EULER EQUATION COMPUTATIONS FOR THE FLOW OVER A HOVERING HELICOPTER ROTOR

Vol. 1

THOMAS WESLEY ROBERTS
B.S., Cornell University (1981)
M.Eng., Cornell University (1981)
S.M., Massachusetts Institute of Technology (1983)

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR
THE DEGREE OF
DOCTOR OF PHILOSOPHY
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
November 1986

Signature of Author _____
Department of Aeronautics and Astronautics
November 13, 1986

Certified by _____
Professor Earll M. Murman
Thesis Supervisor, Department of Aeronautics and Astronautics

Certified by _____
Professor Rene H. Miller
Department of Aeronautics and Astronautics

Certified by _____
Professor Mårten T. Landahl
Department of Aeronautics and Astronautics

Accepted by _____
Professor Harold Y. Wachman
Chairman, Department Graduate Committee

EULER EQUATION COMPUTATIONS FOR THE FLOW
OVER A HOVERING HELICOPTER ROTOR

by

THOMAS WESLEY ROBERTS

Submitted to the Department of Aeronautics and Astronautics
on November 13, 1986, in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy in
Aeronautics and Astronautics

ABSTRACT

A numerical solution technique has been developed for computing the flow field around an isolated helicopter rotor in hover. The flow is governed by the compressible Euler equations which are integrated using a finite volume approach. The Euler equations are coupled to a free wake model of the rotary wing vortical wake. This wake model is incorporated into the finite volume solver using a prescribed flow, or perturbation, technique which eliminates the numerical diffusion of vorticity due to the artificial viscosity of the scheme. The work has been divided into three major parts. In the first part, comparisons of Euler solutions to experimental data for the flow around isolated wings show good agreement with the surface pressures, but poor agreement with the vortical wake structure. In the second part, the perturbation method is developed, and used to compute the interaction of a streamwise vortex with a semispan wing. The rapid diffusion of the vortex when only the basic Euler solver is used is illustrated, and excellent agreement with experimental section lift coefficients is demonstrated when using the perturbation approach. Finally, the free wake solution technique is described and the coupling of the wake to the Euler solver for an isolated rotor is presented. Comparisons with experimental blade load data for several cases show good agreement, with discrepancies largely attributable to the neglect of viscous effects. The computed wake geometries agree less well with experiment, the primary difference being that too rapid a wake contraction is predicted for all the cases.

Thesis Supervisor: Dr. Earll M. Murman

Title: Professor of Aeronautics and Astronautics

Acknowledgments

At the end of this long road (it seems long), it's become glaringly obvious that I could never have gotten this work done all by myself. There are so many people to thank. First, I express my great appreciation to my thesis committee members for all their advice and support during the course of this project. Most of all, I wish to thank my committee chairman, Prof. Earll Murman, for his constant encouragement, advice, and enthusiasm. He has done a great deal to further not only my technical education, but in teaching me the philosophy of research as well. Special thanks must go to Prof. Rene Miller, who shared his wealth of experience in helicopter aerodynamics as well as his interest and enthusiasm. Also, my appreciation to Prof. Mårten Landahl for his valuable help, interest, and advice in this project. Dr. William Thompkins, who served on my committee for only a short time, nevertheless contributed to this document with his critical comments, suggestions, and questions. My appreciation goes to him as well.

This thesis could not have been completed at all were it not for the work of Bob Bruen, our own CFD Fire Chief, who somehow managed to maintain a cheerful countenance despite usually being greeted with such strange salutations as, "Bob, do you know what's wrong with (ORVILL, the network, the RJE link, all of the above)?" To him, for keeping all the parts together, my eternal thanks.

One thing that I have discovered in grad school is that one learns as much from one's fellow students as from the faculty. Most of all I want to express my appreciation to two special friends, Mark Drela and Mike Giles, now Prof. Mark and Prof. Mike. These two guys did so much to help and support me, and never seemed to get annoyed by my constant unscheduled appearances and cries of "What's all this then?" while they were trying to get work done. To the rest of the gang of The Cluster and its vicinity, but especially John Dannenhoffer, Ken Powell, Steve Allmaras, Bernard Loyd, Rich Shapiro—thanks for all your help, guys.

Last but certainly not least, to Mom and Dad, for their love, understanding, and support all these years.

This research was sponsored by NASA Ames Research Center under NASA Grant NAG-2-275.

Contents

List of Figures

Nomenclature

a	speed of sound, or vortex core size
A, B, C	components of flux Jacobian at boundary
b	wing semispan
c	wing chord
c_v, c_p	specific heats at constant volume, pressure
CFL	Courant-Friedrichs-Lewy number
D	dissipation operator
$D^{(2)}, D^{(4)}$	second and fourth difference components of D
e	internal energy per unit mass
E	total energy per unit mass, or complete elliptic integral of the second kind
E_r	total roenergy per unit mass, Equation (4.1)
F	flux operator; also see Equations (4.18a), and (4.18b)
$\vec{\mathbf{F}}(\mathbf{U})$	flux vector
h	enthalpy per unit mass
H	total enthalpy per unit mass
H_r	total rothalpy per unit mass
$\hat{i}, \hat{j}, \hat{k}$	cartesian unit vectors
k	argument of elliptic integrals
K	complete elliptic integral of the first kind
M	Mach number, or number of rotor wake spirals
$\hat{n}$	unit vector normal to boundary
n	coordinate direction normal to boundary
N	number of trailing vortex filaments
N, T_1, T_2	local cartesian coordinate directions at far field boundary
p	pressure
r	radius, or Riemann invariant
R	rotor blade radius, or gas constant

$\mathbf{R}$	residual of finite volume operator
s	entropy
S	cell face projected area, or similarity transform
$S(\vec{x})$	entropy function, p/ρ^{γ}
$\mathbf{S(U)}$	vector of Coriolis and centrifugal acceleration terms
t	time
T	temperature
$\vec{u}$	cartesian velocity vector
u, v, w	components of cartesian velocity vector
$\mathbf{U}$	state vector of conservation variables
$\overline{\mathbf{U}}$	see Equations (2.11) and (2.12)
$\overline{\overline{\mathbf{U}}}$	see Equation (2.34)
$\tilde{\mathbf{U}}$	state vector of characteristic variables
V	control volume
$\vec{x}$	cartesian position vector
x, y, z	components of cartesian position vector
X, Y, Z	computational coordinates
α	angle of attack, or enthalpy damping coefficient
$\alpha_1, \alpha_2, \alpha_3, \alpha_4$	multistage timestepping coefficients
γ	ratio of specific heats, c_p/c_v
Γ	circulation
δ	central difference operator
Δt	time step size
$\epsilon^{(2)}$, $\epsilon^{(4)}$	second and fourth difference artificial viscosity coefficients
η	radius
θ	Glauert variable, Equation (4.26)
$\theta_{.75}$	collective pitch at $.75R$ of rotor
$\kappa^{(2)}$, $\kappa^{(4)}$	second and fourth difference artificial viscosity constants
λ	eigenvalue of the Jacobian matrix

Λ	diagonal matrix of eigenvalues λ
μ	averaging operator
ρ	density
τ	time scale
ϕ	velocity potential
ψ	azimuth angle
$\vec{\omega}$	vorticity vector
ω_b	bound circulation underrelaxation parameter
ω_w	wake underrelaxation parameter
$\vec{\Omega}$	angular velocity of rotor

subscripts

0	prescribed flow value
a	absolute quantity, measured in inertial frame
cyl	vortex cylinder quantity
ex	value extrapolated from computational domain interior
ff	far field value
i, j, k	computational coordinate indices
n	normal to boundary, or wake vortex index
r, ψ, z	quantities referred to r, ψ, and z directions
$sind$	self-induced
tip	quantity at rotor blade tip
v	vortex quantity
$wake$	wake quantity
X, Y, Z	referred to computational coordinate directions
∞	free stream value

superscripts

$'$	quantity referred to inertial reference frame
(n)	quantity at n^{th} multistage level
n	quantity at n^{th} time or iteration level
new	value after relaxation update
old	value before relaxation update

Chapter 1

Introduction

1.1 Background

The helicopter has proven itself as a useful and practical vehicle since the late 1940's. However, the aerodynamics of these aircraft is considerably more complex and difficult to analyze than that of conventional fixed wing aircraft. If one considers the flight envelope of a helicopter, the range of flow regimes covers most of the fluid dynamic phenomena of interest to any aerodynamicist. In forward flight, the flow around the main rotor blades is unsteady and three dimensional. At the tip of the advancing blade, transonic speed may be reached, resulting in shocks. On the retreating blade, high angles of attack can result in dynamic stall, and over the inboard portion of the blade a region of reversed flow is found. Each blade operates in the vortical wake of the other blades of the rotor—in particular there is a strong interaction between the advancing blade and the tip vortex of the preceding blade once every revolution. One must also consider the interaction of the main rotor wake with the tail rotor, or the interference between the two main rotors in a tandem configuration. The presence of the fuselage complicates the picture even more. All this contributes to making the helicopter an aerodynamicist's nightmare—or dream, depending upon his or her attitude.

One thing the helicopter does that conventional aircraft cannot do is

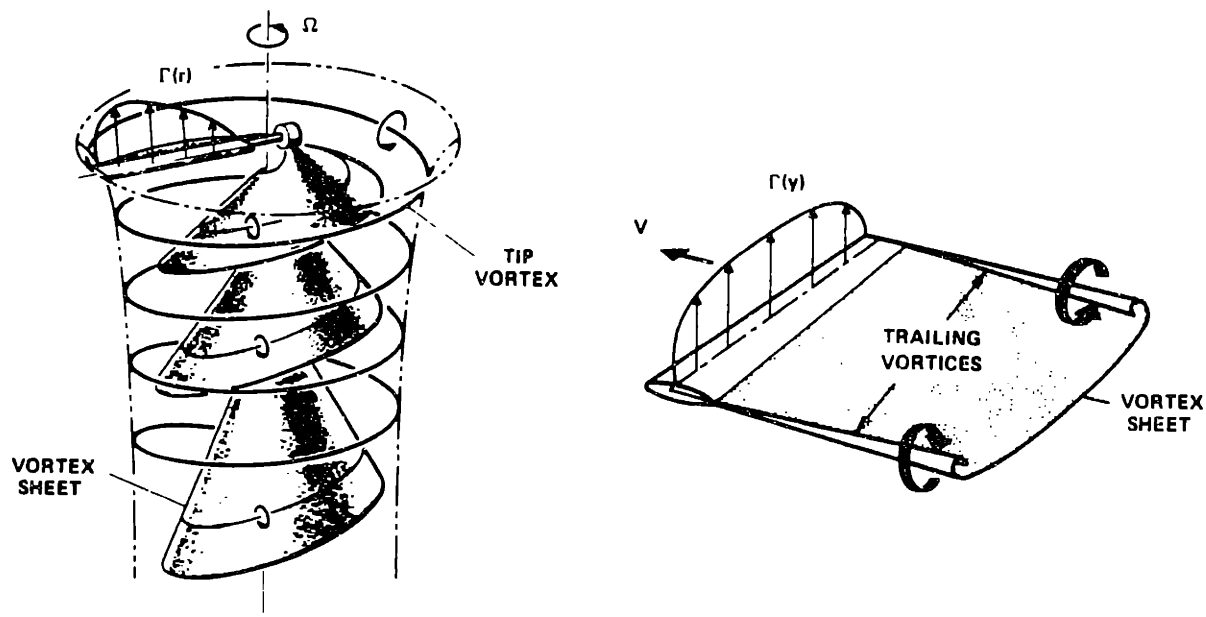

Figure 1.1: Comparison of hovering rotor and fixed wing wakes

hover or fly at very low speeds efficiently. For an isolated rotor in hovering flight the flow picture is simplified since the loads on the rotor blades are steady. By considering the flow field in blade fixed coordinates, one has the analog to steady state flight of a conventional aircraft. Even here, however, are difficulties not found in fixed wing aircraft aerodynamics. The difference between a hovering rotor wake and a classical fixed wing wake can be seen in Figure 1.1, taken from McCroskey [42]. The tip vortex of each rotor blade passes near the following blade, resulting in rapid variations of the spanwise aerodynamic loading near the tip. Also, the wake descends below the rotor, in contrast to the fixed wing wake which is convected downstream of the wing. Because of this, the structure of the rotary wing wake has

a significant impact on the load distribution of the rotor. This is quite different from the classical fixed wing wake for which the effect of the wake roll up is only third order in the angle of attack (Ashley & Landahl [3], pp. 135-136). The implication of this is that even for the seemingly simple case of the hovering rotor, there are important nonlinear fluid dynamic effects that must be accounted for if the aerodynamic loads on the rotor are to be accurately predicted.

The above discussion sheds some light on why the analysis of helicopter aerodynamics lags that of fixed wing aircraft. The complex aerodynamics of rotary wing flow fields is not easily amenable to analytic treatment, and generally the techniques that have developed over the past forty years are based on drastic simplifications of the real flow field. With the advent of high speed computers, the possibility of handling these complicated flow fields numerically is gradually being realized. The development of computational fluid dynamics technology has had a great impact on the design and analysis of conventional aircraft, for which many robust techniques for predicting aerodynamic loads have been developed. Progress in this technology for rotary wing configurations has lagged that for fixed wing aircraft, in good part due to the essentially more complex flow fields associated with the former.

The research reported herein deals with the prediction of the aerodynamic loads on a hovering rotor. The next section of this chapter reviews methods for modeling the vortical wake of a hovering rotor. The computation of the aerodynamic loads and the coupling of the wake model to the solution of the near field flow of the rotor blades is discussed in section 1.3. Finally, the chapter ends with a discussion of the aims of the current research and an outline of the rest of the thesis.

1.2 Hovering Rotor Wake Models

Many approaches to modeling the rotary wing wake exist. Most models are based on the assumption of an inviscid, irrotational, incompressible flow with embedded vorticity. Classical vortex theories of propellers provide a starting point for rotor analyses. These methods model the blades and wake with distributions of bound and trailing vorticity and use the Biot-Savart law to compute the induced velocities. The classical theories assume that the wake may be treated as a rigid, non-contracting helical vortex sheet. This is consistent for high speed propellers for which the induced velocities are small compared to the axial translation speed. For a hovering rotor, this assumption is clearly wrong as the only velocities are the induced velocities. From momentum theory, it is known that the induced velocity over the rotor disk is half the velocity in the fully developed wake. Hence the wake of the rotor must contract. Furthermore, the structure of the hovering rotor wake is found to be much different from the classical picture of a vortex sheet (Figure 1.1). For this reason, propeller vortex theory cannot be directly applied to rotary wings in hover. More realistic models of the wake are required for accurate prediction of the blade loads. There are primarily two approaches to modeling the wake in common use today: prescribed wake and free wake analyses.

Prescribed wake hover prediction methods have been in widespread use in recent years (Landgrebe [38,39], Kocurek & Tangler [35]). These methods use experimental data to derive empirical formulas relating the rotor blade geometric parameters and thrust coefficient to the vortex wake geometry. More refined models have been developed by adding further correlations based on the rotor load distribution (Landgrebe et al. [40], Kocurek & Berkowitz [34]). These methods have been quite successful in predicting hover performance for conventional rotor configurations. However, these schemes do not correctly model the flow physics, namely the transport of

vorticity in the wake. Their most serious limitation is their reliance on experimental data for predicting the wake geometry. This empiricism works well for rotors similar to those of the experimental data base. If unconventional planforms or twist distributions are used, the prescribed wake correlations are no longer valid and the results obtained with this approach must be treated with suspicion. For this reason, these methods are not reliable for new configurations, such as tilt-rotor aircraft, in which the wake geometry may differ considerably from that of a conventional helicopter rotor.

This limitation is overcome by free wake analysis methods (Clark & Leiper [18], Summa [64]). These methods are based on Helmholtz's theorem that vortex lines in an inviscid, incompressible fluid must lie along streamlines. The wake is modeled as vortex sheets and filaments, and the force free positions of these vortices are determined iteratively. Free wake methods are the most general approach to wake modeling currently being used. The price of this generality is that they are computationally expensive. For conventional rotors, free wake methods often give results no more accurate than the prescribed wake methods. For this reason, prescribed wake methods are favored in industry. However, free wake models, being firmly rooted in the flow physics, are better suited to unconventional configurations for which an experimental data base does not exist. Furthermore, free wake methods can provide insights into the physics of rotary wing wakes, something that prescribed wake approaches cannot do.

Recently, Miller [45,44,43] has developed a fast free wake method based on a simplified model of the rotor wake. Miller has replaced the helical wake vortices with either vortex lines (his two dimensional model) or vortex rings (three dimensional model) lying at the mean position of the vortex spirals below the blade. As with the more geometrically detailed free wake models, the force free positions of these vortices are found iteratively. Miller uses only two vortex filaments to represent the trailing vorticity: one tip vortex

and one inboard vortex representing the inboard vortex sheet. This wake model has also been used by Roberts & Murman [54,46] who have shown that more than one vortex filament is required to model the inboard portion of the vortex sheet accurately. The results obtained with these simple free wake models show that reasonably accurate results can be obtained for a small computational cost, making the fast free wake analysis method a useful tool for hover performance prediction.

Although the free wake model of the rotor wake captures the flow physics by allowing the transport of vorticity, some limitations in the model still exist. It is based on the assumption of a potential flow, meaning that distributed vorticity is not admitted; the wake is modeled as vortex sheets and filaments, and convection of the wake elements is treated in a Lagrangian fashion. Any distributed vorticity, such as in the tip vortex core, cannot be treated with potential methods. If rotational flow fields are to be computed, either the Euler or Navier-Stokes equations must be used to model the flow. The solution of these equations requires the use of a fixed grid in an Eulerian reference frame. Liu et al. [41] have presented a solution of the incompressible Navier-Stokes equations for a rotor in hover. Unfortunately, their solution does not show the expected contraction of the wake, possibly because of their lack of a model for the far wake. Also, they considered a flow having a Reynolds number much lower than exists for most rotors of engineering interest. Much work needs to be done in developing Navier-Stokes methods for computing the vortical wake of a hovering rotor.

The modeling of the rotor wake is of course only part of the hover problem. The aerodynamic loads on the blade must also be computed. Furthermore, there is a very close coupling between the detailed near field flow about the rotor blades and the subsequent roll up and convection of the wake. In the next section, the computation of the flow field about the rotor blades and the coupling of this to the wake model are examined.

1.3 Rotor Blade/Wake Coupling

The most common method of computing the aerodynamic loads of helicopter rotor blades is to model them as lifting lines or lifting surfaces. These models are based on the assumption of small disturbances to an inviscid, irrotational flow. Lifting line theory requires the further assumption that three dimensionality may be treated as a perturbation on a two dimensional flow about each rotor blade section. This latter assumption is violated at the tip of the blade, and in the blade/vortex interaction region if the tip vortex is sufficiently strong and close to the blade. Lifting surface theory is three dimensional, so that the blade vortex interaction is more accurately represented. One restriction of these blade models is the linearized treatment of the boundary conditions. More geometric generality may be acheived by using a surface singularity method (panel method) to model the blade (Summa [64], Morino et al. [48]). This approach is based on the Green's function method for the Laplace equation, and strictly speaking is confined to incompressible flows.

Lifting line, lifting surface, and panel methods treat the portion of the trailing vortex wake attached to the blade as either a fixed or free sheet. The prescribed wake methods and the simplified free wake model of Miller [45] fix the position of the trailing vortex sheet. Miller uses the computed bound circulation distribution to determine the strength of the vortices in the free portion of the wake. In the free wake methods of Summa [64] and Morino et al. [48] the paneling of the attached near wake corresponds to the beginning of the free wake, and the positions of the attached wake elements are found iteratively as part of the free wake solution procedure. This provides a natural and very close coupling of the wake geometry solution to the near field flow around the blade.

The governing equation for the lifting line, surface, and panel methods is the Laplace equation, which is exact for an incompressible potential flow.

However, the tip Mach number is often large enough that compressibility effects are significant. This can be accounted for through the use of an approximate compressibility correction, such as the Prandtl-Glauert rule. However, if transonic speeds are reached at the blade tip, the linear governing equation is no longer adequate to model the flow. For these cases either the transonic small perturbation or full potential equation is required to compute the flow about the rotor. To solve these equations, a finite difference or finite volume approach must be used. Such methods require the value of the potential to be defined at fixed points in an Eulerian reference frame. This in turn makes it necessary to generate a grid system around the rotor blade, which is called the computational domain. By solving the equation, the flow field is known throughout the computational domain.

The transonic small perturbation (TSP) equation is based on small geometric disturbances to the flow of an inviscid, irrotational ideal gas at near sonic velocities. This allows the use of linearized boundary conditions at the rotor surface, and simplifies the grid generation task. This equation has been applied to the case of a rotary wing in hover by Caradonna et al. for both non-lifting [13,6] and lifting flows[12]. The TSP equation is strictly valid only in the transonic range, and as with lifting surface theory, the linearized treatment of the boundary conditions is not valid at a blunt leading edge. The full potential equation, on the other hand, is valid from transonic to subsonic speeds, and allows the rotor geometry to be more accurately modeled. This requires the generation of a boundary conforming grid system. It has recently been used for a hovering rotor by Strawn & Caradonna [63] and Egolf & Sparks [24].

Coupling a finite difference potential solver to a rotor wake model is complicated by the fact that the treatment of the vortex wake on a fixed Eulerian mesh is somewhat more difficult than for surface integral methods. Vortex sheets and lines must be represented by branch cuts in the computa-

tional domain. In Caradonna et al. [12], Egolf & Sparks [24], and Strawn & Caradonna [63], the vortex wake is prescribed. The trailing vortex system of the blade is treated as a quasi-planar sheet lying along a coordinate surface. The tip vortex spirals below the blade are fixed in space, and are represented by branch cuts in the domain. Strawn & Caradonna used an experimentally determined wake geometry, while Egolf & Sparks used the Kocurek & Tangler [35] prescribed wake model. This is an effective approach when the wake vortices do not lie too close to the rotor blades. Because the potential equation does not convect vorticity, modeling a free vortex wake requires a Lagrangian treatment of the sheet within the finite difference domain. In general, a free sheet will not lie along a coordinate surface, complicating the branch cut boundary condition. Murman & Stremel [49] and Steinhoff & Suryanarayanan [62] have treated the problem of vortex sheet roll up using a finite difference potential solver. More work remains to be done on this approach.

The formation of the tip vortex and the structure of the trailing vortex sheet shed from the rotor blade are, of course, dependent upon the viscosity of the fluid and the enforcement of the no slip condition at the solid surface. Potential models cannot compute this process. The Kutta condition provides a model for specifying separation from sharp trailing edges and tips. Some additional separation model is needed to approximate the tip vortex formation around a rounded tip (e.g. Summa [64]). Furthermore, if the tip vortex of one blade passes sufficiently close to the following blade, the distortion of the vortex path and changes in the core structure cannot be handled with a potential method. To compute strong blade/vortex interactions as well as the roll up of the wake as it comes off the blade, the Euler or Navier-Stokes equations are needed to solve for the near field flow around the blade. Although viscous forces provide the physical mechanism for the separation of the vortical wake from the blade, the roll up and convection of

the wake is primarily an inviscid phenomenon. This suggests that the Euler equations, because they admit vortical solutions, should be an adequate model for computing the near field flow around a rotor blade, provided they yield a realistic model of separation.

The Euler equations for an inviscid, non-heat conducting ideal gas have been used to compute the flows around fixed wing configurations (e.g. Jameson & Baker [31], Rizzi & Eriksson [53]). Researchers have found that, contrary to expectations, the Kutta condition at a sharp trailing edge need not be explicitly enforced. The usual explanation for this turn of events is that the artificial viscosity of these schemes mimics the effect of real viscosity at sharp edges. More puzzling is the fact that separation is observed around smooth edges, such as rounded wing tips. Again, artificial viscosity is the suspected culprit, and it has been suggested that this separation would not occur if the grid were suitably refined. The mechanism for separation in Euler codes has not been clarified, and much work needs to be done in this area.

Although the cause of separation in Euler codes is not completely understood, it has been observed that the rolled up vortical wakes computed by such methods appear to be realistic models of real wakes. Much work has been done on leading edge separation around slender configurations in particular, with emphasis on understanding the nature of the rolled up vortices. Powell et al. [50] have examined the nature of leading edge vortices computed using the conical form of the Euler equations. The total pressure loss in the vortex core was observed to be insensitive to such numerical parameters such as the magnitude of the artificial viscosity and the refinement of the grid. Furthermore, the total pressure loss was very similar to that observed experimentally. Powell et al. proposed that the total pressure loss is due the discrete nature of the computed vortex sheet. Finite volume solutions of the Euler equations must yield a sheet with a finite thickness rather

than a contact discontinuity. This thickness results in a total pressure loss as the tangential velocity goes through zero across the sheet. Powell and his colleagues further argue that for this reason, the discrete Euler equations mimic the effect of viscosity in a real fluid, and hence realistically model a shear layer in a high Reynolds number flow. If this is the case, then finite volume solutions of the Euler equations should be a better model for vortical flows than might be expected at first. The Navier-Stokes equations may only be necessary for flows in which viscous effects cannot be neglected, such as flows with large scale separation.

Because the Euler equations admit vortical solutions, they can be used to compute the flow in the wake region of a rotor as well as around the blade. This avoids the assumption of an incompressible potential flow with embedded vortex sheets required by the Lagrangian free and prescribed wake methods described in the previous section. In principle, a finite volume grid can be constructed that extends from the rotor blade near field to the wake region below the rotor, and the entire flow field of the rotor may be found as part of the same solution procedure. This eliminates the need to couple a wake model to the rotor blade near field solution, and is similar in philosophy to the free wake panel method of Summa [64]. However, this approach is not practical. The vortical regions in the wake are typically very compact. To be able to resolve the wake structure below the blade, an extremely fine grid is needed in the region of the vortex core. Either a globally fine grid is required or some form of local refinement must be used. The first option results in excessive resolution in regions where the flow gradients are small. The second option requires either a priori knowledge of the location of the wake vortices or an adaptive refinement strategy. This complicates the algorithm for solving the equations. A second problem is that the artificial viscosity required by the Euler solver will result in a non-physical diffusion of the vortex as it is convected below the blade, although real viscosity or

inviscid instabilities may diffuse it. An attempt to overcome these problems has been made by Steger & Kutler [61], who computed the vortical flow in the wake of an aircraft using a very simple adaptive grid strategy and fourth order accurate spatial differencing. Finally, since the rotor wake is of infinite axial extent below the rotor, some model for the portion of the wake lying outside the computational domain is required in order to get the proper wake contraction and induced velocities near the blade. For these reasons, it is preferable to model the wake separately, and to couple it to the Euler solution of the near field of the blade. To take full advantage of the properties of the Euler equations, the coupling strategy should allow the computation of strong blade/vortex interactions in which the tip vortex passes very close to the trailing blade, including the situation in which the blade cuts through the rotational core of the vortex.

Little work has been done to date using the Euler equations for hovering rotors. Sankar et al. [56] have presented one solution technique using the Euler equations coupled to a wake model. Their approach consists of writing the state vector as a perturbation about the velocity field induced by the vortex wake. The wake is modeled as a single tip vortex spiral whose position is prescribed below the rotor blade. A further simplification is to ignore the spanwise and chordwise induced velocity components, and to write the perturbation about the axial component of the induced velocity in a limited region of the computational domain near the rotor blade. Although the method demonstrated by Sankar et al. is relatively simple, it does not make full use of the advantages of the Euler equations over potential methods. This approach is effectively a downwash, or angle of attack, correction at the blade, turning the Euler solver into an extended lifting line method. Also, Sankar and his co-workers cannot compute the strong interactions of a tip vortex with a rotor blade because of the excessively simple inclusion of the wake influence. Finally, a more complete wake model is required to

accurately predict the aerodynamic loads on the rotor.

In a more recent paper, Agarwal and Deese [1] also solve the Euler equations for the flow around a rotor blade. As in Sankar et al., the approach of Agarwal and Deese is essentially to correct the angle of attack of the blade to account for the wake influence. Unlike Sankar and his co-workers, they use the results of a free wake calculation to determine the induced angle of attack at the rotor blade, and this is translated into a effective geometric twist distribution. The free wake solution and the Euler computation are performed independently; there is no coupling or iteration between the two. In the cases they present in [1], they make a further simplification by simply adjusting the collective pitch rather than giving the blade a new twist distribution. As with Sankar et al., this fails to take full advantage of the properties of the Euler equations over the potential equation. Agarwal and Deese point out that to capture the rotor wake with the Euler solver, a highly refined grid must be used. They conclude that coupling the free wake solver to the Euler solver may provide an effective solution algorithm for hovering rotor flows.

In Roberts & Murman [55], an earlier version of the work described in this thesis is reported. As in Sankar et al., the wake is computed separately from the Euler solution around the blade. The wake model used is that of Roberts & Murman [54], which is essentially the vortex ring model developed by Miller [45]. For the case computed in [55], the induced velocity field of the entire semi-infinite wake is used to specify the far field boundary conditions for the Euler solver. This introduces the vortex wake into the finite volume computational domain. To reduce the smearing of the wake vortices due to artificial viscosity, the induced velocity of the entire wake is computed at each grid cell in the computational domain and subtracted from the total velocity field before the smoothing operator is applied. Although this reduces the smearing of the vortices, the truncation error in the

coarse regions of the grid still results in more diffusion of the vortex core than than is desired. Insufficient resolution in the far field means that the core structure of the tip vortex spiral cannot be specified properly at the boundary. Finally, the scheme did not fully couple the prediction of the wake geometry to the near field flow of the rotor blade.

With this background of previous work on the hover problem established, the approach and objectives of the present research will be discussed in the next section. Finally, an outline of the remainder of the thesis will be given.

1.4 Present Research

As discussed in the previous section, application of existing finite volume or finite difference methods to a hovering helicopter rotor is complicated by the fact that the wake of a rotor is very compact, making it difficult to compute the flows without smearing the wake excessively due to inadequate grid resolution and numerical dissipation. If these problems can be overcome, Euler methods will prove a valuable tool for understanding hovering rotary wing flow fields and serve as a necessary step towards a complete Navier-Stokes model.

The objectives of the current research are three-fold. First, the issue of whether numerical solutions of the Euler equations yield realistic models of the vortical structure and the roll up of the wake is addressed. This is done by comparing the solution of the Euler equations to the experimentally measured wake flow field of a conventional wing of moderate aspect ratio. Second, a method of introducing a streamwise vortex into the computational domain such that its structure remains well defined even in coarse regions of the grid is developed. This is to allow efficient computation of the blade/vortex interaction for a hovering rotor. The method is demonstrated by computing the interaction of a single streamwise vortex passing over a low aspect ratio wing and comparing with experiment. Finally, an iterative

method for coupling a free wake solution of a hovering rotor wake with the near field flow around a rotor blade is developed, and the results for a two bladed model rotor are compared to experiment.

In the next chapter, the finite volume algorithm for the Euler equations is presented, and the code validated for the ONERA M6 wing at transonic speed and an aspect ratio 6 wing of rectangular planform at a highly subsonic Mach number. The trailing vortex system computed in the latter case is compared to experiment, and the validity of the Euler equations for computing the structure of the wake is discussed. In chapter 3, a method for introducing a streamwise vortex into the Euler computational domain and computing its interaction with a wing is presented. The method is validated against the experimental data of Smith & Lazzeroni [57]. Chapter 4 discusses the coupling of the Euler solver with Miller's simplified free wake model. An iterative solution procedure for combining the two methods is presented, along with computations of a two bladed rotor in hover. These results are compared to the test data of Ballard et al. [5] and Caradonna & Tung [14]. Finally, conclusions are presented in chapter 5.

Chapter 2

Euler Solution Procedure

In this chapter, the governing equations for an inviscid gas, the Euler equations, are given, and the algorithm for solution of the equations is presented. The method used is the finite volume multistage scheme of Jameson [31]. The generation of the body fitted grid is discussed. An O-O grid topology is used, and the grid generator is the algebraic code of Eriksson [25]. Two test cases are presented for validation of the Euler code. The first is the ONERA M6 wing at transonic speed. Comparisons are made between computed surface pressures and the experimental data of reference [7]. The second test case is the rectangular planform wing tested by Weston [67] at a low Mach number. Comparisons are made with both surface pressures and wake surveys. The purpose of this comparison is to determine whether the numerical solution of the Euler equations yields a realistic model of the trailing vortical wake of a lifting wing.

2.1 Euler Equations

The flows considered here are taken to be steady. The Reynolds number is assumed to be high and the Prandtl number is of order unity, meaning that viscous and thermal effects are confined to thin shear layers. Flows with massive separation are not treated here. The outer inviscid flow is

then governed by the equations for an inviscid, adiabatic, ideal gas, called the Euler equations. With this model, the boundary layers and wakes are ignored, and only the Euler equations are solved for the outer flow. Although only steady flows are of interest here, solutions are found by solving the unsteady Euler equations in a time asymptotic fashion. The unsteady Euler equations are given here in integral form, and are derived from an application of the laws of conservation of mass, momentum, and energy to an arbitrary control volume in an Eulerian reference frame. The boundary conditions necessary for obtaining a steady state solution are also presented.

Consider the control volume V shown in Figure 2.1. Conservation of

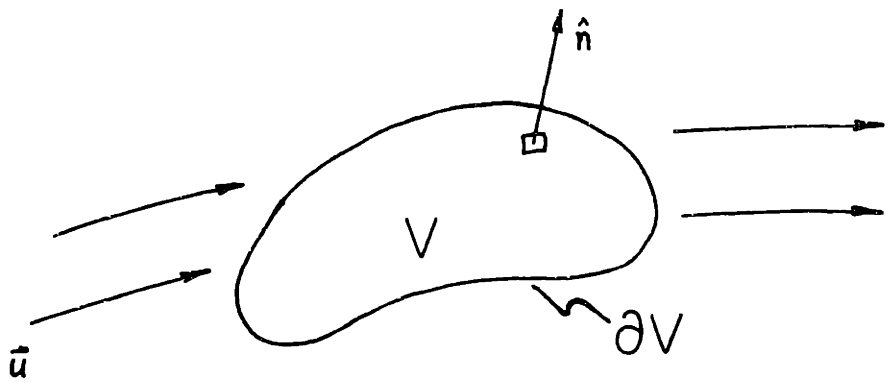

Figure 2.1: Control volume

mass requires that the time rate of change of the mass of fluid in the control volume equal the net flux across the boundaries. Writing this in integral

form gives

$$\frac{\partial}{\partial t}\iiint_V \rho d^3x = -\iint_{\partial V} \rho\vec{u}\cdot\hat{n}d^2x, \tag{2.1}$$

where ρ is the fluid density, $\vec{u}$ is the velocity, t is time, d^3x is a differential volume element, d^2x is a differential area element on the control volume surface, and $\hat{n}$ is the outward pointing normal at the control volume surface.

Applying Newton's second law to the flow through the control volume, we get

$$\frac{\partial}{\partial t}\iiint_V \rho\vec{u}d^3x = -\iint_{\partial V} \rho\vec{u}\,(\vec{u}\cdot\hat{n})\,d^2x - \iint_{\partial V} p\hat{n}d^2x. \tag{2.2}$$

where p is the static pressure of the fluid.

Conservation of energy across the control volume yields the energy equation,

$$\frac{\partial}{\partial t}\iiint_V \rho E d^3x = -\iint_{\partial V} (\rho E + p)\,\vec{u}\cdot\hat{n}d^2x \tag{2.3}$$

where

$$E = c_v T + \frac{\vec{u}\cdot\vec{u}}{2}.$$

Here T is the temperature of the fluid and c_v is the specific heat at constant volume.

Finally to close the system, an equation of state is required. This is given by the ideal gas law,

$$p = \rho RT \tag{2.4}$$

where $R = c_p - c_v$ is the ideal gas constant, c_p being the specific heat at a constant pressure.

The continuity, momentum, and energy equations can be written in a more convenient form given below,

$$\frac{\partial}{\partial t}\iiint_V \mathbf{U}\,d^3x + \iint_{\partial V} \vec{\mathbf{F}}(\mathbf{U})\cdot\hat{\mathbf{n}}\,d^2x = 0, \tag{2.5}$$

where

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}, \qquad \vec{\mathbf{F}}(\mathbf{U}) = \begin{pmatrix} \rho\vec{u} \\ \rho u\vec{u} + p\hat{i} \\ \rho v\vec{u} + p\hat{j} \\ \rho w\vec{u} + p\hat{k} \\ (\rho E + p)\vec{u} \end{pmatrix},$$

and

$$E = \frac{1}{\gamma - 1}\frac{p}{\rho} + \frac{\vec{u}\cdot\vec{u}}{2}.$$

The vector $\mathbf{U}$ is called the state vector, and $\vec{\mathbf{F}}(\mathbf{U})$ is the flux vector; u, v, and w are the cartesian components of the velocity $\vec{u}$ in the x, y, and z directions, respectively, and $\hat{i}$, $\hat{j}$, and $\hat{k}$ are the corresponding unit vectors in those directions. The equation of state has been used to eliminate T from the energy equation; the symbol γ is the ratio of the specific heats, c_p/c_v, and is taken to be equal to 1.4. The steady state is reached when the surface integral in Equation (2.5) is zero.

The steady state boundary conditions required to complete the specification of the steady problem are now presented. At a stationary solid wall, there is no fluid flux across the surface. This is written as

$$\vec{u}\cdot\hat{n} = 0, \tag{2.6}$$

where $\hat{n}$ is the unit normal at the surface. In the far field upstream of the wing, the flow should approach a uniform stream,

$$\lim_{x\to-\infty} \mathbf{U} = \mathbf{U}_\infty. \tag{2.7}$$

where $\mathbf{U}_\infty = (\rho_\infty, \rho_\infty u_\infty, 0, 0, \rho_\infty E_\infty)^T$ is the state vector of the uniform free stream, x being taken as the free stream direction. The flow field is not uniform at downstream infinity for a lifting flow due to the existence of a vortical wake. The usual boundary condition in the Trefftz plane is that the flow perturbation in the streamwise direction vanishes,

$$\lim_{x\to+\infty} \frac{\partial \mathbf{U}}{\partial x} = 0. \tag{2.8}$$

Finally, the flow is required to separate from the body at sharp trailing edges. This is called the Kutta condition.

To solve Equations (2.5) numerically, it is useful to non-dimensionalize the dependent and independent variables. The reference values used to normalize the variables are arbitrary, the only requirement being that the choice of normalization constants be consistent. In this thesis, the density and pressure are normalized by their free stream values, ρ_∞ and p_∞, and the velocity is non-dimensionalized by $a_\infty/\sqrt{\gamma}$, where a_∞ is the free stream speed of sound. Lengths are normalized by an arbitrary length scale c, usually taken to be the chord of the wing, and t is normalized by $c\sqrt{\gamma}/a_\infty$. With these choices for the normalization constants, the non-dimensional Euler equations are identical to Equation (2.5). The non-dimensional free stream state vector becomes

$$\mathbf{U}_\infty = \begin{pmatrix} 1 \\ \sqrt{\gamma} M_\infty \\ 0 \\ 0 \\ \frac{1}{\gamma - 1} + \frac{\gamma M_\infty^2}{2} \end{pmatrix}, \tag{2.9}$$

where M_∞ is the free stream Mach number. In the remainder of this thesis, the non-dimensional equations will be referred to unless otherwise noted.

In the next sections of this chapter, the numerical algorithm for solving Equation (2.5) is presented.

2.2 Spatial Discretization

The finite volume spatial discretization used here is that developed by Jameson & Baker [31] and Rizzi & Eriksson [53]. This consists of dividing the computational domain into hexahedral cells (Figure 2.2). The state vector $\mathbf{U}$ is defined at the center of each cell. The flux vector at the cell center, $\vec{\mathbf{F}}(\mathbf{U})$, is computed from the state vector. To approximate the flux integral on the right hand side of Equation (2.5), the flux vectors at adjoining cells

are averaged to get the flux vector on a cell face. This is dotted into the projected area of the cell face. The sum of the outgoing fluxes across all six faces of the cell is computed to get the approximation to the right hand side of Equation (2.5).

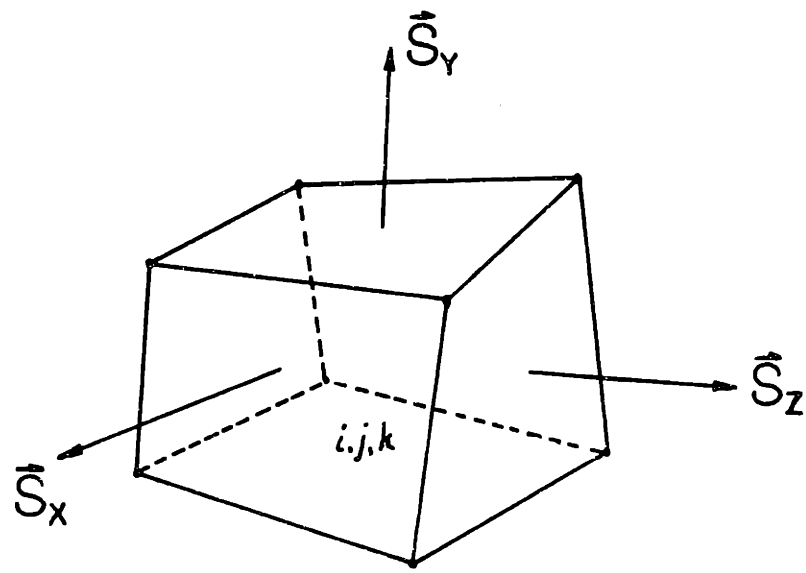

Figure 2.2: Finite volume cell

If X, Y, and Z are the computational coordinates in the i, j, and k directions, the discrete approximation to the flux integral can be written as

$$\begin{aligned} F_{i,j,k} &= \delta_X \left(\vec{S}_X \cdot \mu_X \vec{\mathbf{F}}(\mathbf{U})_{i,j,k}\right) + \delta_Y \left(\vec{S}_Y \cdot \mu_Y \vec{\mathbf{F}}(\mathbf{U})_{i,j,k}\right) \\ &\quad + \delta_Z \left(\vec{S}_Z \cdot \mu_Z \vec{\mathbf{F}}(\mathbf{U})_{i,j,k}\right). \end{aligned} \tag{2.10}$$

The operators δ_X and μ_X are the central difference and averaging operators in the X direction and are defined as

$$\delta_X f_{i,j,k} = f_{i+\frac{1}{2},j,k} - f_{i-\frac{1}{2},j,k}$$

and

$$\mu_X f_{i,j,k} = \frac{1}{2}\left(f_{i+\frac{1}{2},j,k} + f_{i-\frac{1}{2},j,k}\right).$$

Analogous operators are found for the Y and Z directions. The values $\vec{S}_X$, $\vec{S}_Y$, and $\vec{S}_Z$ are the projected face areas in the positive X, Y and Z directions, respectively (Figure 2.2).

This approximation to the flux integral can be seen to yield a central difference discretization of the equations, which is second order accurate on a cartesian grid. Central difference algorithms for first order equations are dispersive rather than dissipative (see Anderson, et al. [2]). This is because the truncation error consists of odd order higher derivatives rather than even order derivatives. For nonlinear equations, this can lead to instability due to aliasing errors. As energy cascades from long wavelengths to short wavelengths, the shortest wavelengths cannot be resolved on the finite volume grid. As a result, these waves show up as distorted long waves. To eliminate this unphysical behavior, a dissipative mechanism must be added to the discrete Euler equations. The form of the dissipation term is described below.

2.3 Artificial Viscosity

The artificial viscosity model, or dissipation operator, used is that of Jameson et al. [32]. It consists of two terms, a fourth difference and a second difference expression. The fourth difference artificial viscosity has the following form:

$$D^{(4)}_{i,j,k} = -\left\{\delta_X\left(\frac{V_{i,j,k}}{\tau_{i,j,k}}\epsilon^{(4)}_{i,j,k}\delta^3_X \overline{\mathbf{U}}_{i,j,k}\right) + \cdots\right\}. \tag{2.11}$$

Only the difference in one coordinate direction is shown; the differences in the other two coordinate directions are similar. In Equation (2.11), $\epsilon^{(4)}_{i,j,k}$ is the fourth difference dissipation coefficient, $V_{i,j,k}$ is the cell volume, and

$\tau_{i,j,k}$ is the time step for the cell, scaled to a *CFL* number of 1; $\overline{\mathbf{U}}_{i,j,k}$ is the state vector at cell i,j,k with ρE replaced by $\rho E + p$ in the energy equation. Since $\rho E + p = \rho H$, where H is the total enthalpy, this has the effect of admitting a constant total enthalpy flow as an exact solution to the discrete equations. This is a desirable property, since the flows considered here will have a constant total enthalpy in the steady state. This also allows the use of enthalpy damping, which will be described in section 2.6. The definition of the fourth order coefficient will be given shortly.

The inclusion of the cell volume $V_{i,j,k}$ is necessary to make the dissipation term consistent with the flux integral term and with the time derivative term in Equations (2.5). The unscaled time step term, $\tau_{i,j,k}$, is equivalent to scaling the dissipation term by the spectral radius of the flux Jacobian, $\partial\vec{\mathbf{F}}/\partial\mathbf{U}$. Pulliam [51] has analyzed artificial dissipation models for the Euler equations, and he shows that by using a Jacobian scaling term in the artificial viscosity in combination with a central difference discretization of the flux integral results in a difference scheme that emulates an upwind differencing algorithm.

The fourth difference dissipation provides a level of background dissipation sufficient to stabilize the time marching algorithm, and to kill the odd-even decoupling of the solution typical of central difference algorithms for the Euler equations. In transonic flows, the fourth difference artificial viscosity operator is insufficient to capture shocks. A second difference artificial viscosity is required. This takes the form

$$D^{(2)}_{i,j,k} = \left\{ \delta_X \left(\frac{V_{i,j,k}}{\tau_{i,j,k}} \epsilon^{(2)}_{i,j,k} \delta_X \overline{\mathbf{U}}_{i,j,k} \right) + \cdots \right\}, \tag{2.12}$$

where $\epsilon^{(2)}_{i,j,k}$ is the second difference dissipation coefficient.

The two smoothing coefficients in the X direction are determined from the formulas

$$\epsilon^{(2)}_{i+\frac{1}{2},j,k} = \kappa^{(2)} \max \left(\left| \frac{\delta^2_X p_{i,j,k}}{4\mu^2_X p_{i,j,k}} \right|, \left| \frac{\delta^2_X p_{i+1,j,k}}{4\mu^2_X p_{i+1,j,k}} \right| \right), \tag{2.13}$$

$$\epsilon^{(4)}_{i+\frac{1}{2},j,k} = \max\left(0, \kappa^{(4)} - \epsilon^{(2)}_{i+\frac{1}{2},j,k}\right) \tag{2.14}$$

with similar expressions for the Y and Z directions. The form of the second difference dissipation coefficient is designed so that it turns on only near shocks. In smooth regions of the flow field, the second difference of pressure is small, and is formally of second order in the grid spacing. This makes $D^{(2)}$ formally of third order in the grid spacing. Near shocks, the pressure switch is of order unity, and locally the second difference artificial viscosity becomes first order. Typical values of $\kappa^{(2)}$ are 0.1 to 0.25.

Because the second difference smoothing is necessary solely to capture shocks, it is not needed for shock free flows. If a purely subsonic flows is computed, it has been found that the fourth difference alone is adequate to stabilize the calculation. For these flows, $\kappa^{(2)}$ is set to zero.

The fourth difference artificial viscosity provides a background dissipation in order to stabilize the time stepping scheme. However, it will cause wiggles, and can possibly be destabilizing, at a shock. The form of $\epsilon^{(4)}$ is chosen such that in smooth regions of the flow, where the pressure gradients are mild, the coefficient takes on its largest values. Near shocks, where the second difference pressure switch becomes of order unity, the fourth difference artificial viscosity coefficient is turned off. Note that the fourth difference artificial viscosity is formally a third order quantity in the grid spacing. The value of $\kappa^{(4)}$ is chosen to be between 0.004 and 0.01.

2.4 Multistage Time Integration

Applying the spatial discretization and artificial viscosity operators to the Euler equations on the finite volume grid, one obtains the semi-discrete equations

$$\frac{d\mathbf{U}_{i,j,k}}{dt} = -\frac{1}{V_{i,j,k}}\left\{F_{i,j,k} - D_{i,j,k}\right\}, \tag{2.15}$$

where the dissipation operator is

$$D_{i,j,k} = D^{(2)}_{i,j,k} + D^{(4)}_{i,j,k}.$$

This forms a large system of coupled nonlinear ordinary differential equations for U. To integrate these equations, the multistage time stepping scheme of Jameson & Baker [31] is used.

The multistage scheme is applied at time level n as follows:

$$\begin{aligned}
\mathbf{U}^{(0)} &= \mathbf{U}^n, \\
\mathbf{U}^{(1)} &= \mathbf{U}^{(0)} - \alpha_1 \frac{\Delta t}{V}\left(F^{(0)} - D^{(0)}\right), && \text{(2.16a)} \\
\mathbf{U}^{(2)} &= \mathbf{U}^{(0)} - \alpha_2 \frac{\Delta t}{V}\left(F^{(1)} - D^{(0)}\right), && \text{(2.16b)} \\
\mathbf{U}^{(3)} &= \mathbf{U}^{(0)} - \alpha_3 \frac{\Delta t}{V}\left(F^{(2)} - D^{(0)}\right), && \text{(2.16c)} \\
\mathbf{U}^{(4)} &= \mathbf{U}^{(0)} - \alpha_4 \frac{\Delta t}{V}\left(F^{(3)} - D^{(0)}\right), && \text{(2.16d)} \\
\mathbf{U}^{n+1} &= \mathbf{U}^{(4)}.
\end{aligned}$$

The multistage coefficients are

$$\alpha_1 = \frac{1}{4}, \alpha_2 = \frac{1}{3}, \alpha_3 = \frac{1}{2}, \alpha_4 = 1.$$

The artificial smoothing is evaluated only at the initial stage of the temporal integration. This is to reduce the operation count for the scheme. The time step Δt for cell i, j, k is found from the formula

$$\Delta t_{i,j,k} = CFL \frac{V_{i,j,k}}{\left|\vec{u} \cdot \vec{S}_{max}\right| + c\left|\vec{S}_{max}\right|}, \qquad (2.17)$$

where c is the speed of sound, $\vec{S}_{max}$ is the vector sum of the maximum projected areas of the cell in the x, y, and z directions, and CFL is the Courant-Friedrichs-Lewy number, which is chosen by the stability limit of the multistage scheme. For the coefficients of the time stepping scheme given here,

$$CFL \leq 2\sqrt{2}$$

(see appendix A). Most of the cases presented in this thesis were run at the maximum allowable *CFL* number given above. As shown here, the time step for each cell is chosen such that the *CFL* number is constant throughout the computational domain. Larger cells will run at larger time steps than small cells. As a result, the integration is no longer time accurate, but the convergence to the steady state is accelerated. Essentially, the temporal integration scheme has become an iteration path to the steady state, but the solution at any intermediate iteration level no longer has any physical meaning.

One important feature of the multistage temporal integration scheme presented here is that the steady state operator, $F_{i,j,k} - D_{i,j,k}$, is independent of the time step used in the integration.

Note here that the value of $\tau_{i,j,k}$ which is used to scale the artificial viscosity operator presented in the last section is simply equal to $\Delta t_{i,j,k}/CFL$. This is necessary to make the artificial viscosity independent of the *CFL* number used to reach steady state.

2.5 Boundary Conditions

Boundary conditions on the computational domain are required to maintain a properly posed initial-boundary value problem, just as they are required analytically. However, the number of boundary conditions which may be prescribed mathematically are not sufficient to close the discrete equation system. Extra relations must be derived from the local analytic behavior of the governing partial differential equations. These extra relations, and their physical significance, are discussed in this section.

2.5.1 Solid Wall

At the solid wall, we have the physical boundary condition given by Equation (2.6), namely, the requirement of no flux through the wall. For

the continuity and energy equation, this is easily incorporated by simply setting the mass and energy fluxes to zero at the wall. However, in the momentum equation, there is still a pressure contribution at the solid wall. To obtain the value of the pressure at the wall, the pressure is extrapolated from the interior using the normal momentum equation as formulated by Rizzi [52]. The expression is obtained by writing the momentum equation at the wall and dotting it into the unit normal,

$$\rho \frac{\partial \vec{u}}{\partial t} \cdot \hat{n} + (\rho \vec{u} \cdot \nabla \vec{u}) \cdot \hat{n} = -\nabla p \cdot \hat{n}$$

where $\hat{n}$ is the unit normal at the wall. Using Equation (2.6) and noting that in wing-fixed coordinates $\partial \hat{n}/\partial t = 0$, the momentum equation may be rewritten as

$$\rho \vec{u} \cdot (\vec{u} \cdot \nabla \hat{n}) = \frac{\partial p}{\partial n}. \tag{2.18}$$

Equation (2.18) gives the pressure gradient normal to the solid wall in terms of the known surface curvature and the velocity at the wall. The velocity at the wall is taken to be the tangential component of the velocity in the first computational cell off the body. With this, Equation (2.18) is solved for $\partial p/\partial n$ at the wall, and the pressure is extrapolated to the wall from the first interior cell.

2.5.2 Far Field

Analytically, we have the requirement that the flow disturbances vanish at upstream infinity, Equation (2.7), and that streamwise perturbations vanish in the Trefftz plane, Equation (2.8). These are more difficult boundary conditions to apply, since the computational domain only extends a few chords from the wing at which distance the disturbances will not have vanished. Also, the far field boundary conditions given above are for the steady state flow. Since unsteady equations are being marched in time to reach the steady state, the problem being solved is an initial-boundary value problem.

The numerical boundary conditions applied at the computational far field are developed with the requirement that the transient disturbances pass through the boundary with mimimum reflections. The far field boundary conditions used are those of Jameson & Baker [31], who use the theory of characteristics to satisify these conditions.

At the far field boundary, the Euler equations are written in coordinates normal and tangential to the boundary,

$$\frac{\partial \mathbf{U}}{\partial t} = A\frac{\partial \mathbf{U}}{\partial N} + B\frac{\partial \mathbf{U}}{\partial T_1} + C\frac{\partial \mathbf{U}}{\partial T_2} \tag{2.19}$$

where N, T_1, and T_2 are local cartesian coordinates normal and tangential to the boundary, and A, B, and C are the N, T_1, and T_2 components of the flux Jacobian $\partial\vec{\mathbf{F}}/\partial\mathbf{U}$. The normal direction is taken to be positive pointing out of the domain. The boundary conditions are developed by assuming all incident waves are normal to the boundary in the far field, so the tangential derivatives may be taken to be zero. Equation (2.19) then reduces to

$$\frac{\partial \mathbf{U}}{\partial t} = A\frac{\partial \mathbf{U}}{\partial N}. \tag{2.20}$$

A similarity transform can be found which diagonalizes the Jacobian matrix A in Equation (2.20). If S^{-1} and S are the matrices of the left and right eigenvectors of A, respectively, then the matrix

$$\Lambda = S^{-1}AS = \begin{bmatrix} u_n & 0 & 0 & 0 & 0 \\ 0 & u_n & 0 & 0 & 0 \\ 0 & 0 & u_n & 0 & 0 \\ 0 & 0 & 0 & u_n + a & 0 \\ 0 & 0 & 0 & 0 & u_n - a \end{bmatrix}, \tag{2.21}$$

is a diagonal matrix of the eigenvalues of A; u_n is the velocity normal to the boundary (positive outward) and a is the speed of sound at the boundary. The one-dimensional equations are then written in compatibility form,

$$S^{-1}\frac{\partial \mathbf{U}}{\partial t} + \Lambda S^{-1}\frac{\partial \mathbf{U}}{\partial N} = 0, \tag{2.22}$$

and the values of the characteristic variables are determined. If the assumption is made that the flow is locally isentropic, then the compatibility equations 2.22 can be rewritten in the form

$$\frac{\partial \tilde{\mathbf{U}}_i}{\partial t} + \lambda_i \frac{\partial \tilde{\mathbf{U}}_i}{\partial N}, \tag{2.23}$$

for $i = 1$ to 5, where λ_i is the i^{th} eigenvalue A and $\tilde{\mathbf{U}}_i$ is the corresponding characteristic variable. The characterisic variables associated with each eigenvalue are:

$$u_n \;:\; s \tag{2.24a}$$

$$u_n \;:\; u_{t1} \tag{2.24b}$$

$$u_n \;:\; u_{t2} \tag{2.24c}$$

$$u_n + a \;:\; u_n + \frac{2a}{\gamma - 1} \tag{2.24d}$$

$$u_n - a \;:\; u_n - \frac{2a}{\gamma - 1} \tag{2.24e}$$

where s is the entropy, and u_{t1} and u_{t2} are the components of velocity tangential to the boundary. The last two variables are the Riemann invariants. The compatibility equations thus can be seen to correspond to entropy, vorticity, and acoustic waves normal to the boundary.

It should be noted that the form of the compatibility equations given here is not unique, and if the isentropic assumption is not used another set of equations may be found (e.g., see Courant & Hilbert [20], pp. 434-436).

From the theory of characteristics, it is known that the number of boundary conditions specified should equal the number of characteristics entering the domain at the boundary, which correspond here to those associated with the negative eigenvalues of A. At a subsonic inflow u_n and $u_n - a$ are negative and $u_n + a$ is positive, so four characteristics enter the domain, corresponding to the incident entropy, vorticity, and downstream running acoustic waves. These four characteristic variables specified. The characteristic variable corresponding to the upstream running acoustic wave is extrapolated from the

interior in order to close the system at the boundary and determine the state vector there. At a subsonic outflow, there is one incoming characteristic, the upstream running acoustic wave, which is specified. The outgoing acoustic, vorticity, and entropy waves are extrapolated.

The implementation of the boundary conditions in the code is now described. Since none of the flows considered here have a supersonic free stream, only the subsonic boundary conditions will be described. The incoming and outgoing Riemann invariants are

$$r_{ex} = \vec{u}_{ex} \cdot \hat{n} + \frac{2a_{ex}}{\gamma - 1} = u_n + \frac{2a}{\gamma - 1}, \tag{2.25a}$$

and

$$r_{\infty} = \vec{u}_{\infty} \cdot \hat{n} - \frac{2a_{\infty}}{\gamma - 1} = u_n - \frac{2a}{\gamma - 1}. \tag{2.25b}$$

Here, $\vec{u}_{ex}$ and a_{ex} are, respectively, the velocity and speed of sound at the first interior point of the domain, and $\vec{u}_{\infty}$ and a_{∞} are the free stream values, and $\hat{n}$ is the unit normal pointing out of the domain.

The velocity normal to the boundary is found from

$$u_n = \frac{1}{2}\left(r_{ex} + r_{\infty}\right) \tag{2.26a}$$

and the speed of sound is given by

$$a = \frac{\gamma - 1}{4}\left(r_{ex} - r_{\infty}\right). \tag{2.26b}$$

At the inflow, the entropy is specified, and is given as

$$\frac{p}{\rho^{\gamma}} = 1. \tag{2.27a}$$

At an outflow, the entropy is extrapolated:

$$\frac{p}{\rho^{\gamma}} = \frac{p_{ex}}{\rho_{ex}^{\gamma}}. \tag{2.27b}$$

The values of p and ρ are then formed from

$$\rho = \left[\frac{a^2}{\gamma\left(\frac{p}{\rho^{\gamma}}\right)}\right]^{\frac{1}{\gamma-1}}$$

and

$$p = \frac{\rho a^2}{\gamma}.$$

The tangential velocity is specified from the free stream at an inflow point and extrapolated at an outflow. This is most easily done by writing the total velocity as

$$\vec{u} = \vec{u}_\infty + (u_n - \vec{u}_\infty \cdot \hat{n})\hat{n} \tag{2.28a}$$

at an inflow boundary and

$$\vec{u} = \vec{u}_{ex} + (u_n - \vec{u}_{ex} \cdot \hat{n})\hat{n} \tag{2.28b}$$

at an outflow boundary. Doing it this way avoids having to explicitly compute the tangential component of velocity at a boundary point.

The boundary conditions are updated at each stage of the multistage time integration.

2.5.3 Artificial Viscosity

Establishing proper boundary conditions for the artificial viscosity terms is difficult. The mathematics of the governing partial differential equations does not tell us anything about the dissipation terms. Physical reasoning also fails us since the additional terms are not physical. Finally, the fact that the artificial viscosity uses a five point difference stencil in each coordinate direction means that special treatment is required at the first two cells adjacent to the boundary. The present treatment of the dissipation terms at the boundaries of the computational domain is based on the analysis of Eriksson [27], who developed a boundary treatment of the artificial viscosity that guarantees that the terms are globally dissipative. The implementation of his approach is presented here.

For the second difference term, the smoothing flux across a boundary face is set to zero. Let the subscripts 1 and 0 represent the values at an interior cell adjecent to the domain boundary and a dummy cell just outside

the boundary, respectively (Figure 2.3). This boundary condition on the

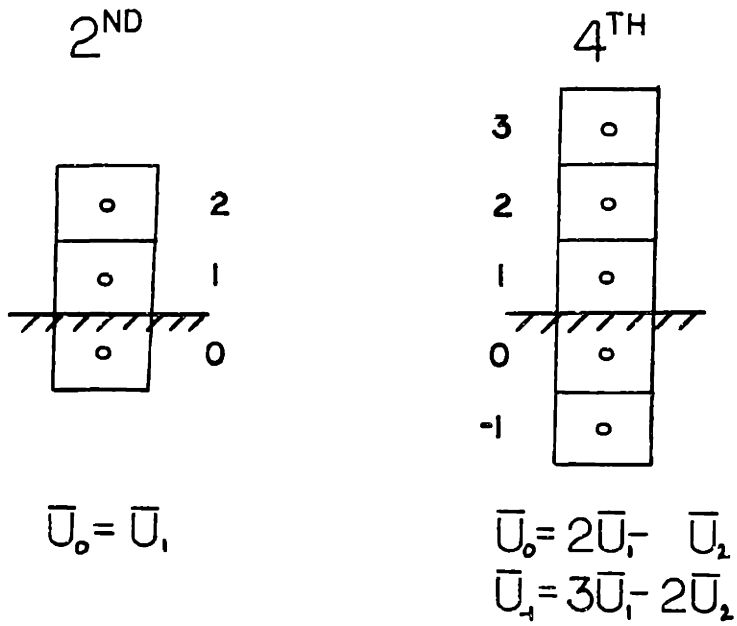

Figure 2.3: Artificial viscosity difference stencil at boundary

second difference dissipation is equivalent to setting

$$\overline{\mathbf{U}}_0 = \overline{\mathbf{U}}_1. \tag{2.29}$$

For the fourth difference artificial viscosity, Eriksson's treatment is equivalent to linearly extrapolating the values from the first two cells inside the domain to two dummy points outside the boundary, i.e.

$$\overline{\mathbf{U}}_0 = 2\overline{\mathbf{U}}_1 - \overline{\mathbf{U}}_2, \tag{2.30a}$$

$$\overline{\mathbf{U}}_{-1} = 3\overline{\mathbf{U}}_1 - 2\overline{\mathbf{U}}_2. \tag{2.30b}$$

The implementation of these boundary conditions is most easily accomplished by linearly extrapolating the state vector $\mathbf{U}$ and the pressure p according to Equations (2.30a) and (2.30b). Because the pressure is also

linearly extrapolated, the second difference dissipation coefficient $\epsilon^{(2)}_{i,j,k}$ becomes zero at the boundary face, which accomplishes the same thing as Equation (2.29). The same difference stencil for the artificial viscosity operator can thus be used at all the interior cells.

This boundary condition for the artificial viscosity is applied both at the solid wall and at the far field boundaries. At the symmetry plane, the flow variables are simply reflected across the boundary.

2.6 Enthalpy Damping

The flows being computed here are steady, and have a constant total enthalpy. The unsteady equations are used simply to provide an iteration path to the steady state, and time accuracy is not of concern. This is the justification for using local time stepping as described in the section on the multistage integration scheme, as this provides one way of accelerating the convergence of the code to the steady state. Another convergence acceleration approach used here is known as enthalpy damping, which has been proposed by Jameson [32]. This approach has been further examined by Turkel [65] and by Jespersen [33]. A detailed derivation of the modified equation set that results from the use of enthalpy damping is given in appendix B. The general outline of Jameson's approach to enthalpy damping is given below.

Jameson gives a heuristic development of the approach based on his experience with iterative solutions of the steady transonic potential equation. The argument he presents is based on an irrotational, unsteady subsonic flow. This flow can be described by the wave equation,

$$\frac{1}{a^2}\phi_{tt} - \phi_{\xi\xi} - \phi_{\eta\eta} - \phi_{\varsigma\varsigma} = 0, \tag{2.31}$$

where $\xi = x - ut$, $\xi = y - vt$, and $\varsigma = z - wt$; this is derived assuming by assuming constant velocity for the transformation (see appendix B). If this

equation is modified by adding a term proportional to the time derivative of ϕ, the telegraph equation,

$$\frac{1}{a^2}\phi_{tt} + \alpha\phi_t - \phi_{\xi\xi} - \phi_{\eta\eta} - \phi_{\zeta\zeta} = 0, \tag{2.32}$$

is obtained. This equation has solutions consisting of exponentially damped waves (see Courant & Hilbert [20], pp. 192-193). Jameson, in his development of methods for the solution of the potential equation for transonic flows, has found that the coefficient α used has a strong effect on convergence rate of the iterative scheme. He proposes that a similar term be added to the Euler equations to simulate the effect of the ϕ_t term in the potential equation. To do this, he notes that for an unsteady, irrotational flow with a uniform free stream, the Bernoulli equation becomes

$$\phi_t = H_\infty - H, \tag{2.33}$$

where H is the total enthalpy and H_∞ is the free stream total enthalpy. Therefore, Jameson suggests modifying the unsteady Euler equations by adding a term proportional to the difference in the local total enthalpy and free stream total enthalpy to the equation system. The modified equation set is

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{U}) + \alpha (H - H_\infty)\,\overline{\overline{\mathbf{U}}} = 0, \tag{2.34}$$

where

$$\overline{\overline{\mathbf{U}}} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho \end{pmatrix}.$$

The reason for replacing ρE with ρ in the energy equation is that otherwise, a term proportional to H^2 appears, which according to Turkel [65] can lead to difficulties, and according to Jameson et al. [32] can be destabilizing. Jameson has found it to be effective then to use the form in Equation (2.34).

There are two properties to note about the modified equation. First, the additional terms vanish when $H = H_\infty$. Thus for steady flows in which the total enthalpy is uniform, the steady state operator is unchanged. Using the modified equation will therefore not affect the steady state solution. The second point of interest is that the modified equation set has no physical meaning. The transients computed with these equations do not represent any physical transients. The justification for the use of the modified equations rests on the fact that only the steady state is of interest, and the path used to reach the steady state is irrelevant. If transient flow phenomena are of interest, Equations (2.34) cannot be used.

In solving this equation set, the approach used is to solve the four stage temporal integration for the standard equation set (2.5), and to add the enthalpy damping terms as a point implicit update of the state vector at the end of the time step. Also, it has been found to be most convenient to replace α with $\frac{\alpha}{\Delta t}$, so that the implicit update step becomes independent of the time step used in the multistage integration. Typically, the enthalpy damping coefficient α is taken to be 0.025, a value determined through numerical experimentation. It has been found that the enthalpy damping does not affect the maximum allowable time step of the basic multistage algorithm. Also, despite the fact that enthalpy damping is theoretically destabilizing (Turkel, [65]) in supersonic regions, it has not been found necessary to turn it off in these regions for transonic cases.

The advantages of the enthalpy damping for accelerating convergence have been found to be most pronounced for low Mach number flows. In the high subsonic to transonic range, enthalpy damping has not greatly affected the convergence rate. For highly subsonic flows, it has had a marked effect. For the lowest speed flows presented in this thesis ($M_\infty \leq 0.2$), the convergence rate has been observed to be as much as 5 times faster with enthalpy damping. Since most of the results shown here are in the subsonic

range, this has been a compelling reason for using enthalpy damping.

2.7 Grid Generation

The finite volume grids used for all the cases but one in this thesis are generated by the algebraic code of Eriksson [25]. The grid generator maps the space between the wing surface and a quasi-spherical outer boundary into a logical cube, as illustrated in Figure 2.4. Figure 2.5 shows the grid generated for the ONERA M6 wing. The grid has an O-O topology, in which both the chordwise and spanwise grid sections have an O-grid topology. The attractive feature of this grid is its relative economy in grid points around the wing. The grid points are clustered in the high gradient regions near the leading and trailing edges, and at the wing tip, and the grid is stretched in the far field where resolution is not needed. Transfinite interpolation is used to generate the grid, and it consists of using interpolating functions to compute the coordinate geometry between boundary planes in the computational domain.

One problem with mapping the space between the wing surface and the outer computational boundary onto a logical cube is that singular lines will arise in the computational domain where coordinate surfaces fold over one another. This can be seen in Figure 2.5, for the O-O grid, where the singular lines are seen to be emitted from the corners of the wing tip. Eriksson has done a classification and study of the coordinate singularities that arise in the mapping of the physical space to the computational space in reference [26]. He shows that the stability of the cell based finite volume scheme is not affected by the presence of the grid singularities. He also concludes that the truncation error in the vicinity of the singular lines becomes zeroeth order, but that the overall error of the scheme lies between first and second order in the grid spacing.

In the grid generator developed by Eriksson, the singular lines are made

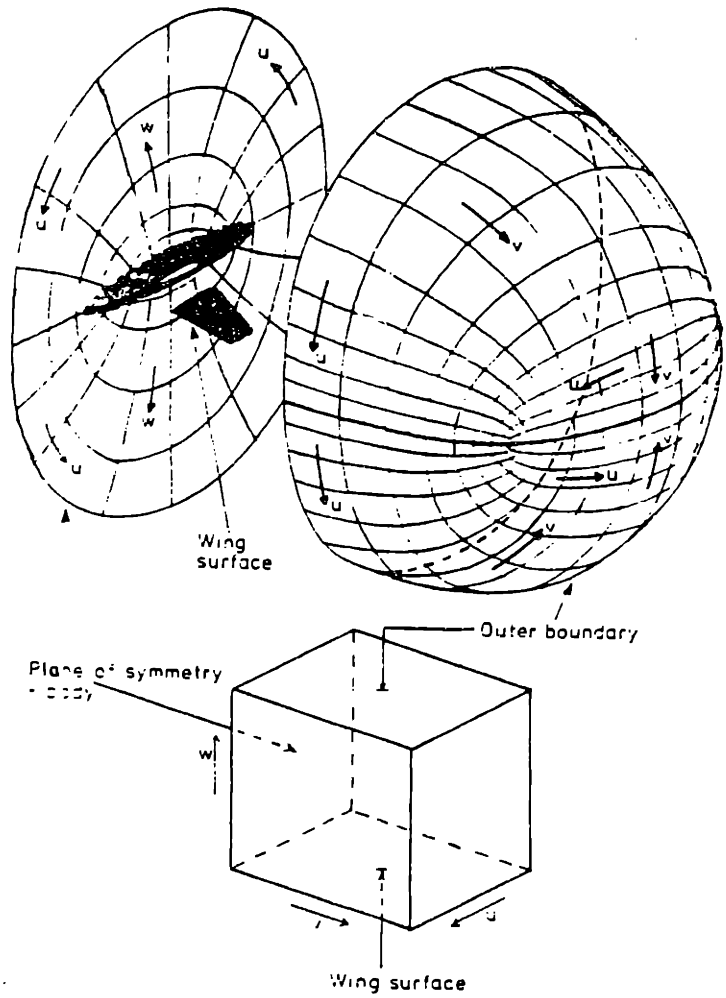

Figure 2.4: Mapping from physical space to computational space (from reference [25])

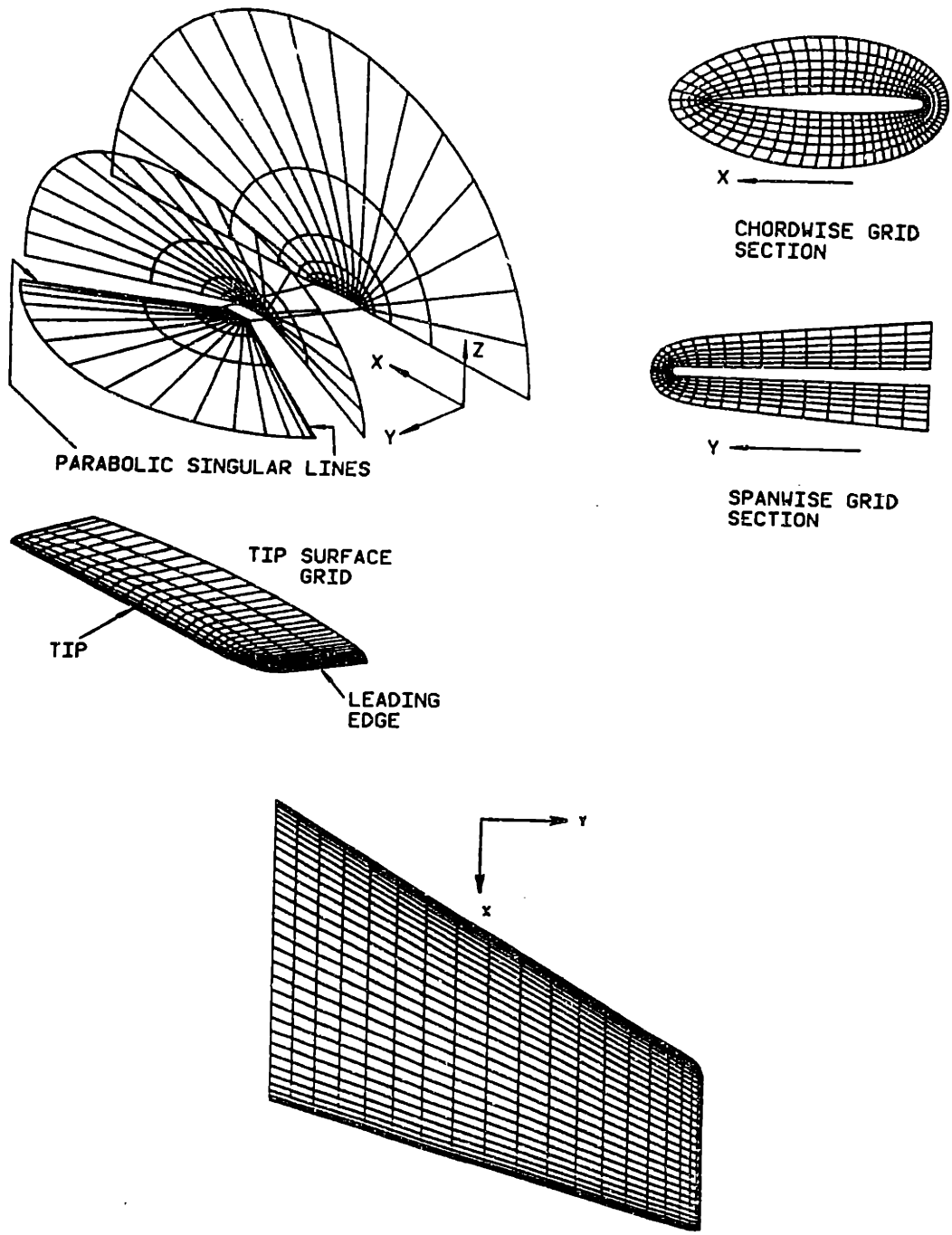

Figure 2.5: Grid generated for the ONERA M6 wing (from reference [53])

to come off the wing tip in the spanwise direction, and intersect the outer boundary along lines approximately 45° to the free stream direction and in the plane of the wing (Figure 2.5). This makes the grid well behaved in the wake region. Since one of the purposes of the present research is to examine the wake computed by the finite volume Euler equations, this is a very desirable property.

In the remainder of this chapter, the solutions for the ONERA M6 wing and the wing tested by Weston are presented and compared to experiment.

2.8 ONERA M6 Test Case

The first case presented here is the ONERA M6 wing at a free stream Mach number of 0.84 and an angle of attack of 3.06 degrees. The purpose of this case is to validate the present finite volume algorithm for the Euler equations. This case has been computed by a number of authors and therefore is a suitable test case of the present method. The computed pressure coefficients for this case are compared to the experimental values of reference [7]. The solution was obtained on a grid of 96 cells in the chordwise direction, 20 cells in the spanwise direction, and 20 cells from the wing surface to the far field boundary. This gives 38,000 grid cells, which is moderate resolution. It should be noted that this grid is identical to that used by Rizzi & Eriksson [53] for this same test case. The artificial viscosity coefficients used were $\kappa^{(2)} = 0.1, \kappa^{(4)} = 0.01$. The *CFL* number was 2.8 and the enthalpy damping coefficient was 0.025. The solution was obtained after 1000 iterations of the Euler solver, and took approximately 13 minutes of CPU time on the Cray X-MP/48. The iteration history is shown in Figure 2.6, where the root mean square of the change in the state vector at each iteration is shown. Chordwise distributions of the pressure coefficient are shown in Figure 2.7. The solid lines are the computed pressure coefficients, and the symbols are experimental values. The ordinate is $\sqrt{x/c}$ rather than the usual x/c. This

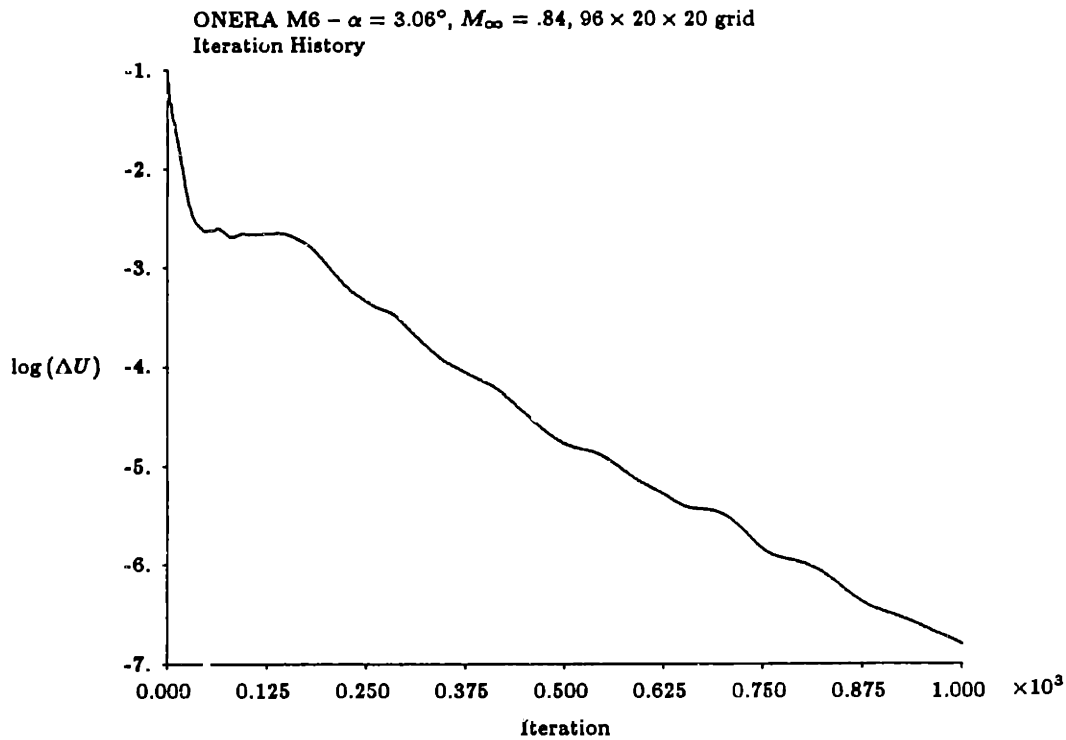

Figure 2.6: Convergence history for the ONERA M6 wing

has the effect of stretching the coordinates at the leading edge, making the rapid expansion in that region more clearly visible. All the surface pressure coefficient plots in this thesis are plotted this way. The agreement is good over most of the wing, with larger discrepancies near the root than at the tip. The weak shock at the leading edge and the stronger shock near the midchord are captured by the scheme, and can be seen to coalesce into a single strong shock near the tip. At the root, the aft shock is seen to be stronger and further aft than was observed experimentally. This difference may be attributed to shock/boundary layer interaction, which weakens the shock in transonic flow, and results in the shock being further upstream

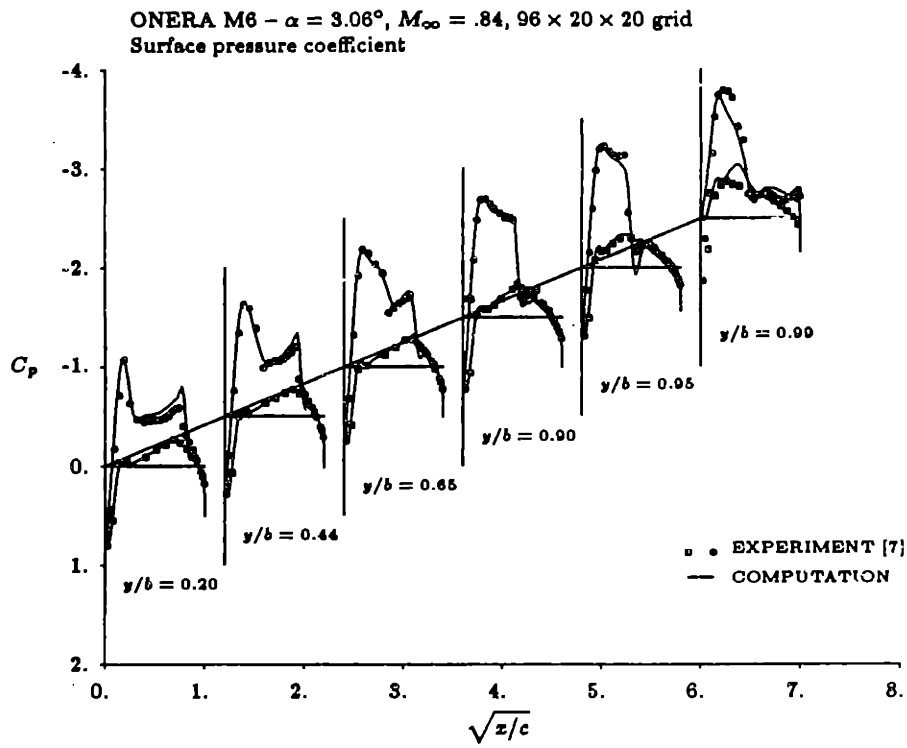

Figure 2.7: Computed and experimental surface pressures, ONERA M6 wing, $M_\infty = 0.84$, $\alpha = 3.06°$

than in an inviscid calculation. The agreement of the surface pressures at the leading edge with the observed suction peak is very good, although this is possibly fortuitous. Very good agreement is seen near the wing tip, where the leading edge and the midchord shocks coalesce into a single strong shock. This merging of the leading edge and midchord shocks is very clearly seen in Figure 2.8, in which Mach contours in the first cell off the wing surface are shown.

One of the more interesting features of this solution is the behavior at the tip ($y/b = 0.99$, Figure 2.7). The experimental data clearly show a

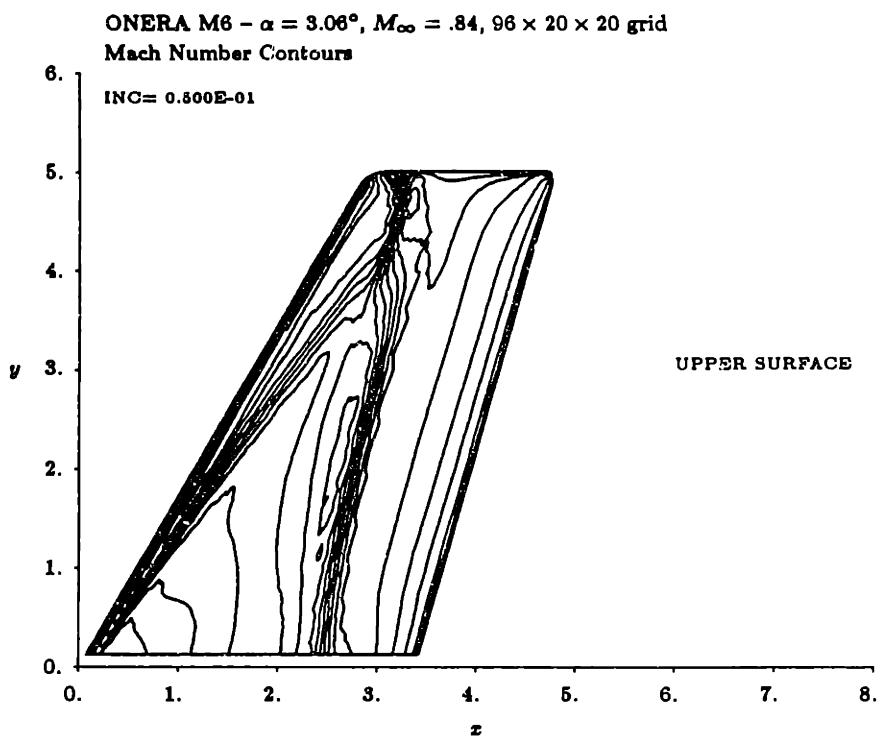

Figure 2.8: Surface Mach number contours, ONERA M6 wing, $M_\infty = 0.84$, $\alpha = 3.06°$

secondary suction peak on the upper surface of the airfoil near the trailing edge. Qualitatively similar behavior is seen in the computed results, in that there is lift being produced over the last 60% of the airfoil chord at the tip section. This behavior can be explained by the formation and roll up of the tip vortex over the wing. Here we have the situation discussed in chapter one, namely the separation of the flow around a rounded wing tip without specifying a Kutta condition. The discrete Euler equations, because of the addition of the numerical dissipation terms, are providing a pseudo-viscous simulation of the flow. That is, the artificial viscosity provides a mechanism

for the separation of the flow around the tip, although the details of this process are not clearly understood. Although this in a sense mimics the physics of a real viscous flow, the details of the separation process in the discrete Euler equations cannot be construed as simulating the real flow in that region. However, the large scale vortical structures in the wake may be insensitive to these details. This latter point will be addressed below.

2.9 Weston Test Case

The second case run for a fixed wing geometry is the wing tested by Weston at Langley Research Center [67]. This wing is of rectangular planform, untwisted, with a semispan to chord ratio of 3 and NACA 0012 airfoil section with a body of revolution tip. It was tested at low speed, corresponding to a Mach number of 0.1425. The experimental data consists of surface pressure coefficient measurements and detailed wake surveys, making it well suited for comparing the present solutions with the actual wake structure. The computations for this case were performed on a $128 \times 32 \times 32$ grid (132,072 cells), which is a reasonably fine resolution grid. The artificial viscosity coefficients were $\kappa^{(2)} = 0$ and $\kappa^{(4)} = 0.004$. The enthalpy damping coefficient was set to 0.1. To reduce the computation time for this case, 500 iterations were run on first on a coarse grid of $64 \times 16 \times 16$ cells, which was generated by ignoring every other grid point on the final grid. The coarse grid solution was then injected onto the fine mesh, and another 500 iterations were run. Total CPU time for this case was approximately 23 minutes on the Cray X-MP/48. The iteration history is shown in Figure 2.9.

The comparison of the surface pressures shown in Figure 2.10 are seen to be good. At the root section, the suction peak is seen to be higher than experiment. This appears to be due primarily to the flow angularity in the wind tunnel. Figure 2.11, taken from Weston [67], shows the measured flow angularity in the empty tunnel, which is considerable near the wing root

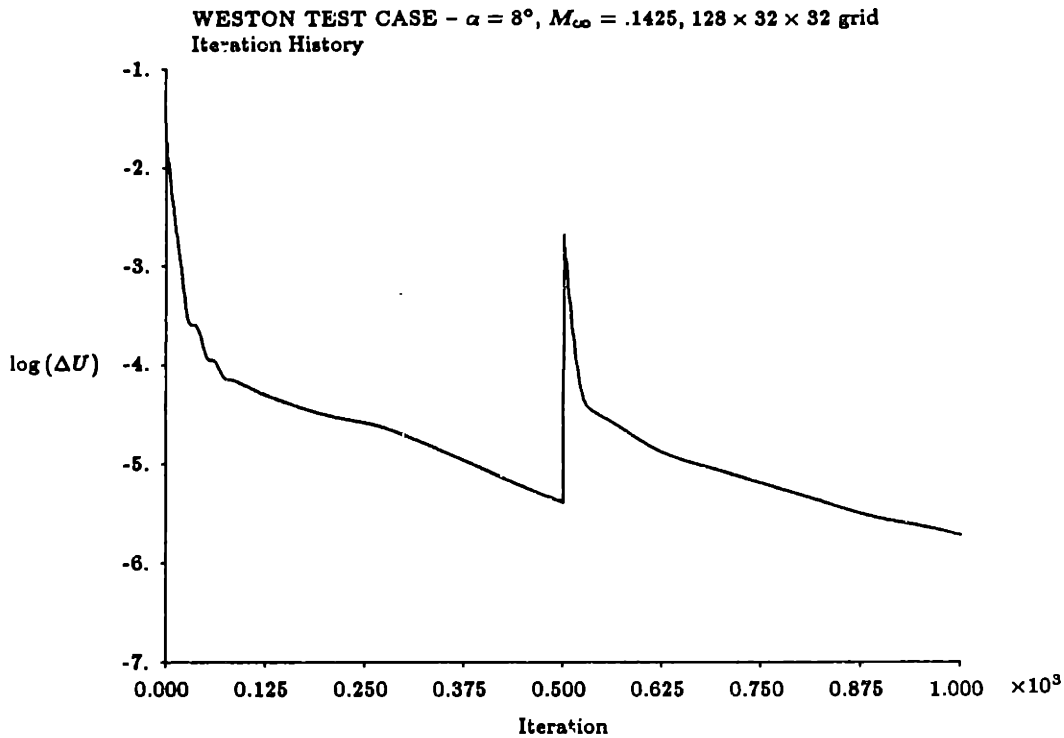

Figure 2.9: Convergence history for Weston wing, Eriksson's grid

location. At each section, the computed pressure at the trailing edge is seen to show more recompression than was observed experimentally, due to the growth of the boundary layer on the wing. However, in moving along the span to the tip, the lift coefficient at each section is seen to fall off more rapidly than was observed experimentally. Again, the considerable flow angularity in the tunnel can account for the discrepancy. Near the tip itself, the computed suction peak at the leading edge is much lower than was observed experimentally. At the leading edge stagnation point near the tip, there is a lower pressure than in the experiment. Similar behavior is seen in the ONERA M6 results at the 99% span section (Figure 2.7). Solutions

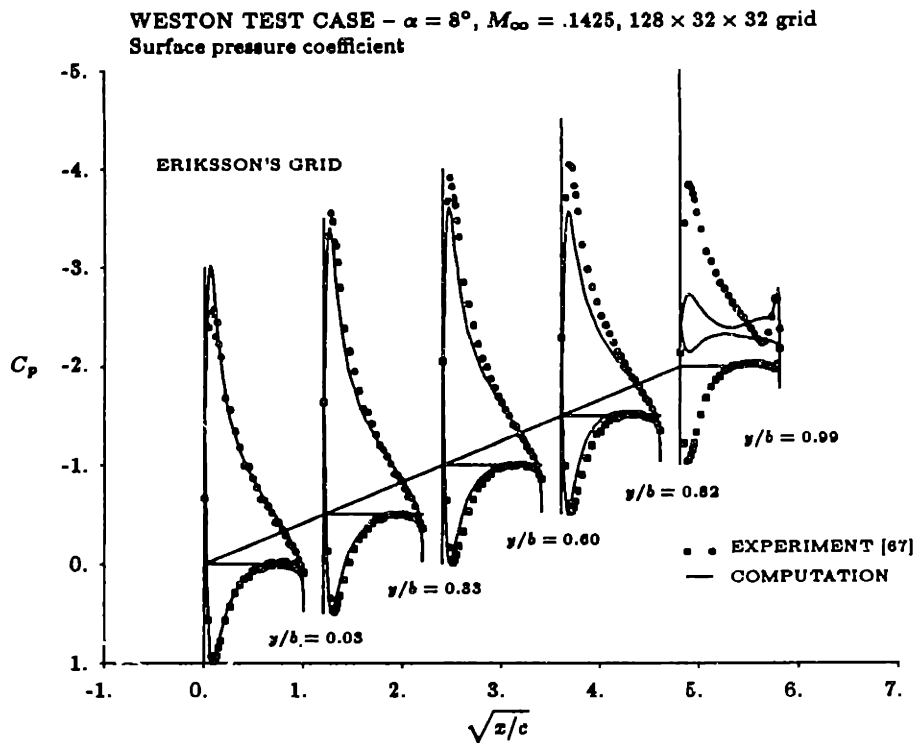

Figure 2.10: Computed and experimental surface pressures, Weston wing, $M_\infty = 0.1425$, $\alpha = 8°$, Eriksson's grid

on a coarser grid also show very similar behavior. One possible cause for this behavior is the difference in the tip geometry between the computation an experiment. The experimental wing model had a body of revolution tip (both for the ONERA M6 and for the Weston wing), while the compuational geometry is rounded but not a body of revolution. Also, the grid singularity intersects the wing at the tip leading edge, and locally the solution accuracy is degraded.

At the tip of the wing ($y/b = 0.99$, Figure 2.10) the pressure coefficients show a secondary suction peak at the trailing edge, qualitatively similar

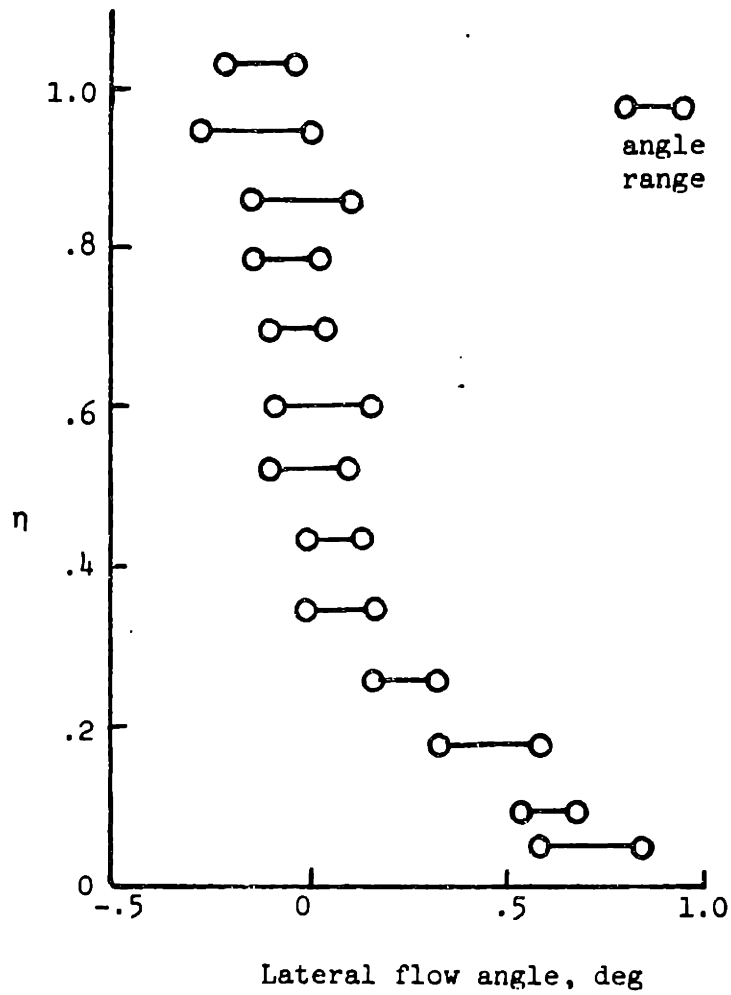

Figure 2.11: Flow angularity in empty wind tunnel

to the experimental observations. The cause of this localized reduction in pressure appears to be due to the formation of the tip vortex as it rolls up over the wing tip. Although the tip is rounded, there is a separation occurring as at the sharp trailing edge, and the flow is qualitatively behaving like the real, viscous flow. The significant quantitative differences in the load distribution do show that the behavior, although mimicking the physical process, is not a reliable model of the physics.

