end of a panel. The velocities at the start and end point of a panel were determined through interpolation between potential flow solution calculated in the middle of each panel. Using this approach, interpolation in between interpolated values, works very well, however it brings inaccuracies into the solution. The subroutine keeps information of the number of interpolation points along each panel since that number will be needed afterwards when averaging and integration of the finite difference solution is performed. The issues associated with a choice of uniform or non-uniform grid, the best finite difference grid to use, and their effect to the solution were discussed in Chapter 3.

Boundary layer characteristics such as local skin friction coefficient, boundary layer thickness, momentum thickness, displacement thickness, and point of separation were calculated within subroutine "velocityBL2.for". In order to perform the analysis on airfoils with sharp and blunt trailing edge, there are two subroutines in the main code: "VelocityBLsh.for" is used for sharp trailing edge airfoils and the "velocityBL2.for" is used for blunt trailing edge airfoils. They are basically the same and the only difference is

the criterion specifying the location of the point of separation in case the flow separates at the very end of the trailing edge.

Another module affected by the difference in trailing edge shape is the subroutine attaching a "wake body" to the airfoil. The appropriate subroutines have to be employed to accommodate the fact that the flow eventually stays attached to the pseudobody through the iterations to calculate the aerodynamic characteristics of the airfoil. The routine of attaching the "wake body" to the airfoil, and the procedure for the determination of the pressure and the friction drag coefficients within the pressure loop will be discussed here very briefly. In addition to that, it should be mentioned that the aerodynamic characteristics determined through that procedure were one of the parameters defining the quality of potential flow solution and coupling between the vortex method and the boundary layer solvers.

The fact that the boundary condition at the outer edge of the viscous region or the velocity Ue obtained from potential flow subroutine is readily introduced into the finite difference calculations did not help with the coupling of the two solutions. As it was explained in Chapter 2., the obtained inviscid velocity distribution was dependent on the coordinates of the panels on the airfoil surface as well as the number of the panels. The point of separation was crucial in determining both the form drag and the friction drag coefficients controlling number of iterations that had to be performed within the pressure loop. Within the pressure loop, new airfoil shape would be formed after the displacement thickness, obtained from velocity boundary layer solver, is added to the original geometry. The potential flow calculation would be performed on the new body shape including a "wake body" attached to the new airfoil, followed by the boundary layer calculations with a new boundary condition.

It has been shown that the boundary layer solution was very sensitive to a change of values obtained as explained and specified on its outer edge. The skin friction coefficient and the location of the point of separation were the boundary layer characteristics of the most importance in coupling the solutions. The cases where that fact could be easily

noticed are when different points defining the contour of the airfoil are specified at the leading edge zone. If the end point of the first panel on the airfoil was specified at 0.5%, 0.7%, or 0.8% of the chord, there would be a different point of separation obtained for each of these input files, as shown in Table 5-1. Even such a small change in coordinates of points defining the airfoil would produce a different inviscid velocity distribution. If the potential solutions for each of these three input files were shown on the same graph (see Fig. 5-1), the difference would be hardly noticeable. However, it would certainly be noticeable when it comes to the point of separation determined through boundary layer calculations. The other boundary layer characteristics were susceptible to a change in inviscid velocity distribution around the airfoil as well. They are not important in the whole procedure of coupling the solutions but certainly illustrate quality of the boundary layer solution.

% of chord the end point of first panel is specified at	Point of separation location at x/chord	Cf at 0.9 x/chord	θ at 0.9 x/chord	δ^* at 0.9 x/chord	δ at 0.9 x/chord
0.4	0.98125	0.00427	0.00018	0.00056	0.0011
0.5	0.97500	0.00268	0.00027	0.00084	0.0017
0.7	0.95375	0.00127	0.00042	0.00142	0.0027
0.8	0.91875	0.00080	0.00049	0.00178	0.0032

Table 5-1 Dependence of the boundary layer characteristics on the panel's grid

Most importantly, susceptibility of the boundary layer velocity solution to slightly different inviscid velocity distribution has certain influence to the main problem considered here, the conjugate heat transfer. In the Figure 5-5, it can be observed that for a change of location of the point of separation within 8%, surface temperature distribution would change roughly the same percentage. The change in the solution propagates even through the boundary element results. Even the averaging over panels that takes place when the panel's heat flux is calculated does not eliminate the effect. All of that required a change to the total heat flux transferred to the body surface, due to the different coordinates used to define the panels, not to be disregarded. The final or the converged solution for surface temperature was changed accordingly.

INPUT data file name (airfoil geometry/panels): 0012sq8		
BEM INPUT data file name (b/c and grid points of panels on inner boundary): bem12		
The BonLay comp region length x direction (m)	L=	0.10000E+01
Number of grid points in the x direction	N=	801
Step size - x direction :	dx=	0.12500E-02
The computation region height y direction (m)	H=	0.39900E-01
Number of grid points in the y direction	M=	400
Step size - y direction :	dy=	0.10000E-03
Free stream velocity (m/s)	UNinf=	0.10000E+02
Aerofoil chord length (m)	Chord=	0.10000E+01
Free stream velc angle with x-axis (degrees)	ALFinfD=	0.00000E+00
Fluid laminar kinematic viscosity (m^2/s)	NI =	0.24540E-04
Density of the fluid (kg/m^3)	ro =	0.91320E+00
Fluid specific heat at constant volume (J/kg K)	Cv =	0.72285E+03
Fluid thermal conductivity (W/m K)	cndt =	0.32500E-01
Blade material thermal conductivity (W/m K)	cndtbl =	0.17700E+03
Conductivities ration cndt/cndtbl	conRat =	0.18362E-03
Temperature outside the boundary layer (deg. C)	Tg =	0.20000E+03
Guessed temperature of the blade surface (deg. C)	Tw =	0.20000E+02

Table 5-2 Fluid stream and the finite difference grid parameters the results shown were obtained for

Since the boundary layer solver does not give the solution after the point of separation, the values in the separated flow zone were not even used in the calculations. The other reason they were disregarded is the fact that there is basically no heat transfer after the point of separation.

The described relation between the size of the panels, different grid points used to define the panels, and the final results is fairly common among all numerical methods used. For any of the numerical methods used today in any type of analysis, even for commercially available programs, a mesh producing a quality and usable solution has to be determined before anything else is done. As always, the evaluation is performed through comparison with the available theoretical and experimental data. The same was done here, which is why the panel grid having the first panel ending at 0.8% of the chord was accepted as the one to be used in evaluation of the code and its results.

After a temperature distribution within a boundary layer solver is obtained, the results have to be transformed into the input form required by the body heat conduction solver.

One of the main advantages of the boundary element method as it was shown, is that it gives very accurate results even with a few elements employed. That is the feature that keeps the computing requirements down and certainly it was to be maintained within the conduction solver used here. The grid utilized for the boundary element method is the same grid used in the potential flow analysis, which is the points defining the panels on the surface of the airfoil. The solution of the heat conduction solver is obtained at a pivotal point in the center of the panel due to a specific boundary element formulation applied, that is based on "constant" elements. Remember, that is the same location where the solution of the potential flow is computed.

In the first iteration, the thermal boundary layer solver calculates temperature distribution within the boundary layer based on the assumed surface temperature. Temperature field around the body as calculated does not mean so much to the heat conduction routine. The required boundary condition for the conjugate heat transfer on the body surface is that the heat flux and the temperature are continuous. So, the temperature on the surface was used to calculate the temperature distribution around the airfoil. Heat flux was determined from obtained temperature field and used as a boundary condition on the surface for the conduction analysis within the body. Thus, input data required by the heat conduction routine is the heat flux as a constant value over each panel, since the boundary element procedure is based on "constant" elements formulation. From a coding prospective, that means that the values from the finite difference grid have to be brought to the boundary element grid. The procedure employed in the code is that along the chord, local convection coefficient and local heat flux are calculated on the finite difference grid. Those values are then averaged over each panel and the determined average heat flux over each panel is used as an input to the heat conduction solver BNELEM.FOR. The integral of the heat flux function over each panel, calculated by the trapezoidal rule, is divided by the length of the panel in order to obtain the value of the average heat flux over a panel. By doing this, the other portion of continuous boundary conditions on the body surface is fulfilled. In addition to the heat flux on the boundary, the set of input data such as coordinates of internal points where we might want to know the temperature and the heat flux, the number of internal boundaries such as cooling ducts, and boundary

conditions associated to them are input specified to the heat conduction subroutine. Those data has to be specified through a separate input file. The result of the heat conduction subroutine is improved surface temperature. It becomes the boundary condition for a new thermal boundary layer solution. That means that the results are imposed on the finite difference grid, which is done through a separate subroutine in the convection-conduction loop of the main code. Iterations continue until the temperature on the body surface is converged. The criterion used to stop iterations is that the difference between the values of calculated temperature in subsequent iterations over each panel has to be less than 0.1 degree.

Another important fact in execution and applicability of the code is the fluid involved and the material of the blade or to be more specific one of their characteristics, the thermal conductivity *k*. Since no fluid motion occurs very close to the surface, energy transfer takes place only by conduction. Therefore, at any distance x from the leading edge, the local heat flux is obtained by applying *Fourier's law* to the fluid as shown in expression (3.3). The conduction heat transfer process in the blade is quantified in terms of the same rate equation, *Fourier's law,* where the thermal conductivity basically determines the heat transfer rate. Therefore, there is a new parameter created and used in this study named *the conductivities ratio CR.* It is defined as a ratio of the fluid over the blade thermal conductivity,

$$CR = k_{\text{fluid}} / k_{\text{blade}} \tag{5.1}$$

As it will be shown in the next section, the CR has a huge influence on the number of iterations required to reach the converged surface temperature. In addition to that, if the values CR is beyond certain value the solution is not converging, which sets the limits of code application.

The traveling of the results between solvers and the grids required by each of them is one of the main issues when it comes to integrating the numerical methods used. The results are sort of captured in the averaging – interpolation loops.

The influence of the applied numerical methods' results traveling between different grids and the averaging-interpolation routines, on the quality of the results, has not been considered in this study.

5.3 Results and Evaluation

5.3.1 Potential Solution

The airfoil analyzed in the study is the well known NACA 0012. Since results of potential flow solver were discussed and compared with available data from literature as shown on the Fig. 2-3, only inviscid velocity distribution obtained for two different grids is presented here.