One difficulty with the grid generator used to create the grid for this case is that the body of revolution tip geometry cannot be accurately modeled. Eriksson's grid generator will round off the tip, but does not provide exactly a body of revolution shape. Furthermore, his treatment of the grid singular lines as they come off the tip places certain restrictions on the degree of freedom allowed in specifying the tip shape geometry without getting too highly skewed a grid. For this reason, computations have also been performed for this case on a grid provided by Wedan [66]. This grid was generated by a hybrid algebraic/elliptic PDE procedure. The two surface grids are shown in Figures 2.12 and 2.13. Wedan's grid has the same number of cells as the Eriksson grid, but with a different distribution of grid points. It also accurately models the body of revolution wing tip (Figure 2.13). Solutions obtained using this grid are shown in Figure 2.14. Because with Wedan's grid the chordwise coordinate planes on the wing surface do not lie along the streamwise direction as they do with Eriksson's grid, the pressure coefficients were interpolated to the experimental spanwise stations. Note the similar results as for the original grid over the inboard portion of the grid, but the differences at the tip. With this grid, pressure distribution at the 99% semispan location more nearly matches the experimental data qualitatively, although the suction peaks at the leading and trailing edges are still lower than experiment. The differences are likely not only to the differences in the tip shape, but in the differences in the spanwise resolution at the tip,

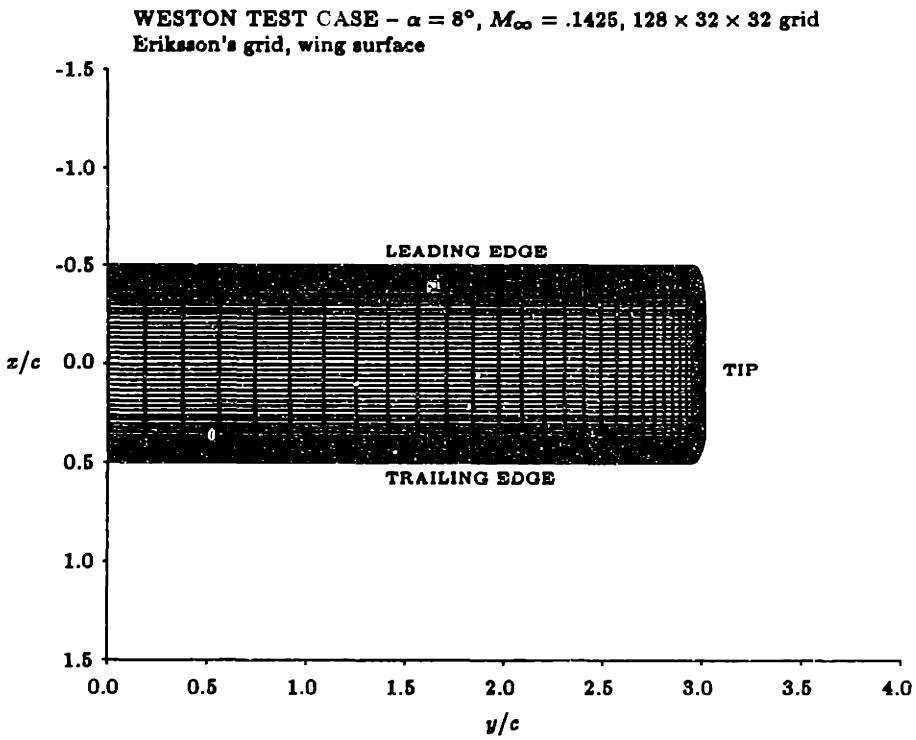

Figure 2.12: Wing surface grid, Eriksson's grid generator

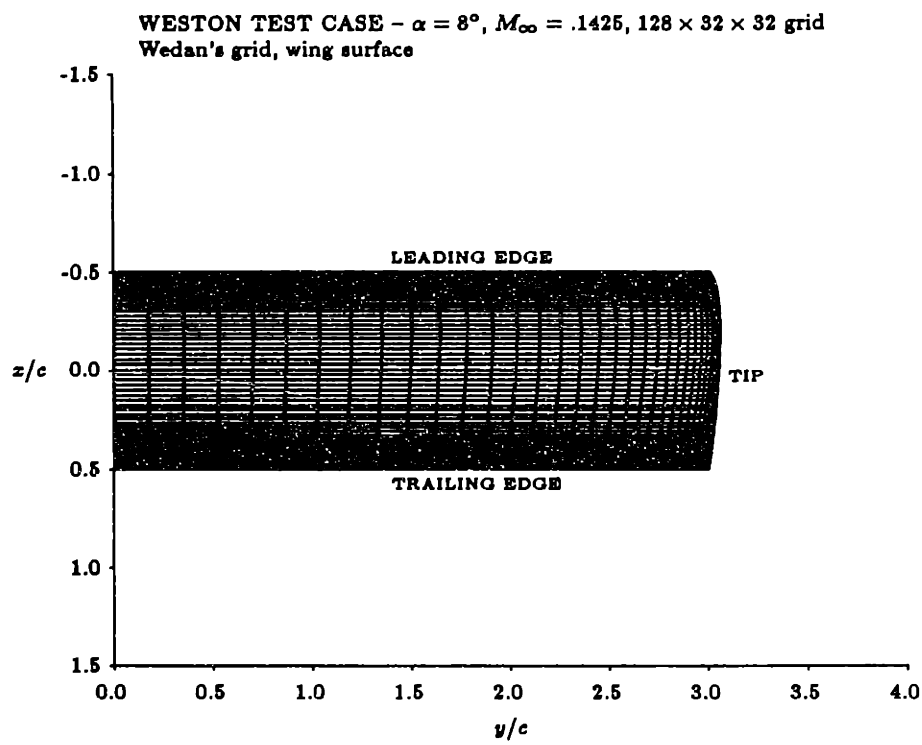

Figure 2.13: Wing surface grid, Wedan's grid generator

which is slightly coarser with Wedan's grid. Clearly, the local behavior of the flow is sensitive to the details of the grid and the tip geometry.

Although the Euler solutions on the two grids are significantly different in detail near the wing tip, both have quite similar surface pressure distributions inboard of the tip and both are consistently underpredicting the experimental suction peaks over most of the wing. As stated above, a likely culprit for this state of affairs is the flow angularity in the wind tunnel. To get a better understanding of this discrepancy, a comparison has also been made between both Euler solutions and the results of a surface singularity potential code (panel method). Since the free stream Mach number is so

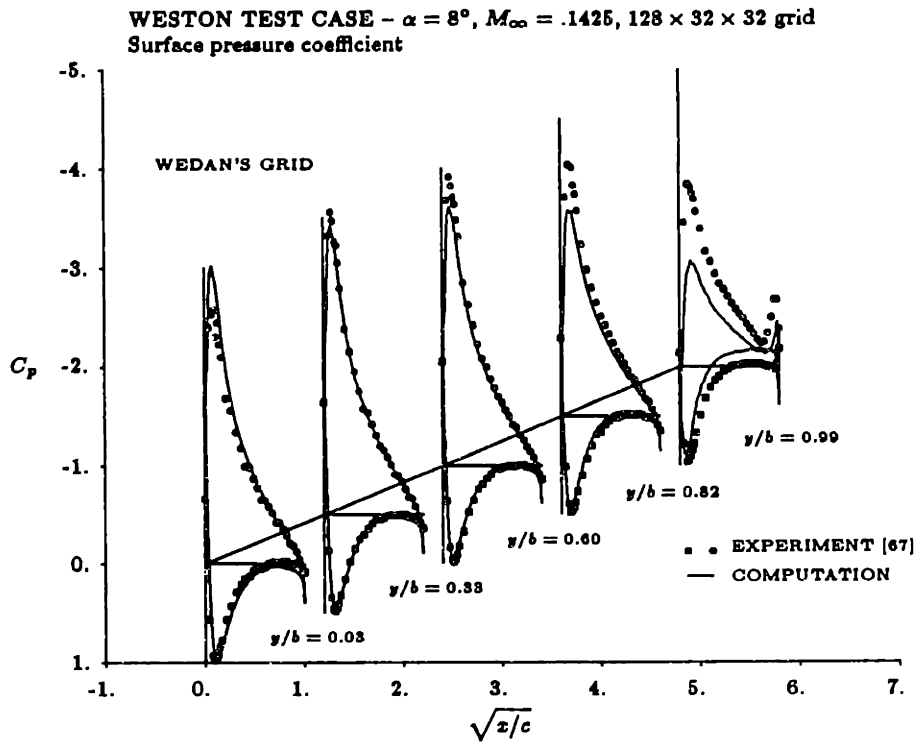

Figure 2.14: Computed and experimental surface pressures, Weston wing, $M_\infty = 0.1425$, $\alpha = 8°$, Wedan's grid

low and the flow is attached over most of the wing, the Euler solution and the potential solution should give very similar results. The panel code used is QUADPAN (Coppersmith, Youngren, & Bouchard [19]), which is a production code used by Lockheed. This code uses quadrilateral panels, with constant source and constant doublet strengths on each panel. A total of 975 panels on the wing surface were used to model the wing, and the wake panel were extended 100 chord lengths behind the wing.

Comparisons between the chordwise pressures obtained with the Euler solver on Eriksson's grid and with QUADPAN are shown in Figure 2.15.

Overall agreement is very good, although the Euler solution shows slightly

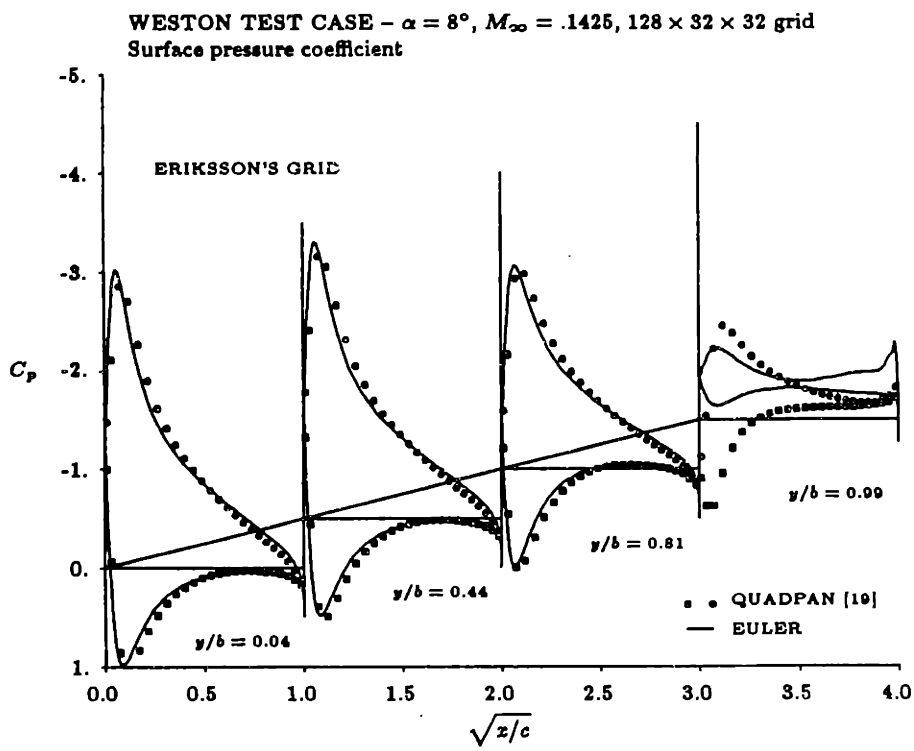

Figure 2.15: Comparison of Euler and QUADPAN solutions, Weston wing, Eriksson's grid

less lift forward of 30% chord and more aft than the panel code. This suggests a slight difference in the thickness distribution of the wing between the two solutions, although both are nominally the same. (In fact, exactly the same airfoil section coordinates were used as input for Eriksson's grid generator and for the panel code.) Significant differences at the tip are visible, as to be expected, since the Euler solution is showing separation around the tip and the panel code has a fixed planar trailing vortex wake that is emitted only along the wing trailing edge. However, the suction

peaks match quite well over the inboard sections.

Surface pressures for the Euler solution on Wedan's grid and the panel code are shown in Figure 2.16. The agreement is excellent over the wing.

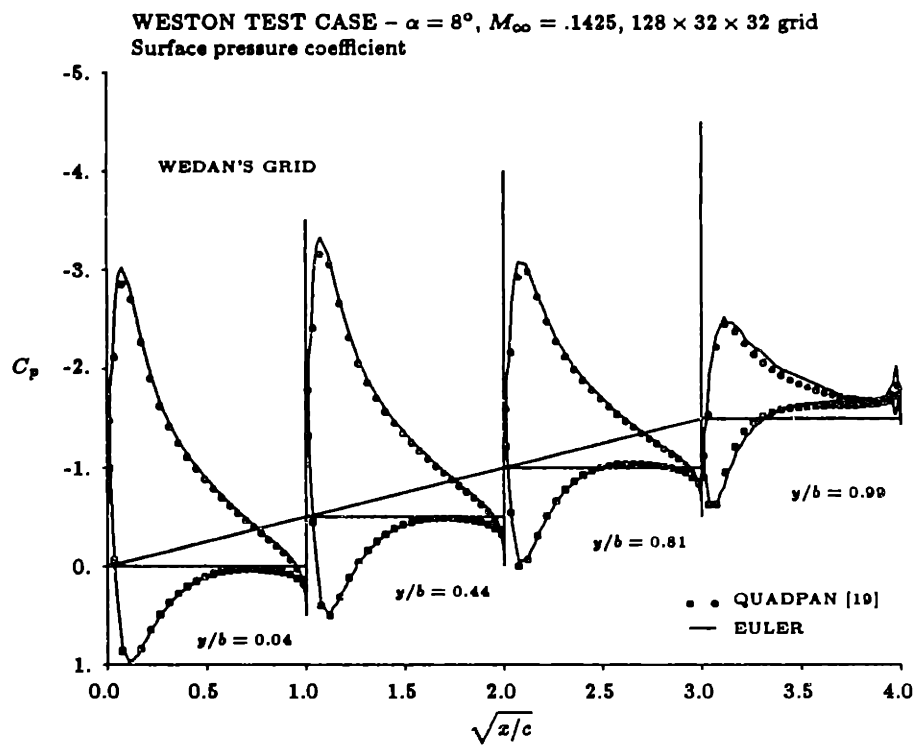

Figure 2.16: Comparison of Euler and QUADPAN solutions, Weston wing, Wedan's grid

Even at the tip, the surface pressures are in amazingly good agreement, especially considering that the tip shape in the panel code is rectangular, not a body of revolution. The agreement in this region may be somewhat fortuitous. As for Eriksson's grid, the suction peaks agree quite well between the Euler and potential solutions.

The agreement between the potential solution and both Euler solutions

is very encouraging, because the two models should be nearly identical for this particular flow. Also, the agreement inboard of the tip indicates that the details of flow near the tip computed by the Euler equations have only a local influence and are dominated by the details of the mesh and tip geometry in that region. Also, the agreement of the Euler and potential solutions further indicates that the discrepancy in the measured and computed surface pressures is due primarily to the flow angularity in the wind tunnel.

Because the flow at the tip is so different for the two Euler solutions on two different grids, but the differences have only a local influence on the loading on the wing, it is of interest to see how sensitive the computed wake structure is to the details at the tip. Also, comparisons with experimental wake surveys will also indicate how well the Euler equations model the physical wake structure.

Comparisons of the wake structure computed by the two Euler solutions, as well as experimental results, have been made at a location of approximately 1/2 chord downstream of the trailing edge. The grids at that station are shown in Figures 2.17 and 2.18. Note that Eriksson's grid provides much better resolution in the wake than Wedan's grid. Figures 2.19, 2.20, and 2.21 show computed and experimental contours of total pressure coefficient, defined as $(p_t - p_{t_\infty}) / \frac{1}{2}\rho u_\infty^2$. A well defined tip vortex is seen both computationally and experimentally. The two computed tip vortices are very similar in location and total pressure loss despite both the differences in the computed load distribution near the wingtip and the differences in the grid resolution in the wake. The solution on Eriksson's grid has a slightly greater total pressure loss in the core of the vortex and is slightly inboard of the vortex computed on Wedan's grid. The vortex position is based on the location of the total pressure minimum in both cases, and the difference is within a grid cell, so the discrepancy is not significant. Overall, the agreement between the two calculation is quite good, despite the differences in

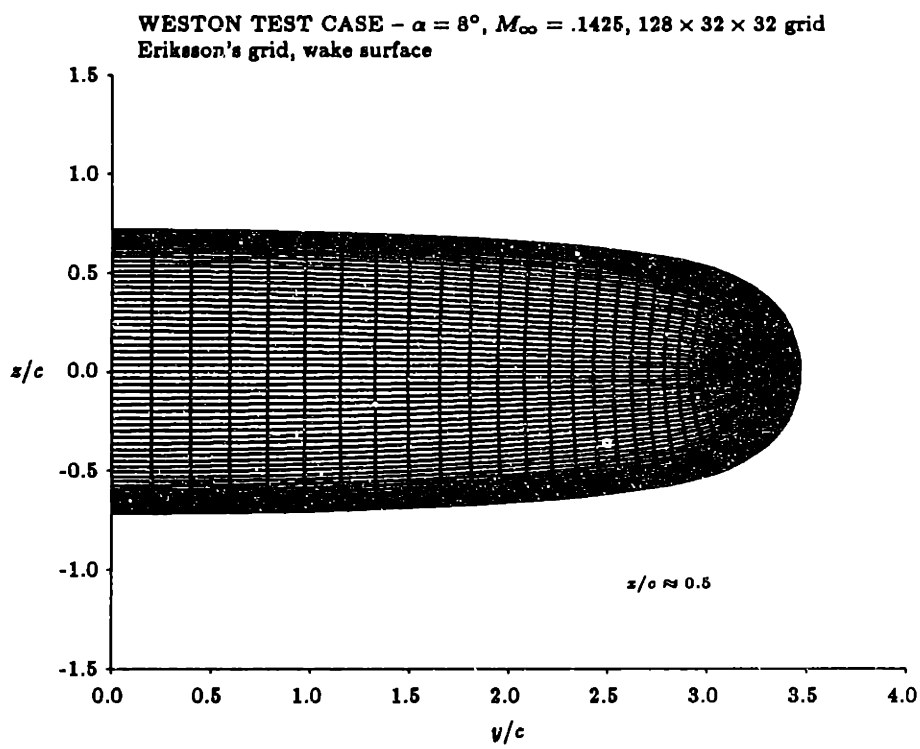

Figure 2.17: Grid surface $\approx 0.5c$ downstream of trailing edge, Eriksson's grid

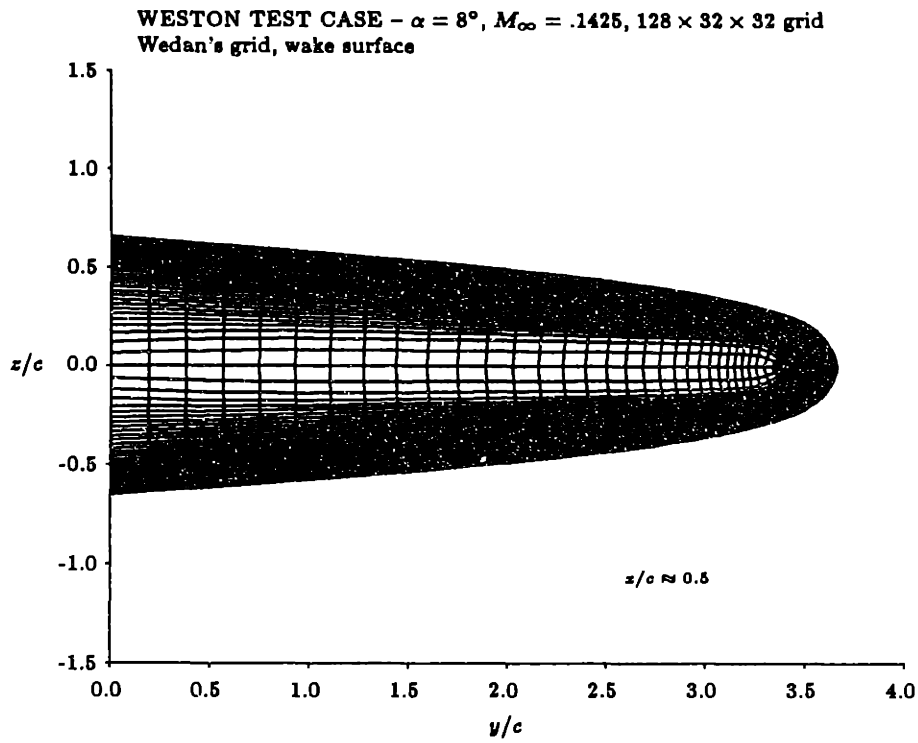

Figure 2.18: Grid surface $\approx 0.5c$ downstream of trailing edge, Wedan's grid

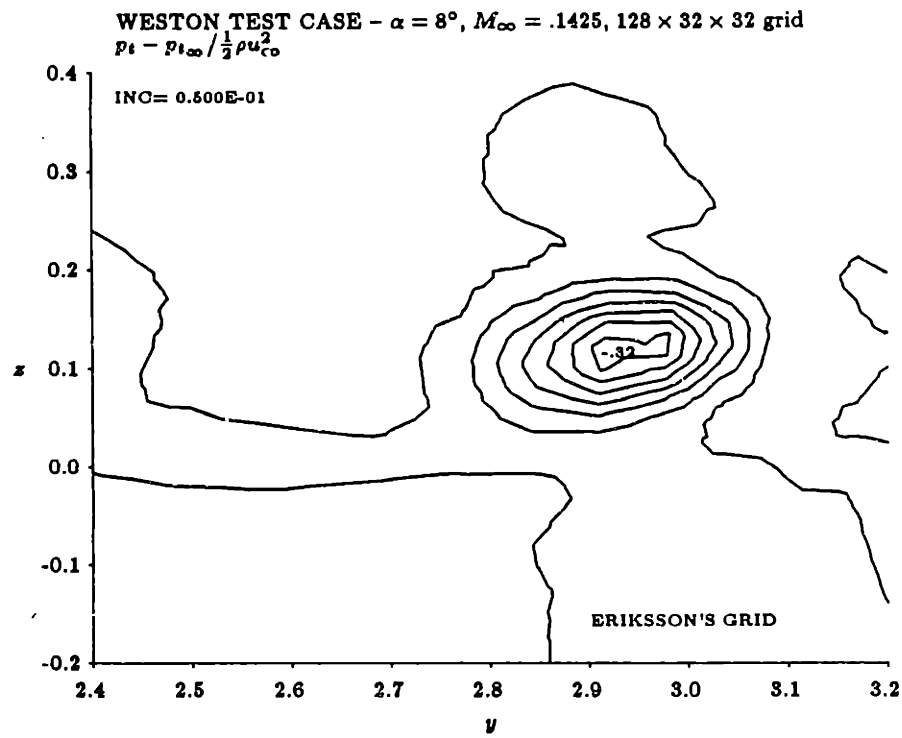

Figure 2.19: Total pressure coefficient, $x/c \approx 0.5$, Eriksson's grid

the geometric modeling of the tip, the computed load distributions at the tip, and the differences in the grid resolution in the wake. This suggests that the initial stage of the wake roll up computed by the Euler equations is not sensitive to the local flow details in the region where separation occurs, nor to detail differences in the grid resolution.

Although each computatation agrees well with the other, the level of total pressure loss is lower in the calculations than in the experiment. (Compare the results in Figures 2.19 and 2.20 to the experimental values in 2.21.) Furthermore, no total pressure loss can be seen in the inboard portion of the vortex sheet in the computation, in contrast to the experiment. The position

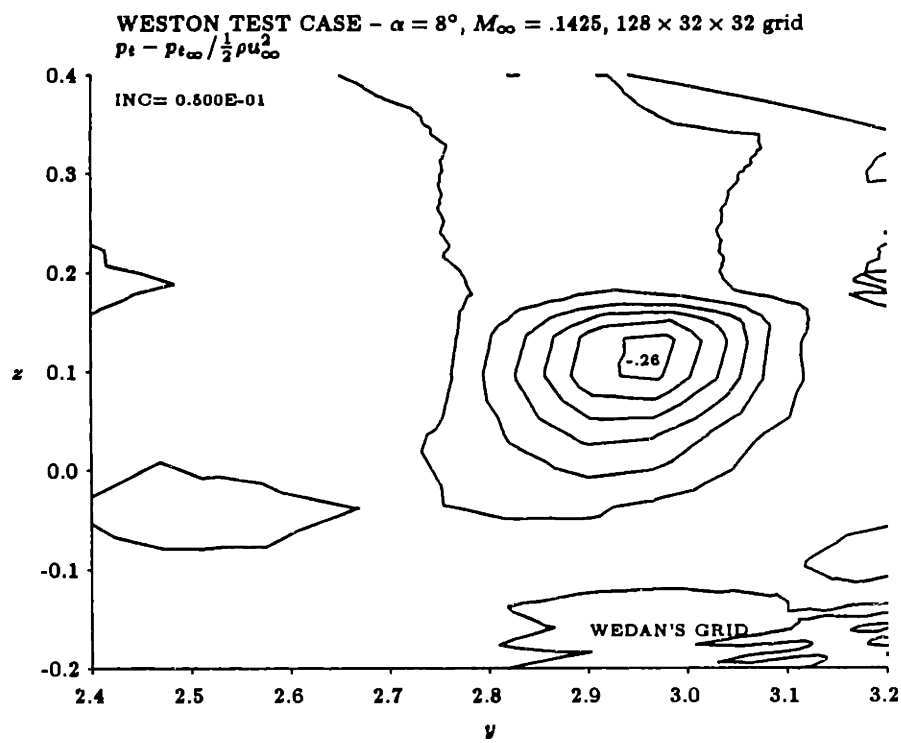

Figure 2.20: Total pressure coefficient, $x/c \approx 0.5$, Wedan's grid

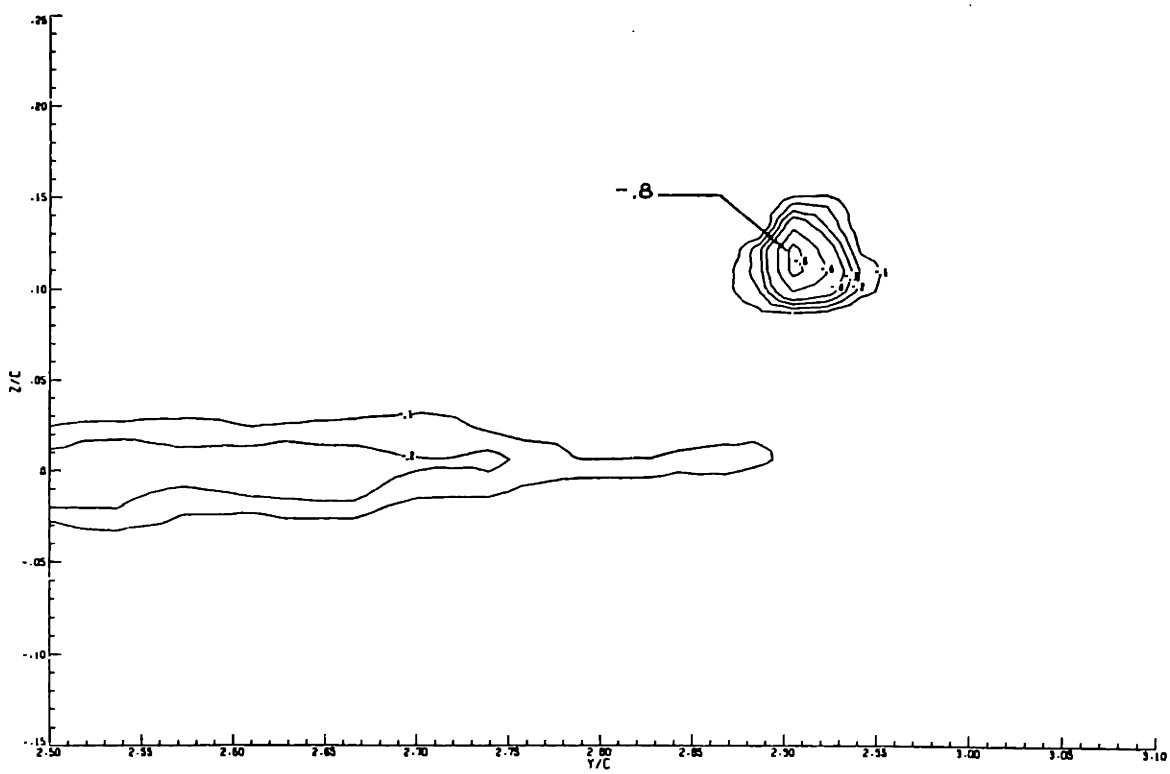

Figure 2.21: Total pressure coefficient, $x/c = 0.5$, experiment

of the computed tip vortices agree well with experiment, as determined by the location of the total pressure minimum. The discrepancy in the level of total pressure loss between the computations and experiment is most likely due to the neglect of viscosity.

Static pressures are shown in Figures 2.22, 2.23, and 2.24 as contours of constant pressure coefficient. The computed pressures in the core of

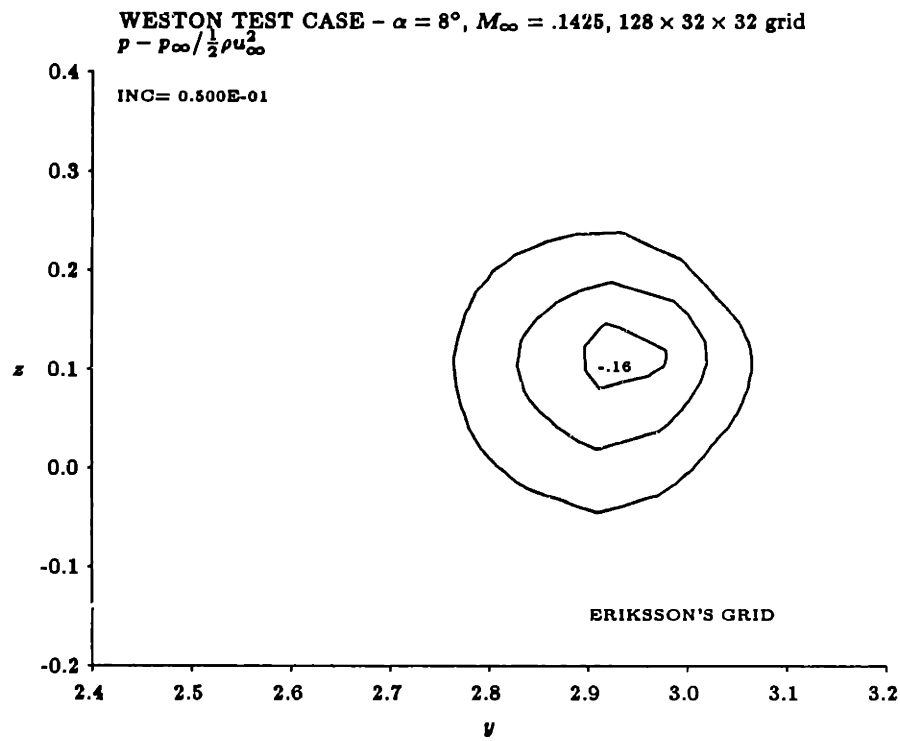

Figure 2.22: Pressure coefficient, $x/c \approx 0.5$, Eriksson's grid

the vortex are higher, and more uniform, than was observed experimentally. The computed pressure coefficients differ from the experimental values by an order of magnitude, although once again the two computations are very similar. This is further evidence that the details at the tip do not have

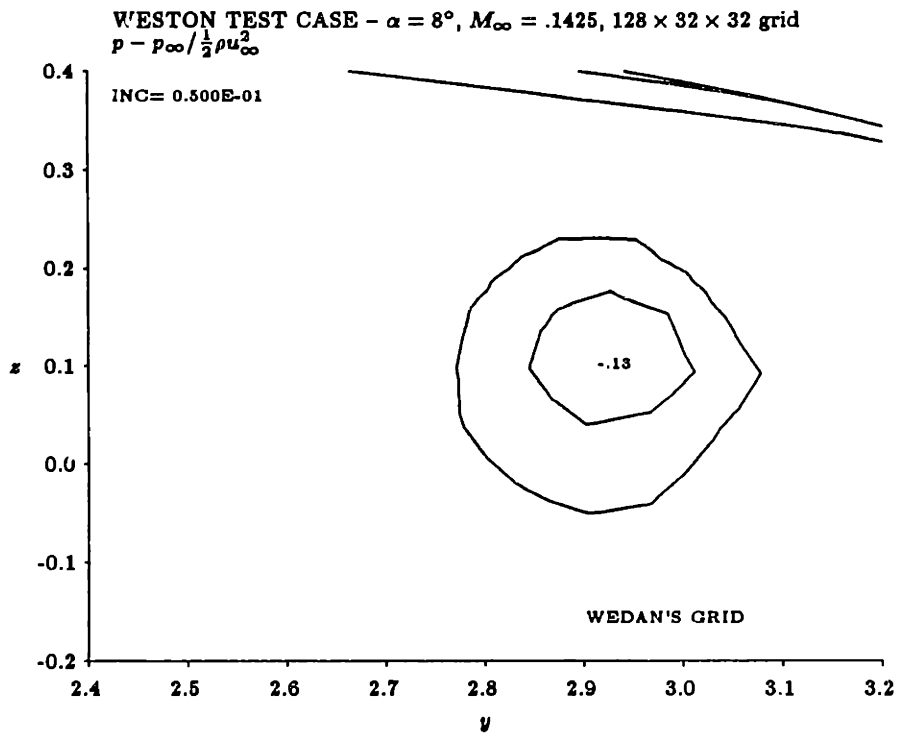

Figure 2.23: Pressure coefficient, $x/c \approx 0.5$, Wedan's grid

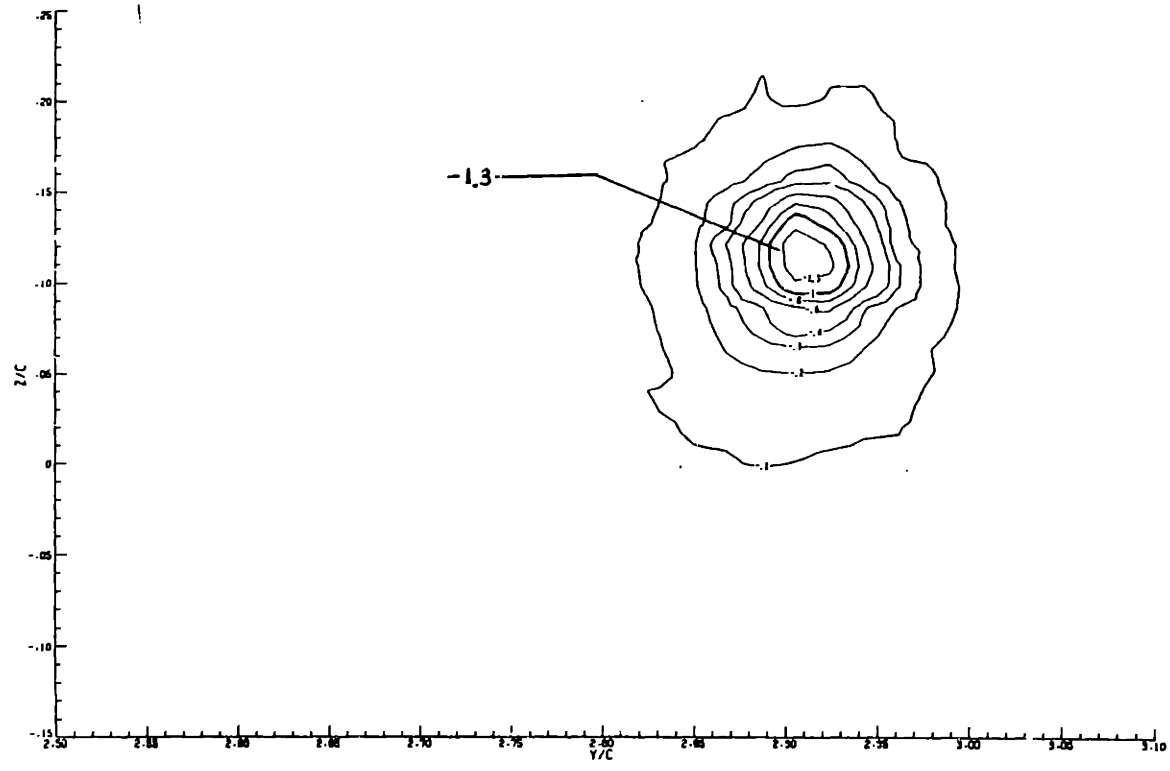

Figure 2.24: Pressure coefficient, $x/c = 0.5$, experiment

a strong influence on the structure of the wake computed with the Euler solver. The quantitative comparison with the measured pressure in the wake is poor, however.

The third wake quantity which was compared was the axial velocity in the vortex core, which are shown in Figures 2.25, 2.26, and 2.27. The results

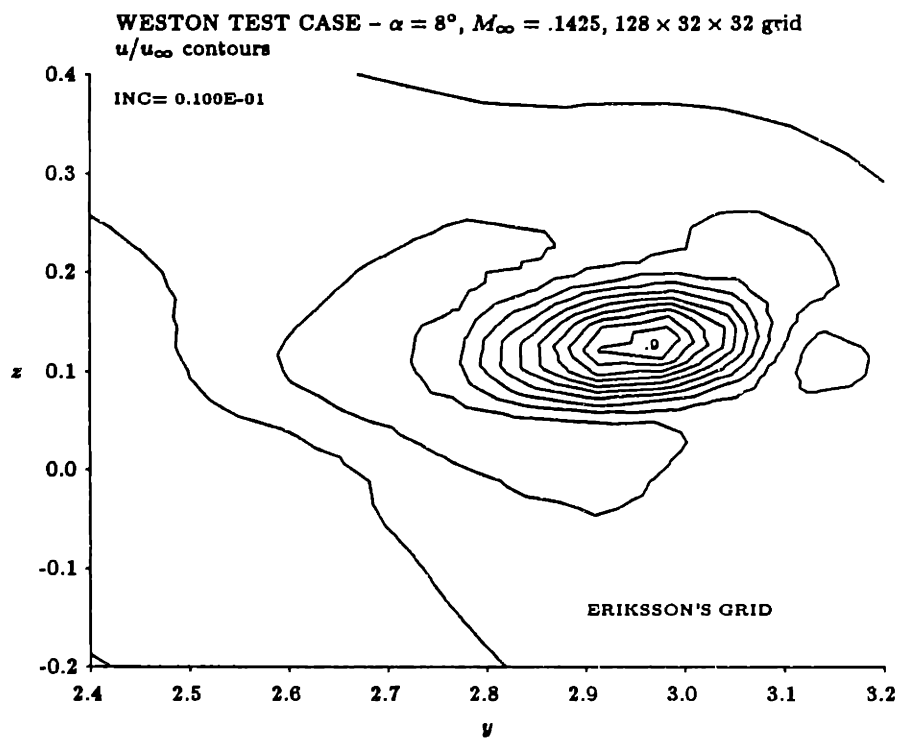

Figure 2.25: Axial velocity, u/u_∞, $x/c \approx 0.5$, Eriksson's grid

are shown as contours of constant axial velocity normalized by the free stream velocity. As with the total and static pressure, the computed results obtained on the two different grids agree well with each other, and do not agree well with the measured values. Note that the computed results show an axial velocity deficit in the core of approximately 10% of the free stream

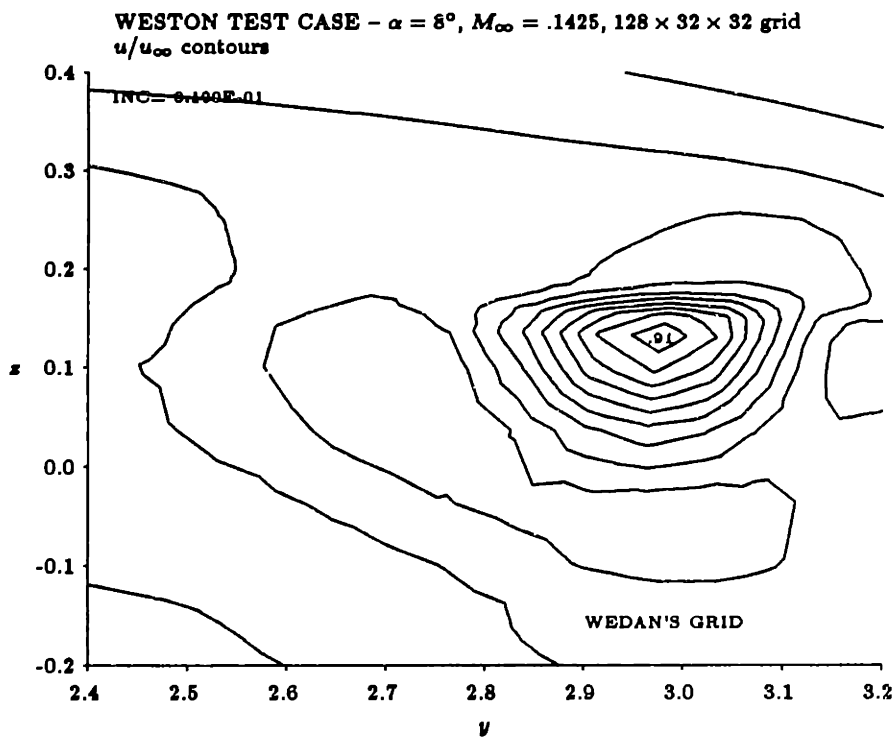

Figure 2.26: Axial velocity, u/u_∞, $x/c \approx 0.5$, Wedan's grid

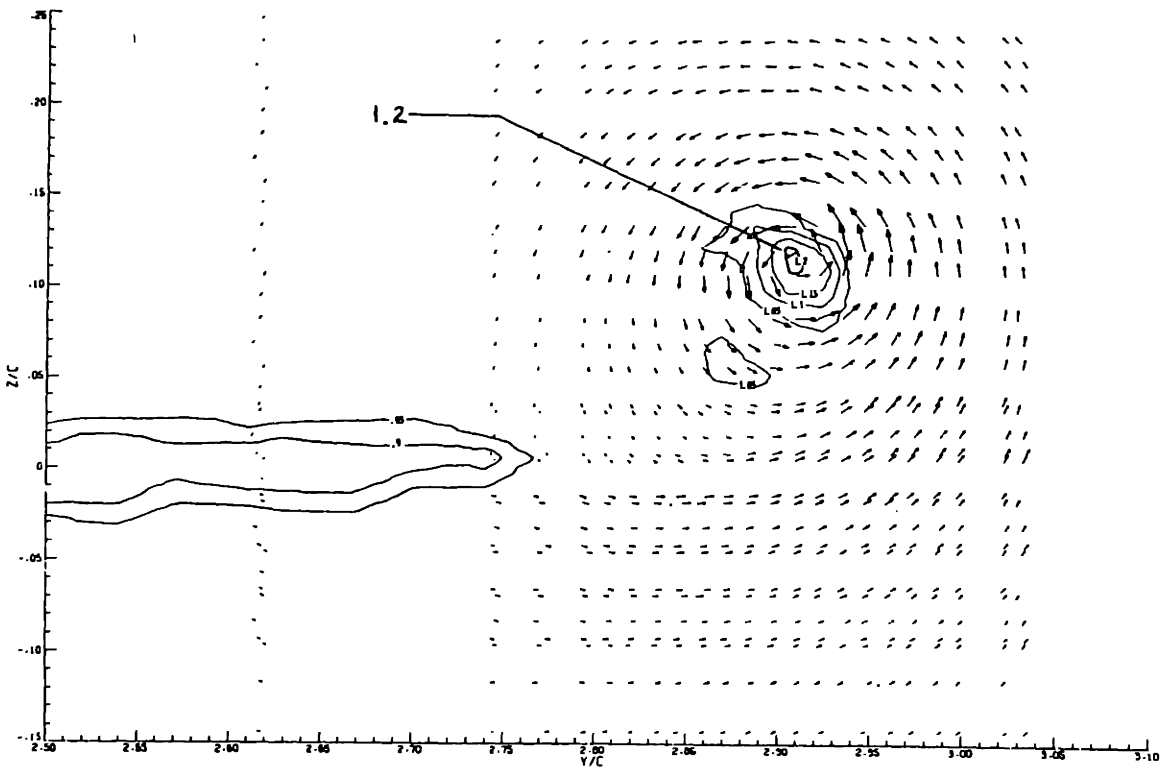

Figure 2.27: Axial velocity, u/u_∞, $x/c = 0.5$, experiment

velocity, in contrast to the 20% velocity excess observed experimentally. The reasons for this are not quite clear. According to Brown [10], whether an axial velocity deficit or excess is seen depends upon the ratio of induced to profile drag, with a deficit occurring when the viscous drag is high. By this argument, an inviscid calculation should show a velocity excess, and it should be greater than the experimental value. Weston [67] also discusses the experimental axial velocity excess based on the theory of Batchelor [8]. Again, the theory states that the axial velocity in the core is generated in a primarily inviscid manner, and for this case should be an excess velocity. The fact that exactly the opposite is seen here suggests that the mechanism by which the tip vortex is formed in the Euler code is considerably different than the physical mechanism.

For the calculations shown here, the Euler equation solutions are seen to be a poor model for the structure of the wake. This is in contrast to the results for leading edge vortex flows computed by Powell et al. [50]. In reference [50], the argument is presented that discrete solutions of the Euler equations should be a realistic model for such flows, in particular for the core of the vortex. Their argument is a kinematic one, namely, the velocity must pass through a minimum in the center of a discrete sheet. If the total enthalpy is constant, and the dynamic boundary condition of no pressure jump across the sheet holds, then there must be a total pressure loss whose base level is set only by the strength of the sheet. The artificial viscosity of the computation, and the physical viscosity of the real flow, affect the total pressure loss only to a higher order.

For the flow shown here, this argument does not hold. Unlike the leading edge vortex flows, the vortex sheets here are much weaker. The low Mach number of the current computation ($M_\infty = 0.1425$) compared to the supersonic leading edge vortex flows means the base total pressure loss here will be much less than for the leading edge vortex. The momentum deficit due

to the viscous boundary layer will produce a total pressure loss of the same order of magnitude as the jump in velocity across the boundary layer. Note that, for the inboard portion of the sheet, the experimental total pressure loss contours virtually overlay the u/u_∞ contours (Figures 2.21 and 2.27). Also, since the wake here is a free vortex sheet, and hence rolls up as it is convected downstream, the streamwise diffusion due to the artificial viscosity will have a significant effect on the development of the wake. To see this, the computed and measured total pressure coefficients two chords behind the wing are shown in Figures 2.28, 2.29, and 2.30. Note that the

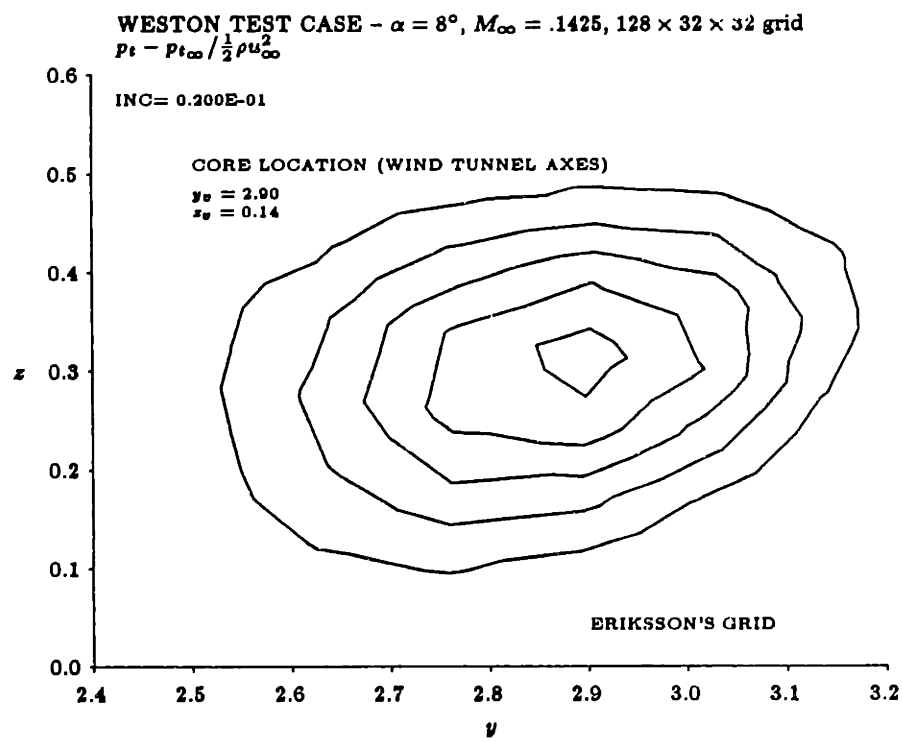

Figure 2.28: Total pressure coefficient, $x/c \approx 2$, Eriksson's grid

experimental core is still quite compact, and in fact its structure is not very

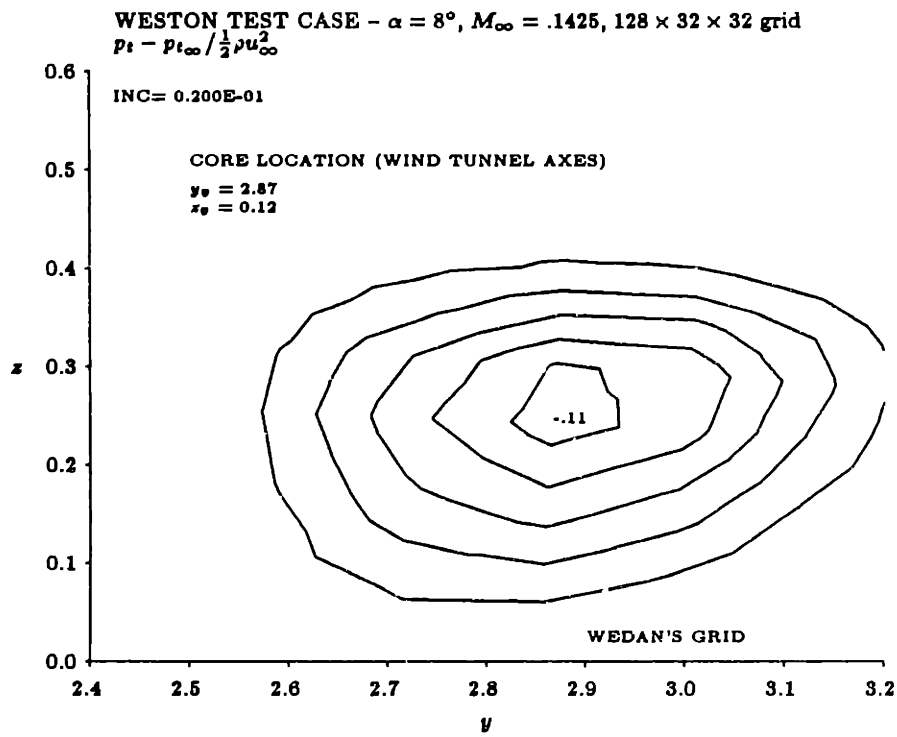

Figure 2.29: Total pressure coefficient, $x/c \approx 2$, Wedan's grid

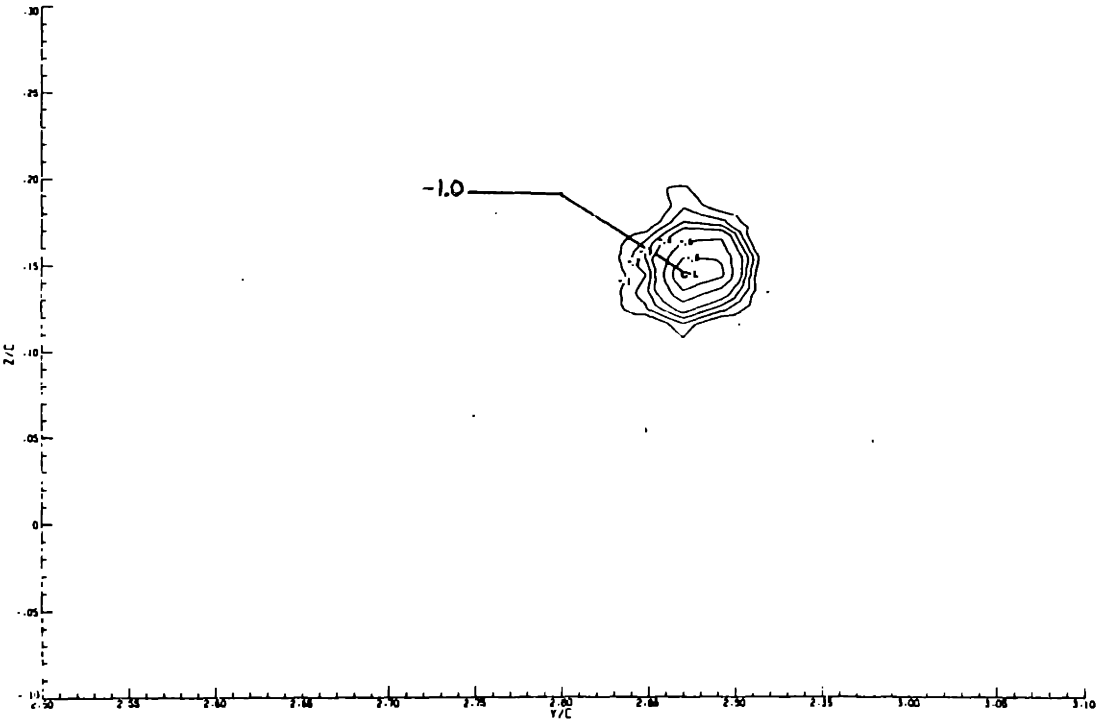

Figure 2.30: Total pressure coefficient, $x/c = 2$, experiment

different from its structure at one half chord behind the trailing edge. The same cannot be said for the computed cores, which are much more diffuse than they were upstream. The grids are much coarser at this location than upstream. This effect does not appear in a conical flow computation, in which the dimensionality of the problem has been reduced by one. Even in fully three dimensional calculations of a leading edge vortex flows, which have also been observed to provide remarkably accurate values of total pressure loss, the conicality or approximate conicality of the flow and the large vortex strengths mean that the results will be similar to a conical solution.

2.10 Summary

The algorithm for solving the Euler equations has been verified for two test cases. The agreement with experimental pressure distributions for both the ONERA M6 wing at transonic speed and the plain rectangular wing tested by Weston at low Mach number is good. Discrepancies in the spanwise load distribution between computations and experiment in the latter case appear to be due to flow angularity in the wind tunnel. The Euler solutions for Weston's wing have been computed using two different grids with different tip geometries and differing resolution in the wake. The load distributions at the tip were seen to be sensitive to the differences in the grids and tip shapes, but the overall location and structure of the tip vortex were relatively insensitive to these local features.

Comparisons of the experimental and computational wake structures were mixed. The location of the vortex core was well predicted, but the details of the structure were very different. Differences in the level of total pressure loss in the core appear to be due to the neglect of viscous effects. More puzzling is the existence of an axial velocity excess observed experimentally in the core of the vortex, in contrast to a computed axial velocity deficit. This suggests that the numerical process which the causes the flow

to separate, and the initial stages of the tip vortex formation computed by the discrete Euler equations, are very different from the physical process in a real flow.

Chapter 3

Perturbation Scheme

In this chapter, a method is developed for computing the interaction between a streamwise vortex and a wing. First, the problems of using the unmodified finite volume algorithm developed in the previous chapter for computing such flows are discussed. The motivation behind the perturbation, or prescribed flow, scheme for introducing a vortex into the finite volume computational domain is presented. The modifications of the basic finite volume solver required by the perturbation approach are then described. A very simple model problem, the steady flow of a streamwise vortex in a rectangular section channel, is presented to illustrate how the perturbation approach eliminates the numerical diffusion of vorticity. Finally, the perturbation scheme is validated against the experimental data of Smith & Lazzeroni [57].

3.1 Problems with Euler Solver

Since the primary topic of this thesis is rotary wing flows in which there is an interaction between a compact tip vortex and the rotor blade, it is necessary to examine the ability of the finite volume scheme to handle such flows. Rather than considering a hovering rotor directly, a simpler problem is used to study this issue. The problem examined is the interaction of a

streamwise vortex with a fixed wing, and is shown schematically in Figure 3.1. This flow configuration is also of practical interest in itself, since it corresponds to such situations as the interaction of the wake of a canard with a main wing, or the problem of a tip or leading edge vortex impinging on a tail surface. The interaction treated here is assumed to be steady, and can be described by the steady Euler equations.

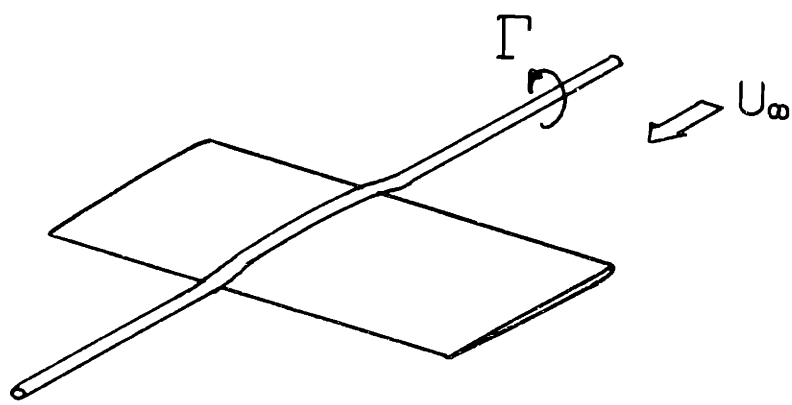

Figure 3.1: Schematic of wing/vortex interaction

One approach to computing such a flow would be to generate a grid around the wing and introduce the vortex through the upstream boundary conditions. In principle, there would appear to be no reason why this would not work. However, two difficulties arise if this simplistic approach is used. The first is that a very fine grid is required at the upstream far field boundary to resolve the vortex core. The second is that the artificial viscosity of the scheme will diffuse the vorticity before it reaches the lifting surface. Both these problems are discussed below.

To understand the first problem, it must be explained how a vortex is

introduced at the inflow boundary. Any vorticity can be introduced into the computational domain only through two of the inflow boundary conditions. The first is the specification of the tangential velocity at the far field. The swirl velocity associated with the vortical flow field results in a tangential velocity component different from the freestream. Secondly, the entropy is specified at an inflow boundary. From Crocco's theorem, we can expect that in the core of the vortex, where the flow is rotational, there will be a gradient of entropy (assuming homenthalpic flow; if there are total enthalpy gradients, the flow may be rotational yet isentropic). Conversely, by specifying a varying entropy and constant total enthalpy at the inflow boundary, a rotational flow must be introduced into the computational domain. The entropy and tangential velocity specification are seen to provide the only mechanism by which vorticity can be introduced into the domain.

A closer examination of the tangential velocity boundary condition shows that it does not give vorticity directly. Specification of the tangential velocity really gives the circulation, not the vorticity itself. This is because the circulation around a closed circuit is related to the vorticity by the integral

$$\Gamma = \iint_S \vec{\omega} \cdot \hat{n} d^2x, \tag{3.1}$$

where Γ is the circulation, $\vec{\omega}$ is the vorticity, and S is the surface bounded by the circuit. It is seen that for a given circulation around the circuit, the vorticity distribution is not uniquely defined. The entropy boundary condition will give vorticity, since for a homenthalpic flow, the distribution of entropy is related to the vorticity through Crocco's theorem. However, if the core of the vortex is more compact than the spacing of the grid at the far field, the structure of the core cannot be properly described. This is shown schematically in Figure 3.2. The size of the rotational core is shown to be of the order of the grid spacing. Clearly the entropy gradient in the core of the vortex cannot be resolved in the far field. As a result, the vor[illegible]tained

in the interior of the computational domain is much larger than the actual vortex. Although the grid resolution increases as the wing is approached, the equations contain no mechanism to cause the vortex core to shrink in the interior of the domain; that is, there is no production of entropy in the core to make up for the failure to adequately resolve the entropy in the far field. For the grids typically used in Euler computations, and the characteristically compact core of a lift generated vortex, this problem will generally exist for most computations.

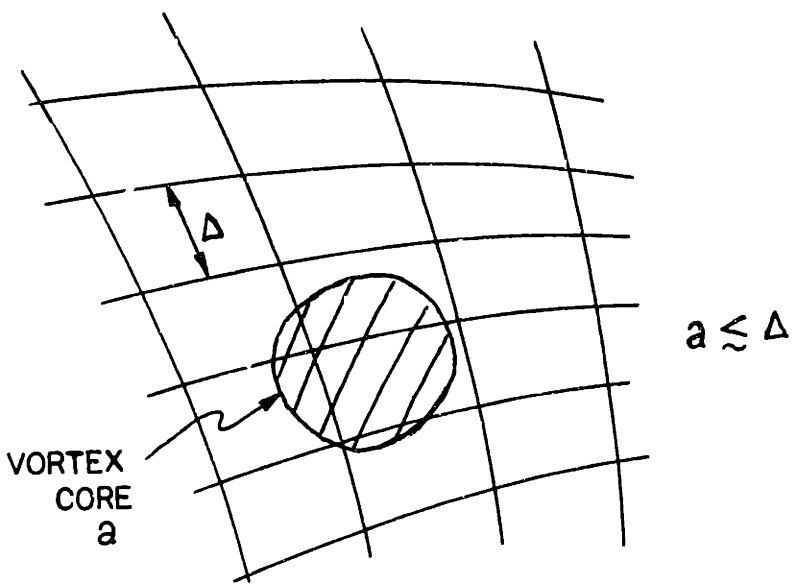

Figure 3.2: Schematic of vortex core size vs. far field grid resolution

The second problem referred to was the effect of the artificial viscosity required for numerical stability. This added diffusive term causes the vortex to be smeared as it is convected downstream, as was seen in the previous chapter. Since the artificial viscosity term is proportional to some power of the grid spacing ($[\Delta x]^3$ here for the fourth difference dissipation), the

numerical diffusion is greatest where the grid is the coarsest. Thus at the far field inflow boundary, where there is generally insufficient resolution to accurately introduce the vortex into the domain, the artificial viscosity aggravates the problem even more. It is seen that a sufficiently fine grid is required not only to resolve the vortex, but to reduce the level of numerical diffusion of vorticity.

It is a difficult task to provide enough resolution to allow the detailed structure of the vortex to be resolved in the far field, and hence allow the proper vortex to be introduced into the domain. First of all, if the grid resolution is increased globally so that a logically cubical computational domain can be maintained, the convergence of the scheme will suffer drastically. The work per iteration scales as the number of grid cells N. The allowable time step of the scheme scales as Δx, which is a typical length scale of a cell. This scales as $N^{\frac{1}{3}}$, meaning the number of iterations required to reach a steady state goes as $N^{\frac{1}{3}}$. Thus the work required to reach the steady state roughly scales as $N^{\frac{4}{3}}$. If, for example, the grid resolution is doubled in each coordinate direction, the work required to reach a steady state is increased by roughly a factor of 16, and the storage required increases 8 times. It is easily seen that to provide a fine enough global grid to be able to resolve an incoming vortex at the far field the scheme will become prohibitively expensive. Not only that, but global grid refinement provides excessive resolution in regions where it is not necessary. This is clearly not a viable solution to the problem.

An alternative way of providing resolution of the vortex core is to use a local refinement of the grid. This is probably best done by using either grid embedding, or patched or overlaid grids along the vortex path. This means that a logically cubical grid structure cannot be maintained, and a pointer system is required to provide the connectivity information of the grid. What is gained by this approach is the economy of grid points, because

resolution is provided only where needed. Care is required at grid interfaces to maintain spatial accuracy and conservation. The effect of the embedding on the quality of the solutions is an important issue. Further difficulties arise if the embedding is to be done adaptively, which is certainly a conceptually attractive idea since the vortex path is not known a priori. A good adaptive grid strategy requires both an effective way of finding the vortex, and a way of refining the grid during the course of the computation that both resolves the vortex and is efficient. Developing such an approach is a far from trivial task, and adds considerably to the complexity of the scheme.

For these reasons, an alternative approach is used here in which the vortex location and structure may be specified, without the need to provide a grid capable of resolving the rapid flow gradients. This approach is based on the idea proposed by Buning & Steger [11] and subsequently used by Chow et al. [16], and Srinivasan and his co-workers [58,59,60]. This method, called the perturbation or prescribed flow scheme, is described in the next section.

It should also be noted before continuing that the discussion in this section on the difficulties of resolving a vortex being convected through the computational domain is not limited to the particular algorithm of this thesis. The comments apply to any method used to solve the Euler equations in an Eulerian frame of reference. Navier-Stokes schemes will encounter the same problems as well. The perturbation scheme that is developed in this chapter is also very general in its application. Although the details of implementing the scheme will depend upon the algorithm used, the basic features of the approach are independent of the Euler solver.

3.2 Perturbation Scheme

The perturbation scheme used here was introduced by Buning & Steger [11] as a generalization of freestream subtraction for the Euler equations.

They used it to compute flows having a non-uniform free stream without the need for excessive grid resolution in the far field. More recently, Srinivasan et al. [58,59,60] have used this approach for computing the unsteady interaction of a vortex with an airfoil. They have demonstrated that the approach allows the vortex to remain well defined and compact, even in coarse regions of the grid. The present investigation implements the scheme differently than Srinivasan and his co-workers, since the flows of interest here are steady. In this respect, the current work is more closely related to the work of Chow et al. [16], who solved the Euler equations for the steady, two dimensional flow around an airfoil using the Buning & Steger perturbation approach. The prescribed flow solutions for their calculations were found using a finite difference full potential equation to reduce the resolution needed for the Euler solver. The fundamental ideas of the current work are described below. To clarify the basis of the approach, the details of the implementation will be left to the next section.

The basis of the perturbation method is that over some region of the flow field of interest the local behavior is similar to that of a simpler flow which may be described analytically. For the case of the interaction of a streamwise vortex with a wing (Figure 3.1), the flow near the vortex core behaves as if it were an isolated vortex, the influence of the wing being weak in that region. In other words, the flow field is dominated locally by the velocity field associated with the vortex. Furthermore, the flow field of an isolated vortex that satisfies the steady Euler equations can be readily found. By computing this vortex induced flow and subtracting it from the discrete approximation to the Euler equations, the need to provide a grid capable of resolving the rapid flow field gradients in the vicinity of the vortex core is eliminated.

To be more specific about how the scheme works, consider the compress-

ible Euler equations (2.5)

$$\frac{\partial}{\partial t}\iiint_V \mathrm{U}\, d^3x + \iint_{\partial V} \vec{\mathbf{F}}(\mathrm{U}) \cdot \hat{\mathbf{n}}\, d^2x = 0. \tag{3.2}$$

The spatial discretization of this equation was described in the previous chapter. It consists of the approximation to the flux integral and the addition of the artificial viscosity terms. The semi-discrete equations may be written as

$$\frac{d\mathbf{U}_{i,j,k}}{dt} = -\frac{1}{V}\mathbf{R}_{i,j,k} \tag{3.3}$$

where the residual $\mathbf{R}$ is defined as

$$\mathbf{R}_{i,j,k} = F_{i,j,k} - D_{i,j,k}, \tag{3.4}$$

$F_{i,j,k}$ being the finite volume flux integral approximation and $D_{i,j,k}$ being the artificial viscosity operator for cell (i,j,k). With the basic finite volume scheme, $\mathbf{R}$ is driven to zero by the pseudo-time integration.

From the induced velocity field of an isolated vortex, a state vector $\mathbf{U}_0 = (\rho_0, \rho_0 u_0, \rho_0 v_0, \rho_0 w_0, \rho_0 E_0)^T$ that satisfies, or approximately satisfies, the steady Euler equations can be readily computed. Subtracting the flux integral associated with this state vector from Equation (3.2) yields

$$\frac{\partial}{\partial t}\iiint_V \mathbf{U}\, d^3x + \iint_{\partial V} \left\{\vec{\mathbf{F}}(\mathbf{U}) - \vec{\mathbf{F}}(\mathbf{U}_0)\right\} \cdot \hat{\mathbf{n}}\, d^2x = 0. \tag{3.5}$$

Since the second surface integral is zero, the equation is unchanged analytically. However, $\mathbf{U}_0$ will not necessarily satisfy the discrete equations due to the truncation error and artificial viscosity. In the limit of vanishing grid spacing, these terms will vanish. However, for a finite grid resolution these terms in general will be non-zero. The magnitude of these terms depends upon the gradients of the flow field as well as the resolution of the grid. By applying the discrete spatial operator (Equation (3.4)) to the flow field $\mathbf{U}_0$, a set of residuals $\mathbf{R}_0$ are found at each cell. These residuals are subtracted

from the residuals $\mathbf{R}$ associated with the complete state vector $\mathbf{U}$. The discrete Euler equations to be solved are

$$\frac{d\mathbf{U}_{i,j,k}}{dt} = -\frac{1}{V}(\mathbf{R} - \mathbf{R}_0)_{i,j,k}. \tag{3.6}$$

In solving this equation, the residuals $\mathbf{R}$ are no longer driven to zero, but are driven to $\mathbf{R}_0$. This allows the truncation error of the scheme to be approximately corrected in the region of the vortex. This is because $\mathbf{R}_0$ represents the truncation error of the scheme applied to the prescribed flow $\mathbf{U}_0$. Near the vortex core, $\mathbf{U}_0$ has large spatial gradients, and $\mathbf{R}_0$ takes on large values due to the difference between the discrete operator and the differential operator applied to $\mathbf{U}_0$ at that location. Since the gradients of the state vector $\mathbf{U}$ are assumed to show similar variations to $\mathbf{U}_0$ near the vortex, the residual $\mathbf{R}$ should show similar behavior as $\mathbf{R}_0$ if the differential operator is being statisfied. Away from the vortex location, the prescribed flow residuals will be small, since the flow field gradients are weak, and the solution in those regions will behave as if the standard finite volume scheme is being used there.

One important issue in using the perturbation scheme is the question of consistency. In Equation (3.5), the state vector $\mathbf{U}_0$ was assumed to satisfy the steady Euler equations. If this is the case, in the limit of vanishing grid spacing the residuals $\mathbf{R}_0$ will vanish, and the Euler equations will be recovered. Thus the scheme is consistent with the steady Euler equations if, and only if, the prescribed flow exactly satisfies the Euler equations. If $\mathbf{U}_0$ is not an exact solution to the steady Euler equations, the prescribed flow residuals $\mathbf{R}_0$ will remain non zero when the grid spacing vanishes, and the scheme is not consistent. In practice, it is possible to find a prescribed flow that exactly satisfies the steady Euler equations only in special circumstances. However, in the next section, a method of obtaining a prescribed flow solution that nearly satisfies the steady Euler equations is developed.

Equations (3.6) are integrated in time using the multistage algorithm

described in the last chapter, with the $\mathbf{R}_0$ terms being treated as source terms. The temporal integration is now of the form

$$\begin{aligned}
\mathbf{U}^{(0)} &= \mathbf{U}^n, \\
\mathbf{U}^{(1)} &= \mathbf{U}^{(0)} - \alpha_1 \frac{\Delta t}{V}\left(F^{(0)} - D^{(0)} - \mathbf{R}_0\right), && (3.7a) \\
\mathbf{U}^{(2)} &= \mathbf{U}^{(0)} - \alpha_2 \frac{\Delta t}{V}\left(F^{(1)} - D^{(0)} - \mathbf{R}_0\right), && (3.7b) \\
\mathbf{U}^{(3)} &= \mathbf{U}^{(0)} - \alpha_3 \frac{\Delta t}{V}\left(F^{(2)} - D^{(0)} - \mathbf{R}_0\right), && (3.7c) \\
\mathbf{U}^{(4)} &= \mathbf{U}^{(0)} - \alpha_4 \frac{\Delta t}{V}\left(F^{(3)} - D^{(0)} - \mathbf{R}_0\right), && (3.7d) \\
\mathbf{U}^{n+1} &= \mathbf{U}^{(4)}.
\end{aligned}$$

It should also be noted that except for the inclusion of the prescribed flow residuals $\mathbf{R}_0$, no changes in the spatial discretization, the artificial viscosity operator, or the enthalpy damping are required. The boundary conditions are slightly changed, in that the far field velocity now consists of a freestream plus the induced velocity of the prescribed vortical field, i.e.

$$\vec{u}_{ff} = \vec{u}_\infty + \vec{u}_0, \tag{3.8}$$

where $\vec{u}_{ff}$ is the velocity at the far field boundary, $\vec{u}_\infty$ is the undisturbed uniform freestream, and $\vec{u}_0$ is the prescribed flow velocity field at the far field boundary, which is taken for the cases here to be the induced velocity of the vortices making up the prescribed flow field. In the far field boundary conditions given in the last chapter, the incoming Riemann variable (Equation (2.25b)) is replaced by

$$r_\infty = \vec{u}_{ff} \cdot \hat{n} - \frac{2a_{ff}}{\gamma - 1}, \tag{3.9}$$

where a_{ff} is given by the equation

$$a_{ff} = \gamma\left(\frac{M_\infty^2}{2} + \frac{1}{\gamma - 1}\right) - \frac{\vec{u}_{ff} \cdot \vec{u}_{ff}}{2} \tag{3.10}$$

and the inflow tangential velocity specification (Equation (2.28a)) is replaced by

$$\vec{u} = \vec{u}_{ff} + (u_n - \vec{u}_{ff} \cdot \hat{n})\hat{n}. \tag{3.11}$$

Also, the entropy at the inflow boundary will not necessarily be uniform, since the prescribed flow is rotational. The specification of a uniform freestream entropy (Equation (2.27a)) is replaced by

$$\frac{p}{\rho^\gamma} = S(\vec{x}) \tag{3.12}$$

where $S(\vec{x})$ is the entropy distribution in the far field. In general, $S(\vec{x})$ will be equal to 1 everywhere except in the rotational core of the prescribed flow vortex, where it must vary as a consequence of Crocco's theorem. The manner in which the non-uniform entropy in the core is determined is described in the next section. The outflow boundary conditions are unchanged except for the new incoming Riemann variable, Equation (3.9).

Except for the changes in the time stepping algorithm and in the far field boundary conditions, the scheme is identical to that presented in the previous chapter. No changes in the stability of the scheme have been observed, and for all the cases presented here, a *CFL* number of 2.8 has been used.

The main difference between the perturbation approach as it is applied here and as it is used by Srinivasan et al. is that the flows being computed here are assumed to be steady. The prescribed flow $\mathbf{U}_0$ is a steady vortical flow, and the Euler equations are integrated in time to reach a steady state. For this reason, the prescribed flow residuals need only be computed once and stored, rather than being computed at each iteration.

A question arises for the case for which the prescribed flow is not a good approximation to the actual flow field over the entire computational domain. For example, consider a streamwise vortex impinging on a wing leading edge. A prescribed flow consisting of an undisturbed streamwise

vortex will be a very poor approximation both in the immediate vicinity of the wing and downstream of the wing. Even for cases in which the vortex does not hit the wing but still passes very close to it, the distortion of the vortex path from its undisturbed position will be large enough that specifying the undisturbed vortex as the prescribed flow will result in large errors, rather than a reduction of the error. However, the resolution of this problem is extremely simple: the terms $\mathbf{R}_0$ may be switched off in regions of the flow where $\mathbf{U}_0$ is a poor approximation to the actual local behavior. For example, in the case in which the vortex impinges on the wing leading edge, $\mathbf{R}_0$ may be set to zero near the wing and downstream of the trailing edge. In these regions, the Euler equations are being solved in the normal way, i.e. the vortical flow is "captured". By specifying the vortex up to the wing, the problem of numerical diffusion is avoided, so that the vortex will be properly defined at the wing. Also the grid resolution is the finest near the wing, meaning that the problem of numerical diffusion of the vortex is least important there. Finally, the structure of the vortex after its encounter with the wing is not known, and neglecting to specify the prescribed flow field in this region is consistent with allowing the wake generated by the wing to be captured by the Euler code.

3.3 Prescribed Flow Specification

Two sets of calculations will be shown to demonstrate the perturbation scheme. One is the convection of a streamwise vortex in a square cross section channel (Figure 3.3). The prescribed flow consists of an infinite vortex in an unbounded fluid with a uniform velocity along the direction of the vortex. The other set of calculations consists of the interaction of a streamwise vortex with a fixed wing. The geometry of the wing/vortex interactions described in this chapter is shown in Figure 3.4. A semispan wing is attached to wall, and a streamwise vortex is generated upstream. It

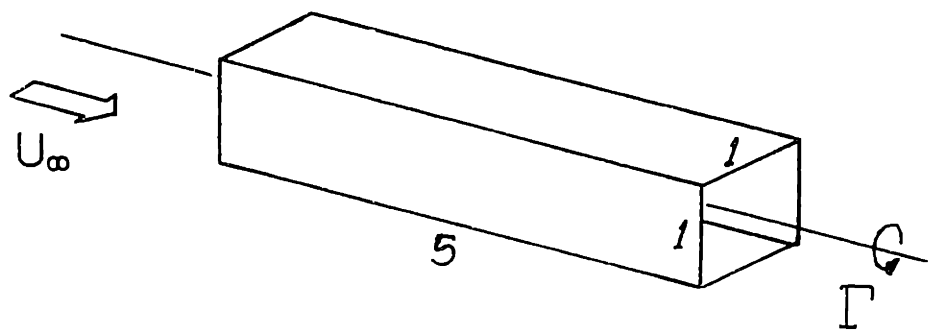

Figure 3.3: Channel geometry

is assumed that the vortex has completely rolled up and thus is axisymmetric by the time is reaches the wing. The distance of the vortex from the wall is denoted by y_v and its displacement above the wing is z_v. The manner of specifying the prescribed flow field $\mathbf{U}_0$ is essentially the same for both configurations, and is described below.

For the channel flow the velocity field of the prescribed flow is assumed to be identical to the induced velocity field of an infinite incompressible line vortex in a uniform freestream. For the wing/vortex cases, an image vortex to account for the symmetry plane is added as well. Since the induced velocity of the vortex pair results in a downward motion of the pair, the two vortices are placed at an angle to the freestream given by their mutual induced downwash. To avoid the singular behavior of the velocity field near the vortex, a finite core structure is necessary. The Rankine vortex is chosen for the channel flow cases due to its very simple core structure. For the wing/vortex interaction cases, a more physically realistic model of the upstream generated vortex is needed, and the Lamb core structure is used due to its smooth variation of vorticity.

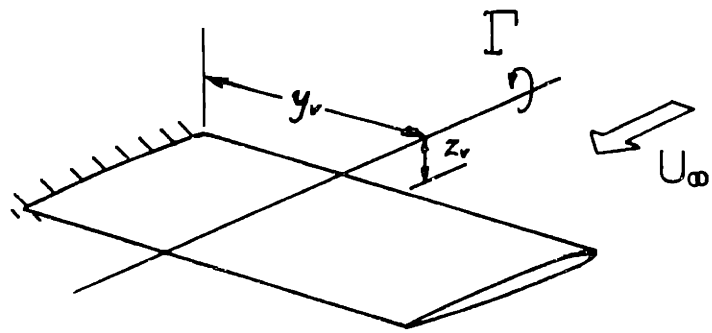

Figure 3.4: Wing/vortex interaction geometry

The tangential velocity of a two-dimensional Rankine vortex is given by

$$u_\theta = \begin{cases} \frac{\Gamma}{2\pi r} & \text{if } r > a \\ \frac{\Gamma r}{2\pi a^2} & \text{if } r \leq a \end{cases} \tag{3.13a}$$

and the tangential velocity of a Lamb vortex is given by

$$u_\theta = \frac{\Gamma}{2\pi r}\left\{1 - e^{-(\frac{r}{a})^2}\right\} \tag{3.13b}$$

where u_θ is the tangential velocity, Γ is the circulation, r is the distance from the center of the vortex and a is the vortex core size. The choices of the vortex core size and the circulation for the wing/vortex calculations are discussed in section 3.5. The induced velocity of the streamwise vortex and its image is computed at the center of each cell of the finite volume grid using Equation (3.13a) or (3.13b), and the uniform freestream velocity is added to that.

After computing the velocity at each cell, the density and pressure are required to complete the specification of the state vector $\mathbf{U}_0$. Outside the

core, these are determined by assuming a constant total enthalpy and entropy at the freestream values. In the core of the vortex, the total enthalpy is taken to be the freestream value. To determine the entropy distribution in the core, use is made of Crocco's theorem, given here in dimensional form:

$$\nabla H = T\nabla s + \vec{u} \times \vec{\omega}, \tag{3.14}$$

where H is the total enthalpy, T is the temperature, s is the specific entropy, $\vec{u}$ is the velocity, and $\vec{\omega} = \nabla \times \vec{u}$ is the vorticity. It is seen that Equation (3.14) implies that the entropy is not constant, but has a radial variation through the rotational core. To determine the entropy distribution through the core, equation Equation (3.14) is rewritten in non-dimensional form to get

$$\nabla H = \frac{p}{\rho} \frac{1}{\gamma - 1} \nabla s + \vec{u} \times \vec{\omega} \tag{3.15}$$

where the equation of state has been used to eliminate T, and s has been non-dimensionalized by c_v. Now, if the total enthalpy is taken to be constant, Equation (3.15) may be simplified further by using the definition of H to eliminate p and ρ. The resulting equation is

$$\frac{1}{\gamma}\left(H - \frac{\vec{u} \cdot \vec{u}}{2}\right) \nabla s = -\vec{u} \times \vec{\omega}. \tag{3.16}$$

Since H is determined by the freestream Mach number and $\vec{u}$ is known from the induced velocity field of the vortices making up the prescribed flow field, s can be found by numerically integrating Equation (3.16) through the vortex core. For the channel flows, this is easy to do since the flow is axisymmetric about the vortex core, and there is only a radial variation in entropy. Strictly speaking, for the wing/vortex flows, the flow will not be axisymmetric due to the influence of the image vortex. However, for the small core vortices treated here, the flow near the core is very nearly that of an isolated vortex, and thus will be treated as axisymmetric for the purposes of integrating Equation (3.16).

For a Rankine core structure, Equation (3.16) for an axisymmetric vortex reduces to

$$\frac{\partial s}{\partial r} = \frac{-\gamma \frac{\Gamma^2 r}{2\pi^2 a^4}}{\frac{\gamma}{\gamma-1} - \frac{\Gamma^2 r^2}{8\pi^2 a^4}}, \tag{3.17a}$$

for $r \leq a$. The entropy is uniform outside the core, $r > a$. This entropy distribution is plotted in Figure 3.5 for $\Gamma/a = 1$. A Lamb vortex has a radial

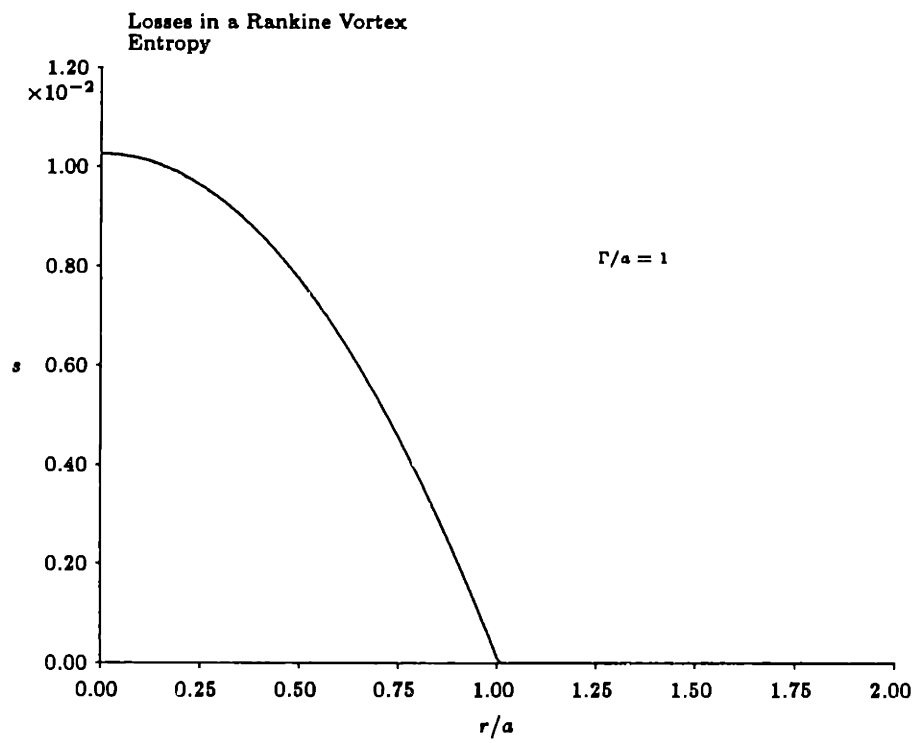

Figure 3.5: Nondimensional entropy distribution in a Rankine vortex core, $\Gamma/a = 1$, $\gamma = 1.4$

entropy distribution given by

$$\frac{\partial s}{\partial r} = \frac{-\gamma \frac{\Gamma^2}{2\pi^2 a^2 r} e^{-\left(\frac{r}{a}\right)^2} \left\{1 - e^{-\left(\frac{r}{a}\right)^2}\right\}}{\frac{\gamma}{\gamma-1} - \frac{\Gamma^2}{8\pi^2 r^2} \left\{1 - e^{-\left(\frac{r}{a}\right)^2}\right\}^2} \tag{3.17b}$$

and is shown in Figure 3.6. The above equations are given in non-dimensional

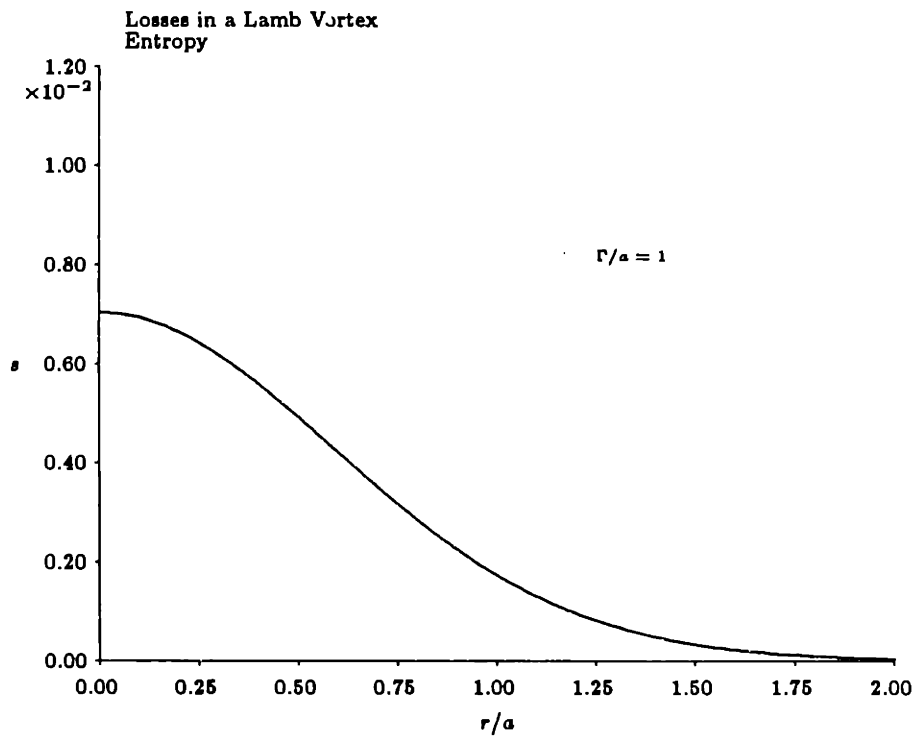

Figure 3.6: Nondimensional entropy distribution in a Lamb vortex core, $\Gamma/a = 1, \gamma = 1.4$

form; these are integrated numerically to find the distribution of entropy in the core. A cubic spline is fitted through the resulting distribution to allow interpolation of the entropy to the centers of the finite volume cells. The spline formula is given in Dahlquist & Björck ([21], pp. 131 to 134). Outside the vortex, where the entropy is uniform, s is set equal to zero.

Once the entropy is known, the entire prescribed state vector $\mathbf{U}_0$ can be determined. To get the density, the equation of state and the constant total enthalpy assumption are used. The equation of state describing the relation

between pressure, density, and entropy, is

$$p = \rho^{\gamma} e^{s}. \tag{3.18a}$$

It is more convenient to define a new variable

$$S = e^{s}$$

and write the equation of state as

$$\frac{p}{\rho^{\gamma}} = S. \tag{3.18b}$$

This definition gives S equal to 1 outside the rotational core, and greater than 1 in the core. This is the same definition of S as the function $S(\vec{x})$ given in the discussion of the far field boundary conditions in the last section. From the known velocity field and total enthalpy, the ratio of pressure to density is

$$\frac{p}{\rho} = \frac{\gamma - 1}{\gamma}\left(H - \frac{\vec{u}\cdot\vec{u}}{2}\right) \tag{3.19}$$

Combining Equations (3.19) and (3.18b) yields the following equation for the density,

$$\rho = \left\{\frac{\gamma - 1}{\gamma S}\left(H - \frac{\vec{u}\cdot\vec{u}}{2}\right)\right\}^{\frac{1}{\gamma-1}}. \tag{3.20}$$

Also, from Equation (3.19), the total energy $E = H - p/\rho$ can be readily found. With the density, velocity, and the total energy known at each cell in the domain, the prescribed flow state vector $\mathbf{U}_0$ is known.

Now that $\mathbf{U}_0$ is specified throughout the computational domain, the finite volume operator for the flux integral and the artificial viscosity operator are applied to $\mathbf{U}_0$. This gives the residuals $\mathbf{R}_0$ of the prescribed flow. These are computed once and stored. During the multi-stage time integration, $\mathbf{R}_0$ is subtracted from the residuals $\mathbf{R}$ of the complete flow field $\mathbf{U}$. Thus in the steady state, the residuals $\mathbf{R}$ are driven to $\mathbf{R}_0$, not to zero. Near the core of the vortex, where the prescribed flow residuals are large, the state vector $\mathbf{U}$

will show the same rapid variations as the undisturbed vortex. Away from the vortex, the residuals $\mathbf{R}_0$ will be small, and the solution will be locally similar to a conventional Euler solution.

It should be pointed out that it is not necessary to have an analytic model for the prescribed vortical flow field. In principle, a numerical solution for the prescribed flow can be used. For example, a high resolution numerical solution of the Navier-Stokes equations for an isolated vortex could take the place of the Rankine or Lamb vortex core structures used here.

3.4 Vortex in Channel

The first set of computations shown are for the flow of a streamwise vortex in a square section channel. The purpose of these results is to illustrate, using a simple model problem, how the perturbation scheme eliminates the numerical diffusion of vorticity. The configuration is shown is Figure 3.3. The channel length is five times the channel height. The grid points are uniformly distributed, with a crossflow resolution of 20 $\times$ 20 cells, and 10 cells in the streamwise direction. The inlet Mach number is 0.5, vortex core radius is $0.1h$, where h is the channel height, and the vortex circulation is $0.1a_\infty h/\sqrt{\gamma}$, where a_∞ is the freestream speed of sound. A Rankine core structure is assumed. The choice of the vortex strength is arbitrary. The core size is such that it lies across approximately 4 $\times$ 4 finite volume cells in the crossflow direction. This is a coarse resolution of the vortex, but is finer than is generally acheived for lift generated wakes such as presented in the last chapter. For the fine mesh results presented in chapter 2, the resolution of the tip vortex was roughly similar to the resolution for these cases. Three cases will be shown: in the first, the Euler solver was run without the perturbation scheme, and the vortex was introduced only through the inlet boundary conditions; in the second, the perturbation scheme was used to prescribe the vortex flow field; and in the third, a prescribed flow

field that did not satisfy the Euler equations was used to demonstrate the consequences of the failure of satisfying the consistency. All cases were run at a *CFL* of 2.8, a fourth difference artificial viscosity coefficient of 0.004, no second difference dissipation, and an enthalpy damping coefficient of 0.025.

Figure 3.7 shows the vorticity vectors at the middle of the channel for

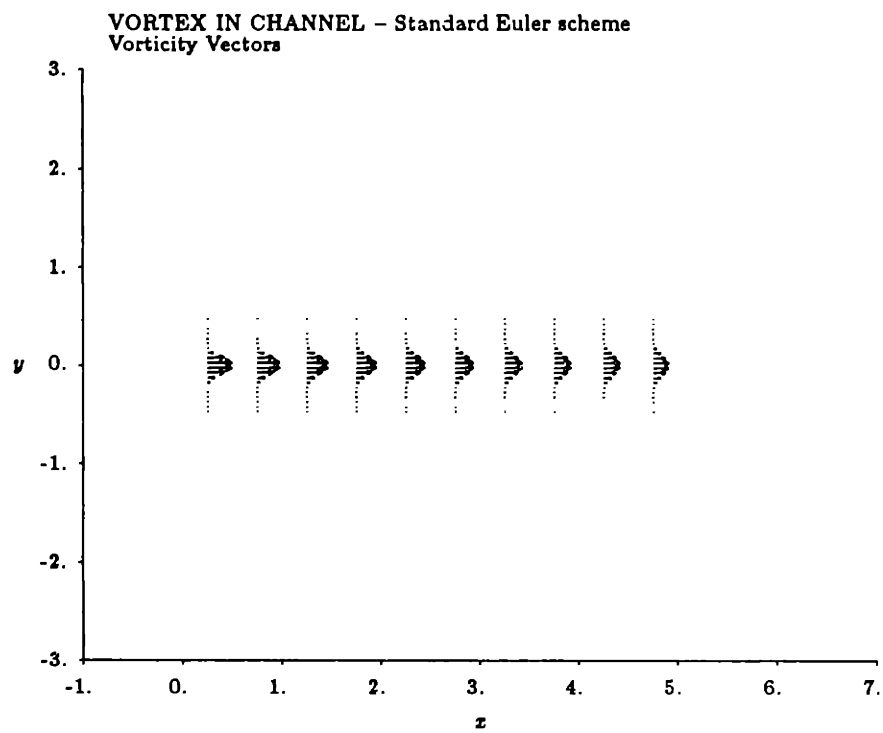

Figure 3.7: Vorticity vectors in channel, standard Euler scheme

the standard Euler scheme. The mean flow is in the same direction as the vorticity vectors. Although the specified vortex has a Rankine core, for which the vorticity is uniform in the core and zero outside, it is seen at the inlet that the vorticity apparently is not uniform in the core. This is due to the fact that the vorticity is computed by a finite difference approximation

to the curl, and the truncation error of the difference formula gives the appearence of a nonuniform vorticity.

It is immediately apparent, looking at Figure 3.7, that the vorticity is being diffused as it is convected downstream. The core size is growing, and the peak vorticity magnitude is decreasing downstream. This is more clearly seen in contour plots of the vorticity magnitude in the cross flow plane at the inlet and outlet (Figures 3.8 and 3.9). Plots of total pressure loss at

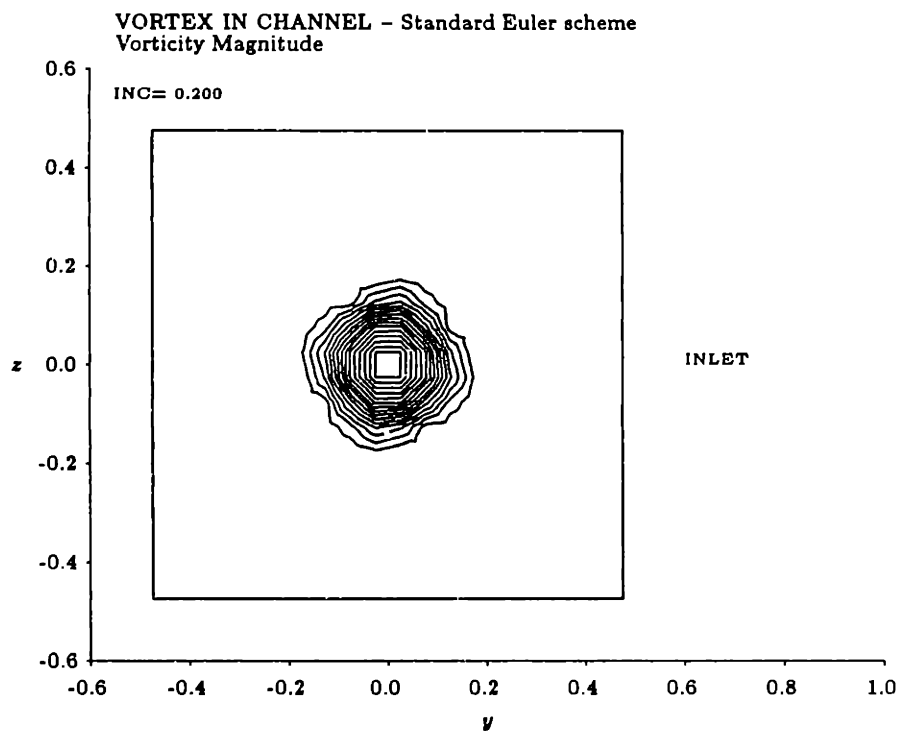

Figure 3.8: Vorticity magnitude contours in channel inlet cross-section, standard Euler scheme

the inlet and outlet (Figures 3.10 and 3.11) also show the growth of the core and the diffusion of vorticity. These effects are purely numerical; the

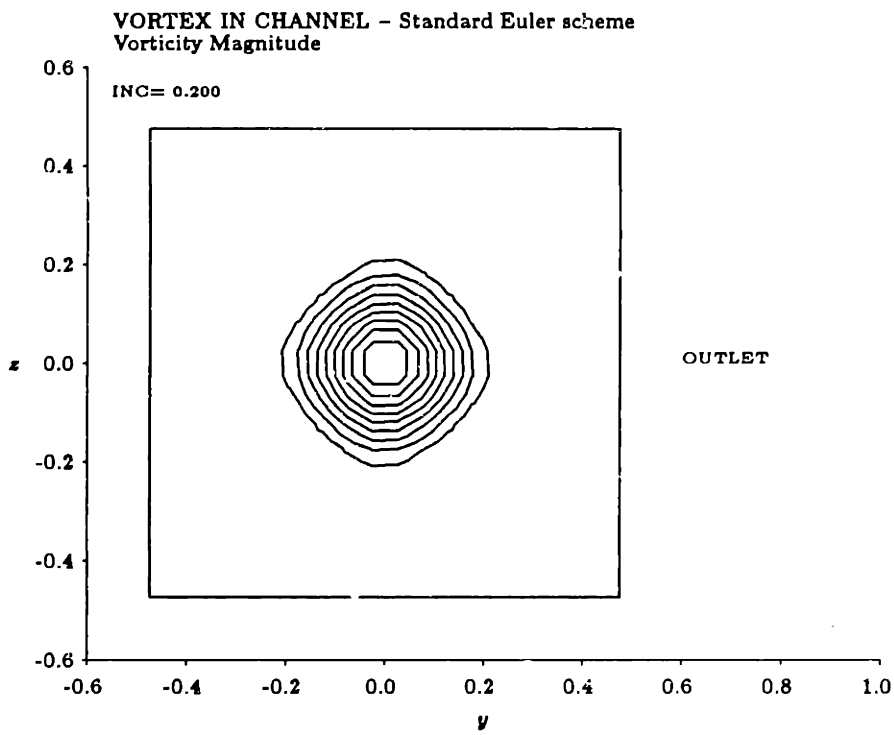

Figure 3.9: Vorticity magnitude contours in channel outlet cross-section, standard Euler scheme

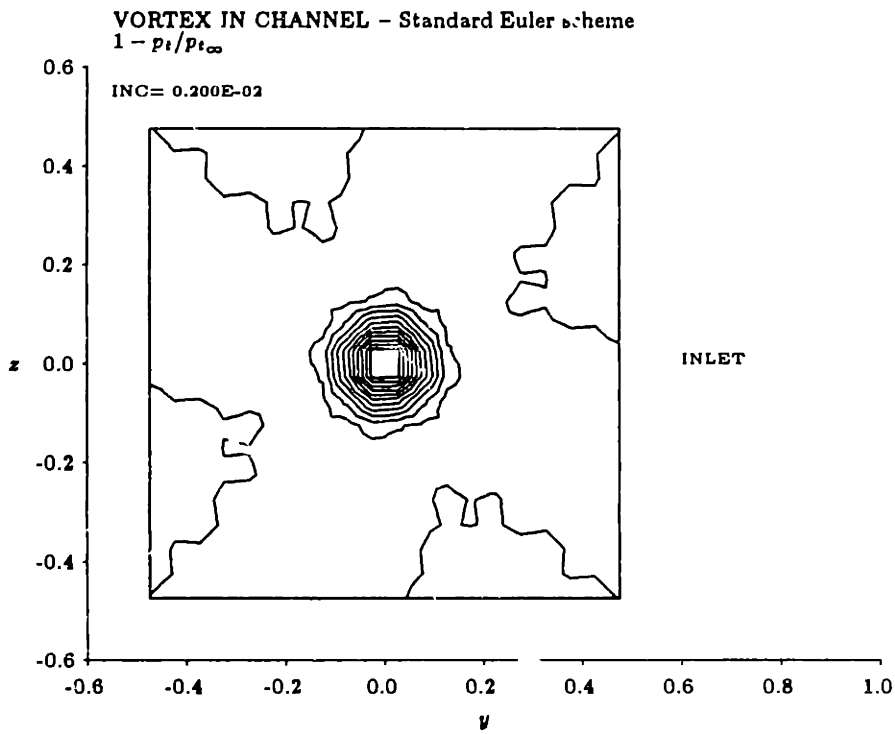

Figure 3.10: Total pressure contours in channel inlet cross-section, standard Euler scheme

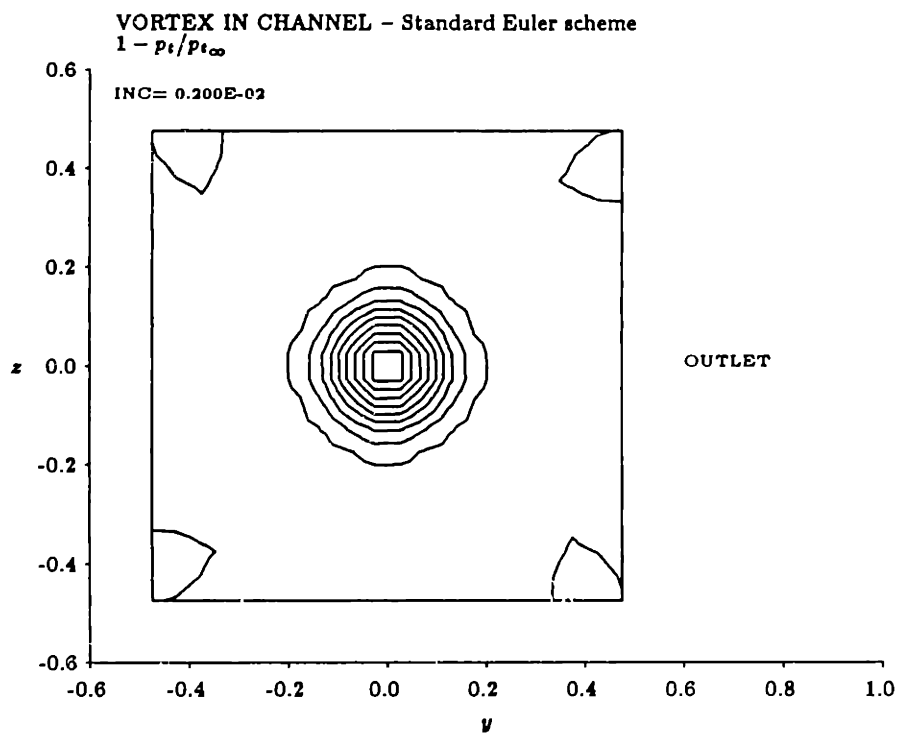

Figure 3.11: Total pressure contours in channel outlet cross-section, standard Euler scheme

Euler equations are not dissipative, so the vortex should not diffuse as it is convected downstream. Also, this model problem is highly idealized, in that the vortex structure is very simple (there is no rolling up of a vortex sheet as with the lift generated wakes of the previous chapter) and the grid is uniform. Yet even here, it is seen that there is considerable numerical diffusion. In most cases of interest, the grid stretching and the evolution of the vortical regions will further aggravate the situation, and numerical diffusion will in all likelihood be even worse.

To illustrate the effectiveness of the perturbation scheme in eliminating the numerical diffusion of vorticity, this same case was run using the perturbation scheme in which the prescribed flow consisted of a streamwise vortex in an unbounded fluid. All the numerical parameters (artificial viscosity coefficients, *CFL* number, and enthalpy damping coefficient) were unchanged. Figures 3.12, 3.13, 3.14, 3.15 and 3.16 show the vorticity vectors, vorticity magnitude contours, and total pressure loss contours at the same locations as in the previous case. It is seen that the vortex core retains its definition throughout the channel. The numerical diffusion of vorticity is entirely eliminated. It is important to note that the perturbation scheme had no effect on the convergence rate of the solver. Figures 3.17 and 3.18 show the iteration history, with the logarithm of the root mean square of the changes ΔU plotted against the iteration number. This shows that the perturbation scheme does not adversely impact the performance of the Euler code. This should be true for any algorithm used to solve the Euler equations, and is not peculiar to the finite volume scheme used here.

The final case shown in this section demonstrates what happens when the prescribed flow does not statisfy the steady Euler equations. In this case, the prescribed flow was that of an infinite vortex at a slight angle to the freestream. The vorticity vectors are shown in Figure 3.19. Two things can be noted about this solution. The first is that the vortex does not experience

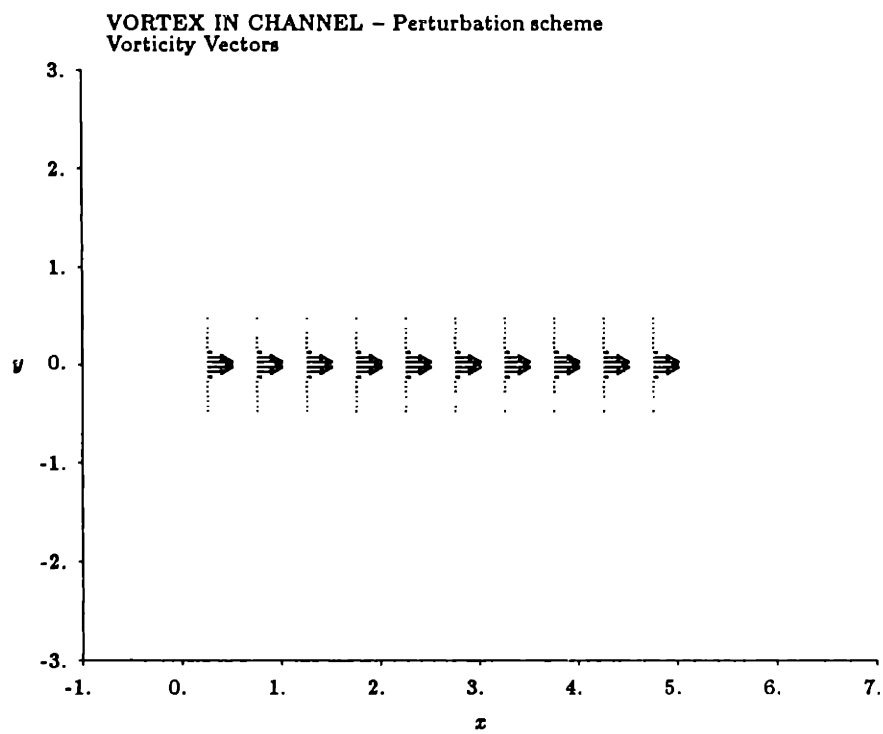

Figure 3.12: Vorticity vectors in channel, perturbation scheme

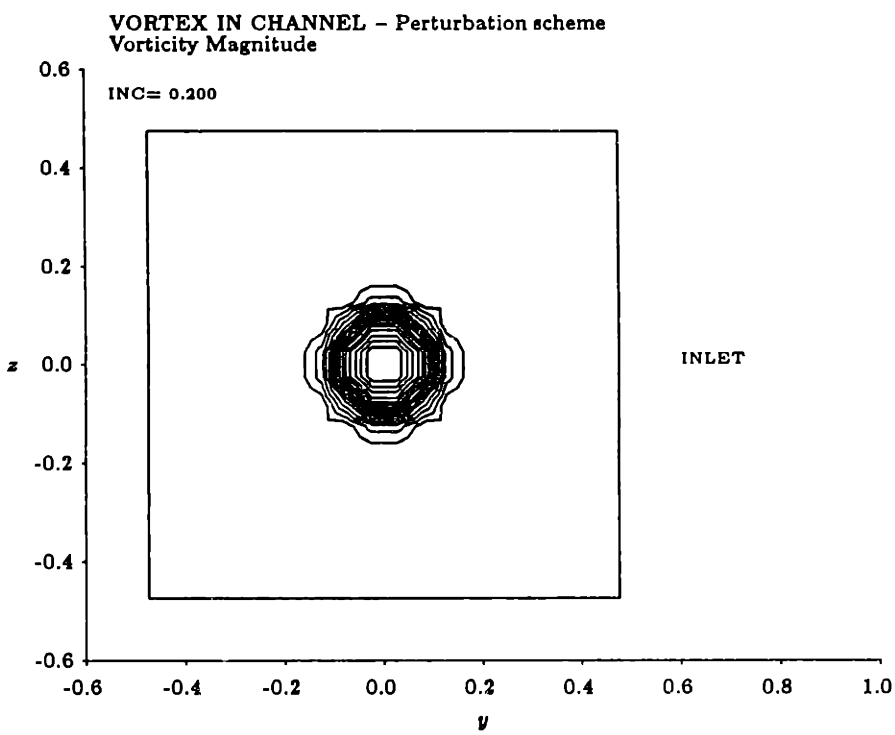

Figure 3.13: Vorticity magnitude contours in channel inlet cross-section, perturbation scheme

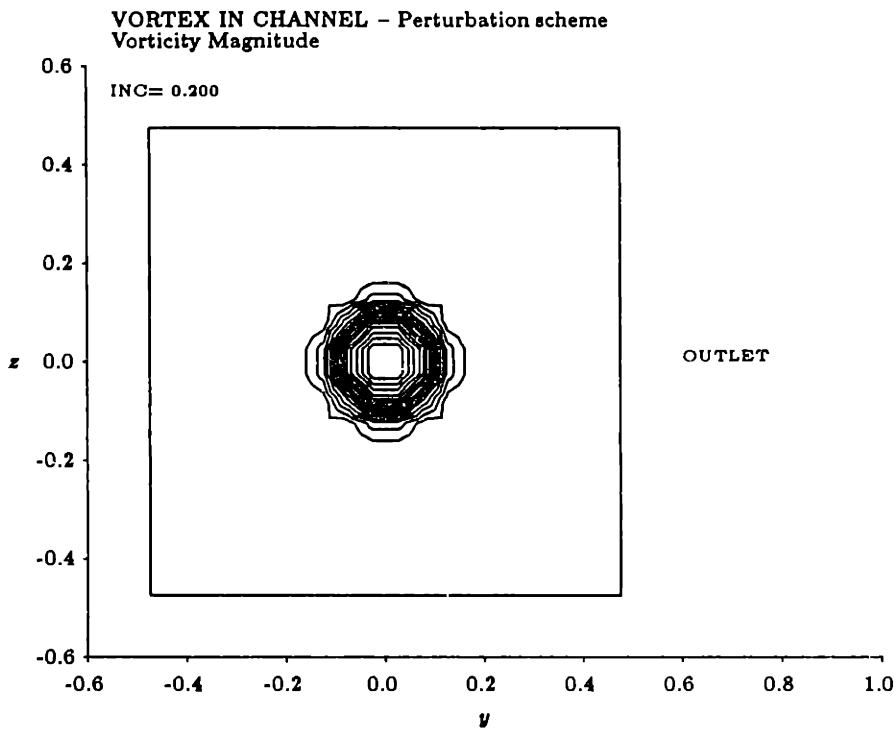

Figure 3.14: Vorticity magnitude contours in channel outlet cross-section, perturbation scheme

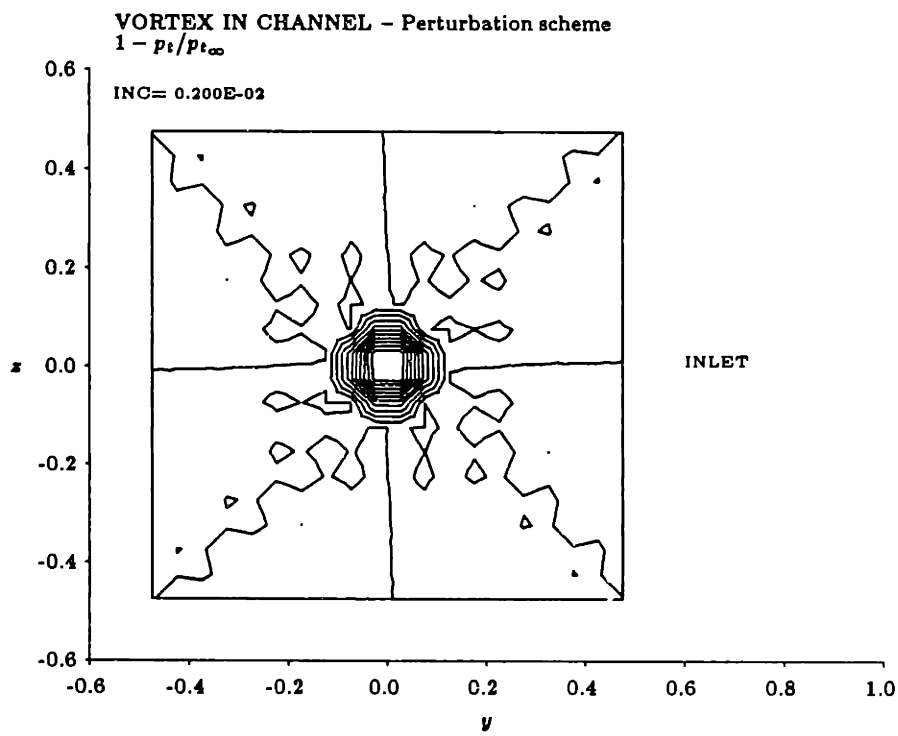

Figure 3.15: Total pressure contours in channel inlet cross-section, perturbation scheme

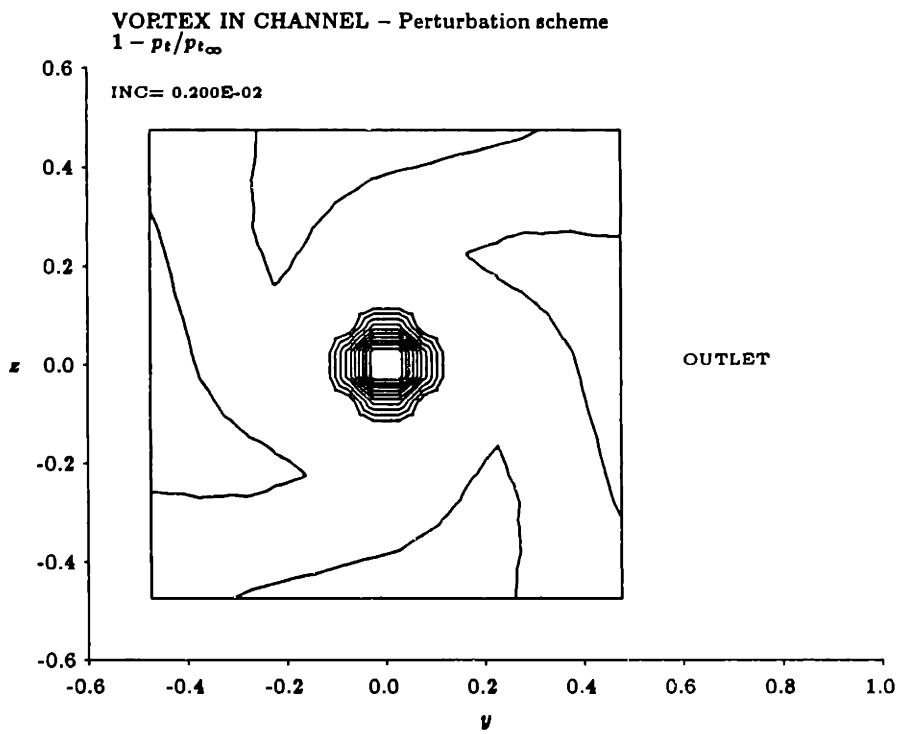

Figure 3.16: Total pressure contours in channel outlet cross-section, perturbation scheme

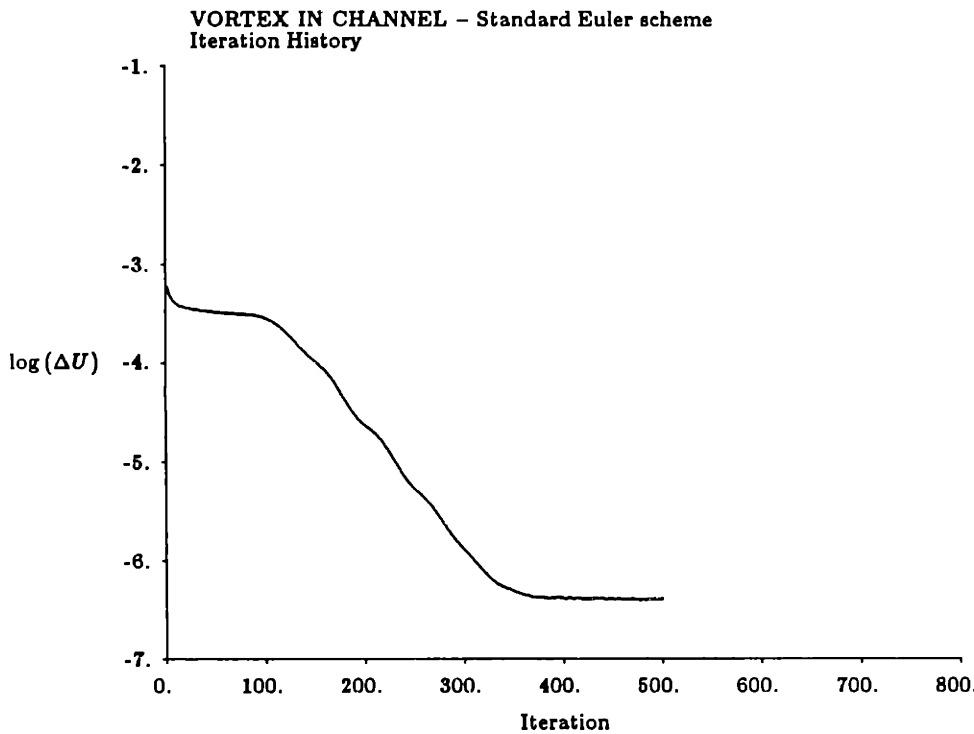

Figure 3.17: Convergence history, standard Euler

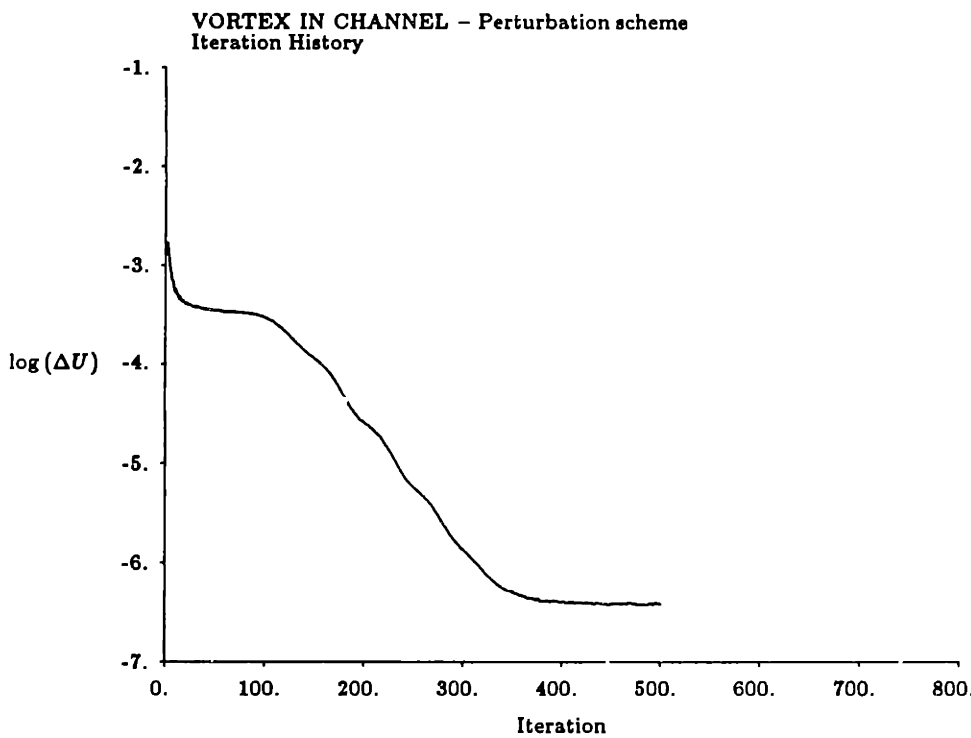

Figure 3.18: Convergence history, perturbation scheme

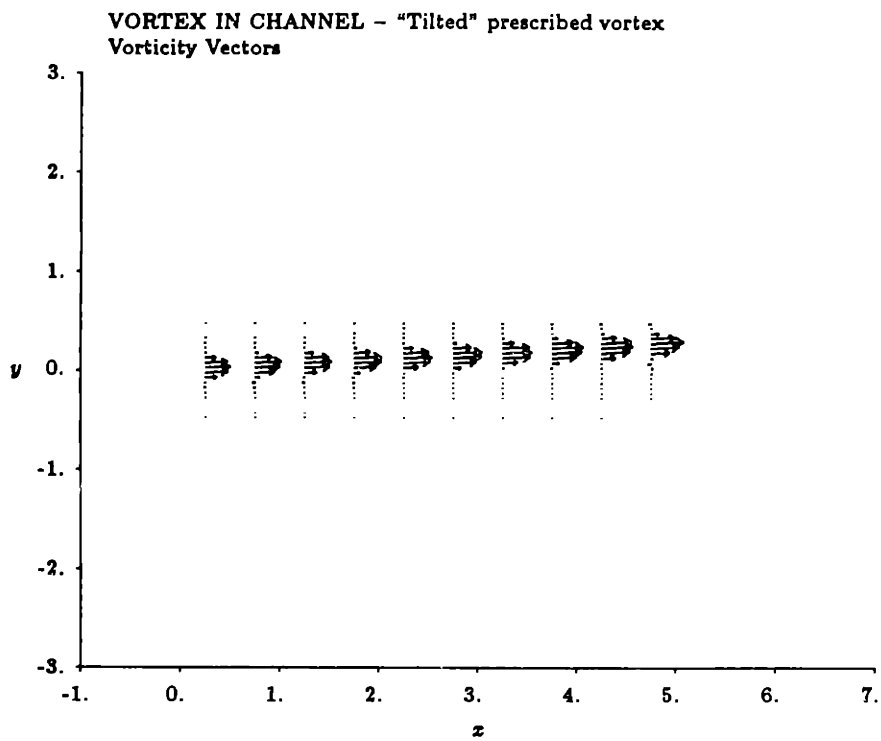

Figure 3.19: Vorticity vectors in channel, perturbation scheme, tilted vortex

numerical diffusion. The second is that the solution makes no physical sense whatsoever. The problem here is that the prescribed flow is not a solution to the Euler equations, unlike the previous case. Furthermore, in the limit of vanishing grid spacing, the solution obtained with this prescribed flow will still show the same behavior, with the vortex lying at an angle to the freestream. Because $\mathbf{U}_0$ does not satisfy the steady Euler equations, the residuals $\mathbf{R}_0$ will not vanish as the grid is refined.

In looking at the plots of vorticity magnitude (Figures 3.20 and 3.21) and total pressure loss (Figures 3.22 and 3.23) at the inlet and outlet, it is seen that the vortex core does not get diffused. More interestingly, the

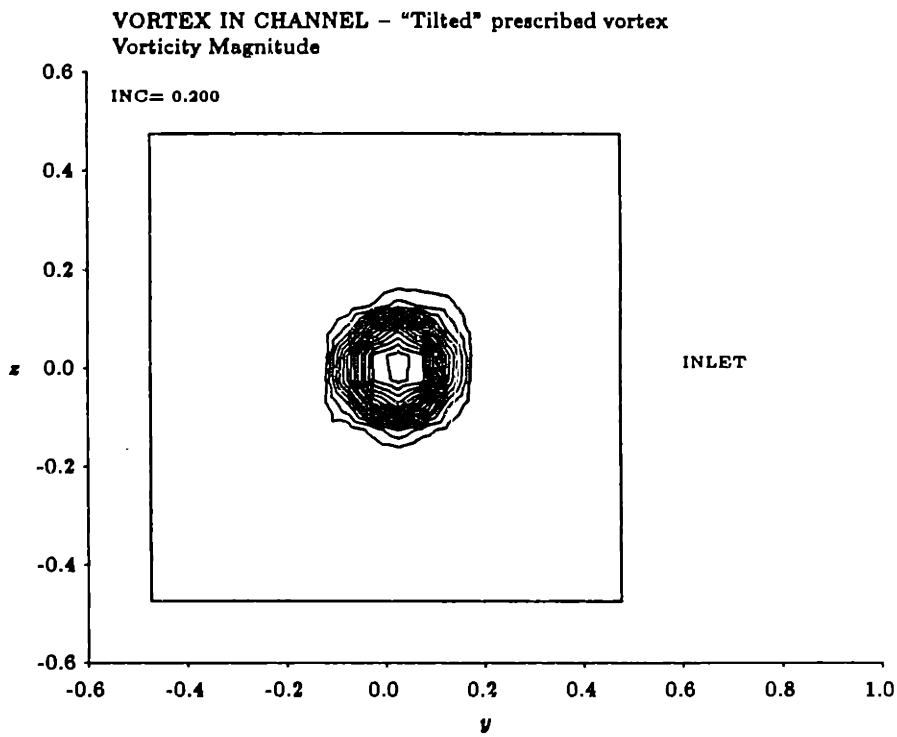

Figure 3.20: Vorticity magnitude contours in channel inlet cross-section, perturbation scheme, tilted vortex

vortex has moved upward slightly from the channel centerline ($z = 0$) at the outlet. The prescribed flow has the vortex tilted in the y direction, but at a constant z. The reason for this is simply because the vortex, being tilted slightly to the freestream direction, must experience a lift. In the Figures 3.20 to 3.23, the primary flow direction is out of the plane of the page, and the sense of the circulation around the vortex is counter-clockwise. Thus the vortex should experience a force in the positive z direction, which explains the displacement of the vortex from its prescribed position.

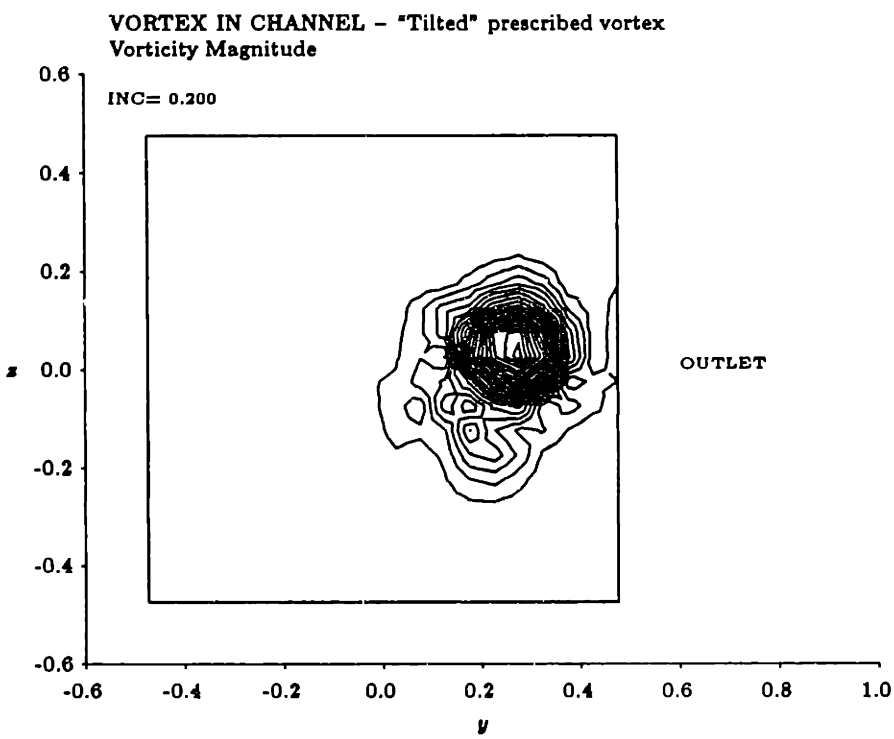

Figure 3.21: Vorticity magnitude contours in channel outlet cross-section, perturbation scheme, tilted vortex

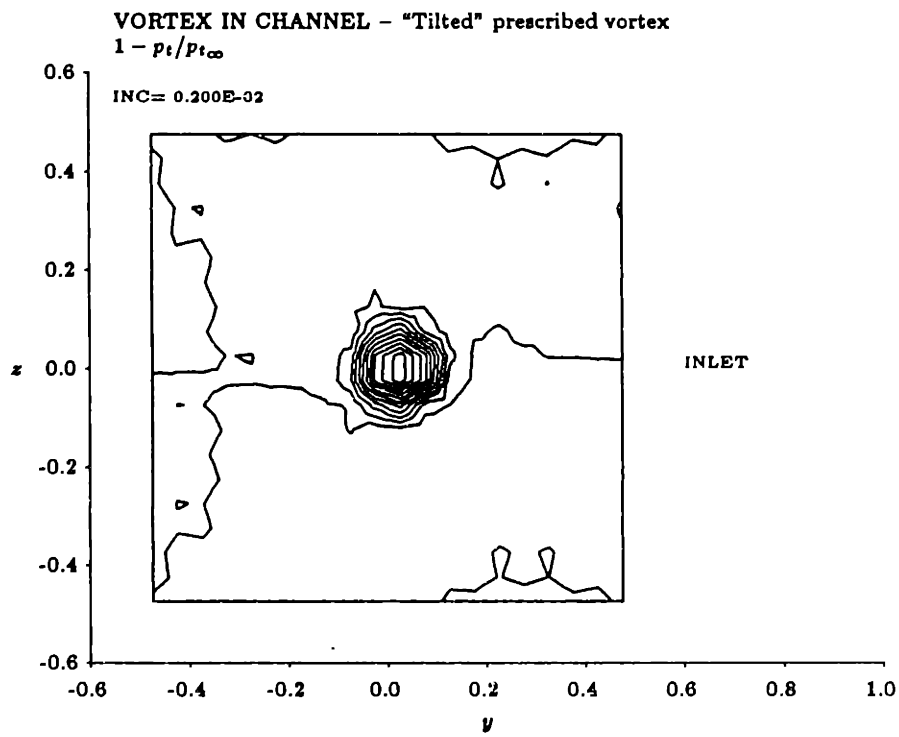

Figure 3.22: Total pressure contours in channel inlet cross-section, perturbation scheme, tilted vortex

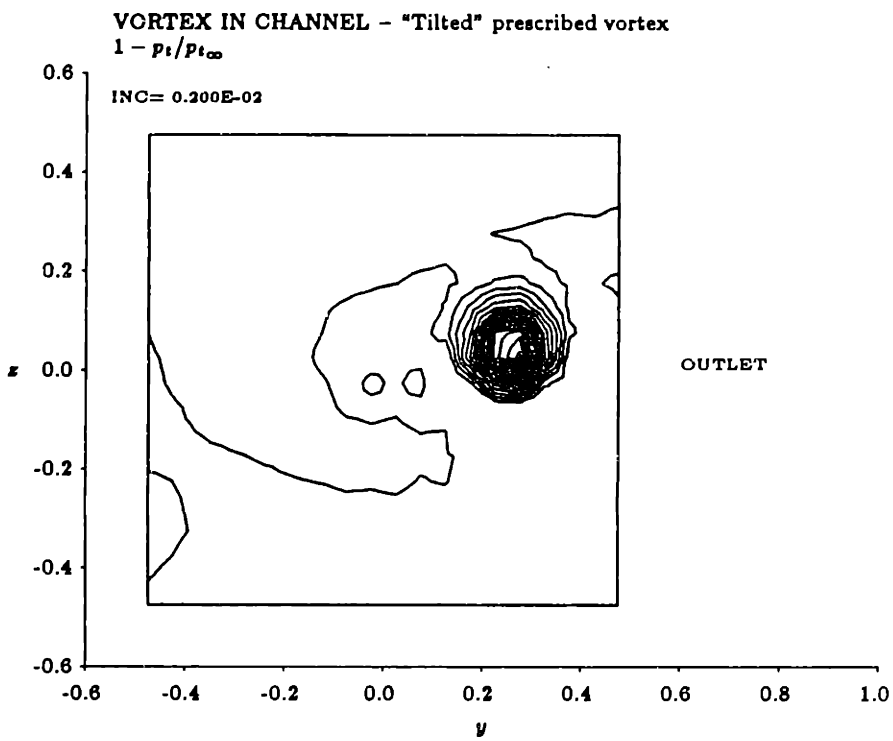

Figure 3.23: Total pressure contours in channel outlet cross-section, perturbation scheme, tilted vortex

3.5 Wing/Vortex Interaction

As a demonstration of the ability of the perturbation approach to accurately handle flows with compact vortical regions, computations of the interaction of a streamwise vortex with a wing have been performed. The configuration is shown in Figure 3.4. The wing has a semispan to chord ratio of 2:1, and a NACA 0006 airfoil section. It is untwisted and untapered. Three cases have been computed and compared to the experimental results of Smith & Lazzeroni [57]. In each calculation, the circulation and core size of the vortex was fixed. The prescribed flow field $\mathbf{U}_0$ was specified as described in the previous section.

The core size and circulation of the vortex is required in order to compute $\mathbf{U}_0$. The core size was determined from the experimental measurements of the downwash in the wake of the vortex generator. By measuring the distance between the maximum and minimum downwash, the core radius was found to be about 0.05 chord. This was taken to be the same for all the cases. The circulation was chosen by matching the experimentally observed lift coefficient at the wing root for the first case presented here by using a lifting line analysis. This gave a vortex circulation of 0.05 normalized by the freestream velocity and wing semispan. It was then assumed that the vortex strength was the same for all the other cases run. This assumption is based on the requirement that the induced downwash of the wing on the vortex generator is the same for each case run. This assumption is reasonable, since the induced lift on the wing did not vary much between the four cases run. To further check the validity of this assumption two checks were done. First, the induced downwash at the vortex generator due to the wing was estimated by treating the wing as a horseshoe vortex. Because for the cases shown here the wing was only very lightly loaded from the mid semispan to the tip, the horseshoe vortex was taken to have the same lift as the wing but only half the span. The induced angle of attack at the vortex generator

on the symmetry plane was was found to be only 0.025°; this is only $\frac{1}{2}$% of the geometric angle of attack of 5°. Within the range of the lift coefficients for the cases here, the induced velocity at the vortex generator due to the wing is negligible, and the assumption of a constant circulation is justified. The second check on the strength of the vortex was done by using a lifting line solution for the isolated vortex generator to determine the peak bound circulation, which should equal the circulation of the fully rolled up tip vortex. The lifting line gave the same vortex circulation as was determined by the above procedure.

Three cases are shown below. For each case, the vortex core size and strength are the same. The wing is at zero angle of attack for all three cases, and the vortex is convected over the wing at a distance of 1/2 semispan from the wall. The vertical location of the vortex above the wing is 1/2, 1/4, and 0 chord lengths, respectively. Computations were done on a grid of $96 \times 20 \times 20$ cells. No second difference dissipation was used, the fourth difference artificial viscosity coefficient was 0.01, and the enthalpy damping coefficient was 0.025 for each case. All cases were run at a *CFL* of 2.8. Two computations were performed for each of the cases, one with the perturbation scheme and one with the standard Euler algorithm. In the latter set of solutions, the vortex was introduced into the computational domain only through the far field boundary conditions. In the first case the vortex is 1/2 chord from the wing. This is sufficiently far from the wing that the vortex path differs little from its undisturbed location, and the prescribed flow residuals are specified throughout the domain. Figure 3.24 compares the spanwise C_l distribution computed with the perturbation scheme to the experimentally measured values. The vortex location is specified throughout the computational domain. Excellent agreement with the experimental data is seen for the perturbation solution, while the conventional Euler solution (Figure 3.25) does not show the rapid gradients at the mid semispan of the

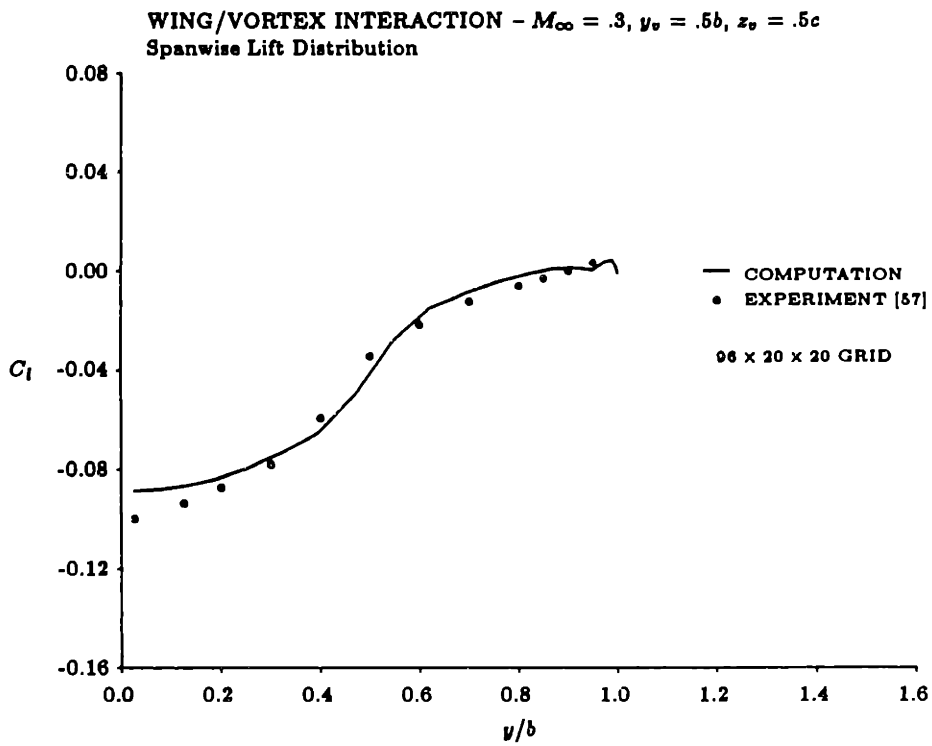

Figure 3.24: Comparison of computed and experimental spanwise lift distributions, $z/c = 0.5$, perturbation scheme

wing. The "blip" in the lift coefficients distributions in the tip region is a result of the local increase in lift due to the rolling up of the tip vortex over the wing tip. Note that the solution obtained with the perturbation scheme shows a slight positive lift over the outboard section of the wing, as in the experiment, while the standard Euler solution has a negative load over the entire wing.

For the case in which the vortex passed 1/4 chord above the wing (Figure 3.26), the lift distribution shows oscillations around the mid semispan when the prescribed flow residuals were specified everywhere. This is pre-

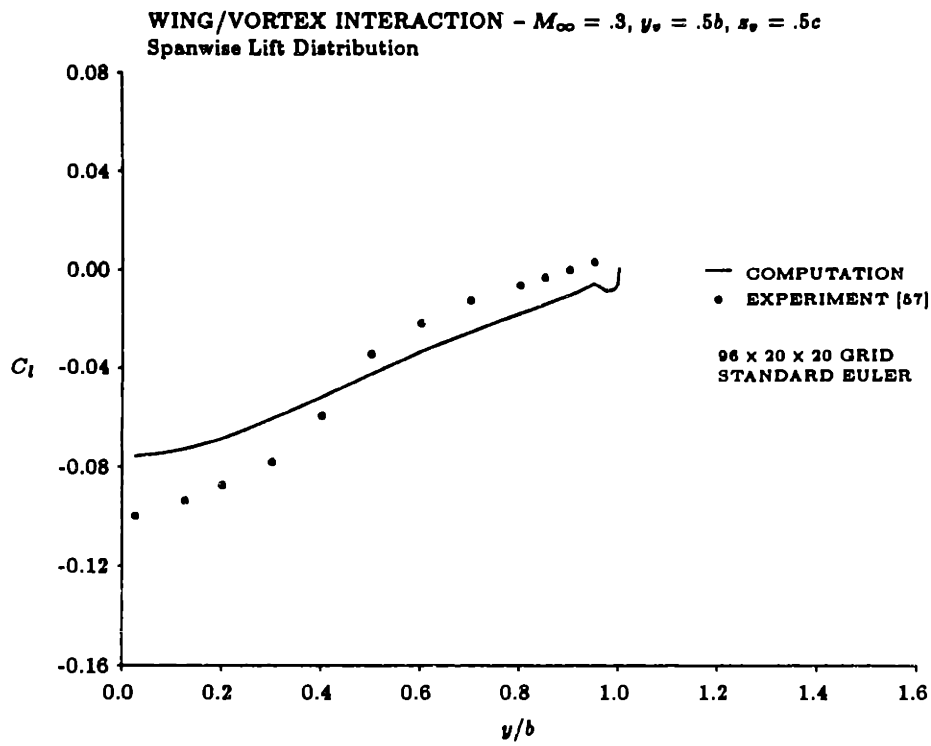

Figure 3.25: Comparison of computed and experimental spanwise lift distributions, $z/c = 0.5$, standard Euler scheme

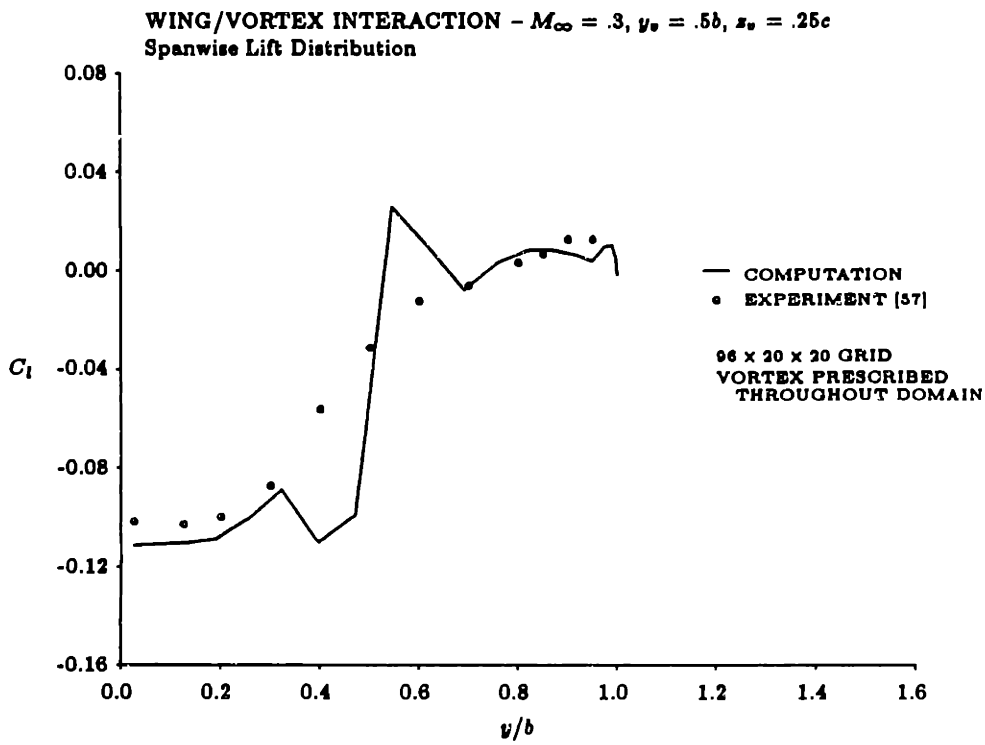

Figure 3.26: Comparison of computed and experimental spanwise lift distributions, $z/c = 0.25$, perturbation scheme, vortex prescribed everywhere

sumably because the undisturbed vortex path is quite different from the actual path, and $\mathbf{U}_0$ is not a good approximation to the local behavior after the vortex passes over the wing. When $\mathbf{R}_0$ is set to zero above the wing the spanwise wiggles disappear. This is the solution shown in Figure 3.27. Again, prescribing the vortex in the region before the wing allows the steep gradients in the spanwise lift distribution to be captured, as opposed to the non-perturbation solution of Figure 3.28 in which $\mathbf{R}_0$ is zero everywhere.

In Figures 3.29 and 3.30, the vortex is impinging on the wing leading edge. Clearly the undisturbed vortex flow field is an extremely poor ap-

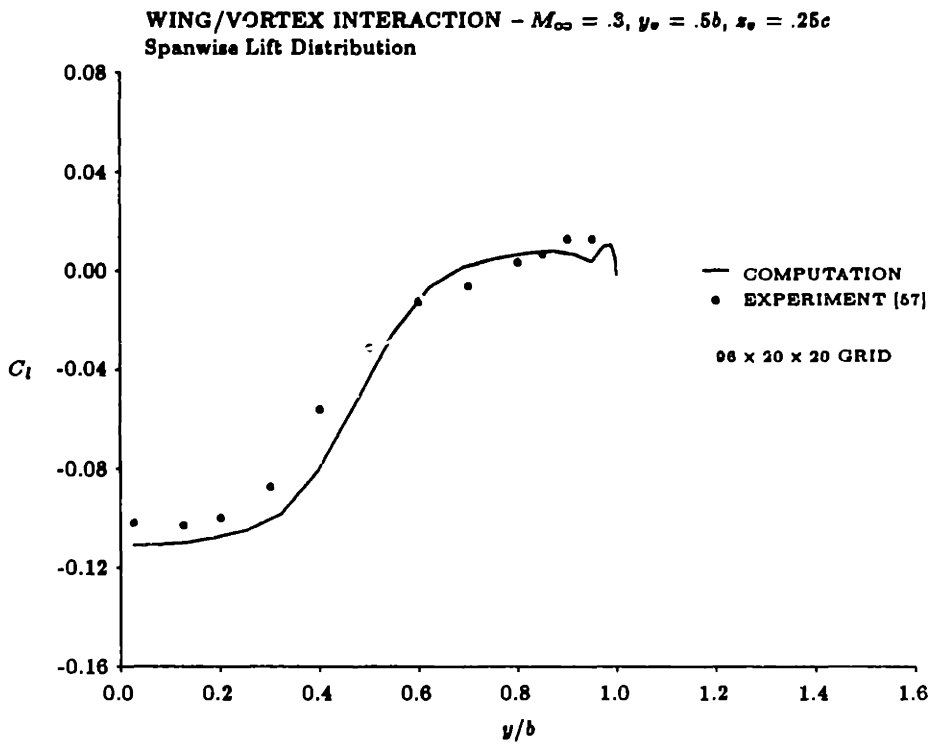

Figure 3.27: Comparison of computed and experimental spanwise lift distributions, $z/c = 0.25$, perturbation scheme, vortex prescribed up to wing

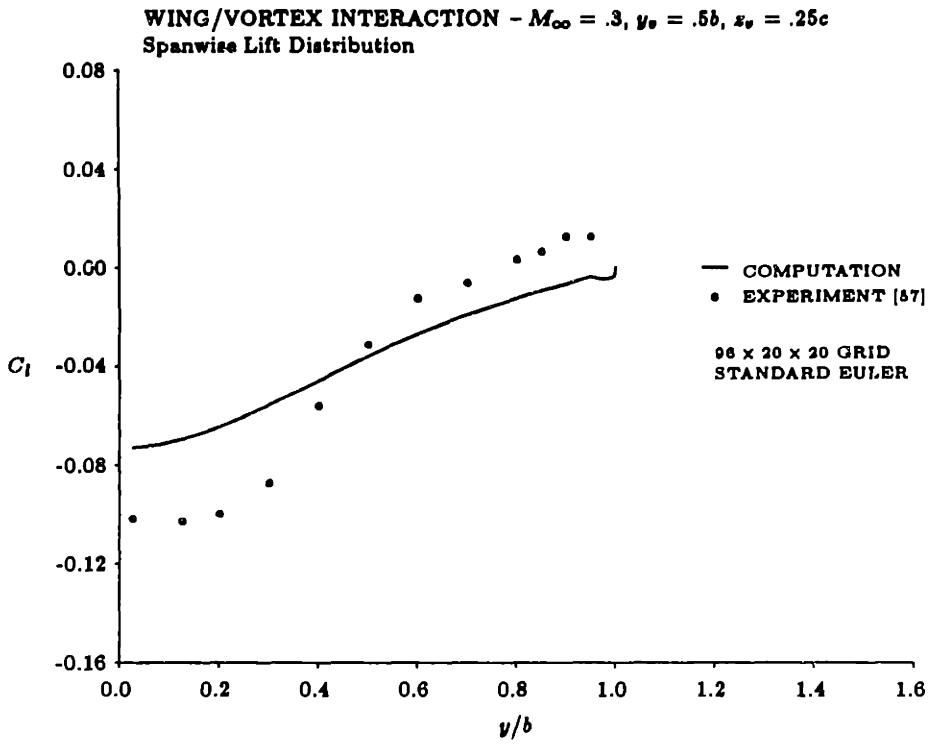

Figure 3.28: Comparison of computed and experimental spanwise lift distributions, $z/c = 0.25$, standard Euler scheme

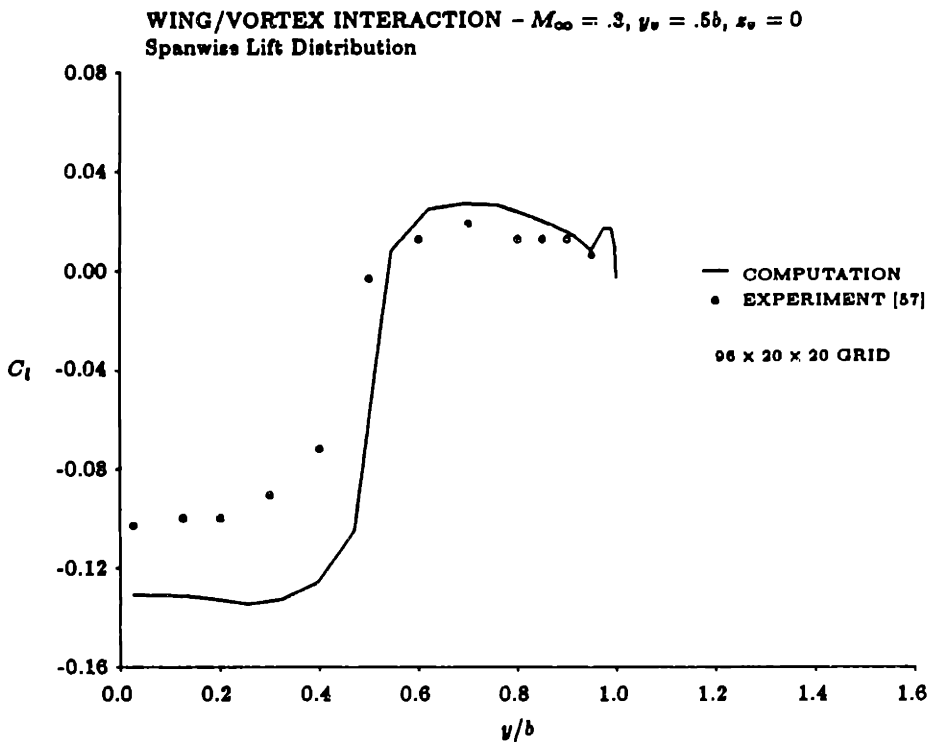

Figure 3.29: Comparison of computed and experimental spanwise lift distributions, $z/c = 0$, perturbation scheme, vortex prescribed up to wing

proximation to the flow after the vortex hits the wing. For this reason, the prescribed flow residuals $\mathbf{R}_0$ are turned off once the vortex reaches the wing. Note that the perturbation solution shows much steeper spanwise gradients in the lift compared to the non-perturbation solution. The rapid change in the lift as the induced velocity changes from downwash to upwash is completely lacking in the conventional Euler solution (Figure 3.30). The perturbation solution agrees less well with experiment than the other two cases, with a much higher computed download at the wing root than was observed experimentally. The discrepancy between computation and

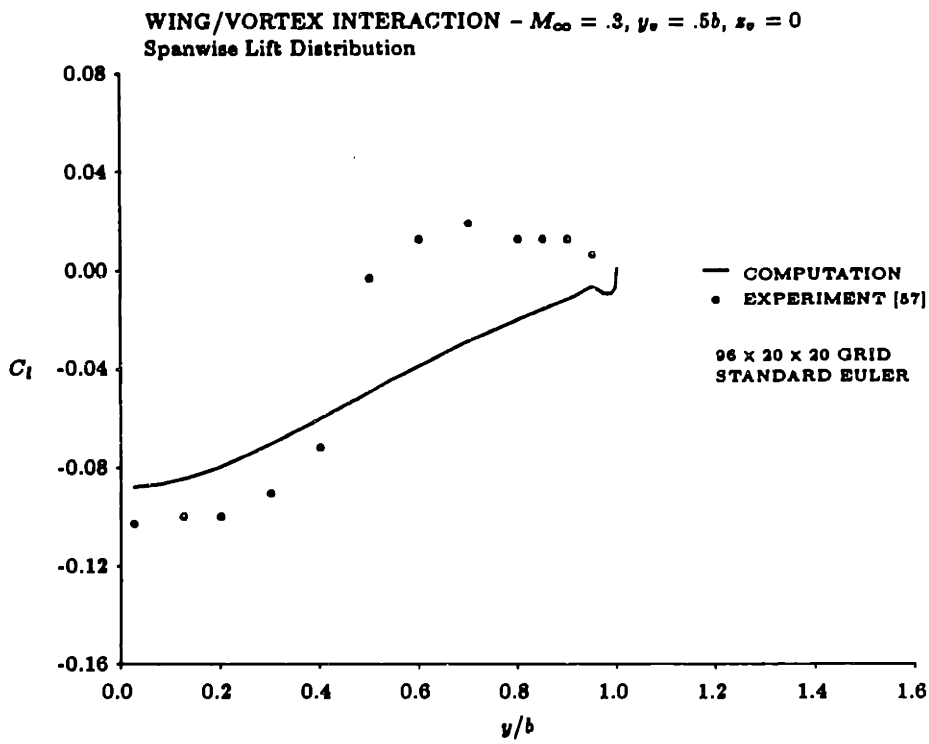

Figure 3.30: Comparison of computed and experimental spanwise lift distributions, $z/c = 0$, standard Euler scheme

experiment is most likely due to the fact that the experimental vortex was not axisymmetric, but was a trailing vortex wake of a wing (the vortex generator). The experimental results showed a marked asymmetry in the induced loads depending upon whether the vortex was above or below the wing. The wing was only three semispans behind the vortex generator, so the wake was not completely rolled up. From the computations of the roll up of the trailing vortex sheet of an elliptically loaded wing performed by Baker [4] and Moore [47], it is estimated that rolled up tip vortex would be expected to contain 90% of the bound circulation of the vortex generator at

that location.

Figure 3.31 and Figure 3.32 show the velocity vectors and total pressure

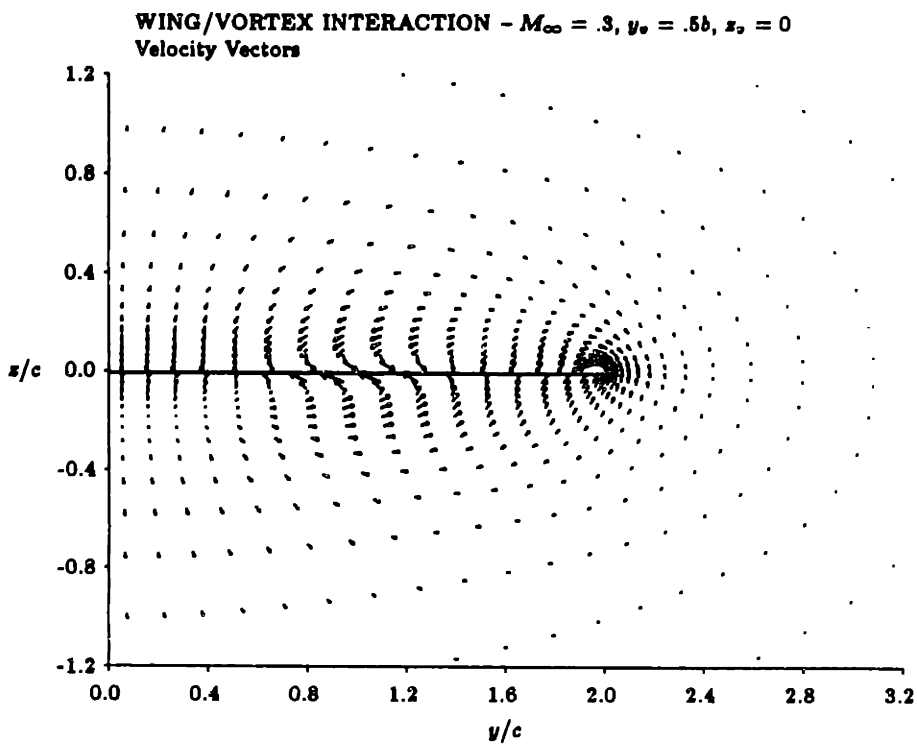

Figure 3.31: Velocity vectors in crossflow plane through wing midchord, $z/c = 0$, perturbation scheme

contours in a spanwise plane through the midchord of the wing. It is clearly seen that the vortex is split into two vortices of the same sense, one passing over the wing and the other passing under the wing. Note that the two vortices moving slightly in the spanwise direction, the upper vortex moving outboard and the lower vortex moving inboard. This is due to the effect of the image of each vortex in the wing.

Another factor affecting the computed results is the location at which

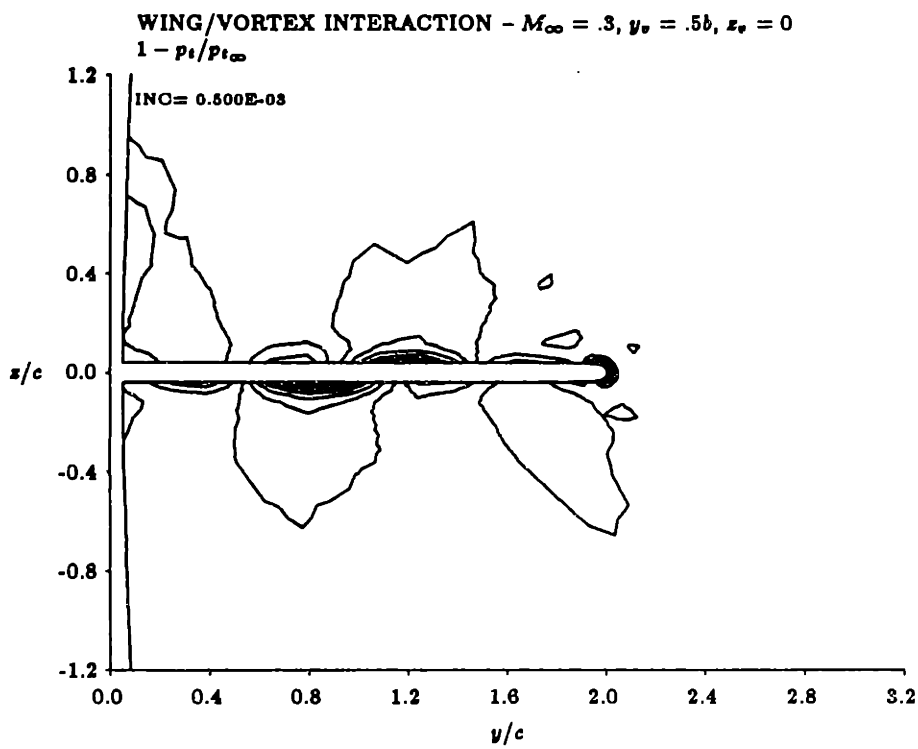

Figure 3.32: Total pressure contours in crossflow plane through wing mid-chord, $z/c = 0$, perturbation scheme

the prescribed flow residuals are turned off. For this case the prescribed flow residuals were specified up to a distance of two cells from the leading edge. It could be expected that the vortex core will increase in size as it approaches the wing due to the adverse pressure gradient in that region. A calculation was also performed in which $\mathbf{R}_0$ was set to zero approximately 1/4 chord upstream of the wing. This calculation is shown in Figure 3.33. The maximum download at the wing root and the upload outboard of the

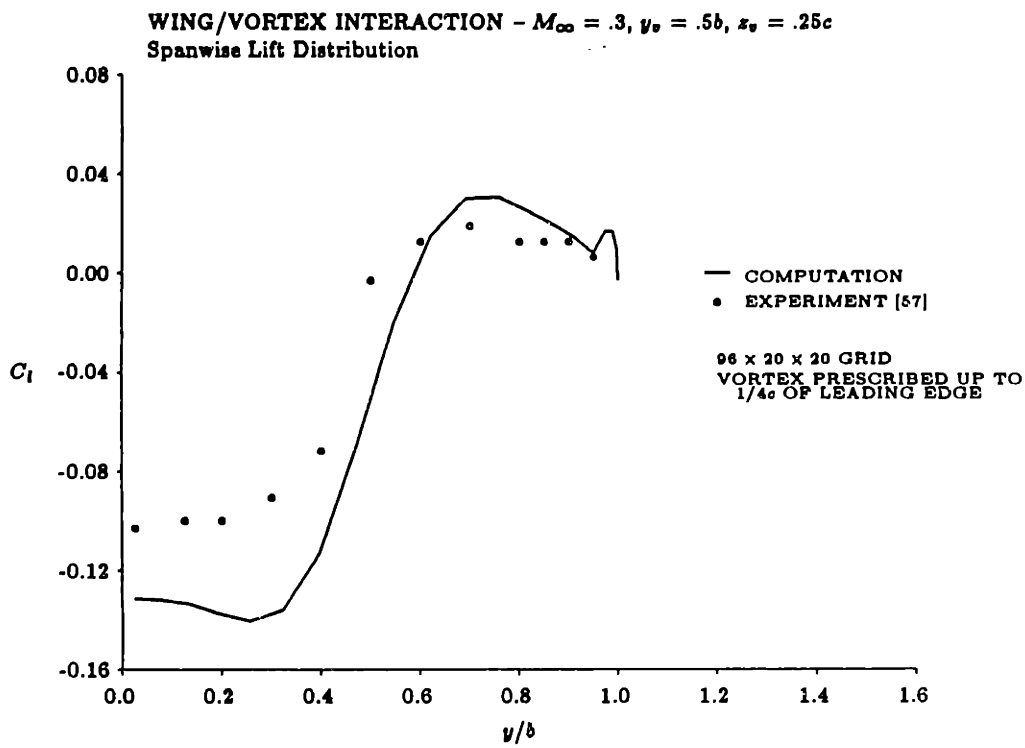

Figure 3.33: Comparison of computed and experimental spanwise lift distributions, $z/c = 0$, vortex "turned off" one quarter chord upstream

vortex are virtually unchanged. This could be expected on the grounds that the induced velocities at these locations depend upon the circulation,

which was unchanged. The gradient in the spanwise lift is slightly less steep than the calculation shown here, indicating an increase in the core size. However, this is may be due as much to numerical diffusion of the vortex in the region upstream of the wing as to the effect of the adverse pressure gradient immediately upstream of the wing.

Figure 3.34 shows the contours of lifting pressure coefficients, defined

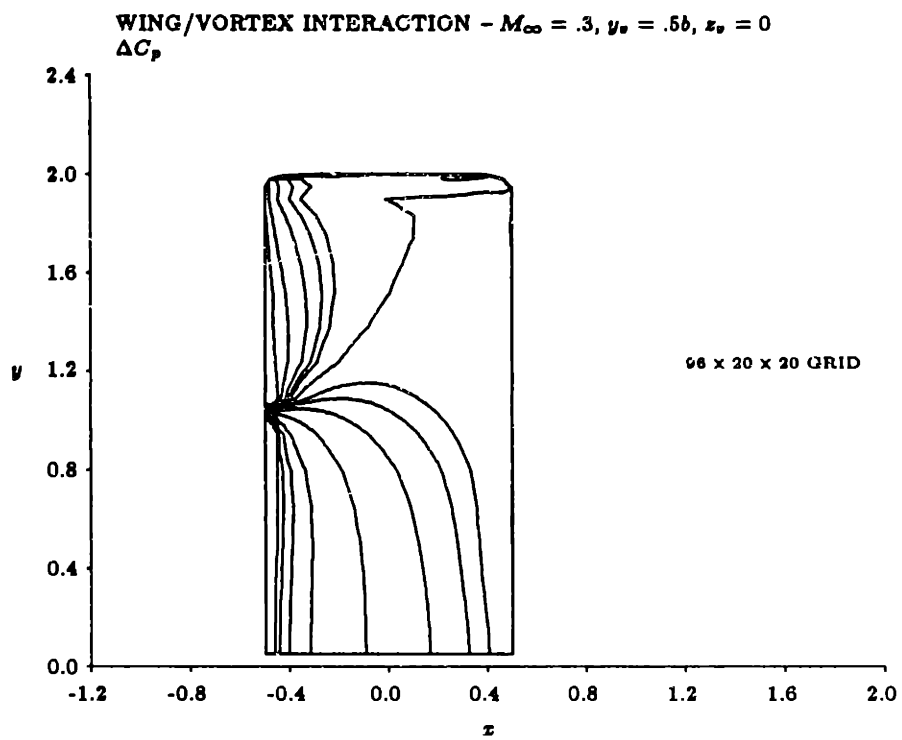

Figure 3.34: Lifting pressure coefficients, $\Delta p / \frac{1}{2}\rho u_\infty^2$, perturbation scheme

as $(p_{lower} - p_{upper}) / \frac{1}{2}\rho u_\infty^2$, on the wing as computed by the perturbation approach. The contour levels are identical to the experimental results of Figure 3.35. The agreement with the experimental data is very good. The location of the zero lift line is the same as in the experiment. The computed

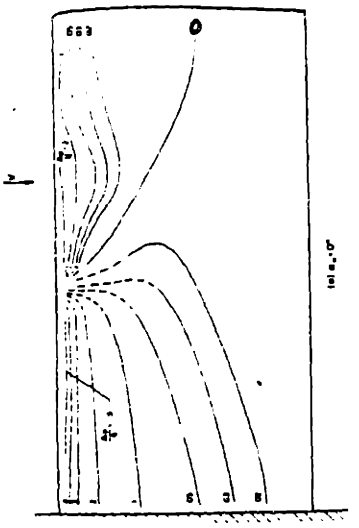

Figure 3.35: Lifting pressure coefficients, $\Delta p / \frac{1}{2}\rho u_\infty^2$, experiment

contours do not match the experimental contours near the root, which again is due to the computed download being larger than was measured. The contours near the leading edge show very similar behavior to that observed in the experiment. In contrast, the lifting pressure coefficients computed with the standard Euler scheme (Figure 3.36) are not even qualitatively similar

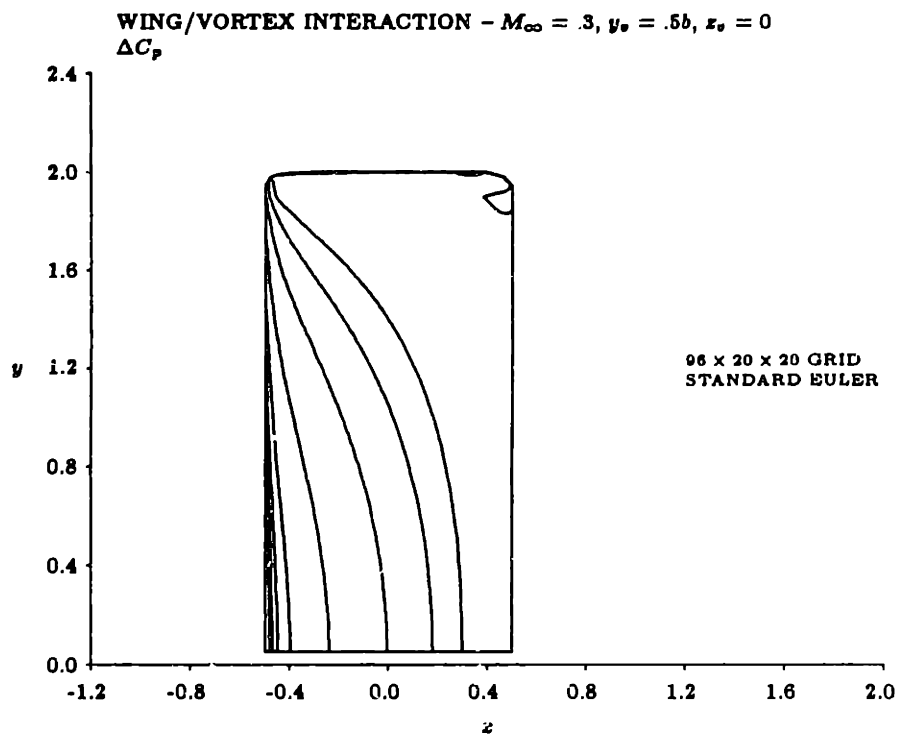

Figure 3.36: Lifting pressure coefficients, $\Delta p/\frac{1}{2}\rho u_\infty^2$, standard Euler scheme

to the experimental results. This case in particular shows the power of the prescribed vortex approach for the computation of wing/vortex interaction flows.

3.6 Summary

A perturbation scheme for the computation of a wing/vortex interaction has been developed. The perturbation scheme is shown to allow the vortex to be introduced into the computational domain without the problem of numerical diffusion of vorticity, and without requiring excessive grid resolution. Furthermore, by setting the prescribed flow residuals to zero in regions where the prescribed flow does not provide a good approximation to the local behavior of the flow, strong interactions between the wing and vortex can be computed. Comparisons of the computed wing/vortex interactions with experiment illustrate the accuracy of the perturbation scheme, and conversely the inability of the standard Euler solution algorithm to compute such flow reliably.

In the next chapter, the perturbation scheme developed here is combined with a free wake model to compute the flow about a helicopter rotor in hover.

Chapter 4

Hovering Rotor Solutions

In this chapter, a method for solving the flow around a hovering helicopter rotor is presented. The approach adopted couples the finite volume Euler solver presented in the previous chapters with a free wake model of the semi-infinite rotor wake. In the next section, the Euler equations in a rotating coordinate system are presented, along with the necessary modifications to the finite volume scheme to compute the flow around rotating blades. Next, the wake model is described, and the free wake iteration procedure is presented. The computation of the prescribed flow for the perturbation scheme applied to the rotor is then described, and the combined free wake/Euler iteration procedure is developed. Hovering solutions are performed and compared to the experimental data of Ballard, Orloff & Luebs [5] and Caradonna & Tung [14].

4.1 Euler Equations in Rotating Coordinates

In hovering flight, the aerodynamic loads on a helicopter rotor are steady. Thus in a frame of reference attached to the rotor blades, the flow field may be assumed steady, and the Euler equations may be solved in a time asymptotic fashion as described in chapter 2. Some modifications to the scheme are required when transforming the Euler equations into a rotating

coordinate frame, and these changes are discussed below.

The equations of motion are re-written in a coordinate frame attached to the rotor, shown in Figure 4.1. In this coordinate system, x points in

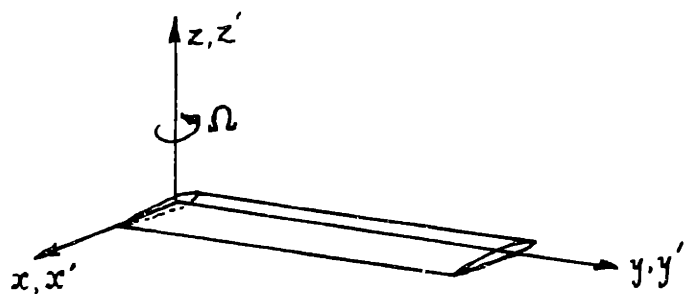

Figure 4.1: Rotor blade coordinate system

the chordwise direction (from leading to trailing edge), y is the spanwise direction, and z lies along the axis of rotation. The inertial coordinates x', y', and z' are taken to coincide with x, y, and z at an instant in time t. Let $\vec{u}'$ be the velocity measured in the inertial reference frame and $\vec{u}$ be the velocity relative to the rotor blade. The velocities $\vec{u}'$ and $\vec{u}$ are related by the equation

$$\vec{u}' = \vec{u} + \vec{\Omega} \times \vec{x},$$

where $\vec{\Omega}$ is the angular velocity of the rotor and $\vec{\Omega} = \hat{k}\Omega$.

With these definitions, a coordinate transformation from the inertial to the rotating coordinates yields the following form of the Euler equations:

$$\frac{\partial}{\partial t}\iiint_V \mathbf{U}\, d^3x + \iint_{\partial V} \vec{\mathbf{F}}(\mathbf{U}) \cdot \hat{\mathbf{n}}\, d^2x + \iiint_V \mathbf{S}(\mathbf{U})\, d^3x = 0, \qquad (4.1)$$

where

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E_r \end{pmatrix}, \qquad \vec{\mathbf{F}}(\mathbf{U}) = \begin{pmatrix} \rho\vec{u} \\ \rho u\vec{u} + p\hat{i} \\ \rho v\vec{u} + p\hat{j} \\ \rho w\vec{u} + p\hat{k} \\ \rho\left(E_r + p\right)\vec{u} \end{pmatrix},$$

$$\mathbf{S}(\mathbf{U}) = \begin{pmatrix} 0 \\ -\rho\left(\Omega^2 x + 2\Omega v\right) \\ -\rho\left(\Omega^2 y - 2\Omega u\right) \\ 0 \\ 0 \end{pmatrix},$$

and

$$E_r = \frac{1}{\gamma - 1}\frac{p}{\rho} + \frac{\vec{u}\cdot\vec{u} - \Omega^2\left(x^2 + y^2\right)}{2}.$$

The control volume V is fixed in the rotating reference frame. The equations in the rotating frame differ from the equations for the fixed wing case by the addition of the terms $\mathbf{S}(\mathbf{U})$ and the replacement of the total energy E by the quantity E_r. The source terms in the x and y momentum equations represent the centrifugal and Coriolis accelerations. The term E_r in the energy equation is analogous to the total energy in an inertial frame of reference, and the quantity

$$H_r = E_r + \frac{p}{\rho}, \tag{4.2}$$

which is called the total rothalpy is analogous to the total enthalpy in the inertial frame. For a flow which is steady in the rotating reference frame, the condition of constant H_r replaces the condition of constant H which occurs in the fixed wing case.

Using the definitions for $\vec{u}$ and $\vec{u}'$, the Equations (4.1) may be rewritten in terms of the absolute rather than the relative velocity components:

$$\frac{\partial}{\partial t}\iiint_V \mathbf{U}_a\, d^3x + \iint_{\partial V} \vec{\mathbf{F}}(\mathbf{U}_a)\cdot\hat{\mathbf{n}}\, d^2x + \iiint_V \mathbf{S}(\mathbf{U}_a)\, d^3x = 0, \tag{4.3}$$

where

$$\mathbf{U}_a = \begin{pmatrix} \rho \\ \rho u' \\ \rho v' \\ \rho w' \\ \rho E_r \end{pmatrix}, \qquad \vec{\mathbf{F}}(\mathbf{U}_a) = \begin{pmatrix} \rho \vec{u}' \\ \rho u' \vec{u} + p\hat{i} \\ \rho v' \vec{u} + p\hat{j} \\ \rho w' \vec{u} + p\hat{k} \\ \rho \left(E_r + p\right) \vec{u} \end{pmatrix},$$

$$\mathbf{S}(\mathbf{U}_a) = \begin{pmatrix} 0 \\ -\rho\Omega v' \\ \rho\Omega u' \\ 0 \\ 0 \end{pmatrix},$$

and E_r is rewritten in the form

$$E_r = \frac{1}{\gamma - 1} \frac{p}{\rho} + \frac{\vec{u}' \cdot \vec{u}' - 2\vec{u}' \cdot \vec{\Omega} \times \vec{x}}{2}.$$

The latter form of the Euler equations is preferable to Equation (4.1) for a couple of reasons. One is that the discrete flux integral is approximated by averaging the absolute velocity components to the cell faces and evaluating the coordinate rotation term $\vec{\Omega} \times \vec{x}$ at each cell face to get the relative velocity $\vec{u}$. This should be more accurate in the far field where the grid is stretched, since the coordinate rotation component is evaluated at the cell face rather than averaged between the cells. The second advantage is that the artificial viscosity is applied only to the absolute momentum components in the x and y momentum equations. Again, this is particularly important in the far field where the grid is stretched and the $\vec{\Omega} \times \vec{x}$ component will vary rapidly between adjacent cells.

The non-dimensionalization of the equations is the same as described in chapter 2. That is, the density and pressure are normalized by their values at infinity, ρ_∞ and p_∞, the velocity is normalized by $a_\infty/\sqrt{\gamma}$, lengths by the blade chord c, and time by $c\sqrt{\gamma}/a_\infty$. This choice of non-dimensionalization yields the following definition for the angular velocity

$$\Omega = \frac{\sqrt{\gamma} M_{tip}}{R}, \tag{4.4}$$

where M_{tip} is the tip Mach number and R is the non-dimensional rotor radius, or rotor blade aspect ratio. Thus the tip Mach number enters the problem through the definition of the angular velocity.

The basic algorithm for solving Equation (4.3) is the same as that presented in chapters 2 and 3 for the fixed wing cases, with the necessary modifications for the differences in the equations. Only these differences are described below.

4.1.1 Flux Integral Evaluation

The discrete approximation to the flux integral is essentially the same as Equation (2.10) in chapter 2. That is, the flux vector is computed at two adjacent cells and then averaged to get the cell face value. However, to compute the flux vector, both the absolute velocity $\vec{u}'$ and the relative velocity $\vec{u}$ are needed; $\vec{u}'$ is found from the state vector $\mathbf{U}_a$, but $\vec{u}$ is not stored. The relative velocity is easily obtained from the absolute velocity by use of the equation $\vec{u} = \vec{u}' - \vec{\Omega} \times \vec{x}$. To form the flux vector from each cell, the coordinate rotation term is evaluated at the cell face, rather than the cell center, and subtracted from the absolute velocity. This allows a more accurate evaluation of the fluxes in regions where the grid undergoes rapid stretching, such as in the far field.

In addition to the flux integral, the centrifugal and Coriolis terms must also be computed. The terms $\mathbf{S}(\mathbf{U}_a)$ are easily evaluated at each cell and are multiplied by the cell volume to approximate the volume integral. These source terms are added to the flux integral residual $\mathbf{R}$.

4.1.2 Artificial Viscosity

The only change in the artificial viscosity from the fixed wing case is the replacement of the total enthalpy H with the total rothalpy H_r in the energy equation. The dissipation terms are otherwise evaluated exactly as in the

fixed wing case, including the treatment at the boundaries. The symmetry condition at the wing root is replaced with a periodicity condition for the rotating blade; this is described below.

4.1.3 Solid Wall Boundary Conditions

As in the fixed wing case, all that is needed at the rotor blade surface is the pressure, which is found by extrapolation from the interior using the normal momentum equation. The normal momentum equation differs from the fixed wing case due to the addition of the centrifugal and Coriolis terms. Since the normal component of the relative, rather than the absolute, velocity is zero at a solid wall, it is more convenient to work with the momentum equation written using the relative velocity $\vec{u}$.

The momentum equation in nonconservative form is

$$\rho \frac{\partial \vec{u}}{\partial t} + \rho \vec{u} \cdot \nabla \vec{u} + \nabla p + \rho \vec{\Omega} \times \left(\vec{\Omega} \times \vec{x} + 2\vec{u} \right) = 0. \tag{4.5}$$

The velocity $\vec{u}$ is obtained from the absolute velocity $\vec{u}'$ by evaluting the coordinate rotation velocity $\vec{\Omega} \times \vec{x}$ at the solid surface and subtracting it from $\vec{u}'$. Dotting this equation into the unit normal at the surface, $\hat{n}$, and using the fact that $\vec{u} \cdot \hat{n} = 0$ and $\partial \hat{n} / \partial t = 0$ yields

$$(\rho \vec{u} \cdot \nabla \hat{n}) \cdot \vec{u} = \frac{\partial p}{\partial n} + \rho \vec{\Omega} \times \left(\vec{\Omega} \times \vec{x} + 2\vec{u} \right) \cdot \hat{n}. \tag{4.6}$$

The term on the left hand side is evaluated in the same way as for the fixed wing; the centrifugal and Coriolis terms are evaluated using the coordinates at the surface and the tangential component of the relative velocity there.

4.1.4 Far Field Boundary Conditions

If the Euler equations are written in terms of the relative velocity and a one dimensional analysis like that described for the far field boundary conditions in chapter 2 is performed, the characteristic variables are identical to

those found in the fixed wing case. That is, the transformation to rotating coordinates does not alter the eigenvalues and eigenvectors of the one dimensional equations, and the similarity tranform is unchanged. Thus the same characteristic boundary conditions as those presented in chapters 2 and 3 are used. These require the specification or extrapolation of the Riemann invariants, entropy, and tangentiai velocity components at the boundary. To evaluate the characteristic variables, the relative velocity components are needed. The far field velocity is

$$\vec{u}_{ff} = \vec{u}'_{wake} - \vec{\Omega} \times \vec{x}, \tag{4.7a}$$

where $\vec{u}'_{wake}$ is the induced velocity field of the semi-infinite vortex wake and $\vec{\Omega} \times \vec{x}$ is evaluated on the boundary. The evaluation of the $\vec{u}'_{wake}$ term will be described in section 3. The extrapolated velocity component is

$$\vec{u}_{ex} = \vec{u}'_{ex} - \vec{\Omega} \times \vec{x}, \tag{4.7b}$$

where $\vec{u}'_{ex}$ is the absolute velocity at the first cell inside the boundary. These two velocity components are used to evaluate the Riemann invariants and the tangential velocity at the boundary. With these invariants, the normal velocity component and the speed of sound are evaluated as before. At an inflow boundary, the specification of the tangential velocity gives the absolute velocity as

$$\vec{u}' = \vec{u}_{ff} + (u_n - \vec{u}_{ff} \cdot \hat{n})\,\hat{n} + \vec{\Omega} \times \vec{x}, \tag{4.8a}$$

and at an outflow point,

$$\vec{u}' = \vec{u}_{ex} + (u_n - \vec{u}_{ex} \cdot \hat{n})\,\hat{n} + \vec{\Omega} \times \vec{x}. \tag{4.8b}$$

The entropy is either specified at an inflow or extrapolated at an outflow boundary exactly as for the fixed wing case. The entropy at the inflow is not necessarily uniform, since it varies through a vortex core. The specification of the entropy distribution in the core is done as for the fixed wing

perturbation approach of the previous chapter. This point will be taken up in more detail in the section on the coupling of the Euler solver with the free wake code.

4.1.5 Periodic Boundary

For a fixed wing, the coordinate surface at the wing root was a symmetry boundary. At a rotor blade root, this is not the case. Instead, there is rotational symmetry (Figure 4.2). To handle this, the coordinate surface

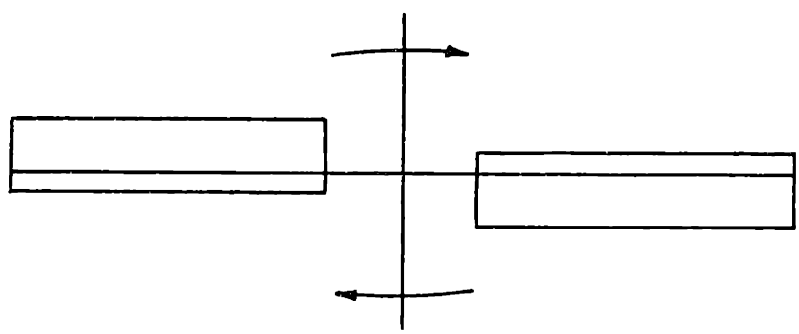

Figure 4.2: Rotational symmetry at the blade root

at the blade root must be symmetrical about the axis of rotation to allow periodic boundary conditions to be applied (Figure 4.3). In this thesis, only two bladed rotors are treated, but the extension to more blades is straightforward. To generate the periodic boundary at the blade root, the airfoil section there is taken to be an ellipse with its major axis lying along the x-direction (normal to the axis of rotation) and the minor axis lying along the z-direction (the axis of rotation). Care is taken so that the grid is

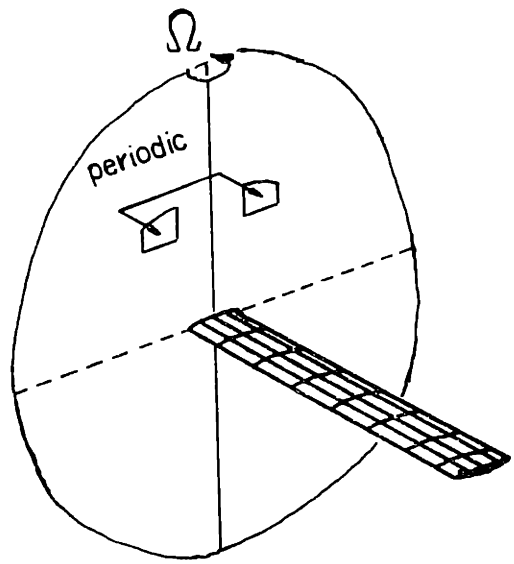

Figure 4.3: Periodic boundary condition at the blade root

symmetric about the z-axis (Figure 4.4). Periodic boundary conditions are then applied in an obvious fashion to both the flux integral and artificial viscosity operators. Although the grid at the blade root does not match a real rotor hub geometry, the error is small since little lift is produced near the root of the blade, and it does have the advantage of satisfying the rotational symmetry condition exactly.

4.1.6 Temporal Integration

There is no change to the multistage time stepping scheme presented in chapter 2. The same multistage coefficients are used and the time step restriction applies. The time step is based on the relative rather than the absolute velocity at each cell. Enthalpy damping is still used, but in the rotating frame of reference it becomes rothalpy damping; that is, the total enthalpy is replaced by the total rothalpy, and the damping is driven by the difference in the local total rothalpy and its uniform steady state value.

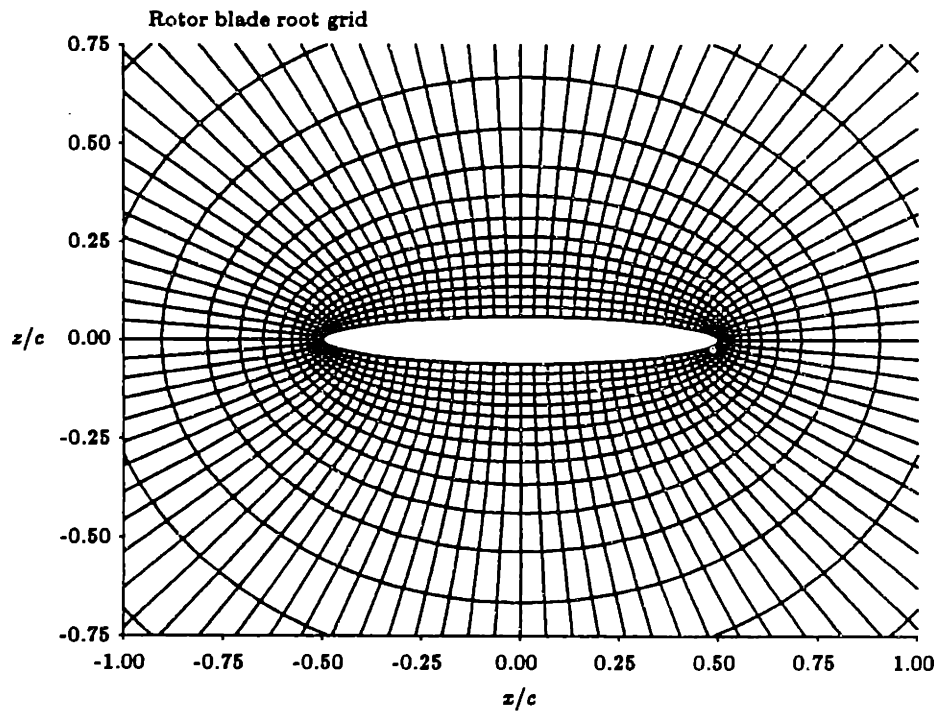

Figure 4.4: Grid at rotor blade root

It is applied exactly as in the fixed wing cases, as an implicit update after a complete multistage integration. The rothalpy damping is desirable for the same reasons as in the fixed wing solutions, in that one series of cases presented below are at a very low tip Mach number. Even for a high tip Mach number, the flow at inboard stations of the rotor is highly subsonic, and the rothalpy damping is useful in accelerating convergence.

In the next section of the chapter, the wake model is described and the free wake iteration procedure is explained. The coupling of the wake solution to the Euler solver will be described in section 3.

4.2 Wake Model

The model for the hovering rotor wake was developed by Miller [45]. The model is based on Helmholtz's vortex theorems for an inviscid, incompressible flow, which state that vortex lines must follow streamlines of the flow. By representing the semi-infinite helical wake as discrete vortex filaments, the geometry of the wake is found by iteratively solving for the force free positions of the filaments. Miller made several simplifying assumptions to yield a very fast solution procedure while still capturing the physics of the wake flow. The model and solution procedure are described here.