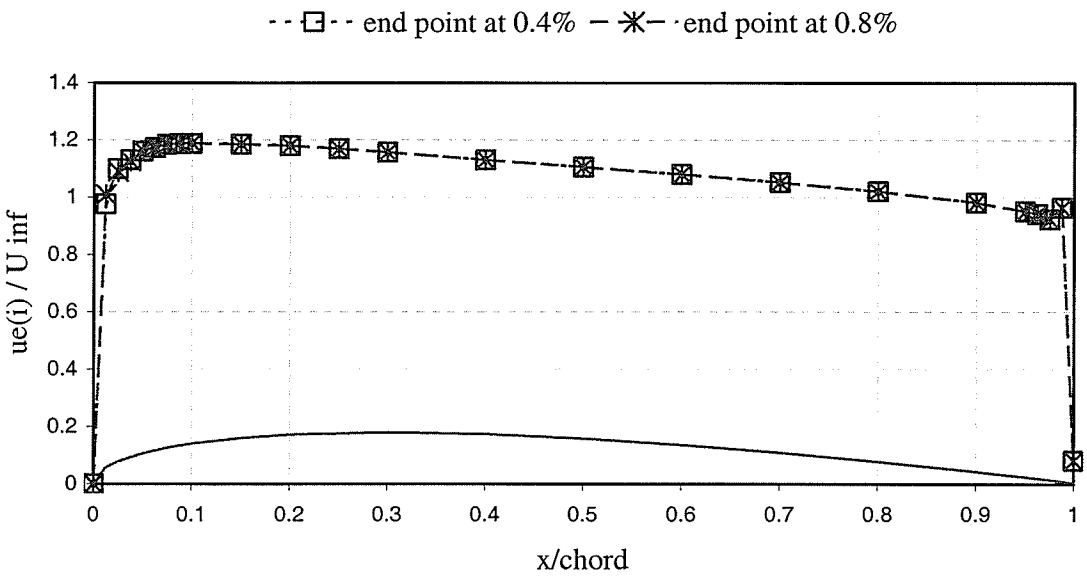

Figure 5-1 Velocity distribution along the airfoil chord for two different panel grids

In the Fig. 5-1, potential flow solution along the chord is shown for two test cases: 1. end point of the first panel on the airfoil was specified at 0.4% and 2. end point of the first panel at 0.8% of the chord. As it was discussed in the previous section, the difference in velocity distribution is hardly noticeable. However, the choice of the grid point has an obvious effect on the boundary layer solution and consequently on the final solution of the main code.

5.3.2 Boundary Layer Solution

Results obtained through the boundary layer solver fit very well within boundaries where they are supposed to be. Curves showing the boundary layer characteristics along the chord follow the expected behavior and trend. Comparing the characteristics like skin friction coefficient *Cf* or displacement thickness δ^* at a certain location on the airfoil with data from literature would be very hard. Even for a well known airfoil such as NACA 0012, specific values of the boundary layer characteristics, for a laminar fluid flow with parameters as defined, would be difficult to find. If the obtained solution for a flow over an airfoil is compared with the solution for flow over a flat plate, where the flow in both cases was defined by the same flow parameters and almost the same grid was used, it could be concluded that the results are reasonable and could be used.

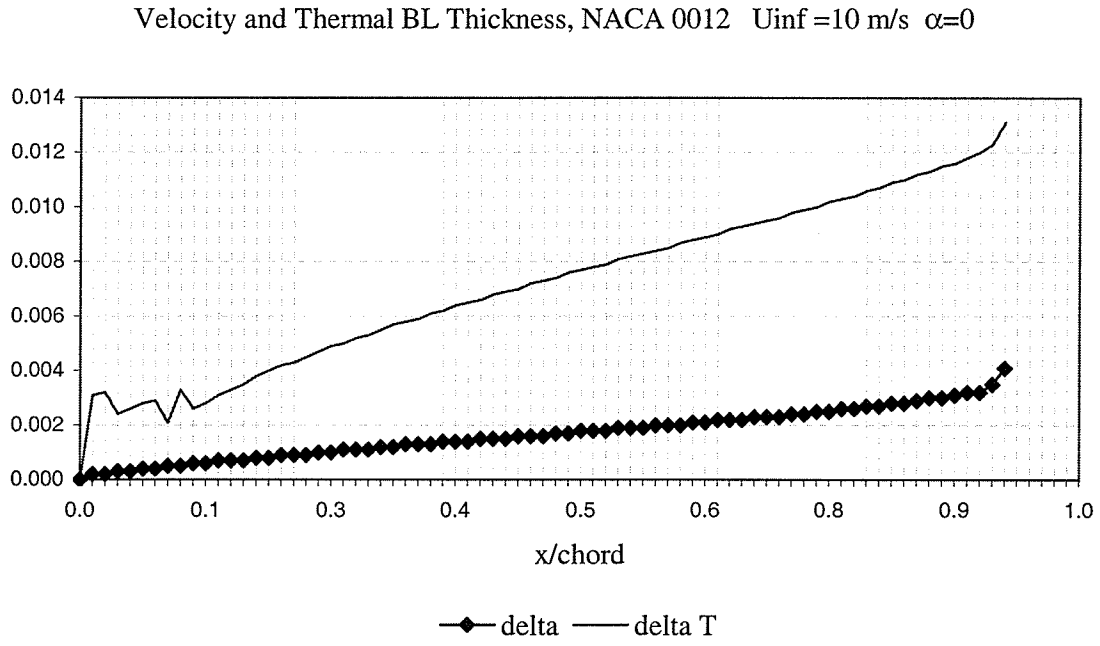

Figure 5-2 Velocity and thermal boundary layer thickness along the airfoil chord

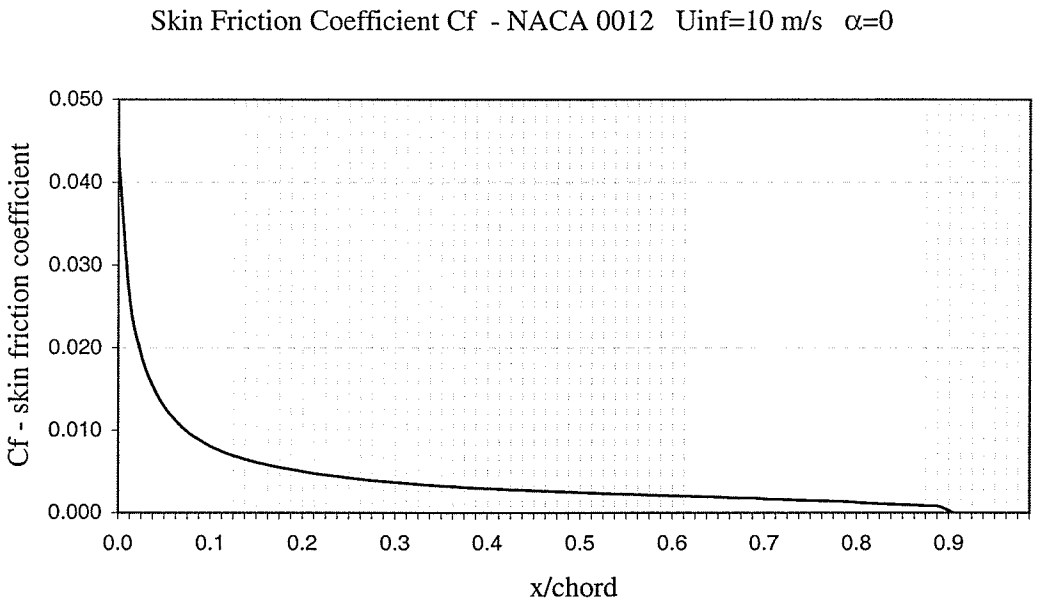

Figure 5-3 Skin friction coefficient Cf along the airfoil chord

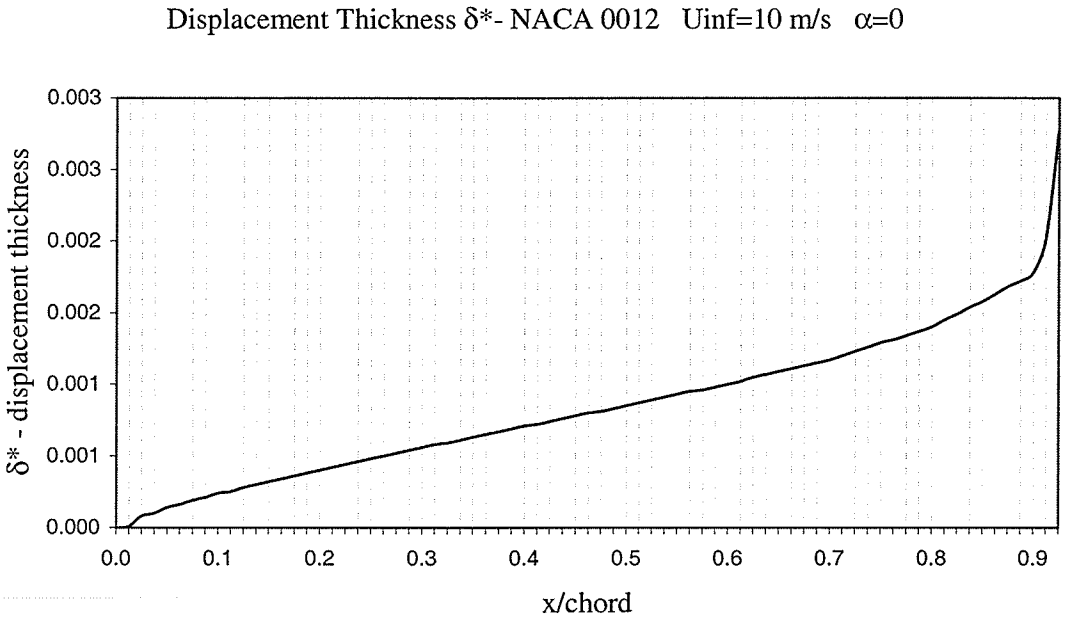

Figure 5-4 Displacement Thickness δ* along the airfoil chord

For the airfoil used in the analysis and for the fluid flow as defined, the location of the point of separation should be about 90% of the chord. If we take a look at the *Cf* coefficient curve shown here, we can see that the values obtained match the expected very well. In addition to that fact, the analysis was done on the same airfoil but for *Re* defined by $U_\infty = 4.8$ m/s and $U_\infty = 3.2$ m/s, that are lower than the one used here, $U_\infty = 10$ m/s. For a lower *Re* numbers, thickness of the boundary layer was increasing and the point of separation was moving toward the leading edge. That observable fact happens in reality, which confirms the solution even more.

5.3.3 Conjugate Heat Transfer Solution

The final results of a computer code used to analyze the conjugate heat transfer over a cooled airfoil shaped object are presented on the following charts. Since the coding issues and their influence to the solution have been already discussed, the charts should be considered in connection with everything that was said in preceding sections. One of the concerns already discussed is the impact of the choice of panel grid on the final solution of the code. The solution, shown as a converged temperature on the surface of the body and its dependence on the location of the end point of the first panel, is shown in the Figure 5-5.

Surface temperature derivative in the direction normal to the surface, accompanied with the surface temperature distribution along the chord, is presented in the Figure 5-6. The flow around the airfoil was defined by the parameters specified in the previous section. In this case, there was only one cooling duct at the center of the body with a cooling fluid at 20C. The effect of the cooling duct on the surface temperature is obvious. The reason why the temperature of the body surface close to the cooling hole is so close to the cooling fluid temperature is the high thermal conductivity of the body material. In other cases when the body material was different, a different surface temperature was obtained.