The helical vortex wake of a hovering rotor is shown in Figure 4.5. The

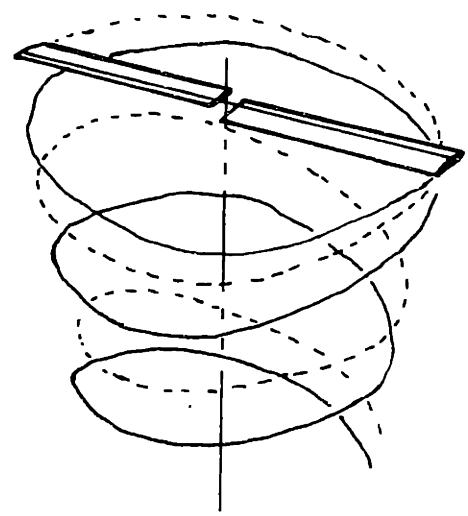

Figure 4.5: Vortex wake of a hovering rotor

trailing vortex wake from each blade is modeled as discrete filaments. Five filaments are used; one to model the tip vortex, and four to represent the inboard portion of the vortex sheet. The choice of five vortices is based on earlier studies of the wake model (Roberts & Murman [54] and Miller et al. [46]) in which it was demonstrated that this provides an adequate

representation of the helical wake. The manner in which the circulation of each filament is set is described in the next section.

Let u_r, u_ψ, and u_z be the radial, azimuthal, and axial velocity components in the frame of reference of the rotor (Figure 4.6), and n is the filament

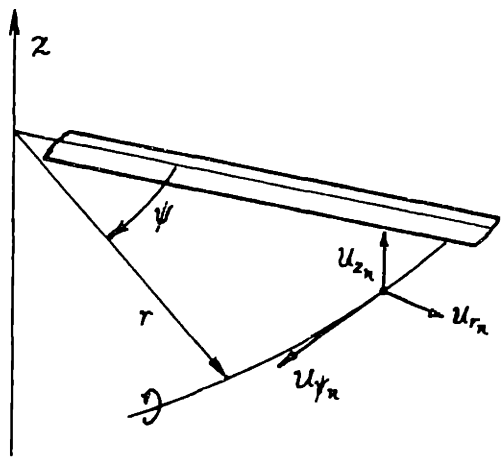

Figure 4.6: Vortex velocity components

index. The requirement that each filament lies along a streamline yields the following equations for the vortex trajectories:

$$\frac{u_{rn}}{dr_n} = \frac{u_{\psi_n}}{r_n d\psi} = \frac{u_{zn}}{dz_n}. \tag{4.9}$$

The essence of the free wake procedure is the solution of the Equations (4.9) for the N vortex filaments. Several approximations are made to simplify the solution procedure. First, the contribution of the rotor blade bound circulation to the induced velocity in the wake is neglected, as the contribution of each blade cancels when averaged over the azimuth. Next, the velocity perturbations in the azimuthal direction are ignored, which allows u_ψ to be written as

$$u_{\psi_n} = r_n \Omega. \tag{4.10}$$

Substituting into Equations (4.9) yields

$$dr_n = \frac{u_{rn}}{\Omega} d\psi, \tag{4.11a}$$

$$dz_n = \frac{u_{zn}}{\Omega} d\psi. \tag{4.11b}$$

Neglecting the azimuthal velocity perturbations is equivalent to treating the flow in a given azimuthal plane to be axisymmetric. This allows the helical vortex filaments to be replaced by vortex rings for the purposes of computing the induced velocities. A further simplification is made by computing the velocities and solving for the position of the vortices only in the azimuthal plane containing the rotor blade, $\psi = 0$. The vortex rings are constructed by taking 180° segments of the helical filaments from each blade and replacing them with vortex rings at their mean locations, as shown in Figure 4.7. The

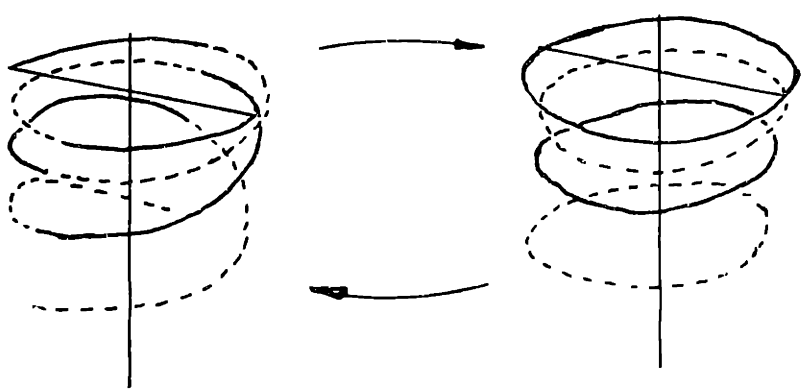

Figure 4.7: Formulation of vortex ring model

first series of rings replaces the segment of each helical filament from $\psi = 90°$ to $\psi = 270°$ from the two blades; the next series of rings is made from the $\psi = 450°$ to $\psi = 630°$ segments, etc. The first quarter spiral of the filaments

from each blade, $\psi = 0$ to $\psi = 90°$ still remain—their contribution to the induced velocities below the rotor blade may be computed by replacing the two quarter spirals of each filament with a single ring of half the circulation of the filament. The location of these rings are fixed in the rotor plane, and are not found as part of the free wake iteration procedure.

Since the wake has an infinite extent in the axial direction below the blade, the position of each filament cannot be determined throughout the wake. The force free position of each filament is found only for the first 720° of its age, or four passes below the rotor blade. After that point, the remaining semi-infinite wake is modeled by two vortex cylinders, one to represent the tip vortex and one to represent the entire inboard portion of the wake. The strength of the tip vortex cylinder, $\frac{d\Gamma}{dz}$, is found by dividing the tip vortex circulation Γ by the axial separation between the last two free tip vortex spirals, Δz (Figure 4.8). The radius of the vortex cylinder is

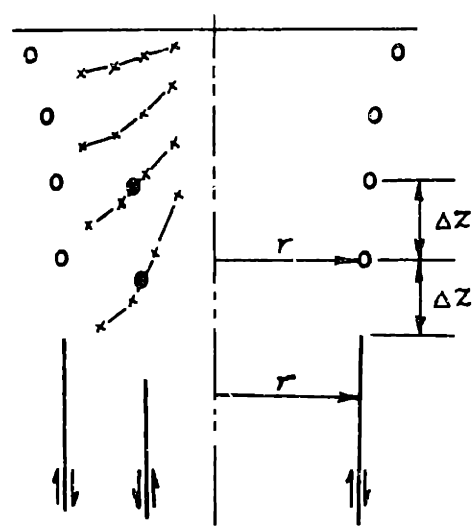

Figure 4.8: Vortex cylinder far wake representation

fixed at the radius of the tip vortex at its fourth passage below the blade,

and it begins a distance Δz below the tip vortex spiral. To set the strength of the cylinder modeling the inboard portion of the trailing vortex sheet, the centroid of the four filaments representing the free portion of the sheet are lumped at their centroids for the last two passes below the blade. The cylinder strength and location is determined as for the tip vortex cylinder, based on the circulation and location of the vortex centroids (Figure 4.8).

With this wake model, the determination of the wake geometry requires the solution of Equations (4.11a) and (4.11b). The determination of the velocities at the vortex locations is discussed first, and then the iterative solution procedure is described.

4.2.1 Velocity Computation

Since the wake model assumes an incompressible, inviscid, irrotational flow with embedded vorticity, the velocities in the wake are found by summing the induced velocities of all the vortex rings and cylinders used to model the wake. The induced velocity of each ring is found by using the Biot-Savart law. The formulas for the induced velocity of a vortex ring and a vortex cylinder are given by Miller in reference [45]; they are repeated here for completeness.

By applying the Biot-Savart law to a vortex ring of radius r and circulation Γ, the radial and axial components of the induced velocity at a radius η and axial distance z from the plane of the ring (Figure 4.9) are given by the following integrals:

$$u_r = \frac{\Gamma}{4\pi} \int_0^{2\pi} \frac{zr\cos\psi\, d\psi}{(\eta^2 + r^2 + z^2 - 2r\eta\cos\psi)^{\frac{3}{2}}}, \tag{4.12a}$$

$$u_z = \frac{\Gamma}{4\pi} \int_0^{2\pi} \frac{r(r - \eta\cos\psi)\, d\psi}{(\eta^2 + r^2 + z^2 - 2r\eta\cos\psi)^{\frac{3}{2}}}. \tag{4.12b}$$

The integrals in Equations (4.12a) and (4.12b) may be expressed in terms

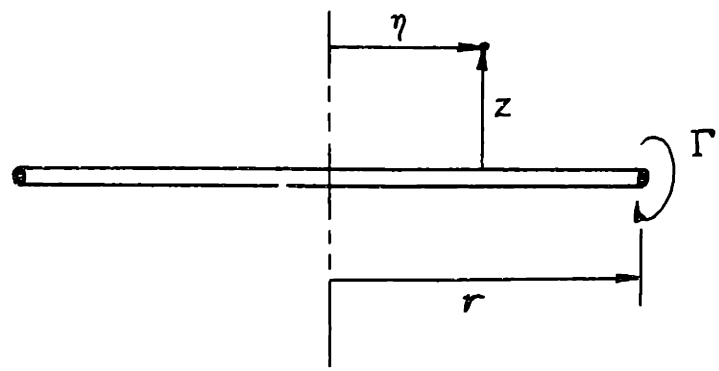

Figure 4.9: Coordinates for vortex ring induced velocity calculation

of complete elliptic integrals of the first and second kind,

$$K = \int_0^{\frac{\pi}{2}} \frac{d\psi}{\left(1 - k^2 \sin^2 \psi\right)^{\frac{1}{2}}}, \tag{4.13a}$$

$$E = \int_0^{\frac{\pi}{2}} \left(1 - k^2 \sin^2 \psi\right)^{\frac{1}{2}} d\psi, \tag{4.13b}$$

where

$$k^2 = \frac{4r\eta}{(r+\eta)^2 + z^2}$$

is the argument of the elliptic integrals K and E. With these definitions, the velocity components u_r and u_z may be written in the following form:

$$u_r = \frac{\Gamma}{4\pi} \frac{z}{2\eta} \sqrt{\frac{k^2}{r\eta}} \left\{ E \frac{2 - k^2}{1 - k^2} - 2K \right\}, \tag{4.14a}$$

$$u_z = \frac{\Gamma}{4\pi} \sqrt{\frac{k^2}{r\eta}} \left\{ K - E \left[\frac{1 - \frac{k^2}{2}\left(1 + \frac{r}{\eta}\right)}{(1 - k^2)} \right] \right\}. \tag{4.14b}$$

The induced velocity of the far wake vortex cylinders must also be computed at each free vortex filament location. The radial and axial velocity components of a vortex cylinder are given by

$$u_{rcyl} = \frac{1}{4\pi}\frac{d\Gamma}{dz}\int_{z}^{\infty}\int_{0}^{2\pi}\frac{zr\cos\psi d\psi dz}{\left(\eta^2+r^2+z^2-2z\eta\cos\psi\right)^{\frac{3}{2}}}, \tag{4.15a}$$

$$u_{zcyl} = \frac{1}{4\pi}\frac{d\Gamma}{dz}\int_{z}^{\infty}\int_{0}^{2\pi}\frac{r\left(r-\eta\cos\psi\right)d\psi dz}{\left(\eta^2+r^2+z^2-2r\eta\cos\psi\right)^{\frac{3}{2}}}. \tag{4.15b}$$

Here, z is the axial distance from the end of the cylinder to the point at which the velocity is to be evaluated, r is the cylinder radius, and η is the radius at the point of evaluation. Integrating the above equations over z gives

$$u_{rcyl} = \frac{1}{4\pi}\frac{d\Gamma}{dz}\int_{0}^{2\pi}\frac{r\cos\psi d\psi}{\left(\eta^2+r^2+z^2-2r\eta\cos\psi\right)^{\frac{1}{2}}}, \tag{4.16a}$$

$$u_{zcyl} = \frac{1}{4\pi}\frac{d\Gamma}{dz}\int_{0}^{2\pi}\frac{r\left(r-\eta\cos\psi\right)}{\eta^2+r^2-2r\eta\cos\psi}\left[1-\frac{z}{\left(\eta^2+r^2+z^2-2r\eta\cos\psi\right)^{\frac{1}{2}}}\right]d\psi. \tag{4.16b}$$

Equation (4.16a) can be rewritten in term of the complete elliptic integrals of the first and second kind,

$$u_{rcyl} = \frac{1}{2\pi k}\frac{d\Gamma}{dz}\sqrt{\frac{r}{\eta}}\left[K\left(2-k^2\right)-2E\right], \tag{4.17}$$

where k, K, and E are as previously defined. The equation for the axial component of induced velocity of the vortex cylinder, Equation (4.16b) is left unchanged for now.

To evaluate the elliptic integrals, the Cayley series solution (reference [15] are used. These series are rapidly convergent, and only a few terms need to be retained. The series for K and E are:

$$\begin{aligned} K \;=\; & F+\left(\frac{1}{2}\right)^2(F-1)\left(1-k^2\right)+\left(\frac{1\cdot 3}{2\cdot 4}\right)^2\left(F-1-\frac{2}{3\cdot 4}\right)\left(1-k^2\right)^2 \\ & +\left(\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\right)^2\left(F-1-\frac{2}{3\cdot 4}-\frac{2}{5\cdot 6}\right)\left(1-k^2\right)^3+\cdots, \end{aligned} \tag{4.18a}$$

$$E = 1 + \frac{1}{2}\left(F - \frac{1}{2}\right)\left(1 - k^2\right) + \left(\frac{1}{2}\right)^2 \frac{3}{4}\left(F - \frac{2}{2} - \frac{1}{3 \cdot 4}\right)\left(1 - k^2\right)^2 + \left(\frac{1 \cdot 3}{2 \cdot 4}\right)^2 \frac{5}{6}\left(F - \frac{2}{2} - \frac{2}{3 \cdot 4} - \frac{1}{5 \cdot 6}\right)\left(1 - k^2\right)^3 + \cdots, \quad (4.18b)$$

where

$$F = \ln \frac{4}{\sqrt{1 - k^2}}.$$

In the current implementation, the first four terms of the series are retained; this choice was determined through numerical experimentation. The expression for the axial component of velocity due to the cylinder, Equation (4.16b), cannot be expressed solely in terms of complete elliptic integrals of the first and second kind, but contains elliptic integrals of the third kind. For this reason, it is more convenient to evaluate the velocity by numerical integration of Equation (4.16b) using the trapezoidal rule. Again through numerical experiments, it was found that eighty integration steps in the ψ direction gave sufficient accuracy. Care must be taken when $\eta = r$, since the denominators of both fractions in the integrand vanish at the endpoints of the integration, $\psi = 0$ and $\psi = 2\pi$. However, the integrand has a well defined limit of zero as $\psi \to 0, 2\pi$. The integrand is set equal to zero at the endpoints when $r = \eta$, and the expression is numerically evaluated for each of the remaining steps.

It should also be noted that the expressions for the induced velocity of a vortex ring are singular at the ring location itself ($\eta = r$, $z = 0$), and thus cannot be used to evaluate the self-induced velocity of the ring. As given by the above formulas, the self-induced velocity of a ring is infinite, which is physically unacceptable because it corresponds to infinite kinetic energy. This behavior is a result of the assumption that the vortex is concentrated onto a filament of zero thickness. In reality, the vortex filament will have a finite core radius due to the action of viscosity, and this will result in a finite but non zero self-induced velocity. The self-induced velocity of a vortex ring with a Rankine core structure and a rotational core radius a is given by

Lamb [36], p. 241, as

$$u_{z\,ind} = \frac{\Gamma}{4\pi r}\left(\ln\frac{8r}{a} - \frac{1}{4}\right). \tag{4.19}$$

This formula is used to compute the self-induced velocity of the vortex ring. The value of the core radius a is chosen, following Miller, to be $0.01R$, where R is the radius of the rotor. This choice of the core size is based on the theory of Landahl [37], and has been extended by Chung [17]. Since the self-induced contribution is logarithmically singular, it is not too sensitive to the exact value of a within the given range. For the cases presented in this chapter, the above value gives a core radius of the order of 7% to 15% of the rotor blade chord c. The value used was chosen to be $0.15c$ for all the cases presented below. (The reason for choosing a value of a in terms of the blade chord rather than the rotor radius is due to the fact that the lengths were non-dimensionalized by the chord rather than the radius in the code. It is thus more convenient to refer the value of a to the chord.)

Using the above formulas for the induced velocity of each vortex ring and cylinder in the wake, the total induced velocity at each vortex location is determined. These velocities are used to solve the vortex trajectory Equations (4.11a) and (4.11b). The iteration procedure for determining the vortex positions is described next.

4.2.2 Wake Iteration Procedure

The position of the wake vortices is determined only in the azimuthal plane $\psi = 0$. The equations of the vortex trajectories, Equations (4.11a) and (4.11b), are solved iteratively. Let N be the number of helical vortex filaments used to model the wake, and M be the number of passes of the wake below the blade; for all the cases in this thesis, $N = 5$ and $M = 4$. The total number of vortex rings used to model the free wake is then $N \times M$. In addition, there are N vortex rings in the tip path plane of the rotor, whose positions are fixed, and two far wake vortex cylinders (Figure 4.10). The

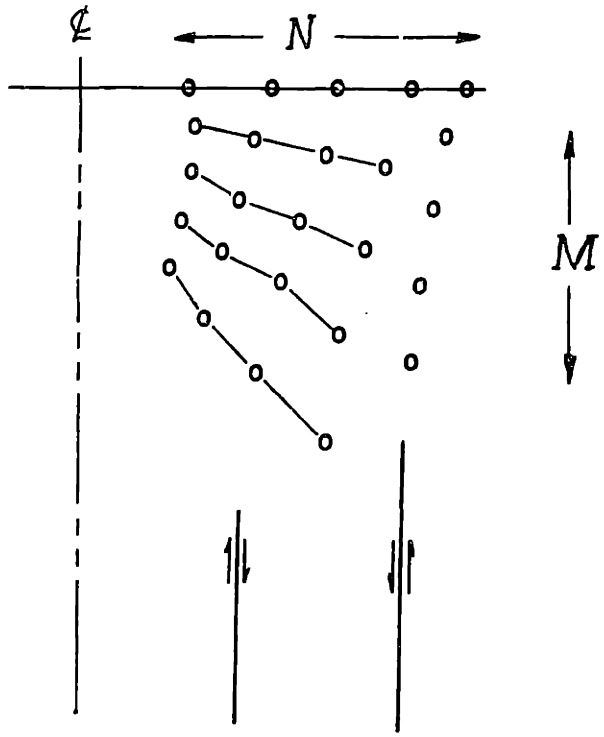

Figure 4.10: Complete wake model

positions of the vortex rings in the wake are not known a priori, and an initial guess at their locations is made. For the first run of the free wake

code within the combined Euler/free wake iteration procedure, the initial guess has been taken to be a solution given by the lifting line/free wake solver developed by Roberts & Murman for the given rotor. This geometry is specified in the input file of the code. The choice of initial guess is not too critical, and a uniform axial spacing of the wake vortices would work just as well. For subsequent runs of the free wake solver, the initial guess for the geometry is taken to be the previously converged wake geometry.

If n is the index of the vortex ring, then the position of the $(n+N)^{th}$ vortex ring is found by integrating the trajectory equations (4.11a) and (4.11b) over the interval $\psi = 0$ to π to get

$$\begin{aligned} r_{n+N}^{new} &= r_n^{old} + \int_0^{\pi} \frac{u_r^{old} d\psi}{\Omega} \\ &= r_n^{old} + \delta r_{n+N}, \end{aligned} \tag{4.20a}$$

$$\begin{aligned} z_{n+N}^{new} &= z_n^{old} + \int_0^{\pi} \frac{u_z^{old} d\psi}{\Omega} \\ &= z_n^{old} + \delta z_{n+N}. \end{aligned} \tag{4.20b}$$

The superscripts *old* and *new* refer to quantities evaluated before and after the vortex wake geometry is updated, respectively. When the wake has converged, the old and new values of the geometry and the vortex velocities are the same. The integrals in Equations (4.20a) and (4.20b) are approximated by taking the average velocity between the vortex locations n and $n+N$. This yields the equations for the change in the vortex location,

$$\delta r_{n+N} = \frac{\pi}{\Omega} \frac{u_{r_n}^{old} + u_{r_{n+N}}^{old}}{2}, \tag{4.21a}$$

$$\delta z_{n+N} = \frac{\pi}{\Omega} \frac{u_{z_n}^{old} + u_{z_{n+N}}^{old}}{2}, \tag{4.21b}$$

The position of the $(n+N)^{th}$ vortex ring given by Equations (4.20a) and (4.20b) will not in general correspond to the original position. The new

position computed from the trajectory equations is used obtain a new wake geometry. However, the trajectory equations form an ill-conditioned system, and taking the positions given by Equations (4.20a) and (4.20b) does not yield a stable iteration procedure. Underrelaxation is required, meaning a weighted average of the current vortex position and predicted vortex position must be used to obtain the new vortex positions. The resulting equations for updating the wake geometry are

$$r_{n+N}^{new} = (1 - \omega_w)\, r_{n+N}^{old} + \omega_w \left(r_n^{old} + \delta r_{n+N} \right), \tag{4.22a}$$

$$z_{n+N}^{new} = (1 - \omega_w)\, z_{n+N}^{old} + \omega_w \left(z_n^{old} + \delta z_{n+N} \right), \tag{4.22b}$$

where ω_w is the underrelaxation parameter, and is generally taken to be 0.2. It is easily seen that when $\omega_w = 1$, Equations (4.20a) and (4.20b) are recovered. After the positions of the free vortex rings are found, the locations and strengths of the two far wake vortex cylinders are recomputed, and the next iteration starts.

To summarize the iteration procedure for determining the wake geometry:

1. the velocity at each of the $N \times M$ free vortex postions is determined by summing the induced velocities of all the vortex rings and cylinders in the wake;

2. the trajectory equations (4.21a) and (4.21b) are integrated;

3. the position of each free vortex ring is updated using Equations (4.22a) and (4.22b);

4. the far wake vortex cylinder locations and strengths are recomputed using the new wake geometry;

5. the iteration is continued until the geometry converges.

No convergence criterion has been developed for the wake geometry iteration procedure, as it has been found more convenient to run a fixed number of iterations. For all the cases presented below, 200 iterations of the free wake algorithm have been used for each free wake solution. The average change in vortex position after 200 iterations is usually on the order of 10^{-6} of the rotor radius.

Having described the Euler equations and free wake model, the manner in which the two have been coupled will be presented.

4.3 Euler/Wake Coupling Procedure

Although the solution procedure for a hovering helicopter rotor is broken into two parts—namely, the Euler solution for the rotor blade near field and the free wake solution for the semi-infinite wake—it is clear that these two parts are not independent. The coupling of the Euler solver with the free wake iteration procedure is the process by which information from each solver is passed to the other. Basically, the free wake solver requires the strength of the vortex filaments modeling the wake to be determined, and this depends upon the spanwise load distribution of the rotor blade. This information is available from the Euler solver for the blade near field flow. In order to accurately compute the blade loads, the induced velocity field of the wake must be used to prescribe the far field boundary conditions for the Euler solver. In addition, since the wake vortices pass through the Euler computational domain—in particular, the tip vortex from the preceding blade passes close to the blade—the portion of the wake within the computational domain must be introduced into the domain in a manner avoiding the numerical diffusion of the wake vorticity. If this last requirement is not met, the distribution of the blade load distribution cannot be accurately computed.

The general outline of the coupled iteration procedure is as follows:

1. the Euler solver initially run for 200 iterations, without including the wake, in order to get a reasonably converged starting solution;

2. the bound circulation distribution along the span of the rotor blade is determined from the Euler solution of the rotor blade near field;

3. the bound circulation distribution is used to set the strength of the wake vortices, and the free wake algorithm is used to solve for the wake geometry;

4. the new wake geometry and vortex strengths are used to determine the induced velocities for the Euler far field boundary conditions, and the wake is introduced into the Euler finite volume grid using the perturbation scheme described in chapter 3;

5. the Euler solver is run another 100 iterations using the new boundary conditions and wake geometry;

6. the procedure from step 2 to step 5 is repeated.

The details of the implementation are given below.

The strengths of the trailing vortices modeling the wake in the free wake solver are determined by the spanwise bound circulation distribution on the rotor blade. This is obtained from the Euler solution by performing a line integral of the velocity,

$$\Gamma(y) = \oint_C \vec{u}' \cdot d\vec{s}, \tag{4.23}$$

around a chordwise contour at a fixed spanwise station of the rotor (Figure 4.11). Because of the sensitivity of the rotor loads to the wake geometry, the bound circulation can change greatly between each free wake solution, especially in the initial stages of the coupled calculation procedure. For this reason, the computation of the bound circulation is underrelaxed,

$$\Gamma^{new} = \Gamma^{old} + \omega_b \left(\Gamma - \Gamma^{old}\right), \tag{4.24}$$

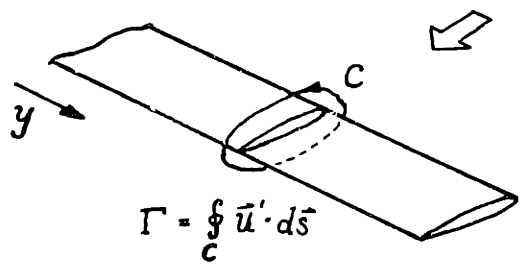

Figure 4.11: Bound circulation determination

where Γ^{old} is the previous bound circulation and Γ is the circulation evaluated using Equation (4.23). The underrelaxation parameter ω_b is usually taken to be 0.5. For the initial iteration, Γ^{old} is set using the combined blade element/momentum theory (Gessow & Myers, [29]). With this bound circulation distribution, the strengths of the trailing vortices are determined using the following roll up schedule:

1. the tip vortex is assumed to roll up from the position of maximum bound circulation Γ_{max} of the blade to the tip, and the strength of the tip vortex is set to be equal to Γ_{max};

2. the distance from the location of Γ_{max} to the blade root is divided into four equally spaced segments, and the change in the bound circulation in each segment gives the strength of the trailing vortex emitted from each segment (Figure 4.12).

The location of the vortices in the tip path plane are fixed by determining the centroid of vorticity within each segment,

$$y_v = \frac{\int_{y_1}^{y_2} \Gamma y dy}{\int_{y_1}^{y_2} \Gamma dy}, \tag{4.25}$$

where y_1 and y_2 are the endpoints of the segment (Figure 4.12). Because

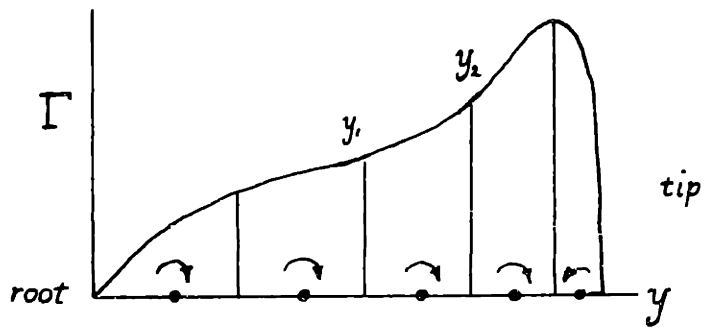

Figure 4.12: Trailing vorticity roll up schedule

the bound circulation can only be found between the spanwise grid nodes on the blade, the bound circulation on the blade must be interpolated between the grid cells along the blade to evaluate the above integral. To do this, the Glauert transformation

$$y = \frac{R}{2}(1 - \cos\theta), \ 0 \leq \theta \leq \pi, \tag{4.26}$$

is first used to get θ at the locations at which Γ is known. With θ known, the distribution $\Gamma(\theta)$ is determined using a cubic spline fit along the blade. Next, Equation (4.25) is rewritten in terms of θ,

$$y_v = \frac{R}{2}\frac{\int_{\theta_1}^{\theta_2} \Gamma\,(1 - \cos\theta)\sin\theta d\theta}{\int_{\theta_1}^{\theta_2} \Gamma \sin\theta d\theta}. \tag{4.27}$$

The integrals in Equation (4.27) are evaluated using a trapezoidal rule integration over each segment y_1 to y_2. Thus the location of the vortex filaments in the plane of the rotor are found, and with the vortex strengths known, the wake geometry solution procedure is started.

With a new wake geometry, the specification of the far field boundary conditions and calculation of the prescribed flow residuals for the Euler

solver can be performed in a fashion similar to that described in chapter 3. This is done using the same model of vortex rings and cylinders given by the converged free wake geometry. The calculation of the far field induced velocities, $\vec{u}'_{wake}$, Equation (4.7a), are found by summing the induced velocities of all the wake vortex rings and cylinders using the formulas given in the previous section. In the same way, the induced velocity at each cell in the computational domain is determined. Since the portion of the trailing vortex wake attached to the rotor blade is to be computed as part of the Euler solution procedure, the vortex rings fixed in the rotor tip path plane are excluded from these velocity calculations. That is, only the wake elements lying below the rotor are used to determine the wake induced velocities in the Euler computational domain.

In computing the induced velocity at each finite volume cell, the problem of avoiding the singularity near the vortex location is encountered, and some core structure must be assumed. Unlike the free wake solver, in which all that was needed was the self-induced velocity of the vortex, the induced velocity field through the vortex core is required. To obtain this, it is assumed that near the vortex itself, the velocity field of the vortex looks like that of a two dimensional vortex plus a uniform velocity corresponding to its self-induced velocity. The two dimensional core structure was taken to be a Lamb vortex core, and the self-induced velocity component was found using Equation (4.19). (Strictly speaking, the velocity given by Equation (4.19) is for a Rankine core structure. Bliss [9] has developed a theory for the self-induced velocity of a vortex with an arbitrary core structure, but the difference here is not significant.) The core radius a was taken to be $0.15c$, as described in the previous section, for all the cases presented here. Since the assumption of a two-dimensional velocity field is valid only near the core, it is necessary to transition from the Lamb velocity equation, Equation (3.13b), to the Equations (4.14a), and (4.14b) for a vortex ring at some

point in the field. For distances $> 4a$ from the center of the vortex core, the velocity field was taken from the formulas for the induced velocity of a vortex ring; at distance$< 2a$ from the center of the vortex, the two dimensional Lamb vortex velocity field plus the self-induced velocity was used. Linear interpolation of the vortex ring and the two dimensional formulas was used in the range $2a$ to $4a$ from the vortex center.

With the induced velocities known at each cell, the prescribed flow state vector $\mathbf{U}_0$ is determined using the same procedure as in chapter 3. The condition of constant total rothalpy replaces the constant total enthalpy condition. To compute the entropy distribution in the vortex cores, the assumption of an isolated two dimensional Lamb vortex structure is used, and Crocco's equation is integrated as in the fixed wing case. The equation solved is Equation (3.17b), given in the last chapter, for the entropy distribution through a Lamb vortex. This equation is also used to obtain the entropy at the far field boundary, which is needed for specifing the entropy at an inflow boundary. With the entropy, total rothalpy, and velocity known, the entire state vector $\mathbf{U}_0$ can be determined. The prescribed flow residuals $\mathbf{R}_0$ are then found, and the Euler equations are integrated in time in the same manner as in chapter 3.

It should be noted here that prescribed flow representing the wake is not an exact solution of the steady Euler equations, since it consists of a collection of stationary vortex rings. As shown in the previous chapter, the finite volume equations are no longer consistent, since the prescribed flow residuals will not vanish as the grid spacing vanishes. This problem can only be avoided if a wake model which exactly satisfies the Euler equations is used.

4.4 Results

The flow fields of two rotors have been computed. The first is two bladed rotor tested by Ballard, Orloff, & Luebs [5] at a low tip Mach number and Reynolds number. In the experiment, the bound circulation distribution along the blade was measured using a laser Doppler velocimeter. Also measured was the location of the tip vortex at its first pass below the rotor blade. The second series of computations compares to the experimental results of Caradonna & Tung [14]. This experiment measured the chordwise pressure distributions at five spanwise stations on the blade, and the tip vortex geometry was measured using a hot wire anemometer. Computations for two tip Mach numbers have been made.

4.4.1 Ballard et al. Test Case

The first rotor geometry computed here was tested by Ballard, Orloff, & Luebs [5]. It is a two bladed rotor with a rectangular planform, an aspect ratio of 13.7, NACA 0012 airfoil section, and a linear twist of 11 degrees from root to tip. The collective pitch at 75% span is 9.8 degrees. The experiment was run at a tip Mach number of 0.225 and a chord Reynolds number of 400,000 at the tip.

The computations were performed on a finite volume grid consisting of 96 cells around the blade chord, 20 cells from the blade to the outer boundary, and 40 cells along the span. The distance from the rotor blade to the outer boundary was six blade chords. The spanwise gridpoint distribution was chosen to increase the resolution in the region of the blade vortex interaction. This can be seen in a plot of the surface grid over the last 40% of the rotor blade radius, shown in Figure 4.13. A spanwise section of the grid is shown in Figure 4.14.

For all the calculations of this rotor, the *CFL* number was 2.8, the second and fourth difference artificial viscosity coefficients were 0 and 0.01, respec-

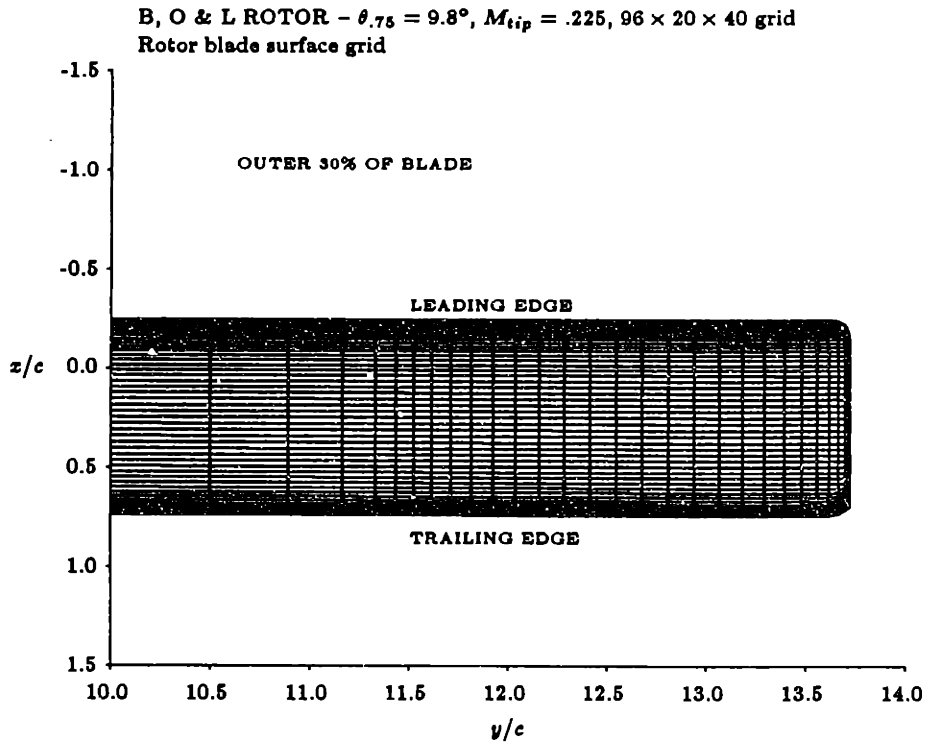

Figure 4.13: Surface grid, outer 40% of Ballard et al. rotor

tively, and the rothalpy damping coefficient was 0.025. The wake geometry underrelaxation parameter ω_w was set to 0.2, and the bound circulation relaxation parameter ω_b was set to 0.5.

The first three computions for this rotor were performed at the geometric collective pitch setting of the experiment, $\theta_{.75} = 9.8$. The three calculations were: (a) an Euler solution for the isolated rotor without the wake coupling; (b) a solution in which the free wake and Euler solvers were coupled, but in which the wake influence was included in the Euler code only through the far field boundary conditions; and (c) a fully coupled Euler/free wake solution using the procedure described in the last section. For case (c), the tip vortex

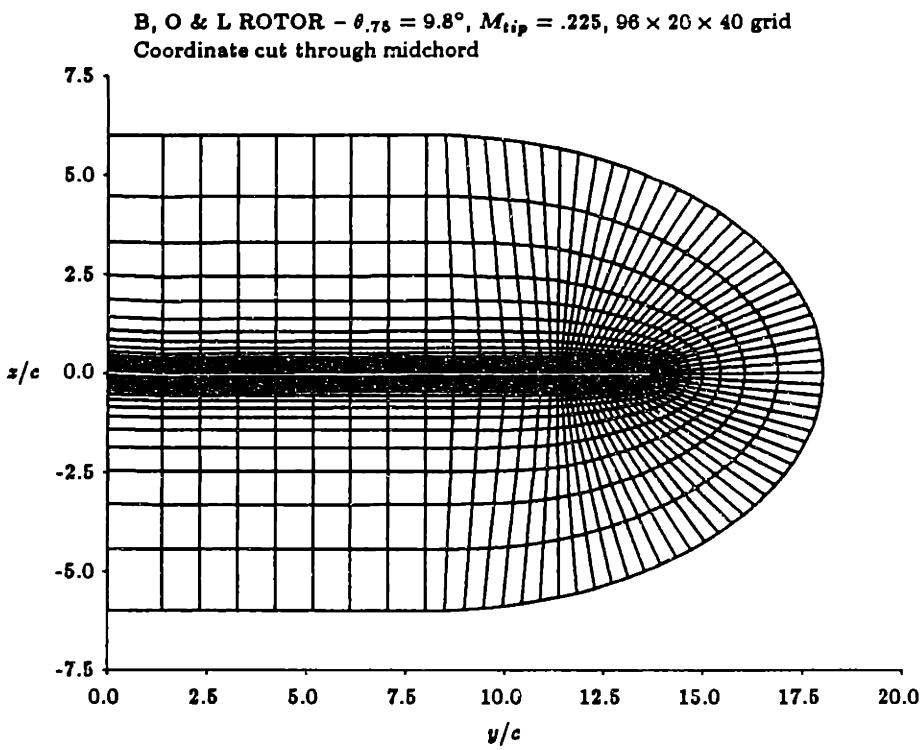

Figure 4.14: Spanwise view of grid through rotor midchord, Ballard et al. rotor

from the preceding blade lies approximately 1 blade chord below the rotor. This is sufficiently far away that the distortion of the vortex path due to the blade is relatively small. Because of this the prescribed flow residuals are specified throughout the domain.

Figure 4.15 shows the bound circulation computed for the three cases, along with the experimentally measured bound circulation. The bound circulation has been non-dimensionalized by the tip speed times the rotor radius. The drastic differences in the blade loading among the three solutions is due entirely to the differences in wake modeling. It is seen that the failure

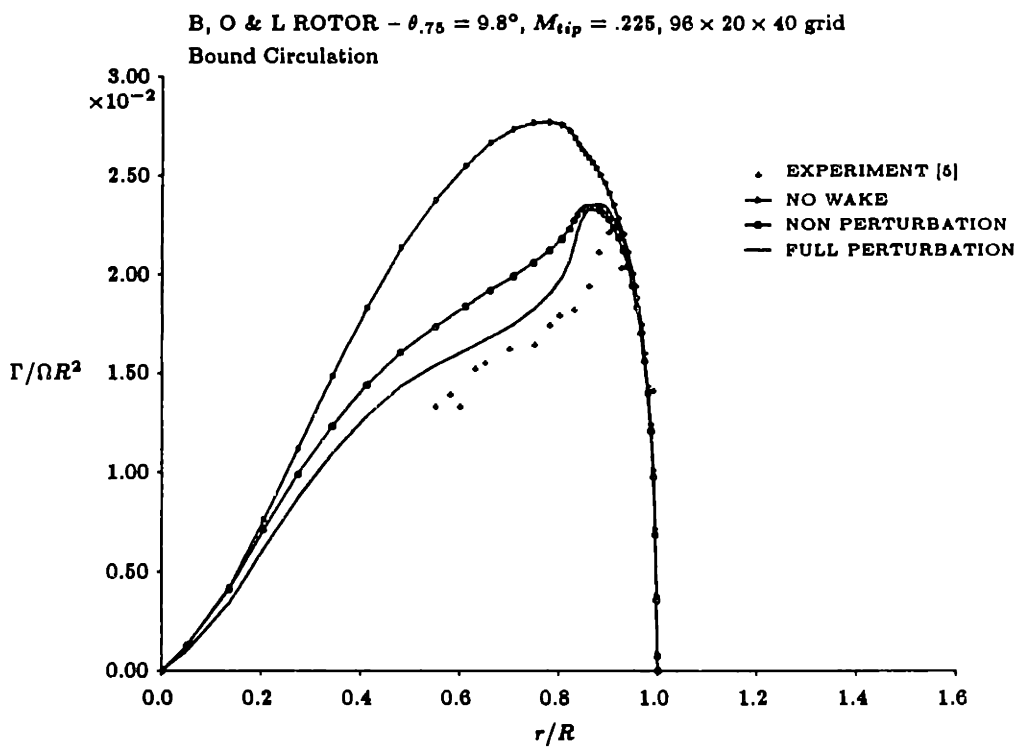

Figure 4.15: Spanwise bound circulation distribution, $\frac{\Gamma}{\Omega R^2}$, Ballard et al. rotor

to include the wake in the Euler solver results in a very large overprediction of the thrust, due to the fact that the downwash at the blade is too low, and hence the angle of attack along the blade is too high. Including the wake through the far field boundary conditions results in some improvement; the maximum bound circulation on the blade is better predicted, and the lift is significantly lower along the blade. However, too much thrust is being produced, and the load distribution does not show the same qualitative behavior as in the experiment. Case (c), the coupled Euler/free wake solution using the perturbation scheme, is significantly better. The distribution of lift

is qualitatively very similar to the experimental data, although the overall thrust is clearly too high. The peak bound circulation is also slightly inboard of the experimental position, corresponding to more wake contraction. It is interesting to note that the peak Γ for cases (b) and (c) are nearly the same, even though the loadings inboard of the peak are quite different. In case (b), more lift is produced inboard due to the fact that the tip vortex, which is introduced into the Euler computational domain only through the boundary conditions, is too diffuse by the time it reaches the rotor. Its contribution to the downwash over the blade is thus not accurately computed. This can be clearly seen in Figures 4.16 and 4.17, in which the velocity vectors in a spanwise section through the midchord of the rotor blade are shown for the two cases. Note that the tip vortex is well defined for case (c), whereas in case (b) the vortex is no longer apparent. What is seen in Figure 4.16 is a more uniform downwash due to the wake, but the rapid variation in the flow velocities and the vortex core structure has vanished entirely. Interestingly, the flow field near the tip is very similar in both cases.

Figures 4.18 and 4.19 show the computed wake geometry for case (b) and case (c), respectively, as well as the measured tip vortex position at the first pass below the blade. Both computations shows a more rapid radial contraction of the wake than was observed experimentally, consistent with the more inboard location of the bound circulation peak on the blade. Also, it appears that the wake is descending at a more rapid rate than in the experiment, although the magnitude of this rapid descent is more difficult to estimate since only one measured tip vortex position is available. However, this is consistent with the greater than observed thrust for these two cases. Interestingly, the wake geometries predicted for the two cases are quite similar despite the differences in the load distributions. This is due to the fact that the tip vortex strengths are nearly the same, and this concentrated vortex plays the dominant role in the evolution of the wake.

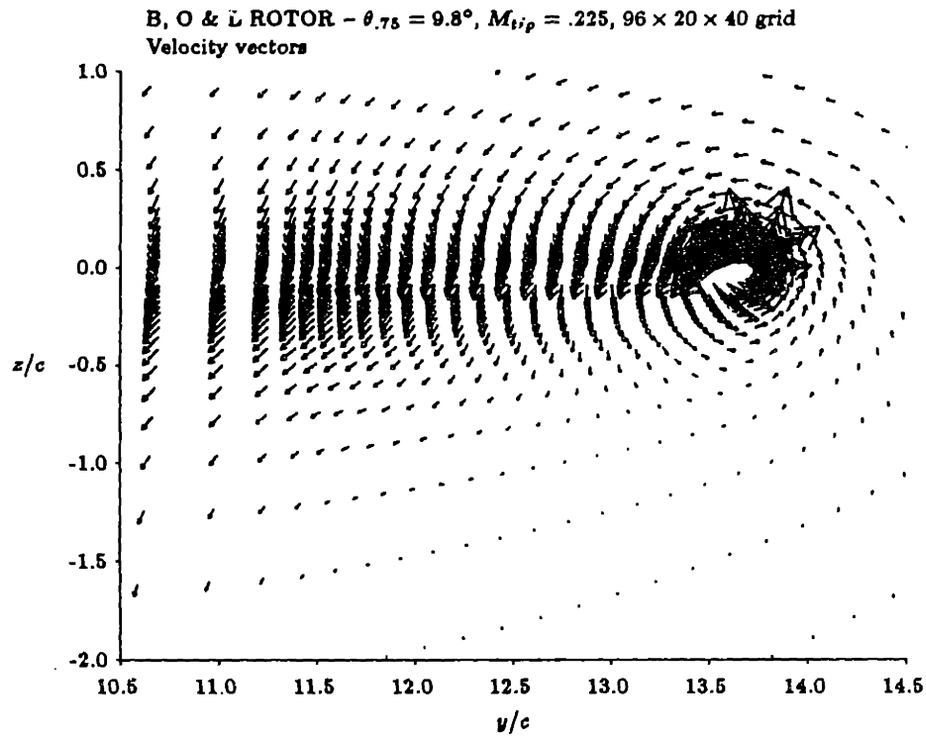

Figure 4.16: Velocity vectors in cross flow plane through rotor midchord, non-perturbation approach

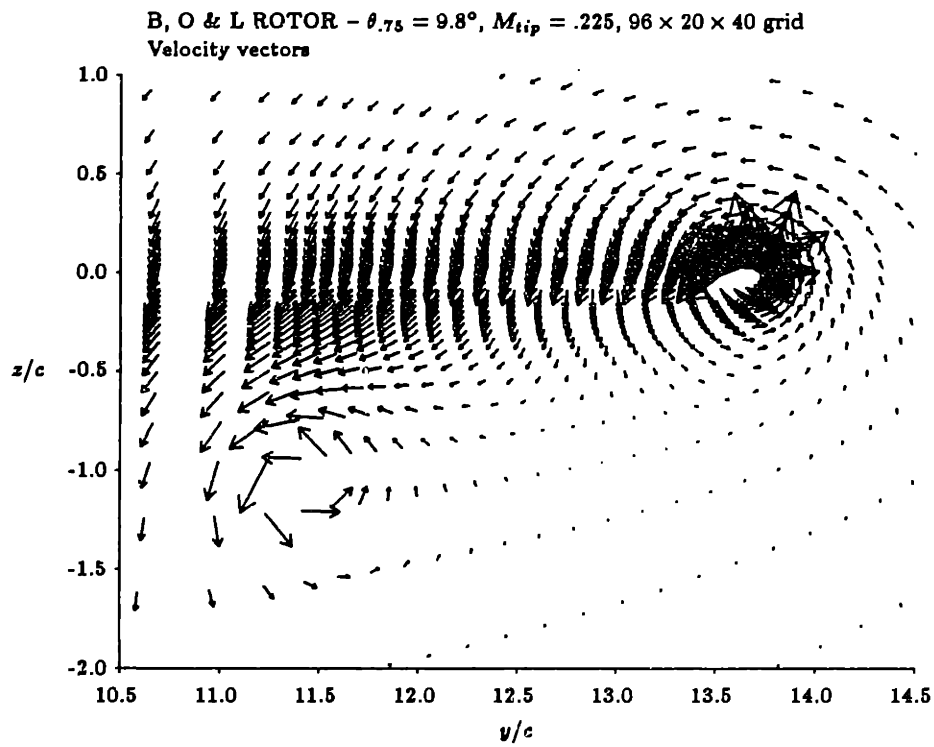

Figure 4.17: Velocity vectors in cross flow plane through rotor midchord, perturbation approach

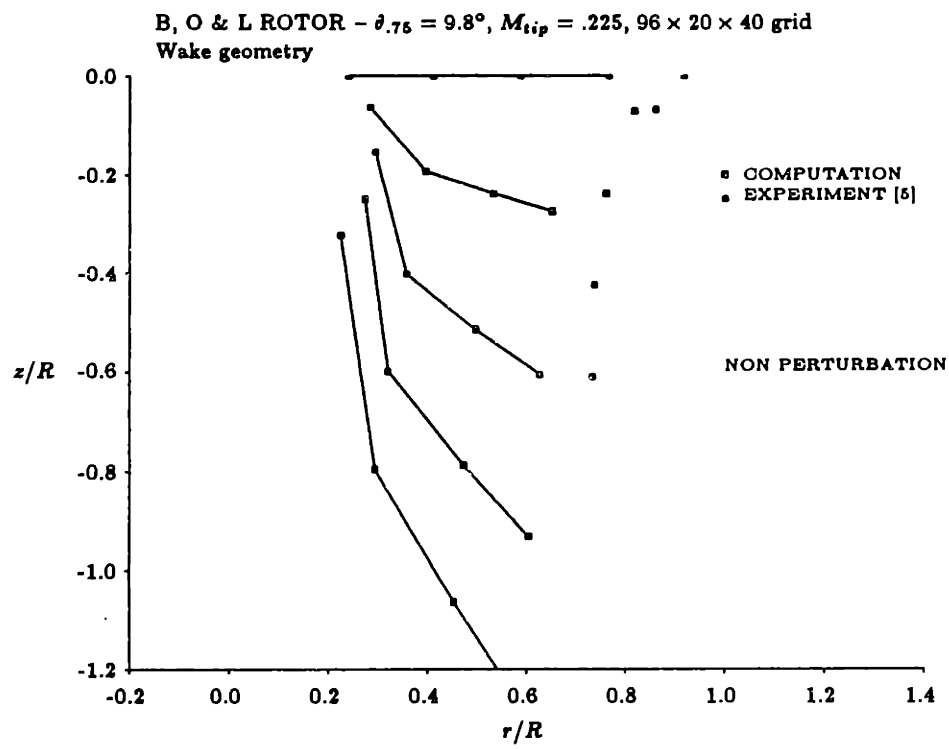

Figure 4.18: Wake geometry, non-perturbation solution, Ballard et al. rotor

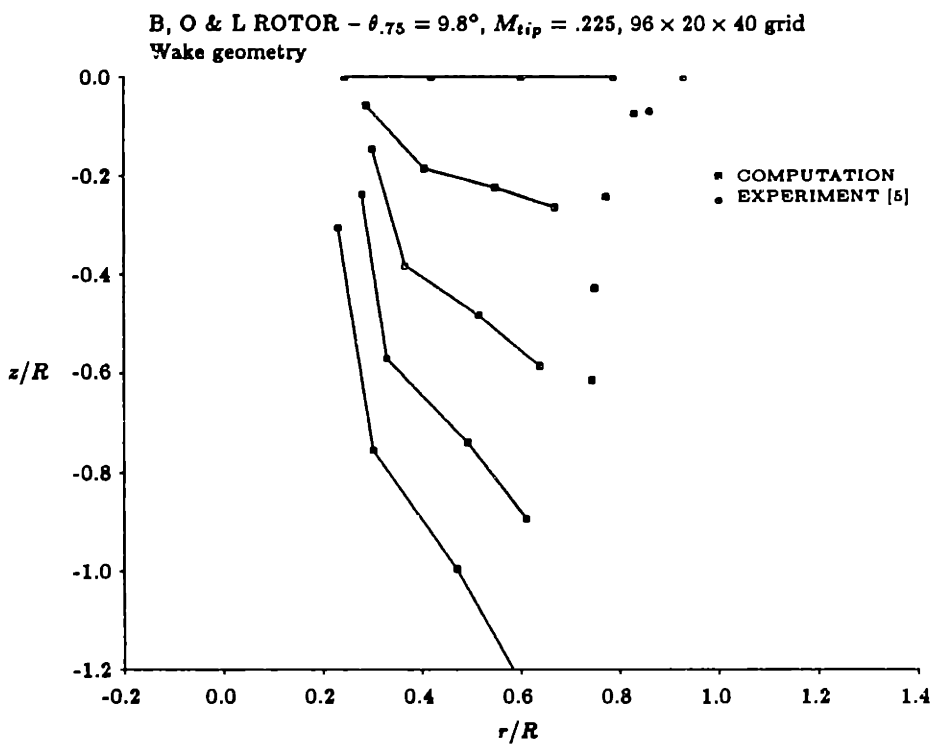

Figure 4.19: Wake geometry, perturbation solution, Ballard et al. rotor

In Figure 4.15, the load distribution near the tip is quite similar for all three cases, despite the differences in the wake influence. Although no surface pressure measurements were made in the experiment, the computed pressures were compared. In Figures 4.20, 4.21, 4.22, and 4.23, the pressure

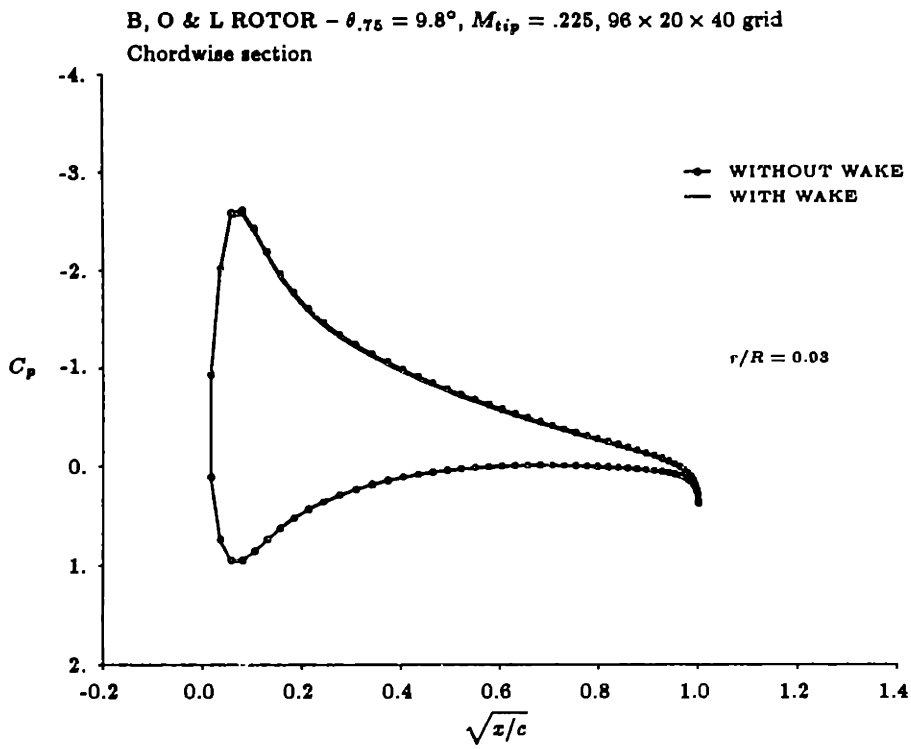

Figure 4.20: Comparison of chordwise loading, with and without the wake, $93\%R$

coefficients for case (a) and case (c) are shown at four chordwise sections within a chord length from the tip. Note the similarity in the pressure distributions, which is remarkable for the fact that one solution has no wake included and the other is the fully coupled Euler/free wake solution. This observation was also noted in the computations for the other rotor presented

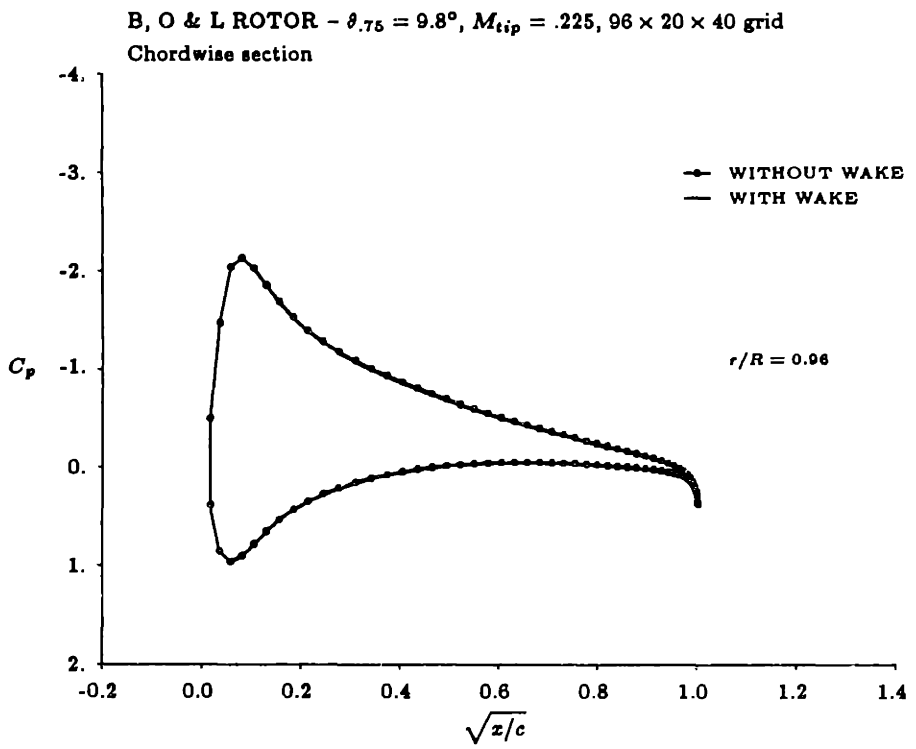

Figure 4.21: Comparison of chordwise loading, with and without the wake, 96%R

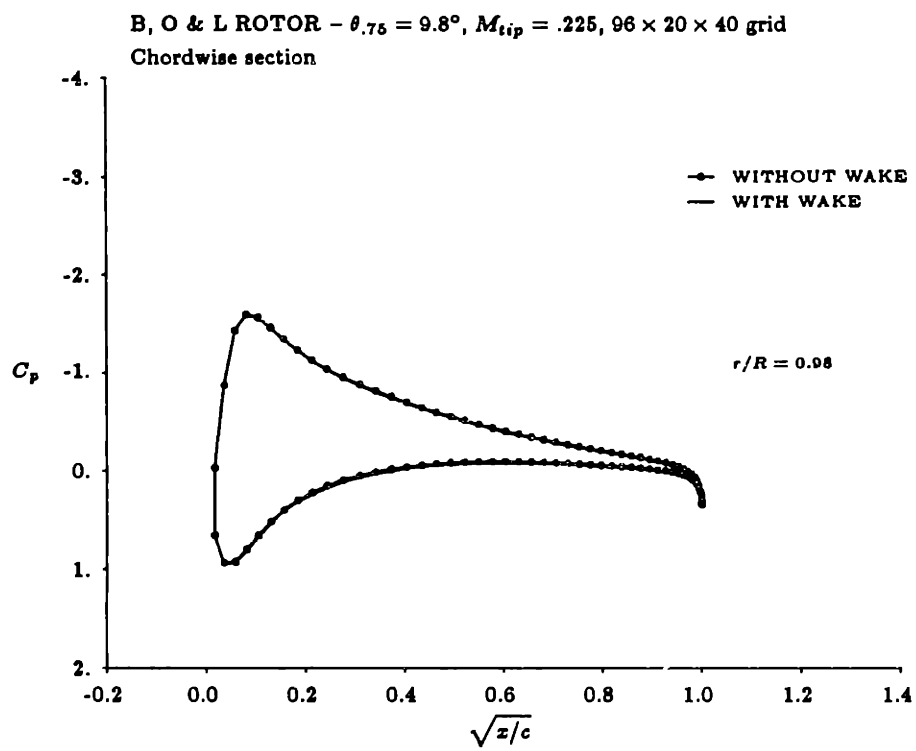

Figure 4.22: Comparison of chordwise loading, with and without the wake, 98%R

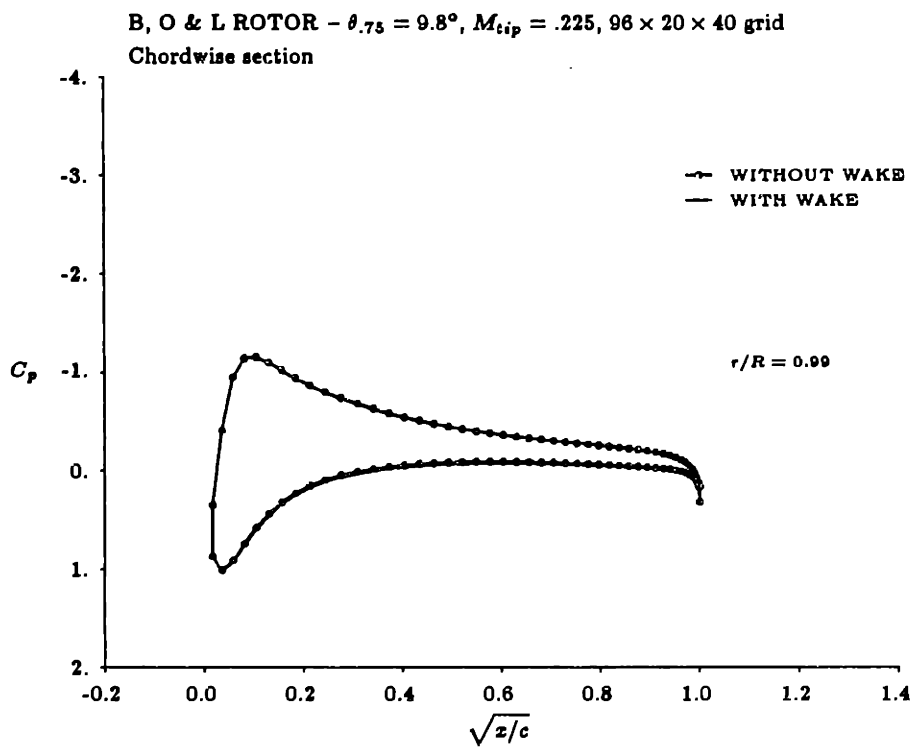

Figure 4.23: Comparison of chordwise loading, with and without the wake, 99%R

in this thesis, the rotor tested by Caradonna & Tung. Those results will be shown below, and an explanation for the results will be presented then.

Although the fully coupled Euler/free wake solution of case (c) is in significantly better agreement with the experimental data than the other two cases, the thrust is still being overpredicted. This can be attributed to viscous effects. Due to the low Reynolds number of the experiment (400,000 at the tip, based on tip chord), there is a significant difference the lift curve slope of the blade compared to the inviscid lift curve slope. The magnitude of lift loss due to viscosity was estimated using the two dimensional Euler code ISES, developed by Michael Giles and Mark Drela [30,22]. This code solves the steady, two dimensional, compressible Euler equations for airfoils and cascades using a direct Newton iteration procedure. It also incorporates an efficient and robust coupling of the inviscid solver to an integral boundary layer procedure. The boundary layer equations have been developed to handle Reynolds numbers as low as 250,000, including transitional separation bubbles. The ability of the code to handle such flows has been demonstrated by Drela in his thesis [22], as well as in a more recent paper [23].

To estimate the viscous effect, ISES was run twice. First, an inviscid calculation was made at a fixed lift coefficient of 0.7 and a Mach number of 0.2. This approximately corresponds to the peak computed lift coefficient on the rotor at about 90% radius. The inviscid angle of attack for these conditions was found to be 5.73 degrees. Holding the angle of attack fixed at this value, a viscous calculation was run at a Reynolds number of 350,000 for the same Mach number. The computed lift coefficient was 0.63, or 10% lower than for the inviscid case. This suggests that the angle of attack should be reduced by 10%, or 0.57°, in the inviscid case to approximately correct for the influence of the boundary layer on the lift. This correction was simply made for the rotor by reducing the collective pitch of the blade

by 0.6°.

In Figure 4.24, the collective pitch has been reduced by 0.6 degrees. The

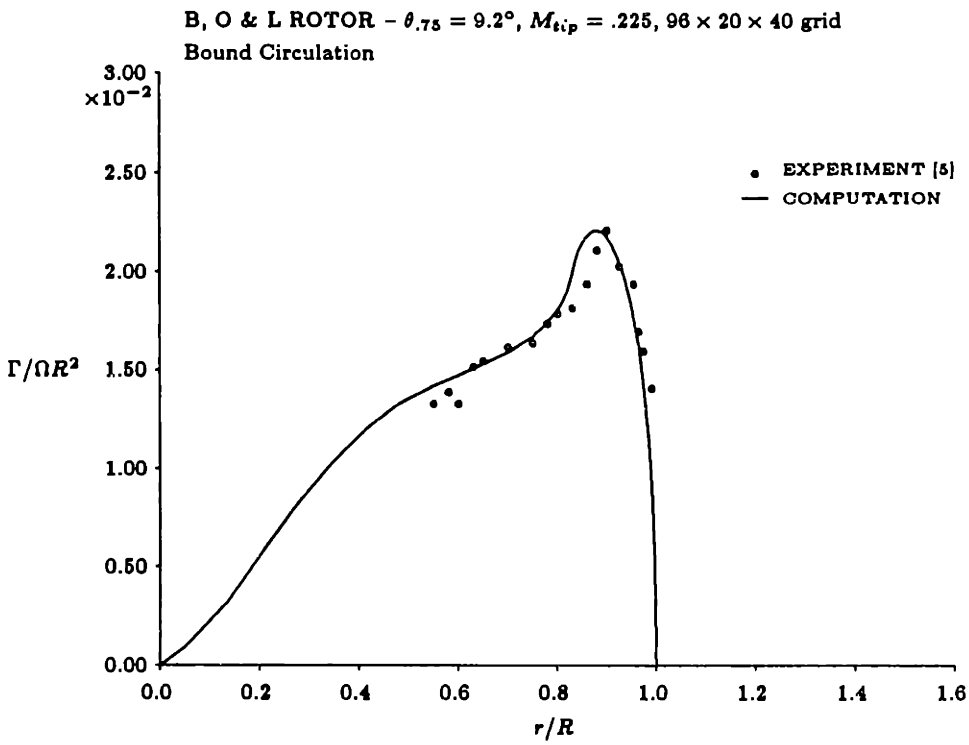

Figure 4.24: Spanwise bound circulation distribution, $\frac{\Gamma}{\Omega R^2}$, Ballard et al. rotor, reduced collective

agreement with experiment is very good. The position of the peak circulation is still slightly inboard of the experimentally observed peak, again indicating that the free wake computation is predicting a more rapid contraction of the wake than was observed experimentally. This is confirmed in Figure 4.25, which shows the wake geometry for this case. Note that the wake contraction is nearly identical with the previous case, but that the axial location of the first tip vortex is identical to its measured value.

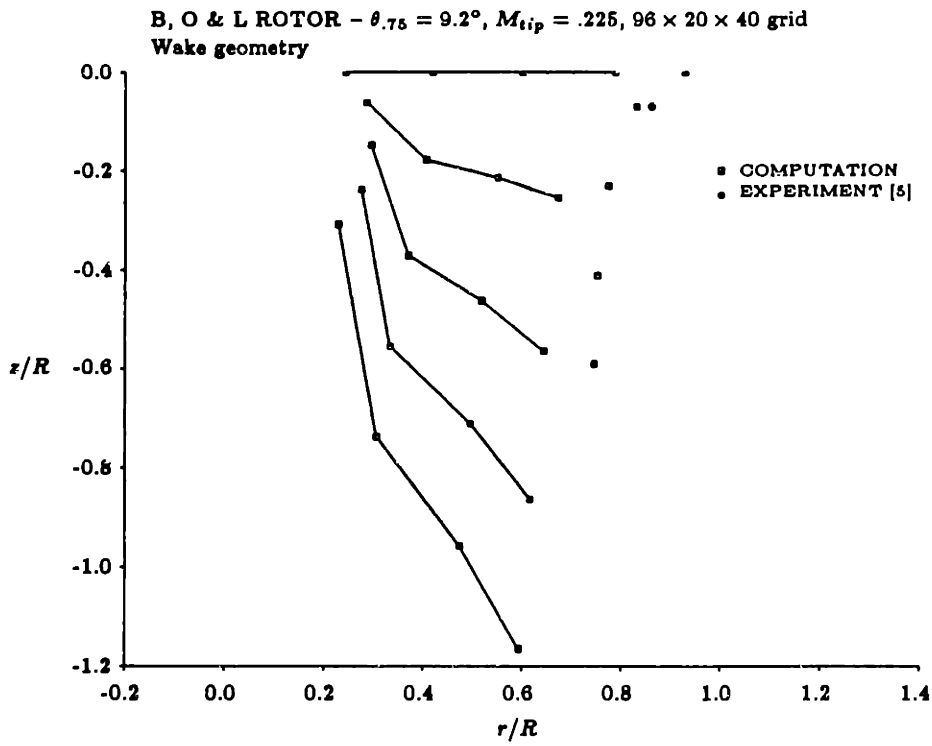

Figure 4.25: Wake geometry, Ballard et al. rotor, reduced collective

Again, since the calculated thrust now matches the measured thrust, this is to be expected. The differences in the measured and experimental wake contraction may be attributed to couple of factors. First, the computed wake contraction is primarily dependent on the far wake model. It is the radial component of the induced velocity due to the entire semi-infinite wake that determines how rapidly the wake contracts. On the experimental side, static thrust test are very sensitive to the test conditions. These experiments were run in an enclosed chamber, meaning that flow recirculation will have an effect on the flow field. The magnitude of these effects are difficult to estimate.

The solutions of the coupled Euler/free wake code required about 43 minutes of CPU time on the Cray X-MP/48 at NASA Ames Research Center for these cases. About 25% of the time was spent in the computation of the prescribed state vector U_0, which was done after each free wake solution. This portion of the code ran in scalar mode, while the Euler solver itself is well vectorized. The time spent in computing the prescribed flow could be reduced by about half if the induced velocities of the vortex cylinders representing the far wake are not included in the specification of the prescribed flow. Since the induced velocities of the far wake are smooth through the finite volume grid, it is probably sufficient to include them only through the far field boundary conditions. The work involved in the free wake iteration procedure itself is negligible, requiring only 1% of the total solution time.

4.4.2 Caradonna & Tung Test Case

The second set of solution computed were for a rotor geometry tested by Caradonna & Tung [14]. The rotor has two blades of aspect ratio 6, is untwisted and untapered, with a NACA 0012 airfoil section. The rotor was tested over a range of collective pitch settings and tip Mach numbers, and the chordwise surface pressures were measured at five spanwise stations along the blade. The tip vortex geometry was measured over the range of 0 to 450° azimuth, using a hot wire anemometer. Computations were performed for two experimental configurations, $M_{tip} = 0.439$, $\theta_{.75} = 8°$ and $M_{tip} = 0.877$, $\theta_{.75} = 8°$. The corresponding Reynolds numbers at the tip for the two cases are 1.56×10^6 and 3.12×10^6, respectively.

The grid for this rotor was similar to that for the previous rotor, with 96 chordwise cells, 20 cells from the blade surface to the far field, but only 32 cells along the span. The reduction in the spanwise resolution was made due to the lower aspect ratio of the blade; the resolution in the region of the blade/vortex interation is similar to that of the previous rotor, as can

be seen in the surface grid plot, Figure 4.26.

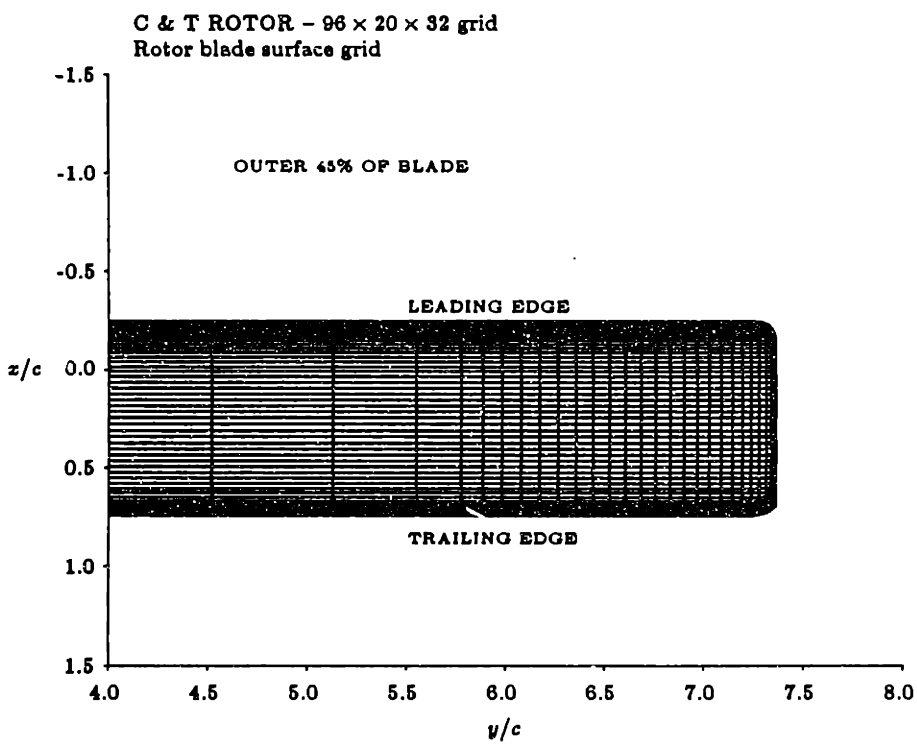

Figure 4.26: Surface grid, Caradonna & Tung rotor

The first case run was for the lower tip Mach number. A *CFL* number of 2.8, no second difference dissipation, a fourth difference dissipation coefficient of 0.01, and a rothalpy damping coefficient of 0.025 were used. In the free wake solver, ω_w was set to 0.1, and ω_b was set to 0.25.

Figure 4.27 shows the computed lift coefficient distribution compared to experiment; the bound circulation was not measured in the experiment, which is why the lift coefficients rather than the bound circulation are compared here. The experimental lift coefficients were computed in reference [14] by a chordwise integration of the measured surface pressure coefficients.

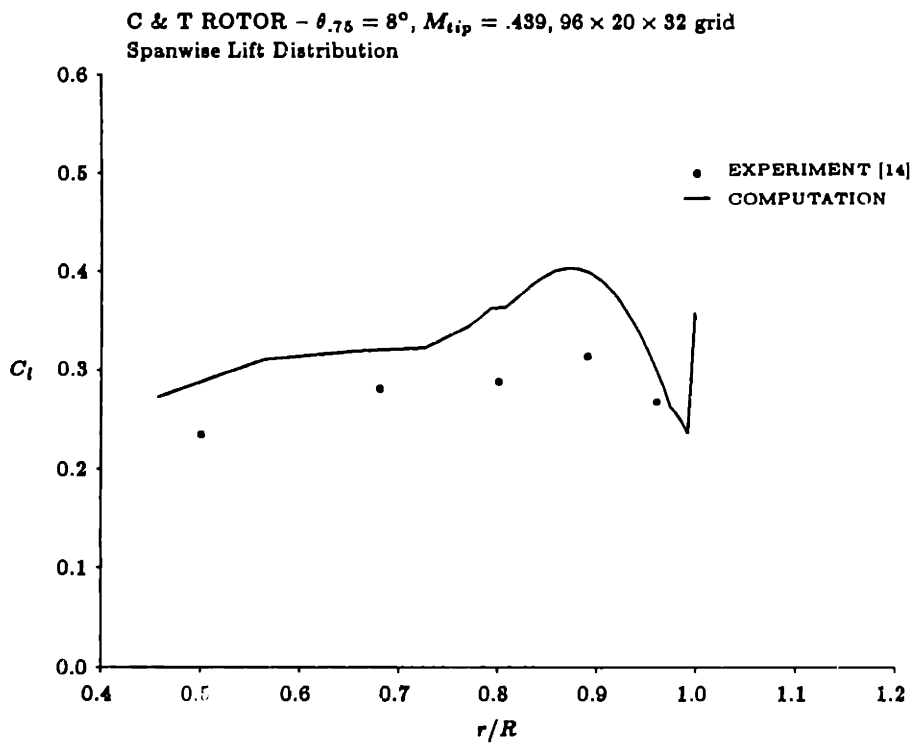

Figure 4.27: Spanwise lift coefficient distribution, Caradonna & Tung rotor, $M_{tip} = 0.439$

The predicted thrust is higher than observed. The peculiar spike in the computed C_l near the tip is due to inaccuracies in the pressure extrapolation to the surface, which is caused by the grid distortion near the leading and trailing edges at the tip. (Recall from chapter 2 that near the coordinate singularity, the difference equations are locally zeroeth order accurate.) The wake geometry is shown in Figure 4.28; it is seen that the free wake

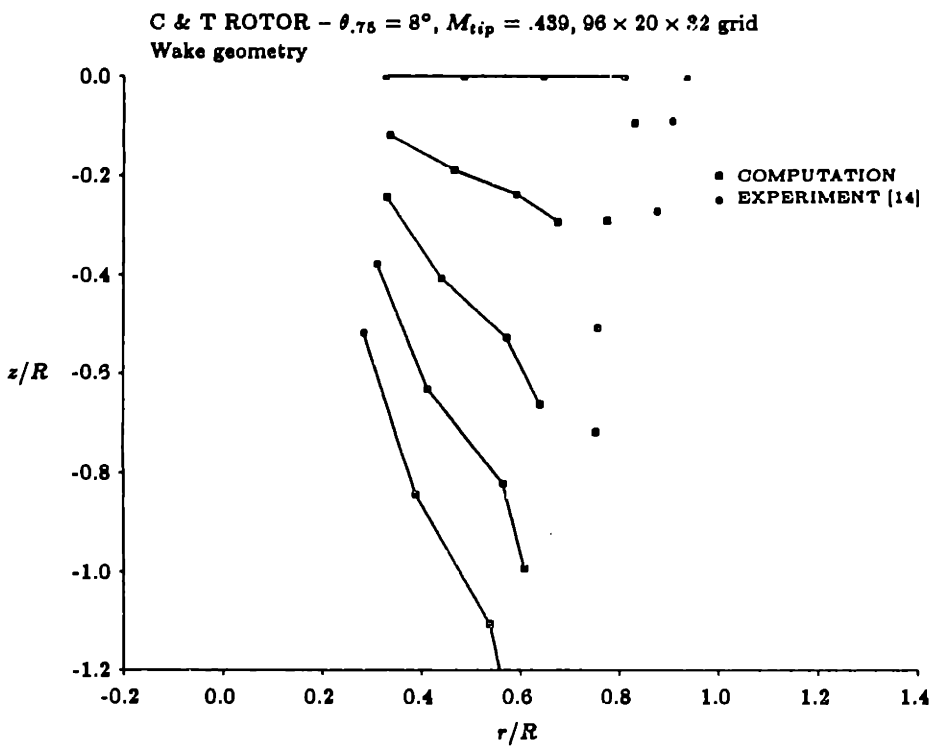

Figure 4.28: Wake geometry, Caradonna & Tung rotor, $M_{tip} = 0.439$

code predicts a much greater wake contraction than measured in the experiment, and the axial descent rate of the tip vortex is overpredicted. This second effect is consistent with the overprediction of the thrust. Figure 4.29 compares the computed and experimental surface pressure coefficients. The

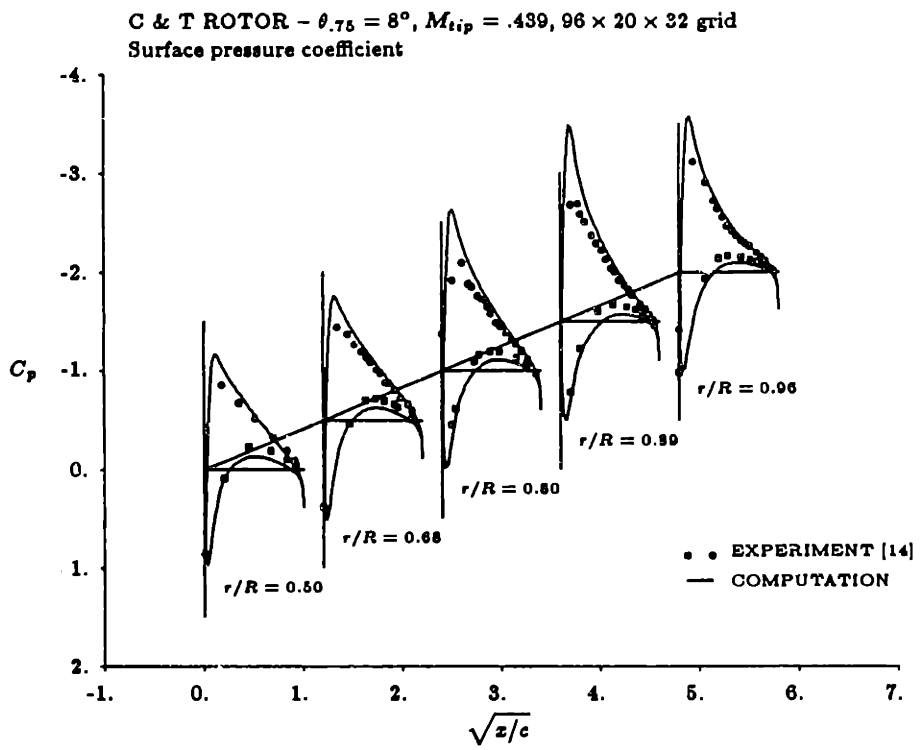

Figure 4.29: Surface pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.439$

agreement is fair, with the greater than observed lift once again apparent.

The second case run was for the tip Mach number of 0.877. This case is obviously highly transonic, and as such is not representative of a real hovering rotor. It does however provide a severe test of the scheme. A *CFL* number of 2, second and fourth difference dissipation coefficients of 0.35 and 0.01, respectively, and a rothalpy damping coefficient of 0.025 were used. The wake and bound circulation relaxation parameters were the same as the previous case.

Figure 4.30 compares the computed and experimental lift coefficients.

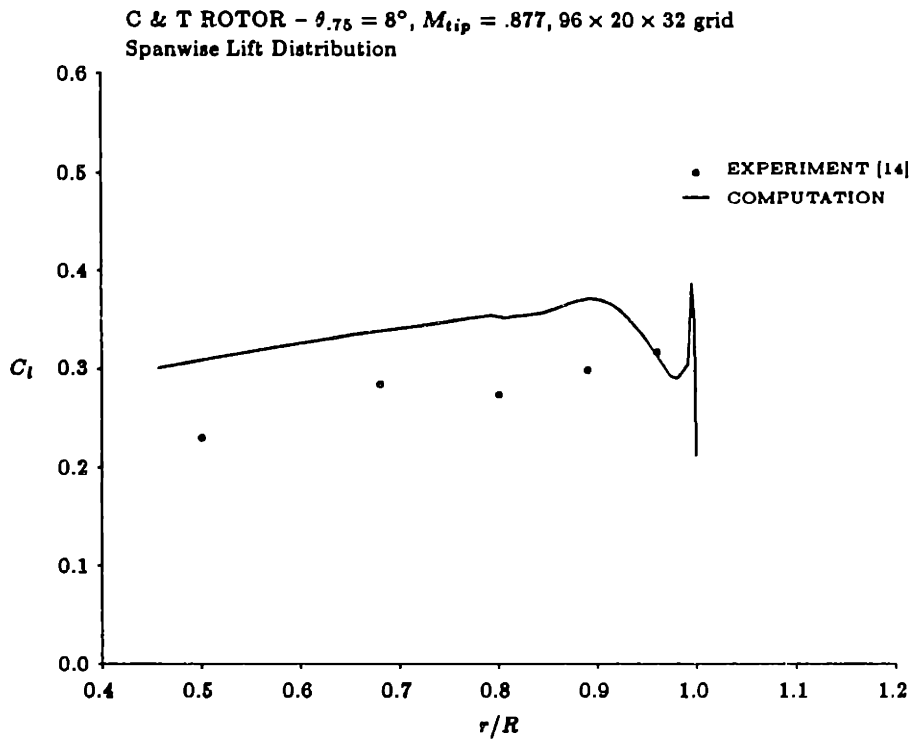

Figure 4.30: Spanwise lift coefficient distribution, Caradonna & Tung rotor, $M_{tip} = 0.877$

As for the lower tip Mach number case, the predicted thrust is higher than observed. The wake geometry is shown in Figure 4.31; the predicted wake

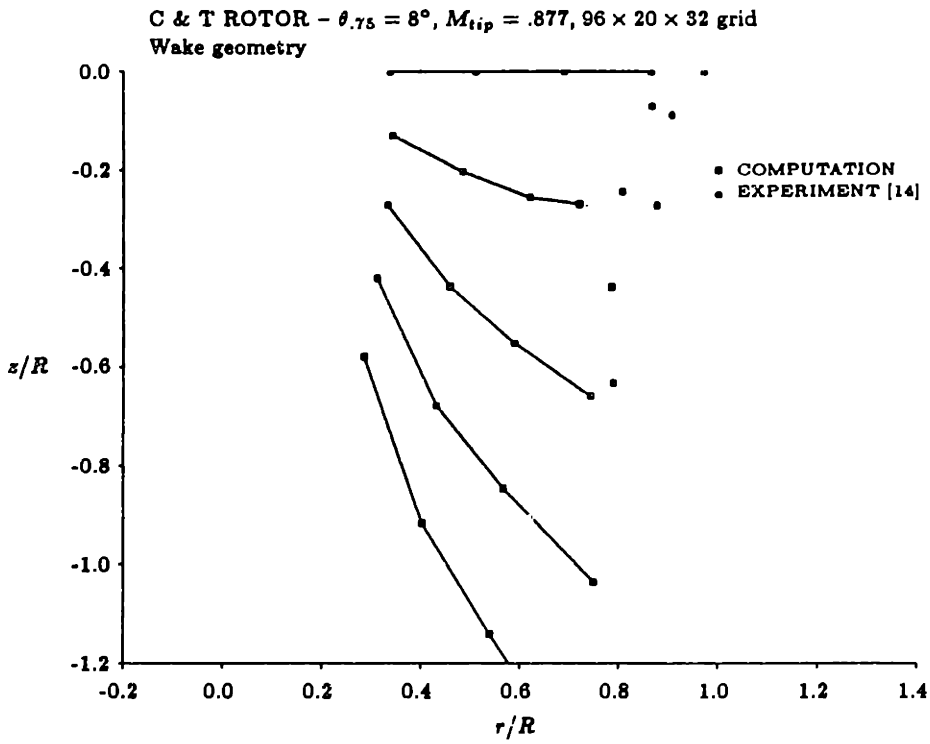

Figure 4.31: Wake geometry, Caradonna & Tung rotor, $M_{tip} = 0.877$

geometry shows less contraction and a slower descent rate than the $M_{tip} = 0.439$ solution. This is in contrast to the experimental results, which showed that the wake geometry was not sensitive to the tip speed. Figure 4.32 compares the computed and experimental surface pressure coefficients. The agreement is fair over the inboard sections, with larger discrepancies near the tip. The shock is stronger and further aft than in the experiment. This is consistent with the neglect of viscous effects in the computation; at the 96% span station, the local Mach number reaches approximately 1.5 before

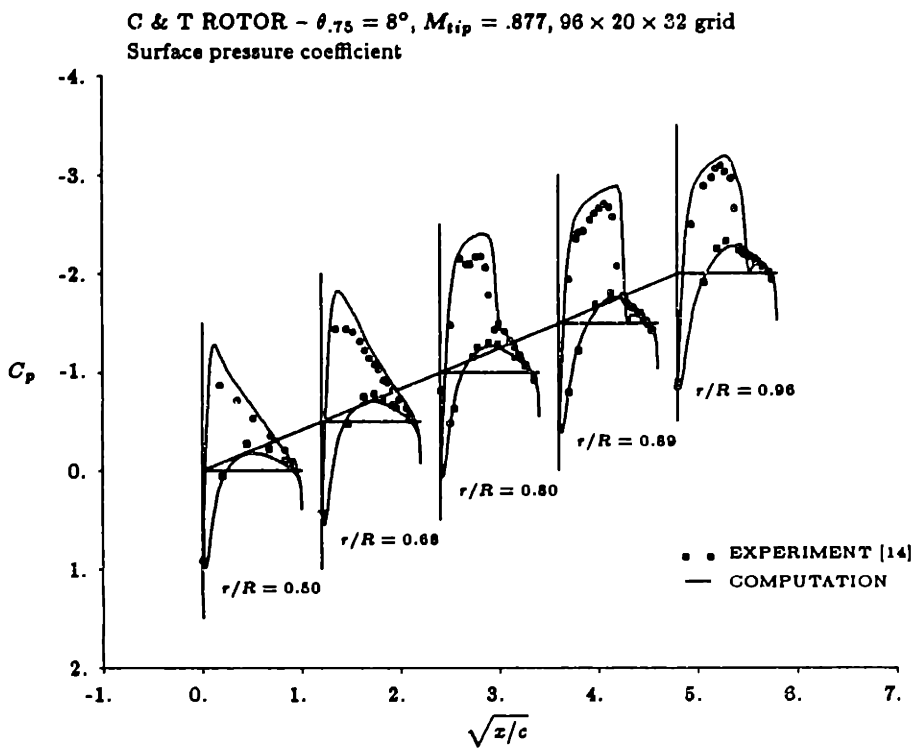

Figure 4.32: Surface pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.877$

the shock. The shock/boundary layer interaction should be strong in this case.

Given the insensitivity of the blade loading near the tip for the Ballard et al. rotor, a similar comparison was made for the Caradonna & Tung rotor, to see if the same effect would be found on a blade of different twist and aspect ratio. Also, comparisons at higher tip Mach numbers could be made with this data. Solutions were obtained for the same collective pitch setting and tip Mach numbers, but the wake was neglected. Comparisons of the pressure coefficients at each of the experimental span stations were made.

Figures 4.33, 4.34, 4.35, 4.36, and 4.37 show the computed pressure coefficients for the $M_{tip} = 0.439$ case with and without the wake. The experimental values are also shown for comparison. Significantly higher lift is obtained over the inboard sections of the blade when the wake is not included in the computation, due to the failure to get the correct downwash along the blade. The agreement of the two solutions gets better further toward the tip. The last two stations shown, 89% and 96% span, show much better agreement; these stations are with about 0.8 chords of the tip. At the 96% station, the differences are quite small.