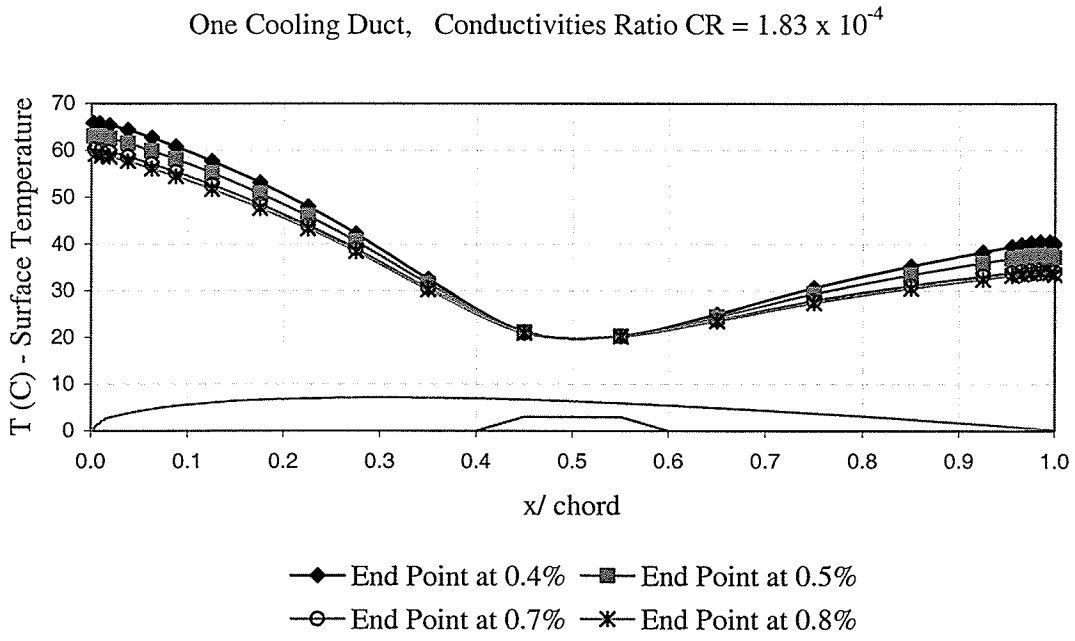

Figure 5-5 Surface temperature dependence on the panel grid choice, NACA0012 surface temperature distribution along the airfoil chord

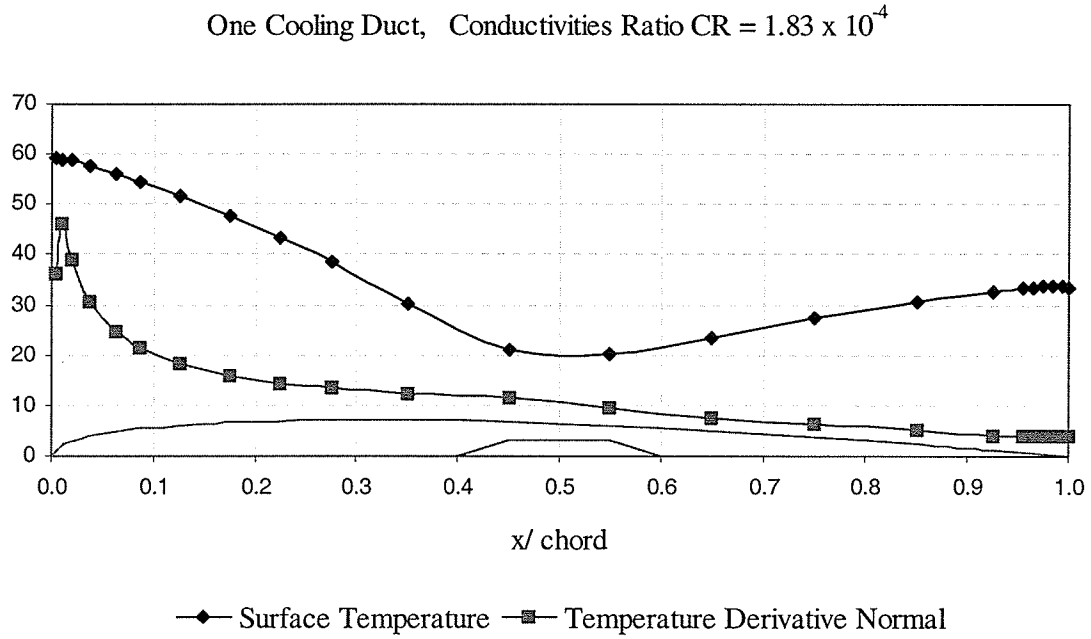

Figure 5-6 Surface temperature distribution and temperature derivative normal to the surface, one cooling duct

The surface temperature derivative in the direction normal to the surface closely follows the panel's convection coefficient down the chord, as shown on the next chart. The values are different as expected, but behavior of the curves is the same, confirming the continuous boundary conditions applied.

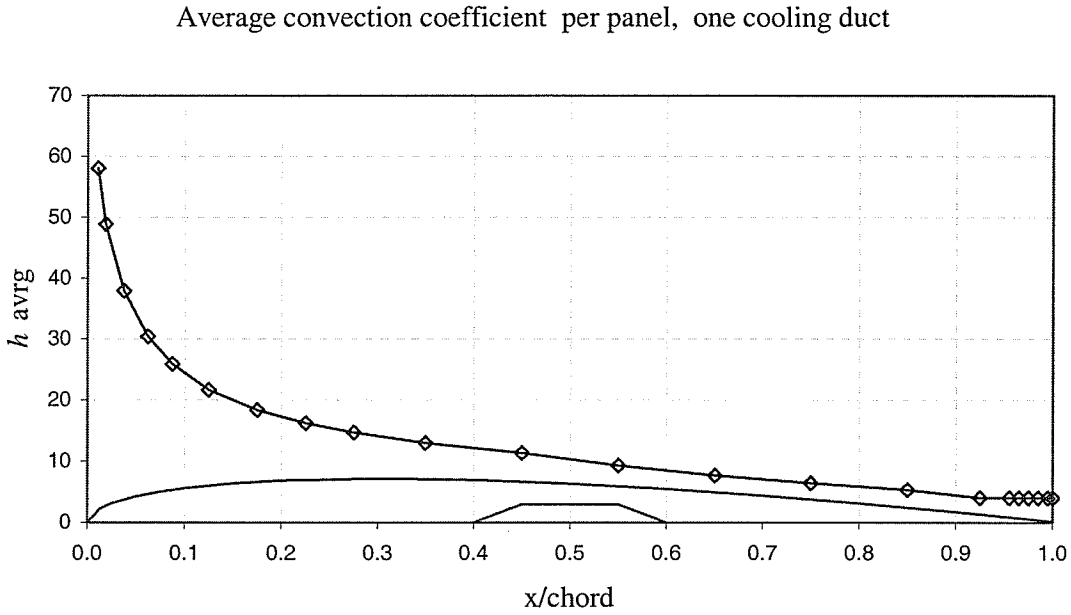

Figure 5-7 Convection heat transfer coefficient *h*, average value per panel

As shown on Fig. 5-8, two cooling ducts would be more than desirable in case we want to lower the surface temperature across the first half of the chord. The number of the cooling tubes, their location, and the boundary conditions along their surfaces are specified within a separate input file from which the heat conduction routine reads the data. It is not hard to change the input file and obtain the desirable temperature distribution determined by the number, size, and location of the cooling tubes. This is something that would probably be done in case the best optimum design of a turbine blade is the intention.

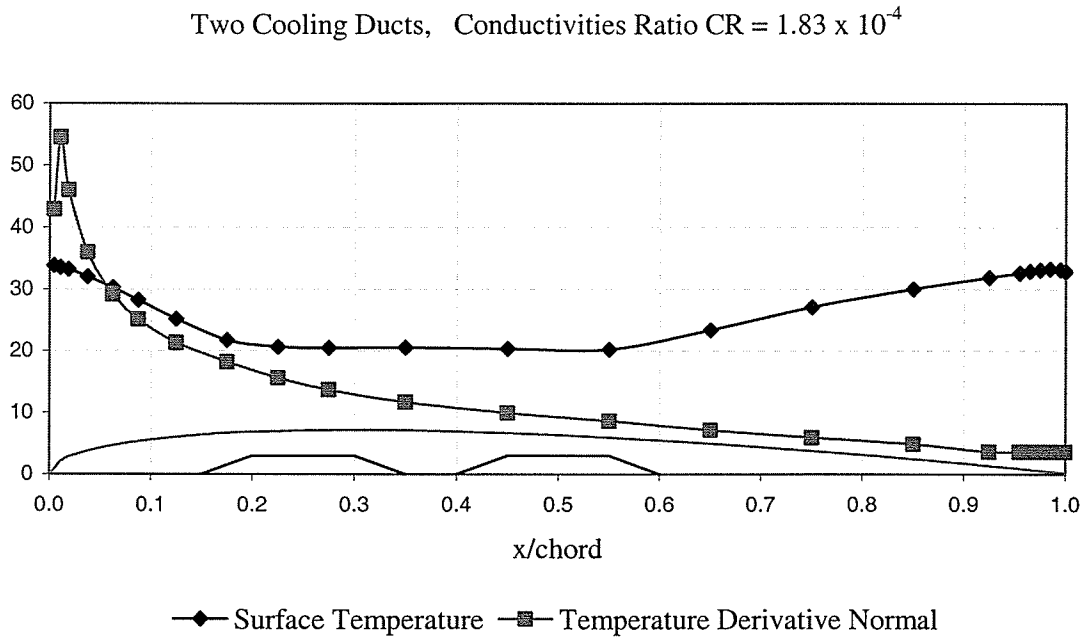

Figure 5-8 Surface temperature distribution and temperature derivative normal to the surface, two cooling ducts

On the following two charts, the convergence of surface temperature at three different locations on the airfoil for two different CR factors is shown. It can be noticed that surface temperature converges much faster for a body made of material with higher thermal conductivity or lower CR factor. The point locations are expressed as a ratio of x coordinate over the airfoil chord.

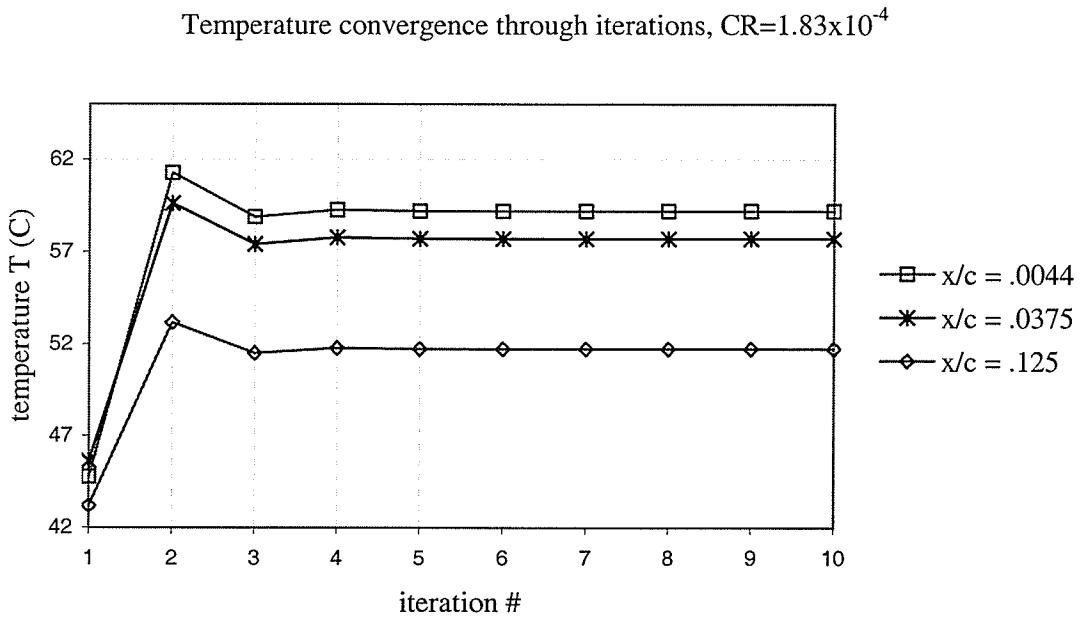

Figure 5-9 Convergence of surface temperature at three points on the airfoil contour CR=1.836x10^{-4}, one cooling duct

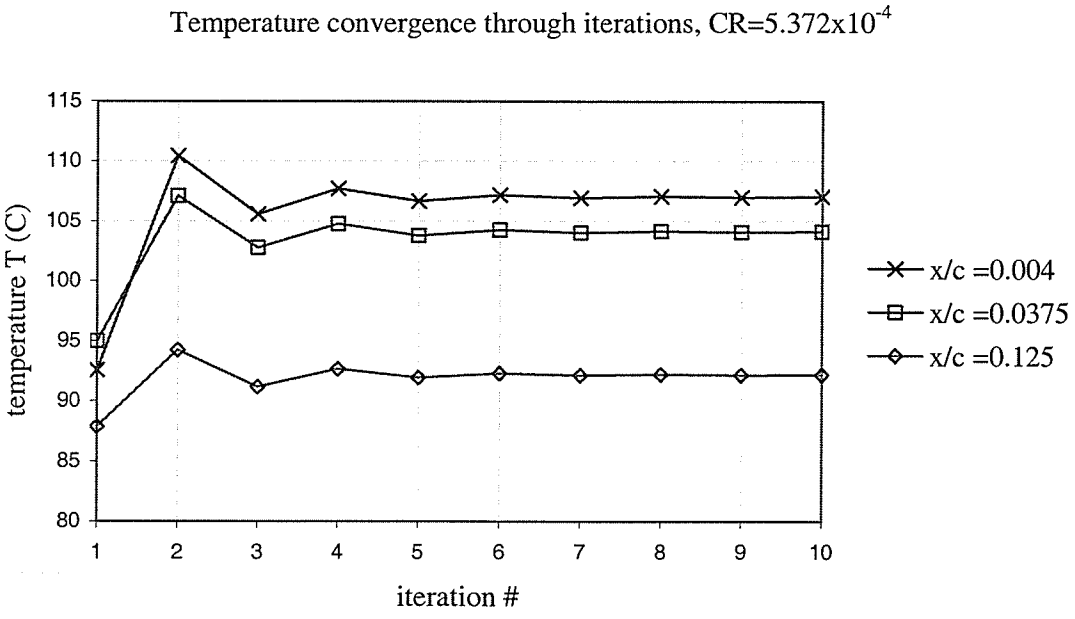

Figure 5-10 Convergence of surface temperature at three points on the airfoil contour CR=5.372x10^{-4}, one cooling duct

The last chart shown is the number of iterations required for the temperature over the whole surface to converge as a function of CR factor. As displayed, the higher the CR the more iterations it takes to compute the final solution. Also, the below graph illustrates that after a certain value of the CR parameter, the temperature on the surface will not converge at all, meaning that the code will not produce a solution. That value of CR could be considered as a limit of code application.

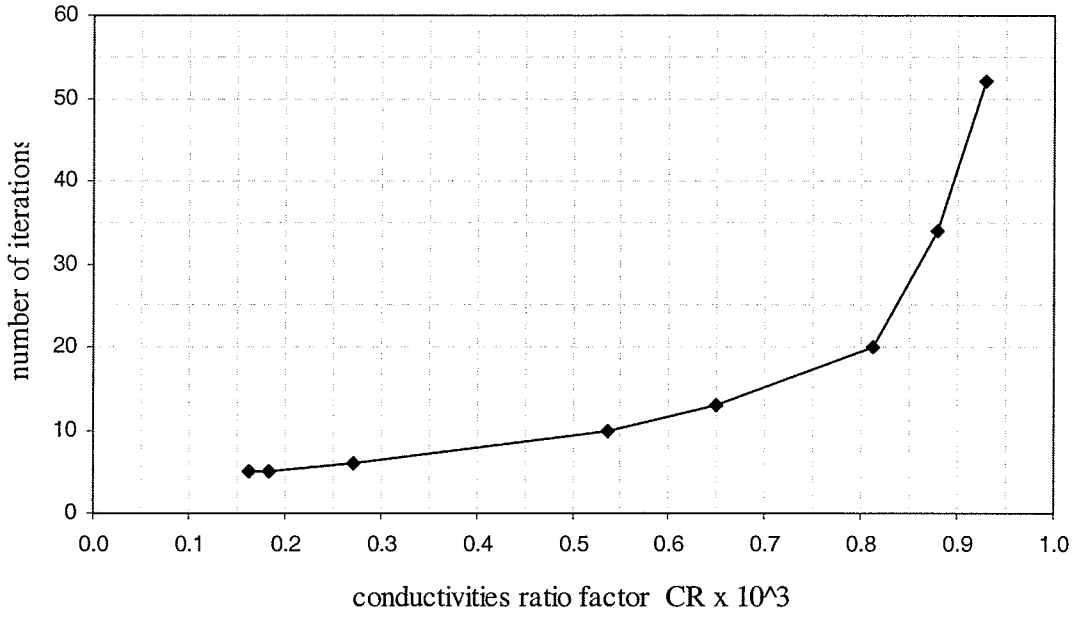

Figure 5-11 Convergence of a solution as a function of CR factor, one cooling duct

All the charts shown are for the fluid stream and the finite difference grid defined in Table 5-2.

The correlation of the CR factor, the coupled numerical methods, the geometry of the body, and the parameters defining the flow such as velocity and temperature of the fluid stream in obtaining the valid solution was not explored in this study but is something that certainly should be done.

The final results shown in this and the preceding sections are not proven through tests or any theoretical work. Thus, some experiments would probably need to be run to confirm them. Regardless of that fact, and based on the evaluations performed on each of the numerical methods through the tests shown in the previous chapters, it can be concluded that the results are reasonable.

6 Conclusions and Future Work

The study concludes with a summary of inferences drawn from the work and possible directions for future efforts.

6.1 Summary

The observations extracted from this study of a steady conjugate heat transfer process between an aerofoil shaped object with a cooling duct that is immersed in a forced convection laminar boundary layer can be summarized as follows:

- Creation of a computer program that performs conjugate heat transfer analysis on an airfoil shaped object in a fluid flow provides valuable insight into all aspects of the fluid flow, the convective heat transfer, and the body heat conduction analysis
- The created numeric procedure established the methodology and a direction for analysis of a more complex and generally curved object and emphasized issues that would have to be resolved if such an object is exposed to conjugate heat transfer within a fluid flow different from the one analyzed here.
- There is high interaction between applied numerical methods due to the fact that the result of one of them is used as input or boundary condition for another one. They can be successfully coupled and a final solution will be reasonable providing major coding issues, such as traveling of the results between different grids, are properly addressed. A different grid for each of the numerical methods would be a desirable option, since advantages of each procedure should be preserved.

- The concerns associated with any numeric type analysis have to be addressed for each of the methods separately, before the same problems are addressed in the complete code. One of these concerns would be a mesh to be applied producing a qualitative and valid solution.
- A conjugate heat transfer problem on a body immersed in a fluid flow can be successfully resolved and a valid solution can be obtained through application and coupling of the surface vorticity boundary integral method, the implicit finite difference method, and the boundary element method. The limits of application of a conjugate heat transfer computer code and quality of the solution it produces, are determined by each of the numerical methods used and their interaction. Other parameters such as fluid and body material properties, together with a specific problem characteristics, magnitude and angle of upstream velocity, temperature of the free stream, and so on, could have a huge influence on applicability of the code and its results. Conductivities ratio CR factor was identified as one of the parameters that determines the convergence of the solution and specifies the limits of application.
- In case a conjugate heat transfer problem is resolved utilizing the methodology and code developed in this study, results should be verified through experiments.

6.2 Future Directions

Based on the work completed and problems encountered in this study, several areas of interest for future research can be suggested. Some of them can be considered as an extension to the work on a conjugate heat transfer problem studied here, while the others are topics of a more general nature.

As it has been shown, for an airfoil shaped object immersed in a hot fluid stream the heat transfer mechanism involves two phenomena that were considered:
Convection from the fluid stream to the surface of the body
Conduction inside the body

The two-dimensional heat conduction problem was solved through application of a boundary element method, which has been demonstrated to be very efficient and accurate. The boundary element method requires discretization of the surface rather than the volume. Hence boundary element codes are easy to use with existing mesh generators, mesh can be easily generated, and modified. Since the method allows a few elements located on the surface of the body to be used, and still produces a valuable solution, the computer time is no longer a primary concern in a boundary element analysis. The improvements in this specific case could be made if a different type of boundary elements is used. The type of boundary elements that could be exploited are linear, discontinuous, quadratic, and higher order elements. Any of the types mentioned should produce a more accurate solution than the boundary element method based on constant elements if the same number of elements is used.

The work or the methods used in this study to complete the analysis of the other phenomenon within the conjugate heat transfer problem, the convection from the fluid stream to the surface, could be significantly improved or replaced with alternative methods. As it was discussed in Chapter 3., the implicit Crank-Nicolson finite difference method was used to approximate the boundary layer momentum equation. As we know, the differential momentum equation is nonlinear, however, the finite difference form of the momentum equation is linear. That was accomplished by evaluating the u in the coefficient of $\delta u/\delta x$ at n, where it is known and not at n+1 where it is not known. The same was applied for the v $\delta u/\delta y$ term. Therefore, even if the procedure employed is called implicit it is not completely implicit. The improvements to the boundary layer solver, created in the study, could be made if the linearization of the momentum equation is done by evaluating the u in the coefficient of $\delta u/\delta x$ at n+1 location. Since the u at n+1 location is not known in the current sweep, the value calculated in the previous sweep would be used. That means that the boundary layer solver would employ an iterative procedure until the velocity field within the boundary layer converges to meet certain criterion. The solver would be fully implicit utilizing the iterative procedure. The same would be applied to the v $\delta u/\delta y$ term. The modification would not be so difficult to implement, since only the coefficients of the tridiagonal system of equations resolved

would be changed. The iteration loop, controlling the values of the velocity filed, located somewhere in the code must not be forgotten.

The alternative to the potential flow and the finite difference solvers, as used in the code created in the thesis or modified as explained above, would be the utilization of another numerical method named "Finite Volume Method". A module of a conjugate heat transfer analysis code, based on finite volume formulation, would be a very powerful substitute for both the potential flow and the boundary layer routines employed in the code developed in this study. The problem of coupling between the potential flow and the finite difference solutions, including the issue of having different grids, would be eliminated. The mesh around the body and a preference of the finite volume formulation, as well as its application, would be some of the major issues to be answered in creating such a code. By knowing the features of both, the boundary element and the finite volume methods, coupling between them should not be a problem.