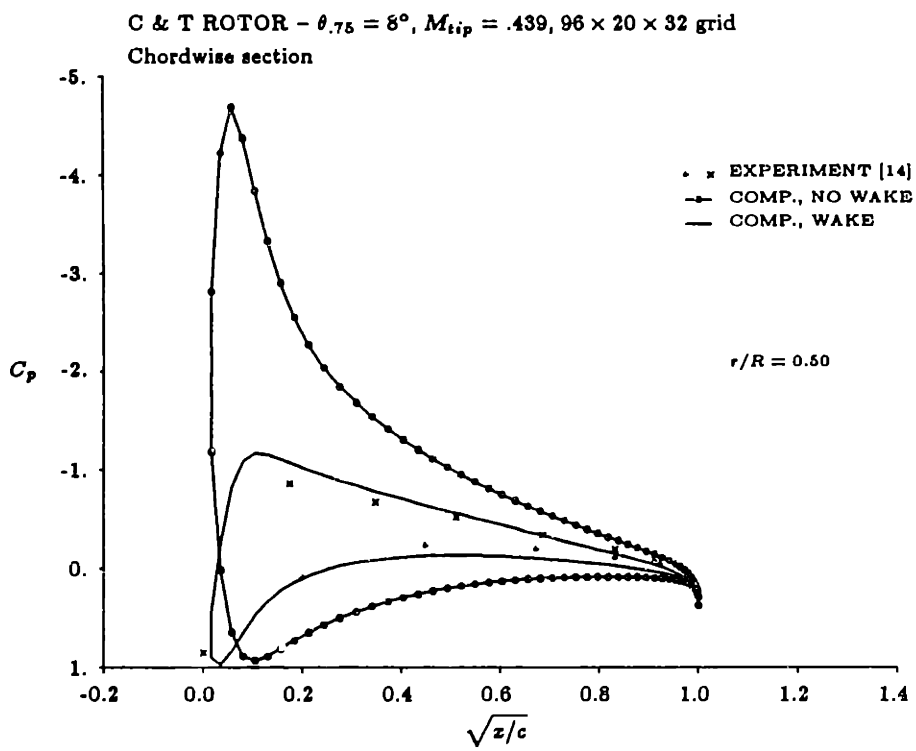

Figure 4.33: Chordwise pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.439$, $r/R = 0.5$, with and without wake

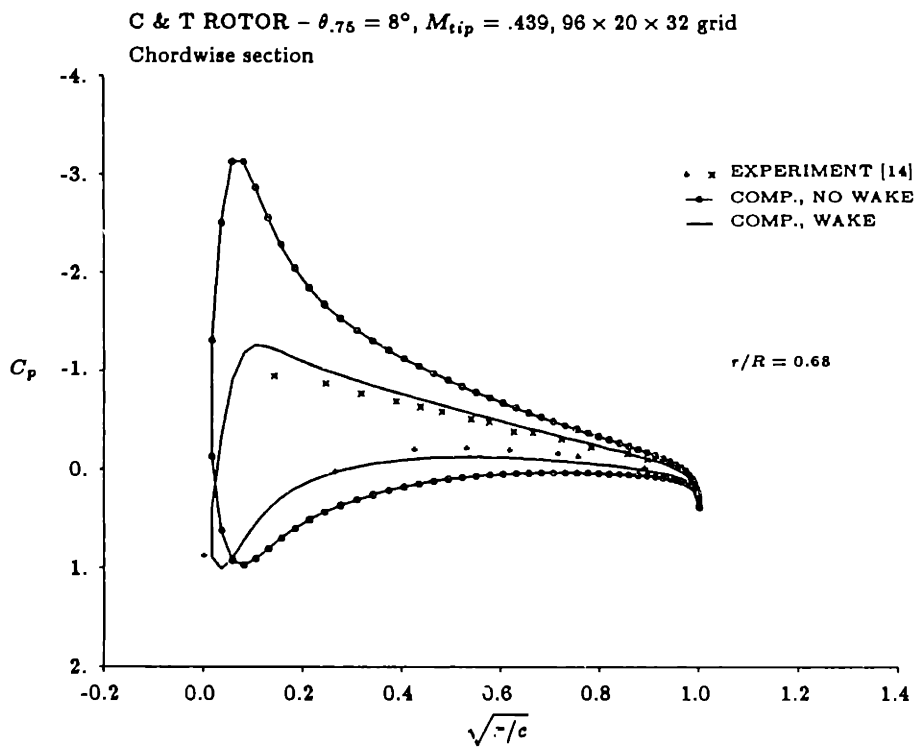

Figure 4.34: Chordwise pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.439$, $r/R = 0.68$, with and without wake

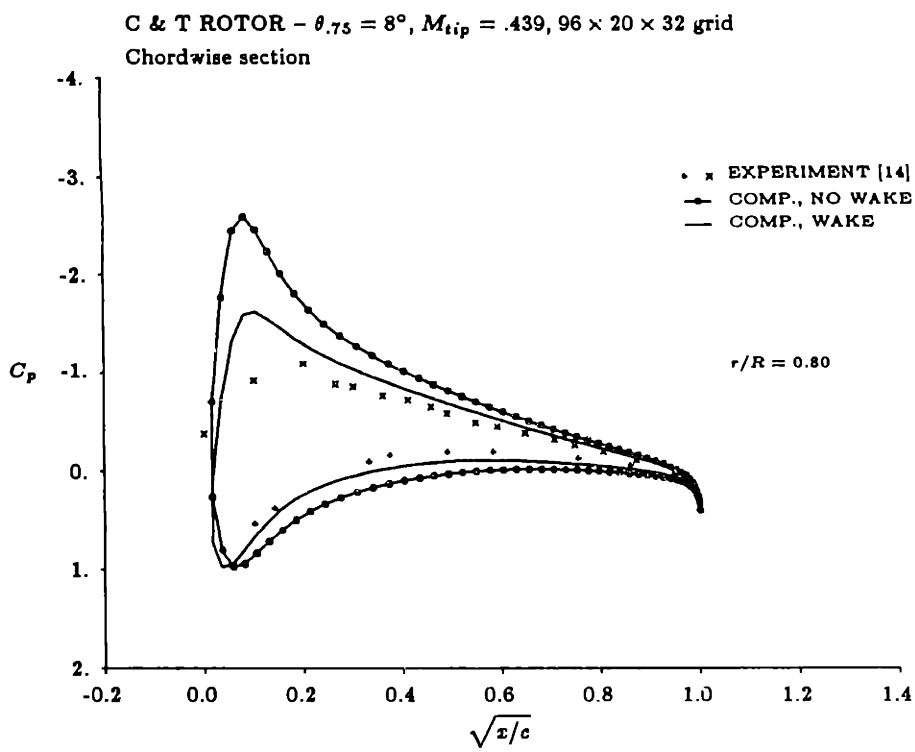

Figure 4.35: Chordwise pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.439$, $r/R = 0.8$, with and without wake

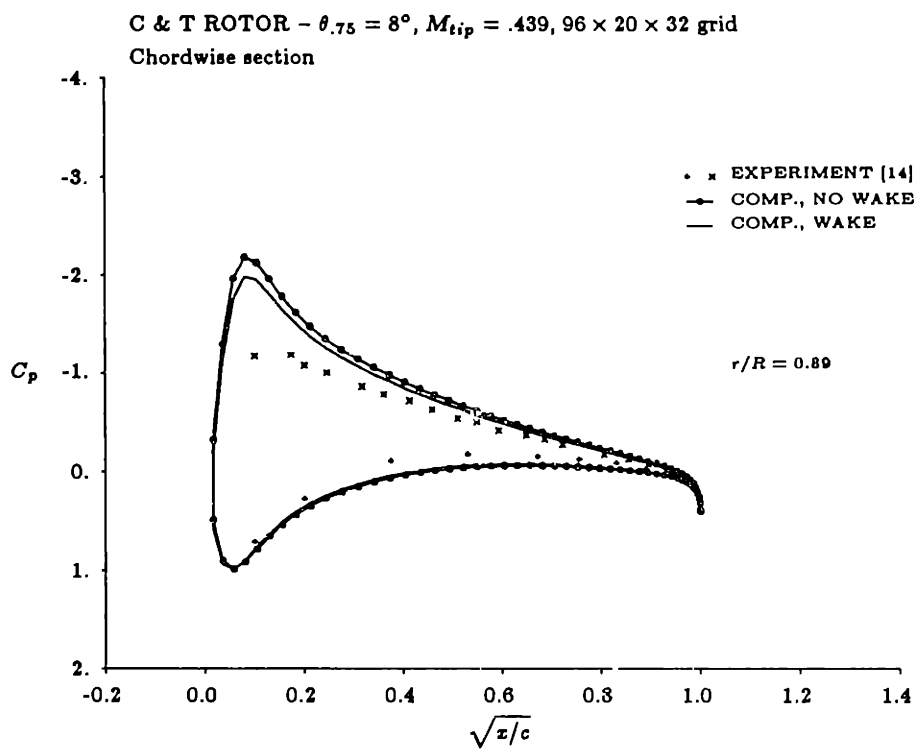

Figure 4.36: Chordwise pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.439$, $r/R = 0.89$, with and without wake

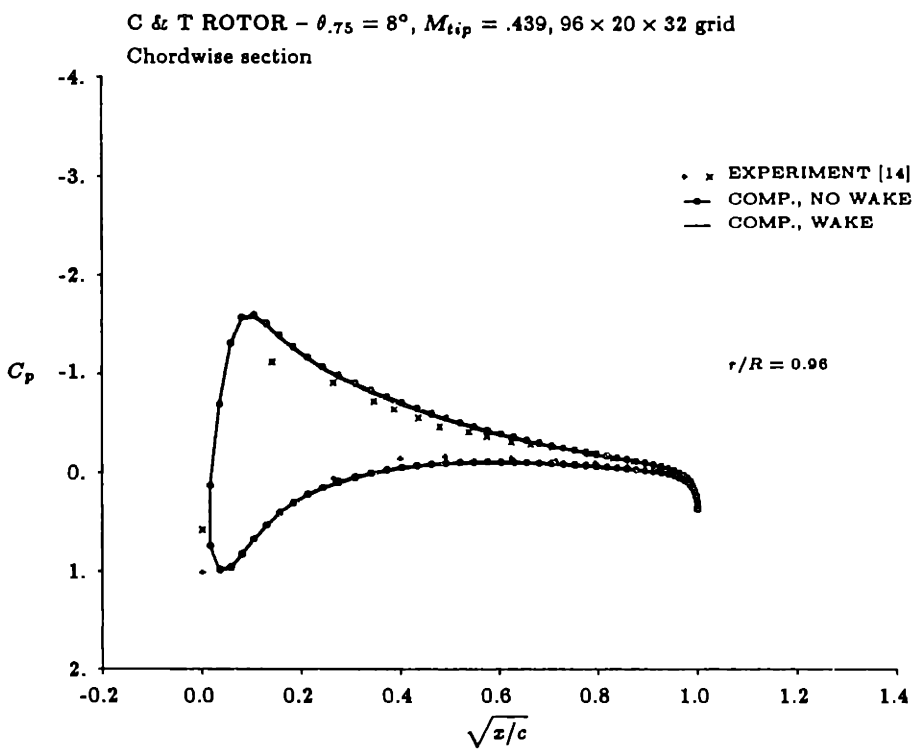

Figure 4.37: Chordwise pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.439$, $r/R = 0.96$, with and without wake

Figures 4.38, 4.39, 4.40, 4.41, and 4.42 show the computed pressure

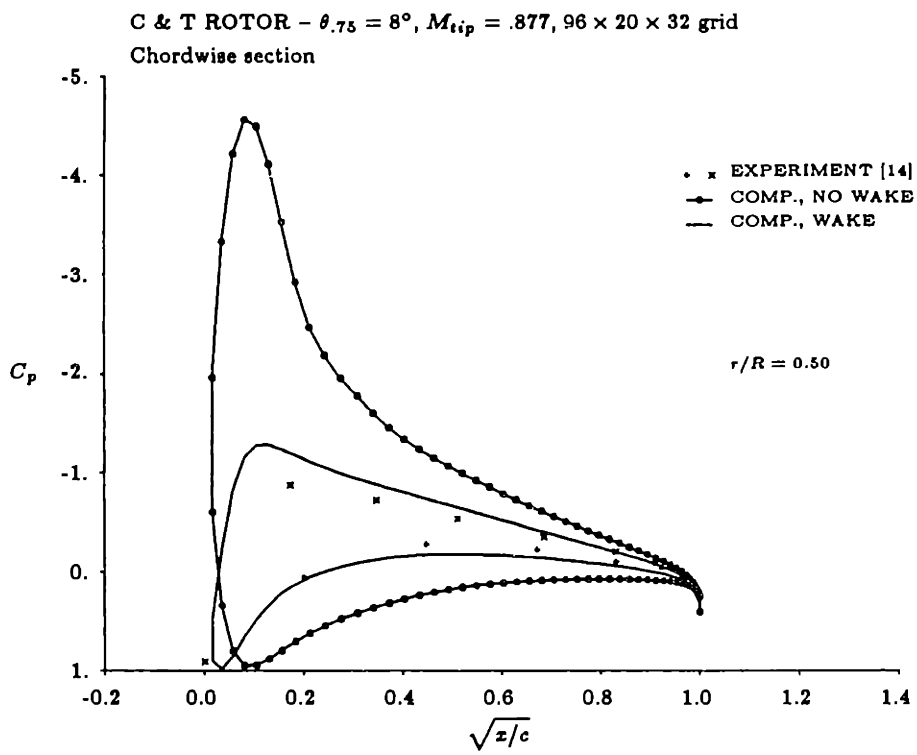

Figure 4.38: Chordwise pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.877$, $r/R = 0.5$, with and without wake

coefficients for the $M_{tip} = 0.877$ case with and without the wake. Similar behavior to the lower tip Mach number case is observed, with significant differences inboard, becoming less near the tip. Greater discrepancies are noted near the tip than were seen for the previous case or for the Ballard et al. solution, but the differences are still quite small. Given the highly transonic flow near the tip, the relative insensitivity of the solution in this region to the inclusion, or lack of inclusion, of the wake is remarkable.

The loading near the tip (within $\approx$ 0.1 chord of the tip) is seen be

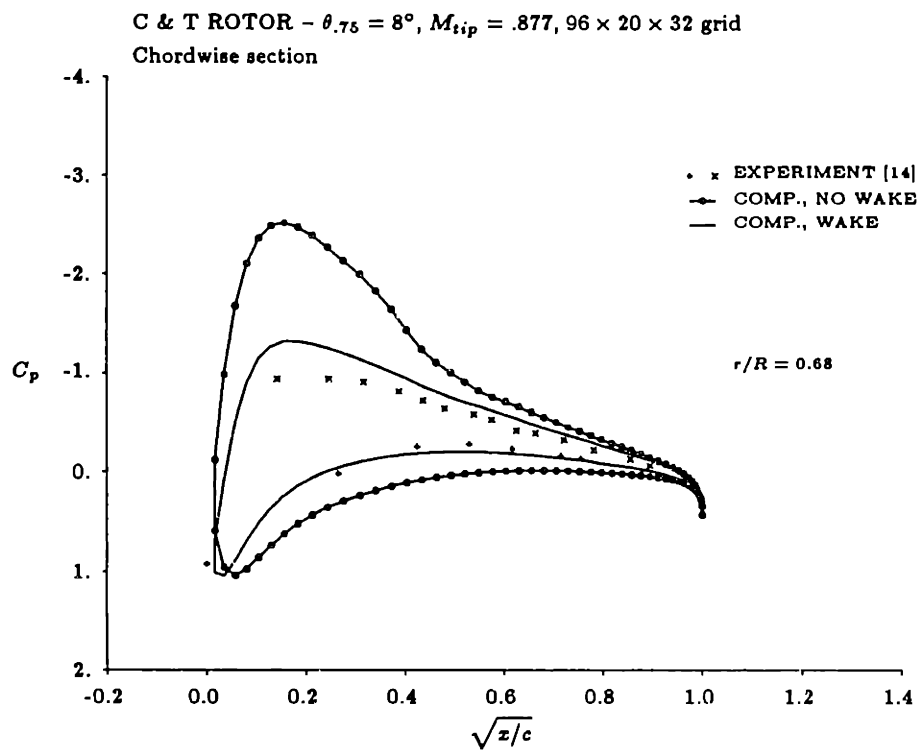

Figure 4.39: Chordwise pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.877$, $r/R = 0.68$, with and without wake

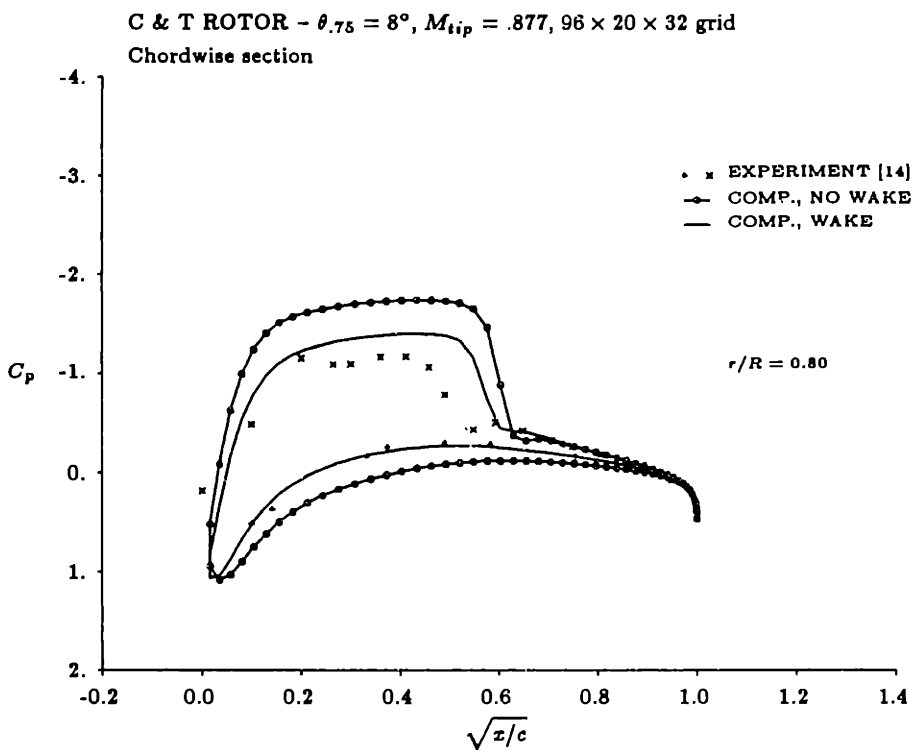

Figure 4.40: Chordwise pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.877$, $r/R = 0.8$, with and without wake

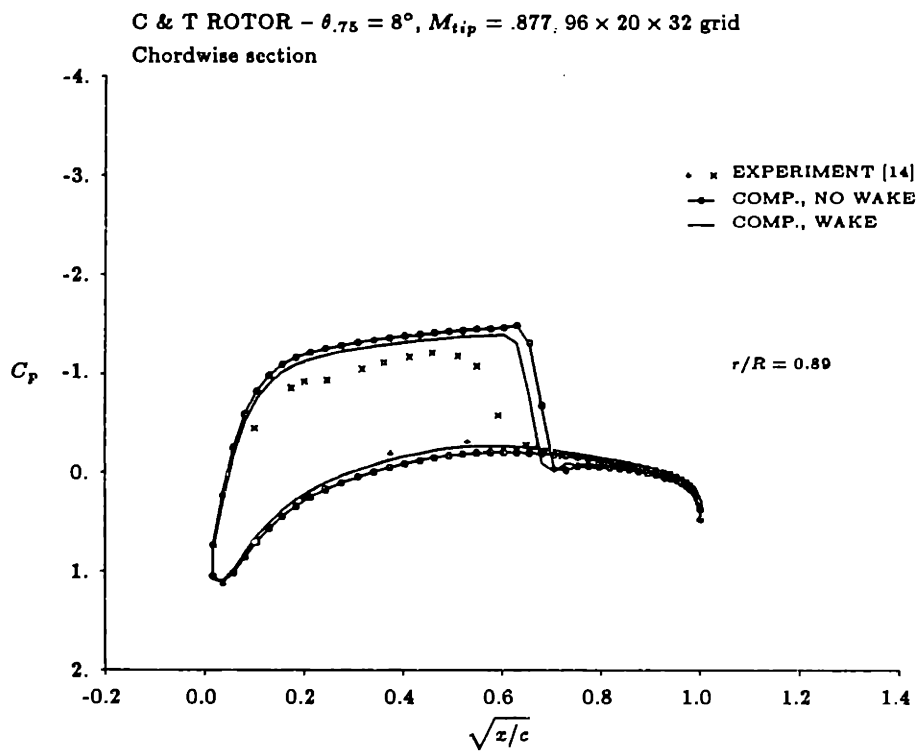

Figure 4.41: Chordwise pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.877$, $r/R = 0.89$, with and without wake

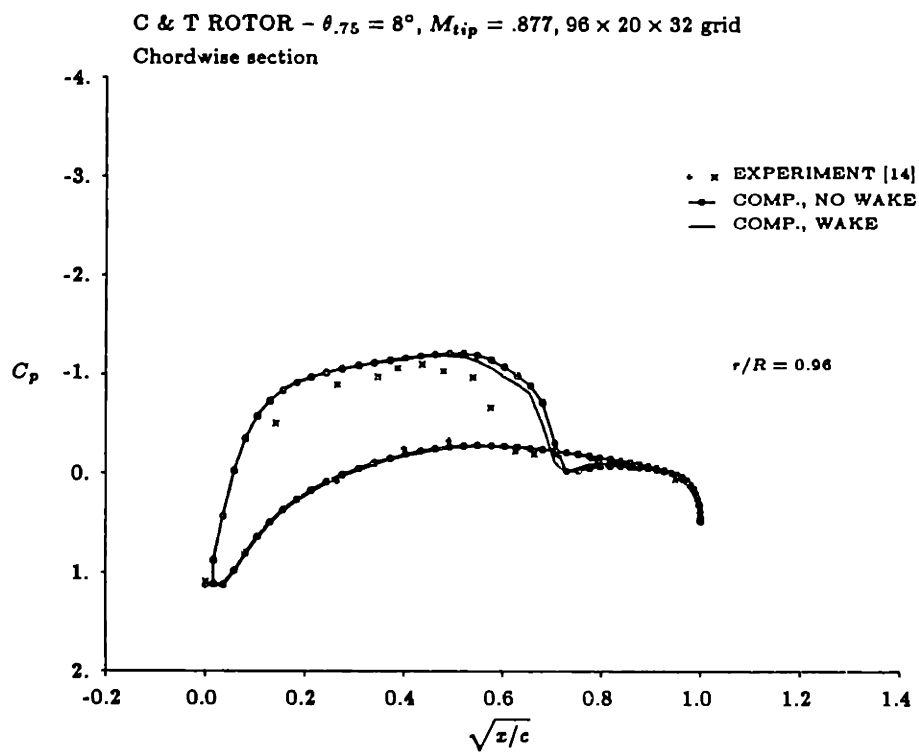

Figure 4.42: Chordwise pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.877$, $r/R = 0.96$, with and without wake

dominated by the geometric angle of attack with a relatively weak influence of the wake induced velocities over a wide range of tip Mach numbers for these two rotors. This may be explained by thinking of the vortex wake as a semi-infinite vortex cylinder. The induced velocity field of a cylinder has a uniform downwash in the center of the cylinder, and is zero outside the cylinder. The vortex wake behaves in a qualitatively similar manner, as can be seen in Figure 4.17 of the velocity field below the Ballard et al. rotor at $\theta_{.75} = 9.8°$. The tip vortex can be thought of as the top of the vortex cylinder. Note that the velocity field below the rotor shows a strong downwash inboard of the tip vortex, but the induced velocities are nearly zero outboard. With this behavior, it is seen that the tip loading will be dominated by geometric factors and the influence of the portion of the trailing vortex sheet attached to the blade rather than the wake geometry and induced velocities.

Another way of thinking of this is to note that although the induced velocity of the tip vortex is upward outboard of the vortex, the vortex is descending at the same time. The downward motion of the vortex cancels the upwash near the tip, yielding nearly zero induced velocity in that region. Again, this implies that geometric factors will dominate the flow field near the tip.

The fact that the loading near the tip is insensitive to the wake has a couple of implications. First, the comparison of calculated and experimental loadings near the tip will not provide much useful information on the accuracy of a wake model. The loading over the inboard sections of the rotor will be far more sensitive to the wake than near the tip. Second, the relative insensitivity of the blade loading in the tip region suggests that the initial stages of the vortex formation and roll up will be dominated by the tip geometry and angle of attack. For experimental or numerical studies of the tip vortex formation process, accurate wake modeling may not be

necessary, but accurate geometric modeling is required.

For the two coupled Euler/free wake solutions presented for this rotor, the overall agreement with experiment is not nearly as good as for the previous rotor. Part of the discrepancy in the computed wake geometries is due to problems with the experimental data. In Caradonna & Tung's report, it was noted that the prescribed wake model of Kocurek & Tangler [35] did not correlate with the experimentally observed tip vortex geometry, the Kocurek & Tangler wake correlations yielding more rapid contraction as well. Lifting surface computations of the rotor loads performed by Caradonna & Tung correlated better when the experimental wake geometry rather than the Kocurek & Tangler model was used. Oddly enough, the best correlation was acheived when the wake model used in the calculations had even less contraction than the experiment. The source of the discrepancy between the experiment and both the present wake model and the Kocurek & Tangler model may be in part due to recirculation in the test chamber.

As for the Ballard et al. rotor, part of cause of the overprediction of the thrust for the Caradonna & Tung rotor is due to the effect of viscosity. The Reynolds numbers in the Caradonna & Tung experiment is significantly higher than for the experiment of Ballard et al., but the actual lift curve slope of the NACA 0012 is lower than the inviscid lift curve slope. No attempt at estimating angle of attack correction based on a viscous analysis was performed for these two cases, as was done for the Ballard et al. rotor. Instead, the angle of attack change required to match the experimental lift coefficient at the 50% radius station was estimated based on the difference in the computed and actual lift coefficients there. The correction was found to be about 1°, so the collective pitch was reduced by that amount, giving $\theta_{.75} = 7°$. The lift coefficient distributions for the two tip Mach numbers, $M_{tip} = 0.439$ and $M_{tip} = 0.877$, are shown in Figures 4.43 and 4.44; the agreement for both tip speeds is much better. The wake geometries for the

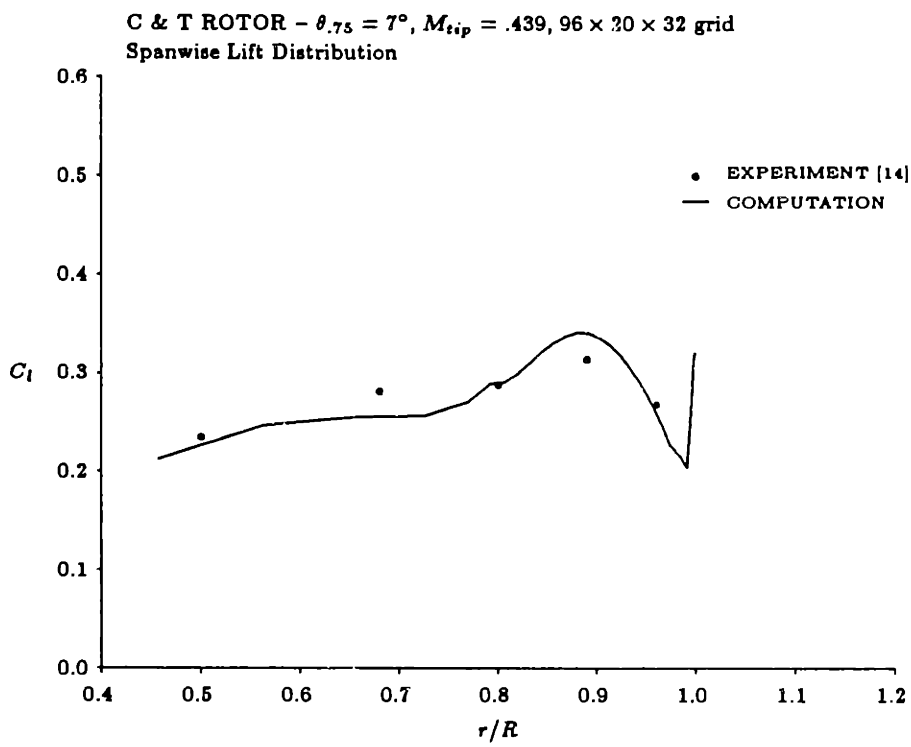

Figure 4.43: Spanwise lift coefficient distribution, Caradonna & Tung rotor, $M_{tip} = 0.439$, reduced collective

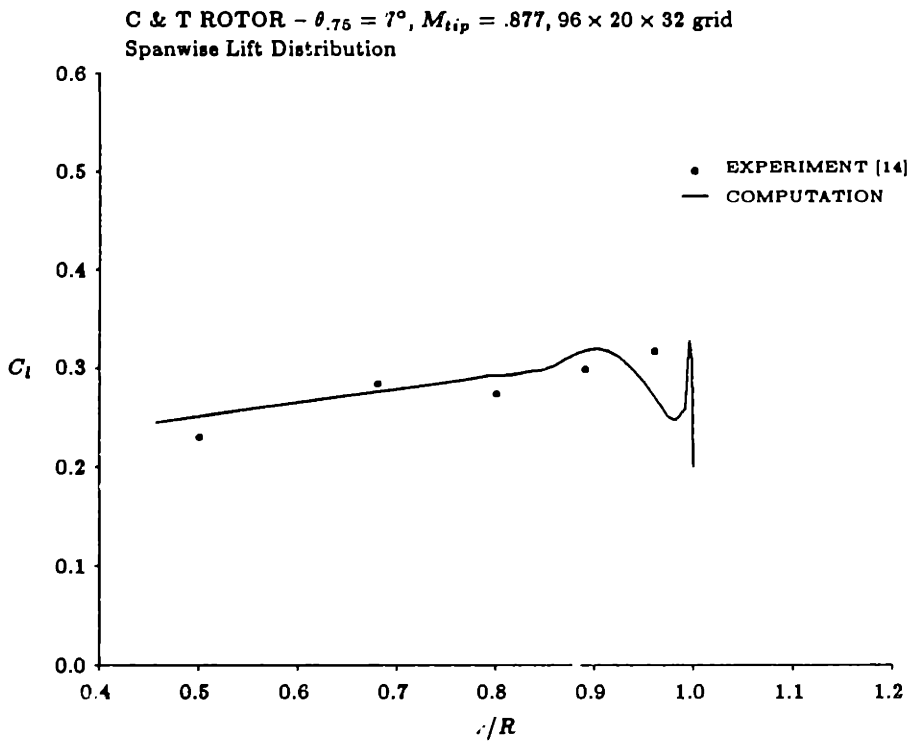

Figure 4.44: Spanwise lift coefficient distribution, Caradonna & Tung rotor, $M_{tip} = 0.877$, reduced collective

two cases are seen in Figures 4.45 and 4.46. The wake for the $M_{tip} = 0.439$

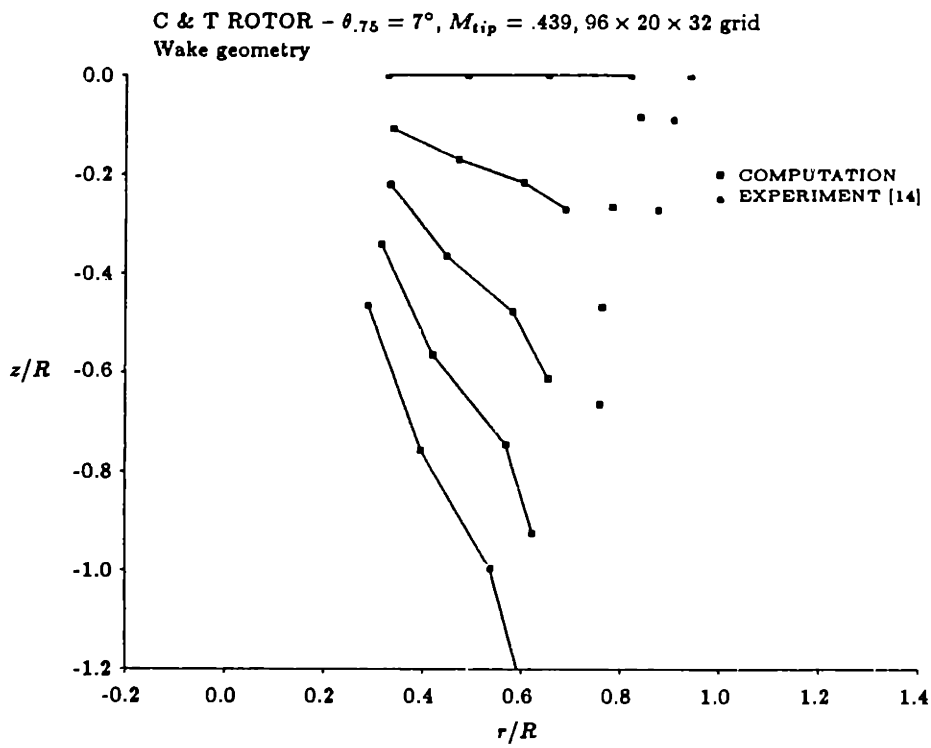

Figure 4.45: Wake geometry, Caradonna & Tung rotor, $M_{tip} = 0.439$, reduced collective

solution shows very good agreement with the experimental descent rate, but still has a much more rapid contraction. The wake for the higher tip speed is in much poorer agreement.

Figure 4.47 shows the computed and measured pressure coefficients for the corrected collective pitch and $M_{tip} = 0.439$. Very good agreement is seen over the whole rotor now, which is especially good considering the differences in the computed and experimental wake geometries. The comparison of surface pressures for the $M_{tip} = 0.877$ case is shown in Figure 4.48; again,

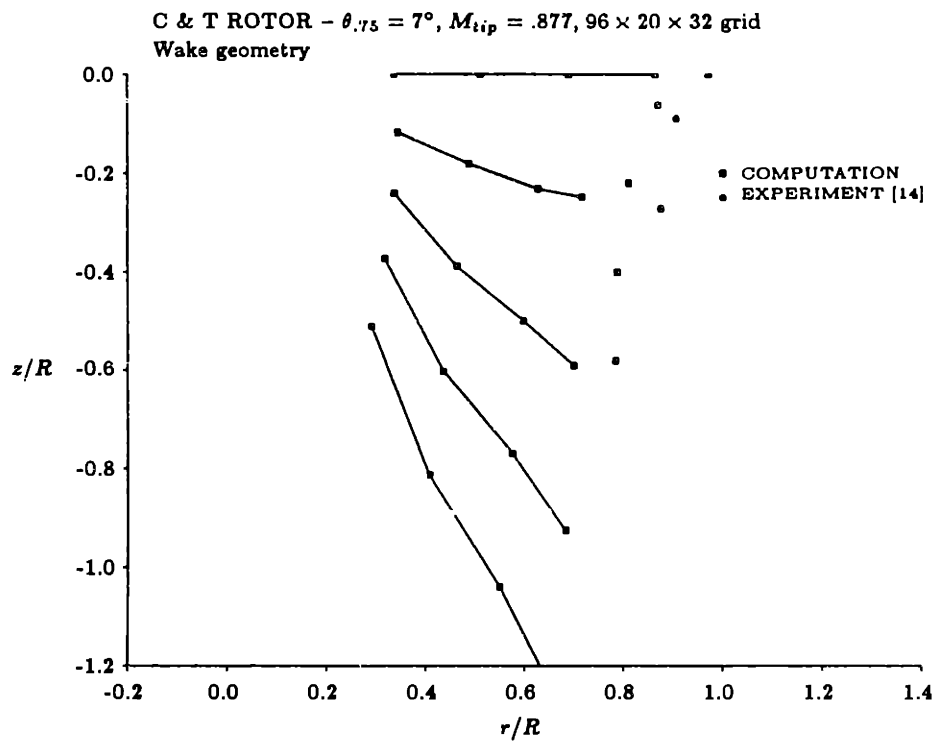

Figure 4.46: Wake geometry, Caradonna & Tung rotor, $M_{tip} = 0.877$, reduced collective

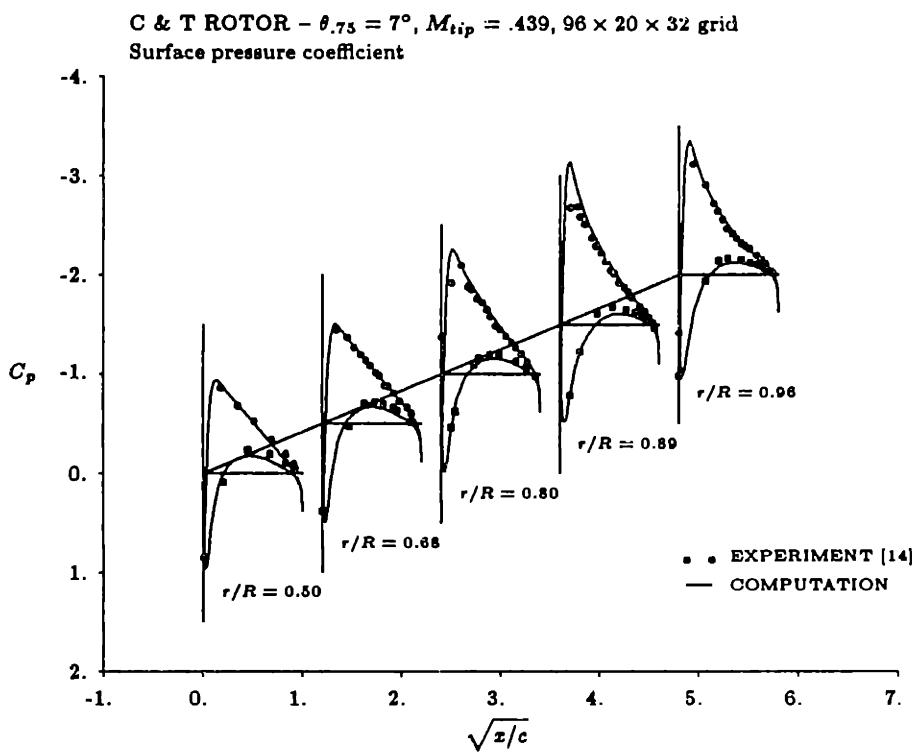

Figure 4.47: Surface pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.439$, reduced collective

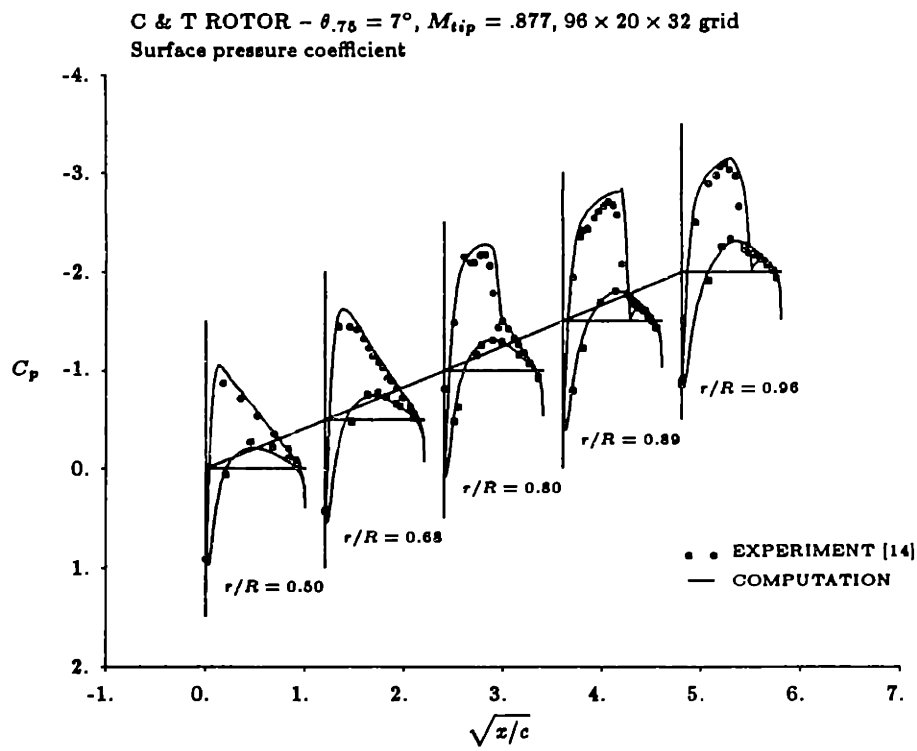

Figure 4.48: Surface pressure distribution, Caradonna & Tung rotor, $M_{tip} = 0.877$, reduced collective

the agreement is better than before, especially over the inboard sections of the blade. The discrepancies over the last three span stations are larger, again as might be expected given the transonic nature of the flow there. Also, the wake geometry computed for this case shows much worse agreement with experiment, both in contraction and in axial descent rate.

As mentioned above, the experiment showed that the wake geometry was insensitive to the tip speed; the present computations do not show the same behavior. Part of the reason for this lies in the algorithm used to set the strengths of the wake vortices. The idea of basing the roll up schedule on

the spanwise gradient of bound circulation is derived from lifting line theory. In effect, there is the implicit assumption that the trailing vortex sheet separates only from the sharp trailing edge, and that the sheet is planar, at least initially. Clearly this assumption is violated near the tip of the rotor blade, where the flow is highly three dimensional. There is separation around the tip and a rapid rolling up of the tip vortex into tight spiral over the rotor blade. Secondly, it is also assumed that the only vorticity in the wake is related to the spanwise lift distribution, and the local lift is in turn related to the bound circulation by the equation

$$\rho_\infty \Omega r \Gamma = \frac{1}{2} \rho_\infty (\Omega r)^2 c C_l, \tag{4.28}$$

where C_l is the lift coefficient. For a transonic flow, this relation no longer holds; there is additional vorticity that is generated by the shocks in the flow. If the shocks are weak, the shock vorticity is neglible. However, if the shock is sufficiently strong, the contribution of the shock vorticity wake may be significant.

These points are illustrated by Figure 4.49 which shows the spanwise bound circulation distribution for the last two cases presented, both with $\theta_{.75} = 7°$. Over the inboard section of the blade, the bound circulation is nearly identical for both the subsonic and transonic cases, but there are significant differences near the tip due to the shock. The maximum bound circulation differs considerably for the subsonic and transonic cases. In Figure 4.50, the lift coefficients, which were computed from integration of the surface pressures, are shown for the two cases. Note that qualitatively, the C_l distributions are more similar than the bound circulation distributions.

For the transonic case here, the shock is quite strong, and the shock vorticity is significant. This can be seen in Figures 4.51 and 4.52, which show contours of constant entropy approximately 1 chord behind the rotor blade trailing edge for the two cases at $\theta_{.75} = 7°$. For the subsonic case, the rolled up tip vortex in the near wake, as well as the prescribed tip vortex

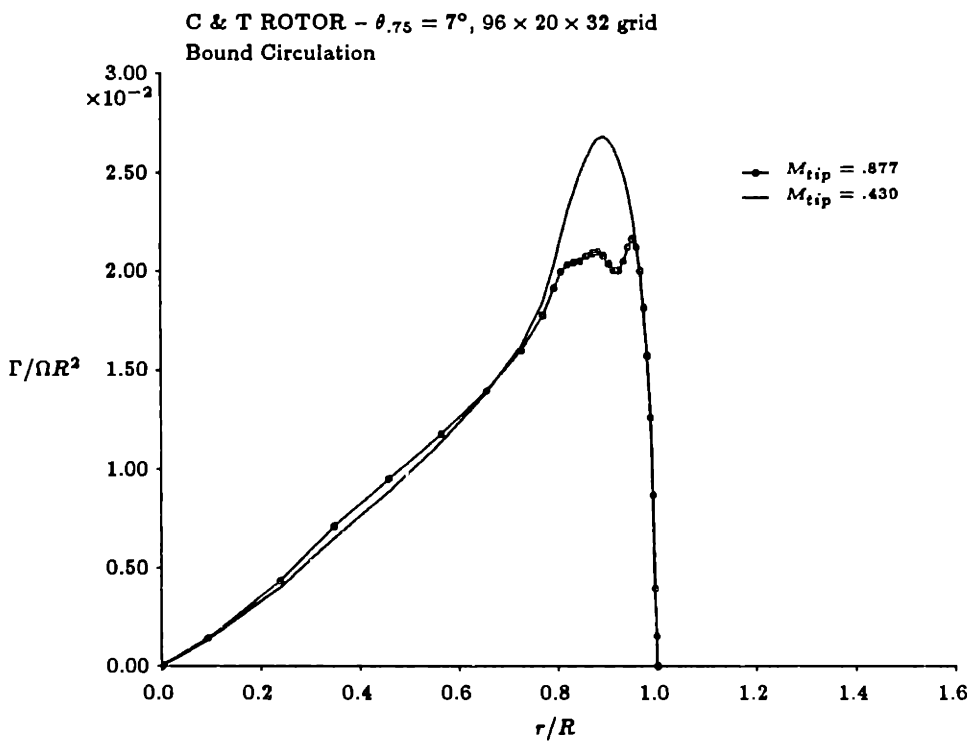

Figure 4.49: Comparison of computed bound circulation, Caradonna & Tung rotor, subsonic and transonic tip speed

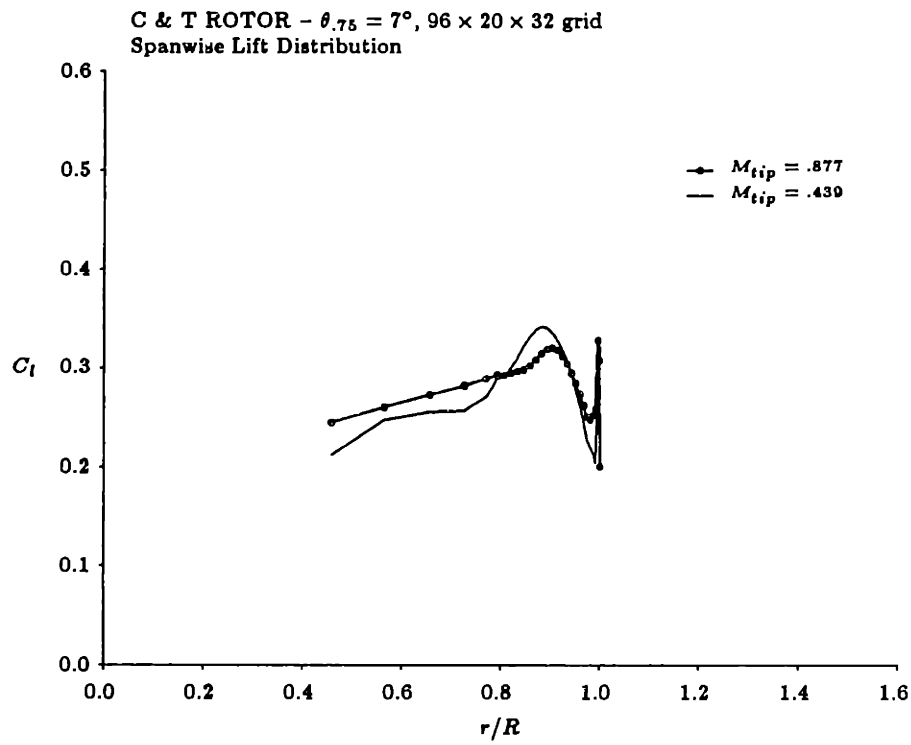

Figure 4.50: Comparison of computed C_l distribution, Caradonna & Tung rotor, subsonic and transonic tip speed

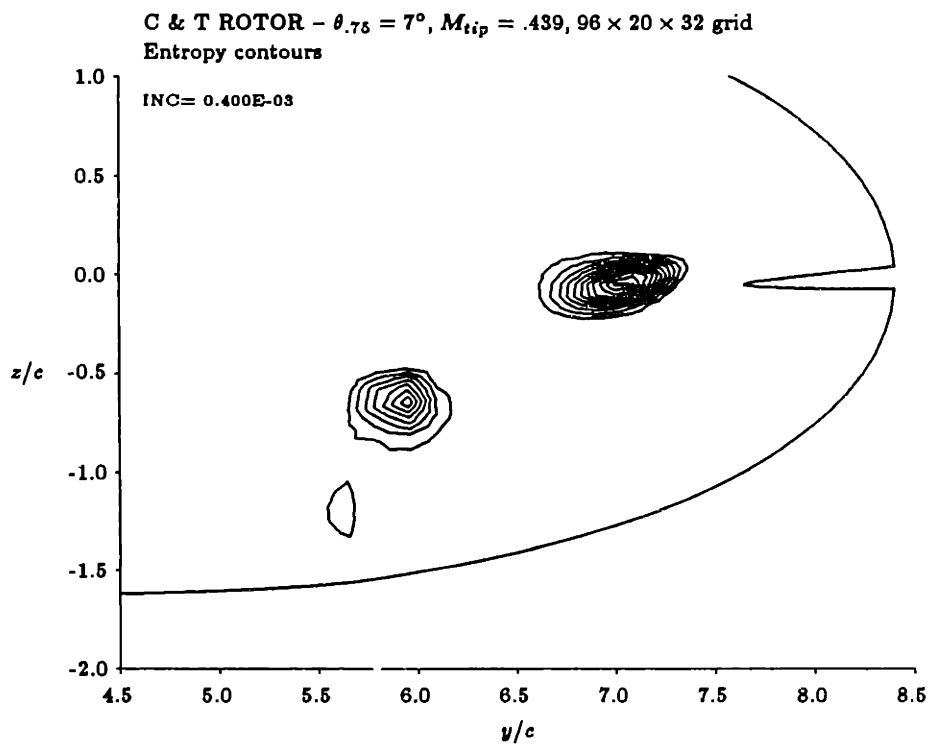

Figure 4.51: Entropy contours, $\approx 1c$ behind trailing edge, Caradonna & Tung rotor, $M_{tip} = 0.439$

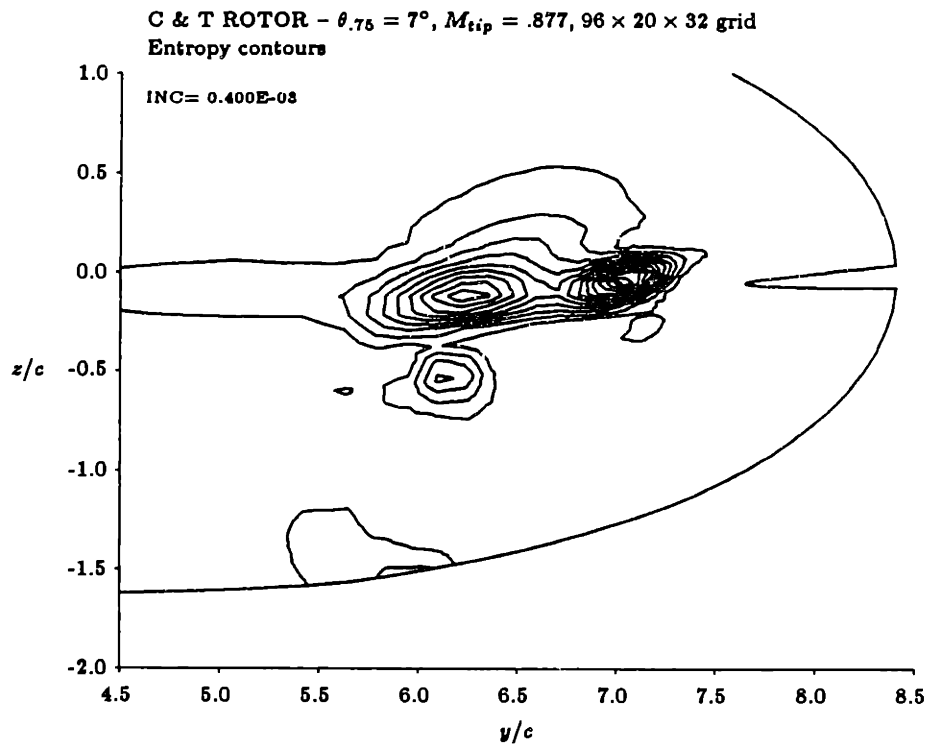

Figure 4.52: Entropy contours, $\approx 1c$ behind trailing edge, Caradonna & Tung rotor, $M_{tip} = 0.877$

from the preceding blade, are clearly identifiable. For the transonic case, the two vortices are still easily identified, but in addition there is another wake structure inboard of the near wake tip vortex trailing from the blade and above the prescribed vortex. The source of this entropy is the shock on the rotor blade. Note that the maximum entropy in this shock wake is comparable to the entropy level in the tip vortex (0.016 versus 0.022). From Crocco's theorem, it is clear that the entropy rise due to the shock is related to the shock generated vorticity. Somehow, this vorticity must get rolled up into the wake, and in setting the strengths of the wake vortices,

the contribution of the shock vorticity must be included.

The relation between the wake vorticity and the spanwise lift distribution needs to be modified in order to account for the shock. The contribution of the shock vorticity to the wake is further complicated by the fact that there is not only streamwise vorticity, but cross-stream vorticity due to the variations in the shock strength. The relation between the wake vortex strengths and the spanwise lift distribution is unclear. Furthermore, the strong nonlinearities and highly three dimensional nature of the flow near the tip may require a more sophisticated roll up model than one based on the spanwise gradient of the lift.

Modification of the wake roll up algorithm to treat transonic tip speeds has not been attempted. The fact that the wake vortex strengths cannot be directly determined from the bound circulation for a shocked flow is of little practical importance in hovering flight. Real rotors do not run at transonic tip speeds in hover, due to the rapid drag rise and hence increased power requirements at such operating conditions. However, in forward flight at high advance ratios, the tip will reach transonic speeds and shocks will form. For wake modeling in forward flight, the shock vorticity will need to be taken into account. It may be possible, with sufficient resolution in the near wake, to develop a wake model based on the computed structure of the near wake.

4.5 Summary

With the use of the perturbation scheme, the solution procedure for the flow around a hovering rotor has been developed. The approach is to couple a free wake algorithm to compute the geometry and influence of the semi-infinite vortex wake with an Euler solution for the rotor blade near field. Calculations have been performed for several hovering rotor cases, and the need for inclusion of the wake and for the use of the perturbation scheme

has been demonstrated.

It was found in the course of the investigation that the blade loading near the tip was remarkably insensitive to the inclusion or neglect of the wake. This suggests that the initial stages of the wake roll up and tip vortex formation can be computed without the need for an accurate wake model. However, the computation of details of the wake roll up will inevitably allow improved wake models to be developed.

In all the cases, the computed wake contraction was greater than the experimentally observed wake contraction. It is not clear exactly what the cause of this discrepancy is. It is due in part to difficulties in obtaining good experimental data because of effects such as recirculation in the test chambers. Since the computed wake contraction depends upon the far wake model, this should be examined and improved upon.

The discretization of the vortex wake and the manner in which the wake vortex strengths were determined worked well for most of the cases. Difficulties arose with the transonic cases attempted, indicating that the simple approach used is inadequate for the highly nonlinear flow field when a shock is present. This is primarily an academic problem, since practical hovering flows are not transonic. However, it does suggest that more sophisticated wake roll up models may be needed to treat the case of a rotor in forward flight.

Chapter 5

Conclusions

In this thesis, three topics have been investigated, all of them aimed at the main goal of the work, namely to predict the flow around a helicopter rotor in hover using a numerical solution of the Euler equations. The first topic, addressed in chapter 2, is the question of whether numerical solutions of the Euler equations yield a realistic model of the trailing vortex wake of a lifting wing. This question is of interest because it was originally hoped that the Euler equations might yield a model of the tip vortex structure of a helicopter rotor. The second topic was the computation of the steady interaction between a streamwise vortex and a wing, which was described in chapter 3. This problem is a model of the interaction of a rotary wing tip vortex with a rotor blade, and was used to develop an approach by which the rotor wake influence could be accurately included in the Euler solution algorithm. In chapter 4, a method for the computation of flow of a hovering helicopter rotor was presented. The unique features of the scheme are the coupling of the Euler solver to a free wake algorithm to account for the influence of the wake, and the use of the Buning & Steger perturbation scheme to introduce the wake vorticity into the Euler computational domain to eliminate the numerical diffusion of vorticity.

A summary of the results of the investigation into each of the above

topics is presented below, along with future recommendations.

5.1 Euler Solutions and Wake Structure

Using the finite volume algorithm of Jameson & Baker, the flow around two lifting wings has been computed and the results compared to experiment. The computed and experimental surface pressures for the ONERA M6 wing, which is a transonic test case, were in good agreement. The agreement between the computed and experimental surface pressures for a wing tested by Weston at a low Mach number was not as good, primarily due to flow angularity in the wind tunnel which affected the spanwise distribution of the lift. Computations on two grids with different grid point distributions and different representations of the tip geometry were performed for the latter case. The details of the load distribution at the tip were quite different for the two grids, indicating that solutions are sensitive to the geometric details in this region. However, the differences were local to the tip region, and the differences between the pressure distributions inboard of the tip and between the computed wake structures were minimal.

The computed wake structure was compared to experimental wake surveys for the Weston test case. Although the position of the computed tip vortex agreed well with experiment, the detailed structure was quite different. The computed wakes had a lower total pressure deficit in comparison to experiment, which is attributable to the neglect of viscous effects in the computation. More puzzling is the presence of a velocity deficit in the core of the tip vortex in the computations, compared to the measured velocity excess. This suggests that the tip vortex formation process in a numerical solution of the Euler equations is very different than the physical process, and does not yield a reliable model of the detailed wake flow. In addition, it is also noted that the effect of the artificial viscosity of the scheme and the decreasing grid resolution in the wake region results in a non physical

diffusion of the tip vortex as it is convected downstream. This results in a more unrealistic wake structure the further one goes downstream.

The differences in the computed and experimental wake structures suggests that the Euler equations cannot be used for determining the structure of a vortical wake of a lifting wing, at least for low Mach numbers. Detailed wake data over a range of Mach numbers is needed to provide a data base for improving our understanding the results of Euler computations of the wake structure. A better understanding of the process by which the flow separates and the wake forms in a numerical solution of the Euler equations is also needed. This will require very high grid resolution in the tip region. In addition, solutions with high resolution in the wake of a lifting wing may be useful in understanding the convection of the wake. Both these objectives will probably be best satisfied by some form of grid embedding to provide local refinement in the tip and wake regions.

The fact that the wake diffuses rapidly due to the effects of artificial viscosity and grid stretching also shows that rotary wing flows, in which the tip vortex follows a spiral path, cannot be accurately computed by solving the Euler equations for the entire flow field. There are two reasons for this. First, the diffusion of the tip vortex as it is convected downstream will result in an incorrect computation of the interaction between the vortex and a rotor blade. Second, since the self-induced velocity of the tip vortex depends upon its core size, the descent rate of the vortex will be dependent upon the numerics, not the physics.

5.2 Wing/Vortex Interaction

The computation of the steady interaction between a streamwise vortex and a wing was done using the Buning & Steger perturbation scheme to introduce the vortex into the computational domain. In this scheme, the state vector (i.e., the density, momentum components, and total energy field)

of the isolated vortex that is generated upstream must be known throughout the Euler computational domain. In specifying this flow field, the vortex is not diffused due to numerical dissipation and the coarse grid resolution in the far field, but remains compact throughout the computational domain. This approach also requires that the specified vortical flow satisfy the steady Euler equations. Although an exact solution of the Euler equations cannot be found except in special cases, a flow field that nearly satisfies the Euler equations can be readily constructed for the flows of interest here.

Comparisons to the experimental wing/vortex interaction results of Smith & Lazzeroni have been made, both with and without the use of the perturbation scheme. The computed spanwise lift distributions agree very well with experiment when the perturbation scheme is used, and very poorly when the perturbation scheme is not used. In cases of a strong interaction—that is, when the vortex passes sufficiently close to the wing that its actual path varies significantly from its prescribed path—the basic Buning & Steger approach is modified by simply prescribing the vortex position up to some location near the wing, and then solving the Euler equations using the standard Jameson & Baker algorithm past that point. This acheives the desired effect of avoiding the numerical diffusion of the vortex before it reaches the wing, but allows the correct interaction of the wing and the vortex to be computed. In particular, this approach allowed the computation of a vortex impinging directly on the leading edge of wing, which could not be computed accurately with either the standard Euler method or the unmodified perturbation approach.

5.3 Hovering Rotor Calculations

The primary aim of the work described in this thesis is the development of a method for computing the flow around a hovering helicopter rotor using the Euler equations. This is the topic of chapter 4, in which the method is

presented and computed results are compared to experiment. The unique feature of the scheme is the splitting of the solution procedure into two parts: the flow in the immediate vicinity of the blade is found using a finite volume Euler solver, and the solution for the wake geometry is treated using the fast free wake algorithm of Miller. The two parts are solved in a coupled fashion. The solution of the Euler equations yields the spanwise bound circulation distribution of the rotor blade, which is used to set the strengths of the vortices in the free wake solver. The free wake solver in turn yields the geometry and the induced velocity field of the wake, whose influence is included in the Euler solver through the far field boundary conditions and the Buning & Steger perturbation scheme. This allows the interaction between the tip vortex and the rotor blade to be accurately computed.

Comparisons of the computed results to the experimental data of Ballard, Orloff & Luebs, and the data of Caradonna & Tung, have been made. The agreement with the spanwise blade loading is very good for the Ballard, et al. test case after a correction to the collective pitch angle to account for the viscous effects has been made. The comparisons to one of the two test cases of Caradonna & Tung is also in good agreement. The tip Mach number for this case is subsonic. A comparison at a transonic tip speed is not as good. This is due in part to the neglect of the viscous effects, which are significant at transonic speeds.

In all the cases, the wake geometry showed more contraction than was observed in the experiment. In part, this is due to the fact that both experiments were run in enclosed chambers; the effects of flow recirculation are a source of experimental error that is very difficult to quantify. The overprediction of the wake contraction for the Caradonna & Tung cases is also consistent with a prescribed wake method for predicting the wake geometry. The computed axial descent rate of the tip vortex is in good agreement with experiment for the Ballard, et al. data and the subsonic tip speed results

of Caradonna & Tung. The transonic test case shows very poor agreement with the wake geometry in both the axial and radial directions. This is largely due to the manner in which the wake vortex strengths are determined. The vortex strengths are found from the computed blade bound circulation. For a highly transonic case with a strong shock, this does not account for the shock generated vorticity. The tip vortex circulation thus determined is lower than the experimental value, and there is subsequently a greater error in the wake geometry. This last point is moot for the case of a practical rotor in hover, in which the tip speed does not reach transonic values, but does suggest that the bound circulation distribution may not be adequate for determining the tip vortex strength in wake models for forward flight.

One very interesting result to come out of the present investigation was the fact that the computed loads near the rotor blade tip are not sensitive to the inclusion or failure to include the wake influence in the Euler solver. The aerodynamic loads in the vicinity of the tip are primarily influenced by the geometric angle of attack, and the induced velocity field of the wake has only a minor effect. As a result, it can be expected that the initial stages of the wake roll up and tip vortex formation can be examined computationally without necessarily needing an accurate wake model. Also, this suggests that experimental studies of the tip vortex formation will not be sensitive to the development of the wake, but that accurate geometric modeling of the rotor blade tip is clearly necessary.

5.4 Recommendations

There are several areas of future research that should be pursued for the case of a hovering rotor. These are listed below.

1. The current method should be extended to rotors with more than two blades. This is a matter of developing the grid generation capability

to handle such cases. For the two bladed rotors considered here, the tip vortex lies sufficiently far from the rotor that its location could be specified throughout the domain. For rotors with more blades, the blade/vortex interaction will be stronger, since the vortex will pass closer to the rotor blade. For such cases, the vortex location can be prescribed up to a point near the blade, and then the interaction itself may be computed using the standard Euler solution algorithm. This will allow the full power of the current scheme to be realized.

2. Improved wake modeling is needed. In particular, computations of the structure of the tip vortex using the Navier-Stokes equations could provide a better model of the tip vortex than the Lamb core structure assumed here. Also, the model for the far wake should be examined more thoroughly, as it is this part of the wake model that effects the rate at which the wake contracts. Since the largest discrepancies between computation and experiment are in the wake contraction, this is an important issue, and it is necessary to sort out the reasons for the differences. For a real hovering flow, the presence of the fuselage will of course have a very strong effect on the wake development, further complicating the wake modeling task.

3. The current algorithm uses a simple procedure for determining the wake vortex strengths from the computed blade bound circulation. It has been shown that this approach is not accurate for a highly transonic flow with a strong shock. Although this is of academic interest rather than practical importance in hover, it is significant in forward flight at high advance ratios where transonic tip speeds are reached. This implies that if a similar approach is used to determine wake vortex strengths in forward flight, the procedure used here will not be adequate. Further examination of the algorithm used to determine the wake vortex strengths in transonic cases is required.

4. The wake roll up criterion used here is very simple, namely the tip vortex is rolled up from the bound circulation peak to the tip, and the inboard portion of the trailing vortex sheet is divided into four equally spaced vortices. It is desirable to have a more accurate representation of the wake roll up, confirmed by comparison with experimental data of the near wake. Although the fixed wing calculations show that the detailed wake structure is not well predicted, the gross features such as the tip vortex location are well predicted. By using an adaptive grid embedding approach to provide more resolution in the wake of the rotor, it may be possible to provide a more complete picture of the rolled up wake structure and thus eliminate or validate the assumptions inherent in the present roll up criterion. This may be particularly important for unconventional rotor blade shapes with large amounts of twist or taper, for which the present roll up criterion may prove inadequate.

References

[1] R.K. Agarwal and J.E. Deese. Euler calculations for flowfield of a helicopter rotor in hover. AIAA Paper 86-1782CP, in *AIAA 4th Applied Aerodynamics Conference* proceedings, 1986.

[2] Dale A. Anderson, John C. Tannehill, and Richard H. Pletcher. *Computational Fluid Mechanics and Heat Transfer.* McGraw-Hill, 1984.

[3] Holt Ashley and Mårten Landahl. *Aerodynamics of Wings and Bodies.* Dover, 1985.

[4] G.R. Baker. The "cloud-in-cell" technique applied to the roll up of vortex sheets. *Journal of Computational Physics*, 31:76–95, 1979.

[5] J.D. Ballard, K.L. Orloff, and A.B. Luebs. Effect of tip planform on blade loading characteristics for a two-bladed rotor in hover. NASA TM 78615, 1979.

[6] W.F. Ballhaus and F.X. Caradonna. The effect of planform shape on the transonic flow past rotor tips. In *AGARD Conference Proceedings No. 111 on Aerodynamics of Rotary Wings*, 1972.

[7] J. Barche, editor. *Experimental Data Base for Computer Program Assessment.* AGARD, 1979. AGARD AR-138.

[8] G.K. Batchelor. Axial flow in trailing line vortices. *Journal of Fluid Mechanics*, 20:645–658, 1964.

[9] D.B. Bliss. *The Dynamics of Curved Rotational Vortex Lines.* Master's thesis, Massachusetts Institute of Technology, 1970.

[10] C.E. Brown. Aerodynamics of wake vortices. *AIAA Journal*, 11(4):531–536, April 1973.

[11] P.G. Buning and J.L. Steger. Solution of the two-dimensional Euler equations with generalized coodinate transformation using flux splitting. AIAA Paper 82-0971, 1982.

[12] F.X. Caradonna, A. Desopper, and C. Tung. Finite difference modeling of rotor flows including wake effects. In *Proceedings of the Eighth European Rotorcraft and Powered Lift Forum*, 1982.

[13] F.X. Caradonna and M.P. Isom. Subsonic and transonic potential flow over helicopter rotor blades. *AIAA Journal*, 10(12):1606–1612, December 1972.

[14] F.X. Caradonna and C. Tung. Experimental and analytical studies of a model helicopter rotor in hover. *Vertica*, 5:149–161, 1981.

[15] A. Cayley. *A Treatise on Elliptic Functions.* Dover, 2nd edition, 1961.

[16] L.J. Chow, T.H. Pulliam, and J.L. Steger. A general perturbation approach for the equations of fluid dynamics. AIAA Paper 83-1903CP, in *AIAA Computational Fluid Dynamics Conference* proceedings, 1983.

[17] S.-Y. Chung. *Formal Optimization of Hovering Performance Using Free Wake Lifting Surface Theory.* PhD thesis, Massachusetts Institute of Technology, 1986.

[18] D.R. Clark and A.C. Leiper. The free wake analysis: a method for the prediction of helicopter rotor hovering performance. *Journal of the American Helicopter Society*, 15(1):3–11, January 1970.

[19] R.M. Coppersmith, H.H. Youngren, and E.E. Bouchard. Quadrilateral Element Panel Method (QUADPAN). Theoretical Report (Version 3), Lockheed-California Company, LR 30500, 1983.

[20] R. Courant and D. Hilbert. *Methods of Mathematical Physics.* Volume 2: Partial Differential Equations, John Wiley & Sons, 1962.

[21] Germund Dahlquist and Åke Björck. *Numerical Methods. Series in Automatic Computation*, Prentice-Hall, 1974.

[22] M. Drela. *Two-Dimensional Transonic Aerodynamic Design and Analysis Using the Euler Equations.* PhD thesis, Massachusetts Institute of Technology, 1986.

[23] M. Drela and M.B. Giles. Viscous-inviscid analysis of transonic and low Reynolds number airfoils. AIAA Paper 86-1786CP, in *AIAA 4th Applied Aerodynamics Conference*, 1986.

[24] T.A. Egolf and S.P. Sparks. Hovering rotor airload prediction using a full-potential flow analysis with realistic wake geometry. Presented at the 41st Annual Forum of the American Helicopter Society, Ft. Worth, Texas, 1985.

[25] L.-E. Eriksson. Generation of boundary-conforming grids around wing-body configurations using transfinite interpolation. *AIAA Journal*, 20(10):1313–1320, October 1982.

[26] L.-E. Eriksson. A study of mesh singularities and their effects on numerical errors. FFA (Sweden) TN 1984-10, 1984.

[27] L.-E. Eriksson and A. Rizzi. Analysis by computer of the convergence to steady state of discrete approximations to the Euler equations. AIAA Paper 83-1951CP, in *AIAA Computational Fluid Dynamics Conference* proceedings, 1983.

[28] I.E. Garrick. Nonsteady wing characteristics. In A.F. Donovan and H.R. Lawrence, editors, *Aerodynamic Components of Aircraft at High Speeds*, Princeton University Press, 1957. Section F of Volume VII, Princeton Series in High Speed Aerodynamics and Jet Propulsion.

[29] Alfred Gessow and Garry C. Myers, Jr. *Aerodynamics of the Helicopter.* Frederick Ungar, 1967.

[30] M.B. Giles. *Newton Solution of Steady Two-Dimensional Transonic Flow.* PhD thesis, Massachusetts Institute of Technology, 1985.

[31] A. Jameson and T.J. Baker. Solution of the Euler equations for complex configurations. AIAA Paper 83-1929CP, in *AIAA Computational Fluid Dynamics Conference* proceedings, 1983.

[32] A. Jameson, W. Schmidt, and E. Turkel. Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes. AIAA Paper 81-1259, 1981.

[33] D. Jespersen. Enthalpy damping for the steady Euler equations. NASA TM 86008, 1984.

[34] J.D. Kocurek and L.F. Berkowitz. Velocity coupling—a new concept for hover and axial flow wake analysis and design. In *AGARD Conference Proceedings No. 334 on Prediction of Aerodynamic Loads on Rotorcraft*, 1982.

[35] J.D. Kocurek and J.L. Tangler. A prescribed wake lifting surface hover performance analysis. *Journal of the American Helicopter Society*, 22(1):24–35, January 1976.

[36] Horace Lamb. *Hydrodynamics.* Dover, 6th edition, 1945.

[37] M.T. Landahl. Roll-up model for rotor wake vortices. ASRL TR 194-4, Massachusetts Institute of Technology, 1981.

[38] A.J. Landgrebe. An analytical and experimental investigation of helicopter rotor hover performance and wake geometry characteristics. USAAMRDL Technical Report 71-24, 1971.

[39] A.J. Landgrebe and Jr. M.C. Cheney. Rotor wakes–key to performance prediction. In *AGARD Conference Proceedings No. 111 on Aerodynamics of Rotary Wings*, 1972.

[40] A.J. Landgrebe, R.C. Moffitt, and D.R. Clark. Aerodynamic technology for advanced rotorcraft. *Journal of the American Helicopter Society*, 22(2):21–27, April 1977.

[41] C.H. Liu, J.L. Thomas, and C. Tung. Navier-Stokes calculations for the vortex wake of a rotor in hover. AIAA Paper 83-1676, 1983.

[42] W.J. McCroskey. Special opportunities in helicopter aerodynamics. NASA TM 84396 (also USAAVRADCOM TR-83-A-15), 1983.

[43] R.H. Miller. Rotor hovering performance using the method of fast free wake analysis. *Journal of Aircraft*, 20(3):257–261, March 1983.

[44] R.H. Miller. A simplified approach to the free wake analysis of a hovering rotor. *Vertica*, 6(2):89–95, 1982.

[45] R.H. Miller. Simplified free wake analyses for rotors. FFA (Sweden) TN 1982-7, 1981.

[46] R.H. Miller, T.W. Roberts, and E.M. Murman. Computational methods for wake modeling and blade airload determination in hover and forward flight. *Computers & Mathematics with Applications*, 12A(1):29–52, January 1986.

[47] D.W. Moore. A numerical study of the roll-up of a finite vortex sheet. *Journal of Fluid Dynamics*, 63:225–235, 1974.

[48] L. Morino, Z. Kaprielian, Jr., and S.R. Sipcic. Free wake analysis of helicopter rotors. *Vertica*, 9(2):127–140, 1985.

[49] E.M. Murman and P.M. Stremel. A vortex wake capturing method for potential flow calculations. AIAA Paper 82-0947, 1982.

[50] K.G. Powell, E.M. Murman, E. Perez, and J.R. Baron. Total pressure loss in vortical solutions of the conical Euler equations. AIAA Paper 85-1701, 1985.

[51] T.H. Pulliam. Artificial dissipation models for the Euler equations. AIAA Paper 85-0438, 1985.

[52] A. Rizzi. Numerical implementation of solid-wall boundary conditions for the Euler equations. *Zeitschrift für Angewandte Mathematik und Mechanik*, 58(7):T301–T304, July 1978.

[53] A. Rizzi and L.-E. Eriksson. Computation of flow around wings based on the Euler equations. *Journal of Fluid Mechanics*, 148:45–71, November 1984.

[54] T.W. Roberts and E.M. Murman. A computational method for helicopter vortex wakes. AIAA Paper 84-1554, 1984.

[55] T.W. Roberts and E.M. Murman. Solution method for a hovering helicopter rotor using the Euler equations. AIAA Paper 85-0436, 1985.

[56] N.L. Sankar, B.E. Wake, and S.G. Lekoudis. Solution of the unsteady Euler equations for fixed and rotor wing configurations. *Journal of Aircraft*, 23(4):283–289, April 1986.

[57] W.G. Smith and F.A. Lazzeroni. Experimental and theoretical study of a rectangular wing in a vortical wake at low speed. NASA TN D-339, 1960.

[58] G.R. Srinivasan. Computations of two-dimensional airfoil-vortex interactions. NASA CR 3885, 1985.

[59] G.R. Srinivasan, W.J. McCroskey, and J.D. Baeder. Aerodynamics of two-dimensional blade-vortex interaction. AIAA Paper 85-1560, 1985.

[60] G.R. Srinivasan, W.J. McCroskey, and P. Kutler. Numerical simulation of the interaction of a vortex with stationary airfoil in transonic flow. AIAA Paper 84-0254, 1984.

[61] J.L. Steger and P. Kutler. Implicit finite-difference procedures for the computation of vortex wakes. *AIAA Journal*, 15:581–590, 1977.

[62] J. Steinhoff and K. Suryanarayanan. The treatment of vortex sheets in compressible potential flow. AIAA Paper 83-1881CP, in *AIAA Computational Fluid Dynamics Conference* proceedings, 1983.

[63] R.C. Strawn and F.X. Caradonna. Numerical modeling of rotor flows with a conservative form of the full-potential equations. AIAA Paper 86-0079, 1986.

[64] J.M. Summa. Advanced rotor analysis methods for the aerodynamics of vortex/blade interactions in hover. In *Proceedings of the Eighth European Rotorcraft and Powered Lift Forum*, 1982.

[65] E. Turkel. Acceleration to a steady state for the Euler equations. ICASE Report No. 84-32, 1984.

[66] B. Wedan. private communication.

[67] R.P. Weston. *Refinement of a Method for Determining the Induced and Profile Drag of a Finite Wing from Detailed Wake Measurements.* PhD thesis, Univerisity of Florida, 1981.

Appendix A

Stability Analysis for Multistage Scheme

The determination of the maximum allowable time step of the four stage time integration scheme with frozen dissipation is found from an analysis of the one dimensional linear wave equation with an added fourth order viscosity,

$$\frac{\partial u}{\partial t} = -a\frac{\partial u}{\partial x} - \mu(\Delta x)^3\frac{\partial^4 u}{\partial x^4}. \tag{A.1}$$

Here, μ is the fourth difference viscosity coefficient, a is the wave speed, and Δx is the grid spacing, assumed to be uniform. The function u is assumed to have the form

$$u(x,t) = \hat{u}(t)e^{ikx}, \tag{A.2}$$

where k is the spatial wave number. By approximating the spatial derivatives in Equation (A.1) with second order centered difference formulas, we obtain the semi-discrete equation

$$\frac{d\hat{u}}{dt} = -\frac{a}{\Delta x}\left(i\sin k\Delta x + 16\frac{\mu}{a}\sin^4\frac{k\Delta x}{2}\right)\hat{u}. \tag{A.3}$$

This equation is integrated in time using the four stage algorithm presented in chapter 2, with the dissipation terms frozen at the first stage,

$$
\begin{aligned}
\hat{u}^{(0)} &= \hat{u}^n, \\
\hat{u}^{(1)} &= \hat{u}^{(0)} - \alpha_1 CFL \left(i\hat{u}^{(0)} z_i + \hat{u}^{(0)} z_r \right), && \text{(A.4a)} \\
\hat{u}^{(2)} &= \hat{u}^{(0)} - \alpha_2 CFL \left(i\hat{u}^{(1)} z_i + \hat{u}^{(0)} z_r \right), && \text{(A.4b)} \\
\hat{u}^{(3)} &= \hat{u}^{(0)} - \alpha_3 CFL \left(i\hat{u}^{(2)} z_i + \hat{u}^{(0)} z_r \right), && \text{(A.4c)} \\
\hat{u}^{(4)} &= \hat{u}^{(0)} - \alpha_4 CFL \left(i\hat{u}^{(3)} z_i + \hat{u}^{(0)} z_r \right), && \text{(A.4d)} \\
\hat{u}^{n+1} &= \hat{u}^{(4)}.
\end{aligned}
$$

where $CFL = a\Delta t/\Delta x$ is the Courant-Friedrichs-Lewy number, $z_i = \sin k\Delta x$, and $z_r = 16\mu/a \sin^4 \frac{k\Delta x}{2}$. The multistage coefficients are the standard coefficients given in chapter 2, and repeated here for convenience:

$$\alpha_1 = \frac{1}{4},\ \alpha_2 = \frac{1}{3},\ \alpha_3 = \frac{1}{2},\ \alpha_4 = 1.$$

The multistage integration can be written in the form

$$\hat{u}^{n+1} = \hat{u}^n G(z\,CFL), \quad z = z_r + iz_i \tag{A.5}$$

where $G(z\,CFL)$ is the amplification factor. Stability requires that the condition

$$|G| \leq 1 \tag{A.6}$$

be satisfied.

Figure A.1 shows contours of constant $|G|$ in the z plane. The contour increment is 0.1 and the outermost contour is $|G| = 1$. That is, the region of stability lies inside the outermost contour. The stability limit for the inviscid equation ($\mu = 0$) is given by the intersection of the $|G| = 1$ contour with the imaginary axis,

$$z_i\,CFL = 2\sqrt{2}. \tag{A.7}$$

It is seen that z_i has a maximum of 1 when $k = \pi/2\Delta x$. Thus the scheme is stable for

$$CFL \leq 2\sqrt{2}, \tag{A.8}$$

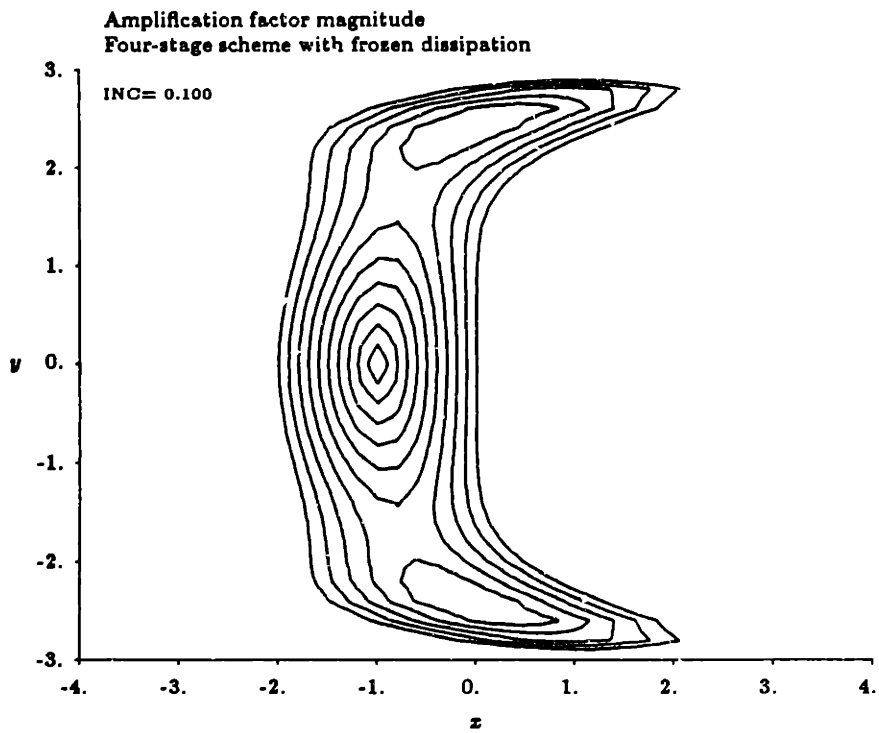

Figure A.1: Contours of constant amplification factor magnitude $|G|$

as stated in chapter 2.

When μ is positive, the locus of the amplification factor for the scheme lies in the left half z plane. The limit on the coefficient μ is determined by the intersection of the $|G| = 1$ contour with the real axis, which gives

$$16\,CFL\,\frac{\mu}{a} \leq 2. \tag{A.9}$$

Also it should be noted that the contour of amplification factor equal to 1 intersects the imaginary axis with a positive slope. This means that as the fourth difference smoothing coefficient is increased from zero, the CFL number must be lowered from $2\sqrt{2}$ in order for the scheme to remain stable,

although for moderate values of μ the reduction is small. For most of the cases in this thesis, *CFL* was taken to be equal to 2.8, very nearly the the maximum allowable for the inviscid equation. That this caused no stability problems may be attributed to the fact that the computation of the time step for each cell, which was based on the mean projected areas in the x, y, and z coordinate directions, was rather conservative. Also, since most of the calculations presented here were for highly subsonic flows, the nonlinearities were weak, rendering the linear analysis presented here to be adequate for determining the stability limit of the full Euler equations. In fact, the one case in this thesis that required a significant reduction of the *CFL* was the transonic hover case, with a very strong ($M_1 \approx 1.5$) shock.

Appendix B

Enthalpy Damping

The basis for enthalpy damping has been derived by Jameson, et al. [32] and Jespersen [33] using a heuristic argument based on equations for an unsteady, inviscid, isentropic, irrotational ideal gas. The Euler equations for such a flow may be reduced to a single equation for a scalar potential. To derive this potential equation, the Euler equations are first written in non-conservative form,

$$\frac{1}{\rho}\frac{D\rho}{Dt} = -\nabla\cdot\vec{u}, \tag{B.1a}$$

$$\frac{D\vec{v}}{Dt} = -\frac{\nabla p}{\rho}, \tag{B.1b}$$

$$\frac{Ds}{Dt} = 0, \tag{B.1c}$$

$$s = \ln\frac{p}{\rho^{\gamma}}. \tag{B.1d}$$

where D/Dt is the material derivative, $\frac{\partial}{\partial t} + \vec{u}\cdot\nabla$. If the flow is isentropic, then

$$\frac{p}{\rho^{\gamma}} = 1 \tag{B.2}$$

and this allows a variable P to be defined as

$$P \equiv \int \frac{dp}{\rho} = \frac{a^2}{\gamma - 1} \tag{B.3}$$

where a is the speed of sound and is defined by

$$a^2 = \frac{dp}{d\rho}. \tag{B.4}$$

The Euler equations can now be reduced to the following form:

$$\frac{DP}{Dt} = -a^2 \nabla \cdot \vec{u}, \tag{B.5a}$$

$$\frac{D\vec{u}}{Dt} = -\nabla P. \tag{B.5b}$$

The assumption that the flow is irrotational allows the velocity to be written as the gradient of a scalar potential,

$$\vec{u} = \nabla \phi, \tag{B.6}$$

and this can be used to reduce the momentum equation to

$$\nabla \left(\frac{\partial \phi}{\partial t} + \frac{\nabla \phi \cdot \nabla \phi}{2} + P \right) = 0, \tag{B.7}$$

which yields

$$\frac{\partial \phi}{\partial t} + \frac{\nabla \phi \cdot \nabla \phi}{2} + P = f(t), \tag{B.8}$$

where $f(t)$ is an arbitrary function of time. If the flow is steady and uniform at infinity, then $f(t) = H_\infty$, where H_∞ is the freestream total enthalpy.

By using the momentum equation to eliminate P, the continuity equation can be written in the following form:

$$\frac{\partial^2 \phi}{\partial t^2} + 2 \nabla \phi \cdot \nabla \frac{\partial \phi}{\partial t} + \nabla \phi \cdot \nabla \left(\frac{\nabla \phi \cdot \nabla \phi}{2} \right) = a^2 \nabla^2 \phi. \tag{B.9}$$

This may be rewritten, using Equation (B.6), as

$$\left(\frac{\partial^2 \phi}{\partial t^2} + \vec{u} \cdot \nabla \frac{\partial \phi}{\partial t} \right) + \vec{u} \cdot \nabla \left(\frac{\partial \phi}{\partial t} + \frac{\nabla \phi \cdot \nabla \phi}{2} \right) = a^2 \nabla^2 \phi. \tag{B.10}$$

Now, using the idea introduced by Garrick [28], let the symbol $\vec{u}_c$ be used to denote that the velocity is to be treated as constant under differentiation. With this notation, Equation (B.10) may be written in the following form:

$$\frac{\partial}{\partial t} \left(\frac{\partial \phi}{\partial t} + \vec{u}_c \cdot \nabla \phi \right) + \vec{u}_c \cdot \nabla \left(\frac{\partial \phi}{\partial t} + \vec{u}_c \cdot \nabla \phi \right) = a^2 \nabla^2 \phi, \tag{B.11}$$

or

$$\frac{1}{a^2}\left(\frac{\partial}{\partial t}+\vec{u}_c\cdot\nabla\right)^2\phi=\nabla^2\phi. \tag{B.12}$$

This equation may be interpreted as a wave equation for ϕ following the particle path. By applying the coordinate transformation

$$\begin{aligned}\xi &= x-u_ct,\\ \eta &= y-v_ct,\\ \varsigma &= z-w_ct,\end{aligned}$$

Equation (B.12) may be written as

$$\frac{1}{a^2}\frac{\partial^2\phi}{\partial t^2}=\frac{\partial^2\phi}{\partial\xi^2}+\frac{\partial^2\phi}{\partial\eta^2}+\frac{\partial^2\phi}{\partial\varsigma^2}, \tag{B.13}$$

which is the same as Equation (2.31) in chapter 2.

The basis of the enthalpy damping concept is to add a term proportional to $\partial\phi/\partial t$ to the left hand side of Equation (B.13), yielding the telegraph equation,

$$\frac{1}{a^2}\frac{\partial^2\phi}{\partial t^2}+\alpha\frac{\partial\phi}{\partial t}=\frac{\partial^2\phi}{\partial\xi^2}+\frac{\partial^2\phi}{\partial\eta^2}+\frac{\partial^2\phi}{\partial\varsigma^2}, \tag{B.14}$$

which is seen to be identical to Equation (2.32). Now reversing the entire transformation to get the continuity equation back into something like its original form, we get

$$-\frac{1}{a^2}\frac{DP}{Dt}+\alpha\frac{\partial\phi}{\partial t}=\nabla^2\phi, \tag{B.15}$$

which becomes, after substituting for P and $\nabla\phi$,

$$\frac{1}{\rho}\frac{D\rho}{Dt}-\alpha\frac{\partial\phi}{\partial t}=-\nabla\cdot\vec{u}. \tag{B.16}$$

Writing the above equation in conservative form, we get

$$\frac{\partial\rho}{\partial t}+\nabla\cdot(\rho\vec{u})-\alpha\rho\frac{\partial\phi}{\partial t}=0. \tag{B.17}$$

The final form of the continuity equation can be obtained by going back to Equation (B.8), and considering the case in which the flow at infinity is uniform and steady. In this case, we have

$$\frac{\partial \phi}{\partial t} = H_\infty - \left(\frac{\nabla\phi \cdot \nabla\phi}{2} + P \right). \tag{B.18}$$

But this equation may be rewritten by noting that

$$\frac{\nabla\phi \cdot \nabla\phi}{2} + P = \frac{\vec{u}\cdot\vec{u}}{2} + \frac{a^2}{\gamma - 1} = H, \tag{B.19}$$

where H is the total enthalpy of the flow. This allows Equation (B.17) to be put into the form

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\vec{u}) + \alpha\rho\,(H - H_\infty) = 0. \tag{B.20}$$

To complete the equation set, Equation (B.20) is combined with the non-conservative form of the momentum equation and with the entropy equation in order to get the conservative form of the momentum and energy equations. The resulting equation set is

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\vec{u}) + \alpha\rho\,(H - H_\infty) = 0, \tag{B.21a}$$

$$\frac{\partial \rho\vec{u}}{\partial t} + \nabla\cdot(\rho\vec{u}\vec{u}) + \nabla p + \alpha\rho\vec{u}\,(H - H_\infty) = 0, \tag{B.21b}$$

$$\frac{\partial \rho E}{\partial t} + \nabla\cdot(\rho H\vec{u}) + \alpha\rho H\,(H - H_\infty) = 0. \tag{B.21c}$$

Jespersen [33] has shown that different forms of the energy equation are obtained by combining the modified continuity equation (B.20) with different non-conservative forms of the energy equation. He uses this analysis to derive a "rational" enthalpy damping formulation that avoids Jameson's ad hoc fix to the energy equation, which consists of replacing the last term on the left hand side with $\alpha\rho\,(H - H_\infty)$.

Appendix C

Program Listings

This appendix contains three program listings. The first is the listing of the basic finite volume Euler solver described in chapter 2. This program solves for the flow around an isolated semispan wing with a plane of symmetry. The second listing is of the Euler solver incorporating the perturbation scheme as discussed in chapter 3. This program was developed to computed wing/vortex interactions. The final code is the rotary wing Euler solver and free wake solver, discussed in chapter 4. This code computes the flow field of a two bladed rotor in hover.