The procedure and the computer code produced in this work could be easily used and upgraded in a study of the heat transfer process on airfoils at different angles of attack, and flat and cylindrical bodies with or without heat source immersed in a fluid flow. Applicability could be further stretched to conjugate heat transfer occurring during the autoclave cure process of manufacturing curved composite panels. If some work is devoted to issues such as application and implementation of turbulent boundary layer and transition region in between laminar and turbulent boundary layers, the domain of the procedure applicability could be substantial.

References

[1] White, F.M. (1991): "Viscous Fluid Flow", McGraw-Hill, Inc.

[2] Schetz, J.A. (1993): "Boundary Layer Analysis", Prentice-Hall, Inc.

[3] Lewis, R.I. (1996): "Turbomachinery Performance Analysis", Arnold in UK/ John Wiley & Sons, Inc. in North America

[4] Incropera, F.P./DeWitt, D.P. (1996): "Introduction to Heat Transfer", John Wiley & Sons, Inc.

[5] Munson, B.R./Young, D.F./Okiishi, T.H. (1998): "Fundamentals of Fluid Mechanics", John Wiley & Sons, Inc.

[6] Brebbia, C.A./Dominguez, J. (1992): "Boundary Elements An Introductory Course", Computational Mechanics Publications, Boston/ McGraw-Hill Book Company New York

[7] Moran, J. (1984): "An introduction to Theoretical and Computational Aerodynamics", John Wiley & Sons, Inc.

[8] Petrovic, Z./Stupar, S. (1992): "Projektovanje Racunarom Metod Konacnih Razlika", Masinski Fakultet Beograd

[9] Oates, G.C. (1985): "Aerothermodynamics of Aircraft Engine Components", American Institute of Aeronautics and Astronautics, Inc.

[10] Batchelor, G.K. (1970): "An Introduction to Fluid Dynamics", Cambridge University Press

[11] Peric, M./ Ferziger, J.H. (1999): "Computational Methods for Fluid Dynamics", Springer-Verlag Berlin Heidelberg

[12] Patankar, S.V. (1980): " Numerical Heat Transfer and Fluid Flow", Hemisphere Publishing Corporation

[13] Smetana, F.O. (1997): " Introductory Aerodynamics and Hydrodynamics of Wings and Bodies: A Software-Based Approach", American Institute of Aeronautics and Astronautics, Inc.

[14] Jilani, G./ Jayaraj, S./Adeel Ahmad, M. (2002): " Conjugate forced convection-conduction heat transfer analysis of a heat generating vertical cylinder", International Journal of Heat and Mass Transfer 45 (2002) 331-341

[15] Luikov, A.V. (1974): "Conjugate convective heat transfer problems", International Journal of Heat and Mass Transfer 17 (1974) 257-265

[16] Payvar, P. (1977): "Convective heat transfer to laminar low over a plate of finite thickness", International Journal of Heat and Mass Transfer 20 (1977) 431-433

[17] Trevino, C./ Espinoza, A./ Mendez, F. (1997): "Asymptotic analysis of the transient conjugate heat transfer process between two forced counterflowing streams", Society for Industrial and Applied Mathematics, Vol. 57, No. 3, pp 577-596

[18] Nakayama, W./ Park, S. (1996): "Conjugate Heat Transfer From a Single Surface-Mounted Block to Forced Convective Air Flow in a Channel", ASME Journal of Heat Transfer, Vol. 118, pp. 301-309

[19] Krishna, K. (1996): "Conjugate Heat Transfer Analysis of a Heat Sink with Power Transistors", Internet, EMRC-INDIA

[20] Short, B.E. (1998-2003): "Experimentally Validated Numerical Model of Conjugate Heat Transfer in a Cast Pin Fin Coldwall for Electronic System Cooling System Cooling", Fluent Asia Pacific Co., Ltd.

[21] Sparrow, E.M./ Chyu, M.K. (1982): "Conjugate forced convection-conduction analysis of heat transfer in a plate fin", ASME Journal of Heat Transfer, 104, pp. 204-206

[22] Wu, J.C. (1961): " On the Finite Difference Solution of Laminar Boundary Layer Problems", Stanford University Press

APPENDICES

Appendix I. The Main Code

```
!
! M. JOJIC  17/JANUARY/03
!           Problem of Internally Cooled Turbine Blade:
!           PROGRAM   ConjugateHT
!
!           *Incompressible Inviscid Flow was analysed using
!            THE VORTEX ELEMENT BOUNDARY INTEGRAL METHOD.
!            Computation is executed within subroutine "PANEL.FOR"
!           *The Boundary Layer Equations describing
!            Two Dimensional Steady Laminar Incompressible Flow
!            were solved through Numerically Exact Solutions
!            FINITE DIFFERENCE METHOD
!            Crank-Nicolson Implicit Technique
!            (Parabolic Differential Equations)
!     Computation is executed within subroutine "VELOCITYBL.FOR"
!     *Treatment of Viscous Effects Within the Inviscid Formulation
!     done through attachment of the WAKE BODY to original airfoil
!            Computation is executed within subroutine "PSEUDOBODY.FOR"
!           *The Boundary Layer ENERGY Equation:  Constant-Property
!            Two Dimensional Steady Laminar Incompressible  Flow
!            was solved through Numerically Exact Solutions
!            FINITE DIFFERENCE METHOD
!            Crank-Nicolson Implicit Technique
!          Computation is executed within subroutine "TEMPERATURE.FOR"
!     *Heat Transfer Process or CONDUCTION ANALYSIS within the blade
!     performed through application of THE BOUNDARY ELEMENT METHOD
!     The code is valid for isotropic materials and uses con. elements
!     Computation is executed within subroutine "BNDELEM.FOR"
!
      Program           ConjugateHT
        parameter       (KK=10000,S=700,VV=200)
        parameter       (PI=3.14159)
        integer   M     !Number of grid  points in the y direction
        integer   N     !Number of grid  points in the x direction
        integer   Mest  !Auxiliary  variable
        integer   int   !Interval for data output
        integer   Numb  !Index of trailing edge grid point-original a/f
        integer   te    !Number of panels on upper surface-original a/f
        integer   MM    !Number of panels around a/f
        integer   teP   !Number of panels on upper surface-pseudobody
        integer   MMP   !Number of panels around pseudobody
        integer   NoIter !Number of iterations PRESS  "do loop"
        integer   NoIterCC !Number of iterations COND_CONV  "do loop"
integer     idtsp !i-grid point before sepr. (1 to Numb) any iter.
integer     idtspm !i-grid point before sepr. (1 to Numb) first iter.
integer     nufl  !i-grid point before sepr. (1 to N) any iter.
integer     nuflm !i-grid point before sepr. (1 to N) first iter.
integer     Iter,ID,JD,fi,di  !Auxiliary  variable
integer     TTf,pf,ppf   !Auxiliary variable
```

```
integer  kory    !Number of dy spaces between grid points used to
                 !calculate flux
integer  ispis   !ispis=1(yes, print velocity profile) ispis=0 (no)
integer  isp2    !isp2=1(yes, print shear stress) isp2=0 (no)
integer  isp3    !isp3=1(yes, print pseudobody coord) isp3=0 (no)
integer  isp4    !isp4=1(yes, print panel flux and h) isp4=0 (no)
integer  pnte    !pnte=1(pseudobody1.for) pnte=0 (pseudobody.for)

real        delt(KK)     !Thickness of the boundary layer
real        deltT(KK)    !Thickness of the temp boundary layer
real        CF(KK)       !Skin friction coefficient any iter.
real        CFm(KK)      !Skin friction coefficient first iter.
real        DSt(KK)      !Displacement thickness ( integral )
real        DStG(S)      !Displacement thickness at grid points
real        Tet(KK)      !Momentum thickness ( integral )
real        ue(KK)       !Velocity
real        ue1(KK)      !Velocity
real        corX(KK)     !Location in coordinate  x
real        u(KK,S)      !U-Velocity within boundary layer
real        v(KK,S)      !V-velocity within boundary layer
real        u1(KK,S)     !U-Velocity within boundary layer first iter
real        v1(KK,S)     !V-velocity within boundary layer first iter
real        Temp(KK,S)   !Temperature within the boundary layer
real        T(KK,S)      !Temperature within the boundary layer
real        Ttr(S)       !Auxiliary array
real(8)     dvi          !Dynamic viscosity
real        a(S),b(S),c(S),d(S)!Coefficients of tri-diagonal system
real        utr(S)
real        Bue(VV)      !Velocities at grid points
real        xch(VV)      !X-coordinate of grid points original a/f
real        ych(VV)      !Y-coordinate of grid points original a/f
real        Xpsb(VV)     !X-coordinate of grid points pseudobody
real        Ypsb(VV)     !Y-coordinate of grid points pseudobody
real        xchord(VV),ueuin(VV),ychord(VV)
real        gamaP(VV)    !Vorticity vector final solution of velocities
real        Bp(VV)       !Slope of an element (angle) original a/f
real        BpP(VV)      !Slope of an element (angle) pseudobody
real        xp(VV)       !X-coordinate of a pivotal point within a/f
real        xpP(VV)      !X-coordinate of a pivotal point pseudobody
real        CpP(VV)      !The pressure coefficient Cp
real        ds(VV)       !Length of a surface vorticity element
real        g(S)         !Auxiliary  array
real        fDstar(S)    !Auxiliary  array
real        fTet(S)      !Auxiliary  array
real        niz(S)       !Auxiliary array
real        alf(S),gamaT(S)!Auxiliary  variable within trid.for
real        alfg(S)
real        Tpnl(S)      !Temperature over a panel
real        FictT(S)     !Auxiliary  array-interim values of panel temp.
real        pointS           !Point of separation
real        Chord        !chord of the airfoil
real        ChordP       !chord of the pseudobody
real        NI           !Laminar kinematic viscosity
real        L            !Length - x direction
real        H            !Height - y direction
real        dy           !Step size - y direction
real        dx           !Step size - x direction
```

```
      real      UNinf       !Free stream velocity - infinity
      real      ro          !Density of the air
      real      Cv          !Specific heat of the air at cnst. volume
      real      cndt        !Thermal conductivity of the fluid (air)
      real      cndtbl      !Thermal conductivity of the blade
      real      Tg          !Temperature outside the boundary layer
      real      Tw          !Temperature of the body on the surface
      real      CDP         !Pressure DRAG Coefficient
      real      CDF         !Friction DRAG Coefficient
      real      CD          !Total DRAG coefficient
      real      Re,Deno
      real      prod,Dpd,bdely,bdelx
      real      concfL(KK)  !Local convection coefficient
      real      concfA(S)   !Average convection coeff of a panel
      real      flux(KK)    !Local heat flux
      real      fluxA(S)    !Total heat flux over a panel
      real      integ1,integ2     !Auxiliary variable
      real      prt,prtsk   !Auxiliary  variable
      real      p,z,PP      !Auxiliary  variable
      real      CFispis     !Auxiliary  variable
      real      kek,keki    !Auxiliary  variable
      real      KS,KT       !Auxiliary  variable
      real      POM,BB,BB1  !Auxiliary  variable
      real      gg,AA,cc    !Auxiliary  variable
      real      TT,FF,xx    !Auxiliary  variable
      real      ErT,ErT1    !Auxiliary  variable
      real      FictCDP     !Auxiliary  variable
      real      ALFinfD
      data      Uo/0.0/     !Boundary  cond.- velocity on the surface
      data      ErrorT/0.1/ !Allowed difference between panel temp
      data      ErrorCDP/0.02/!Allowed difference between press. DRAG
                              !coefficient values calculated in iter
C--Data Statements
      character filein*10,fileout*10
      CHARACTER DATAFN*12,OUTPFN*12,fileout1*10
      DATA IUNIT/1.0/,IOFLE/1.0/,ext/2.0/
      data inp/1.0/,ipr/6.0/

       ID=KK
       JD=S
C     Variables INPUT  subroutine

       CALL variablein(ID,JD,N,M,int,NoIter,NoIterCC,L,H,dx,dy,
      K                      UNinf,Chord,ALFinfD,NI,ro,Cv,cndt,cndtbl,
      K                      Tg,Tw,filein,fileout,fileout1,DATAFN,
      K                      ispis,isp2,isp3,isp4,pnte)

       WRITE(*,'(1X,A,A,A)') 'Reading input data from file: ',
      K                           DATAFN,' ...'

       OPEN(unit=1,file=DATAFN,form='formatted',status='unknown')

       CALL INPUTPAN(ID,JD,IUNIT,OUTPFN,xchord,ychord,Numb,MM)

       CLOSE(unit=1)
```

```
      do i=1,MM+1
            xch(i)=xchord(i)*chord
            ych(i)=ychord(i)*chord
      end do

      N=((Chord-xch(1))/dx)+1
      te=Numb-1

      OPEN(unit=ext,file=fileout1,form='formatted',status='unknown')

C     Variables OUTPUT subroutine

      CALL variableout(ID,JD,M,N,int,NoIter,NoIterCC,L,H,dx,dy,
     K                         UNinf,Chord,ALFinfD,NI,ro,Cv,cndt,cndtbl,
     K                         Tg,Tw,filein,fileout,fileout1,DATAFN)

C     The PRESSURE iterations loop

      FictCDP=10.0  !Number 10 could be anything, it is to start a loop

      Iter=1
      DO WHILE (Iter.LE.NoIter)

      if (Iter.EQ.1)then
            do i=1,MM+1
            Xpsb(i)=xch(i)
            Ypsb(i)=ych(i)
            end do
            teP=te
            MMP=MM
            chordP=chord
      end if

      write(ext,1110)
1110  format('',79('=')//)
      write (ext,1160)Iter
1160  format(5x,'This is PRESSURE loop / iteration  number Iter =',I3)
      write (ext,1161)teP
1161  format(5x,'Numb of elm on upper surf of pseudobody    teP =',I3)
      write (ext,1162)MMP
1162  format(5x,'Numb of pivot points around pseudobody     MMP =',I3)
      write (ext,1163)chordP
1163  format(5x,'Chord of the pseudobody                chordP
=',F10.4)

C     Flow past an aerofoil with unknown circulation G
C     velocities calculated at panels pivotal points

      CALL PANEL(OUTPFN,UNinf,ALFinfD,Xpsb,Ypsb,teP,MMP,chordP,
     K           gamaP,CpP,BpP,xpP,te,CDP)

      if(Iter.EQ.1)then
            do i=1,MM+1
            Bp(i)=BpP(i)
            xp(i)=xpP(i)
            end do
      end if
```

```
C  calculation of velocities at grid points before BL calculations

       Bue(1)=0.00001
       if (Iter.EQ.1)then
       Bue(Numb)=gamaP(Numb-1)
       else
       Bue(Numb)=gamaP(Numb)
       end if
       do i=2,Numb-1
       xx=(xch(i)-xp(i-1))/(xp(i)-xp(i-1))
       Bue(i)=xx*(gamaP(i)-gamaP(i-1))+gamaP(i-1)
       end do

C  interpolation

       Call INTERPOL(ue,ID,JD,dx,UNinf,Chord,niz,Numb,xch,ych,Bue)
       Call POCETNI(u,v,M,N,ue,Uo,KK,S)          !Boundary conditions

C      The Boundary Layer Equations calculations
C       - velocity field within boundary layer
C       - dependent variables and boundary layer characteristics

       if(pnte.eq.0)then
       CALL velocityBLsh(VV,ID,JD,N,M,nufl,Numb,int,idtsp,delt,CF,
      K
       corX,DSt,DStG,Tet,UNinf,ue,u,v,xch,ych,niz,
      K                        ispis,dx,dy,NI,pointS)
       else
       CALL velocityBL2(VV,ID,JD,N,M,nufl,Numb,int,idtsp,delt,CF,
      K
       corX,DSt,DStG,Tet,UNinf,ue,u,v,xch,ych,niz,
      K                        ispis,dx,dy,NI,pointS)
       end if

       if(Iter.EQ.1)then
       nuflm=nufl
       idtspm=idtsp
             do i=1,N
             CFm(i)=CF(i)
             ue1(i)=ue(i)
                   do j=1,M
                   u1(i,j)=u(i,j)
                   v1(i,j)=v(i,j)
                   end do
             end do
       end if

C new coordinates of grid points xch,ych  and slopes at grid points

       if(pnte.eq.0)then
       CALL PSEUDOBODY(Numb,idtsp,chord,Bp,DStG,xch,ych,Iter,
      K                isp3,teP,MMP,chordP,Xpsb,Ypsb,Numb2)
       else
       CALL PSEUDOBODY1(Numb,idtsp,chord,Bp,DStG,xch,ych,Iter,
      K                isp3,teP,MMP,chordP,Xpsb,Ypsb,Numb2)
       end if
```

```
      CALL friction(ID,JD,N,dx,nuflm,idtspm,te,UNinf,CFm,ro,
     K                    isp2,xch,ych,Numb,MM,niz,ue1,CDF)

      CDP=CDP/Chord
      CD=CDP+CDF

      write (ext,1170)CDP
1170  format(5x,'Pressure DRAG Coefficient                  CDP
=',F9.5/)
      write (ext,1180)CD
1180  format(5x,'Total DRAG Coefficient                     CD
=',F9.5/)

      ErT1=ABS((CDP-FictCDP)/CDP)
      if (ErT1.GT.ErrorCDP)then
      go to 15
      else
      go to 16
      end if

15    FictCDP=CDP

C this is the end of PRESSURE  "DO  WHILE" loop ....

      Iter=Iter+1

      END DO
C_____________________________________________________________________
16    do i=1,MM
      FictT(i)=10        !Number 10 could be anything, it is to start a
loop
      end do

      Iter=1
      DO WHILE (Iter.LE.NoIterCC)

      write(ext,1250)
1250  format('',79('=')//)
      write (ext,1260)Iter
1260  format(5x,'This is CD_CV do loop iteration #     Iter = : ',I8/)

      if (Iter.EQ.1)then
      CALL INITIALTEMP(ID,JD,M,N,Tg,Tw,Temp)
      else
      CAll SURFTEMP(ID,JD,M,N,te,Tg,Tw,niz,Tpnl,Temp)
      end if

      CALL TEMPERATURE(ID,JD,M,N,u1,v1,NI,ro,cndt,Cv,dx,dy,Tg,Tw,
     K                         ispis,Temp,deltT,int)

      CALL HEATFLUX(ID,JD,M,N,cndt,dx,dy,Tg,Tw,nuflm,idtspm,int,te,
     K                    isp4,Temp,Numb,MM,niz,concfA,fluxA)

      open(unit=inp,file=filein,form='formatted',status='unknown')
      open(unit=ipr,file=fileout,form='formatted',status='unknown')
```

```
      CALL BNDELEMM(ID,JD,MM,xch,ych,fluxA,cndtbl,Tpnl)

      close (unit=inp)
      close (unit=ipr)
      do i=1,MM
      ErT=ABS(Tpnl(i)-FictT(i))
            if (ErT.GT.ErrorT)then
            go to 11
            end if
      end do
      go to 12

11    do i=1,MM
      FictT(i)=Tpnl(i)
      end do

      Iter=Iter+1
C this is the end of CONDUCTION_CONVECTION coupling "DO WHILE" loop ..

      END DO
C     close INPUT and OUTPUT files

12    CLOSE(unit=ext)
      stop ' ConjugateHT - Kraj'
      end program

      include 'pocetni.for'
      include 'pisia.for'
      include 'interpol.for'
      include 'trid.for'
      include 'inputpan.for'
      include 'panel.for'
      include 'matinv.for'
      include 'determinant.for'
      include 'pisiaf.for'
      include 'pseudobody.for'
      include 'pseudobody1.for'
      include 'psebodyout.for'
      include 'initialtemp.for'
      include 'temperature.for'
      include 'tempout.for'
      include 'heatflux.for'
      include 'bndelemm.for'
      include 'beinpumpc1.for'
      include 'ghmampc.for'
      include 'slnpd.for'
      include 'intempc.for'
      include 'outpmpc.for'
      include 'extinpc.for'
      include 'locinpc.for'
      include 'surftemp.for'
      include 'velocityBLsh.for'
      include 'velocityBL2.for'
      include 'variablein.for'
      include 'variableout.for'
      include 'friction.for'
```