C.1 Basic Euler code

```
C*************************************************************************
C                                                                        *
C            PROGRAM EULER:  Tom Roberts, May 1986 (rev.)                *
C                                                                        *
C   A program to solve the Euler equations of motion for an inviscid,   *
C   ideal gas in three dimensions around a wing with a plane of sym-     *
C   symmetry at subsonic and transonic free stream Mach numbers,         *
C   using Jameson's four-stage finite volume scheme.                     *
C                                                                        *
C   This code is written to vectorize efficiently (I think) on the       *
C   Cray.                                                                *
C                                                                        *
C                                                                        *
C*************************************************************************
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/VAR / U(5,-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             P(-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             R(5,ISIZ+1,JSIZ+1,KSIZ+1),NCELLS
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      COMMON/CELL/ FACEX(3,ISIZ,0:JSIZ,KSIZ),FACEY(3,ISIZ,0:JSIZ,KSIZ),
     &             FACEZ(3,ISIZ,0:JSIZ,KSIZ),VOLUME(ISIZ,0:JSIZ,KSIZ)
      COMMON/BOUN/ PRESYM(ISIZ,JSIZ),PRESWG(ISIZ,KSIZ),WALL(ISIZ,JSIZ)
      COMMON/DISP/ D(5,ISIZ,JSIZ,KSIZ),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH,CIN
      COMMON/TIME/ DT(0:ISIZ+1,0:JSIZ+1,0:KSIZ+1),ALPHA1,ALPHA2,ALPHA3
      COMMON/AREA/ AXM(ISIZ,JSIZ,KSIZ),AYM(ISIZ,JSIZ,KSIZ),
     &             AZM(ISIZ,JSIZ,KSIZ)
      COMMON/BOU2/ WUN(3,0:ISIZ+1,0:KSIZ+1)
      REAL KAP2, KAP4, MACH
C
C---  open necessary input and output files:
C     WING.BIN contains the grid (input);
C     RESIDUAL.PLT contains the residual history (output);
C     PRESSURE.PLT contains the surface pressure coefficients (output);
C     EULER.DMP contains the entire state vector (output);
C     INPUT.DAT contains the problem parameters (input, obviously).
C
C---  In addition, a file named RESTART.DMP is required if the code is
C     being restarted from a previous solution.  This file is identical
C     in format to the output file EULER.DMP.
C
      OPEN (1,STATUS='OLD',FORM='UNFORMATTED',FILE='WING.BIN')
      OPEN (2,STATUS='NEW',FORM='UNFORMATTED',FILE='RESIDUAL.PLT')
      OPEN (3,STATUS='NEW',FORM='UNFORMATTED',FILE='PRESSURE.PLT')
      OPEN (4,STATUS='NEW',FORM='UNFORMATTED',FILE='EULER.DMP')
      OPEN (7,STATUS='OLD',FILE='INPUT.DAT')
      CLOSE (4)
C
      PI = 4.*ATAN(1.)
C
C---  Read in the coordinates of the grid cell vertices and initialize
C     the grid from a binary data file.  An O-O mesh topology is assumed.
C
C     NOTE: the grid indices (i,j,k) are assumed to form a LEFT-HANDED
C     coordinate system in the computational space.  If a right-handed
C     grid is used, the calculation of the finite volume cell face
C     projected areas will be screwed up (some normals pointing
C     into the cell, some pointing out) thus wreaking havoc upon the
```

```
C     flux calculations.  (The historical background for the choice of
C     left-handed coordinates is that this code was written to use grids
C     generated by Eriksson's code, which are left-handed.  Don't ask me
C     why.)
C
      READ (1) IMAX,JMAX,KMAX
      IM = IMAX - 1
      JM = JMAX - 1
      KM = KMAX - 1
      NCELLS = IM*JM*KM
      READ (1) (((X(I,J,K), I = 1,IMAX), J = 1,JMAX), K = 1,KMAX),
     &         (((Y(I,J,K), I = 1,IMAX), J = 1,JMAX), K = 1,KMAX),
     &         (((Z(I,J,K), I = 1,IMAX), J = 1,JMAX), K = 1,KMAX)
      CLOSE (1)
      CALL CELL
      CALL NORMAL
      CALL AMEAN
C
C---  read in the problem parameters.  All formats are either 3G10.5
C     (for real input) or 3I5 (for integers).
C     If the code is being restarted from a previous solution, the value
C     input for ITER must be > 1; if you're starting from scratch,
C     ITER = 1
C
      READ (7,1000) AOA,MACH,GAM           ! self explanatory
      READ (7,1000) ALPHA1,ALPHA2,ALPHA3   ! coefs. for time-stepping
      READ (7,1000) KAP2,KAP4,CFL          ! art. visc. and CFL no.
      READ (7,1000) AENTH                  ! enthalpy damping coef.
      READ (7,1001) ITMAX,ITPRIN,ITER
      CLOSE (7)
      WRITE (2) ITER
      WRITE (2) MACH,AOA,CFL,IM,JM,KM
      CLOSE (2)
      WRITE (3) MACH,AOA,CFL,IM,JM,KM
      CALL SURFCO
      CLOSE (3)
C
C--- initialize the state vector;  note that density and pressure are
C    non-dimensionalized by their freestream values, which means that
C    velocities must be non-dimensionized by the freestream speed of
C    sound over the square root of gamma for consistency.
C
      AOA = PI*AOA/180.
      UIN = SQRT(GAM)*MACH*COS(AOA)          ! free stream x-velocity
      WIN = SQRT(GAM)*MACH*SIN(AOA)          ! free stream y-velocity
      HIN = GAM*(MACH**2/2. + 1./(GAM - 1.)) ! free stream total enthalpy
      CIN = SQRT(GAM)
      CALL INIT(ITER,*9998)
C
C---  start Euler solution procedure;  establish boundary conditions,
C     dissipation, determine initial value of residuals, and determine
C     size of local time step
C
    1 CALL BNDRYC
      CALL FLUX
      CALL DTSIZE
      CALL DISSIP
C
C---  call four-stage time stepping procedure
C
      CALL TIMSTP
```

```
C
C---  write out solution (restart file) and surface pressures every
C     ITPRIN iterations, or when done (ITER = ITMAX)
C
      IF (MOD(ITER,ITPRIN). EQ .0. OR .ITER. EQ .ITMAX) THEN
       CALL BNDRYC
       OPEN (3,STATUS='OLD',FORM='UNFORMATTED',ACCESS='APPEND',
     &  FILE='PRESSURE.PLT')
       CALL PROUT(ITER)
       CLOSE (3)
       OPEN (4,STATUS='OLD',FORM='UNFORMATTED',FILE='EULER.DMP')
       WRITE (4) IM,JMAX,KM
       WRITE (4) ((((U(N,I,J,K), N = 1,5), I = 1,IM),
     &                 J = 1,JMAX), K = 1,KM)
       CLOSE (4)
      END IF
C
      ITER = ITER + 1
      IF (ITER. GT .ITMAX) GO TO 10
      GO TO 1
   10 CONTINUE
C
C---  determine average and maximum values of the total enthalpy
C     throughout the field
C
      HMAX = 0.
      HMIN = 10.*HIN
      HTOT = 0.
      SMAX = -100.
      SMIN = 100.
      DO 40 K = 1,KM
        DO 30 J = 1,JM
          DO 20 I = 1,IM
            HTEM = (U(5,I,J,K) + P(I,J,K))/U(1,I,J,K)
            HTOT = HTOT + HTEM
            IF (HTEM. GT .HMAX) THEN
             HMAX = HTEM
             IHMAX = I
             JHMAX = J
             KHMAX = K
            END IF
            IF (HTEM. LT .HMIN) THEN
             HMIN = HTEM
             IHMIN = I
             JHMIN = J
             KHMIN = K
            END IF
            STEM = LOG(P(I,J,K)/U(1,I,J,K)**GAM)
            IF (STEM. GT .SMAX) THEN
             SMAX = STEM
             ISMAX = I
             JSMAX = J
             KSMAX = K
            END IF
            IF (STEM. LT .SMIN) THEN
             SMIN = STEM
             ISMIN = I
             JSMIN = J
             KSMIN = K
            END IF
   20     CONTINUE
```

```
   30   CONTINUE
   40 CONTINUE
      HAVG = HTOT/FLOAT(NCELLS)
      HAVG = HAVG/HIN*100.
      HMAX = HMAX/HIN*100.
      HMIN = HMIN/HIN*100.
      AOA = AOA*180./PI
      WRITE (6,1005) MACH,AOA,CFL,IM,JM,KM,ITMAX
      WRITE (6,1002) HAVG,HMAX,HMIN
      WRITE (6,1003) IHMAX,JHMAX,KHMAX
      WRITE (6,1004) IHMIN,JHMIN,KHMIN
      WRITE (6,1006) SMAX,ISMAX,JSMAX,KSMAX
      WRITE (6,1007) SMIN,ISMIN,JSMIN,KSMIN
C
 1000 FORMAT(3G10.5)
 1001 FORMAT(3I5)
 1002 FORMAT(/2X,'Average Enthalpy',4X,'Maximum Enthalpy',4X,
     & 'Minimum Enthalpy'/2X,16('-'),4X,16('-'),4X,16('-')/
     & F14.5,2F20.5)
 1003 FORMAT(/1X,'Cell of maximum stagnation enthalpy:',3I5)
 1004 FORMAT(1X,'Cell of minimum stagnation enthalpy:',3I5)
 1005 FORMAT(1X,'EULERCODE solution'//1X,'Mach no.:',3X,F6.3,6X,
     & 'Angle of attack (degrees):',F7.3/1X,'Courant no.:',F6.3,
     & 6X,'Grid size:',I4,' x',I4,' x',I4//1X,'Iterations:',I5/)
 1006 FORMAT(/1X,'Maximum entropy:',F8.4,3I5)
 1007 FORMAT(1X,'Minimum entropy:',F8.4,3I5)
C
      GO TO 9999
 9998 WRITE (6,*) ' Input array does not match array size.'
 9999 STOP
      END
```

```
C
C
C  SUBROUTINE INIT:  initializes the state vector for the Euler
C  solver by setting the variables to their freestream values or reading
C  in old solution to restart a calculation
C
      SUBROUTINE INIT(ISTART,*)
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/VAR / U(5,-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             P(-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             R(5,ISIZ+1,JSIZ+1,KSIZ+1),NCELLS
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH,CIN
      REAL MACH
      IF (ISTART. NE .1) GO TO 40
C
C---  new run:  initialize the flow field
C
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            U(1,I,J,K) = 1.
            U(2,I,J,K) = UIN
            U(3,I,J,K) = 0.
            U(4,I,J,K) = WIN
            U(5,I,J,K) = HIN - 1.
            P(I,J,K) = 1.
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      RETURN
C
C---  if code is to be restarted, read in old solution; the file
C     RESTART.DMP is identical format to the output file EULER.DMP
C
   40 CONTINUE
      OPEN (8,STATUS='OLD',FORM='UNFORMATTED',FILE='RESTART.DMP')
      READ (8) I,J,K
      IF (I. NE .IM. OR .J. NE .JMAX. OR .K. NE .KM) THEN
       CLOSE (8)
       RETURN 1
      ELSE
       READ (8) ((((U(N,I,J,K), N = 1,5), I = 1,IM),
     &              J = 1,JMAX), K = 1,KM)
       CLOSE (8)
      END IF
      RETURN
      END
```

```
C
C
C  SUBROUTINE BNDRYC:  sets the far-field and solid wall boundary
C  conditions for Jameson's explicit four-stage scheme.  At the
C  wing surface, the pressure is extrapolated from interior values;
C  far-field conditions are determined by assuming one-dimensional
C  flow normal to the boundary, and using Riemann invariants in
C  the normal direction.
C
      SUBROUTINE BNDRYC
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/VAR / U(5,-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             P(-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             R(5,ISIZ+1,JSIZ+1,KSIZ+1),NCELLS
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      COMMON/CELL/ FACEX(3,ISIZ,0:JSIZ,KSIZ),FACEY(3,ISIZ,0:JSIZ,KSIZ),
     &             FACEZ(3,ISIZ,0:JSIZ,KSIZ),VOLUME(ISIZ,0:JSIZ,KSIZ)
      COMMON/BOUN/ PRESYM(ISIZ,JSIZ),PRESWG(ISIZ,KSIZ),WALL(ISIZ,JSIZ)
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH,CIN
      COMMON/BOU2/ WUN(3,0:ISIZ+1,0:KSIZ+1)
      REAL MACH
C
C---  first, determine the pressure field
C
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            RHO = U(1,I,J,K)
            U1  = U(2,I,J,K)/RHO
            V1  = U(3,I,J,K)/RHO
            W1  = U(4,I,J,K)/RHO
            EN  = U(5,I,J,K)
            P(I,J,K) = (EN - .5*(U1**2 + V1**2 + W1**2)*RHO)*
     &       (GAM - 1.)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      DO 31 K = 1,KM
        P(0,1,K) = P(IM,1,K)
        P(IMAX,1,K) = P(1,1,K)
   31 CONTINUE
      DO 32 I = 1,IM
        P(I,1,0) = P(I,1,1)
        P(I,1,KMAX) = P(IMAX-I,1,KMAX-1)
   32 CONTINUE
C
C---  next, determine the far-field conditions
C
      DO 60 K = 1,KM
        DO 50 I = 1,IM
C
C---  find density, velocity, and pressure at the last point
C     inside the computational domain, (i,J-1,k)
C
          RHOEX = U(1,I,JM,K)
          UEX   = U(2,I,JM,K)/RHOEX
          VEX   = U(3,I,JM,K)/RHOEX
          WEX   = U(4,I,JM,K)/RHOEX
          PEX   =   P(I,JM,K)
C
```

```
C---  find the outward unit normal to the computational domain
C     and determine the normal component of the free stream and
C     extrapolated velocities and sonic speeds
C
          ONX = FACEX(2,I,JM,K)
          ONY = FACEY(2,I,JM,K)
          ONZ = FACEZ(2,I,JM,K)
          ON  = SQRT(ONX**2 + ONY**2 + ONZ**2)
          ONX = ONX/ON
          ONY = ONY/ON
          ONZ = ONZ/ON
C
          UINN = ONX*UIN +           ONZ*WIN
          UEXN = ONX*UEX + ONY*VEX + ONZ*WEX
          CEX = SQRT(GAM*PEX/RHOEX)
          EV4 = UINN + CIN
          EV5 = UEXN - CEX
C
C---  compute the incoming and outgoing Riemann invariants
C
          RIN = UINN - 2.*CIN/(GAM - 1.)
          REX = UEXN + 2.*CEX/(GAM - 1.)
C
C---  supersonic inflow:  all variables specified
C
          IF (EV4. LT .0.) REX = UINN + 2.*CIN/(GAM - 1.)
C
C---  supersonic outflow:  all variables extrapolated
C
          IF (EV5. GT .0.) RIN = UEXN - 2.*CEX/(GAM - 1.)
C
C---  these two lines are the Cray vectorizable statements that do the
C     same thing as the above IF statements
C
C          REX = CVMGM((UINN + 2.*CIN/(GAM - 1.)),(UEXN + 2.*CEX/(GAM -
C     &     1.)),EV4)
C          RIN = CVMGP((UEXN - 2.*CEX/(GAM - 1.)),(UINN - 2.*CIN/(GAM -
C     &     1.)),EV5)
C
C---  determine the normal velocity and the sonic speed at the boundary
C     cell (i,J,k)
C
          UN = (REX + RIN)/2.
          C  = (REX - RIN)*(GAM - 1.)/4.
          IF (UN. GT .0.) THEN
C
C---  outflow boundary:  extrapolate tangential velocity, entropy
C
           UP = UEX + (UN - UEXN)*ONX
           VP = VEX + (UN - UEXN)*ONY
           WP = WEX + (UN - UEXN)*ONZ
           S = PEX/RHOEX**GAM
          ELSE
C
C---  inflow boundary:  free stream tangential velocity, entropy
C
           UP = UIN + (UN - UINN)*ONX
           VP =       (UN - UINN)*ONY
           WP = WIN + (UN - UINN)*ONZ
           S = 1.
          END IF
```

```
C
C---  these four lines are the Cray vectorizable statements that do the
C     same thing as the above IF THEN ELSE block
C
C          UP = CVMGP((UEX + (UN - UEXN)*ONX),(UIN + (UN - UINN)*ONX),
C     &     UN)
C          VP = CVMGP((VEX + (UN - UEXN)*ONY),((UN - UINN)*ONY),
C     &     UN)
C          WP = CVMGP((WEX + (UN - UEXN)*ONZ),(WIN + (UN - UINN)*ONZ),
C     &     UN)
C          S = CVMGP((PEX/RHOEX**GAM),1.,UN)
C
C
C---  determine density, energy;  set conservation variables at boundary
C     cell
C
          ENER = (UP**2 + VP**2 + WP**2)/2. + C**2/(GAM*(GAM - 1.))
          RHO = (C**2/(GAM*S))**(1./(GAM - 1.))
C
          U(1,I,JMAX,K) = RHO
          U(2,I,JMAX,K) = RHO*UP
          U(3,I,JMAX,K) = RHO*VP
          U(4,I,JMAX,K) = RHO*WP
          U(5,I,JMAX,K) = RHO*ENER
          P(I,JMAX,K) = RHO*C**2/GAM
   50   CONTINUE
   60 CONTINUE
C
C---  determine the pressure at the wing and symmetry plane by
C     extrapolation from the interior--set (dP/dn) = 0 at the
C     symmetry plane; the normal momentum equation is used to
C     determine the pressure on the wing.  On the wing, the local
C     coordinates xi,eta,zeta correspond to the indice i,k,j,
C     respectively (that is not a typo).  This makes xi,eta,zeta a
C     right-handed system since i,j,k form a left-handed system.
C     Perfectly clear, right?
C
C---  First, do the symmetry plane
C
      DO 80 J = 1,JM
        DO 70 I = 1,IM
          PRESYM(I,J) = P(I,J,1)
   70   CONTINUE
   80 CONTINUE
C
C---  Next, set do the wing surface
C
      DO 100 K = 1,KM
        DO 90 I = 1,IM
C
C---  determine the velocity and density in cell (i,1,k)
C
          RHO = U(1,I,1,K)
          U1  = U(2,I,1,K)/RHO
          V1  = U(3,I,1,K)/RHO
          W1  = U(4,I,1,K)/RHO
C
C---  determine the cartesian components of the surface velocity
C
          UN = WUN(1,I,K)*U1 + WUN(2,I,K)*V1 + WUN(3,I,K)*W1
          US = U1 - UN*WUN(1,I,K)
```

```
          VS = V1 - UN*WUN(2,I,K)
          WS = W1 - UN*WUN(3,I,K)
C
C---  find the contravariant components of the velocity and the unit
C     normal
C
          UC = US*FACEX(1,I,0,K) + VS*FACEY(1,I,0,K) + WS*FACEZ(1,I,0,K)
          VC = US*FACEX(3,I,0,K) + VS*FACEY(3,I,0,K) + WS*FACEZ(3,I,0,K)
C
          ONXI = WUN(1,I,K)*FACEX(1,I,0,K) + WUN(2,I,K)*FACEY(1,I,0,K) +
     &     WUN(3,I,K)*FACEZ(1,I,0,K)
          ONET = WUN(1,I,K)*FACEX(3,I,0,K) + WUN(2,I,K)*FACEY(3,I,0,K) +
     &     WUN(3,I,K)*FACEZ(3,I,0,K)
          ONZE = WUN(1,I,K)*FACEX(2,I,0,K) + WUN(2,I,K)*FACEY(2,I,0,K) +
     &     WUN(3,I,K)*FACEZ(2,I,0,K)
C
          UC = UC/VOLUME(I,0,K)
          VC = VC/VOLUME(I,0,K)
C
          ONXI = ONXI/VOLUME(I,0,K)
          ONET = ONET/VOLUME(I,0,K)
          ONZE = ONZE/VOLUME(I,0,K)
C
C---  determine dn/dxi, dn/deta
C
          DNXDXI = .5*(WUN(1,I+1,K) - WUN(1,I-1,K))
          DNYDXI = .5*(WUN(2,I+1,K) - WUN(2,I-1,K))
          DNZDXI = .5*(WUN(3,I+1,K) - WUN(3,I-1,K))
C
          DNXDET = .5*(WUN(1,I,K+1) - WUN(1,I,K-1))
          DNYDET = .5*(WUN(2,I,K+1) - WUN(2,I,K-1))
          DNZDET = .5*(WUN(3,I,K+1) - WUN(3,I,K-1))
C
C---  solve for the l.h.s. of the normal momentum equation
C
          RHOUU = RHO*(US*(UC*DNXDXI + VC*DNXDET) +
     &                 VS*(UC*DNYDXI + VC*DNYDET) +
     &                 WS*(UC*DNZDXI + VC*DNZDET))
C
C---  determine the surface pressure gradient
C
          DPDXI = .5*(P(I+1,1,K) - P(I-1,1,K))
          DPDET = .5*(P(I,1,K+1) - P(I,1,K-1))
C
C---  solve for dP/dzeta
C
          DPDZE = (RHOUU - ONXI*DPDXI - ONET*DPDET)/ONZE
C
C---  find the surface pressure
C
          PRESWG(I,K) = P(I,1,K) - .5*DPDZE
   90   CONTINUE
  100 CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE FLUX:  determine the fluxes across each finite volume cell
C  face, and the residual R for each cell.  This is done by averaging
C  the flux vectors between neighboring cells
C
      SUBROUTINE FLUX
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/VAR / U(5,-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             P(-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             R(5,ISIZ+1,JSIZ+1,KSIZ+1),NCELLS
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      COMMON/CELL/ FACEX(3,ISIZ,0:JSIZ,KSIZ),FACEY(3,ISIZ,0:JSIZ,KSIZ),
     &             FACEZ(3,ISIZ,0:JSIZ,KSIZ),VOLUM(ISIZ,0:JSIZ,KSIZ)
      COMMON/BOUN/ PRESYM(ISIZ,JSIZ),PRESWG(ISIZ,KSIZ),WALL(ISIZ,JSIZ)
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH,CIN
      REAL MACH
C
C---  set-up dummy points
C
      DO 20 J = 1,JM
        DO 10 I = 1,IM
          U(1,I,J,KMAX) = U(1,IMAX-I,J,KMAX-1)
          U(2,I,J,KMAX) = U(2,IMAX-I,J,KMAX-1)
          U(3,I,J,KMAX) = U(3,IMAX-I,J,KMAX-1)
          U(4,I,J,KMAX) = U(4,IMAX-I,J,KMAX-1)
          U(5,I,J,KMAX) = U(5,IMAX-I,J,KMAX-1)
          P(I,J,KMAX) = P(IMAX-I,J,KMAX-1)
   10   CONTINUE
   20 CONTINUE
C
      DO 40 K = 1,KM
        DO 30 J = 1,JM
          U(1,IMAX,J,K) = U(1,1,J,K)
          U(2,IMAX,J,K) = U(2,1,J,K)
          U(3,IMAX,J,K) = U(3,1,J,K)
          U(4,IMAX,J,K) = U(4,1,J,K)
          U(5,IMAX,J,K) = U(5,1,J,K)
          P(IMAX,J,K) = P(1,J,K)
   30   CONTINUE
   40 CONTINUE
C
C---  first, initialize the residual field
C
      DO 70 K = 1,KMAX
        DO 60 J = 1,JMAX
          DO 50 I = 1,IMAX
            R(1,I,J,K) = 0.
            R(2,I,J,K) = 0.
            R(3,I,J,K) = 0.
            R(4,I,J,K) = 0.
            R(5,I,J,K) = 0.
   50     CONTINUE
   60   CONTINUE
   70 CONTINUE
C
C---  determine the residual at the symmetry plane cells
C
      DO 90 J = 1,JM
        DO 80 I = 1,IM
```

```
          R(3,I,J,1) = WALL(I,J)*PRESYM(I,J)
   80   CONTINUE
   90 CONTINUE
C
C---  add contribution due to pressures at the wing surface
C
      DO 110 K = 1,KM
        DO 100 I = 1,IM
          R(2,I,1,K) = R(2,I,1,K) + FACEX(2,I,0,K)*PRESWG(I,K)
          R(3,I,1,K) = R(3,I,1,K) + FACEY(2,I,0,K)*PRESWG(I,K)
          R(4,I,1,K) = R(4,I,1,K) + FACEZ(2,I,0,K)*PRESWG(I,K)
  100   CONTINUE
  110 CONTINUE
C
      DO 160 K = 1,KM
        DO 150 J = 1,JM
C
C---  first do the flux across the i-face
C
          DO 120 I = 1,IM
C
C---  find the flux vector for cells (i,j,k) and (i+1,j,k)
C
            QA = (U(2,I,J,K)*FACEX(1,I,J,K) + U(3,I,J,K)*FACEY(1,I,J,K)
     &       + U(4,I,J,K)*FACEZ(1,I,J,K))/U(1,I,J,K)
            H1 = U(1,I,J,K)*QA
            H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEX(1,I,J,K)
            H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEY(1,I,J,K)
            H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEZ(1,I,J,K)
            H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
            QAP = (U(2,I+1,J,K)*FACEX(1,I,J,K) + U(3,I+1,J,K)*
     &       FACEY(1,I,J,K) + U(4,I+1,J,K)*FACEZ(1,I,J,K))/U(1,I+1,J,K)
            H1P = U(1,I+1,J,K)*QAP
            H2P = U(2,I+1,J,K)*QAP + P(I+1,J,K)*FACEX(1,I,J,K)
            H3P = U(3,I+1,J,K)*QAP + P(I+1,J,K)*FACEY(1,I,J,K)
            H4P = U(4,I+1,J,K)*QAP + P(I+1,J,K)*FACEZ(1,I,J,K)
            H5P = (U(5,I+1,J,K) + P(I+1,J,K))*QAP
C
C---  find the average flux vector between cells (i,j,k) and (i+1,j,k)
C
            DR1 = .5*(H1 + H1P)
            DR2 = .5*(H2 + H2P)
            DR3 = .5*(H3 + H3P)
            DR4 = .5*(H4 + H4P)
            DR5 = .5*(H5 + H5P)
C
C---  subtract outgoing flux from cell (i,j,k) and add it to
C     cell (i+1,j,k)
C
            R(1,I+1,J,K) = R(1,I+1,J,K) + DR1
            R(2,I+1,J,K) = R(2,I+1,J,K) + DR2
            R(3,I+1,J,K) = R(3,I+1,J,K) + DR3
            R(4,I+1,J,K) = R(4,I+1,J,K) + DR4
            R(5,I+1,J,K) = R(5,I+1,J,K) + DR5
C
            R(1,I,J,K) = R(1,I,J,K) - DR1
            R(2,I,J,K) = R(2,I,J,K) - DR2
            R(3,I,J,K) = R(3,I,J,K) - DR3
            R(4,I,J,K) = R(4,I,J,K) - DR4
            R(5,I,J,K) = R(5,I,J,K) - DR5
```

```
C
  120     CONTINUE
C
C---  next do the flux across the j-face
C
          DO 130 I = 1,IM
C
C---  find the flux vector for cells (i,j,k) and (i,j+1,k)
C
             QA = (U(2,I,J,K)*FACEX(2,I,J,K) + U(3,I,J,K)*FACEY(2,I,J,K)
     &        + U(4,I,J,K)*FACEZ(2,I,J,K))/U(1,I,J,K)
             H1 = U(1,I,J,K)*QA
             H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEX(2,I,J,K)
             H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEY(2,I,J,K)
             H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEZ(2,I,J,K)
             H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
             QAP = (U(2,I,J+1,K)*FACEX(2,I,J,K) + U(3,I,J+1,K)*
     &        FACEY(2,I,J,K) + U(4,I,J+1,K)*FACEZ(2,I,J,K))/U(1,I,J+1,K)
             H1P = U(1,I,J+1,K)*QAP
             H2P = U(2,I,J+1,K)*QAP + P(I,J+1,K)*FACEX(2,I,J,K)
             H3P = U(3,I,J+1,K)*QAP + P(I,J+1,K)*FACEY(2,I,J,K)
             H4P = U(4,I,J+1,K)*QAP + P(I,J+1,K)*FACEZ(2,I,J,K)
             H5P = (U(5,I,J+1,K) + P(I,J+1,K))*QAP
C
C---  find the average flux vector between cells (i,j,k) and (i,j+1,k)
C
             DR1 = .5*(H1 + H1P)
             DR2 = .5*(H2 + H2P)
             DR3 = .5*(H3 + H3P)
             DR4 = .5*(H4 + H4P)
             DR5 = .5*(H5 + H5P)
C
C---  subtract outgoing flux from cell (i,j,k) and add it to
C     cell (i,j,k+1)
C
             R(1,I,J,K) = R(1,I,J,K) - DR1
             R(2,I,J,K) = R(2,I,J,K) - DR2
             R(3,I,J,K) = R(3,I,J,K) - DR3
             R(4,I,J,K) = R(4,I,J,K) - DR4
             R(5,I,J,K) = R(5,I,J,K) - DR5
C
             R(1,I,J+1,K) = R(1,I,J+1,K) + DR1
             R(2,I,J+1,K) = R(2,I,J+1,K) + DR2
             R(3,I,J+1,K) = R(3,I,J+1,K) + DR3
             R(4,I,J+1,K) = R(4,I,J+1,K) + DR4
             R(5,I,J+1,K) = R(5,I,J+1,K) + DR5
C
  130     CONTINUE
C
C---  finally do the flux across the k-face
C
          DO 140 I = 1,IM
C
C---  find the flux vector for cells (i,j,k) and (i,j,k+1)
C
             QA = (U(2,I,J,K)*FACEX(3,I,J,K) + U(3,I,J,K)*FACEY(3,I,J,K)
     &        + U(4,I,J,K)*FACEZ(3,I,J,K))/U(1,I,J,K)
             H1 = U(1,I,J,K)*QA
             H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEX(3,I,J,K)
             H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEY(3,I,J,K)
```

```
            H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEZ(3,I,J,K)
            H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
            QAP = (U(2,I,J,K+1)*FACEX(3,I,J,K) + U(3,I,J,K+1)*
     &       FACEY(3,I,J,K) + U(4,I,J,K+1)*FACEZ(3,I,J,K))/U(1,I,J,K+1)
            H1P = U(1,I,J,K+1)*QAP
            H2P = U(2,I,J,K+1)*QAP + P(I,J,K+1)*FACEX(3,I,J,K)
            H3P = U(3,I,J,K+1)*QAP + P(I,J,K+1)*FACEY(3,I,J,K)
            H4P = U(4,I,J,K+1)*QAP + P(I,J,K+1)*FACEZ(3,I,J,K)
            H5P = (U(5,I,J,K+1) + P(I,J,K+1))*QAP
C
C---  find the average flux vector between cells (i,j,k) and (i,j,k+1)
C
            DR1 = .5*(H1 + H1P)
            DR2 = .5*(H2 + H2P)
            DR3 = .5*(H3 + H3P)
            DR4 = .5*(H4 + H4P)
            DR5 = .5*(H5 + H5P)
C
C---  subtract outgoing flux from cell (i,j,k) and add it to
C     cell (i,j,k+1)
C
            R(1,I,J,K) = R(1,I,J,K) - DR1
            R(2,I,J,K) = R(2,I,J,K) - DR2
            R(3,I,J,K) = R(3,I,J,K) - DR3
            R(4,I,J,K) = R(4,I,J,K) - DR4
            R(5,I,J,K) = R(5,I,J,K) - DR5
C
            R(1,I,J,K+1) = R(1,I,J,K+1) + DR1
            R(2,I,J,K+1) = R(2,I,J,K+1) + DR2
            R(3,I,J,K+1) = R(3,I,J,K+1) + DR3
            R(4,I,J,K+1) = R(4,I,J,K+1) + DR4
            R(5,I,J,K+1) = R(5,I,J,K+1) + DR5
C
  140     CONTINUE
  150   CONTINUE
  160 CONTINUE
C
C---  now fix-up the residuals across the coordinate cut in the
C     i-direction (chordwise)
C
      DO 180 K = 1,KM
        DO 170 J = 1,JM
          R(1,1,J,K) = R(1,1,J,K) + R(1,IMAX,J,K)
          R(2,1,J,K) = R(2,1,J,K) + R(2,IMAX,J,K)
          R(3,1,J,K) = R(3,1,J,K) + R(3,IMAX,J,K)
          R(4,1,J,K) = R(4,1,J,K) + R(4,IMAX,J,K)
          R(5,1,J,K) = R(5,1,J,K) + R(5,IMAX,J,K)
  170   CONTINUE
  180 CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE DISSIP:  calculates second- and fourth-order
C  dissipation terms in Jameson's explicit, multi-stage Euler
C  scheme
C
      SUBROUTINE DISSIP
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/VAR / U(5,-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             P(-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             R(5,ISIZ+1,JSIZ+1,KSIZ+1),NCELLS
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      COMMON/BOUN/ PRESYM(ISIZ,JSIZ),PRESWG(ISIZ,KSIZ),WALL(ISIZ,JSIZ)
      COMMON/DISP/ D(5,ISIZ,JSIZ,KSIZ),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH,CIN
      COMMON/TIME/ DT(0:ISIZ+1,0:JSIZ+1,0:KSIZ+1),ALPHA1,ALPHA2,ALPHA3
      COMMON/BOU2/ WUN(3,0:ISIZ+1,0:KSIZ+1)
      DIMENSION EP2(0:ISIZ), EP4(0:ISIZ)
      REAL KAP2, KAP4, MACH, NUM, NUP, NUMM, NUPM, MM, MP
C
C---  The value for the dissipation is found from the operator
C
C                      D  + D  + D  = D(u)
C                       X    Y    Z       ijk
C
C     as defined by Jameson, yielding second- and fourth-order
C     differences.  The operator in the X,Y,Z (I,J,K) coordinate
C     directions are determined in order.  Eriksson's treatment of the
C     smoothing at the solid wall and far-field boundaries is used.
C
C---  set-up dummy points in the i-, j-, and k-directions.
C
      DO 20 K = 1,KM
        DO 10 I = 1,IM
          U(1,I,JMAX+1,K) = 3.*U(1,I,JM,K) - 2.*U(1,I,JM-1,K)
          U(2,I,JMAX+1,K) = 3.*U(2,I,JM,K) - 2.*U(2,I,JM-1,K)
          U(3,I,JMAX+1,K) = 3.*U(3,I,JM,K) - 2.*U(3,I,JM-1,K)
          U(4,I,JMAX+1,K) = 3.*U(4,I,JM,K) - 2.*U(4,I,JM-1,K)
          U(5,I,JMAX+1,K) = 3.*U(5,I,JM,K) - 2.*U(5,I,JM-1,K)
          P(I,JMAX+1,K)   =   3.*P(I,JM,K) -   2.*P(I,JM-1,K)
C
          U(1,I,JMAX,K) = 2.*U(1,I,JM,K) - U(1,I,JM-1,K)
          U(2,I,JMAX,K) = 2.*U(2,I,JM,K) - U(2,I,JM-1,K)
          U(3,I,JMAX,K) = 2.*U(3,I,JM,K) - U(3,I,JM-1,K)
          U(4,I,JMAX,K) = 2.*U(4,I,JM,K) - U(4,I,JM-1,K)
          U(5,I,JMAX,K) = 2.*U(5,I,JM,K) - U(5,I,JM-1,K)
          P(I,JMAX,K)   =   2.*P(I,JM,K) -   P(I,JM-1,K)
C
          U(1,I,0,K) = 2.*U(1,I,1,K) - U(1,I,2,K)
          U(2,I,0,K) = 2.*U(2,I,1,K) - U(2,I,2,K)
          U(3,I,0,K) = 2.*U(3,I,1,K) - U(3,I,2,K)
          U(4,I,0,K) = 2.*U(4,I,1,K) - U(4,I,2,K)
          U(5,I,0,K) = 2.*U(5,I,1,K) - U(5,I,2,K)
          P(I,0,K)   =   2.*P(I,1,K) -   P(I,2,K)
C
          U(1,I,-1,K) = 3.*U(1,I,1,K) - 2.*U(1,I,2,K)
          U(2,I,-1,K) = 3.*U(2,I,1,K) - 2.*U(2,I,2,K)
          U(3,I,-1,K) = 3.*U(3,I,1,K) - 2.*U(3,I,2,K)
          U(4,I,-1,K) = 3.*U(4,I,1,K) - 2.*U(4,I,2,K)
          U(5,I,-1,K) = 3.*U(5,I,1,K) - 2.*U(5,I,2,K)
```

```
          P(I,-1,K)    =   3.*P(I,1,K) -   2.*P(I,2,K)
C
          DT(I,JMAX,K) = DT(I,JM,K)
          DT(I,0,K) = DT(I,1,K)
   10   CONTINUE
   20 CONTINUE
C
      DO 40 J = 1,JM
        DO 30 I = 1,IM
          U(1,I,J,KMAX) = U(1,IMAX-I,J,KMAX-1)
          U(2,I,J,KMAX) = U(2,IMAX-I,J,KMAX-1)
          U(3,I,J,KMAX) = U(3,IMAX-I,J,KMAX-1)
          U(4,I,J,KMAX) = U(4,IMAX-I,J,KMAX-1)
          U(5,I,J,KMAX) = U(5,IMAX-I,J,KMAX-1)
          P(I,J,KMAX)   =   P(IMAX-I,J,KMAX-1)
C
          U(1,I,J,KMAX+1) = U(1,IMAX-I,J,KMAX-2)
          U(2,I,J,KMAX+1) = U(2,IMAX-I,J,KMAX-2)
          U(3,I,J,KMAX+1) = U(3,IMAX-I,J,KMAX-2)
          U(4,I,J,KMAX+1) = U(4,IMAX-I,J,KMAX-2)
          U(5,I,J,KMAX+1) = U(5,IMAX-I,J,KMAX-2)
          P(I,J,KMAX+1)   =   P(IMAX-I,J,KMAX-2)
C
          U(1,I,J,0) =  U(1,I,J,1)
          U(2,I,J,0) =  U(2,I,J,1)
          U(3,I,J,0) = -U(3,I,J,1)
          U(4,I,J,0) =  U(4,I,J,1)
          U(5,I,J,0) =  U(5,I,J,1)
          P(I,J,0)   =    P(I,J,1)
C
          U(1,I,J,-1) =  U(1,I,J,2)
          U(2,I,J,-1) =  U(2,I,J,2)
          U(3,I,J,-1) = -U(3,I,J,2)
          U(4,I,J,-1) =  U(4,I,J,2)
          U(5,I,J,-1) =  U(5,I,J,2)
          P(I,J,-1)   =    P(I,J,2)
C
          DT(I,J,0) = DT(I,J,1)
          DT(I,J,KMAX) = DT(IMAX-I,J,KM)
   30   CONTINUE
   40 CONTINUE
C
      DO 60 K = 1,KM
        DO 50 J = 1,JM
          U(1,IMAX+1,J,K) = U(1,2,J,K)
          U(2,IMAX+1,J,K) = U(2,2,J,K)
          U(3,IMAX+1,J,K) = U(3,2,J,K)
          U(4,IMAX+1,J,K) = U(4,2,J,K)
          U(5,IMAX+1,J,K) = U(5,2,J,K)
          P(IMAX+1,J,K)   =   P(2,J,K)
C
          U(1,IMAX,J,K) = U(1,1,J,K)
          U(2,IMAX,J,K) = U(2,1,J,K)
          U(3,IMAX,J,K) = U(3,1,J,K)
          U(4,IMAX,J,K) = U(4,1,J,K)
          U(5,IMAX,J,K) = U(5,1,J,K)
          P(IMAX,J,K)   =   P(1,J,K)
C
          U(1,0,J,K) = U(1,IM,J,K)
          U(2,0,J,K) = U(2,IM,J,K)
          U(3,0,J,K) = U(3,IM,J,K)
```

```
          U(4,0,J,K) = U(4,IM,J,K)
          U(5,0,J,K) = U(5,IM,J,K)
          P(0,J,K)   =   P(IM,J,K)
C
          U(1,-1,J,K) = U(1,IM-1,J,K)
          U(2,-1,J,K) = U(2,IM-1,J,K)
          U(3,-1,J,K) = U(3,IM-1,J,K)
          U(4,-1,J,K) = U(4,IM-1,J,K)
          U(5,-1,J,K) = U(5,IM-1,J,K)
          P(-1,J,K)   =   P(IM-1,J,K)
C
          DT(0,J,K) = DT(IM,J,K)
          DT(IMAX,J,K) = DT(1,J,K)
   50   CONTINUE
   60 CONTINUE
      DO 110 K = 1,KM
        DO 100 J = 1,JM
C
C---  begin with the i-direction:  find the values of the
C     dissipation coefficients
C
          DO 70 I = 0,IM
            NUM = ABS((P(I+1,J,K) - 2.*P(I,J,K) + P(I-1,J,K))/
     &                (P(I+1,J,K) + 2.*P(I,J,K) + P(I-1,J,K)))
            NUP = ABS((P(I+2,J,K) - 2.*P(I+1,J,K) + P(I,J,K))/
     &                (P(I+2,J,K) + 2.*P(I+1,J,K) + P(I,J,K)))
            DTH = .5*CFL*(1./DT(I+1,J,K) + 1./DT(I,J,K))
C
            EP = KAP2 * MAX(NUM,NUP)
            EP2(I) = EP*DTH
            EP4(I) = MAX(0.,(KAP4 - EP))*DTH
   70     CONTINUE
C
C---  calculate dissipation d(i,j,k)
C
          DO 80 I = 1,IM
            DS1 = EP2(I)*(U(1,I+1,J,K) - U(1,I,J,K))
            DF1 = EP4(I)*(U(1,I+2,J,K) - 3.*(U(1,I+1,J,K) -
     &              U(1,I,J,K)) - U(1,I-1,J,K))
            DS2 = EP2(I)*(U(2,I+1,J,K) - U(2,I,J,K))
            DF2 = EP4(I)*(U(2,I+2,J,K) - 3.*(U(2,I+1,J,K) -
     &              U(2,I,J,K)) - U(2,I-1,J,K))
            DS3 = EP2(I)*(U(3,I+1,J,K) - U(3,I,J,K))
            DF3 = EP4(I)*(U(3,I+2,J,K) - 3.*(U(3,I+1,J,K) -
     &              U(3,I,J,K)) - U(3,I-1,J,K))
            DS4 = EP2(I)*(U(4,I+1,J,K) - U(4,I,J,K))
            DF4 = EP4(I)*(U(4,I+2,J,K) - 3.*(U(4,I+1,J,K) -
     &              U(4,I,J,K)) - U(4,I-1,J,K))
            DS5 = EP2(I)*(U(5,I+1,J,K) - U(5,I,J,K) + P(I+1,J,K) -
     &              P(I,J,K))
            DF5 = EP4(I)*(U(5,I+2,J,K) - 3.*(U(5,I+1,J,K) -
     &              U(5,I,J,K)) - U(5,I-1,J,K) + P(I+2,J,K) - 3.*
     &              (P(I+1,J,K) - P(I,J,K)) - P(I-1,J,K))
C
            D(1,I,J,K) = DS1 - DF1
            D(2,I,J,K) = DS2 - DF2
            D(3,I,J,K) = DS3 - DF3
            D(4,I,J,K) = DS4 - DF4
            D(5,I,J,K) = DS5 - DF5
   80     CONTINUE
C
```

```
          DO 90 I = 1,IM
            DS1M = EP2(I-1)*(U(1,I,J,K) - U(1,I-1,J,K))
            DF1M = EP4(I-1)*(U(1,I+1,J,K) - 3.*(U(1,I,J,K) -
     &             U(1,I-1,J,K)) - U(1,I-2,J,K))
            DS2M = EP2(I-1)*(U(2,I,J,K) - U(2,I-1,J,K))
            DF2M = EP4(I-1)*(U(2,I+1,J,K) - 3.*(U(2,I,J,K) -
     &             U(2,I-1,J,K)) - U(2,I-2,J,K))
            DS3M = EP2(I-1)*(U(3,I,J,K) - U(3,I-1,J,K))
            DF3M = EP4(I-1)*(U(3,I+1,J,K) - 3.*(U(3,I,J,K) -
     &             U(3,I-1,J,K)) - U(3,I-2,J,K))
            DS4M = EP2(I-1)*(U(4,I,J,K) - U(4,I-1,J,K))
            DF4M = EP4(I-1)*(U(4,I+1,J,K) - 3.*(U(4,I,J,K) -
     &             U(4,I-1,J,K)) - U(4,I-2,J,K))
            DS5M = EP2(I-1)*(U(5,I,J,K) - U(5,I-1,J,K) + P(I,J,K) -
     &             P(I-1,J,K))
            DF5M = EP4(I-1)*(U(5,I+1,J,K) - 3.*(U(5,I,J,K) -
     &             U(5,I-1,J,K)) - U(5,I-2,J,K) + P(I+1,J,K) - 3.*
     &             (P(I,J,K) - P(I-1,J,K)) - P(I-2,J,K))
C
            D(1,I,J,K) = D(1,I,J,K) - DS1M + DF1M
            D(2,I,J,K) = D(2,I,J,K) - DS2M + DF2M
            D(3,I,J,K) = D(3,I,J,K) - DS3M + DF3M
            D(4,I,J,K) = D(4,I,J,K) - DS4M + DF4M
            D(5,I,J,K) = D(5,I,J,K) - DS5M + DF5M
   90     CONTINUE
  100   CONTINUE
  110 CONTINUE
      DO 170 K = 1,KM
        DO 160 J = 1,JM
C
C---  next do the j-direction:  find the values of the
C     dissipation coefficients
C
          DO 120 I = 1,IM
            NUM = ABS((P(I,J+1,K) - 2.*P(I,J,K) + P(I,J-1,K))/
     &                (P(I,J+1,K) + 2.*P(I,J,K) + P(I,J-1,K)))
            NUP = ABS((P(I,J+2,K) - 2.*P(I,J+1,K) + P(I,J,K))/
     &                (P(I,J+2,K) + 2.*P(I,J+1,K) + P(I,J,K)))
            DTH = .5*CFL*(1./DT(I,J+1,K) + 1./DT(I,J,K))
C
            EPP = KAP2 * MAX(NUM,NUP)
            EP2(I) = EPP*DTH
            EP4(I) = MAX(0.,(KAP4 - EPP))*DTH
  120     CONTINUE
C
C---  calculate dissipation d(i,j,k)
C
          DO 130 I = 1,IM
            DS1 = EP2(I)*(U(1,I,J+1,K) - U(1,I,J,K))
            DF1 = EP4(I)*(U(1,I,J+2,K) - 3.*(U(1,I,J+1,K) -
     &            U(1,I,J,K)) - U(1,I,J-1,K))
            DS2 = EP2(I)*(U(2,I,J+1,K) - U(2,I,J,K))
            DF2 = EP4(I)*(U(2,I,J+2,K) - 3.*(U(2,I,J+1,K) -
     &            U(2,I,J,K)) - U(2,I,J-1,K))
            DS3 = EP2(I)*(U(3,I,J+1,K) - U(3,I,J,K))
            DF3 = EP4(I)*(U(3,I,J+2,K) - 3.*(U(3,I,J+1,K) -
     &            U(3,I,J,K)) - U(3,I,J-1,K))
            DS4 = EP2(I)*(U(4,I,J+1,K) - U(4,I,J,K))
            DF4 = EP4(I)*(U(4,I,J+2,K) - 3.*(U(4,I,J+1,K) -
     &            U(4,I,J,K)) - U(4,I,J-1,K))
            DS5 = EP2(I)*(U(5,I,J+1,K) - U(5,I,J,K) + P(I,J+1,K) -
```

```
     &            P(I,J,K))
            DF5 = EP4(I)*(U(5,I,J+2,K) - 3.*(U(5,I,J+1,K) -
     &            U(5,I,J,K)) - U(5,I,J-1,K) + P(I,J+2,K) - 3.*
     &            (P(I,J+1,K) - P(I,J,K)) - P(I,J-1,K))
C
            D(1,I,J,K) = D(1,I,J,K) + DS1 - DF1
            D(2,I,J,K) = D(2,I,J,K) + DS2 - DF2
            D(3,I,J,K) = D(3,I,J,K) + DS3 - DF3
            D(4,I,J,K) = D(4,I,J,K) + DS4 - DF4
            D(5,I,J,K) = D(5,I,J,K) + DS5 - DF5
  130     CONTINUE
C
          DO 140 I = 1,IM
            NUMM = ABS((P(I,J,K) - 2.*P(I,J-1,K) + P(I,J-2,K))/
     &                 (P(I,J,K) + 2.*P(I,J-1,K) + P(I,J-2,K)))
            NUPM = ABS((P(I,J+1,K) - 2.*P(I,J,K) + P(I,J-1,K))/
     &                 (P(I,J+1,K) + 2.*P(I,J,K) + P(I,J-1,K)))
            DTH = .5*CFL*(1./DT(I,J,K) + 1./DT(I,J-1,K))
C
            EPM = KAP2 * MAX(NUMM,NUPM)
            EP2(I) = EPM*DTH
            EP4(I) = MAX(0.,(KAP4 - EPM))*DTH
  140     CONTINUE
C
          DO 150 I = 1,IM
            DS1M = EP2(I)*(U(1,I,J,K) - U(1,I,J-1,K))
            DF1M = EP4(I)*(U(1,I,J+1,K) - 3.*(U(1,I,J,K) -
     &             U(1,I,J-1,K)) - U(1,I,J-2,K))
            DS2M = EP2(I)*(U(2,I,J,K) - U(2,I,J-1,K))
            DF2M = EP4(I)*(U(2,I,J+1,K) - 3.*(U(2,I,J,K) -
     &             U(2,I,J-1,K)) - U(2,I,J-2,K))
            DS3M = EP2(I)*(U(3,I,J,K) - U(3,I,J-1,K))
            DF3M = EP4(I)*(U(3,I,J+1,K) - 3.*(U(3,I,J,K) -
     &             U(3,I,J-1,K)) - U(3,I,J-2,K))
            DS4M = EP2(I)*(U(4,I,J,K) - U(4,I,J-1,K))
            DF4M = EP4(I)*(U(4,I,J+1,K) - 3.*(U(4,I,J,K) -
     &             U(4,I,J-1,K)) - U(4,I,J-2,K))
            DS5M = EP2(I)*(U(5,I,J,K) - U(5,I,J-1,K) + P(I,J,K) -
     &             P(I,J-1,K))
            DF5M = EP4(I)*(U(5,I,J+1,K) - 3.*(U(5,I,J,K) -
     &             U(5,I,J-1,K)) - U(5,I,J-2,K) + P(I,J+1,K) - 3.*
     &             (P(I,J,K) - P(I,J-1,K)) - P(I,J-2,K))
C
            D(1,I,J,K) = D(1,I,J,K) - DS1M + DF1M
            D(2,I,J,K) = D(2,I,J,K) - DS2M + DF2M
            D(3,I,J,K) = D(3,I,J,K) - DS3M + DF3M
            D(4,I,J,K) = D(4,I,J,K) - DS4M + DF4M
            D(5,I,J,K) = D(5,I,J,K) - DS5M + DF5M
  150     CONTINUE
  160   CONTINUE
  170 CONTINUE
C
      DO 230 K = 1,KM
        DO 220 J = 1,JM
C
C---  next do the k-direction:  find the values of the
C     dissipation coefficients
C
          DO 180 I = 1,IM
            NUM = ABS((P(I,J,K+1) - 2.*P(I,J,K) + P(I,J,K-1))/
     &                (P(I,J,K+1) + 2.*P(I,J,K) + P(I,J,K-1)))
```

```
            NUP = ABS((P(I,J,K+2) - 2.*P(I,J,K+1) + P(I,J,K))/
     &                (P(I,J,K+2) + 2.*P(I,J,K+1) + P(I,J,K)))
            DTH = .5*CFL*(1./DT(I,J,K+1) + 1./DT(I,J,K))
C
            EPP = KAP2 * MAX(NUM,NUP)
            EP2(I) = EPP*DTH
            EP4(I) = MAX(0.,(KAP4 - EPP))*DTH
  180     CONTINUE
C
C---  calculate dissipation d(i,j,k)
C
          DO 190 I = 1,IM
            DS1 = EP2(I)*(U(1,I,J,K+1) - U(1,I,J,K))
            DF1 = EP4(I)*(U(1,I,J,K+2) - 3.*(U(1,I,J,K+1) -
     &            U(1,I,J,K)) - U(1,I,J,K-1))
            DS2 = EP2(I)*(U(2,I,J,K+1) - U(2,I,J,K))
            DF2 = EP4(I)*(U(2,I,J,K+2) - 3.*(U(2,I,J,K+1) -
     &            U(2,I,J,K)) - U(2,I,J,K-1))
            DS3 = EP2(I)*(U(3,I,J,K+1) - U(3,I,J,K))
            DF3 = EP4(I)*(U(3,I,J,K+2) - 3.*(U(3,I,J,K+1) -
     &            U(3,I,J,K)) - U(3,I,J,K-1))
            DS4 = EP2(I)*(U(4,I,J,K+1) - U(4,I,J,K))
            DF4 = EP4(I)*(U(4,I,J,K+2) - 3.*(U(4,I,J,K+1) -
     &            U(4,I,J,K)) - U(4,I,J,K-1))
            DS5 = EP2(I)*(U(5,I,J,K+1) - U(5,I,J,K) + P(I,J,K+1) -
     &            P(I,J,K))
            DF5 = EP4(I)*(U(5,I,J,K+2) - 3.*(U(5,I,J,K+1) -
     &            U(5,I,J,K)) - U(5,I,J,K-1) + P(I,J,K+2) - 3.*
     &            (P(I,J,K+1) - P(I,J,K)) - P(I,J,K-1))
C
            D(1,I,J,K) = D(1,I,J,K) + DS1 - DF1
            D(2,I,J,K) = D(2,I,J,K) + DS2 - DF2
            D(3,I,J,K) = D(3,I,J,K) + DS3 - DF3
            D(4,I,J,K) = D(4,I,J,K) + DS4 - DF4
            D(5,I,J,K) = D(5,I,J,K) + DS5 - DF5
  190     CONTINUE
C
          DO 200 I = 1,IM
            NUMM = ABS((P(I,J,K) - 2.*P(I,J,K-1) + P(I,J,K-2))/
     &                (P(I,J,K) + 2.*P(I,J,K-1) + P(I,J,K-2)))
            NUPM = ABS((P(I,J,K+1) - 2.*P(I,J,K) + P(I,J,K-1))/
     &                (P(I,J,K+1) + 2.*P(I,J,K) + P(I,J,K-1)))
            DTH = .5*CFL*(1./DT(I,J,K) + 1./DT(I,J,K-1))
C
            EPM = KAP2 * MAX(NUMM,NUPM)
            EP2(I) = EPM*DTH
            EP4(I) = MAX(0.,(KAP4 - EPM))*DTH
  200     CONTINUE
C
          DO 210 I = 1,IM
            DS1M = EP2(I)*(U(1,I,J,K) - U(1,I,J,K-1))
            DF1M = EP4(I)*(U(1,I,J,K+1) - 3.*(U(1,I,J,K) -
     &            U(1,I,J,K-1)) - U(1,I,J,K-2))
            DS2M = EP2(I)*(U(2,I,J,K) - U(2,I,J,K-1))
            DF2M = EP4(I)*(U(2,I,J,K+1) - 3.*(U(2,I,J,K) -
     &            U(2,I,J,K-1)) - U(2,I,J,K-2))
            DS3M = EP2(I)*(U(3,I,J,K) - U(3,I,J,K-1))
            DF3M = EP4(I)*(U(3,I,J,K+1) - 3.*(U(3,I,J,K) -
     &            U(3,I,J,K-1)) - U(3,I,J,K-2))
            DS4M = EP2(I)*(U(4,I,J,K) - U(4,I,J,K-1))
            DF4M = EP4(I)*(U(4,I,J,K+1) - 3.*(U(4,I,J,K) -
```

```
     &             U(4,I,J,K-1)) - U(4,I,J,K-2))
           DS5M = EP2(I)*(U(5,I,J,K) - U(5,I,J,K-1) + P(I,J,K) -
     &             P(I,J,K-1))
           DF5M = EP4(I)*(U(5,I,J,K+1) - 3.*(U(5,I,J,K) -
     &             U(5,I,J,K-1)) - U(5,I,J,K-2) + P(I,J,K+1) - 3.*
     &             (P(I,J,K) - P(I,J,K-1)) - P(I,J,K-2))
C
           D(1,I,J,K) = D(1,I,J,K) - DS1M + DF1M
           D(2,I,J,K) = D(2,I,J,K) - DS2M + DF2M
           D(3,I,J,K) = D(3,I,J,K) - DS3M + DF3M
           D(4,I,J,K) = D(4,I,J,K) - DS4M + DF4M
           D(5,I,J,K) = D(5,I,J,K) - DS5M + DF5M
  210     CONTINUE
  220   CONTINUE
  230 CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE DTSIZE:  calculates the value of the local time step
C  size based on the local CFL number restriction
C
      SUBROUTINE DTSIZE
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/VAR / U(5,-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             P(-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             R(5,ISIZ+1,JSIZ+1,KSIZ+1),NCELLS
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      COMMON/CELL/ FACEX(3,ISIZ,0:JSIZ,KSIZ),FACEY(3,ISIZ,0:JSIZ,KSIZ),
     &             FACEZ(3,ISIZ,0:JSIZ,KSIZ),VOLUME(ISIZ,0:JSIZ,KSIZ)
      COMMON/DISP/ D(5,ISIZ,JSIZ,KSIZ),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH,CIN
      COMMON/TIME/ DT(0:ISIZ+1,0:JSIZ+1,0:KSIZ+1),ALPHA1,ALPHA2,ALPHA3
      COMMON/AREA/ AXM(ISIZ,JSIZ,KSIZ),AYM(ISIZ,JSIZ,KSIZ),
     &             AZM(ISIZ,JSIZ,KSIZ)
      REAL KAP2, KAP4, MACH
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            RHO = U(1,I,J,K)
            U1  = U(2,I,J,K)/RHO
            V1  = U(3,I,J,K)/RHO
            W1  = U(4,I,J,K)/RHO
            C = SQRT(GAM*P(I,J,K)/RHO)
C
            AX = AXM(I,J,K)
            AY = AYM(I,J,K)
            AZ = AZM(I,J,K)
            AR = SQRT(AX**2 + AY**2 + AZ**2)
C
            UA = ABS(U1*AX)
            VA = ABS(V1*AY)
            WA = ABS(W1*AZ)
            CA = C*AR
            DT(I,J,K) = CFL/(UA + VA + WA + CA)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE TIMSTP:  advances solution Euler equations by a four-
C  stage time stepping scheme, using a spacially varying time step
C  and frozen dissipative terms in order to speed convergence
C
      SUBROUTINE TIMSTP
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/VAR / U(5,-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             P(-1:ISIZ+2,-1:JSIZ+2,-1:KSIZ+2),
     &             R(5,ISIZ+1,JSIZ+1,KSIZ+1),NCELLS
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      COMMON/CELL/ FACEX(3,ISIZ,0:JSIZ,KSIZ),FACEY(3,ISIZ,0:JSIZ,KSIZ),
     &             FACEZ(3,ISIZ,0:JSIZ,KSIZ),VOLUME(ISIZ,0:JSIZ,KSIZ)
      COMMON/DISP/ D(5,ISIZ,JSIZ,KSIZ),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH,CIN
      COMMON/TIME/ DT(0:ISIZ+1,0:JSIZ+1,0:KSIZ+1),ALPHA1,ALPHA2,ALPHA3
      DIMENSION UTEMP(5,ISIZ,JSIZ,KSIZ)
      REAL KAP2, KAP4, MACH
C
C---  store current state vector in temporary array UTEMP
C
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            UTEMP(1,I,J,K) = U(1,I,J,K)
            UTEMP(2,I,J,K) = U(2,I,J,K)
            UTEMP(3,I,J,K) = U(3,I,J,K)
            UTEMP(4,I,J,K) = U(4,I,J,K)
            UTEMP(5,I,J,K) = U(5,I,J,K)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C
C--- start four-stage time stepping procedure
C
      DO 60 K = 1,KM
        DO 50 J = 1,JM
          DO 40 I = 1,IM
            DTLOC = ALPHA1*DT(I,J,K)
            U(1,I,J,K) = UTEMP(1,I,J,K) + DTLOC*(R(1,I,J,K) +
     &       D(1,I,J,K))
            U(2,I,J,K) = UTEMP(2,I,J,K) + DTLOC*(R(2,I,J,K) +
     &       D(2,I,J,K))
            U(3,I,J,K) = UTEMP(3,I,J,K) + DTLOC*(R(3,I,J,K) +
     &       D(3,I,J,K))
            U(4,I,J,K) = UTEMP(4,I,J,K) + DTLOC*(R(4,I,J,K) +
     &       D(4,I,J,K))
            U(5,I,J,K) = UTEMP(5,I,J,K) + DTLOC*(R(5,I,J,K) +
     &       D(5,I,J,K))
   40     CONTINUE
   50   CONTINUE
   60 CONTINUE
      CALL BNDRYC
      CALL FLUX
C
C---  second stage
C
      DO 90 K = 1,KM
        DO 80 J = 1,JM
```

```
          DO 70 I = 1,IM
            DTLOC = ALPHA2*DT(I,J,K)
            U(1,I,J,K) = UTEMP(1,I,J,K) + DTLOC*(R(1,I,J,K) +
     &       D(1,I,J,K))
            U(2,I,J,K) = UTEMP(2,I,J,K) + DTLOC*(R(2,I,J,K) +
     &       D(2,I,J,K))
            U(3,I,J,K) = UTEMP(3,I,J,K) + DTLOC*(R(3,I,J,K) +
     &       D(3,I,J,K))
            U(4,I,J,K) = UTEMP(4,I,J,K) + DTLOC*(R(4,I,J,K) +
     &       D(4,I,J,K))
            U(5,I,J,K) = UTEMP(5,I,J,K) + DTLOC*(R(5,I,J,K) +
     &       D(5,I,J,K))
   70     CONTINUE
   80   CONTINUE
   90 CONTINUE
      CALL BNDRYC
      CALL FLUX
C
C--- third stage
C
      DO 120 K = 1,KM
        DO 110 J = 1,JM
          DO 100 I = 1,IM
            DTLOC = ALPHA3*DT(I,J,K)
            U(1,I,J,K) = UTEMP(1,I,J,K) + DTLOC*(R(1,I,J,K) +
     &       D(1,I,J,K))
            U(2,I,J,K) = UTEMP(2,I,J,K) + DTLOC*(R(2,I,J,K) +
     &       D(2,I,J,K))
            U(3,I,J,K) = UTEMP(3,I,J,K) + DTLOC*(R(3,I,J,K) +
     &       D(3,I,J,K))
            U(4,I,J,K) = UTEMP(4,I,J,K) + DTLOC*(R(4,I,J,K) +
     &       D(4,I,J,K))
            U(5,I,J,K) = UTEMP(5,I,J,K) + DTLOC*(R(5,I,J,K) +
     &       D(5,I,J,K))
  100     CONTINUE
  110   CONTINUE
  120 CONTINUE
      CALL BNDRYC
      CALL FLUX
C
C---  final stage
C
      DO 150 K = 1,KM
        DO 140 J = 1,JM
          DO 130 I = 1,IM
            DTLOC = DT(I,J,K)
            U(1,I,J,K) = UTEMP(1,I,J,K) + DTLOC*(R(1,I,J,K) +
     &       D(1,I,J,K))
            U(2,I,J,K) = UTEMP(2,I,J,K) + DTLOC*(R(2,I,J,K) +
     &       D(2,I,J,K))
            U(3,I,J,K) = UTEMP(3,I,J,K) + DTLOC*(R(3,I,J,K) +
     &       D(3,I,J,K))
            U(4,I,J,K) = UTEMP(4,I,J,K) + DTLOC*(R(4,I,J,K) +
     &       D(4,I,J,K))
            U(5,I,J,K) = UTEMP(5,I,J,K) + DTLOC*(R(5,I,J,K) +
     &       D(5,I,J,K))
  130     CONTINUE
  140   CONTINUE
  150 CONTINUE
C
C---  add enthalpy damping and determine the pressure field (don't ask
```

```
C     me how it works, I got this from FLO-52)
C
C      DO 180 K = 1,KM
C        DO 170 J = 1,JM
C          DO 160 I = 1,IM
C            RHO = U(1,I,J,K)
C            U1  = U(2,I,J,K)/RHO
C            V1  = U(3,I,J,K)/RHO
C            W1  = U(4,I,J,K)/RHO
C            EN  = U(5,I,J,K)
C            PR  = (EN - .5*(U1**2 + V1**2 + W1**2)*RHO)*(GAM - 1.)
C            HMH = (EN + PR)/RHO - HIN
C            H = 1./(1. + AENTH*HMH)
C            U(1,I,J,K) = RHO*H
C            U(2,I,J,K) = RHO*U1*H
C            U(3,I,J,K) = RHO*V1*H
C            U(4,I,J,K) = RHO*W1*H
C            RH = ((RHO*HMH - PR) - AENTH*PR)/(1. + AENTH)
C            U(5,I,J,K) = RH + RHO*H*HIN
C            P(I,J,K) = RHO*H*HMH - RH
C  160     CONTINUE
C  170   CONTINUE
C  180 CONTINUE
C
C---  calculate the residual, rms(delta-U)
C
      RMSRES = 0.
      DO 210 K = 1,KM
        DO 200 J = 1,JM
          DO 190 I = 1,IM
            RES = (U(1,I,J,K) - UTEMP(1,I,J,K))**2
            RES = RES + (U(2,I,J,K) - UTEMP(2,I,J,K))**2
            RES = RES + (U(3,I,J,K) - UTEMP(3,I,J,K))**2
            RES = RES + (U(4,I,J,K) - UTEMP(4,I,J,K))**2
            RES = RES + (U(5,I,J,K) - UTEMP(5,I,J,K))**2
            RMSRES = RMSRES + RES
  190     CONTINUE
  200   CONTINUE
  210 CONTINUE
      RMSRES = SQRT(RMSRES/(5.*FLOAT(NCELLS)))
      OPEN (2,STATUS='OLD',FORM='UNFORMATTED',ACCESS='APPEND',
     & FILE='RESIDUAL.PLT')
      WRITE (2) RMSRES
      CLOSE (2)
      RETURN
      END
```

```
C
C
C  SUBROUTINE CELL:  This subroutine calculates the volume of each
C  grid cell from the grid point locations for an arbitrary mesh
C
      SUBROUTINE CELL
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            NTYPE = MOD((I+J+K),2)
            IF (NTYPE .GT. 0) CALL AVOL(I,J,K)
            IF (NTYPE .EQ. 0) CALL BVOL(I,J,K)
10        CONTINUE
20      CONTINUE
30    CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE AVOL:  calculates volume of an "A" type cell
C
      SUBROUTINE AVOL (I,J,K)
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      COMMON/CELL/ FACEX(3,ISIZ,0:JSIZ,KSIZ),FACEY(3,ISIZ,0:JSIZ,KSIZ),
     &             FACEZ(3,ISIZ,0:JSIZ,KSIZ),VOLUME(ISIZ,0:JSIZ,KSIZ)
      COMMON/VERT/ XB,XC,XD,YB,YC,YD,ZB,ZC,ZD
      VOL = 0.
      X1 = X(I,J,K)
      Y1 = Y(I,J,K)
      Z1 = Z(I,J,K)
C
C---  to get cell volume, volume of individual tetrahedra composing
C     cell are determined individually and summed
C     T - 1236
C
      XB = X(I,J,K+1) - X1
      YB = Y(I,J,K+1) - Y1
      ZB = Z(I,J,K+1) - Z1
      XC = X(I+1,J,K+1) - X1
      YC = Y(I+1,J,K+1) - Y1
      ZC = Z(I+1,J,K+1) - Z1
      XD = X(I,J+1,K+1) - X1
      YD = Y(I,J+1,K+1) - Y1
      ZD = Z(I,J+1,K+1) - Z1
      CALL TETRA (VOL)
C
C---  T - 1368
C
      XB = XC
      YB = YC
      ZB = ZC
      XC = XD
      YC = YD
      ZC = ZD
      XD = X(I+1,J+1,K) - X1
      YD = Y(I+1,J+1,K) - Y1
      ZD = Z(I+1,J+1,K) - Z1
      CALL TETRA (VOL)
C
C---  T - 1685
C
      XB = XC
      YB = YC
      ZB = ZC
      XC = XD
      YC = YD
      ZC = ZD
      XD = X(I,J+1,K) - X1
      YD = Y(I,J+1,K) - Y1
      ZD = Z(I,J+1,K) - Z1
      CALL TETRA (VOL)
C
C---  T - 1834
C
      XB = XC
      YB = YC
```

```
      ZB = ZC
      XC = X(I+1,J,K+1) - X1
      YC = Y(I+1,J,K+1) - Y1
      ZC = Z(I+1,J,K+1) - Z1
      XD = X(I+1,J,K) - X1
      YD = Y(I+1,J,K) - Y1
      ZD = Z(I+1,J,K) - Z1
      CALL TETRA (VOL)
C
C---  T - 8637
C
      XC = X(I+1,J,K+1) - (X1 + XB)
      YC = Y(I+1,J,K+1) - (Y1 + YB)
      ZC = Z(I+1,J,K+1) - (Z1 + ZB)
      XD = X(I+1,J+1,K+1) - (X1 + XB)
      YD = Y(I+1,J+1,K+1) - (Y1 + YB)
      ZD = Z(I+1,J+1,K+1) - (Z1 + ZB)
      XB = X(I,J+1,K+1) - (X1 + XB)
      YB = Y(I,J+1,K+1) - (Y1 + YB)
      ZB = Z(I,J+1,K+1) - (Z1 + ZB)
      CALL TETRA (VOL)
C
C---  assign total cell volume to cell
C
      VOLUME(I,J,K) = VOL
      RETURN
      END
```

```
C
C
C  SUBROUTINE BVOL:  calculates volume of an "B" type cell
C
      SUBROUTINE BVOL (I,J,K)
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      COMMON/CELL/ FACEX(3,ISIZ,0:JSIZ,KSIZ),FACEY(3,ISIZ,0:JSIZ,KSIZ),
     &             FACEZ(3,ISIZ,0:JSIZ,KSIZ),VOLUME(ISIZ,0:JSIZ,KSIZ)
      COMMON/VERT/ XB,XC,XD,YB,YC,YD,ZB,ZC,ZD
      VOL = 0.
      X2 = X(I,J,K+1)
      Y2 = Y(I,J,K+1)
      Z2 = Z(I,J,K+1)
C
C---  to get cell volume, volume of individual tetrahedra composing
C     cell are determined individually and summed
C     T - 2347
C
      XB = X(I+1,J,K+1) - X2
      YB = Y(I+1,J,K+1) - Y2
      ZB = Z(I+1,J,K+1) - Z2
      XC = X(I+1,J,K) - X2
      YC = Y(I+1,J,K) - Y2
      ZC = Z(I+1,J,K) - Z2
      XD = X(I+1,J+1,K+1) - X2
      YD = Y(I+1,J+1,K+1) - Y2
      ZD = Z(I+1,J+1,K+1) - Z2
      CALL TETRA (VOL)
C
C---  T - 2475
C
      XB = XC
      YB = YC
      ZB = ZC
      XC = XD
      YC = YD
      ZC = ZD
      XD = X(I,J+1,K) - X2
      YD = Y(I,J+1,K) - Y2
      ZD = Z(I,J+1,K) - Z2
      CALL TETRA (VOL)
C
C---  T - 2756
C
      XB = XC
      YB = YC
      ZB = ZC
      XC = XD
      YC = YD
      ZC = ZD
      XD = X(I,J+1,K+1) - X2
      YD = Y(I,J+1,K+1) - Y2
      ZD = Z(I,J+1,K+1) - Z2
      CALL TETRA (VOL)
C
C---  T - 2541
C
      XB = XC
      YB = YC
```

```
      ZB = ZC
      XC = X(I+1,J,K) - X2
      YC = Y(I+1,J,K) - Y2
      ZC = Z(I+1,J,K) - Z2
      XD = X(I,J,K) - X2
      YD = Y(I,J,K) - Y2
      ZD = Z(I,J,K) - Z2
      CALL TETRA (VOL)
C
C---  T - 5478
C
      XC = X(I+1,J+1,K+1) - (X2 + XB)
      YC = Y(I+1,J+1,K+1) - (Y2 + YB)
      ZC = Z(I+1,J+1,K+1) - (Z2 + ZB)
      XD = X(I+1,J+1,K) - (X2 + XB)
      YD = Y(I+1,J+1,K) - (Y2 + YB)
      ZD = Z(I+1,J+1,K) - (Z2 + ZB)
      XB = X(I+1,J,K) - (X2 + XB)
      YB = Y(I+1,J,K) - (Y2 + YB)
      ZB = Z(I+1,J,K) - (Z2 + ZB)
      CALL TETRA (VOL)
C
C---  assign total cell volume to cell
C
      VOLUME(I,J,K) = VOL
      RETURN
      END
```

```
C
C
C  SUBROUTINE TETRA:  calculates volume of individual tetradron
C  given positions of vertices relative to local origin
C
      SUBROUTINE TETRA (V)
      COMMON/VERT/ XB,XC,XD,YB,YC,YD,ZB,ZC,ZD
      VOL = XB*(YD*ZC - ZD*YC) + YB*(XC*ZD - ZC*XD) + ZB*(YC*XD - XC*YD)
      V = V + ABS(VOL)/6.
      RETURN
      END
```

```
C
C
C  SUBROUTINE NORMAL:  This subroutine calculates the cell-face
C  normals of the finite volume cells
C
      SUBROUTINE NORMAL
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      COMMON/CELL/ FACEX(3,ISIZ,0:JSIZ,KSIZ),FACEY(3,ISIZ,0:JSIZ,KSIZ),
     &             FACEZ(3,ISIZ,0:JSIZ,KSIZ),VOLUME(ISIZ,0:JSIZ,KSIZ)
      COMMON/BOUN/ PRESYM(ISIZ,JSIZ),PRESWG(ISIZ,KSIZ),WALL(ISIZ,JSIZ)
      COMMON/BOU2/ WUN(3,0:ISIZ+1,0:KSIZ+1)
C
C---  determine the outward pointing normal on each cell face.
C     The name of the variable (FACEX, FACEY, FACEZ) identifies the
C     physical space component of the normal, and the first index of
C     the variable (1, 2, 3) identifies the computational space com-
C     ponent of the normal.  For example, FACEX(2,I,J,K) is the
C     x-component of the normal pointing in the j-direction out of cell
C     (I,J,K).  (You'll figure it out, don't worry.)
C     As was mentioned in the main routine, the computational coordinates
C     have been assumed to form a left-handed system, so be careful if
C     you make any changes.
C
C---  first, the I - face
C
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            X1 = X(I+1,J+1,K) - X(I+1,J,K+1)
            Y1 = Y(I+1,J+1,K) - Y(I+1,J,K+1)
            Z1 = Z(I+1,J+1,K) - Z(I+1,J,K+1)
            X2 = X(I+1,J+1,K+1) - X(I+1,J,K)
            Y2 = Y(I+1,J+1,K+1) - Y(I+1,J,K)
            Z2 = Z(I+1,J+1,K+1) - Z(I+1,J,K)
C
            FACEX(1,I,J,K) = -(Y1*Z2 - Z1*Y2)/2.
            FACEY(1,I,J,K) = -(Z1*X2 - X1*Z2)/2.
            FACEZ(1,I,J,K) = -(X1*Y2 - Y1*X2)/2.
C
C---  second, the J - face
C
            X1 = X(I+1,J+1,K+1) - X(I,J+1,K)
            Y1 = Y(I+1,J+1,K+1) - Y(I,J+1,K)
            Z1 = Z(I+1,J+1,K+1) - Z(I,J+1,K)
            X2 = X(I+1,J+1,K) - X(I,J+1,K+1)
            Y2 = Y(I+1,J+1,K) - Y(I,J+1,K+1)
            Z2 = Z(I+1,J+1,K) - Z(I,J+1,K+1)
C
            FACEX(2,I,J,K) = -(Y1*Z2 - Z1*Y2)/2.
            FACEY(2,I,J,K) = -(Z1*X2 - X1*Z2)/2.
            FACEZ(2,I,J,K) = -(X1*Y2 - Y1*X2)/2.
C
C---  third, the K - face
C
            X1 = X(I+1,J+1,K+1) - X(I,J,K+1)
            Y1 = Y(I+1,J+1,K+1) - Y(I,J,K+1)
            Z1 = Z(I+1,J+1,K+1) - Z(I,J,K+1)
            X2 = X(I,J+1,K+1) - X(I+1,J,K+1)
            Y2 = Y(I,J+1,K+1) - Y(I+1,J,K+1)
```

```
            Z2 = Z(I,J+1,K+1) - Z(I+1,J,K+1)
C
            FACEX(3,I,J,K) = -(Y1*Z2 - Z1*Y2)/2.
            FACEY(3,I,J,K) = -(Z1*X2 - X1*Z2)/2.
            FACEZ(3,I,J,K) = -(X1*Y2 - Y1*X2)/2.
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C
C---  find the normals on the symmetry plane
C
      DO 50 I = 1,IM
        DO 40 J = 1,JM
          X1 = X(I+1,J+1,1) - X(I,J,1)
          Z1 = Z(I+1,J+1,1) - Z(I,J,1)
          X2 = X(I+1,J,1) - X(I,J+1,1)
          Z2 = Z(I+1,J,1) - Z(I,J+1,1)
C
          WALL(I,J) = (Z1*X2 - X1*Z2)/2.
   40   CONTINUE
   50 CONTINUE
C
C---  find the normals on the wing
C
      DO 70 I = 1,IM
        DO 60 K = 1,KM
          X1 = X(I+1,1,K+1) - X(I,1,K)
          Y1 = Y(I+1,1,K+1) - Y(I,1,K)
          Z1 = Z(I+1,1,K+1) - Z(I,1,K)
          X2 = X(I,1,K+1) - X(I+1,1,K)
          Y2 = Y(I,1,K+1) - Y(I+1,1,K)
          Z2 = Z(I,1,K+1) - Z(I+1,1,K)
C
          FACEX(2,I,0,K) = (Y1*Z2 - Z1*Y2)/2.
          FACEY(2,I,0,K) = (Z1*X2 - X1*Z2)/2.
          FACEZ(2,I,0,K) = (X1*Y2 - Y1*X2)/2.
   60   CONTINUE
   70 CONTINUE
C
C---  find the unit normals on the wing
C
      DO 90 K = 1,KM
        DO 80 I = 1,IM
          ON = SQRT(FACEX(2,I,0,K)**2 + FACEY(2,I,0,K)**2 +
     &     FACEZ(2,I,0,K)**2)
          WUN(1,I,K) = FACEX(2,I,0,K)/ON
          WUN(2,I,K) = FACEY(2,I,0,K)/ON
          WUN(3,I,K) = FACEZ(2,I,0,K)/ON
   80   CONTINUE
   90 CONTINUE
        DO 100 K = 1,KM
          ONX = .5*(WUN(1,1,K) - WUN(1,IMAX-1,K))
          ONY = .5*(WUN(2,1,K) + WUN(2,IMAX-1,K))
          ONZ = .5*(WUN(3,1,K) - WUN(3,IMAX-1,K))
          ON = SQRT(ONX**2 + ONY**2 + ONZ**2)
C
          WUN(1,0,K) = ONX/ON
          WUN(2,0,K) = ONY/ON
          WUN(3,0,K) = ONZ/ON
          WUN(1,IMAX,K) = -ONX/ON
          WUN(2,IMAX,K) =  ONY/ON
```

```
        WUN(3,IMAX,K) = -ONZ/ON
  100 CONTINUE
      DO 110 I = 1,IM
        WUN(1,I,0) =  WUN(1,I,1)
        WUN(2,I,0) = -WUN(2,I,1)
        WUN(3,I,0) =  WUN(3,I,1)
        WUN(1,I,KMAX) = WUN(1,IMAX-I,KMAX-1)
        WUN(2,I,KMAX) = WUN(2,IMAX-I,KMAX-1)
        WUN(3,I,KMAX) = WUN(3,IMAX-I,KMAX-1)
  110 CONTINUE
C
C---  determine the metric coefficients at the wing surface for the
C     determination of the normal pressure gradient b.c.
C
      DO 130 K = 1,KM
        DO 120 I = 1,IM
          DXDXI = .5*(X(I+1,1,K) + X(I+1,1,K+1) - X(I,1,K) - X(I,1,K+1))
          DYDXI = .5*(Y(I+1,1,K) + Y(I+1,1,K+1) - Y(I,1,K) - Y(I,1,K+1))
          DZDXI = .5*(Z(I+1,1,K) + Z(I+1,1,K+1) - Z(I,1,K) - Z(I,1,K+1))
C
          DXDET = .5*(X(I,1,K+1) + X(I+1,1,K+1) - X(I,1,K) - X(I+1,1,K))
          DYDET = .5*(Y(I,1,K+1) + Y(I+1,1,K+1) - Y(I,1,K) - Y(I+1,1,K))
          DZDET = .5*(Z(I,1,K+1) + Z(I+1,1,K+1) - Z(I,1,K) - Z(I+1,1,K))
C
          DXDZE = -.125*(X(I,3,K) + X(I+1,3,K) + X(I,3,K+1) +
     &     X(I+1,3,K+1) - 4.*(X(I,2,K) + X(I+1,2,K) + X(I,2,K+1) +
     &     X(I+1,2,K+1)) + 3.*(X(I,1,K) + X(I+1,1,K) + X(I,1,K+1) +
     &     X(I+1,1,K+1)))
          DYDZE = -.125*(Y(I,3,K) + Y(I+1,3,K) + Y(I,3,K+1) +
     &     Y(I+1,3,K+1) - 4.*(Y(I,2,K) + Y(I+1,2,K) + Y(I,2,K+1) +
     &     Y(I+1,2,K+1)) + 3.*(Y(I,1,K) + Y(I+1,1,K) + Y(I,1,K+1) +
     &     Y(I+1,1,K+1)))
          DZDZE = -.125*(Z(I,3,K) + Z(I+1,3,K) + Z(I,3,K+1) +
     &     Z(I+1,3,K+1) - 4.*(Z(I,2,K) + Z(I+1,2,K) + Z(I,2,K+1) +
     &     Z(I+1,2,K+1)) + 3.*(Z(I,1,K) + Z(I+1,1,K) + Z(I,1,K+1) +
     &     Z(I+1,1,K+1)))
C
C---  determine the Jacobian of the transformation at the surface
C
          VOLUME(I,0,K) = FACEX(2,I,0,K)*DXDZE + FACEY(2,I,0,K)*
     &     DYDZE + FACEZ(2,I,0,K)*DZDZE
C
C---  store the surface metrics
C
          FACEX(1,I,0,K) = DYDET*DZDZE - DZDET*DYDZE
          FACEY(1,I,0,K) = DZDET*DXDZE - DXDET*DZDZE
          FACEZ(1,I,0,K) = DXDET*DYDZE - DYDET*DXDZE
C
          FACEX(3,I,0,K) = DYDZE*DZDXI - DZDZE*DYDXI
          FACEY(3,I,0,K) = DZDZE*DXDXI - DXDZE*DZDXI
          FACEZ(3,I,0,K) = DXDZE*DYDXI - DYDZE*DXDXI
  120   CONTINUE
  130 CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE ANEAN:  determines mean projected area in x,y,and z
C  directions of each grid cell for use in calculation of time step
C
      SUBROUTINE ANEAN
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      COMMON/CELL/ FACEX(3,ISIZ,0:JSIZ,KSIZ),FACEY(3,ISIZ,0:JSIZ,KSIZ),
     &             FACEZ(3,ISIZ,0:JSIZ,KSIZ),VOLUME(ISIZ,0:JSIZ,KSIZ)
      COMMON/BOUN/ PRESYM(ISIZ,JSIZ),PRESWG(ISIZ,KSIZ),WALL(ISIZ,JSIZ)
      COMMON/AREA/ AXM(ISIZ,JSIZ,KSIZ),AYM(ISIZ,JSIZ,KSIZ),
     &             AZM(ISIZ,JSIZ,KSIZ)
      DO 30 K = 2,KM
        KK = K - 1
        DO 20 J = 2,JM
          JJ = J - 1
          II = IM
          DO 10 I = 1,IM
            A1 = ABS(FACEX(1,I,J,K))
            A2 = ABS(FACEX(2,I,J,K))
            A3 = ABS(FACEX(3,I,J,K))
            A4 = ABS(FACEX(1,II,J,K))
            A5 = ABS(FACEX(2,I,JJ,K))
            A6 = ABS(FACEX(3,I,J,KK))
            AX = 0.
            AX = MAX(AX,(A1 + A2 + A3))
            AX = MAX(AX,(A2 + A3 + A4))
            AX = MAX(AX,(A2 + A4 + A6))
            AX = MAX(AX,(A1 + A2 + A6))
            AX = MAX(AX,(A1 + A3 + A5))
            AX = MAX(AX,(A3 + A4 + A5))
            AX = MAX(AX,(A4 + A5 + A6))
            AX = MAX(AX,(A1 + A5 + A6))
            AXM(I,J,K) = AX
C
            A1 = ABS(FACEY(1,I,J,K))
            A2 = ABS(FACEY(2,I,J,K))
            A3 = ABS(FACEY(3,I,J,K))
            A4 = ABS(FACEY(1,II,J,K))
            A5 = ABS(FACEY(2,I,JJ,K))
            A6 = ABS(FACEY(3,I,J,KK))
            AY = 0.
            AY = MAX(AY,(A1 + A2 + A3))
            AY = MAX(AY,(A2 + A3 + A4))
            AY = MAX(AY,(A2 + A4 + A6))
            AY = MAX(AY,(A1 + A2 + A6))
            AY = MAX(AY,(A1 + A3 + A5))
            AY = MAX(AY,(A3 + A4 + A5))
            AY = MAX(AY,(A4 + A5 + A6))
            AY = MAX(AY,(A1 + A5 + A6))
            AYM(I,J,K) = AY
C
            A1 = ABS(FACEZ(1,I,J,K))
            A2 = ABS(FACEZ(2,I,J,K))
            A3 = ABS(FACEZ(3,I,J,K))
            A4 = ABS(FACEZ(1,II,J,K))
            A5 = ABS(FACEZ(2,I,JJ,K))
            A6 = ABS(FACEZ(3,I,J,KK))
            AZ = 0.
```

```
            AZ = MAX(AZ,(A1 + A2 + A3))
            AZ = MAX(AZ,(A2 + A3 + A4))
            AZ = MAX(AZ,(A2 + A4 + A6))
            AZ = MAX(AZ,(A1 + A2 + A6))
            AZ = MAX(AZ,(A1 + A3 + A5))
            AZ = MAX(AZ,(A3 + A4 + A5))
            AZ = MAX(AZ,(A4 + A5 + A6))
            AZ = MAX(AZ,(A1 + A5 + A6))
            AZM(I,J,K) = AZ
C
            II = I
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C
      II = IM
      DO 50 I = 1,IM
        DO 40 J = 2,JM
          JJ = J - 1
          A1 = ABS(FACEX(1,I,J,1))
          A2 = ABS(FACEX(2,I,J,1))
          A3 = ABS(FACEX(3,I,J,1))
          A4 = ABS(FACEX(1,II,J,1))
          A5 = ABS(FACEX(2,I,JJ,1))
          A6 = 0.
          AX = 0.
          AX = MAX(AX,(A1 + A2 + A3))
          AX = MAX(AX,(A2 + A3 + A4))
          AX = MAX(AX,(A2 + A4 + A6))
          AX = MAX(AX,(A1 + A2 + A6))
          AX = MAX(AX,(A1 + A3 + A5))
          AX = MAX(AX,(A3 + A4 + A5))
          AX = MAX(AX,(A4 + A5 + A6))
          AX = MAX(AX,(A1 + A5 + A6))
          AXM(I,J,1) = AX
C
          A1 = ABS(FACEY(1,I,J,1))
          A2 = ABS(FACEY(2,I,J,1))
          A3 = ABS(FACEY(3,I,J,1))
          A4 = ABS(FACEY(1,II,J,1))
          A5 = ABS(FACEY(2,I,JJ,1))
          A6 = ABS(WALL(I,J))
          AY = 0.
          AY = MAX(AY,(A1 + A2 + A3))
          AY = MAX(AY,(A2 + A3 + A4))
          AY = MAX(AY,(A2 + A4 + A6))
          AY = MAX(AY,(A1 + A2 + A6))
          AY = MAX(AY,(A1 + A3 + A5))
          AY = MAX(AY,(A3 + A4 + A5))
          AY = MAX(AY,(A4 + A5 + A6))
          AY = MAX(AY,(A1 + A5 + A6))
          AYM(I,J,1) = AY
C
          A1 = ABS(FACEZ(1,I,J,1))
          A2 = ABS(FACEZ(2,I,J,1))
          A3 = ABS(FACEZ(3,I,J,1))
          A4 = ABS(FACEZ(1,II,J,1))
          A5 = ABS(FACEZ(2,I,JJ,1))
          A6 = 0.
          AZ = 0.
          AZ = MAX(AZ,(A1 + A2 + A3))
```

```
            AZ = MAX(AZ,(A2 + A3 + A4))
            AZ = MAX(AZ,(A2 + A4 + A6))
            AZ = MAX(AZ,(A1 + A2 + A6))
            AZ = MAX(AZ,(A1 + A3 + A5))
            AZ = MAX(AZ,(A3 + A4 + A5))
            AZ = MAX(AZ,(A4 + A5 + A6))
            AZ = MAX(AZ,(A1 + A5 + A6))
            AZM(I,J,1) = AZ
   40     CONTINUE
          II = I
   50   CONTINUE
C
        II = IM
        DO 70 I = 1,IM
          DO 60 K = 1,KM
            KK = K - 1
            A1 = ABS(FACEX(1,I,1,K))
            A2 = ABS(FACEX(2,I,1,K))
            A3 = ABS(FACEX(3,I,1,K))
            A4 = ABS(FACEX(1,II,1,K))
            A5 = ABS(FACEX(2,I,0,K))
            A6 = 0.
            IF (KK. GT .0) A6 = ABS(FACEX(3,I,1,KK))
            AX = 0.
            AX = MAX(AX,(A1 + A2 + A3))
            AX = MAX(AX,(A2 + A3 + A4))
            AX = MAX(AX,(A2 + A4 + A6))
            AX = MAX(AX,(A1 + A2 + A6))
            AX = MAX(AX,(A1 + A3 + A5))
            AX = MAX(AX,(A3 + A4 + A5))
            AX = MAX(AX,(A4 + A5 + A6))
            AX = MAX(AX,(A1 + A5 + A6))
            AXM(I,1,K) = AX
C
            A1 = ABS(FACEY(1,I,1,K))
            A2 = ABS(FACEY(2,I,1,K))
            A3 = ABS(FACEY(3,I,1,K))
            A4 = ABS(FACEY(1,II,1,K))
            A5 = ABS(FACEY(2,I,0,K))
            A6 = ABS(WALL(I,1))
            IF (KK. GT .0) A6 = ABS(FACEY(3,I,1,KK))
            AY = 0.
            AY = MAX(AY,(A1 + A2 + A3))
            AY = MAX(AY,(A2 + A3 + A4))
            AY = MAX(AY,(A2 + A4 + A6))
            AY = MAX(AY,(A1 + A2 + A6))
            AY = MAX(AY,(A1 + A3 + A5))
            AY = MAX(AY,(A3 + A4 + A5))
            AY = MAX(AY,(A4 + A5 + A6))
            AY = MAX(AY,(A1 + A5 + A6))
            AYM(I,1,K) = AY
C
            A1 = ABS(FACEZ(1,I,1,K))
            A2 = ABS(FACEZ(2,I,1,K))
            A3 = ABS(FACEZ(3,I,1,K))
            A4 = ABS(FACEZ(1,II,1,K))
            A5 = ABS(FACEZ(2,I,0,K))
            A6 = 0.
            IF (KK. GT .0) A6 = ABS(FACEZ(3,I,1,KK))
            AZ = 0.
            AZ = MAX(AZ,(A1 + A2 + A3))
```

```
            AZ = MAX(AZ,(A2 + A3 + A4))
            AZ = MAX(AZ,(A2 + A4 + A6))
            AZ = MAX(AZ,(A1 + A2 + A6))
            AZ = MAX(AZ,(A1 + A3 + A5))
            AZ = MAX(AZ,(A3 + A4 + A5))
            AZ = MAX(AZ,(A4 + A5 + A6))
            AZ = MAX(AZ,(A1 + A5 + A6))
            AZM(I,1,K) = AZ
60       CONTINUE
         II = I
70    CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE SURFCO:  writes the coordinates of the wing surface
C  to be used in plotting the pressure coefficients
C
      SUBROUTINE SURFCO
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      WRITE (3) IM,KM
      ILE = (IMAX + 1)/2
      I = ILE/4
      SPAN = Y(I,1,KMAX)
      DO 20 K = 1,KM
C
C---  find coordinate of spanwise station
C
        YAV = .5*((Y(1,1,K) + Y(1,1,K+1)))/SPAN
        WRITE (3) YAV
C
C---  find chord at span station K
C
        XTE = .5*(X(1,1,K) + X(1,1,K+1))
        XLE = .5*(X(ILE,1,K) + X(ILE,1,K+1))
        CHORD = XTE - XLE
C
C---  write chordwise coordinates at span station K
C
        DO 10 I = 1,IM
          XAV = .25*(X(I,1,K) + X(I+1,1,K) + X(I,1,K+1) + X(I+1,1,K+1))
          XAV = (XAV - XLE)/CHORD
          WRITE (3) XAV
   10   CONTINUE
   20 CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE PROUT:  writes surface pressure coefficients to a data
C  file for plotting.  Also prints out sectional and total lift and
C  drag coefficients.
C
      SUBROUTINE PROUT(ITER)
      PARAMETER (ISIZ = 48, JSIZ = 10, KSIZ = 10)
      COMMON/GRID/ X(ISIZ+1,JSIZ+1,KSIZ+1),Y(ISIZ+1,JSIZ+1,KSIZ+1),
     &             Z(ISIZ+1,JSIZ+1,KSIZ+1),IMAX,JMAX,KMAX,IM,JM,KM
      COMMON/CELL/ FACEX(3,ISIZ,0:JSIZ,KSIZ),FACEY(3,ISIZ,0:JSIZ,KSIZ),
     &             FACEZ(3,ISIZ,0:JSIZ,KSIZ),VOLUME(ISIZ,0:JSIZ,KSIZ)
      COMMON/BOUN/ PRESYM(ISIZ,JSIZ),PRESWG(ISIZ,KSIZ),WALL(ISIZ,JSIZ)
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH,CIN
      REAL MACH
      ILE = IM/2
      WRITE (3) ITER
      WRITE (6,1000) ITER
      ALPHA = ATAN2(WIN,UIN)
      SA = SIN(ALPHA)
      CA = COS(ALPHA)
      S = 0.
      CZT = 0.
      CXT = 0.
      DO 20 K = 1,KM
        CZ = 0.
        CX = 0.
        ASEC = 0.
        DO 10 I = 1,IM
          CP = 2.*(PRESWG(I,K) - 1.)/(GAM*MACH**2)
          WRITE (3) CP
          CZ = CZ - CP*FACEZ(2,I,0,K)
          CX = CX - CP*FACEX(2,I,0,K)
          IF (I. LE .ILE) ASEC = ASEC + ABS(FACEZ(2,I,0,K))
   10   CONTINUE
        CZT = CZT + CZ
        CXT = CXT + CX
        S = S + ASEC
C
C---  compute sectional Cl, Cd
C
        CL = (CZ*CA - CX*SA)/ASEC
        CD = (CZ*SA + CX*CA)/ASEC
        WRITE (6,1001) K,CL,CD
   20 CONTINUE
C
C---  compute wing CL, CD
C
      CL = (CZT*CA - CXT*SA)/S
      CD = (CZT*SA + CXT*CA)/S
      WRITE (6,1002) CL,CD
C
 1000 FORMAT(1X,'Iteration:',I5//1X,'span station',8X,'Cl'12X,'Cd'/
     & 1X,12('-'),4X,10('-'),4X,10('-'))
 1001 FORMAT(6X,I2,3X,2F14.4)
 1002 FORMAT(/1X,'wing Cl:',F8.4,4X,'wing Cd:',F8.4)
      RETURN
      END
```