Appendix II. The Vortex Method Subroutine

```
!
!     M. JOJIC  28/Jun/01  Program AerofoilG
!     Two Dimensional Body Immersed in a Uniform Stream W
!     Flow past an aerofoil with unknown circulation G
!     Bound vorticity G strongly controlled by flow at trailing edge
!     Kutta-Joukowski trailing edge condition imposed
!     Numerically exact solutions
!     Flow modeled by clothing the body surface with a vorticity sheet
!     Martensen's boundary integral equation with modified
!     coupling coefficients K(sm,sn)
!
      SUBROUTINE PANEL(OUTPFN,VELinf,ALFinfD,x,y,te,m,chord,
     K           gama,Cp,Bp,xp,te1,CDP)

       PARAMETER  (V=100,S=100)
       PARAMETER  (PI = 3.14159 )
        integer   m     !Number of pivotal points around aerofoil
        integer   i,j,k,PP    !Auxiliary numbers
        integer   te    !Number of elements on pseudobody upper surface
        integer   te1   !Number of elements on aerofoil upper surface
        integer   int   !Step of results output
        real      coup(V,S)   !Coupling coefficients matrix
        real      invcoup(V,S)!Inverse of Coupling coefficients matrix
        real      x(V)       !X-coordinate of a point that defines slice
        real      y(V)       !Y-coordinate of a point that defines slice
        real      xp(V)      !X-coordinate of a pivotal point
        real      yp(V)      !Y-coordinate of a pivotal point
        real      ds(V)       !Length of a surface vorticity element
        real      Bp(V)       !Slope of an element (angle)
        real      thet(V)     !Slope of an element (angle) used for CDP
        real      sinBp(V)    !Sinus of the slope of an element
        real      cosBp(V)    !Cosinus of the slope of an element
        real      RHS1(V)     !Right hand side vector of a matrix
        real      RHS2(V)     !Right hand side vector of a matrix
        real      gama1(V)    !Vorticity vector first equation
        real      gama2(V)    !Vorticity vector (second Eqn for G=1)
        real      gama(V)     !Vorticity vector final solution
        real      sumcol(V)   !Length of a surface vorticity element
        real      Cp(V)       !The pressure coefficient Cp
        real      cons,sum     !Auxiliary  variable
        real      UU,VV,R      !Auxiliary  variable
        real      VELinf       !Free stream velocity - infinity
        real      ALFinf       !Free stream velocity angle with x-
        real      ALFinfD      !Free stream velocity angle with x-
        real      Uinf         !Free stream velocity x-axis  component
        real      Vinf         !Free stream velocity y-axis
        real      abscos       !Auxiliary  variable
        real      t            !Auxiliary  variable
        real      G            !Prescribed bound vorticity
        real      chord        !Aerofoil chord
        real            CDP          !Pressure DRAG Coefficient
```

```
        CHARACTER DATAFN*12,OUTPFN*12
        CHARACTER(8)DATUM
        CHARACTER(10)VREME
        CHARACTER(5)ZONA

C--Data Statements

        DATA IUNIT/1.0/,IOFLE/1.0/
        DATA ex/0.000001/ !Auxiliary number a window in calculations

        CALL DATE_AND_TIME(DATE=DATUM,TIME=VREME,ZONE=ZONA)

      ID=V
      JD=S
C--Prompt for input data file name

C     print '(1x,A,$)','Step of results output (velc s & Cp s) int= '
C     read (*,*) int

      int=1

      OPEN(unit=1,file=VREME,form='formatted',status='unknown')

      WRITE(1,'(5X,A,A,A)')'This is an output file:   ',VREME
      write(1,1001)
1001  format(5x,'Subroutine PANEL (Program AerofoilG)- RESULTS -- '/)
      write(1,1081)
1081  format(5x,'This file is named after time it is executed at')
      write(1,1082)
1082  format(5x,'in order to have results of each iteration written
     K in a separate file. ')
      WRITE(1,'(5X,A,A,A)')'Program execution date: ',DATUM
      WRITE(1,'(5X,A,A,A)')'Program execution time: ',VREME
      write(1,'(1x)')

      write(1,1002)
1002  format(5x,'2-D body immersed in a uniform stream  VELinf ')
      write(1,1003)
1003  format(5x,'Flow past an aerofoil with unknown circulation G ')
      write(1,1080)
1080  format(5x,'Bound vorticity G strongly controlled by flow at t/e
')
      write(1,1090)
1090  format(5x,'Kutta-Joukowski trailing edge condition imposed ')
      write(1,1004)
1004  format(5x,'Numerically exact solutions')
      write(1,1005)
1005  format(5x,'Flow modeled by clothing the body surface with a')
      write(1,1050)
1050  format(5x,'vorticity sheet')
      write(1,1060)
1060  format(5x,'Martensen s boundary integral equation with modified')
      write(1,1070)
1070  format(5x,'coupling coefficients K(sm,sn)'//)
```

```
! step and angle in radians

      ALFinf=(PI/180)*ALFinfD

! calculation of pivotal points

      do i=1,m

      xp(i)=(x(i)+x(i+1))/2
      yp(i)=(y(i)+y(i+1))/2

      ds(i)=((x(i+1)-x(i))**2+(y(i+1)-y(i))**2)**0.5

      sinBp(i)=(y(i+1)-y(i))/ds(i)
      cosBp(i)=(x(i+1)-x(i))/ds(i)

      t=atan((sinBp(i))/(cosBp(i)))

      abscos=abs(cosBp(i))

      if (abscos.gt.ex)then
      Bp(i)=t
      else
      Bp(i)=(PI/2)*((sinBp(i))/(abs(sinBp(i))))
      end if

      if (cosBp(i).lt.(-ex))then
      Bp(i)=t-PI
      end if

      end do

! calculation of self inducing (i=j) coupling coefficients

      coup(1,1)=-0.5-((Bp(2)-Bp(m)-2*PI)/(8*PI))
      coup(m,m)=-0.5-((Bp(1)-Bp(m-1)-2*PI)/(8*PI))

      do i=2,m-1

      coup(i,i)=-0.5-((Bp(i+1)-Bp(i-1))/(8*PI))

      end do

! calculation of coupling coefficients  for  i<>j

      cons=1.0/(2*PI)

      do 10 i=1,m       !rows   up to  m  ( i means m in equations )
            do 10 j=1,m !columns  up to  m  ( j means n in equations )

            if (j.ne.i)then

      R=(xp(i)-xp(j))**2+(yp(i)-yp(j))**2

      UU=((yp(i)-yp(j))/R)*cons
      VV=-((xp(i)-xp(j))/R)*cons
```

```
      coup(i,j)=((UU*cosBp(i))+(VV*sinBp(i)))*ds(j)
            else
            go to 10
            end if
10    end do

! back-diagonal correction

      PP=0
      do j=1,m
      sum=0
            do i=1,m
            if (i.ne.(m-j+1))then
            sum=sum+(coup(i,j)*ds(i))
            end if
            end do
      k=m-PP
      PP=PP+1
      coup(k,j)=-(sum/ds(k))

      end do

! test of the back-diagonal correction  loop

!     do j=1,m
!     sumcol(j)=0
!           do i=1,m
!           sumcol(j)=sumcol(j)+coup(i,j)
!           end do
!     write (1,1080)j,sumcol(j)
!     end do
!1080 format(5x,'Sum of the column',I3,' is sumcol =',F7.3)
      write(1,'(1x)')

! coefficients K(i,j) changed to    K(i,j)+ds(j)

      do i=1,m
            do j=1,m
            coup(i,j)=coup(i,j)+ds(j)
            end do
      end do

! circulation  G

      Uinf=VELinf*cos(ALFinf)
      Vinf=VELinf*sin(ALFinf)

      do i=1,m
      RHS1(i)=-(Uinf*cosBp(i))-(Vinf*sinBp(i))
      RHS2(i)=1
      end do

! calculation of inverse matrix - invcoup(i,j)

      CALL MATINV(ID,JD,m,coup,invcoup)
```

```
! multiplication of inverse matrix - invcoup(i,j)  with rhs(j) vector

      gama1=matmul(invcoup,RHS1)
      gama2=matmul(invcoup,RHS2)

! calculation of the solution for the aerofoil bound vorticity G

      G=-(gama1(te)+gama1(te+1))/(gama2(te)+gama2(te+1))

! calculation of the final solution of velocities on each element and
! calculation of the pressure coefficient Cp on the surface of the
aerofoil

      do i=1,m
      gama(i)=gama1(i)+(G*gama2(i))
      Cp(i)=1-((gama(i)/VELinf)**2)
      end do

      CDP=0.0
      do i=1,te1
      thet(i)=ABS(ASIN(sinBp(i)))
            if(sinBp(i).GT.0)then
            CDP=CDP+(Cp(i)*(cos((PI/2)-thet(i)))*ds(i))
            else
            CDP=CDP+(-Cp(i)*(cos((PI/2)-thet(i)))*ds(i))
            end if
      end do

      do i=m-te1+1,m
      thet(i)=ABS(ASIN(sinBp(i)))
            if(sinBp(i).GT.0)then
            CDP=CDP+(Cp(i)*(cos((PI/2)-thet(i)))*ds(i))
            else
            CDP=CDP+(-Cp(i)*(cos((PI/2)-thet(i)))*ds(i))
            end if
      end do

! write all the results and what is needed to check the results

      Call PISIAF(ID,JD,int,m,te,chord,G,
     K          ALFinfD,VElinf,Uinf,Vinf,
     K          Bp,gama,Cp,xp,CDP)        !Write the  results

      close(unit=1)
      return
      end
```