C.2 Perturbation Euler code

```
      PROGRAM EULERV
C*************************************************************************
C                                                                        *
C            PROGRAM EULERV:  Tom Roberts, September 1985                *
C                                                                        *
C   A program to solve the Euler equations of motion for an inviscid,   *
C   ideal gas in three dimensions around a wing with a plane of sym-     *
C   symmetry at subsonic and transonic free stream Mach numbers,         *
C   using Jameson's four-stage finite volume scheme.                     *
C                                                                        *
C   This version of the Euler code is for debugging the Buning and       *
C   Steger perturbation scheme for the case of a wing interaction with   *
C   a streamwise vortex.  A Lamb vortex is used and the perturbation     *
C   flow field is assumed to be homenthalpic.  Enthalpy damping is       *
C   used to speed convergence.                                           *
C                                                                        *
C*************************************************************************
      COMMON/VAR / U(5,-1:50,-1:12,-1:12),P(-1:50,-1:12,-1:12),
     &             R(5,49,11,11),NCELLS,RO(5,48,10,10)
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      COMMON/CELL/ FACEX(3,48,0:10,10),FACEY(3,48,0:10,10),
     &             FACEZ(3,48,0:10,10),VOLUME(48,0:10,10)
      COMMON/BOUN/ PRESYM(48,10),PRESWG(48,10),WALL(48,10),UFAR(48,10),
     &             VFAR(48,10),WFAR(48,10),ENT(48,10)
      COMMON/DISP/ D(5,48,10,10),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH
      COMMON/TIME/ DT(48,10,10),ALPHA1,ALPHA2,ALPHA3
      COMMON/AREA/ AXM(48,10,10),AYM(48,10,10),AZM(48,10,10)
      COMMON/BOU2/ WUN(3,0:49,0:11)
      REAL KAP2, KAP4, MACH
C
      OPEN (1,STATUS='OLD',FORM='UNFORMATTED',FILE='WING.BIN')
      OPEN (2,STATUS='NEW',FORM='UNFORMATTED',FILE='RESIDUAL.PLT')
      OPEN (3,STATUS='NEW',FORM='UNFORMATTED',FILE='PRESSURE.PLT')
      OPEN (4,STATUS='NEW',FORM='UNFORMATTED',FILE='EULER.DMP')
      OPEN (7,STATUS='OLD',FILE='INPUT.DAT')
      CLOSE (4)
C
      PI = 4.*ATAN(1.)
C
C---  Read in the coordinates of the grid cell vertices and initialize
C     the grid
C
      READ (1) IMAX,JMAX,KMAX
      IM = IMAX - 1
      JM = JMAX - 1
      KM = KMAX - 1
      NCELLS = IM*JM*KM
      READ (1) (((X(I,J,K), I = 1,IMAX), J = 1,JMAX), K = 1,KMAX),
     &         (((Y(I,J,K), I = 1,IMAX), J = 1,JMAX), K = 1,KMAX),
     &         (((Z(I,J,K), I = 1,IMAX), J = 1,JMAX), K = 1,KMAX)
      CLOSE (1)
      CALL CELL
      CALL NORMAL
      CALL AMEAN
C
C---  read in the problem parameters
C
```

```
      READ (7,1000) AOA,MACH,GAM
      READ (7,1000) ALPHA1,ALPHA2,ALPHA3   ! coefs. for time-stepping
      READ (7,1000) KAP2,KAP4,CFL          ! art. visc. and CFL no.
      READ (7,1000) AENTH                  ! enthalpy damping coef.
      READ (7,1001) ITMAX,ITPRIN,ITER      ! max. no. of iterations
      WRITE (2) ITER
      WRITE (2) MACH,AOA,CFL,IM,JM,KM
      CLOSE (2)
      WRITE (3) MACH,AOA,CFL,IM,JM,KM
      CALL SURFCO
      CLOSE (3)
C
C---  calculate U-inf, W-inf, and free  stream total enthalpy
C
      AOA = PI*AOA/180.
      UIN = SQRT(GAM)*MACH*COS(AOA)
      WIN = SQRT(GAM)*MACH*SIN(AOA)
      HIN = GAM*(MACH**2/2. + 1./(GAM - 1.))
C
C---  initialize the state vector, and calculate the perturbation
C     flow field
C
      CALL INIT(ITER,*9998,*9999,AOA)
      CLOSE (7)
C
C---  start Euler solution procedure;  establish boundary conditions,
C     dissipation, determine initial value of residuals, and determine
C     size of local time step
C
    1 CALL BNDRYC
      CALL FLUX
      CALL DTSIZE
      CALL DISSIP
C
C---  call four-stage time stepping procedure
C
      CALL TIMSTP
C
      IF (MOD(ITER,ITPRIN). EQ .0) THEN
       CALL BNDRYC
       OPEN (3,STATUS='OLD',FORM='UNFORMATTED',ACCESS='APPEND',
     &  FILE='PRESSURE.PLT')
       CALL PROUT(ITER)
       CLOSE (3)
       OPEN (4,STATUS='OLD',FORM='UNFORMATTED',FILE='EULER.DMP')
       WRITE (4) IM,JMAX,KM
       WRITE (4) ((((U(N,I,J,K), N = 1,5), I = 1,IM),
     &               J = 1,JMAX), K = 1,KM)
       CLOSE (4)
      END IF
C
      ITER = ITER + 1
      IF (ITER. GT .ITMAX) GO TO 10
      GO TO 1
   10 CONTINUE
C
C---  determine average and maximum values of the total enthalpy
C     throughout the field
C
      HMAX = 0.
      HMIN = 10.*HIN
```

```
      HTOT = 0.
      SMAX = 0.
      SMIN = 100.
      DO 40 K = 1,KM
        DO 30 J = 1,JM
          DO 20 I = 1,IM
            HTEM = (U(5,I,J,K) + P(I,J,K))/U(1,I,J,K)
            HTOT = HTOT + HTEM
            IF (HTEM. GT .HMAX) THEN
             HMAX = HTEM
             IHMAX = I
             JHMAX = J
             KHMAX = K
            END IF
            IF (HTEM. LT .HMIN) THEN
             HMIN = HTEM
             IHMIN = I
             JHMIN = J
             KHMIN = K
            END IF
            STEM = P(I,J,K)/U(1,I,J,K)**GAM
            IF (STEM. GT .SMAX) THEN
             SMAX = STEM
             ISMAX = I
             JSMAX = J
             KSMAX = K
            END IF
            IF (STEM. LT .SMIN) THEN
             SMIN = STEM
             ISMIN = I
             JSMIN = J
             KSMIN = K
            END IF
   20     CONTINUE
   30   CONTINUE
   40 CONTINUE
      HAVG = HTOT/FLOAT(NCELLS)
      HAVG = HAVG/HIN*100.
      HMAX = HMAX/HIN*100.
      HMIN = HMIN/HIN*100.
      AOA = AOA*180./PI
      WRITE (6,1005) MACH,AOA,CFL,IM,JM,KM,ITMAX
      WRITE (6,1002) HAVG,HMAX,HMIN
      WRITE (6,1003) IHMAX,JHMAX,KHMAX
      WRITE (6,1004) IHMIN,JHMIN,KHMIN
      WRITE (6,1006) SMAX,ISMAX,JSMAX,KSMAX
      WRITE (6,1007) SMIN,ISMIN,JSMIN,KSMIN
C
 1000 FORMAT(3G10.5)
 1001 FORMAT(3I5)
 1002 FORMAT(/2X,'Average Enthalpy',4X,'Maximum Enthalpy',4X,
     & 'Minimum Enthalpy'/2X,16('-'),4X,16('-'),4X,16('-')/
     & F14.5,2F20.5)
 1003 FORMAT(/1X,'Cell of maximum stagnation enthalpy:',3I5)
 1004 FORMAT(1X,'Cell of minimum stagnation enthalpy:',3I5)
 1005 FORMAT(1X,'EULERCODE solution'//1X,'Mach no.:',3X,F6.3,6X,
     & 'Angle of attack (degrees):',F7.3/1X,'Courant no.:',F6.3,
     & 6X,'Grid size:',I4,' x',I4,' x',I4//1X,'Iterations:',I5/)
 1006 FORMAT(/1X,'Maximum entropy:',F8.4,3I5)
 1007 FORMAT(1X,'Minimum entropy:',F8.4,3I5)
C
```

```
      GO TO 9999
9998  WRITE (6,*) ' Input array does not match array size.'
9999  STOP
      END
```

```
C
C
C  SUBROUTINE BNDRYC:  sets the far-field and solid wall boundary
C  conditions for Jameson's explicit four-stage scheme.  At the
C  wing surface, the pressure is extrapolated from interior values;
C  far-field conditions are determined by assuming one-dimensional
C  flow normal to the boundary, and using Riemann invariants in
C  the normal direction.
C
      SUBROUTINE BNDRYC
      COMMON/VAR / U(5,-1:50,-1:12,-1:12),P(-1:50,-1:12,-1:12),
     &             R(5,49,11,11),NCELLS,RO(5,48,10,10)
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      COMMON/CELL/ FACEX(3,48,0:10,10),FACEY(3,48,0:10,10),
     &             FACEZ(3,48,0:10,10),VOLUME(48,0:10,10)
      COMMON/BOUN/ PRESYM(48,10),PRESWG(48,10),WALL(48,10),UFAR(48,10),
     &             VFAR(48,10),WFAR(48,10),ENT(48,10)
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH
      COMMON/BOU2/ WUN(3,0:49,0:11)
      REAL MACH
C
C---  first, determine the pressure field
C
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            RHO = U(1,I,J,K)
            U1  = U(2,I,J,K)/RHO
            V1  = U(3,I,J,K)/RHO
            W1  = U(4,I,J,K)/RHO
            EN  = U(5,I,J,K)
            P(I,J,K) = (EN - .5*(U1**2 + V1**2 + W1**2)*RHO)*
     &        (GAM - 1.)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      DO 31 K = 1,KM
        P(0,1,K) = P(IM,1,K)
        P(IMAX,1,K) = P(1,1,K)
   31 CONTINUE
      DO 32 I = 1,IM
        P(I,1,0) = P(I,1,1)
        P(I,1,KMAX) = P(IMAX-I,1,KMAX-1)
   32 CONTINUE
C
C---  next, determine the far-field conditions
C
      DO 60 K = 1,KM
        DO 50 I = 1,IM
C
C---  find density, velocity, and pressure at the last point
C     inside the computational domain, (i,J-1,k)
C
          RHOEX = U(1,I,JM,K)
          UEX   = U(2,I,JM,K)/RHOEX
          VEX   = U(3,I,JM,K)/RHOEX
          WEX   = U(4,I,JM,K)/RHOEX
          PEX   =   P(I,JM,K)
C
C---  find the outward unit normal to the computational domain
```

```
C     and determine the normal component of the free stream and
C     extrapolated velocities and sonic speeds
C
          ONX = FACEX(2,I,JM,K)
          ONY = FACEY(2,I,JM,K)
          ONZ = FACEZ(2,I,JM,K)
          ON  = SQRT(ONX**2 + ONY**2 + ONZ**2)
          ONX = ONX/ON
          ONY = ONY/ON
          ONZ = ONZ/ON
C
          UFF = UFAR(I,K) + UIN
          VFF = VFAR(I,K)
          WFF = WFAR(I,K) + WIN
          UINN = ONX*UFF + ONY*VFF + ONZ*WFF
          UEXN = ONX*UEX + ONY*VEX + ONZ*WEX
          CEX = SQRT(GAM*PEX/RHOEX)
          CIN = SQRT((GAM - 1.)*(HIN - .5*(UFF**2 + VFF**2 +
     &     WFF**2)))
          EV4 = UINN + CIN
          EV5 = UEXN - CEX
C
C---  compute the incoming and outgoing Riemann invariants
C
          RIN = UINN - 2.*CIN/(GAM - 1.)
          REX = UEXN + 2.*CEX/(GAM - 1.)
C
C---  supersonic inflow:  all variables specified
C
          IF (EV4. LT .0.) REX = UINN + 2.*CIN/(GAM - 1.)
C
C---  supersonic outflow:  all variables extrapolated
C
          IF (EV5. GT .0.) RIN = UEXN - 2.*CEX/(GAM - 1.)
C
C---  determine the normal velocity and the sonic speed at the boundary
C     cell (i,J,k)
C
          UN = (REX + RIN)/2.
          C  = (REX - RIN)*(GAM - 1.)/4.
          IF (UN. GT .0.) THEN
C
C---  outflow boundary:  extrapolate tangential velocity, entropy
C
           UP = UEX + (UN - UEXN)*ONX
           VP = VEX + (UN - UEXN)*ONY
           WP = WEX + (UN - UEXN)*ONZ
           S = PEX/RHOEX**GAM
          ELSE
C
C---  inflow boundary:  free stream tangential velocity, entropy
C
           UP = UFF + (UN - UINN)*ONX
           VP = VFF + (UN - UINN)*ONY
           WP = WFF + (UN - UINN)*ONZ
           S = ENT(I,K)
          END IF
C
C
C---  determine density, energy;  set conservation variables at boundary
C     cell
```

```
C
          ENER = (UP**2 + VP**2 + WP**2)/2. + C**2/(GAM*(GAM - 1.))
          RHO = (C**2/(GAM*S))**(1./(GAM - 1.))
C
          U(1,I,JMAX,K) = RHO
          U(2,I,JMAX,K) = RHO*UP
          U(3,I,JMAX,K) = RHO*VP
          U(4,I,JMAX,K) = RHO*WP
          U(5,I,JMAX,K) = RHO*ENER
          P(I,JMAX,K) = RHO*C**2/GAM
   50   CONTINUE
   60 CONTINUE
C
C---  determine the pressure at the wing and symmetry plane by
C     extrapolation from the interior--set (dP/dn) = 0 at the
C     symmetry plane; the normal momentum equation is used to
C     determine the pressure on the wing
C
C---  First, do the symmetry plane
C
      DO 80 J = 1,JM
        DO 70 I = 1,IM
          PRESYM(I,J) = P(I,J,1)
   70   CONTINUE
   80 CONTINUE
C
C---  Next, set do the wing surface
C
      DO 100 K = 1,KM
        DO 90 I = 1,IM
C
C---  determine the velocity and density in cell (i,1,k)
C
          RHO = U(1,I,1,K)
          U1  = U(2,I,1,K)/RHO
          V1  = U(3,I,1,K)/RHO
          W1  = U(4,I,1,K)/RHO
C
C---  determine the cartesian components of the surface velocity
C
          UN = WUN(1,I,K)*U1 + WUN(2,I,K)*V1 + WUN(3,I,K)*W1
          US = U1 - UN*WUN(1,I,K)
          VS = V1 - UN*WUN(2,I,K)
          WS = W1 - UN*WUN(3,I,K)
C
C---  find the contravariant components of the velocity and the unit
C     normal
C
          UC = US*FACEX(1,I,0,K) + VS*FACEY(1,I,0,K) + WS*FACEZ(1,I,0,K)
          VC = US*FACEX(3,I,0,K) + VS*FACEY(3,I,0,K) + WS*FACEZ(3,I,0,K)
C
          ONXI = WUN(1,I,K)*FACEX(1,I,0,K) + WUN(2,I,K)*FACEY(1,I,0,K) +
     &     WUN(3,I,K)*FACEZ(1,I,0,K)
          ONET = WUN(1,I,K)*FACEX(3,I,0,K) + WUN(2,I,K)*FACEY(3,I,0,K) +
     &     WUN(3,I,K)*FACEZ(3,I,0,K)
          ONZE = WUN(1,I,K)*FACEX(2,I,0,K) + WUN(2,I,K)*FACEY(2,I,0,K) +
     &     WUN(3,I,K)*FACEZ(2,I,0,K)
C
          UC = UC/VOLUME(I,0,K)
          VC = VC/VOLUME(I,0,K)
C
```

```
          ONXI = ONXI/VOLUME(I,0,K)
          ONET = ONET/VOLUME(I,0,K)
          ONZE = ONZE/VOLUME(I,0,K)
C
C---  determine dn/dxi, dn/deta
C
          DNXDXI = .5*(WUN(1,I+1,K) - WUN(1,I-1,K))
          DNYDXI = .5*(WUN(2,I+1,K) - WUN(2,I-1,K))
          DNZDXI = .5*(WUN(3,I+1,K) - WUN(3,I-1,K))
C
          DNXDET = .5*(WUN(1,I,K+1) - WUN(1,I,K-1))
          DNYDET = .5*(WUN(2,I,K+1) - WUN(2,I,K-1))
          DNZDET = .5*(WUN(3,I,K+1) - WUN(3,I,K-1))
C
C---  solve for the l.h.s. of the normal momentum equation
C
          RHOUU = RHO*(US*(UC*DNXDXI + VC*DNXDET) +
     &                 VS*(UC*DNYDXI + VC*DNYDET) +
     &                 WS*(UC*DNZDXI + VC*DNZDET))
C
C---  determine the surface pressure gradient
C
          DPDXI = .5*(P(I+1,1,K) - P(I-1,1,K))
          DPDET = .5*(P(I,1,K+1) - P(I,1,K-1))
C
C---  solve for dP/dzeta
C
          DPDZE = (RHOUU - ONXI*DPDXI - ONET*DPDET)/ONZE
C
C---  find the surface pressure
C
          PRESWG(I,K) = P(I,1,K) - .5*DPDZE
   90   CONTINUE
  100 CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE FLUX:  determine the fluxes across each finite volume cell
C  face, and the residual R for each cell.  This is done by averaging the
C  flux vectors between neighboring cells
C
      SUBROUTINE FLUX
      COMMON/VAR / U(5,-1:50,-1:12,-1:12),P(-1:50,-1:12,-1:12),
     &             R(5,49,11,11),NCELLS,RO(5,48,10,10)
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      COMMON/CELL/ FACEX(3,48,0:10,10),FACEY(3,48,0:10,10),
     &             FACEZ(3,48,0:10,10),VOLUME(48,0:10,10)
      COMMON/BOUN/ PRESYM(48,10),PRESWG(48,10),WALL(48,10),UFAR(48,10),
     &             VFAR(48,10),WFAR(48,10),ENT(48,10)
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH
      REAL MACH
C
C---  set-up dummy points
C
      DO 20 J = 1,JM
        DO 10 I = 1,IM
          U(1,I,J,KMAX) = U(1,IMAX-I,J,KMAX-1)
          U(2,I,J,KMAX) = U(2,IMAX-I,J,KMAX-1)
          U(3,I,J,KMAX) = U(3,IMAX-I,J,KMAX-1)
          U(4,I,J,KMAX) = U(4,IMAX-I,J,KMAX-1)
          U(5,I,J,KMAX) = U(5,IMAX-I,J,KMAX-1)
          P(I,J,KMAX) = P(IMAX-I,J,KMAX-1)
   10   CONTINUE
   20 CONTINUE
C
      DO 40 K = 1,KM
        DO 30 J = 1,JM
          U(1,IMAX,J,K) = U(1,1,J,K)
          U(2,IMAX,J,K) = U(2,1,J,K)
          U(3,IMAX,J,K) = U(3,1,J,K)
          U(4,IMAX,J,K) = U(4,1,J,K)
          U(5,IMAX,J,K) = U(5,1,J,K)
          P(IMAX,J,K) = P(1,J,K)
   30   CONTINUE
   40 CONTINUE
C
C---  first, initialize the residual field
C
      DO 70 K = 1,KMAX
        DO 60 J = 1,JMAX
          DO 50 I = 1,IMAX
            R(1,I,J,K) = 0.
            R(2,I,J,K) = 0.
            R(3,I,J,K) = 0.
            R(4,I,J,K) = 0.
            R(5,I,J,K) = 0.
   50     CONTINUE
   60   CONTINUE
   70 CONTINUE
C
C---  determine the residual at the symmetry plane cells
C
      DO 90 J = 1,JM
        DO 80 I = 1,IM
          R(3,I,J,1) = WALL(I,J)*PRESYM(I,J)
```

```
   80   CONTINUE
   90 CONTINUE
C
C---  add contribution due to pressures at the wing surface
C
      DO 110 K = 1,KM
        DO 100 I = 1,IM
          R(2,I,1,K) = R(2,I,1,K) + FACEX(2,I,0,K)*PRESWG(I,K)
          R(3,I,1,K) = R(3,I,1,K) + FACEY(2,I,0,K)*PRESWG(I,K)
          R(4,I,1,K) = R(4,I,1,K) + FACEZ(2,I,0,K)*PRESWG(I,K)
  100   CONTINUE
  110 CONTINUE
C
      DO 160 K = 1,KM
        DO 150 J = 1,JM
C
C---  first do the flux across the i-face
C
          DO 120 I = 1,IM
C
C---  find the flux vector for cells (i,j,k) and (i+1,j,k)
C
            QA = (U(2,I,J,K)*FACEX(1,I,J,K) + U(3,I,J,K)*FACEY(1,I,J,K)
     &       + U(4,I,J,K)*FACEZ(1,I,J,K))/U(1,I,J,K)
            H1 = U(1,I,J,K)*QA
            H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEX(1,I,J,K)
            H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEY(1,I,J,K)
            H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEZ(1,I,J,K)
            H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
            QAP = (U(2,I+1,J,K)*FACEX(1,I,J,K) + U(3,I+1,J,K)*
     &       FACEY(1,I,J,K) + U(4,I+1,J,K)*FACEZ(1,I,J,K))/U(1,I+1,J,K)
            H1P = U(1,I+1,J,K)*QAP
            H2P = U(2,I+1,J,K)*QAP + P(I+1,J,K)*FACEX(1,I,J,K)
            H3P = U(3,I+1,J,K)*QAP + P(I+1,J,K)*FACEY(1,I,J,K)
            H4P = U(4,I+1,J,K)*QAP + P(I+1,J,K)*FACEZ(1,I,J,K)
            H5P = (U(5,I+1,J,K) + P(I+1,J,K))*QAP
C
C---  find the average flux vector between cells (i,j,k) and (i+1,j,k)
C
            DR1 = .5*(H1 + H1P)
            DR2 = .5*(H2 + H2P)
            DR3 = .5*(H3 + H3P)
            DR4 = .5*(H4 + H4P)
            DR5 = .5*(H5 + H5P)
C
C---  subtract outgoing flux from cell (i,j,k) and add it to
C     cell (i+1,j,k)
C
            R(1,I+1,J,K) = R(1,I+1,J,K) + DR1
            R(2,I+1,J,K) = R(2,I+1,J,K) + DR2
            R(3,I+1,J,K) = R(3,I+1,J,K) + DR3
            R(4,I+1,J,K) = R(4,I+1,J,K) + DR4
            R(5,I+1,J,K) = R(5,I+1,J,K) + DR5
C
            R(1,I,J,K) = R(1,I,J,K) - DR1
            R(2,I,J,K) = R(2,I,J,K) - DR2
            R(3,I,J,K) = R(3,I,J,K) - DR3
            R(4,I,J,K) = R(4,I,J,K) - DR4
            R(5,I,J,K) = R(5,I,J,K) - DR5
C
```

```
  120     CONTINUE
C
C---  next do the flux across the j-face
C
          DO 130 I = 1,IM
C
C---  find the flux vector for cells (i,j,k) and (i,j+1,k)
C
            QA = (U(2,I,J,K)*FACEX(2,I J,K) + U(3,I,J,K)*FACEY(2,I,J,K)
     &       + U(4,I,J,K)*FACEZ(2,I,J,K))/U(1,I,J,K)
            H1 = U(1,I,J,K)*QA
            H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEX(2,I,J,K)
            H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEY(2,I,J,K)
            H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEZ(2,I,J,K)
            H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
            QAP = (U(2,I,J+1,K)*FACEX(2,I,J,K) + U(3,I,J+1,K)*
     &       FACEY(2,I,J,K) + U(4,I,J+1,K)*FACEZ(2,I,J,K))/U(1,I,J+1,K)
            H1P = U(1,I,J+1,K)*QAP
            H2P = U(2,I,J+1,K)*QAP + P(I,J+1,K)*FACEX(2,I,J,K)
            H3P = U(3,I,J+1,K)*QAP + P(I,J+1,K)*FACEY(2,I,J,K)
            H4P = U(4,I,J+1,K)*QAP + P(I,J+1,K)*FACEZ(2,I,J,K)
            H5P = (U(5,I,J+1,K) + P(I,J+1,K))*QAP
C
C---  find the average flux vector between cells (i,j,k) and (i,j+1,k)
C
            DR1 = .5*(H1 + H1P)
            DR2 = .5*(H2 + H2P)
            DR3 = .5*(H3 + H3P)
            DR4 = .5*(H4 + H4P)
            DR5 = .5*(H5 + H5P)
C
C---  subtract outgoing flux from cell (i,j,k) and add it to
C     cell (i,j,k+1)
C
            R(1,I,J,K) = R(1,I,J,K) - DR1
            R(2,I,J,K) = R(2,I,J,K) - DR2
            R(3,I,J,K) = R(3,I,J,K) - DR3
            R(4,I,J,K) = R(4,I,J,K) - DR4
            R(5,I,J,K) = R(5,I,J,K) - DR5
C
            R(1,I,J+1,K) = R(1,I,J+1,K) + DR1
            R(2,I,J+1,K) = R(2,I,J+1,K) + DR2
            R(3,I,J+1,K) = R(3,I,J+1,K) + DR3
            R(4,I,J+1,K) = R(4,I,J+1,K) + DR4
            R(5,I,J+1,K) = R(5,I,J+1,K) + DR5
C
  130     CONTINUE
C
C---  finally do the flux across the k-face
C
          DO 140 I = 1,IM
C
C---  find the flux vector for cells (i,j,k) and (i,j,k+1)
C
            QA = (U(2,I,J,K)*FACEX(3,I,J,K) + U(3,I,J,K)*FACEY(3,I,J,K)
     &       + U(4,I,J,K)*FACEZ(3,I,J,K))/U(1,I,J,K)
            H1 = U(1,I,J,K)*QA
            H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEX(3,I,J,K)
            H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEY(3,I,J,K)
            H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEZ(3,I,J,K)
```

```
            H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
            QAP = (U(2,I,J,K+1)*FACEX(3,I,J,K) + U(3,I,J,K+1)*
     &       FACEY(3,I,J,K) + U(4,I,J,K+1)*FACEZ(3,I,J,K))/U(1,I,J,K+1)
            H1P = U(1,I,J,K+1)*QAP
            H2P = U(2,I,J,K+1)*QAP + P(I,J,K+1)*FACEX(3,I,J,K)
            H3P = U(3,I,J,K+1)*QAP + P(I,J,K+1)*FACEY(3,I,J,K)
            H4P = U(4,I,J,K+1)*QAP + P(I,J,K+1)*FACEZ(3,I,J,K)
            H5P = (U(5,I,J,K+1) + P(I,J,K+1))*QAP
C
C---  find the average flux vector between cells (i,j,k) and (i,j,k+1)
C
            DR1 = .5*(H1 + H1P)
            DR2 = .5*(H2 + H2P)
            DR3 = .5*(H3 + H3P)
            DR4 = .5*(H4 + H4P)
            DR5 = .5*(H5 + H5P)
C
C---  subtract outgoing flux from cell (i,j,k) and add it to
C     cell (i,j,k+1)
C
            R(1,I,J,K) = R(1,I,J,K) - DR1
            R(2,I,J,K) = R(2,I,J,K) - DR2
            R(3,I,J,K) = R(3,I,J,K) - DR3
            R(4,I,J,K) = R(4,I,J,K) - DR4
            R(5,I,J,K) = R(5,I,J,K) - DR5
C
            R(1,I,J,K+1) = R(1,I,J,K+1) + DR1
            R(2,I,J,K+1) = R(2,I,J,K+1) + DR2
            R(3,I,J,K+1) = R(3,I,J,K+1) + DR3
            R(4,I,J,K+1) = R(4,I,J,K+1) + DR4
            R(5,I,J,K+1) = R(5,I,J,K+1) + DR5
C
  140     CONTINUE
  150   CONTINUE
  160 CONTINUE
C
      DO 180 K = 1,KM
        DO 170 J = 1,JM
          R(1,1,J,K) = R(1,1,J,K) + R(1,IMAX,J,K)
          R(2,1,J,K) = R(2,1,J,K) + R(2,IMAX,J,K)
          R(3,1,J,K) = R(3,1,J,K) + R(3,IMAX,J,K)
          R(4,1,J,K) = R(4,1,J,K) + R(4,IMAX,J,K)
          R(5,1,J,K) = R(5,1,J,K) + R(5,IMAX,J,K)
  170   CONTINUE
  180 CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE DISSIP:  calculates second- and fourth-order
C  dissipation terms in Jameson's explicit, multi-stage Euler
C  scheme
C
      SUBROUTINE DISSIP
      COMMON/VAR / U(5,-1:50,-1:12,-1:12),P(-1:50,-1:12,-1:12),
     &             R(5,49,11,11),NCELLS,RO(5,48,10,10)
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      COMMON/BOUN/ PRESYM(48,10),PRESWG(48,10),WALL(48,10),UFAR(48,10),
     &             VFAR(48,10),WFAR(48,10),ENT(48,10)
      COMMON/DISP/ D(5,48,10,10),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH
      COMMON/BOU2/ WUN(3,0:49,0:11)
      DIMENSION EP2(0:48), EP4(0:48)
      REAL KAP2, KAP4, MACH, NUM, NUP, NUMM, NUPM, MM, MP
C
C---  The value for the dissipation is found from the operator
C
C                      D  + D  + D  = D(u)
C                       X    Y    Z       ijk
C
C     as defined by Jameson, yielding second- and fourth-order
C     differences.  The operator in the X,Y,Z (I,J,K) coordinate
C     directions are determined in order.  Eriksson's treatment of the
C     smoothing at the solid wall and far-field boundaries is used.
C
C---  set-up dummy points in the i-, j-, and k-directions.
C
      DO 20 K = 1,KM
        DO 10 I = 1,IM
          U(1,I,JMAX+1,K) = 3.*U(1,I,JM,K) - 2.*U(1,I,JM-1,K)
          U(2,I,JMAX+1,K) = 3.*U(2,I,JM,K) - 2.*U(2,I,JM-1,K)
          U(3,I,JMAX+1,K) = 3.*U(3,I,JM,K) - 2.*U(3,I,JM-1,K)
          U(4,I,JMAX+1,K) = 3.*U(4,I,JM,K) - 2.*U(4,I,JM-1,K)
          U(5,I,JMAX+1,K) = 3.*U(5,I,JM,K) - 2.*U(5,I,JM-1,K)
          P(I,JMAX+1,K)   =   3.*P(I,JM,K) -   2.*P(I,JM-1,K)
C
          U(1,I,JMAX,K) = 2.*U(1,I,JM,K) - U(1,I,JM-1,K)
          U(2,I,JMAX,K) = 2.*U(2,I,JM,K) - U(2,I,JM-1,K)
          U(3,I,JMAX,K) = 2.*U(3,I,JM,K) - U(3,I,JM-1,K)
          U(4,I,JMAX,K) = 2.*U(4,I,JM,K) - U(4,I,JM-1,K)
          U(5,I,JMAX,K) = 2.*U(5,I,JM,K) - U(5,I,JM-1,K)
          P(I,JMAX,K)   =   2.*P(I,JM,K) -   P(I,JM-1,K)
C
          U(1,I,0,K) = 2.*U(1,I,1,K) - U(1,I,2,K)
          U(2,I,0,K) = 2.*U(2,I,1,K) - U(2,I,2,K)
          U(3,I,0,K) = 2.*U(3,I,1,K) - U(3,I,2,K)
          U(4,I,0,K) = 2.*U(4,I,1,K) - U(4,I,2,K)
          U(5,I,0,K) = 2.*U(5,I,1,K) - U(5,I,2,K)
          P(I,0,K)   =   2.*P(I,1,K) -   P(I,2,K)
C
          U(1,I,-1,K) = 3.*U(1,I,1,K) - 2.*U(1,I,2,K)
          U(2,I,-1,K) = 3.*U(2,I,1,K) - 2.*U(2,I,2,K)
          U(3,I,-1,K) = 3.*U(3,I,1,K) - 2.*U(3,I,2,K)
          U(4,I,-1,K) = 3.*U(4,I,1,K) - 2.*U(4,I,2,K)
          U(5,I,-1,K) = 3.*U(5,I,1,K) - 2.*U(5,I,2,K)
          P(I,-1,K)   =   3.*P(I,1,K) -   2.*P(I,2,K)
   10   CONTINUE
```

```
   20 CONTINUE
C
      DO 40 J = 1,JM
        DO 30 I = 1,IM
          U(1,I,J,KMAX) = U(1,IMAX-I,J,KMAX-1)
          U(2,I,J,KMAX) = U(2,IMAX-I,J,KMAX-1)
          U(3,I,J,KMAX) = U(3,IMAX-I,J,KMAX-1)
          U(4,I,J,KMAX) = U(4,IMAX-I,J,KMAX-1)
          U(5,I,J,KMAX) = U(5,IMAX-I,J,KMAX-1)
          P(I,J,KMAX)   =   P(IMAX-I,J,KMAX-1)
C
          U(1,I,J,KMAX+1) = U(1,IMAX-I,J,KMAX-2)
          U(2,I,J,KMAX+1) = U(2,IMAX-I,J,KMAX-2)
          U(3,I,J,KMAX+1) = U(3,IMAX-I,J,KMAX-2)
          U(4,I,J,KMAX+1) = U(4,IMAX-I,J,KMAX-2)
          U(5,I,J,KMAX+1) = U(5,IMAX-I,J,KMAX-2)
          P(I,J,KMAX+1)   =   P(IMAX-I,J,KMAX-2)
C
          U(1,I,J,0) =  U(1,I,J,1)
          U(2,I,J,0) =  U(2,I,J,1)
          U(3,I,J,0) = -U(3,I,J,1)
          U(4,I,J,0) =  U(4,I,J,1)
          U(5,I,J,0) =  U(5,I,J,1)
          P(I,J,0)   =    P(I,J,1)
C
          U(1,I,J,-1) =  U(1,I,J,2)
          U(2,I,J,-1) =  U(2,I,J,2)
          U(3,I,J,-1) = -U(3,I,J,2)
          U(4,I,J,-1) =  U(4,I,J,2)
          U(5,I,J,-1) =  U(5,I,J,2)
          P(I,J,-1)   =    P(I,J,2)
   30   CONTINUE
   40 CONTINUE
C
      DO 60 K = 1,KM
        DO 50 J = 1,JM
          U(1,IMAX+1,J,K) = U(1,2,J,K)
          U(2,IMAX+1,J,K) = U(2,2,J,K)
          U(3,IMAX+1,J,K) = U(3,2,J,K)
          U(4,IMAX+1,J,K) = U(4,2,J,K)
          U(5,IMAX+1,J,K) = U(5,2,J,K)
          P(IMAX+1,J,K)   =   P(2,J,K)
C
          U(1,IMAX,J,K) = U(1,1,J,K)
          U(2,IMAX,J,K) = U(2,1,J,K)
          U(3,IMAX,J,K) = U(3,1,J,K)
          U(4,IMAX,J,K) = U(4,1,J,K)
          U(5,IMAX,J,K) = U(5,1,J,K)
          P(IMAX,J,K)   =   P(1,J,K)
C
          U(1,0,J,K) = U(1,IM,J,K)
          U(2,0,J,K) = U(2,IM,J,K)
          U(3,0,J,K) = U(3,IM,J,K)
          U(4,0,J,K) = U(4,IM,J,K)
          U(5,0,J,K) = U(5,IM,J,K)
          P(0,J,K)   =   P(IM,J,K)
C
          U(1,-1,J,K) = U(1,IM-1,J,K)
          U(2,-1,J,K) = U(2,IM-1,J,K)
          U(3,-1,J,K) = U(3,IM-1,J,K)
          U(4,-1,J,K) = U(4,IM-1,J,K)
```

```
          U(5,-1,J,K) = U(5,IM-1,J,K)
          P(-1,J,K)   =   P(IM-1,J,K)
   50   CONTINUE
   60 CONTINUE
      DO 110 K = 1,KM
        DO 100 J = 1,JM
C
C---  begin with the i-direction:  find the values of the
C     dissipation coefficients
C
          DO 70 I = 0,IM
            NUM = ABS((P(I+1,J,K) - 2.*P(I,J,K) + P(I-1,J,K))/
     &                (P(I+1,J,K) + 2.*P(I,J,K) + P(I-1,J,K)))
            NUP = ABS((P(I+2,J,K) - 2.*P(I+1,J,K) + P(I,J,K))/
     &                (P(I+2,J,K) + 2.*P(I+1,J,K) + P(I,J,K)))
C
            EP = KAP2 * MAX(NUM,NUP)
            EP2(I) = EP
            EP4(I) = MAX(0.,(KAP4 - EP))
   70     CONTINUE
C
C---  calculate dissipation d(i,j,k)
C
          DO 80 I = 1,IM
            DS1 = EP2(I)*(U(1,I+1,J,K) - U(1,I,J,K))
            DF1 = EP4(I)*(U(1,I+2,J,K) - 3.*(U(1,I+1,J,K) -
     &            U(1,I,J,K)) - U(1,I-1,J,K))
            DS2 = EP2(I)*(U(2,I+1,J,K) - U(2,I,J,K))
            DF2 = EP4(I)*(U(2,I+2,J,K) - 3.*(U(2,I+1,J,K) -
     &            U(2,I,J,K)) - U(2,I-1,J,K))
            DS3 = EP2(I)*(U(3,I+1,J,K) - U(3,I,J,K))
            DF3 = EP4(I)*(U(3,I+2,J,K) - 3.*(U(3,I+1,J,K) -
     &            U(3,I,J,K)) - U(3,I-1,J,K))
            DS4 = EP2(I)*(U(4,I+1,J,K) - U(4,I,J,K))
            DF4 = EP4(I)*(U(4,I+2,J,K) - 3.*(U(4,I+1,J,K) -
     &            U(4,I,J,K)) - U(4,I-1,J,K))
            DS5 = EP2(I)*(U(5,I+1,J,K) - U(5,I,J,K) + P(I+1,J,K) -
     &            P(I,J,K))
            DF5 = EP4(I)*(U(5,I+2,J,K) - 3.*(U(5,I+1,J,K) -
     &            U(5,I,J,K)) - U(5,I-1,J,K) + P(I+2,J,K) - 3.*
     &            (P(I+1,J,K) - P(I,J,K)) - P(I-1,J,K))
C
            D(1,I,J,K) = DS1 - DF1
            D(2,I,J,K) = DS2 - DF2
            D(3,I,J,K) = DS3 - DF3
            D(4,I,J,K) = DS4 - DF4
            D(5,I,J,K) = DS5 - DF5
   80     CONTINUE
C
          DO 90 I = 1,IM
            DS1M = EP2(I-1)*(U(1,I,J,K) - U(1,I-1,J,K))
            DF1M = EP4(I-1)*(U(1,I+1,J,K) - 3.*(U(1,I,J,K) -
     &             U(1,I-1,J,K)) - U(1,I-2,J,K))
            DS2M = EP2(I-1)*(U(2,I,J,K) - U(2,I-1,J,K))
            DF2M = EP4(I-1)*(U(2,I+1,J,K) - 3.*(U(2,I,J,K) -
     &             U(2,I-1,J,K)) - U(2,I-2,J,K))
            DS3M = EP2(I-1)*(U(3,I,J,K) - U(3,I-1,J,K))
            DF3M = EP4(I-1)*(U(3,I+1,J,K) - 3.*(U(3,I,J,K) -
     &             U(3,I-1,J,K)) - U(3,I-2,J,K))
            DS4M = EP2(I-1)*(U(4,I,J,K) - U(4,I-1,J,K))
            DF4M = EP4(I-1)*(U(4,I+1,J,K) - 3.*(U(4,I,J,K) -
```

```
     &            U(4,I-1,J,K)) - U(4,I-2,J,K))
           DS5M = EP2(I-1)*(U(5,I,J,K) - U(5,I-1,J,K) + P(I,J,K) -
     &            P(I-1,J,K))
           DF5M = EP4(I-1)*(U(5,I+1,J,K) - 3.*(U(5,I,J,K) -
     &            U(5,I-1,J,K)) - U(5,I-2,J,K) + P(I+1,J,K) - 3.*
     &            (P(I,J,K) - P(I-1,J,K)) - P(I-2,J,K))
C
           D(1,I,J,K) = D(1,I,J,K) - DS1M + DF1M
           D(2,I,J,K) = D(2,I,J,K) - DS2M + DF2M
           D(3,I,J,K) = D(3,I,J,K) - DS3M + DF3M
           D(4,I,J,K) = D(4,I,J,K) - DS4M + DF4M
           D(5,I,J,K) = D(5,I,J,K) - DS5M + DF5M
   90    CONTINUE
  100  CONTINUE
  110 CONTINUE
      DO 170 K = 1,KM
        DO 160 J = 1,JM
C
C---  next do the j-direction:  find the values of the
C     dissipation coefficients
C
          DO 120 I = 1,IM
            NUM = ABS((P(I,J+1,K) - 2.*P(I,J,K) + P(I,J-1,K))/
     &                (P(I,J+1,K) + 2.*P(I,J,K) + P(I,J-1,K)))
            NUP = ABS((P(I,J+2,K) - 2.*P(I,J+1,K) + P(I,J,K))/
     &                (P(I,J+2,K) + 2.*P(I,J+1,K) + P(I,J,K)))
C
            EPP = KAP2 * MAX(NUM,NUP)
            EP2(I) = EPP
            EP4(I) = MAX(0.,(KAP4 - EPP))
  120     CONTINUE
C
C---  calculate dissipation d(i,j,k)
C
          DO 130 I = 1,IM
            DS1 = EP2(I)*(U(1,I,J+1,K) - U(1,I,J,K))
            DF1 = EP4(I)*(U(1,I,J+2,K) - 3.*(U(1,I,J+1,K) -
     &            U(1,I,J,K)) - U(1,I,J-1,K))
            DS2 = EP2(I)*(U(2,I,J+1,K) - U(2,I,J,K))
            DF2 = EP4(I)*(U(2,I,J+2,K) - 3.*(U(2,I,J+1,K) -
     &            U(2,I,J,K)) - U(2,I,J-1,K))
            DS3 = EP2(I)*(U(3,I,J+1,K) - U(3,I,J,K))
            DF3 = EP4(I)*(U(3,I,J+2,K) - 3.*(U(3,I,J+1,K) -
     &            U(3,I,J,K)) - U(3,I,J-1,K))
            DS4 = EP2(I)*(U(4,I,J+1,K) - U(4,I,J,K))
            DF4 = EP4(I)*(U(4,I,J+2,K) - 3.*(U(4,I,J+1,K) -
     &            U(4,I,J,K)) - U(4,I,J-1,K))
            DS5 = EP2(I)*(U(5,I,J+1,K) - U(5,I,J,K) + P(I,J+1,K) -
     &            P(I,J,K))
            DF5 = EP4(I)*(U(5,I,J+2,K) - 3.*(U(5,I,J+1,K) -
     &            U(5,I,J,K)) - U(5,I,J-1,K) + P(I,J+2,K) - 3.*
     &            (P(I,J+1,K) - P(I,J,K)) - P(I,J-1,K))
C
            D(1,I,J,K) = D(1,I,J,K) + DS1 - DF1
            D(2,I,J,K) = D(2,I,J,K) + DS2 - DF2
            D(3,I,J,K) = D(3,I,J,K) + DS3 - DF3
            D(4,I,J,K) = D(4,I,J,K) + DS4 - DF4
            D(5,I,J,K) = D(5,I,J,K) + DS5 - DF5
  130     CONTINUE
C
          DO 140 I = 1,IM
```

```
            NUMM = ABS((P(I,J,K) - 2.*P(I,J-1,K) + P(I,J-2,K))/
     &                 (P(I,J,K) + 2.*P(I,J-1,K) + P(I,J-2,K)))
            NUPM = ABS((P(I,J+1,K) - 2.*P(I,J,K) + P(I,J-1,K))/
     &                 (P(I,J+1,K) + 2.*P(I,J,K) + P(I,J-1,K)))
C
            EPM = KAP2 * MAX(NUMM,NUPM)
            EP2(I) = EPM
            EP4(I) = MAX(0.,(KAP4 - EPM))
  140     CONTINUE
C
          DO 150 I = 1,IM
            DS1M = EP2(I)*(U(1,I,J,K) - U(1,I,J-1,K))
            DF1M = EP4(I)*(U(1,I,J+1,K) - 3.*(U(1,I,J,K) -
     &             U(1,I,J-1,K)) - U(1,I,J-2,K))
            DS2M = EP2(I)*(U(2,I,J,K) - U(2,I,J-1,K))
            DF2M = EP4(I)*(U(2,I,J+1,K) - 3.*(U(2,I,J,K) -
     &             U(2,I,J-1,K)) - U(2,I,J-2,K))
            DS3M = EP2(I)*(U(3,I,J,K) - U(3,I,J-1,K))
            DF3M = EP4(I)*(U(3,I,J+1,K) - 3.*(U(3,I,J,K) -
     &             U(3,I,J-1,K)) - U(3,I,J-2,K))
            DS4M = EP2(I)*(U(4,I,J,K) - U(4,I,J-1,K))
            DF4M = EP4(I)*(U(4,I,J+1,K) - 3.*(U(4,I,J,K) -
     &             U(4,I,J-1,K)) - U(4,I,J-2,K))
            DS5M = EP2(I)*(U(5,I,J,K) - U(5,I,J-1,K) + P(I,J,K) -
     &             P(I,J-1,K))
            DF5M = EP4(I)*(U(5,I,J+1,K) - 3.*(U(5,I,J,K) -
     &             U(5,I,J-1,K)) - U(5,I,J-2,K) + P(I,J+1,K) - 3.*
     &             (P(I,J,K) - P(I,J-1,K)) - P(I,J-2,K))
C
            D(1,I,J,K) = D(1,I,J,K) - DS1M + DF1M
            D(2,I,J,K) = D(2,I,J,K) - DS2M + DF2M
            D(3,I,J,K) = D(3,I,J,K) - DS3M + DF3M
            D(4,I,J,K) = D(4,I,J,K) - DS4M + DF4M
            D(5,I,J,K) = D(5,I,J,K) - DS5M + DF5M
  150     CONTINUE
  160   CONTINUE
  170 CONTINUE
C
      DO 230 K = 1,KM
        DO 220 J = 1,JM
C
C---  next do the k-direction:  find the values of the
C     dissipation coefficients
C
          DO 180 I = 1,IM
            NUM = ABS((P(I,J,K+1) - 2.*P(I,J,K) + P(I,J,K-1))/
     &                (P(I,J,K+1) + 2.*P(I,J,K) + P(I,J,K-1)))
            NUP = ABS((P(I,J,K+2) - 2.*P(I,J,K+1) + P(I,J,K))/
     &                (P(I,J,K+2) + 2.*P(I,J,K+1) + P(I,J,K)))
C
            EPP = KAP2 * MAX(NUM,NUP)
            EP2(I) = EPP
            EP4(I) = MAX(0.,(KAP4 - EPP))
  180     CONTINUE
C
C---  calculate dissipation d(i,j,k)
C
          DO 190 I = 1,IM
            DS1 = EP2(I)*(U(1,I,J,K+1) - U(1,I,J,K))
            DF1 = EP4(I)*(U(1,I,J,K+2) - 3.*(U(1,I,J,K+1) -
     &            U(1,I,J,K)) - U(1,I,J,K-1))
```

```
            DS2 = EP2(I)*(U(2,I,J,K+1) - U(2,I,J,K))
            DF2 = EP4(I)*(U(2,I,J,K+2) - 3.*(U(2,I,J,K+1) -
     &            U(2,I,J,K)) - U(2,I,J,K-1))
            DS3 = EP2(I)*(U(3,I,J,K+1) - U(3,I,J,K))
            DF3 = EP4(I)*(U(3,I,J,K+2) - 3.*(U(3,I,J,K+1) -
     &            U(3,I,J,K)) - U(3,I,J,K-1))
            DS4 = EP2(I)*(U(4,I,J,K+1) - U(4,I,J,K))
            DF4 = EP4(I)*(U(4,I,J,K+2) - 3.*(U(4,I,J,K+1) -
     &            U(4,I,J,K)) - U(4,I,J,K-1))
            DS5 = EP2(I)*(U(5,I,J,K+1) - U(5,I,J,K) + P(I,J,K+1) -
     &            P(I,J,K))
            DF5 = EP4(I)*(U(5,I,J,K+2) - 3.*(U(5,I,J,K+1) -
     &            U(5,I,J,K)) - U(5,I,J,K-1) + P(I,J,K+2) - 3.*
     &            (P(I,J,K+1) - P(I,J,K)) - P(I,J,K-1))
C
            D(1,I,J,K) = D(1,I,J,K) + DS1 - DF1
            D(2,I,J,K) = D(2,I,J,K) + DS2 - DF2
            D(3,I,J,K) = D(3,I,J,K) + DS3 - DF3
            D(4,I,J,K) = D(4,I,J,K) + DS4 - DF4
            D(5,I,J,K) = D(5,I,J,K) + DS5 - DF5
  190     CONTINUE
C
          DO 200 I = 1,IM
            NUMM = ABS((P(I,J,K) - 2.*P(I,J,K-1) + P(I,J,K-2))/
     &                 (P(I,J,K) + 2.*P(I,J,K-1) + P(I,J,K-2)))
            NUPM = ABS((P(I,J,K+1) - 2.*P(I,J,K) + P(I,J,K-1))/
     &                 (P(I,J,K+1) + 2.*P(I,J,K) + P(I,J,K-1)))
C
            EPM = KAP2 * MAX(NUMM,NUPM)
            EP2(I) = EPM
            EP4(I) = MAX(0.,(KAP4 - EPM))
  200     CONTINUE
C
          DO 210 I = 1,IM
            DS1M = EP2(I)*(U(1,I,J,K) - U(1,I,J,K-1))
            DF1M = EP4(I)*(U(1,I,J,K+1) - 3.*(U(1,I,J,K) -
     &             U(1,I,J,K-1)) - U(1,I,J,K-2))
            DS2M = EP2(I)*(U(2,I,J,K) - U(2,I,J,K-1))
            DF2M = EP4(I)*(U(2,I,J,K+1) - 3.*(U(2,I,J,K) -
     &             U(2,I,J,K-1)) - U(2,I,J,K-2))
            DS3M = EP2(I)*(U(3,I,J,K) - U(3,I,J,K-1))
            DF3M = EP4(I)*(U(3,I,J,K+1) - 3.*(U(3,I,J,K) -
     &             U(3,I,J,K-1)) - U(3,I,J,K-2))
            DS4M = EP2(I)*(U(4,I,J,K) - U(4,I,J,K-1))
            DF4M = EP4(I)*(U(4,I,J,K+1) - 3.*(U(4,I,J,K) -
     &             U(4,I,J,K-1)) - U(4,I,J,K-2))
            DS5M = EP2(I)*(U(5,I,J,K) - U(5,I,J,K-1) + P(I,J,K) -
     &             P(I,J,K-1))
            DF5M = EP4(I)*(U(5,I,J,K+1) - 3.*(U(5,I,J,K) -
     &             U(5,I,J,K-1)) - U(5,I,J,K-2) + P(I,J,K+1) - 3.*
     &             (P(I,J,K) - P(I,J,K-1)) - P(I,J,K-2))
C
            D(1,I,J,K) = D(1,I,J,K) - DS1M + DF1M
            D(2,I,J,K) = D(2,I,J,K) - DS2M + DF2M
            D(3,I,J,K) = D(3,I,J,K) - DS3M + DF3M
            D(4,I,J,K) = D(4,I,J,K) - DS4M + DF4M
            D(5,I,J,K) = D(5,I,J,K) - DS5M + DF5M
  210     CONTINUE
  220   CONTINUE
  230 CONTINUE
      RETURN
```

```
END
```

```
C
C
C  SUBROUTINE DTSIZE:  calculates the value of the local time step
C  size based on the local CFL number restriction
C
      SUBROUTINE DTSIZE
      COMMON/VAR / U(5,-1:50,-1:12,-1:12),P(-1:50,-1:12,-1:12),
     &             R(5,49,11,11),NCELLS,RO(5,48,10,10)
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      COMMON/CELL/ FACEX(3,48,0:10,10),FACEY(3,48,0:10,10),
     &             FACEZ(3,48,0:10,10),VOLUME(48,0:10,10)
      COMMON/DISP/ D(5,48,10,10),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH
      COMMON/TIME/ DT(48,10,10),ALPHA1,ALPHA2,ALPHA3
      COMMON/AREA/ AXM(48,10,10),AYM(48,10,10),AZM(48,10,10)
      REAL KAP2, KAP4, MACH
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            RHO = U(1,I,J,K)
            U1  = U(2,I,J,K)/RHO
            V1  = U(3,I,J,K)/RHO
            W1  = U(4,I,J,K)/RHO
            C = SQRT(GAM*P(I,J,K)/RHO)
C
            AX = AXM(I,J,K)
            AY = AYM(I,J,K)
            AZ = AZM(I,J,K)
            AR = SQRT(AX**2 + AY**2 + AZ**2)
C
            UA = ABS(U1*AX)
            VA = ABS(V1*AY)
            WA = ABS(W1*AZ)
            CA = C*AR
            DT(I,J,K) = CFL/(UA + VA + WA + CA)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE TIMSTP:  advances solution Euler equations by a four-
C  stage time stepping scheme, using a spacially varying time step
C  and frozen dissipative terms in order to speed convergence
C
      SUBROUTINE TIMSTP
      COMMON/VAR / U(5,-1:50,-1:12,-1:12),P(-1:50,-1:12,-1:12),
     &             R(5,49,11,11),NCELLS,RO(5,48,10,10)
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      COMMON/CELL/ FACEX(3,48,0:10,10),FACEY(3,48,0:10,10),
     &             FACEZ(3,48,0:10,10),VOLUME(48,0:10,10)
      COMMON/DISP/ D(5,48,10,10),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH
      COMMON/TIME/ DT(48,10,10),ALPHA1,ALPHA2,ALPHA3
      DIMENSION UTEMP(5,48,10,10)
      REAL KAP2, KAP4, MACH
C
C---  store current state vector in temporary array UTEMP
C
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            UTEMP(1,I,J,K) = U(1,I,J,K)
            UTEMP(2,I,J,K) = U(2,I,J,K)
            UTEMP(3,I,J,K) = U(3,I,J,K)
            UTEMP(4,I,J,K) = U(4,I,J,K)
            UTEMP(5,I,J,K) = U(5,I,J,K)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C
C--- start four-stage time stepping procedure
C
      DO 60 K = 1,KM
        DO 50 J = 1,JM
          DO 40 I = 1,IM
            DTLOC = ALPHA1*DT(I,J,K)
            U(1,I,J,K) = UTEMP(1,I,J,K) + DTLOC*R(1,I,J,K) + ALPHA1*
     &       CFL*D(1,I,J,K) - DTLOC*RO(1,I,J,K)
            U(2,I,J,K) = UTEMP(2,I,J,K) + DTLOC*R(2,I,J,K) + ALPHA1*
     &       CFL*D(2,I,J,K) - DTLOC*RO(2,I,J,K)
            U(3,I,J,K) = UTEMP(3,I,J,K) + DTLOC*R(3,I,J,K) + ALPHA1*
     &       CFL*D(3,I,J,K) - DTLOC*RO(3,I,J,K)
            U(4,I,J,K) = UTEMP(4,I,J,K) + DTLOC*R(4,I,J,K) + ALPHA1*
     &       CFL*D(4,I,J,K) - DTLOC*RO(4,I,J,K)
            U(5,I,J,K) = UTEMP(5,I,J,K) + DTLOC*R(5,I,J,K) + ALPHA1*
     &       CFL*D(5,I,J,K) - DTLOC*RO(5,I,J,K)
   40     CONTINUE
   50   CONTINUE
   60 CONTINUE
      CALL BNDRYC
      CALL FLUX
C
C---  second stage
C
      DO 90 K = 1,KM
        DO 80 J = 1,JM
          DO 70 I = 1,IM
            DTLOC = ALPHA2*DT(I,J,K)
```

```
            U(1,I,J,K) = UTEMP(1,I,J,K) + DTLOC*R(1,I,J,K) + ALPHA2*
     &       CFL*D(1,I,J,K) - DTLOC*RO(1,I,J,K)
            U(2,I,J,K) = UTEMP(2,I,J,K) + DTLOC*R(2,I,J,K) + ALPHA2*
     &       CFL*D(2,I,J,K) - DTLOC*RO(2,I,J,K)
            U(3,I,J,K) = UTEMP(3,I,J,K) + DTLOC*R(3,I,J,K) + ALPHA2*
     &       CFL*D(3,I,J,K) - DTLOC*RO(3,I,J,K)
            U(4,I,J,K) = UTEMP(4,I,J,K) + DTLOC*R(4,I,J,K) + ALPHA2*
     &       CFL*D(4,I,J,K) - DTLOC*RO(4,I,J,K)
            U(5,I,J,K) = UTEMP(5,I,J,K) + DTLOC*R(5,I,J,K) + ALPHA2*
     &       CFL*D(5,I,J,K) - DTLOC*RO(5,I,J,K)
   70     CONTINUE
   80   CONTINUE
   90 CONTINUE
      CALL BNDRYC
      CALL FLUX
C
C--- third stage
C
      DO 120 K = 1,KM
        DO 110 J = 1,JM
          DO 100 I = 1,IM
            DTLOC = ALPHA3*DT(I,J,K)
            U(1,I,J,K) = UTEMP(1,I,J,K) + DTLOC*R(1,I,J,K) + ALPHA3*
     &       CFL*D(1,I,J,K) - DTLOC*RO(1,I,J,K)
            U(2,I,J,K) = UTEMP(2,I,J,K) + DTLOC*R(2,I,J,K) + ALPHA3*
     &       CFL*D(2,I,J,K) - DTLOC*RO(2,I,J,K)
            U(3,I,J,K) = UTEMP(3,I,J,K) + DTLOC*R(3,I,J,K) + ALPHA3*
     &       CFL*D(3,I,J,K) - DTLOC*RO(3,I,J,K)
            U(4,I,J,K) = UTEMP(4,I,J,K) + DTLOC*R(4,I,J,K) + ALPHA3*
     &       CFL*D(4,I,J,K) - DTLOC*RO(4,I,J,K)
            U(5,I,J,K) = UTEMP(5,I,J,K) + DTLOC*R(5,I,J,K) + ALPHA3*
     &       CFL*D(5,I,J,K) - DTLOC*RO(5,I,J,K)
  100     CONTINUE
  110   CONTINUE
  120 CONTINUE
      CALL BNDRYC
      CALL FLUX
C
C---  final stage
C
      DO 150 K = 1,KM
        DO 140 J = 1,JM
          DO 130 I = 1,IM
            DTLOC = DT(I,J,K)
            U(1,I,J,K) = UTEMP(1,I,J,K) + DTLOC*R(1,I,J,K) + CFL*
     &       D(1,I,J,K) - DTLOC*RO(1,I,J,K)
            U(2,I,J,K) = UTEMP(2,I,J,K) + DTLOC*R(2,I,J,K) + CFL*
     &       D(2,I,J,K) - DTLOC*RO(2,I,J,K)
            U(3,I,J,K) = UTEMP(3,I,J,K) + DTLOC*R(3,I,J,K) + CFL*
     &       D(3,I,J,K) - DTLOC*RO(3,I,J,K)
            U(4,I,J,K) = UTEMP(4,I,J,K) + DTLOC*R(4,I,J,K) + CFL*
     &       D(4,I,J,K) - DTLOC*RO(4,I,J,K)
            U(5,I,J,K) = UTEMP(5,I,J,K) + DTLOC*R(5,I,J,K) + CFL*
     &       D(5,I,J,K) - DTLOC*RO(5,I,J,K)
  130     CONTINUE
  140   CONTINUE
  150 CONTINUE
C
C---  add enthalpy damping and determine the pressure field
C
      DO 180 K = 1,KM
```

```
        DO 170 J = 1,JM
          DO 160 I = 1,IM
            RHO = U(1,I,J,K)
            U1  = U(2,I,J,K)/RHO
            V1  = U(3,I,J,K)/RHO
            W1  = U(4,I,J,K)/RHO
            EN  = U(5,I,J,K)
            PR  = (EN - .5*(U1**2 + V1**2 + W1**2)*RHO)*(GAM - 1.)
            HMH = (EN + PR)/RHO - HIN
            H = 1./(1. + AENTH*HMH)
            U(1,I,J,K) = RHO*H
            U(2,I,J,K) = RHO*U1*H
            U(3,I,J,K) = RHO*V1*H
            U(4,I,J,K) = RHO*W1*H
            RH = ((RHO*HMH - PR) - AENTH*PR)/(1. + AENTH)
            U(5,I,J,K) = RH + RHO*H*HIN
            P(I,J,K) = RHO*H*HMH - RH
  160     CONTINUE
  170   CONTINUE
  180 CONTINUE
C
C---  calculate the residual, rms(delta-U)
C
      RMSRES = 0.
      DO 210 K = 1,KM
        DO 200 J = 1,JM
          DO 190 I = 1,IM
            RES = (U(1,I,J,K) - UTEMP(1,I,J,K))**2
            RES = RES + (U(2,I,J,K) - UTEMP(2,I,J,K))**2
            RES = RES + (U(3,I,J,K) - UTEMP(3,I,J,K))**2
            RES = RES + (U(4,I,J,K) - UTEMP(4,I,J,K))**2
            RES = RES + (U(5,I,J,K) - UTEMP(5,I,J,K))**2
            RMSRES = RMSRES + RES
  190     CONTINUE
  200   CONTINUE
  210 CONTINUE
      RMSRES = SQRT(RMSRES/(5.*FLOAT(NCELLS)))
      OPEN (2,STATUS='OLD',FORM='UNFORMATTED',ACCESS='APPEND',
     & FILE='RESIDUAL.PLT')
      WRITE (2) RMSRES
      CLOSE (2)
      RETURN
      END
```

```
C
C
C  SUBROUTINE CELL:  This subroutine calculates the volume of each
C  grid cell from the grid point locations for an arbitrary mesh
C
      SUBROUTINE CELL
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            NTYPE = MOD((I+J+K),2)
            IF (NTYPE .GT. 0) CALL AVOL(I,J,K)
            IF (NTYPE .EQ. 0) CALL BVOL(I,J,K)
10        CONTINUE
20      CONTINUE
30    CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE AVOL:  calculates volume of an "A" type cell
C
      SUBROUTINE AVOL (I,J,K)
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      COMMON/CELL/ FACEX(3,48,0:10,10),FACEY(3,48,0:10,10),
     &             FACEZ(3,48,0:10,10),VOLUME(48,0:10,10)
      COMMON/VERT/ XB,XC,XD,YB,YC,YD,ZB,ZC,ZD
      VOL = 0.
      X1 = X(I,J,K)
      Y1 = Y(I,J,K)
      Z1 = Z(I,J,K)
C
C---  to get cell volume, volume of individual tetrahedra composing
C     cell are determined individually and summed
C     T - 1236
C
      XB = X(I,J,K+1) - X1
      YB = Y(I,J,K+1) - Y1
      ZB = Z(I,J,K+1) - Z1
      XC = X(I+1,J,K+1) - X1
      YC = Y(I+1,J,K+1) - Y1
      ZC = Z(I+1,J,K+1) - Z1
      XD = X(I,J+1,K+1) - X1
      YD = Y(I,J+1,K+1) - Y1
      ZD = Z(I,J+1,K+1) - Z1
      CALL TETRA (VOL)
C
C---  T - 1368
C
      XB = XC
      YB = YC
      ZB = ZC
      XC = XD
      YC = YD
      ZC = ZD
      XD = X(I+1,J+1,K) - X1
      YD = Y(I+1,J+1,K) - Y1
      ZD = Z(I+1,J+1,K) - Z1
      CALL TETRA (VOL)
C
C---  T - 1685
C
      XB = XC
      YB = YC
      ZB = ZC
      XC = XD
      YC = YD
      ZC = ZD
      XD = X(I,J+1,K) - X1
      YD = Y(I,J+1,K) - Y1
      ZD = Z(I,J+1,K) - Z1
      CALL TETRA (VOL)
C
C---  T - 1834
C
      XB = XC
      YB = YC
      ZB = ZC
```

```
      XC = X(I+1,J,K+1) - X1
      YC = Y(I+1,J,K+1) - Y1
      ZC = Z(I+1,J,K+1) - Z1
      XD = X(I+1,J,K) - X1
      YD = Y(I+1,J,K) - Y1
      ZD = Z(I+1,J,K) - Z1
      CALL TETRA (VOL)
C
C---  T - 6537
C
      XC = X(I+1,J,K+1) - (X1 + XB)
      YC = Y(I+1,J,K+1) - (Y1 + YB)
      ZC = Z(I+1,J,K+1) - (Z1 + ZB)
      XD = X(I+1,J+1,K+1) - (X1 + XB)
      YD = Y(I+1,J+1,K+1) - (Y1 + YB)
      ZD = Z(I+1,J+1,K+1) - (Z1 + ZB)
      XB = X(I,J+1,K+1) - (X1 + XB)
      YB = Y(I,J+1,K+1) - (Y1 + YB)
      ZB = Z(I,J+1,K+1) - (Z1 + ZB)
      CALL TETRA (VOL)
C
C---  assign total cell volume to cell
C
      VOLUME(I,J,K) = VOL
      RETURN
      END
```

```
C
C
C  SUBROUTINE BVOL:  calculates volume of an "B" type cell
C
      SUBROUTINE BVOL (I,J,K)
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      COMMON/CELL/ FACEX(3,48,0:10,10),FACEY(3,48,0:10,10),
     &             FACEZ(3,48,0:10,10),VOLUME(48,0:10,10)
      COMMON/VERT/ XB,XC,XD,YB,YC,YD,ZB,ZC,ZD
      VOL = 0.
      X2 = X(I,J,K+1)
      Y2 = Y(I,J,K+1)
      Z2 = Z(I,J,K+1)
C
C---  to get cell volume, volume of individual tetrahedra composing
C     cell are determined individually and summed
C     T - 2347
C
      XB = X(I+1,J,K+1) - X2
      YB = Y(I+1,J,K+1) - Y2
      ZB = Z(I+1,J,K+1) - Z2
      XC = X(I+1,J,K) - X2
      YC = Y(I+1,J,K) - Y2
      ZC = Z(I+1,J,K) - Z2
      XD = X(I+1,J+1,K+1) - X2
      YD = Y(I+1,J+1,K+1) - Y2
      ZD = Z(I+1,J+1,K+1) - Z2
      CALL TETRA (VOL)
C
C---  T - 2475
C
      XB = XC
      YB = YC
      ZB = ZC
      XC = XD
      YC = YD
      ZC = ZD
      XD = X(I,J+1,K) - X2
      YD = Y(I,J+1,K) - Y2
      ZD = Z(I,J+1,K) - Z2
      CALL TETRA (VOL)
C
C---  T - 2756
C
      XB = XC
      YB = YC
      ZB = ZC
      XC = XD
      YC = YD
      ZC = ZD
      XD = X(I,J+1,K+1) - X2
      YD = Y(I,J+1,K+1) - Y2
      ZD = Z(I,J+1,K+1) - Z2
      CALL TETRA (VOL)
C
C---  T - 2541
C
      XB = XC
      YB = YC
      ZB = ZC
```

```
      XC = X(I+1,J,K) - X2
      YC = Y(I+1,J,K) - Y2
      ZC = Z(I+1,J,K) - Z2
      XD = X(I,J,K) - X2
      YD = Y(I,J,K) - Y2
      ZD = Z(I,J,K) - Z2
      CALL TETRA (VOL)
C
C---  T - 5478
C
      XC = X(I+1,J+1,K+1) - (X2 + XB)
      YC = Y(I+1,J+1,K+1) - (Y2 + YB)
      ZC = Z(I+1,J+1,K+1) - (Z2 + ZB)
      XD = X(I+1,J+1,K) - (X2 + XB)
      YD = Y(I+1,J+1,K) - (Y2 + YB)
      ZD = Z(I+1,J+1,K) - (Z2 + ZB)
      XB = X(I+1,J,K) - (X2 + XB)
      YB = Y(I+1,J,K) - (Y2 + YB)
      ZB = Z(I+1,J,K) - (Z2 + ZB)
      CALL TETRA (VOL)
C
C---  assign total cell volume to cell
C
      VOLUME(I,J,K) = VOL
      RETURN
      END
```

```
C
C
C  SUBROUTINE TETRA:  calculates volume of individual tetradron
C  given positions of vertices relative to local origin
C
      SUBROUTINE TETRA (V)
      COMMON/VERT/ XB,XC,XD,YB,YC,YD,ZB,ZC,ZD
      VOL = XB*(YD*ZC - ZD*YC) + YB*(XC*ZD - ZC*XD) + ZB*(YC*XD - XC*YD)
      V = V + ABS(VOL)/6.
      RETURN
      END
```

```
C
C
C  SUBROUTINE NORMAL:  This subroutine calculates the cell-face
C  normals of the finite volume cells
C
      SUBROUTINE NORMAL
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      COMMON/CELL/ FACEX(3,48,0:10,10),FACEY(3,48,0:10,10),
     &             FACEZ(3,48,0:10,10),VOLUME(48,0:10,10)
      COMMON/BOUN/ PRESYM(48,10),PRESWG(48,10),WALL(48,10),UFAR(48,10),
     &             VFAR(48,10),WFAR(48,10),ENT(48,10)
      COMMON/BOU2/ WUN(3,0:49,0:11)
C
C---  determine the outward pointing normal on each cell face
C     first, the I - face
C
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            X1 = X(I+1,J+1,K) - X(I+1,J,K+1)
            Y1 = Y(I+1,J+1,K) - Y(I+1,J,K+1)
            Z1 = Z(I+1,J+1,K) - Z(I+1,J,K+1)
            X2 = X(I+1,J+1,K+1) - X(I+1,J,K)
            Y2 = Y(I+1,J+1,K+1) - Y(I+1,J,K)
            Z2 = Z(I+1,J+1,K+1) - Z(I+1,J,K)
C
            FACEX(1,I,J,K) = -(Y1*Z2 - Z1*Y2)/2.
            FACEY(1,I,J,K) = -(Z1*X2 - X1*Z2)/2.
            FACEZ(1,I,J,K) = -(X1*Y2 - Y1*X2)/2.
C
C---  second, the J - face
C
            X1 = X(I+1,J+1,K+1) - X(I,J+1,K)
            Y1 = Y(I+1,J+1,K+1) - Y(I,J+1,K)
            Z1 = Z(I+1,J+1,K+1) - Z(I,J+1,K)
            X2 = X(I+1,J+1,K) - X(I,J+1,K+1)
            Y2 = Y(I+1,J+1,K) - Y(I,J+1,K+1)
            Z2 = Z(I+1,J+1,K) - Z(I,J+1,K+1)
C
            FACEX(2,I,J,K) = -(Y1*Z2 - Z1*Y2)/2.
            FACEY(2,I,J,K) = -(Z1*X2 - X1*Z2)/2.
            FACEZ(2,I,J,K) = -(X1*Y2 - Y1*X2)/2.
C
C---  third, the K - face
C
            X1 = X(I+1,J+1,K+1) - X(I,J,K+1)
            Y1 = Y(I+1,J+1,K+1) - Y(I,J,K+1)
            Z1 = Z(I+1,J+1,K+1) - Z(I,J,K+1)
            X2 = X(I,J+1,K+1) - X(I+1,J,K+1)
            Y2 = Y(I,J+1,K+1) - Y(I+1,J,K+1)
            Z2 = Z(I,J+1,K+1) - Z(I+1,J,K+1)
C
            FACEX(3,I,J,K) = -(Y1*Z2 - Z1*Y2)/2.
            FACEY(3,I,J,K) = -(Z1*X2 - X1*Z2)/2.
            FACEZ(3,I,J,K) = -(X1*Y2 - Y1*X2)/2.
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C
C---  find the normals on the symmetry plane
```

```
C
      DO 50 I = 1,IM
        DO 40 J = 1,JM
          X1 = X(I+1,J+1,1) - X(I,J,1)
          Z1 = Z(I+1,J+1,1) - Z(I,J,1)
          X2 = X(I+1,J,1) - X(I,J+1,1)
          Z2 = Z(I+1,J,1) - Z(I,J+1,1)
C
          WALL(I,J) = (Z1*X2 - X1*Z2)/2.
   40   CONTINUE
   50 CONTINUE
C
C---  find the normals on the wing
C
      DO 70 I = 1,IM
        DO 60 K = 1,KM
          X1 = X(I+1,1,K+1) - X(I,1,K)
          Y1 = Y(I+1,1,K+1) - Y(I,1,K)
          Z1 = Z(I+1,1,K+1) - Z(I,1,K)
          X2 = X(I,1,K+1) - X(I+1,1,K)
          Y2 = Y(I,1,K+1) - Y(I+1,1,K)
          Z2 = Z(I,1,K+1) - Z(I+1,1,K)
C
          FACEX(2,I,0,K) = (Y1*Z2 - Z1*Y2)/2.
          FACEY(2,I,0,K) = (Z1*X2 - X1*Z2)/2.
          FACEZ(2,I,0,K) = (X1*Y2 - Y1*X2)/2.
   60   CONTINUE
   70 CONTINUE
C
C---  find the unit normals on the wing
C
      DO 90 K = 1,KM
        DO 80 I = 1,IM
          ON = SQRT(FACEX(2,I,0,K)**2 + FACEY(2,I,0,K)**2 +
     &      FACEZ(2,I,0,K)**2)
          WUN(1,I,K) = FACEX(2,I,0,K)/ON
          WUN(2,I,K) = FACEY(2,I,0,K)/ON
          WUN(3,I,K) = FACEZ(2,I,0,K)/ON
   80   CONTINUE
   90 CONTINUE
      DO 100 K = 1,KM
        WUN(1,0,K) = .5*(-WUN(1,IMAX-1,K) + WUN(1,1,K))
        WUN(2,0,K) = .5*( WUN(2,IMAX-1,K) + WUN(2,1,K))
        WUN(3,0,K) = .5*(-WUN(3,IMAX-1,K) + WUN(3,1,K))
        WUN(1,IMAX,K) = .5*(WUN(1,IMAX-1,K) - WUN(1,1,K))
        WUN(2,IMAX,K) = .5*(WUN(2,IMAX-1,K) + WUN(2,1,K))
        WUN(3,IMAX,K) = .5*(WUN(3,IMAX-1,K) - WUN(3,1,K))
  100 CONTINUE
      DO 110 I = 1,IM
        WUN(1,I,0) =  WUN(1,I,1)
        WUN(2,I,0) = -WUN(2,I,1)
        WUN(3,I,0) =  WUN(3,I,1)
        WUN(1,I,KMAX) = WUN(1,IMAX-I,KMAX-1)
        WUN(2,I,KMAX) = WUN(2,IMAX-I,KMAX-1)
        WUN(3,I,KMAX) = WUN(3,IMAX-I,KMAX-1)
  110 CONTINUE
C
C---  determine the metric coefficients at the wing surface for the
C     determination of the normal pressure gradient b.c.
C
      DO 130 K = 1,KM
```

```
        DO 120 I = 1,IM
          DXDXI = .5*(X(I+1,1,K) + X(I+1,1,K+1) - X(I,1,K) - X(I,1,K+1))
          DYDXI = .5*(Y(I+1,1,K) + Y(I+1,1,K+1) - Y(I,1,K) - Y(I,1,K+1))
          DZDXI = .5*(Z(I+1,1,K) + Z(I+1,1,K+1) - Z(I,1,K) - Z(I,1,K+1))
C
          DXDET = .5*(X(I,1,K+1) + X(I+1,1,K+1) - X(I,1,K) - X(I+1,1,K))
          DYDET = .5*(Y(I,1,K+1) + Y(I+1,1,K+1) - Y(I,1,K) - Y(I+1,1,K))
          DZDET = .5*(Z(I,1,K+1) + Z(I+1,1,K+1) - Z(I,1,K) - Z(I+1,1,K))
C
          DXDZE = -.125*(X(I,3,K) + X(I+1,3,K) + X(I,3,K+1) +
     &     X(I+1,3,K+1) - 4.*(X(I,2,K) + X(I+1,2,K) + X(I,2,K+1) +
     &     X(I+1,2,K+1)) + 3.*(X(I,1,K) + X(I+1,1,K) + X(I,1,K+1) +
     &     X(I+1,1,K+1)))
          DYDZE = -.125*(Y(I,3,K) + Y(I+1,3,K) + Y(I,3,K+1) +
     &     Y(I+1,3,K+1) - 4.*(Y(I,2,K) + Y(I+1,2,K) + Y(I,2,K+1) +
     &     Y(I+1,2,K+1)) + 3.*(Y(I,1,K) + Y(I+1,1,K) + Y(I,1,K+1) +
     &     Y(I+1,1,K+1)))
          DZDZE = -.125*(Z(I,3,K) + Z(I+1,3,K) + Z(I,3,K+1) +
     &     Z(I+1,3,K+1) - 4.*(Z(I,2,K) + Z(I+1,2,K) + Z(I,2,K+1) +
     &     Z(I+1,2,K+1)) + 3.*(Z(I,1,K) + Z(I+1,1,K) + Z(I,1,K+1) +
     &     Z(I+1,1,K+1)))
C
C---  determine the Jacobian of the transformation at the surface
C
          VOLUME(I,0,K) = FACEX(2,I,0,K)*DXDZE + FACEY(2,I,0,K)*
     &     DYDZE + FACEZ(2,I,0,K)*DZDZE
C
C---  store the surface metrics
C
          FACEX(1,I,0,K) = DYDET*DZDZE - DZDET*DYDZE
          FACEY(1,I,0,K) = DZDET*DXDZE - DXDET*DZDZE
          FACEZ(1,I,0,K) = DXDET*DYDZE - DYDET*DXDZE
C
          FACEX(3,I,0,K) = DYDZE*DZDXI - DZDZE*DYDXI
          FACEY(3,I,0,K) = DZDZE*DXDXI - DXDZE*DZDXI
          FACEZ(3,I,0,K) = DXDZE*DYDXI - DYDZE*DXDXI
  120   CONTINUE
  130 CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE AMEAN:  determines mean projected area in x,y,and z
C  directions of each grid cell for use in calculation of time step
C
      SUBROUTINE AMEAN
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      COMMON/CELL/ FACEX(3,48,0:10,10),FACEY(3,48,0:10,10),
     &             FACEZ(3,48,0:10,10),VOLUME(48,0:10,10)
      COMMON/BOUN/ PRESYM(48,10),PRESWG(48,10),WALL(48,10),UFAR(48,10),
     &             VFAR(48,10),WFAR(48,10),ENT(48,10)
      COMMON/AREA/ AXM(46,10,10),AYM(48,10,10),AZM(48,10,10)
      DO 30 K = 2,KM
        KK = K - 1
        DO 20 J = 2,JM
          JJ = J - 1
          II = IM
          DO 10 I = 1,IM
            A1 = ABS(FACEX(1,I,J,K))
            A2 = ABS(FACEX(2,I,J,K))
            A3 = ABS(FACEX(3,I,J,K))
            A4 = ABS(FACEX(1,II,J,K))
            A5 = ABS(FACEX(2,I,JJ,K))
            A6 = ABS(FACEX(3,I,J,KK))
            AX = 0.
            AX = MAX(AX,(A1 + A2 + A3))
            AX = MAX(AX,(A2 + A3 + A4))
            AX = MAX(AX,(A2 + A4 + A6))
            AX = MAX(AX,(A1 + A2 + A6))
            AX = MAX(AX,(A1 + A3 + A5))
            AX = MAX(AX,(A3 + A4 + A5))
            AX = MAX(AX,(A4 + A5 + A6))
            AX = MAX(AX,(A1 + A5 + A6))
            AXM(I,J,K) = AX
C
            A1 = ABS(FACEY(1,I,J,K))
            A2 = ABS(FACEY(2,I,J,K))
            A3 = ABS(FACEY(3,I,J,K))
            A4 = ABS(FACEY(1,II,J,K))
            A5 = ABS(FACEY(2,I,JJ,K))
            A6 = ABS(FACEY(3,I,J,KK))
            AY = 0.
            AY = MAX(AY,(A1 + A2 + A3))
            AY = MAX(AY,(A2 + A3 + A4))
            AY = MAX(AY,(A2 + A4 + A6))
            AY = MAX(AY,(A1 + A2 + A6))
            AY = MAX(AY,(A1 + A3 + A5))
            AY = MAX(AY,(A3 + A4 + A5))
            AY = MAX(AY,(A4 + A5 + A6))
            AY = MAX(AY,(A1 + A5 + A6))
            AYM(I,J,K) = AY
C
            A1 = ABS(FACEZ(1,I,J,K))
            A2 = ABS(FACEZ(2,I,J,K))
            A3 = ABS(FACEZ(3,I,J,K))
            A4 = ABS(FACEZ(1,II,J,K))
            A5 = ABS(FACEZ(2,I,JJ,K))
            A6 = ABS(FACEZ(3,I,J,KK))
            AZ = 0.
            AZ = MAX(AZ,(A1 + A2 + A3))
```

```
            AZ = MAX(AZ,(A2 + A3 + A4))
            AZ = MAX(AZ,(A2 + A4 + A6))
            AZ = MAX(AZ,(A1 + A2 + A6))
            AZ = MAX(AZ,(A1 + A3 + A5))
            AZ = MAX(AZ,(A3 + A4 + A5))
            AZ = MAX(AZ,(A4 + A5 + A6))
            AZ = MAX(AZ,(A1 + A5 + A6))
            AZM(I,J,K) = AZ
C
            II = I
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C
      II = IM
      DO 50 I = 1,IM
        DO 40 J = 2,JM
          JJ = J - 1
          A1 = ABS(FACEX(1,I,J,1))
          A2 = ABS(FACEX(2,I,J,1))
          A3 = ABS(FACEX(3,I,J,1))
          A4 = ABS(FACEX(1,II,J,1))
          A5 = ABS(FACEX(2,I,JJ,1))
          A6 = 0.
          AX = 0.
          AX = MAX(AX,(A1 + A2 + A3))
          AX = MAX(AX,(A2 + A3 + A4))
          AX = MAX(AX,(A2 + A4 + A6))
          AX = MAX(AX,(A1 + A2 + A6))
          AX = MAX(AX,(A1 + A3 + A5))
          AX = MAX(AX,(A3 + A4 + A5))
          AX = MAX(AX,(A4 + A5 + A6))
          AX = MAX(AX,(A1 + A5 + A6))
          AXM(I,J,1) = AX
C
          A1 = ABS(FACEY(1,I,J,1))
          A2 = ABS(FACEY(2,I,J,1))
          A3 = ABS(FACEY(3,I,J,1))
          A4 = ABS(FACEY(1,II,J,1))
          A5 = ABS(FACEY(2,I,JJ,1))
          A6 = ABS(WALL(I,J))
          AY = 0.
          AY = MAX(AY,(A1 + A2 + A3))
          AY = MAX(AY,(A2 + A3 + A4))
          AY = MAX(AY,(A2 + A4 + A6))
          AY = MAX(AY,(A1 + A2 + A6))
          AY = MAX(AY,(A1 + A3 + A5))
          AY = MAX(AY,(A3 + A4 + A5))
          AY = MAX(AY,(A4 + A5 + A6))
          AY = MAX(AY,(A1 + A5 + A6))
          AYM(I,J,1) = AY
C
          A1 = ABS(FACEZ(1,I,J,1))
          A2 = ABS(FACEZ(2,I,J,1))
          A3 = ABS(FACEZ(3,I,J,1))
          A4 = ABS(FACEZ(1,II,J,1))
          A5 = ABS(FACEZ(2,I,JJ,1))
          A6 = 0.
          AZ = 0.
          AZ = MAX(AZ,(A1 + A2 + A3))
          AZ = MAX(AZ,(A2 + A3 + A4))
```

```
          AZ = MAX(AZ,(A2 + A4 + A6))
          AZ = MAX(AZ,(A1 + A2 + A6))
          AZ = MAX(AZ,(A1 + A3 + A5))
          AZ = MAX(AZ,(A3 + A4 + A5))
          AZ = MAX(AZ,(A4 + A5 + A6))
          AZ = MAX(AZ,(A1 + A5 + A6))
          AZM(I,J,1) = AZ
   40   CONTINUE
        II = I
   50 CONTINUE
C
      II = IM
      DO 70 I = 1,IM
        DO 60 K = 1,KM
          KK = K - 1
          A1 = ABS(FACEX(1,I,1,K))
          A2 = ABS(FACEX(2,I,1,K))
          A3 = ABS(FACEX(3,I,1,K))
          A4 = ABS(FACEX(1,II,1,K))
          A5 = ABS(FACEX(2,I,0,K))
          A6 = 0.
          IF (KK. GT .0) A6 = ABS(FACEX(3,I,1,KK))
          AX = 0.
          AX = MAX(AX,(A1 + A2 + A3))
          AX = MAX(AX,(A2 + A3 + A4))
          AX = MAX(AX,(A2 + A4 + A6))
          AX = MAX(AX,(A1 + A2 + A6))
          AX = MAX(AX,(A1 + A3 + A5))
          AX = MAX(AX,(A3 + A4 + A5))
          AX = MAX(AX,(A4 + A5 + A6))
          AX = MAX(AX,(A1 + A5 + A6))
          AXM(I,1,K) = AX
C
          A1 = ABS(FACEY(1,I,1,K))
          A2 = ABS(FACEY(2,I,1,K))
          A3 = ABS(FACEY(3,I,1,K))
          A4 = ABS(FACEY(1,II,1,K))
          A5 = ABS(FACEY(2,I,0,K))
          A6 = ABS(WALL(I,1))
          IF (KK. GT .0) A6 = ABS(FACEY(3,I,1,KK))
          AY = 0.
          AY = MAX(AY,(A1 + A2 + A3))
          AY = MAX(AY,(A2 + A3 + A4))
          AY = MAX(AY,(A2 + A4 + A6))
          AY = MAX(AY,(A1 + A2 + A6))
          AY = MAX(AY,(A1 + A3 + A5))
          AY = MAX(AY,(A3 + A4 + A5))
          AY = MAX(AY,(A4 + A5 + A6))
          AY = MAX(AY,(A1 + A5 + A6))
          AYM(I,1,K) = AY
C
          A1 = ABS(FACEZ(1,I,1,K))
          A2 = ABS(FACEZ(2,I,1,K))
          A3 = ABS(FACEZ(3,I,1,K))
          A4 = ABS(FACEZ(1,II,1,K))
          A5 = ABS(FACEZ(2,I,0,K))
          A6 = 0.
          IF (KK. GT .0) A6 = ABS(FACEZ(3,I,1,KK))
          AZ = 0.
          AZ = MAX(AZ,(A1 + A2 + A3))
          AZ = MAX(AZ,(A2 + A3 + A4))
```

```
      AZ = MAX(AZ,(A2 + A4 + A6))
      AZ = MAX(AZ,(A1 + A2 + A6))
      AZ = MAX(AZ,(Ai + A3 + A5))
      AZ = MAX(AZ,(A3 + A4 + A5))
      AZ = MAX(AZ,(A4 + A5 + A6))
      AZ = MAX(AZ,(A1 + A5 + A6))
      AZM(I,1,K) = AZ
60    CONTINUE
      II = I
70  CONTINUE
    RETURN
    END
```

```
C
C
C  SUBROUTINE SURFCO:  writes the coordinates of the wing surface
C  to be used in plotting the pressure coefficients
C
      SUBROUTINE SURFCO
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      WRITE (3) IM,KM
      ILE = (IMAX + 1)/2
      I = ILE/4
      SPAN = Y(I,1,KMAX)
      DO 20 K = 1,KM
C
C---  find coordinate of spanwise station
C
        YAV = .5*((Y(1,1,K) + Y(1,1,K+1)))/SPAN
        WRITE (3) YAV
C
C---  find chord at span station K
C
        XTE = .5*(X(1,1,K) + X(1,1,K+1))
        XLE = .5*(X(ILE,1,K) + X(ILE,1,K+1))
        CHORD = XTE - XLE
C
C---  write chordwise coordinates at span station K
C
        DO 10 I = 1,IM
          XAV = .25*(X(I,1,K) + X(I+1,1,K) + X(I,1,K+1) + X(I+1,1,K+1))
          XAV = (XAV - XLE)/CHORD
          WRITE (3) XAV
   10   CONTINUE
   20 CONTINUE
      RETURN
      END
```

```
C
C
C  SUBROUTINE PROUT:  writes surface pressure coefficients to a data
C  file for plotting.  Also prints out sectional and total lift and
C  drag coefficients.
C
      SUBROUTINE PROUT(ITER)
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      COMMON/CELL/ FACEX(3,48,0:10,10),FACEY(3,48,0:10,10),
     &             FACEZ(3,48,0:10,10),VOLUME(48,0:10,10)
      COMMON/BOUN/ PRESYM(48,10),PRESWG(48,10),WALL(48,10),UFAR(48,10),
     &             VFAR(48,10),WFAR(48,10),ENT(48,10)
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH
      REAL MACH
      ILE = IM/2
      WRITE (3) ITER
      WRITE (6,1000) ITER
      ALPHA = ATAN2(WIN,UIN)
      SA = SIN(ALPHA)
      CA = COS(ALPHA)
      S = 0.
      CZT = 0.
      CXT = 0.
      DO 20 K = 1,KM
        CZ = 0.
        CX = 0.
        ASEC = 0.
        DO 10 I = 1,IM
          CP = 2.*(PRESWG(I,K) - 1.)/(GAM*MACH**2)
          WRITE (3) CP
          CZ = CZ - CP*FACEZ(2,I,0,K)
          CX = CX - CP*FACEX(2,I,0,K)
          IF (I. LE .ILE) ASEC = ASEC + ABS(FACEZ(2,I,0,K))
   10   CONTINUE
        CZT = CZT + CZ
        CXT = CXT + CX
        S = S + ASEC
C
C---  compute sectional Cl, Cd
C
        CL = (CZ*CA - CX*SA)/ASEC
        CD = (CZ*SA + CX*CA)/ASEC
        WRITE (6,1001) K,CL,CD
   20 CONTINUE
C
C---  compute wing CL, CD
C
      CL = (CZT*CA - CXT*SA)/S
      CD = (CZT*SA + CXT*CA)/S
      WRITE (6,1002) CL,CD
C
 1000 FORMAT(1X,'Iteration:',I5//1X,'span station',8X,'Cl'12X,'Cd'/
     & 1X,12('-'),4X,10('-'),4X,10('-'))
 1001 FORMAT(6X,I2,3X,2F14.4)
 1002 FORMAT(/1X,'wing Cl:',F9.5,4X,'wing Cd:',F10.6)
      RETURN
      END
```

```
C
C
C  SUBROUTINE INIT:  initializes the state vector for the Euler
C  solver by setting the variables to their freestream values;  the
C  perturbation field due to a specified vortex is also calculated
C  and the result is stored in RO.  Also, the solution may be restarted
C  by reading in an old solution.
C
      SUBROUTINE INIT(ISTART,*,*,AOA)
      COMMON/VAR / U(5,-1:50,-1:12,-1:12),P(-1:50,-1:12,-1:12),
     &             R(5,49,11,11),NCELLS,RO(5,48,10,10)
      COMMON/GRID/ X(49,11,11),Y(49,11,11),Z(49,11,11),IMAX,JMAX,KMAX,
     &             IM,JM,KM
      COMMON/CELL/ FACEX(3,48,0:10,10),FACEY(3,48,0:10,10),
     &             FACEZ(3,48,0:10,10),VOLUME(48,0:10,10)
      COMMON/BOUN/ PRESYM(48,10),PRESWG(48,10),WALL(48,10),UFAR(48,10),
     &             VFAR(48,10),WFAR(48,10),ENT(48,10)
      COMMON/DISP/ D(5,48,10,10),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH
      COMMON/TIME/ DT(48,10,10),ALPHA1,ALPHA2,ALPHA3
      REAL MACH
      DIMENSION SC(50), RC(50), RHS(50)
      PI = 4.*ATAN(1.)
C
C---  First, read in the position, strength, and core size of the
C     streamwise vortex
C
      READ (7,1000) YVOR,ZVOR,CIRC
      READ (7,1000) CORE
      AIND = CIRC/(4.*PI*YVOR*SQRT(GAM)*MACH)
      COSA = COS(AOA-AIND)
      SINA = SIN(AOA-AIND)
C
C---  compute the entropy in the vortex core assuming a Lamb vortex;
C     spline fit the entropy distribution
C
      NSP = 41
      CALL CORENT(SC,RHS,RC,NSP,CORE,CIRC)
      CALL SPLINE(SC,RHS,RC,NSP)
      IF (ISTART. NE .1) GO TO 140
C
C---  New run:  calculate the prescribed solution state vector for all
C     the interior cells
C
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            XC = .125*(X(I,J,K) + X(I+1,J,K) + X(I,J+1,K) + X(I+1,J+1,K)
     &       + X(I,J,K+1) + X(I+1,J,K+1) + X(I,J+1,K+1) + X(I+1,J+1,K+1))
            YC = .125*(Y(I,J,K) + Y(I+1,J,K) + Y(I,J+1,K) + Y(I+1,J+1,K)
     &       + Y(I,J,K+1) + Y(I+1,J,K+1) + Y(I,J+1,K+1) + Y(I+1,J+1,K+1))
            ZC = .125*(Z(I,J,K) + Z(I+1,J,K) + Z(I,J+1,K) + Z(I+1,J+1,K)
     &       + Z(I,J,K+1) + Z(I+1,J,K+1) + Z(I,J+1,K+1) + Z(I+1,J+1,K+1))
            YCR = YC + YVOR
            YC = YC - YVOR
            ZC = -XC*SINA + ZC*COSA - ZVOR
            RVOR = SQRT(YC**2 + ZC**2)
            RVORR = SQRT(YCR**2 + ZC**2)
C
C---  calculate the induced velocity field of the vortex and it image
C     for each cell
```

```
C
            UINDR = -CIRC/(2.*PI*RVORR)*(1. - EXP(-(RVORR/CORE)**2))
            ROCSQ = (RVOR/CORE)**2
            IF (ROCSQ. GT ..0001) THEN
             UIND = CIRC/(2.*PI*RVOR)*(1. - EXP(-ROCSQ))
             U(2,I,J,K) = -SINA*(UIND*YC/RVOR + UINDR*YCR/RVORR) + UIN
             U(3,I,J,K) = -UIND*ZC/RVOR - UINDR*ZC/RVORR
             U(4,I,J,K) =  COSA*(UIND*YC/RVOR + UINDR*YCR/RVORR) + WIN
            ELSE
             UIND = CIRC/(2.*PI*CORE)*(1. - (ROCSQ/2.)*(1. - (ROCSQ/3.)*
     &        (1. - ROCSQ/4.)))
             U(2,I,J,K) = -SINA*(UIND*YC/CORE + UINDR*YCR/RVORR) + UIN
             U(3,I,J,K) = -UIND*ZC/CORE - UINDR*ZC/RVORR
             U(4,I,J,K) =  COSA*(UIND*YC/CORE + UINDR*YCR/RVORR) + WIN
            END IF
C
C---  find the entropy at the cell center
C
            IF (RVOR. LE .(4.*CORE)) THEN
             RR = RVOR/CORE
             SS = SEVAL(RR,SC,RHS,RC,NSP)
             S = EXP(SS)
            ELSE
             S = 1.
            END IF
C
C---  find the density and pressure at each cell assuming homenthalpic
C     flow; determine the state vector at each cell
C
            POR = (GAM - 1.)/GAM*(HIN - .5*(U(2,I,J,K)**2 +
     &       U(3,I,J,K)**2 + U(4,I,J,K)**2))
            RHO = (POR/S)**(1./(GAM - 1.))
            PR = RHO**GAM*S
            U(1,I,J,K) = RHO
            U(2,I,J,K) = RHO*U(2,I,J,K)
            U(3,I,J,K) = RHO*U(3,I,J,K)
            U(4,I,J,K) = RHO*U(4,I,J,K)
            U(5,I,J,K) = RHO*HIN - PR
            P(I,J,K) = PR
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C
C---  determine the far field boundary values of the velocity and
C     entropy
C
      DO 50 K = 1,KM
        DO 40 I = 1,IM
          XC = .25*(X(I,JMAX,K) + X(I+1,JMAX,K) + X(I,JMAX,K+1) +
     &     X(I+1,JMAX,K+1))
          YC = .25*(Y(I,JMAX,K) + Y(I+1,JMAX,K) + Y(I,JMAX,K+1) +
     &     Y(I+1,JMAX,K+1))
          ZC = .25*(Z(I,JMAX,K) + Z(I+1,JMAX,K) + Z(I,JMAX,K+1) +
     &     Z(I+1,JMAX,K+1))
          YCR = YC + YVOR
          YC = YC - YVOR
          ZC = -XC*SINA +ZC*COSA - ZVOR
          RVOR = SQRT(YC**2 + ZC**2)
          RVORR = SQRT(YCR**2 + ZC**2)
          UINDR = -CIRC/(2.*PI*RVORR)*(1. - EXP(-(RVORR/CORE)**2))
          ROCSQ = (RVOR/CORE)**2
```

```
          IF (ROCSQ. GT ..0001) THEN
           UIND = CIRC/(2.*PI*RVOR)*(1. - EXP(-ROCSQ))
           UFAR(I,K) = -SINA*(UIND*YC/RVOR + UINDR*YCR/RVORR)
           VFAR(I,K) = -UIND*ZC/RVOR - UINDR*ZC/RVORR
           WFAR(I,K) =  COSA*(UIND*YC/RVOR + UINDR*YCR/RVORR)
          ELSE
           UIND = CIRC/(2.*PI*CORE)*(1. - (ROCSQ/2.)*(1. - (ROCSQ/3.)*
     &      (1. - ROCSQ/4.)))
           UFAR(I,K) = -SINA*(UIND*YC/CORE + UINDR*YCR/RVORR)
           VFAR(I,K) = -UIND*ZC/CORE - UINDR*ZC/RVORR
           WFAR(I,K) =  COSA*(UIND*YC/CORE + UINDR*YCR/RVORR)
          END IF
C
C---  find the entropy at the boundary cell face
C
          IF (RVOR. LE .(4.*CORE)) THEN
           RR = RVOR/CORE
           SS = SEVAL(RR,SC,RHS,RC,NSP)
           ENT(I,K) = EXP(SS)
          ELSE
           ENT(I,K) = 1.
          END IF
   40   CONTINUE
   50 CONTINUE
C
C---  calculate fluxes, dissipation terms, and time steps for the
C     perturbation field
C
      CALL DTSIZE(*9999)
      CALL BNDRYC
      CALL FLUX
      CALL DISSIP
C
C---  correct the residuals of the perturbation field at the wing
C     surface boundary
C
      DO 70 K = 1,KM
        DO 60 I = 1,IM
C
C---  first, determine the induced velocities, pressure and density
C     at the wing surface
C
          XC = .25*(X(I,1,K) + X(I+1,1,K) + X(I,1,K+1) + X(I+1,1,K+1))
          YC = .25*(Y(I,1,K) + Y(I+1,1,K) + Y(I,1,K+1) + Y(I+1,1,K+1))
          ZC = .25*(Z(I,1,K) + Z(I+1,1,K) + Z(I,1,K+1) + Z(I+1,1,K+1))
          YCR = YC + YVOR
          YC = YC - YVOR
          ZC = -XC*SINA +ZC*COSA - ZVOR
          RVOR = SQRT(YC**2 + ZC**2)
          RVORR = SQRT(YCR**2 + ZC**2)
          UINDR = -CIRC/(2.*PI*RVORR)*(1. - EXP(-(RVORR/CORE)**2))
          ROCSQ = (RVOR/CORE)**2
          IF (ROCSQ. GT ..0001) THEN
           UIND = CIRC/(2.*PI*RVOR)*(1. - EXP(-ROCSQ))
           UW = -SINA*(UIND*YC/RVOR + UINDR*YCR/RVORR) + UIN
           VW = -UIND*ZC/RVOR - UINDR*ZC/RVORR
           WW =  COSA*(UIND*YC/RVOR + UINDR*YCR/RVORR) + WIN
          ELSE
           UIND = CIRC/(2.*PI*CORE)*(1. - (ROCSQ/2.)*(1. - (ROCSQ/3.)*
     &      (1. - ROCSQ/4.)))
           UW = -SINA*(UIND*YC/CORE + UINDR*YCR/RVORR) + UIN
```

```
            VW = -UIND*ZC/CORE - UINDR*ZC/RVORR
            WW =  COSA*(UIND*YC/CORE + UINDR*YCR/RVORR) + WIN
           END IF
C
C---  find the entropy at the cell face
C
           IF (RVOR. LE .(4.*CORE)) THEN
            RR = RVOR/CORE
            SS = SEVAL(RR,SC,RHS,RC,NSP)
            S = EXP(SS)
           ELSE
            S = 1.
           END IF
           POR = (GAM - 1.)/GAM*(HIN - .5*(UW**2 + VW**2 + WW**2))
           RHO = (POR/S)**(1./(GAM - 1.))
           PR = RHO**GAM*S
           RUW = RHO*UW
           RVW = RHO*VW
           RWW = RHO*WW
C
C---  now calculate the contribution to the fluxes at the solid surface
C
           QWALL = FACEX(2,I,0,K)*UW + FACEY(2,I,0,K)*VW +
     &      FACEZ(2,I,0,K)*WW
           R(1,I,1,K) = R(1,I,1,K) + QWALL*RHO
           R(2,I,1,K) = R(2,I,1,K) + QWALL*RUW + FACEX(2,I,0,K)*
     &      (PR - PRESWG(I,K))
           R(3,I,1,K) = R(3,I,1,K) + QWALL*RVW + FACEY(2,I,0,K)*
     &      (PR - PRESWG(I,K))
           R(4,I,1,K) = R(4,I,1,K) + QWALL*RWW + FACEZ(2,I,0,K)*
     &      (PR - PRESWG(I,K))
           R(5,I,1,K) = R(5,I,1,K) + QWALL*RHO*HIN
   60    CONTINUE
   70 CONTINUE
C
C---  set values in RO array
C
      DO 100 K = 1,KM
        DO 90 J = 1,JM
          DO 80 I = 1,IM
            RO(1,I,J,K) = 0.
            RO(2,I,J,K) = 0.
            RO(3,I,J,K) = 0.
            RO(4,I,J,K) = 0.
            RO(5,I,J,K) = 0.
   80     CONTINUE
   90    CONTINUE
  100 CONTINUE
      DO 103 K = 1,KM
        DO 102 J = 3,JM
          DO 101 I = 19,30
            RO(1,I,J,K) = R(1,I,J,K) + CFL/DT(I,J,K)*D(1,I,J,K)
            RO(2,I,J,K) = R(2,I,J,K) + CFL/DT(I,J,K)*D(2,I,J,K)
            RO(3,I,J,K) = R(3,I,J,K) + CFL/DT(I,J,K)*D(3,I,J,K)
            RO(4,I,J,K) = R(4,I,J,K) + CFL/DT(I,J,K)*D(4,I,J,K)
            RO(5,I,J,K) = R(5,I,J,K) + CFL/DT(I,J,K)*D(5,I,J,K)
  101     CONTINUE
  102   CONTINUE
  103 CONTINUE
C
C---  initialize the state vector to free stream conditions
```

```
C
      DO 130 K = 1,KM
        DO 120 J = 1,JM
          DO 110 I = 1,IM
            U(1,I,J,K) = 1.
            U(2,I,J,K) = UIN
            U(3,I,J,K) = 0.
            U(4,I,J,K) = WIN
            U(5,I,J,K) = HIN - 1.
  110     CONTINUE
  120   CONTINUE
  130 CONTINUE
C
C---  write out perturbation field RO to a restart file and return
C     to main routine
C
      OPEN (9,STATUS='NEW',FORM='UNFORMATTED',FILE='PERTURB.DMP')
      WRITE (9) IM,JM,KM
      WRITE (9) ((((RO(N,I,J,K), N = 1,5), I = 1,IM), J = 1,JM),
     &                K = 1,KM)
      CLOSE (9)
      RETURN
C
C---  if code is to be restarted, read in old solution
C
  140 CONTINUE
      OPEN (8,STATUS='OLD',FORM='UNFORMATTED',FILE='RESTART.DMP')
      READ (8) I,J,K
      IF (I. NE .IM. OR .J. NE .JMAX. OR .K. NE .KM) THEN
       CLOSE (8)
       RETURN 1
      ELSE
       READ (8) ((((U(N,I,J,K), N = 1,5), I = 1,IM),
     &                J = 1,JMAX), K = 1,KM)
       CLOSE (8)
      END IF
      OPEN (9,STATUS='OLD',FORM='UNFORMATTED',FILE='PERTURB.DMP')
      READ (9) I,J,K
      IF (I. NE .IM. OR .J. NE .JM. OR .K. NE .KM) THEN
       CLOSE (9)
       RETURN 1
      ELSE
       READ (9) ((((RO(N,I,J,K), N = 1,5), I = 1,IM),
     &                J = 1,JM), K = 1,KM)
       CLOSE (9)
      END IF
C
C---  determine the far field boundary values of the velocity
C
      DO 160 K = 1,KM
        DO 150 I = 1,IM
          XC = .25*(X(I,JMAX,K) + X(I+1,JMAX,K) + X(I,JMAX,K+1) +
     &     X(I+1,JMAX,K+1))
          YC = .25*(Y(I,JMAX,K) + Y(I+1,JMAX,K) + Y(I,JMAX,K+1) +
     &     Y(I+1,JMAX,K+1))
          ZC = .25*(Z(I,JMAX,K) + Z(I+1,JMAX,K) + Z(I,JMAX,K+1) +
     &     Z(I+1,JMAX,K+1))
          YCR = YC + YVOR
          YC = YC - YVOR
          ZC = -XC*SINA + ZC*COSA - ZVOR
          RVOR = SQRT(YC**2 + ZC**2)
```

```
          RVORR = SQRT(YCR**2 + ZC**2)
          UINDR = -CIRC/(2.*PI*RVORR)*(1. - EXP(-(RVORR/CORE)**2))
          ROCSQ = (RVOR/CORE)**2
          IF (ROCSQ. GT ..0001) THEN
           UIND = CIRC/(2.*PI*RVOR)*(1. - EXP(-ROCSQ))
           UFAR(I,K) = -SINA*(UIND*YC/RVOR + UINDR*YCR/RVORR)
           VFAR(I,K) = -UIND*ZC/RVOR - UINDR*ZC/RVORR
           WFAR(I,K) =  COSA*(UIND*YC/RVOR + UINDR*YCR/RVORR)
          ELSE
           UIND = CIRC/(2.*PI*CORE)*(1. - (ROCSQ/2.)*(1. - (ROCSQ/3.)*
     &      (1. - ROCSQ/4.)))
           UFAR(I,K) = -SINA*(UIND*YC/CORE + UINDR*YCR/RVORR)
           VFAR(I,K) = -UIND*ZC/CORE - UINDR*ZC/RVORR
           WFAR(I,K) =  COSA*(UIND*YC/CORE + UINDR*YCR/RVORR)
          END IF
C
C---  find the entropy at the boundary cell face
C
          IF (RVOR. LE .(4.*CORE)) THEN
           RR = RVOR/CORE
           SS = SEVAL(RR,SC,RHS,RC,NSP)
           ENT(I,K) = EXP(SS)
          ELSE
           ENT(I,K) = 1.
          END IF
  150   CONTINUE
  160 CONTINUE
C
 1000 FORMAT(3G10.5)
      RETURN
 9999 WRITE (6,*) ' Error in perturbation field.'
      RETURN 2
      END
```

```
C
C
C  SUBROUTINE CORENT:  computes the entropy distribution through the
C  core of the prescribed vortex.  A Lamb vortex core structure is
C  assumed in order to get the velocity and vorticity distribution
C  and Crocco's relation is integrated numerically to obtain the
C  entropy in the core.
C
      SUBROUTINE CORENT(SC,RHS,RC,NSP,CORE,CIRC)
      COMMON/PARA/ MACH,GAM,UIN,WIN,HIN,CFL,AENTH
      DIMENSION SC(1),RHS(1),RC(1)
      REAL MACH, KAPPA
      PI = 4.*ATAN(1.)
      KAPPA = CIRC/PI
      CORESQ = CORE**2
      DR = 4.*CORE/FLOAT(NSP-1)
      R = 0.
      FL = 0.
      RC(1) = 0.
      SC(1) = 0.
      DO 10 N = 2,NSP
        R = R + DR
        EXPR = EXP(-(R/CORE)**2)
        UT = KAPPA/(2.*R)*(1. - EXPR)
        OM = KAPPA/CORESQ*EXPR
        T = 1./(GAM - 1.) - UT**2/(2.*GAM)
        FR = -UT*OM/T
        SC(N) = SC(N-1) + (FL + FR)*DR/2.
        RC(N) = R/CORE
        FL = FR
   10 CONTINUE
      SINF = SC(NSP)
      DO 20 N = 1,NSP
        SC(N) = SC(N) - SINF
   20 CONTINUE
      RETURN
      END
```

```
C*************************************************C
C                                                 C
C   Spline routine swiped from...                 C
C                  ... M. Giles and M. Drela      C
C                                                 C
C*************************************************C
      SUBROUTINE SPLINE(X,XP,S,II)
C
      DIMENSION X(1),XP(1),S(1)
      DIMENSION A(480),B(480),C(480)
C
      IF(II.GT.480) STOP 'SPLINE: Array overflow'
      DO 1 I = 2, II-1
C
        IO = I
        IM = I - 1
        IP = I + 1
        DSM = S(IO) - S(IM)
        DSP = S(IP) - S(IO)
C
        B(IO) = DSP
        A(IO) = 2. * ( DSM + DSP )
        C(IO) = DSM
        XP(IO) = 3.*(DSM*(X(IP)-X(IO))/DSP + DSP*(X(IO)-X(IM))/DSM)
C
  1   CONTINUE
C
      A(1) = 2.
      C(1) = 1.
      XP(1) = 3. * (X(2)-X(1)) / (S(2)-S(1))
      B(II) = 1.
      A(II) = 2.
      XP(II) = 3. * (X(II)-X(II-1)) / (S(II)-S(II-1))
C
      CALL TRISOL(A,B,C,XP,II)
C
      RETURN
      END ! SPLINE
C
C
C
      SUBROUTINE TRISOL(A,B,C,D,KK)
C
      DIMENSION A(1),B(1),C(1),D(1)
C
      DO 1 K = 2, KK
        KM = K - 1
        C(KM) = C(KM) / A(KM)
        D(KM) = D(KM) / A(KM)
        A(K) = A(K) - B(K) * C(KM)
        D(K) = D(K) - B(K) * D(KM)
  1   CONTINUE
C
      D(KK) = D(KK) / A(KK)
C
      DO 2 K = KK-1, 1, -1
        D(K) = D(K) - C(K) * D(K+1)
  2   CONTINUE
C
      RETURN
      END ! TRISOL
```

```
C
C
C
      FUNCTION SEVAL(SS,X,XP,S,N)
      REAL X(1), XP(1), S(1)
C
      ILOW = 1
      I = N
C
   10 IF(I-ILOW .LE. 1) GO TO 11
C
      IMID = (I+ILOW)/2
      IF(SS .LT. S(IMID)) THEN
       I = IMID
      ELSE
       ILOW = IMID
      ENDIF
      GO TO 10
C
   11 DS = S(I) - S(I-1)
      T = (SS-S(I-1)) / DS
      CX1 = DS*XP(I-1) - X(I) + X(I-1)
      CX2 = DS*XP(I)   - X(I) + X(I-1)
      SEVAL = T*X(I) + (1.0-T)*X(I-1) + (T-T*T)*((1.0-T)*CX1 - T*CX2)
      RETURN
      END ! SEVAL
C
C
C
      FUNCTION DEVAL(SS,X,XP,S,N)
      REAL X(1), XP(1), S(1)
C
      ILOW = 1
      I = N
C
   10 IF(I-ILOW .LE. 1) GO TO 11
C
      IMID = (I+ILOW)/2
      IF(SS .LT. S(IMID)) THEN
       I = IMID
      ELSE
       ILOW = IMID
      ENDIF
      GO TO 10
C
   11 DS = S(I) - S(I-1)
      T = (SS-S(I-1)) / DS
      CX1 = DS*XP(I-1) - X(I) + X(I-1)
      CX2 = DS*XP(I)   - X(I) + X(I-1)
      DEVAL = (X(I)-X(I-1) + (1.-4.*T+3.*T*T)*CX1 + T*(3.*T-2.)*CX2)/DS
      RETURN
      END ! DEVAL
```