Appendix III. The Boundary Layer Subroutine

```
!
! M. JOJIC  31/May/01  Program  "velocityBL.for"
!           Finite Difference Method
!           Crank-Nicolson Implicit Technique
!           This subroutine calculates:
!           - velocity field within boundary layer
!           - dependent variables and boundary layer characteristics
!             calculated from velocity distribution within BL
!           - other  variables needed for functioning of this sub
!             within the main program
!

      SUBROUTINE velocityBL2(VV,ID,JD,N,M,nufl,Numb,int,idtsp,delt,CF,
     K                       corX,DSt,DStG,Tet,UNinf,ue,u,v,xch,ych,niz,
     K                       ispis,dx,dy,NI,pointS)

        integer   N     !Number of grid  points in the x direction
        integer   M     !Number of grid  points in the y direction
        integer   ID,JD,i,j,VV !Auxiliary  variable
        integer   Mest  !Auxiliary  variable
        integer   nufl  !i-grid point before sepr. (1 to N)
        integer   Numb  !Index of trailing edge grid point-original a/f
        integer   int   !Interval for data output
        integer   idtsp !i-grid point before sepr. (1 to Numb)
        integer   du1,du2 !i-grid points (auxiliary  variables)
        integer   ispis   !ispis=1(yes, print velocity profile) ispis=0

        real      delt(ID)     !Thickness of the boundary layer
        real      CF(ID)       !Skin friction coefficient
        real      corX(ID)     !Location in coordinate  x
        real      DSt(ID)      !Displacement thickness BL calcs
        real      DStG(JD)     !Displacement thickness at grid points
        real      Tet(ID)      !Momentum thickness (integral)
        real      ue(ID)       !Velocity
        real      u(ID,JD)     !Velocity within the boundary layer
        real      v(ID,JD)     !V-velocity within the boundary layer
        real      a(JD),b(JD)  !Coefficients of tri-diagonal system
        real      c(JD),d(JD)  !Coefficients of tri-diagonal system
        real      utr(JD)      !Resultant vector from TRID.FOR
        real      xch(VV)      !X-coordinate of grid points original a/f
        real      ych(VV)      !Y-coordinate of grid points original a/f
        real      g(JD)        !Auxiliary  array
        real      fDstar(JD)   !Auxiliary  array
        real      fTet(JD)     !Auxiliary  array
        real      niz(JD)      !Auxiliary array
        real      dy           !Step size - y direction
        real      dx           !Step size - x direction
        real      NI           !Laminar kinematic viscosity
        real      pointS       !Point of separation x-coordinate
        real      UNinf        !Free stream velocity - infinity
        real      AA,BB,BB1    !Auxiliary  variable
        real      KS,KT        !Auxiliary  variable
```

```
        real      p,POM       !Auxiliary  variable
        real      prt,prtsk   !Auxiliary  variable
        real      CFispis     !Auxiliary  variable
        real      kek,keki    !Auxiliary  variable
        real      gg,cc       !Auxiliary  variable
        real      TT,FF,xx    !Auxiliary  variable
        real      zz,zzz      !Auxiliary  variable
        real      Naux1,Naux2 !Auxiliary  variable

C--Data Statements
        Data Uo/0.0/ !Boundary  cond.- velocity on the  surface
        data IUNIT/1.0/,IOFLE/1.0/,ext/2.0/
        data inp/1.0/,ipr/6.0/

      AA=1/(4*dy)
      BB=NI/(2*(dy**2))
      BB1=(dy/dx)*0.5

      delt(1)=0
      CF(1)=0
      corX(1)=0
      DSt(1)=0
      Tet(1)=0
      CFispis=1
      corX(1)=xch(1)
C     idtsp=Numb
C     nufl=N-1
C     Auxiliary numbers

      Naux1=3
      Naux2=N-Naux1

      DO  I=2,N   !Make N steps in the x direction

      g(1)=Uo/ue(i)
      KS=0
      KT=0
      p=0.99*ue(i)
      POM=NI/(dy*(ue(i)**2))
C     POM=NI/(dy*(UNinf**2))
      prtsk=ue(i-1)*(ue(i)-ue(i-1))

      b(1)=1
      c(1)=0
      d(1)=u(i,1)
      a(M)=0
      b(M)=1
      d(M)=ue(i)
            do j=2,M-1
            prt=prtsk/dx
            kek=v(i-1,j)*AA*(u(i-1,j+1)-u(i-1,j-1))
            cc=BB
            keki=cc*(u(i-1,j+1)-2*u(i-1,j)+u(i-1,j-1))
      d(j)=prt-kek+keki+(u(i-1,j)**2)/dx
      a(j)=-(v(i-1,j)*AA+BB)
      b(j)=(u(i-1,j)/dx)+(2*BB)
      c(j)=(v(i-1,j)*AA-BB)
```

```
            end do

      CALL TRID(a,b,c,d,utr,M)

            do j=1,M
            u(i,j)=utr(j)
            end do

            do j=2,M-1
            gg=(u(i,j)-u(i-1,j))+(u(i,j-1)-u(i-1,j-1))
            v(i,j)=v(i,j-1)-BB1*gg
            end do

            do j=2,M-1
            g(j)=u(i,j)/ue(i)

            fDstar(j)=1-g(j)
            fDstar(j-1)=1-g(j-1)

            fTet(j)=g(j)*fDstar(j)
            fTet(j-1)=g(j-1)*fDstar(j-1)

            KS=KS+(fDstar(j)+fDstar(j-1))*0.5*dy
            KT=KT+(fTet(j)+fTet(j-1))*0.5*dy

            if(u(i,j).GT.p)then
            Mest=j
            go to 120
            end if

            end do

120   delt(i)=(Mest-1)*dy
      CF(i)=POM*(u(i,3)-u(i,1))
C     CF(i)=POM*(4*u(i,2)-u(i,3))
      corX(i)=xch(1)+((i-1)*dx)
      DSt(i)=KS
      Tet(i)=KT


      if(CFispis.LT.2.and.i.GT.Naux2+1)then

            nufl=i-1
            pointS=corX(i-1)
            CFispis=CFispis+1
      else

C     if(CF(i).GT.CF(i-1).or.CF(i).LT.0.00001)then
C     if(CF(i).LT.0.00001)then
            if(CF(i).GT.(1.2*CF(i-1)).or.CF(i).LE.0.0001)then

            if (CFispis.LT.2.and.corX(i).GT.0.5)then
            nufl=i-1

      write (ext,2000)corX(i)
2000  format(5x,'Separation point at length
corX=:',F12.6)
```

```
      write (ext,2010)nufl
2010  format(5x,'BL calculation i-grid point before corX  nufl=:',I6/)

            CFispis=CFispis+1
            pointS=corX(i)
            du1=nufl-1
            du2=nufl+3
            end if
            end if

      end if

C this is the end of DO I=2,N   loop  ...along  x-direction

      END DO

      do i=1,Numb
            if (xch(i).GT.pointS)then
            idtsp=i-1
      write (ext,2020)pointS
2020  format(5x,'Separation point at length          pointS =:',F12.6)
      write (ext,2030)xch(i-1)
2030  format(5x,'x-coord of grid point before pointS  xch(i) =:',F12.6)
      write (ext,2040)ych(i-1)
2040  format(5x,'y-coord of grid point before pointS  ych(i) =:',F12.6)
      write (ext,2050)idtsp
2050  format(5x,'Panel s  i-grid point before pointS   idtsp =:',I3/)
            go to 150
            end if
      end do

150   continue

C     print '(1x,A,$)','#2-CP. BL calculations-DONE. Key in a # '
C     read (*,*)zz

      Call PISIA(corX,delt,DSt,CF,Tet,N,M,ue,int,u,ID,JD,KK,S) !Write
the reslt

C     print '(1x,A,$)','#3-CP. BL results written-DONE. Key in a # '
C     read (*,*)zzz

C-----write velocity distribution around pointS within boundary layer -

      if(ispis.eq.1.0)then

C     DO i=du1,du2

      DO i=1,N

      if(mod(i-1,int).eq.0)then

      write (ext,1270)
1270  format(5x,'_____________________________________________________ ')

      write (ext,1280)
1280  format(5x,'Velocity distribution within BL  from j=1 to delt'/)
```

```
      write (ext,1290)i,corX(i)
1290  format(5x,'i = ',I5,'  - grid point number (along X axis)'
     K5x,'corX(i) = ',E14.5,' - coordinate along X axis'//)

      write (ext,1300)
1300  format(4x,'j-grid',3x,'Velc.  U(i,j)',3x,'Velc. V(i,j)'/)

      Mest=(delt(i)/dy)+1

      do j=1,Mest
      write (ext,1310)j,u(i,j),v(i,j)
1310  format(3x,I5,2E14.5)
      end do
      write(ext,'(1x)')
      write(ext,'(1x)')
      end if
      END DO

      end if
C------------writing  the velocity distribution  done -----------------
C transfer of data from long DSt to short array DStG (panel end points)

      DStG(1)=DSt(1)
      DStG(Numb)=DSt(N)

C     write(ext,2060)DStG(1),DStG(Numb)
C2060 format(5x,' DStG(1)=',F10.5,'  and      DStG(Numb)=:',F10.5///)
C     write(ext,2070)
C2070 format(6x,'i',6x,'niz(i)',9x,'FF',9x,'xch',9x,'DSt',9x,'DStG'/)

      TT=1
      do i=2,Numb-1
      FF=TT+niz(i-1)
      DStG(i)=DSt(FF)

C     write (ext,2080)i,niz(i-1),FF,xch(i),DSt(FF),DStG(i)
C2080 format(1x,I6,5F12.5)

      TT=FF
      end do

      write(ext,'(1x)')
      return
      end
```

Appendix IV. The Boundary Element Subroutine

```
C-------------------------------------------------------------+

      SUBROUTINE BNDELEMM(ID,JD,MM,xch,ych,fluxA,cndtbl,Tpnl)

C           PROGRAM 23
C
C    This program solves 2D potential problems with
C    multiboundary domains using "CONSTANT BOUNDARY ELEMENTS" or CONBE
C

      PARAMETER   (VV=200)
      PARAMETER  (PI=3.14159)

      integer ID,JD   !Auxiliary variable
      integer MM      !Number of panels around a/f
      integer Bbnd    !Number of different boundaries
      integer BPIS(15)!Number of elements on each of inside boundaries
      integer Bintp   !Number of internal points where funct calculated
      integer Nbem    !Total  number of elements on ALL surfaces
      integer NbemINS !Total  number of elements on ALL INSIDE surfaces
      integer NC(15)  !Array containing the last elmnts of boundaries
      integer dum1,dum2,dum3,NX  !Auxiliary variable
      real D
      real xbnd(101),ybnd(101) !Coord. of points of the elmnts
      real xib(30),yib(30)     !Coord of points of BE on inside surf.
      real cx(20),cy(20)       !Coord. of internal points of the elmnts
      real kode(101)      !Variable defining the bound. cond. on elmnts
      real xm(101),ym(101) !Nodal coordinates at the center of a panel
      real fi(101),dfi(101) !Potential and potential derivative in subs
      real pot(20)          !Potential at internal points
      real flux1(20),flux2(20)!Flux X and Y at internal points
      real Gmtx(100,100)      !Matrix
      real Hmtx(100,100)      !Matrix
      real Tpnl(JD)           !Temperature over a panel
      real fluxA(JD)          !Total heat flux over a panel
      real xch(VV)            !X-coordinate of grid points original a/f
      real ych(VV)            !Y-coordinate of grid points original a/f
      real kodeib(30),fiib(30)!Boundary Conditions on inside boundary
      real bndcon(101)  !Boundary Conditions on both boundaries
      real cndtbl       !Thermal conductivity of the blade material

      character filein*10,fileout*10
C
C     assign numbers for input and output files
C
      data inp/1.0/,ipr/6.0/,ext/2.0/

C     set maximum dimension of the system of equations (NX)
C     this number must be equal or smaller than the dimen. of XM,etc)

      NX=100
```

```
C     read data
C
C     CALL INPUMPC(cx,cy,x,y,kode,fi,N,L,M,NC)

      CALL BEINPUMPC1(cx,cy,xib,yib,kodeib,fiib,
     K               Bbnd,BPIS,Bintp,NbemINS)

      Nbem=MM+NbemINS
      NC(1)=MM

      do i=2,Bbnd
      NC(i)=NC(i-1)+BPIS(i-1)
      end do

      if(Bbnd)40,40,30
30    write(ipr,999)Bbnd,(NC(K),K=1,Bbnd)
999   format(/2x,'Number of different boundaries =',I3/2x,'Last
     K node of each boundary =',5(I3,','))

C     dum1=MM+1
40    dum2=0
      do i=1,MM
      xbnd(i)=xch(MM+1-dum2)
      ybnd(i)=ych(MM+1-dum2)
      dum2=dum2+1
      end do

      do i=1,NbemINS
      xbnd(MM+i)=xib(i)
      ybnd(MM+i)=yib(i)
      end do

      write(ipr,170)
170   format(//2x,'Coordinates of extreme points of the Boundary
     K Elements on both boundaries ='//1x,'Point',7x,'X',15x,'Y')
      do i=1,Nbem
      write(ipr,180)i,xbnd(i),ybnd(i)
180   format(2x,I3,2(2x,E14.5))
      end do

      do i=1,Nbem
            if (i.le.MM)then
            kode(i)=1
            bndcon(i)=(-1/cndtbl)*fluxA(i)
            else
            kode(i)=kodeib(i-MM)
            bndcon(i)=fiib(i-MM)
            end if
      end do

      write(ipr,190)
190   format(//2x,'Boundary Conditions on both boundaries '//
     K2x,'Node',6x,'Code',7x,'Prescribed Value')
      do i=1,Nbem
      write(ipr,200)i,kode(i),bndcon(i)
200   format(2x,I3,7x,F5.2,6x,E14.5)
```

```
      end do

C     compute H and G matrices and form system (A X = F )

      CALL GHMAMPC(Nbem,NX,Bbnd,NC,xbnd,ybnd,xm,ym,Gmtx,Hmtx,
     K             bndcon,dfi,kode)

C     solve system of equations

      CALL SLNPD(Gmtx,dfi,D,Nbem,NX)

C     compute potential and fluxes at internal points

      CALL INTEMPC(Nbem,Bintp,Bbnd,NC,bndcon,dfi,kode,cx,cy,
     K             xbnd,ybnd,pot,flux1,flux2)

      write(ext,210)
210   format('',79('*')//1x,'Results from  BNDELEM.FOR '//
     K2x,'Boundary nodes on blade ext surf and derivt. within a
blade'//
     K8x,'X',15x,'Y',11x,'Temperature',3x,'Temper. Derivative'/)

      dum3=0
      do i=1,MM
      Tpnl(MM-dum3)=bndcon(i)
      write(ext,220)xm(i),ym(i),Tpnl(MM-dum3),dfi(i)
220   format(4(2x,E14.5))
      dum3=dum3+1
      end do

      write(ext,230)
230   format('',79('*')//)

C     print results at boundary nodes and internal points

      CALL OUTPMPC(xm,ym,bndcon,dfi,cx,cy,pot,flux1,flux2,Nbem,Bintp)

      return
      end
```