C.3 Rotary wing code

```
      PROGRAM ROTOR
C*************************************************************************
C                                                                        *
C             PROGRAM ROTOR:  Tom Roberts, March 1986                    *
C                                                                        *
C   A program to solve the Euler equations of motion for an inviscid,   *
C   ideal gas in three dimensions around a helicopter rotor in hover.    *
C   The Euler solver is coupled to a fast free wake code based on the    *
C   model of Miller.  The Euler solver uses Jameson's four-stage         *
C   finite volume algorithm.                                           *
C                                                                        *
C   This code uses the Buning & Steger perturbation scheme to include    *
C   the influence of that portion of the vortex wake that passes         *
C   through the Euler finite volume grid.  The vortex positions are      *
C   found from the free wake code.  After updating the vortex wake       *
C   geometry, the residuals of the prescribed flow field are computed.   *
C                                                                        *
C*************************************************************************
      COMMON/VAR / U(5,-1:50,-1:12,-1:18),P(-1:50,-1:12,-1:18),
     &             R(5,49,11,17),IMAX,JMAX,KMAX,IM,JM,KM,NCELLS,
     &             RO(5,48,10,16),X(3,49,11,17)
      COMMON/CELL/ FACEI(3,48,0:10,16),FACEJ(3,48,0:10,16),
     &             FACEK(3,48,0:10,16),VOLUME(48,0:10,16)
      COMMON/BOUN/ PRESWG(48,16),WALL(48,10),WUN(3,0:49,0:17),
     &             QFAR(4,48,16)
      COMMON/DISP/ D(5,48,10,16),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      COMMON/TIME/ DT(0:49,0:11,0:17),ALPHA1,ALPHA2,ALPHA3
      COMMON/AREA/ AM(3,48,10,16)
      COMMON/FWAK/ RV(50),ZV(50),DRV(45),DZV(45),URV(50),UZV(50),
     &             GAMV(50),CORE,NWAKE,NVORT,URP,RCYLT,RCYLI,
     &             ZCYLT,ZCYLI,DGDZT,DGDZI,NSTART,NEND,ITWAKE,IRL,IRU
      COMMON/ELCO/ A11,A12,B11,B12,C11,C12,A21,A22,B21,B22,C21,C22
      COMMON/BGAM/ ROR(0:17),THET(0:17),BOUN(0:17),OMCIRC
      REAL KAP2, KAP4, MACH
C
      OPEN (1,STATUS='OLD',FORM='UNFORMATTED',FILE='ROTOR.BIN')
      OPEN (2,STATUS='NEW',FORM='UNFORMATTED',FILE='RESIDROT.PLT')
      OPEN (3,STATUS='NEW',FORM='UNFORMATTED',FILE='PRESSROT.PLT')
      OPEN (4,STATUS='NEW',FORM='UNFORMATTED',FILE='ROTSOL.DMP')
      OPEN (7,STATUS='NEW',FILE='WAKE.PLT')
      CLOSE (4)
      CLOSE (7)
C
C---  initialize all necessary parameters
C
      PI = 4.*ATAN(1.)
      A11 = .25
      A12 = 1.
      B11 = 9./64.
      B12 = A12 + 1./6.
      C11 = 225./2304.
      C12 = B12 + 1./15.
C
      A21 = .5
      A22 = .5
      B21 = 3./16.
      B22 = A22 + .5 + 1./12.
      C21 = 45./384.
```

```
      C22 = B22 + 1./12. + 1./30.
C
C---  read in the grid; compute the grid metrics
C
      READ (1) IMAX,JMAX,KMAX
      IM = IMAX - 1
      JM = JMAX - 1
      KM = KMAX - 1
      ILE = (IMAX - 1)/2
      NCELLS = IM*JM*KM
      READ (1) (((X(1,I,J,K), I = 1,IMAX), J = 1,JMAX), K = 1,KMAX),
     &         (((X(2,I,J,K), I = 1,IMAX), J = 1,JMAX), K = 1,KMAX),
     &         (((X(3,I,J,K), I = 1,IMAX), J = 1,JMAX), K = 1,KMAX)
      CLOSE (1)
      CALL CELL
      CALL NORMAL
      CALL AMEAN
C
C---  read the input parameters
C
      OPEN (1,STATUS='OLD',FILE='INPUT.ROT')
      READ (1,1000) MACH,GAM,ASRA         ! tip Mach, spec. heat ratio
      READ (1,1000) ALPHA1,ALPHA2,ALPHA3 ! coefs. for time-stepping
      READ (1,1000) KAP2,KAP4,CFL         ! art. visc. and CFL no.
      READ (1,1000) AENTH                 ! rothalpy damping coef.
      READ (1,1001) ITMAX,ITPRIN,ITER     ! max. no. of iterations
      READ (1,1001) NVORT,NWAKE,NEND      ! vortices per wake, # of wakes
      READ (1,1000) (RV(N), ZV(N), GAMV(N), N = 1,NVORT*NWAKE)
      READ (1,1000) RCYLI,ZCYLI,DGDZI     ! initial position of the far
      READ (1,1000) RCYLT,ZCYLT,DGDZT     ! wake vortex cylinders
      READ (1,1000) URP,OMCIRC,CORE       ! free wake relaxation para.
      READ (1,1001) ITEULW,ITWAKE         ! wake iteration limits
      READ (1,1001) IRL,IRU
      READ (1,1000) DCLDA,THET75,TWIST
      READ (1,1000) SIGMA
      CLOSE (1)
      NSTART = NVORT + 1
      DO 10 N = 1,NVORT*NWAKE
        RV(N) = ASRA*RV(N)
        ZV(N) = ASRA*ZV(N)
        GAMV(N) = SQRT(GAM)*MACH*ASRA*GAMV(N)
   10 CONTINUE
      DO 20 N = 1,NVORT
        GAMV(N) = .5*GAMV(N)
   20 CONTINUE
      RCYLI = ASRA*RCYLI
      ZCYLI = ASRA*ZCYLI
      DGDZI = SQRT(GAM)*MACH*ASRA*DGDZI
      RCYLT = ASRA*RCYLT
      ZCYLT = ASRA*ZCYLT
      DGDZT = SQRT(GAM)*MACH*ASRA*DGDZT
C
      WRITE (2) ITER
      AOA = 0.                    ! dummy variable
      WRITE (2) MACH,AOA,CFL,IM,JM,KM
      CLOSE (2)
      WRITE (3) MACH,AOA,CFL,IM,JM,KM
      CALL SURFCO
      CLOSE (3)
C
C---  compute the rotational velocity, rothalpy
```

```
C
      OMEGA = SQRT(GAM)*MACH/ASRA
      HIN = GAM/(GAM - 1.)
C
C---  initialize the ROR and THET arrays
C
      ROR(0) = 0.
      THET(0) = 0.
      DO 30 K = 1,KM
        ROR(K) = .5*(X(2,ILE+1,1,K) + X(2,ILE+1,1,K+1))
        THET(K) = ACOS(1. - 2.*ROR(K)/ASRA)
   30 CONTINUE
      ROR(KMAX) = ASRA
      THET(KMAX) = PI
C
C---  initialize the bound circulation and the state vector
C
      CALL BCINIT(DCLDA,SIGMA,THET75,TWIST)
      CALL PERT(ITER)
      ITCOUP = 1
C
C---  start Euler solution procedure;  establish boundary conditions,
C     dissipation, determine initial value of residuals, and determine
C     size of local time step
C
   31 CALL BNDRYC
      CALL DTSIZE
      CALL FLUX
      CALL DISSIP
C
C---  call four-stage time stepping procedure
C
      CALL TIMSTP
C
      IF (MOD(ITER,ITPRIN). EQ .0. OR .ITER. EQ .ITMAX) THEN
       CALL BNDRYC
       OPEN (3,STATUS='OLD',FORM='UNFORMATTED',FILE='PRESSROT.PLT',
     &  ACCESS='APPEND')
       CALL PROUT(ITER)
       CLOSE (3)
       OPEN (4,STATUS='OLD',FORM='UNFORMATTED',FILE='ROTSOL.DMP')
       WRITE (4) IM,JMAX,KM
       WRITE (4) ((((U(N,I,J,K), N = 1,5), I = 1,IM), J = 1,JMAX),
     &                  K = 1,KM)
       CLOSE (4)
      END IF
      IF (ITER. EQ .ITMAX) GO TO 40
C
C---  every ITEULW iterations, start the free wake iteration procedure
C
      IF (MOD(ITER,ITEULW). EQ .0) THEN
       WRITE (6,1002) ITCOUP,ITER
       CALL BCIRC(ITCOUP)
       CALL FRWAKE
       CALL PERT(ITER)
       IF (ITCOUP. EQ .1) THEN
        ITEULW = 100
       END IF
       ITCOUP = ITCOUP + 1
      END IF
C
```

```
      ITER = ITER + 1
      GO TO 31
   40 CONTINUE
      WRITE (6,1003) MACH,CFL,IM,JM,KM,ITMAX
C
 1000 FORMAT(3G10.5)
 1001 FORMAT(3I5)
 1002 FORMAT(/1X,' Itcoup:',I5,5X,'Iteration:',I5)
 1003 FORMAT(1X,'ROTOR solution     '//1X,'Mach no.:',3X,F6.3/1X,
     & 'Courant no.:',F6.3,6X,'Grid size:',I4,' x',I4,' x',I4//1X,
     & 'Iterations:',I5/)
C
 9999 STOP
      END ! ROTOR
```

```
C
C
C  SUBROUTINE BNDRYC:  sets the far-field and solid wall boundary
C  conditions for Jameson's explicit four-stage scheme.  At the
C  rotor surface, the pressure is extrapolated from interior values;
C  far-field conditions are determined by assuming one-dimensional
C  flow normal to the boundary, and using Riemann invariants in
C  the normal direction.
C
      SUBROUTINE BNDRYC
      COMMON/VAR / U(5,-1:50,-1:12,-1:18),P(-1:50,-1:12,-1:18),
     &             R(5,49,11,17),IMAX,JMAX,KMAX,IM,JM,KM,NCELLS,
     &             RO(5,48,10,16),X(3,49,11,17)
      COMMON/CELL/ FACEI(3,48,0:10,16),FACEJ(3,48,0:10,16),
     &             FACEK(3,48,0:10,16),VOLUME(48,0:10,16)
      COMMON/BOUN/ PRESWG(48,16),WALL(48,10),WUN(3,0:49,0:17),
     &             QFAR(4,48,16)
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      REAL MACH
C
C---  first, determine the pressure field
C
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            UR = -OMEGA*(X(2,I,J,K) + X(2,I+1,J,K) + X(2,I,J+1,K) +
     &       X(2,I+1,J+1,K) + X(2,I,J,K+1) + X(2,I+1,J,K+1) +
     &       X(2,I,J+1,K+1) + X(2,I+1,J+1,K+1))*.125
            VR = OMEGA*(X(1,I,J,K) + X(1,I+1,J,K) + X(1,I,J+1,K) +
     &       X(1,I+1,J+1,K) + X(1,I,J,K+1) + X(1,I+1,J,K+1) +
     &       X(1,I,J+1,K+1) + X(1,I+1,J+1,K+1))*.125
            U1 = U(2,I,J,K)/U(1,I,J,K)
            V1 = U(3,I,J,K)/U(1,I,J,K)
            W1 = U(4,I,J,K)/U(1,I,J,K)
            P(I,J,K) = (U(5,I,J,K) - (.5*(U1**2 + V1**2 + W1**2) -
     &       UR*U1 - VR*V1)*U(1,I,J,K))*(GAM - 1.)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      II = ILE + 1
      DO 31 I = 1,ILE
        P(I,1,0) = P(II-I,1,1)
        P(I,1,KMAX) = P(IMAX-I,1,KMAX-1)
   31 CONTINUE
      II = IMAX + ILE
      DO 32 I = ILE+1,IM
        P(I,1,0) = P(II-I,1,1)
        P(I,1,KMAX) = P(IMAX-I,1,KMAX-1)
   32 CONTINUE
C
C---  set pressure at dummy j=1 cells across the i-coordinate branch cut
C
      DO 34 K = 1,KM
        P(0,1,K) = P(IM,1,K)
        P(IMAX,1,K) = P(1,1,K)
   34 CONTINUE
C
C---  next, determine the far-field conditions
C
      DO 50 K = 1,KM
        DO 40 I = 1,IM
```

```
C
C---  find the components of the rotational velocity at the far-field
C     boundary
C
          UR = -OMEGA*(X(2,I,JMAX,K) + X(2,I+1,JMAX,K) +
     &     X(2,I,JMAX,K+1) + X(2,I+1,JMAX,K+1))*.25
          VR = OMEGA*(X(1,I,JMAX,K) + X(1,I+1,JMAX,K) +
     &     X(1,I,JMAX,K+1) + X(1,I+1,JMAX,K+1))*.25
C
C---  find density, velocity, and pressure at the last point
C     inside the computational domain, (i,J-1,k)
C
          RHOEX = U(1,I,JM,K)
          UEX   = U(2,I,JM,K)/RHOEX - UR
          VEX   = U(3,I,JM,K)/RHOEX - VR
          WEX   = U(4,I,JM,K)/RHOEX
          PEX   =   P(I,JM,K)
C
C---  find the outward unit normal to the computational domain
C     and determine the normal component of the free stream and
C     extrapolated velocities and sonic speeds
C
          ON  = SQRT(FACEJ(1,I,JM,K)**2 + FACEJ(2,I,JM,K)**2 +
     &     FACEJ(3,I,JM,K)**2)
          ONX = FACEJ(1,I,JM,K)/ON
          ONY = FACEJ(2,I,JM,K)/ON
          ONZ = FACEJ(3,I,JM,K)/ON
C
          UFF = -UR + QFAR(1,I,K)
          VFF = -VR + QFAR(2,I,K)
          WFF =       QFAR(3,I,K)
          UINN = ONX*UFF + ONY*VFF + ONZ*WFF
          UEXN = ONX*UEX + ONY*VEX + ONZ*WEX
          CEX = SQRT(GAM*PEX/RHOEX)
          CIN = SQRT((GAM - 1.)*(HIN - .5*(UFF**2 + VFF**2 + WFF**2 -
     &     UR**2 - VR**2)))
          EV4 = UINN + CIN
          EV5 = UEXN - CEX
C
C---  compute the incoming and outgoing Riemann invariants
C
          REX = UEXN + 2.*CEX/(GAM - 1.)
          RIN = UINN - 2.*CIN/(GAM - 1.)
C
C---  supersonic inflow:  all variables specified
C
          IF (EV4. LT .0.) REX = UINN + 2.*CIN/(GAM - 1.)
C
C---  supersonic outflow:  all variables extrapolated
C
          IF (EV5. GE .0.) RIN = UEXN - 2.*CEX/(GAM - 1.)
C
C---  determine the normal velocity and the sonic speed at the boundary
C     cell (i,J,k)
C
          UN = (REX + RIN)*.5
          C  = (REX - RIN)*(GAM - 1.)*.25
          IF (UN. GT .0.) THEN
C
C---  outflow boundary:  extrapolate tangential velocity, entropy
C
```

```
            UP = UEX + (UN - UEXN)*ONX
            VP = VEX + (UN - UEXN)*ONY
            WP = WEX + (UN - UEXN)*ONZ
            S = PEX/RHOEX**GAM
          ELSE
C
C---  inflow boundary:  free stream tangential velocity, entropy
C
            UP = UFF + (UN - UINN)*ONX
            VP = VFF + (UN - UINN)*ONY
            WP = WFF + (UN - UINN)*ONZ
            S = QFAR(4,I,K)
          END IF
C
C---  determine density, energy;  set conservation variables at boundary
C     cell
C
          ENER = (UP**2 + VP**2 + WP**2 - UR**2 - VR**2)*.5 +
     &     C**2/(GAM*(GAM - 1.))
          RHO = (C**2/(GAM*S))**(1./(GAM - 1.))
C
          U(1,I,JMAX,K) = RHO
          U(2,I,JMAX,K) = RHO*(UP + UR)
          U(3,I,JMAX,K) = RHO*(VP + VR)
          U(4,I,JMAX,K) = RHO*WP
          U(5,I,JMAX,K) = RHO*ENER
          P(I,JMAX,K) = RHO*C**2/GAM
   40   CONTINUE
   50 CONTINUE
C
C---  determine the pressure at the wing by extrapolation from the
C     interior--the normal momentum equation is used to determine the
C     normal pressure gradient at the wing surface
C
C
      DO 70 K = 1,KM
        DO 60 I = 1,IM
C
C---  find the components of the rotational velocity at the solid
C     boundary
C
          UR = -OMEGA*(X(2,I,1,K) + X(2,I+1,1,K) + X(2,I,1,K+1) +
     &     X(2,I+1,1,K+1))*.25
          VR = OMEGA*(X(1,I,1,K) + X(1,I+1,1,K) + X(1,I,1,K+1) +
     &     X(1,I+1,1,K+1))*.25
C
C---  determine the velocity and density in cell (i,1,k)
C
          RHO = U(1,I,1,K)
          U1  = U(2,I,1,K)/RHO - UR
          V1  = U(3,I,1,K)/RHO - VR
          W1  = U(4,I,1,K)/RHO
C
C---  determine the cartesian components of the surface velocity
C
          UN = WUN(1,I,K)*U1 + WUN(2,I,K)*V1 + WUN(3,I,K)*W1
          US = U1 - UN*WUN(1,I,K)
          VS = V1 - UN*WUN(2,I,K)
          WS = W1 - UN*WUN(3,I,K)
C
C---  find the contravariant components of the velocity
```

```
C
          UC = US*FACEI(1,I,0,K) + VS*FACEI(2,I,0,K) +
     &     WS*FACEI(3,I,0,K)
          VC = US*FACEK(1,I,0,K) + VS*FACEK(2,I,0,K) +
     &     WS*FACEK(3,I,0,K)
C
          UC = UC/VOLUME(I,0,K)
          VC = VC/VOLUME(I,0,K)
C
C---  determine dn/dxi, dn/deta
C
          DNXDXI = .5*(WUN(1,I+1,K) - WUN(1,I-1,K))
          DNYDXI = .5*(WUN(2,I+1,K) - WUN(2,I-1,K))
          DNZDXI = .5*(WUN(3,I+1,K) - WUN(3,I-1,K))
C
          DNXDET = .5*(WUN(1,I,K+1) - WUN(1,I,K-1))
          DNYDET = .5*(WUN(2,I,K+1) - WUN(2,I,K-1))
          DNZDET = .5*(WUN(3,I,K+1) - WUN(3,I,K-1))
C
C---  solve for the l.h.s. of the normal momentum equation
C
          RHOUU = RHO*(US*(UC*DNXDXI + VC*DNXDET) +
     &                 VS*(UC*DNYDXI + VC*DNYDET) +
     &                 WS*(UC*DNZDXI + VC*DNZDET))
C
C---  add the centrifugal and Coriolis terms to the momentum equation
C
          RHOUU = RHOUU + RHO*OMEGA*(WUN(1,I,K)*(2.*VS + VR) -
     &     WUN(2,I,K)*(2.*US + UR))
C
C---  find the contravariant components of the unit normal
C
          ONXI = WUN(1,I,K)*FACEI(1,I,0,K) + WUN(2,I,K)*FACEI(2,I,0,K)
     &     + WUN(3,I,K)*FACEI(3,I,0,K)
          ONET = WUN(1,I,K)*FACEK(1,I,0,K) + WUN(2,I,K)*FACEK(2,I,0,K)
     &     + WUN(3,I,K)*FACEK(3,I,0,K)
          ONZE = WUN(1,I,K)*FACEJ(1,I,0,K) + WUN(2,I,K)*FACEJ(2,I,0,K)
     &     + WUN(3,I,K)*FACEJ(3,I,0,K)
C
          ONXI = ONXI/VOLUME(I,0,K)
          ONET = ONET/VOLUME(I,0,K)
          ONZE = ONZE/VOLUME(I,0,K)
C
C---  determine the surface pressure gradient
C
          DPDXI = .5*(P(I+1,1,K) - P(I-1,1,K))
          DPDET = .5*(P(I,1,K+1) - P(I,1,K-1))
C
C---  solve for dP/dzeta
C
          DPDZE = (RHOUU - ONXI*DPDXI - ONET*DPDET)/ONZE
C
C---  find the surface pressure
C
          PRESWG(I,K) = P(I,1,K) - .5*DPDZE
   60   CONTINUE
   70 CONTINUE
      RETURN
      END ! BNDRYC
```

```
C
C
C  SUBROUTINE FLUX:  determine the fluxes across each finite volume cell
C  face, and the residual R for each cell.  This is done by averaging the
C  flux vectors between neighboring cells
C
      SUBROUTINE FLUX
      COMMON/VAR / U(5,-1:50,-1:12,-1:18),P(-1:50,-1:12,-1:18),
     &             R(5,49,11,17),IMAX,JMAX,KMAX,IM,JM,KM,NCELLS,
     &             RO(5,48,10,16),X(3,49,11,17)
      COMMON/CELL/ FACEI(3,48,0:10,16),FACEJ(3,48,0:10,16),
     &             FACEK(3,48,0:10,16),VOLUME(48,0:10,16)
      COMMON/BOUN/ PRESWG(48,16),WALL(48,10),WUN(3,0:49,0:17),
     &             QFAR(4,48,16)
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      REAL MACH
C
C---  initialize the residual field
C
      DO 30 K = 1,KMAX
        DO 20 J = 1,JMAX
          DO 10 I = 1,IMAX
            R(1,I,J,K) = 0.
            R(2,I,J,K) = 0.
            R(3,I,J,K) = 0.
            R(4,I,J,K) = 0.
            R(5,I,J,K) = 0.
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C
C---  set-up dummy points
C
      DO 50 K = 1,KM
        DO 40 J = 1,JM
          U(1,IMAX,J,K) = U(1,1,J,K)
          U(2,IMAX,J,K) = U(2,1,J,K)
          U(3,IMAX,J,K) = U(3,1,J,K)
          U(4,IMAX,J,K) = U(4,1,J,K)
          U(5,IMAX,J,K) = U(5,1,J,K)
          P(IMAX,J,K)   =   P(1,J,K)
   40   CONTINUE
   50 CONTINUE
      DO 70 J = 1,JM
        DO 60 I = 1,IMAX
          U(1,I,J,KMAX) = U(1,IMAX-I,J,KMAX-1)
          U(2,I,J,KMAX) = U(2,IMAX-I,J,KMAX-1)
          U(3,I,J,KMAX) = U(3,IMAX-I,J,KMAX-1)
          U(4,I,J,KMAX) = U(4,IMAX-I,J,KMAX-1)
          U(5,I,J,KMAX) = U(5,IMAX-I,J,KMAX-1)
          P(I,J,KMAX)   =   P(IMAX-I,J,KMAX-1)
   60   CONTINUE
   70 CONTINUE
C
C---  compute the residuals at the periodic boundary at the root of the
C     rotor
C
      DO 100 J = 1,JM
        II = ILE + 1
        DO 30 I = 1,ILE
          VR = OMEGA*(X(1,I,J,1) + X(1,I+1,J,1) + X(1,I,J+1,1) +
```

```
     &     X(1,I+1,J+1,1))*.25
           QA = -(U(3,I,J,1)/U(1,I,J,1) - VR)*WALL(I,J)
           H1 = U(1,I,J,1)*QA
           H2 = U(2,I,J,1)*QA
           H3 = U(3,I,J,1)*QA - P(I,J,1)*WALL(I,J)
           H4 = U(4,I,J,1)*QA
           H5 = (U(5,I,J,1) + P(I,J,1))*QA
C
           QAP = (U(3,II-I,J,1)/U(1,II-I,J,1) + VR)*WALL(I,J)
           H1P =  U(1,II-I,J,1)*QAP
           H2P = -U(2,II-I,J,1)*QAP
           H3P = -U(3,II-I,J,1)*QAP - P(II-I,J,1)*WALL(I,J)
           H4P =  U(4,II-I,J,1)*QAP
           H5P = (U(5,II-I,J,1) + P(II-I,J,1))*QAP
C
           R(1,I,J,1) = -(H1 + H1P)*.5
           R(2,I,J,1) = -(H2 + H2P)*.5
           R(3,I,J,1) = -(H3 + H3P)*.5
           R(4,I,J,1) = -(H4 + H4P)*.5
           R(5,I,J,1) = -(H5 + H5P)*.5
   80    CONTINUE
         II = IMAX + ILE
         DO 90 I = ILE+1,IM
           VR = OMEGA*(X(1,I,J,1) + X(1,I+1,J,1) + X(1,I,J+1,1) +
     &     X(1,I+1,J+1,1))*.25
           QA = -(U(3,I,J,1)/U(1,I,J,1) - VR)*WALL(I,J)
           H1 = U(1,I,J,1)*QA
           H2 = U(2,I,J,1)*QA
           H3 = U(3,I,J,1)*QA - P(I,J,1)*WALL(I,J)
           H4 = U(4,I,J,1)*QA
           H5 = (U(5,I,J,1) + P(I,J,1))*QA
C
           QAP = (U(3,II-I,J,1)/U(1,II-I,J,1) + VR)*WALL(I,J)
           H1P =  U(1,II-I,J,1)*QAP
           H2P = -U(2,II-I,J,1)*QAP
           H3P = -U(3,II-I,J,1)*QAP - P(II-I,J,1)*WALL(I,J)
           H4P =  U(4,II-I,J,1)*QAP
           H5P = (U(5,II-I,J,1) + P(II-I,J,1))*QAP
C
           R(1,I,J,1) = -(H1 + H1P)*.5
           R(2,I,J,1) = -(H2 + H2P)*.5
           R(3,I,J,1) = -(H3 + H3P)*.5
           R(4,I,J,1) = -(H4 + H4P)*.5
           R(5,I,J,1) = -(H5 + H5P)*.5
   90    CONTINUE
  100  CONTINUE
C
C---   add contribution due to pressures at the wing surface
C
       DO 120 K = 1,KM
         DO 110 I = 1,IM
           R(2,I,1,K) = R(2,I,1,K) + FACEJ(1,I,0,K)*PRESWG(I,K)
           R(3,I,1,K) = R(3,I,1,K) + FACEJ(2,I,0,K)*PRESWG(I,K)
           R(4,I,1,K) = R(4,I,1,K) + FACEJ(3,I,0,K)*PRESWG(I,K)
  110    CONTINUE
  120  CONTINUE
C
C---   compute the residuals for the interior cells
C
       DO 170 K = 1,KM
         DO 160 J = 1,JM
```

```
      DO 130 I = 1,IM
C
C---  first do the flux across the i-face; compute the component of
C     velocity due to the coordinate rotation
C
      UR = -OMEGA*(X(2,I+1,J,K) + X(2,I+1,J+1,K) + X(2,I+1,J,K+1)
     &  + X(2,I+1,J+1,K+1))*FACEI(1,I,J,K)*.25
      VR = OMEGA*(X(1,I+1,J,K) + X(1,I+1,J+1,K) + X(1,I+1,J,K+1) +
     &  X(1,I+1,J+1,K+1))*FACEI(2,I,J,K)*.25
C
C---  find the flux vector for cells (i,j,k) and (i+1,j,k)
C
      QA = (U(2,I,J,K)*FACEI(1,I,J,K) + U(3,I,J,K)*
     &  FACEI(2,I,J,K) + U(4,I,J,K)*FACEI(3,I,J,K))/U(1,I,J,K)
     &  - UR - VR
      H1 = U(1,I,J,K)*QA
      H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEI(1,I,J,K)
      H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEI(2,I,J,K)
      H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEI(3,I,J,K)
      H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
      QAP = (U(2,I+1,J,K)*FACEI(1,I,J,K) + U(3,I+1,J,K)*
     &  FACEI(2,I,J,K) + U(4,I+1,J,K)*FACEI(3,I,J,K))/U(1,I+1,J,K)
     &  - UR - VR
      H1P = U(1,I+1,J,K)*QAP
      H2P = U(2,I+1,J,K)*QAP + P(I+1,J,K)*FACEI(1,I,J,K)
      H3P = U(3,I+1,J,K)*QAP + P(I+1,J,K)*FACEI(2,I,J,K)
      H4P = U(4,I+1,J,K)*QAP + P(I+1,J,K)*FACEI(3,I,J,K)
      H5P = (U(5,I+1,J,K) + P(I+1,J,K))*QAP
C
C---  find the average flux vector between cells (i,j,k) and (i+1,j,k)
C
      DR1 = .5*(H1 + H1P)
      DR2 = .5*(H2 + H2P)
      DR3 = .5*(H3 + H3P)
      DR4 = .5*(H4 + H4P)
      DR5 = .5*(H5 + H5P)
C
C---  subtract outgoing flux from cell (i,j,k) and add it to
C     cell (i+1,j,k)
C
      R(1,I+1,J,K) = R(1,I+1,J,K) + DR1
      R(2,I+1,J,K) = R(2,I+1,J,K) + DR2
      R(3,I+1,J,K) = R(3,I+1,J,K) + DR3
      R(4,I+1,J,K) = R(4,I+1,J,K) + DR4
      R(5,I+1,J,K) = R(5,I+1,J,K) + DR5
C
      R(1,I,J,K) = R(1,I,J,K) - DR1
      R(2,I,J,K) = R(2,I,J,K) - DR2
      R(3,I,J,K) = R(3,I,J,K) - DR3
      R(4,I,J,K) = R(4,I,J,K) - DR4
      R(5,I,J,K) = R(5,I,J,K) - DR5
  130 CONTINUE
      DO 140 I = 1,IM
C
C---  next do the flux across the j-face; compute the component of
C     velocity due to the coordinate rotation
C
      UR = -OMEGA*(X(2,I,J+1,K) + X(2,I+1,J+1,K) + X(2,I,J+1,K+1)
     &  + X(2,I+1,J+1,K+1))*FACEJ(1,I,J,K)*.25
      VR = OMEGA*(X(1,I,J+1,K) + X(1,I+1,J+1,K) + X(1,I,J+1,K+1) +
```

```
     &       X(1,I+1,J+1,K+1))*FACEJ(2,I,J,K)*.25
C
C---  find the flux vector for cells (i,j,k) and (i,j+1,k)
C
            QA = (U(2,I,J,K)*FACEJ(1,I,J,K) + U(3,I,J,K)*
     &       FACEJ(2,I,J,K) + U(4,I,J,K)*FACEJ(3,I,J,K))/U(1,I,J,K)
     &       - UR - VR
            H1 = U(1,I,J,K)*QA
            H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEJ(1,I,J,K)
            H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEJ(2,I,J,K)
            H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEJ(3,I,J,K)
            H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
            QAP = (U(2,I,J+1,K)*FACEJ(1,I,J,K) + U(3,I,J+1,K)*
     &       FACEJ(2,I,J,K) + U(4,I,J+1,K)*FACEJ(3,I,J,K))/U(1,I,J+1,K)
     &       - UR - VR
            H1P = U(1,I,J+1,K)*QAP
            H2P = U(2,I,J+1,K)*QAP + P(I,J+1,K)*FACEJ(1,I,J,K)
            H3P = U(3,I,J+1,K)*QAP + P(I,J+1,K)*FACEJ(2,I,J,K)
            H4P = U(4,I,J+1,K)*QAP + P(I,J+1,K)*FACEJ(3,I,J,K)
            H5P = (U(5,I,J+1,K) + P(I,J+1,K))*QAP
C
C---  find the average flux vector between cells (i,j,k) and (i,j+1,k)
C
            DR1 = .5*(H1 + H1P)
            DR2 = .5*(H2 + H2P)
            DR3 = .5*(H3 + H3P)
            DR4 = .5*(H4 + H4P)
            DR5 = .5*(H5 + H5P)
C
C---  subtract outgoing flux from cell (i,j,k) and add it to
C     cell (i,j+1,k)
C
            R(1,I,J,K) = R(1,I,J,K) - DR1
            R(2,I,J,K) = R(2,I,J,K) - DR2
            R(3,I,J,K) = R(3,I,J,K) - DR3
            R(4,I,J,K) = R(4,I,J,K) - DR4
            R(5,I,J,K) = R(5,I,J,K) - DR5
C
            R(1,I,J+1,K) = R(1,I,J+1,K) + DR1
            R(2,I,J+1,K) = R(2,I,J+1,K) + DR2
            R(3,I,J+1,K) = R(3,I,J+1,K) + DR3
            R(4,I,J+1,K) = R(4,I,J+1,K) + DR4
            R(5,I,J+1,K) = R(5,I,J+1,K) + DR5
  140     CONTINUE
          DO 150 I = 1,IM
C
C---  finally do the flux across the k-face
C
            UR = -OMEGA*(X(2,I,J,K+1) + X(2,I+1,J,K+1) + X(2,I,J+1,K+1)
     &       + X(2,I+1,J+1,K+1))*FACEK(1,I,J,K)*.25
            VR = OMEGA*(X(1,I,J,K+1) + X(1,I+1,J,K+1) + X(1,I,J+1,K+1) +
     &       X(1,I+1,J+1,K+1))*FACEK(2,I,J,K)*.25
C
C---  find the flux vector for cells (i,j,k) and (i,j,k+1)
C
            QA = (U(2,I,J,K)*FACEK(1,I,J,K) + U(3,I,J,K)*
     &       FACEK(2,I,J,K) + U(4,I,J,K)*FACEK(3,I,J,K))/U(1,I,J,K)
     &       - UR - VR
            H1 = U(1,I,J,K)*QA
            H2 = U(2,I,J,K)*QA + P(I,J,K)*FACEK(1,I,J,K)
```

```
            H3 = U(3,I,J,K)*QA + P(I,J,K)*FACEK(2,I,J,K)
            H4 = U(4,I,J,K)*QA + P(I,J,K)*FACEK(3,I,J,K)
            H5 = (U(5,I,J,K) + P(I,J,K))*QA
C
            QAP = (U(2,I,J,K+1)*FACEK(1,I,J,K) + U(3,I,J,K+1)*
     &       FACEK(2,I,J,K) + U(4,I,J,K+1)*FACEK(3,I,J,K))/U(1,I,J,K+1)
     &       - UR - VR
            H1P = U(1,I,J,K+1)*QAP
            H2P = U(2,I,J,K+1)*QAP + P(I,J,K+1)*FACEK(1,I,J,K)
            H3P = U(3,I,J,K+1)*QAP + P(I,J,K+1)*FACEK(2,I,J,K)
            H4P = U(4,I,J,K+1)*QAP + P(I,J,K+1)*FACEK(3,I,J,K)
            H5P = (U(5,I,J,K+1) + P(I,J,K+1))*QAP
C
C---  find the average flux vector between cells (i,j,k) and (i,j,k+1)
C
            DR1 = .5*(H1 + H1P)
            DR2 = .5*(H2 + H2P)
            DR3 = .5*(H3 + H3P)
            DR4 = .5*(H4 + H4P)
            DR5 = .5*(H5 + H5P)
C
C---  subtract outgoing flux from cell (i,j,k) and add it to
C     cell (i,j,k+1)
C
            R(1,I,J,K) = R(1,I,J,K) - DR1
            R(2,I,J,K) = R(2,I,J,K) - DR2
            R(3,I,J,K) = R(3,I,J,K) - DR3
            R(4,I,J,K) = R(4,I,J,K) - DR4
            R(5,I,J,K) = R(5,I,J,K) - DR5
C
            R(1,I,J,K+1) = R(1,I,J,K+1) + DR1
            R(2,I,J,K+1) = R(2,I,J,K+1) + DR2
            R(3,I,J,K+1) = R(3,I,J,K+1) + DR3
            R(4,I,J,K+1) = R(4,I,J,K+1) + DR4
            R(5,I,J,K+1) = R(5,I,J,K+1) + DR5
  150     CONTINUE
  160   CONTINUE
  170 CONTINUE
C
      DO 190 K = 1,KM
        DO 180 J = 1,JM
          R(1,1,J,K) = R(1,1,J,K) + R(1,IMAX,J,K)
          R(2,1,J,K) = R(2,1,J,K) + R(2,IMAX,J,K)
          R(3,1,J,K) = R(3,1,J,K) + R(3,IMAX,J,K)
          R(4,1,J,K) = R(4,1,J,K) + R(4,IMAX,J,K)
          R(5,1,J,K) = R(5,1,J,K) + R(5,IMAX,J,K)
  180   CONTINUE
  190 CONTINUE
C
C---  add "source terms" to momentum equations
C
      DO 220 K = 1,KM
        DO 210 J = 1,JM
          DO 200 I = 1,IM
            R(2,I,J,K) = R(2,I,J,K) + OMEGA*U(3,I,J,K)*VOLUME(I,J,K)
            R(3,I,J,K) = R(3,I,J,K) - OMEGA*U(2,I,J,K)*VOLUME(I,J,K)
  200     CONTINUE
  210   CONTINUE
  220 CONTINUE
      RETURN
      END ! FLUX
```

```
C
C
C  SUBROUTINE DISSIP:  calculates second- and fourth-order
C  dissipation terms in Jameson's explicit, multi-stage Euler
C  scheme
C
      SUBROUTINE DISSIP
      COMMON/VAR / U(5,-1:50,-1:12,-1:18),P(-1:50,-1:12,-1:18),
     &             R(5,49,11,17),IMAX,JMAX,KMAX,IM,JM,KM,NCELLS,
     &             RO(5,48,10,16),X(3,49,11,17)
      COMMON/DISP/ D(5,48,10,16),KAP2,KAP4
      COMMON/TIME/ DT(0:49,0:11,0:17),ALPHA1,ALPHA2,ALPHA3
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      DIMENSION EP2(0:48), EP4(0:48)
      REAL KAP2, KAP4, MACH, NUM, NUP, NUMM, NUPM
C
C---  The value for the dissipation is found from the operator
C
C                     D  + D  + D  = D(u)
C                      X    Y    Z       ijk
C
C     as defined by Jameson, yielding second- and fourth-order
C     differences.  The operator in the X,Y,Z (I,J,K) coordinate
C     directions are determined in order.  Eriksson's treatment of the
C     smoothing at the solid wall and far-field boundaries is used.
C
C---  set-up dummy points at the periodic boundary
C
      DO 30 J = 1,JM
      II = ILE + 1
        DO 10 I = 1,ILE
          U(1,I,J,0) =  U(1,II-I,J,1)
          U(2,I,J,0) = -U(2,II-I,J,1)
          U(3,I,J,0) = -U(3,II-I,J,1)
          U(4,I,J,0) =  U(4,II-I,J,1)
          U(5,I,J,0) =  U(5,II-I,J,1)
          P(I,J,0)   =    P(II-I,J,1)
C
          U(1,I,J,-1) =  U(1,II-I,J,2)
          U(2,I,J,-1) = -U(2,II-I,J,2)
          U(3,I,J,-1) = -U(3,II-I,J,2)
          U(4,I,J,-1) =  U(4,II-I,J,2)
          U(5,I,J,-1) =  U(5,II-I,J,2)
          P(I,J,-1)   =    P(II-I,J,2)
C
          U(1,I,J,KMAX) = U(1,IMAX-I,J,KMAX-1)
          U(2,I,J,KMAX) = U(2,IMAX-I,J,KMAX-1)
          U(3,I,J,KMAX) = U(3,IMAX-I,J,KMAX-1)
          U(4,I,J,KMAX) = U(4,IMAX-I,J,KMAX-1)
          U(5,I,J,KMAX) = U(5,IMAX-I,J,KMAX-1)
          P(I,J,KMAX)   =   P(IMAX-I,J,KMAX-1)
C
          U(1,I,J,KMAX+1) = U(1,IMAX-I,J,KMAX-2)
          U(2,I,J,KMAX+1) = U(2,IMAX-I,J,KMAX-2)
          U(3,I,J,KMAX+1) = U(3,IMAX-I,J,KMAX-2)
          U(4,I,J,KMAX+1) = U(4,IMAX-I,J,KMAX-2)
          U(5,I,J,KMAX+1) = U(5,IMAX-I,J,KMAX-2)
          P(I,J,KMAX+1)   =   P(IMAX-I,J,KMAX-2)
C
          DT(I,J,0) = DT(II-I,J,1)
          DT(I,J,KMAX) = DT(IMAX-I,J,KMAX-1)
```

```
   10   CONTINUE
        II = IMAX + ILE
        DO 20 I = ILE+1,IM
          U(1,I,J,0) =  U(1,II-I,J,1)
          U(2,I,J,0) = -U(2,II-I,J,1)
          U(3,I,J,0) = -U(3,II-I,J,1)
          U(4,I,J,0) =  U(4,II-I,J,1)
          U(5,I,J,0) =  U(5,II-I,J,1)
          P(I,J,0)   =    P(II-I,J,1)
C
          U(1,I,J,-1) =  U(1,II-I,J,2)
          U(2,I,J,-1) = -U(2,II-I,J,2)
          U(3,I,J,-1) = -U(3,II-I,J,2)
          U(4,I,J,-1) =  U(4,II-I,J,2)
          U(5,I,J,-1) =  U(5,II-I,J,2)
          P(I,J,-1)   =    P(II-I,J,2)
C
          U(1,I,J,KMAX) = U(1,IMAX-I,J,KMAX-1)
          U(2,I,J,KMAX) = U(2,IMAX-I,J,KMAX-1)
          U(3,I,J,KMAX) = U(3,IMAX-I,J,KMAX-1)
          U(4,I,J,KMAX) = U(4,IMAX-I,J,KMAX-1)
          U(5,I,J,KMAX) = U(5,IMAX-I,J,KMAX-1)
          P(I,J,KMAX)   =   P(IMAX-I,J,KMAX-1)
C
          U(1,I,J,KMAX+1) = U(1,IMAX-I,J,KMAX-2)
          U(2,I,J,KMAX+1) = U(2,IMAX-I,J,KMAX-2)
          U(3,I,J,KMAX+1) = U(3,IMAX-I,J,KMAX-2)
          U(4,I,J,KMAX+1) = U(4,IMAX-I,J,KMAX-2)
          U(5,I,J,KMAX+1) = U(5,IMAX-I,J,KMAX-2)
          P(I,J,KMAX+1)   =   P(IMAX-I,J,KMAX-2)
C
          DT(I,J,0) = DT(II-I,J,1)
          DT(I,J,KMAX) = DT(IMAX-I,J,KMAX-1)
   20   CONTINUE
   30 CONTINUE
C
C--- set-up dummy points at the far field and wing boundaries
C
      DO 50 K = 1,KM
        DO 40 I = 1,IM
          U(1,I,JMAX+1,K) = 3.*U(1,I,JM,K) - 2.*U(1,I,JM-1,K)
          U(2,I,JMAX+1,K) = 3.*U(2,I,JM,K) - 2.*U(2,I,JM-1,K)
          U(3,I,JMAX+1,K) = 3.*U(3,I,JM,K) - 2.*U(3,I,JM-1,K)
          U(4,I,JMAX+1,K) = 3.*U(4,I,JM,K) - 2.*U(4,I,JM-1,K)
          U(5,I,JMAX+1,K) = 3.*U(5,I,JM,K) - 2.*U(5,I,JM-1,K)
          P(I,JMAX+1,K) =     3.*P(I,JM,K) -   2.*P(I,JM-1,K)
C
          U(1,I,JMAX,K) = 2.*U(1,I,JM,K) - U(1,I,JM-1,K)
          U(2,I,JMAX,K) = 2.*U(2,I,JM,K) - U(2,I,JM-1,K)
          U(3,I,JMAX,K) = 2.*U(3,I,JM,K) - U(3,I,JM-1,K)
          U(4,I,JMAX,K) = 2.*U(4,I,JM,K) - U(4,I,JM-1,K)
          U(5,I,JMAX,K) = 2.*U(5,I,JM,K) - U(5,I,JM-1,K)
          P(I,JMAX,K)   =   2.*P(I,JM,K) -   P(I,JM-1,K)
C
          U(1,I,0,K) = 2.*U(1,I,1,K) - U(1,I,2,K)
          U(2,I,0,K) = 2.*U(2,I,1,K) - U(2,I,2,K)
          U(3,I,0,K) = 2.*U(3,I,1,K) - U(3,I,2,K)
          U(4,I,0,K) = 2.*U(4,I,1,K) - U(4,I,2,K)
          U(5,I,0,K) = 2.*U(5,I,1,K) - U(5,I,2,K)
          P(I,0,K) =     2.*P(I,1,K) -   P(I,2,K)
C
```

```
          U(1,I,-1,K) = 3.*U(1,I,1,K) - 2.*U(1,I,2,K)
          U(2,I,-1,K) = 3.*U(2,I,1,K) - 2.*U(2,I,2,K)
          U(3,I,-1,K) = 3.*U(3,I,1,K) - 2.*U(3,I,2,K)
          U(4,I,-1,K) = 3.*U(4,I,1,K) - 2.*U(4,I.2,K)
          U(5,I,-1,K) = 3.*U(5,I,1,K) - 2.*U(5,I,2,K)
          P(I,-1,K) =     3.*P(I,1,K) -   2.*P(I,2,K)
C
          DT(I,JMAX,K) = DT(I,JM,K)
          DT(I,0,K) = DT(I,1,K)
   40   CONTINUE
   50 CONTINUE
C
C---   set-up dummy points across the i-coordinate branch cut
C
       DO 70 K = 1,KM
         DO 60 J = 1,JM
           U(1,IMAX+1,J,K) = U(1,2,J,K)
           U(2,IMAX+1,J,K) = U(2,2,J,K)
           U(3,IMAX+1,J,K) = U(3,2,J,K)
           U(4,IMAX+1,J,K) = U(4,2,J,K)
           U(5,IMAX+1,J,K) = U(5,2,J,K)
           P(IMAX+1,J,K)   =   P(2,J,K)
C
           U(1,IMAX,J,K) = U(1,1,J,K)
           U(2,IMAX,J,K) = U(2,1,J,K)
           U(3,IMAX,J,K) = U(3,1,J,K)
           U(4,IMAX,J,K) = U(4,1,J,K)
           U(5,IMAX,J,K) = U(5,1,J,K)
           P(IMAX,J,K)   =   P(1,J,K)
C
           U(1,0,J,K) = U(1,IM,J,K)
           U(2,0,J,K) = U(2,IM,J,K)
           U(3,0,J,K) = U(3,IM,J,K)
           U(4,0,J,K) = U(4,IM,J,K)
           U(5,0,J,K) = U(5,IM,J,K)
           P(0,J,K)   =   P(IM,J,K)
C
           U(1,-1,J,K) = U(1,IM-1,J,K)
           U(2,-1,J,K) = U(2,IM-1,J,K)
           U(3,-1,J,K) = U(3,IM-1,J,K)
           U(4,-1,J,K) = U(4,IM-1,J,K)
           U(5,-1,J,K) = U(5,IM-1,J,K)
           P(-1,J,K)   =   P(IM-1,J,K)
C
           DT(IMAX,J,K) = DT(1,J,K)
           DT(0,J,K) = DT(IM,J,K)
   60    CONTINUE
   70 CONTINUE
      DO 200 K = 1,KM
        DO 190 J = 1,JM
C
C---   begin with the i-direction:  find the values of the
C      dissipation coefficients
C
           DO 80 I = 0,IM
             NUM = ABS((P(I+1,J,K) - 2.*P(I,J,K) + P(I-1,J,K))/
     &                 (P(I+1,J,K) + 2.*P(I,J,K) + P(I-1,J,K)))
             NUP = ABS((P(I+2,J,K) - 2.*P(I+1,J,K) + P(I,J,K))/
     &                 (P(I+2,J,K) + 2.*P(I+1,J,K) + P(I,J,K)))
             DTH = .5*CFL*(1./DT(I+1,J,K) + 1./DT(I,J,K))
C
```

```
            EP = KAP2 * MAX(NUM,NUP)
            EP2(I) = EP*DTH
            EP4(I) = MAX(0.,(KAP4 - EP))*DTH
   80     CONTINUE
C
C---  calculate dissipation d(i,j,k)
C
          DO 90 I = 1,IM
            DS1 = EP2(I)*(U(1,I+1,J,K) - U(1,I,J,K))
            DF1 = EP4(I)*(U(1,I+2,J,K) - 3.*(U(1,I+1,J,K) -
     &            U(1,I,J,K)) - U(1,I-1,J,K))
            DS2 = EP2(I)*(U(2,I+1,J,K) - U(2,I,J,K))
            DF2 = EP4(I)*(U(2,I+2,J,K) - 3.*(U(2,I+1,J,K) -
     &            U(2,I,J,K)) - U(2,I-1,J,K))
            DS3 = EP2(I)*(U(3,I+1,J,K) - U(3,I,J,K))
            DF3 = EP4(I)*(U(3,I+2,J,K) - 3.*(U(3,I+1,J,K) -
     &            U(3,I,J,K)) - U(3,I-1,J,K))
            DS4 = EP2(I)*(U(4,I+1,J,K) - U(4,I,J,K))
            DF4 = EP4(I)*(U(4,I+2,J,K) - 3.*(U(4,I+1,J,K) -
     &            U(4,I,J,K)) - U(4,I-1,J,K))
            DS5 = EP2(I)*(U(5,I+1,J,K) - U(5,I,J,K) + P(I+1,J,K) -
     &            P(I,J,K))
            DF5 = EP4(I)*(U(5,I+2,J,K) - 3.*(U(5,I+1,J,K) -
     &            U(5,I,J,K)) - U(5,I-1,J,K) + P(I+2,J,K) - 3.*
     &            (P(I+1,J,K) - P(I,J,K)) - P(I-1,J,K))
C
            D(1,I,J,K) = DS1 - DF1
            D(2,I,J,K) = DS2 - DF2
            D(3,I,J,K) = DS3 - DF3
            D(4,I,J,K) = DS4 - DF4
            D(5,I,J,K) = DS5 - DF5
   90     CONTINUE
C
          DO 100 I = 1,IM
            DS1M = EP2(I-1)*(U(1,I,J,K) - U(1,I-1,J,K))
            DF1M = EP4(I-1)*(U(1,I+1,J,K) - 3.*(U(1,I,J,K) -
     &             U(1,I-1,J,K)) - U(1,I-2,J,K))
            DS2M = EP2(I-1)*(U(2,I,J,K) - U(2,I-1,J,K))
            DF2M = EP4(I-1)*(U(2,I+1,J,K) - 3.*(U(2,I,J,K) -
     &             U(2,I-1,J,K)) - U(2,I-2,J,K))
            DS3M = EP2(I-1)*(U(3,I,J,K) - U(3,I-1,J,K))
            DF3M = EP4(I-1)*(U(3,I+1,J,K) - 3.*(U(3,I,J,K) -
     &             U(3,I-1,J,K)) - U(3,I-2,J,K))
            DS4M = EP2(I-1)*(U(4,I,J,K) - U(4,I-1,J,K))
            DF4M = EP4(I-1)*(U(4,I+1,J,K) - 3.*(U(4,I,J,K) -
     &             U(4,I-1,J,K)) - U(4,I-2,J,K))
            DS5M = EP2(I-1)*(U(5,I,J,K) - U(5,I-1,J,K) + P(I,J,K) -
     &             P(I-1,J,K))
            DF5M = EP4(I-1)*(U(5,I+1,J,K) - 3.*(U(5,I,J,K) -
     &             U(5,I-1,J,K)) - U(5,I-2,J,K) + P(I+1,J,K) - 3.*
     &             (P(I,J,K) - P(I-1,J,K)) - P(I-2,J,K))
C
            D(1,I,J,K) = D(1,I,J,K) - DS1M + DF1M
            D(2,I,J,K) = D(2,I,J,K) - DS2M + DF2M
            D(3,I,J,K) = D(3,I,J,K) - DS3M + DF3M
            D(4,I,J,K) = D(4,I,J,K) - DS4M + DF4M
            D(5,I,J,K) = D(5,I,J,K) - DS5M + DF5M
  100     CONTINUE
C
C---  next do the j-direction:  find the values of the
C     dissipation coefficients
```

```
C
      DO 110 I = 1,IM
        NUM = ABS((P(I,J+1,K) - 2.*P(I,J,K) + P(I,J-1,K))/
     &            (P(I,J+1,K) + 2.*P(I,J,K) + P(I,J-1,K)))
        NUP = ABS((P(I,J+2,K) - 2.*P(I,J+1,K) + P(I,J,K))/
     &            (P(I,J+2,K) + 2.*P(I,J+1,K) + P(I,J,K)))
        DTH = .5*CFL*(1./DT(I,J+1,K) + 1./DT(I,J,K))
C
        EPP = KAP2 * MAX(NUM,NUP)
        EP2(I) = EPP*DTH
        EP4(I) = MAX(0.,(KAP4 - EPP))*DTH
  110 CONTINUE
C
C---  calculate dissipation d(i,j,k)
C
      DO 120 I = 1,IM
        DS1 = EP2(I)*(U(1,I,J+1,K) - U(1,I,J,K))
        DF1 = EP4(I)*(U(1,I,J+2,K) - 3.*(U(1,I,J+1,K) -
     &        U(1,I,J,K)) - U(1,I,J-1,K))
        DS2 = EP2(I)*(U(2,I,J+1,K) - U(2,I,J,K))
        DF2 = EP4(I)*(U(2,I,J+2,K) - 3.*(U(2,I,J+1,K) -
     &        U(2,I,J,K)) - U(2,I,J-1,K))
        DS3 = EP2(I)*(U(3,I,J+1,K) - U(3,I,J,K))
        DF3 = EP4(I)*(U(3,I,J+2,K) - 3.*(U(3,I,J+1,K) -
     &        U(3,I,J,K)) - U(3,I,J-1,K))
        DS4 = EP2(I)*(U(4,I,J+1,K) - U(4,I,J,K))
        DF4 = EP4(I)*(U(4,I,J+2,K) - 3.*(U(4,I,J+1,K) -
     &        U(4,I,J,K)) - U(4,I,J-1,K))
        DS5 = EP2(I)*(U(5,I,J+1,K) - U(5,I,J,K) + P(I,J+1,K) -
     &        P(I,J,K))
        DF5 = EP4(I)*(U(5,I,J+2,K) - 3.*(U(5,I,J+1,K) -
     &        U(5,I,J,K)) - U(5,I,J-1,K) + P(I,J+2,K) - 3.*
     &        (P(I,J+1,K) - P(I,J,K)) - P(I,J-1,K))
C
        D(1,I,J,K) = D(1,I,J,K) + DS1 - DF1
        D(2,I,J,K) = D(2,I,J,K) + DS2 - DF2
        D(3,I,J,K) = D(3,I,J,K) + DS3 - DF3
        D(4,I,J,K) = D(4,I,J,K) + DS4 - DF4
        D(5,I,J,K) = D(5,I,J,K) + DS5 - DF5
  120 CONTINUE
C
      DO 130 I = 1,IM
        NUMM = ABS((P(I,J,K) - 2.*P(I,J-1,K) + P(I,J-2,K))/
     &             (P(I,J,K) + 2.*P(I,J-1,K) + P(I,J-2,K)))
        NUPM = ABS((P(I,J+1,K) - 2.*P(I,J,K) + P(I,J-1,K))/
     &             (P(I,J+1,K) + 2.*P(I,J,K) + P(I,J-1,K)))
        DTH = .5*CFL*(1./DT(I,J,K) + 1./DT(I,J-1,K))
C
        EPM = KAP2 * MAX(NUMM,NUPM)
        EP2(I) = EPM*DTH
        EP4(I) = MAX(0.,(KAP4 - EPM))*DTH
  130 CONTINUE
C
      DO 140 I = 1,IM
        DS1M = EP2(I)*(U(1,I,J,K) - U(1,I,J-1,K))
        DF1M = EP4(I)*(U(1,I,J+1,K) - 3.*(U(1,I,J,K) -
     &         U(1,I,J-1,K)) - U(1,I,J-2,K))
        DS2M = EP2(I)*(U(2,I,J,K) - U(2,I,J-1,K))
        DF2M = EP4(I)*(U(2,I,J+1,K) - 3.*(U(2,I,J,K) -
     &         U(2,I,J-1,K)) - U(2,I,J-2,K))
        DS3M = EP2(I)*(U(3,I,J,K) - U(3,I,J-1,K))
```

```
            DF3M = EP4(I)*(U(3,I,J+1,K) - 3.*(U(3,I,J,K) -
     &             U(3,I,J-1,K)) - U(3,I,J-2,K))
            DS4M = EP2(I)*(U(4,I,J,K) - U(4,I,J-1,K))
            DF4M = EP4(I)*(U(4,I,J+1,K) - 3.*(U(4,I,J,K) -
     &             U(4,I,J-1,K)) - U(4,I,J-2,K))
            DS5M = EP2(I)*(U(5,I,J,K) - U(5,I,J-1,K) + P(I,J,K) -
     &             P(I,J-1,K))
            DF5M = EP4(I)*(U(5,I,J+1,K) - 3.*(U(5,I,J,K) -
     &             U(5,I,J-1,K)) - U(5,I,J-2,K) + P(I,J+1,K) - 3.*
     &             (P(I,J,K) - P(I,J-1,K)) - P(I,J-2,K))
C
            D(1,I,J,K) = D(1,I,J,K) - DS1M + DF1M
            D(2,I,J,K) = D(2,I,J,K) - DS2M + DF2M
            D(3,I,J,K) = D(3,I,J,K) - DS3M + DF3M
            D(4,I,J,K) = D(4,I,J,K) - DS4M + DF4M
            D(5,I,J,K) = D(5,I,J,K) - DS5M + DF5M
  140     CONTINUE
C
C---  next do the k-direction:  find the values of the
C     dissipation coefficients
C
          DO 150 I = 1,IM
            NUM = ABS((P(I,J,K+1) - 2.*P(I,J,K) + P(I,J,K-1))/
     &             (P(I,J,K+1) + 2.*P(I,J,K) + P(I,J,K-1)))
            NUP = ABS((P(I,J,K+2) - 2.*P(I,J,K+1) + P(I,J,K))/
     &             (P(I,J,K+2) + 2.*P(I,J,K+1) + P(I,J,K)))
            DTH = .5*CFL*(1./DT(I,J,K+1) + 1./DT(I,J,K))
C
            EPP = KAP2 * MAX(NUM,NUP)
            EP2(I) = EPP*DTH
            EP4(I) = MAX(0.,(KAP4 - EPP))*DTH
  150     CONTINUE
C
C---  calculate dissipation d(i,j,k)
C
          DO 160 I = 1,IM
            DS1 = EP2(I)*(U(1,I,J,K+1) - U(1,I,J,K))
            DF1 = EP4(I)*(U(1,I,J,K+2) - 3.*(U(1,I,J,K+1) -
     &             U(1,I,J,K)) - U(1,I,J,K-1))
            DS2 = EP2(I)*(U(2,I,J,K+1) - U(2,I,J,K))
            DF2 = EP4(I)*(U(2,I,J,K+2) - 3.*(U(2,I,J,K+1) -
     &             U(2,I,J,K)) - U(2,I,J,K-1))
            DS3 = EP2(I)*(U(3,I,J,K+1) - U(3,I,J,K))
            DF3 = EP4(I)*(U(3,I,J,K+2) - 3.*(U(3,I,J,K+1) -
     &             U(3,I,J,K)) - U(3,I,J,K-1))
            DS4 = EP2(I)*(U(4,I,J,K+1) - U(4,I,J,K))
            DF4 = EP4(I)*(U(4,I,J,K+2) - 3.*(U(4,I,J,K+1) -
     &             U(4,I,J,K)) - U(4,I,J,K-1))
            DS5 = EP2(I)*(U(5,I,J,K+1) - U(5,I,J,K) + P(I,J,K+1) -
     &             P(I,J,K))
            DF5 = EP4(I)*(U(5,I,J,K+2) - 3.*(U(5,I,J,K+1) -
     &             U(5,I,J,K)) - U(5,I,J,K-1) + P(I,J,K+2) - 3.*
     &             (P(I,J,K+1) - P(I,J,K)) - P(I,J,K-1))
C
            D(1,I,J,K) = D(1,I,J,K) + DS1 - DF1
            D(2,I,J,K) = D(2,I,J,K) + DS2 - DF2
            D(3,I,J,K) = D(3,I,J,K) + DS3 - DF3
            D(4,I,J,K) = D(4,I,J,K) + DS4 - DF4
            D(5,I,J,K) = D(5,I,J,K) + DS5 - DF5
  160     CONTINUE
C
```

```
          DO 170 I = 1,IM
            NUMM = ABS((P(I,J,K) - 2.*P(I,J,K-1) + P(I,J,K-2))/
     &                 (P(I,J,K) + 2.*P(I,J,K-1) + P(I,J,K-2)))
            NUPM = ABS((P(I,J,K+1) - 2.*P(I,J,K) + P(I,J,K-1))/
     &                 (P(I,J,K+1) + 2.*P(I,J,K) + P(I,J,K-1)))
            DTH = .5*CFL*(1./DT(I,J,K) + 1./DT(I,J,K-1))
C
            EPM = KAP2 * MAX(NUMM,NUPM)
            EP2(I) = EPM*DTH
            EP4(I) = MAX(0.,(KAP4 - EPM))*DTH
  170     CONTINUE
C
          DO 180 I = 1,IM
            DS1M = EP2(I)*(U(1,I,J,K) - U(1,I,J,K-1))
            DF1M = EP4(I)*(U(1,I,J,K+1) - 3.*(U(1,I,J,K) -
     &             U(1,I,J,K-1)) - U(1,I,J,K-2))
            DS2M = EP2(I)*(U(2,I,J,K) - U(2,I,J,K-1))
            DF2M = EP4(I)*(U(2,I,J,K+1) - 3.*(U(2,I,J,K) -
     &             U(2,I,J,K-1)) - U(2,I,J,K-2))
            DS3M = EP2(I)*(U(3,I,J,K) - U(3,I,J,K-1))
            DF3M = EP4(I)*(U(3,I,J,K+1) - 3.*(U(3,I,J,K) -
     &             U(3,I,J,K-1)) - U(3,I,J,K-2))
            DS4M = EP2(I)*(U(4,I,J,K) - U(4,I,J,K-1))
            DF4M = EP4(I)*(U(4,I,J,K+1) - 3.*(U(4,I,J,K) -
     &             U(4,I,J,K-1)) - U(4,I,J,K-2))
            DS5M = EP2(I)*(U(5,I,J,K) - U(5,I,J,K-1) + P(I,J,K) -
     &             P(I,J,K-1))
            DF5M = EP4(I)*(U(5,I,J,K+1) - 3.*(U(5,I,J,K) -
     &             U(5,I,J,K-1)) - U(5,I,J,K-2) + P(I,J,K+1) - 3.*
     &             (P(I,J,K) - P(I,J,K-1)) - P(I,J,K-2))
C
            D(1,I,J,K) = D(1,I,J,K) - DS1M + DF1M
            D(2,I,J,K) = D(2,I,J,K) - DS2M + DF2M
            D(3,I,J,K) = D(3,I,J,K) - DS3M + DF3M
            D(4,I,J,K) = D(4,I,J,K) - DS4M + DF4M
            D(5,I,J,K) = D(5,I,J,K) - DS5M + DF5M
  180     CONTINUE
  190   CONTINUE
  200 CONTINUE
      RETURN
      END ! DISSIP
```

```
C
C
C  SUBROUTINE DTSIZE:  calculates the value of the local time step
C  size based on the local CFL number restriction
C
      SUBROUTINE DTSIZE
      COMMON/VAR / U(5,-1:50,-1:12,-1:18),P(-1:50,-1:12,-1:18),
     &             R(5,49,11,17),IMAX,JMAX,KMAX,IM,JM,KM,NCELLS,
     &             RO(5,48,10,16),X(3,49,11,17)
      COMMON/CELL/ FACEI(3,48,0:10,16),FACEJ(3,48,0:10,16),
     &             FACEK(3,48,0:10,16),VOLUME(48,0:10,16)
      COMMON/DISP/ D(5,48,10,16),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      COMMON/TIME/ DT(0:49,0:11,0:17),ALPHA1,ALPHA2,ALPHA3
      COMMON/AREA/ AM(3,48,10,16)
      REAL KAP2, KAP4, MACH
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 20 I = 1,IM
            UR = -OMEGA*(X(2,I,J,K) + X(2,I+1,J,K) + X(2,I,J+1,K) +
     &       X(2,I+1,J+1,K) + X(2,I,J,K+1) + X(2,I+1,J,K+1) +
     &       X(2,I,J+1,K+1) + X(2,I+1,J+1,K+1))*.125
            VR = OMEGA*(X(1,I,J,K) + X(1,I+1,J,K) + X(1,I,J+1,K) +
     &       X(1,I+1,J+1,K) + X(1,I,J,K+1) + X(1,I+1,J,K+1) +
     &       X(1,I,J+1,K+1) + X(1,I+1,J+1,K+1))*.125
            U1  = U(2,I,J,K)/U(1,I,J,K) - UR
            V1  = U(3,I,J,K)/U(1,I,J,K) - VR
            W1  = U(4,I,J,K)/U(1,I,J,K)
            C = SQRT(GAM*P(I,J,K)/U(1,I,J,K))
C
            AX = AM(1,I,J,K)
            AY = AM(2,I,J,K)
            AZ = AM(3,I,J,K)
            AR = SQRT(AX**2 + AY**2 + AZ**2)
C
            UA = ABS(U1*AX)
            VA = ABS(V1*AY)
            WA = ABS(W1*AZ)
            CA = C*AR
            DT(I,J,K) = CFL/(UA + VA + WA + CA)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      RETURN
      END ! DTSIZE
```

```
C
C
C  SUBROUTINE TIMSTP:  advances solution Euler equations by a four-
C  stage time stepping scheme, using a spacially varying time step
C  and frozen dissipative terms in order to speed convergence
C
      SUBROUTINE TIMSTP
      COMMON/VAR / U(5,-1:50,-1:12,-1:18),P(-1:50,-1:12,-1:18),
     &             R(5,49,11,17),IMAX,JMAX,KMAX,IM,JM,KM,NCELLS,
     &             RO(5,48,10,16),X(3,49,11,17)
      COMMON/CELL/ FACEI(3,48,0:10,16),FACEJ(3,48,0:10,16),
     &             FACEK(3,48,0:10,16),VOLUME(48,0:10,16)
      COMMON/DISP/ D(5,48,10,16),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      COMMON/TIME/ DT(0:49,0:11,0:17),ALPHA1,ALPHA2,ALPHA3
      DIMENSION UTEMP(5,48,10,16)
      REAL KAP2, KAP4, MACH
C
C---  store current state vector in U-0
C
      DO 30 K = 1,KM
        DO 20 J = 1,JM
          DO 10 I = 1,IM
            UTEMP(1,I,J,K) = U(1,I,J,K)
            UTEMP(2,I,J,K) = U(2,I,J,K)
            UTEMP(3,I,J,K) = U(3,I,J,K)
            UTEMP(4,I,J,K) = U(4,I,J,K)
            UTEMP(5,I,J,K) = U(5,I,J,K)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C
C---  first stage
C
      DO 60 K = 1,KM
        DO 50 J = 1,JM
          DO 40 I = 1,IM
            DTLOC = ALPHA1*DT(I,J,K)
            U(1,I,J,K) = UTEMP(1,I,J,K) + DTLOC*(R(1,I,J,K) +
     &        D(1,I,J,K))! - RO(1,I,J,K))
            U(2,I,J,K) = UTEMP(2,I,J,K) + DTLOC*(R(2,I,J,K) +
     &        D(2,I,J,K))! - RO(2,I,J,K))
            U(3,I,J,K) = UTEMP(3,I,J,K) + DTLOC*(R(3,I,J,K) +
     &        D(3,I,J,K))! - RO(3,I,J,K))
            U(4,I,J,K) = UTEMP(4,I,J,K) + DTLOC*(R(4,I,J,K) +
     &        D(4,I,J,K))! - RO(4,I,J,K))
            U(5,I,J,K) = UTEMP(5,I,J,K) + DTLOC*(R(5,I,J,K) +
     &        D(5,I,J,K))! - RO(5,I,J,K))
   40     CONTINUE
   50   CONTINUE
   60 CONTINUE
      CALL BNDRYC
      CALL FLUX
C
C---  second stage
C
      DO 90 K = 1,KM
        DO 80 J = 1,JM
          DO 70 I = 1,IM
            DTLOC = ALPHA2*DT(I,J,K)
            U(1,I,J,K) = UTEMP(1,I,J,K) + DTLOC*(R(1,I,J,K) +
```

```
     &       D(1,I,J,K))! - RO(1,I,J,K))
             U(2,I,J,K) = UTEMP(2,I,J,K) + DTLOC*(R(2,I,J,K) +
     &       D(2,I,J,K))! - RO(2,I,J,K))
             U(3,I,J,K) = UTEMP(3,I,J,K) + DTLOC*(R(3,I,J,K) +
     &       D(3,I,J,K))! - RO(3,I,J,K))
             U(4,I,J,K) = UTEMP(4,I,J,K) + DTLOC*(R(4,I,J,K) +
     &       D(4,I,J,K))! - RO(4,I,J,K))
             U(5,I,J,K) = UTEMP(5,I,J,K) + DTLOC*(R(5,I,J,K) +
     &       D(5,I,J,K))! - RO(5,I,J,K))
   70      CONTINUE
   80    CONTINUE
   90 CONTINUE
      CALL BNDRYC
      CALL FLUX
C
C--- third stage
C
      DO 120 K = 1,KM
        DO 110 J = 1,JM
          DO 100 I = 1,IM
            DTLOC = ALPHA3*DT(I,J,K)
            U(1,I,J,K) = UTEMP(1,I,J,K) + DTLOC*(R(1,I,J,K) +
     &       D(1,I,J,K))! - RO(1,I,J,K))
            U(2,I,J,K) = UTEMP(2,I,J,K) + DTLOC*(R(2,I,J,K) +
     &       D(2,I,J,K))! - RO(2,I,J,K))
            U(3,I,J,K) = UTEMP(3,I,J,K) + DTLOC*(R(3,I,J,K) +
     &       D(3,I,J,K))! - RO(3,I,J,K))
            U(4,I,J,K) = UTEMP(4,I,J,K) + DTLOC*(R(4,I,J,K) +
     &       D(4,I,J,K))! - RO(4,I,J,K))
            U(5,I,J,K) = UTEMP(5,I,J,K) + DTLOC*(R(5,I,J,K) +
     &       D(5,I,J,K))! - RO(5,I,J,K))
  100     CONTINUE
  110   CONTINUE
  120 CONTINUE
      CALL BNDRYC
      CALL FLUX
C
C---  final stage
C
      DO 150 K = 1,KM
        DO 140 J = 1,JM
          DO 130 I = 1,IM
            DTLOC = DT(I,J,K)
            U(1,I,J,K) = UTEMP(1,I,J,K) + DTLOC*(R(1,I,J,K) +
     &       D(1,I,J,K))! - RO(1,I,J,K))
            U(2,I,J,K) = UTEMP(2,I,J,K) + DTLOC*(R(2,I,J,K) +
     &       D(2,I,J,K))! - RO(2,I,J,K))
            U(3,I,J,K) = UTEMP(3,I,J,K) + DTLOC*(R(3,I,J,K) +
     &       D(3,I,J,K))! - RO(3,I,J,K))
            U(4,I,J,K) = UTEMP(4,I,J,K) + DTLOC*(R(4,I,J,K) +
     &       D(4,I,J,K))! - RO(4,I,J,K))
            U(5,I,J,K) = UTEMP(5,I,J,K) + DTLOC*(R(5,I,J,K) +
     &       D(5,I,J,K))! - RO(5,I,J,K))
  130     CONTINUE
  140   CONTINUE
  150 CONTINUE
C
C---  do rothalpy damping and iteration history loop
C
      DO 180 K = 1,KM
        DO 170 J = 1,JM
```

```
            DO 160 I = 1,IM
              UR = -OMEGA*(X(2,I,J,K) + X(2,I+1,J,K) + X(2,I,J+1,K) +
     &        X(2,I+1,J+1,K) + X(2,I,J,K+1) + X(2,I+1,J,K+1) +
     &        X(2,I,J+1,K+1) + X(2,I+1,J+1,K+1))*.125
              VR = OMEGA*(X(1,I,J,K) + X(1,I+1,J,K) + X(1,I,J+1,K) +
     &        X(1,I+1,J+1,K) + X(1,I,J,K+1) + X(1,I+1,J,K+1) +
     &        X(1,I,J+1,K+1) + X(1,I+1,J+1,K+1))*.125
              RHO = U(1,I,J,K)
              U1  = U(2,I,J,K)/RHO
              V1  = U(3,I,J,K)/RHO
              W1  = U(4,I,J,K)/RHO
              EN  = U(5,I,J,K)
              PR  = (EN - (.5*(U1**2 + V1**2 + W1**2) - UR*U1 - VR*V1)*
     &        RHO)*(GAM - 1.)
              HMH = (EN + PR)/RHO - HIN
              H = 1./(1. + AENTH*HMH)
              U(1,I,J,K) = RHO*H
              U(2,I,J,K) = RHO*U1*H
              U(3,I,J,K) = RHO*V1*H
              U(4,I,J,K) = RHO*W1*H
              RH = ((RHO*HMH - PR) - AENTH*PR)/(1. + AENTH)
              U(5,I,J,K) = RH + RHO*H*HIN
  160       CONTINUE
  170     CONTINUE
  180   CONTINUE
C
C---    calculate the "residual", rms(delta-U)
C
        RMSRES = 0.
        DO 210 K = 1,KM
          DO 200 J = 1,JM
            DO 190 I = 1,IM
              RMSRES = RMSRES + (U(1,I,J,K) - UTEMP(1,I,J,K))**2
              RMSRES = RMSRES + (U(2,I,J,K) - UTEMP(2,I,J,K))**2
              RMSRES = RMSRES + (U(3,I,J,K) - UTEMP(3,I,J,K))**2
              RMSRES = RMSRES + (U(4,I,J,K) - UTEMP(4,I,J,K))**2
              RMSRES = RMSRES + (U(5,I,J,K) - UTEMP(5,I,J,K))**2
  190       CONTINUE
  200     CONTINUE
  210   CONTINUE
C
C---    write out the rms delta-u
C
        RMSRES = SQRT(RMSRES/(5.*FLOAT(NCELLS)))
        OPEN (2,STATUS='OLD',FORM='UNFORMATTED',FILE='RESIDROT.PLT',
       &  ACCESS='APPEND')
        WRITE (2) RMSRES
        CLOSE (2)
        RETURN
        END ! TIMSTP
```

```
C
C
C  SUBROUTINE PROUT:  writes surface pressure coefficients to a data
C  file for plotting.  Also prints out sectional and total lift and
C  drag coefficients.
C
      SUBROUTINE PROUT(ITER)
      COMMON/VAR / U(5,-1:50,-1:12,-1:18),P(-1:50,-1:12,-1:18),
     &             R(5,49,11,17),IMAX,JMAX,KMAX,IM,JM,KM,NCELLS,
     &             RO(5,48,10,16),X(3,49,11,17)
      COMMON/CELL/ FACEI(3,48,0:10,16),FACEJ(3,48,0:10,16),
     &             FACEK(3,48,0:10,16),VOLUME(48,0:10,16)
      COMMON/BOUN/ PRESWG(48,16),WALL(48,10),WUN(3,0:49,0:17),
     &             QFAR(4,48,16)
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      COMMON/BGAM/ ROR(0:17),THET(0:17),BOUN(0:17),OMCIRC
      REAL MACH
      WRITE (3) ITER
      WRITE (6,1000) ITER
      S = 0.
      CZT = 0.
      CXT = 0.
      DO 20 K = 1,KM
        CZ = 0.
        CX = 0.
        ASEC = 0.
        DO 10 I = 1,IM
          CP = 2.*(PRESWG(I,K) - 1.)/(GAM*(MACH*ROR(K))**2)*ASRA**2
          WRITE (3) CP
          CZ = CZ - CP*FACEJ(3,I,0,K)
          CX = CX - CP*FACEJ(1,I,0,K)
          IF (I. LE .ILE) ASEC = ASEC + ABS(FACEJ(3,I,0,K))
   10   CONTINUE
        CZT = CZT + CZ
        CXT = CXT + CX
        S = S + ASEC
C
C---  compute sectional Cl, Cd
C
        CL = CZ/ASEC
        CD = CX/ASEC
        WRITE (6,1001) K,CL,CD
   20 CONTINUE
C
C---  compute wing CL, CD
C
      CL = CZT/S
      CD = CXT/S
      WRITE (6,1002) CL,CD
C
 1000 FORMAT(1X,'Iteration:',I5//1X,'span station',8X,'Cl'12X,'Cd'/
     & 1X,12('-'),4X,10('-'),4X,10('-'))
 1001 FORMAT(6X,I2,3X,2F14.4)
 1002 FORMAT(/1X,'wing Cl:',F8.4,4X,'wing Cd:',F8.4)
      RETURN
      END ! PROUT
```

```
C
C
C  SUBROUTINE PERT:  computes the residuals of the prescribed vortex
C  wake flow field.  This subroutine is called to initialize the RO
C  vector, and after each free wake iteration procedure.
C
      SUBROUTINE PERT(ITER)
      COMMON/VAR / U(5,-1:50,-1:12,-1:18),P(-1:50,-1:12,-1:18),
     &             R(5,49,11,17),IMAX,JMAX,KMAX,IM,JM,KM,NCELLS,
     &             RO(5,48,10,16),X(3,49,11,17)
      COMMON/CELL/ FACEI(3,48,0:10,16),FACEJ(3,48,0:10,16),
     &             FACEK(3,48,0:10,16),VOLUME(48,0:10,16)
      COMMON/BOUN/ PRESWG(48,16),WALL(48,10),WUN(3,0:49,0:17),
     &             QFAR(4,48,16)
      COMMON/DISP/ D(5,48,10,16),KAP2,KAP4
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      COMMON/TIME/ DT(0:49,0:11,0:17),ALPHA1,ALPHA2,ALPHA3
      COMMON/FWAK/ RV(50),ZV(50),DRV(45),DZV(45),URV(50),UZV(50),
     &             GAMV(50),CORE,NWAKE,NVORT,URP,RCYLT,RCYLI,
     &             ZCYLT,ZCYLI,DGDZT,DGDZI,NSTART,NEND,ITWAKE,IRL,IRU
      COMMON/ELCO/ A11,A12,B11,B12,C11,C12,A21,A22,B21,B22,C21,C22
      DIMENSION SC(50), RCC(50), RHS(50), UTEMP(5,48,10,16)
C      EQUIVALENCE (UTEMP(1,1,1,1), RO(1,1,1,1))
      REAL KAP2, KAP4, MACH
      NSP = 41
C
C---  see if this is the initial call to PERT.  If it is not, then
C     store the current state vector in temporary storage.  This is so we
C     won't foolishly wipe out the current Euler code solution that we have
C     worked so hard to get.  If this is the first time through, then U
C     contains nothing yet and the prescribed vortex wake flow field will be
C     used as an initial guess.
C
      IF (ITER. NE .1) THEN
       DO 30 K = 1,KM
         DO 20 J = 1,JM
           DO 10 I = 1,IM
             UTEMP(1,I,J,K) = U(1,I,J,K)
             UTEMP(2,I,J,K) = U(2,I,J,K)
             UTEMP(3,I,J,K) = U(3,I,J,K)
             UTEMP(4,I,J,K) = U(4,I,J,K)
             UTEMP(5,I,J,K) = U(5,I,J,K)
   10      CONTINUE
   20    CONTINUE
   30  CONTINUE
      END IF
      IF (IRL. LE .2) THEN
       IL = 1
      ELSE
       IL = IRL - 2
      END IF
      IF (IRU. GE .IM-1) THEN
       IU = IM
      ELSE
       IU = IRU + 2
      END IF
C
C---  initialize the state vector
C
      DO 36 K = 1,KM
        DO 34 J = 1,JM
```

```
            DO 33 I = 1,IM
              U(1,I,J,K) = 1.
              U(2,I,J,K) = 0.
              U(3,I,J,K) = 0.
              U(4,I,J,K) = 0.
              U(5,I,J,K) = HIN - 1.
   33       CONTINUE
   34     CONTINUE
          DO 35 I = 1,IM
            QFAR(1,I,K) = 0.
            QFAR(2,I,K) = 0.
            QFAR(3,I,K) = 0.
            QFAR(4,I,K) = 1.
   35     CONTINUE
   36   CONTINUE
        IF (ITER. EQ .1) RETURN
C
        DO 90 NNN = NSTART,NEND
          IF (NNN. EQ .NSTART) THEN
           DO 39 K = 1,KM
             DO 38 J = 1,JM
               DO 37 I = IL,IU
                 U(5,I,J,K) = 1.
   37          CONTINUE
   38        CONTINUE
   39       CONTINUE
          END IF
C
C
C---    calculate the entropy through the Lamb vortex core and spline
C       fit the distribution
C
          CIRC = GAMV(NNN)
          RVOR = RV(NNN)
          ZVOR = ZV(NNN)
          WSIN = CIRC/(4.*PI*RVOR)*(LOG(8.*RVOR/CORE) - .25)
          CALL CORENT(SC,RHS,RCC,NSP,CORE,CIRC)
          CALL SPLINE(SC,RHS,RCC,NSP)
C
C---    calculate the wake induced velocity field at the interior cells;
C       this does not include the influence of the semi-infinite vortex
C       cylinders in the far wake
C
          DO 60 K = 1,KM
            DO 50 J = 1,JM
              DO 40 I = IL,IU
                XC = (X(1,I,J,K) + X(1,I+1,J,K) + X(1,I,J+1,K) +
     &           X(1,I+1,J+1,K) + X(1,I,J,K+1) + X(1,I+1,J,K+1) +
     &           X(1,I,J+1,K+1) + X(1,I+1,J+1,K+1))*.125
                YC = (X(2,I,J,K) + X(2,I+1,J,K) + X(2,I,J+1,K) +
     &           X(2,I+1,J+1,K) + X(2,I,J,K+1) + X(2,I+1,J,K+1) +
     &           X(2,I,J+1,K+1) + X(2,I+1,J+1,K+1))*.125
                ZC = (X(3,I,J,K) + X(3,I+1,J,K) + X(3,I,J+1,K) +
     &           X(3,I+1,J+1,K) + X(3,I,J,K+1) + X(3,I+1,J,K+1) +
     &           X(3,I,J+1,K+1) + X(3,I+1,J+1,K+1))*.125
                RC = SQRT(YC**2 + XC**2)
                DDRV = SQRT((RC - RVOR)**2 + (ZC - ZVOR)**2)
C
C---    calculate the induced velocity field of the vortex for each cell,
C       assuming a Lamb vortex core structure
C
```

```
              ROC = DDRV/CORE
              ROCSQ = ROC**2
              IF (ROC. LE ..01) THEN
               UIND = CIRC/(2.*PI*CORE)*(1. - (ROCSQ/2.)*(1. - (ROCSQ/
     &          3.)*(1. - ROCSQ/4.)))
               UAXI = -UIND*(RC - RVOR)/CORE + WSIN
               URAD =  UIND*(ZC - ZVOR)/CORE
              ELSE IF (ROC. GT ..01. AND .ROC. LE .2.) THEN
               UIND = CIRC/(2.*PI*DDRV)*(1. - EXP(-ROCSQ))
               UAXI = -UIND*(RC - RVOR)/DDRV + WSIN
               URAD =  UIND*(ZC - ZVOR)/DDRV
              ELSE IF (ROC. GT .2. AND .ROC. LE .4.) THEN
               UIND = CIRC/(2.*PI*DDRV)*(1. - EXP(-ROCSQ))
               UAXI = (-UIND*(RC - RVOR)/DDRV + WSIN)*(4. - ROC)/2.
               URAD = ( UIND*(ZC - ZVOR)/DDRV)*(4. - ROC)/2.
C
C---  in here, compute the elliptic integrals (approximately)
C
               ARG = 4.*RVOR*RC/((RVOR + RC)**2 + (ZVOR - ZC)**2)
               F = LOG(4./SQRT(1. - ARG))
               EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*
     &          (F - B12) + (1. - ARG)*C11*(F - C12)))
               EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*
     &          (F - B22) + (1. - ARG)*C21*(F - C22)))
C
C---  calculate the induced velocities
C
               DZ = ZC - ZVOR
               UAXI = (CIRC/(4.*PI)*SQRT(ARG/RVOR/RC)*(EL1 - EL2*(1. -
     &          .5*ARG*(1. + RVOR/RC))/(1. - ARG)))*(ROC - 2.)/2. + UAXI
               URAD = (CIRC*DZ/(8.*PI*RC)*SQRT(ARG/RC/RVOR)*(EL2*(2. -
     &          ARG)/(1. - ARG) - 2.*EL1))*(ROC - 2.)/2. + URAD
              ELSE
C
C---  in here, compute the elliptic integrals (approximately)
C
               ARG = 4.*RVOR*RC/((RVOR + RC)**2 + (ZVOR - ZC)**2)
               F = LOG(4./SQRT(1. - ARG))
               EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*
     &          (F - B12) + (1. - ARG)*C11*(F - C12)))
               EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*
     &          (F - B22) + (1. - ARG)*C21*(F - C22)))
C
C---  calculate the induced velocities
C
               DZ = ZC - ZVOR
               UAXI = CIRC/(4.*PI)*SQRT(ARG/RVOR/RC)*(EL1 - EL2*(1. -
     &          .5*ARG*(1. + RVOR/RC))/(1. - ARG))
               URAD = CIRC*DZ/(8.*PI*RC)*SQRT(ARG/RC/RVOR)*(EL2*(2. -
     &          ARG)/(1. - ARG) - 2.*EL1)
              END IF
C
C---  determine the entropy at the cell center and store it in U-5
C
              IF (ROC. LE .4.) THEN
               U(5,I,J,K) = SEVAL(ROC,SC,RHS,RCC,NSP)
              END IF
C
C---  store the induced velocities and the entropy in the state vector U
C
              U(2,I,J,K) = U(2,I,J,K) + XC*URAD/RC
```

```
            U(3,I,J,K) = U(3,I,J,K) + YC*URAD/RC
            U(4,I,J,K) = U(4,I,J,K) + UAXI
   40       CONTINUE
   50     CONTINUE
   60   CONTINUE
C
C---  determine the far field boundary values of the velocity and
C     entropy; first, calculate the wake induced velocities, excluding
C     the influence of the semi-infinite vortex cylinders in the far
C     wake
C
        DO 80 K = 1,KM
          DO 70 I = 1,IM
            XC = (X(1,I,JMAX,K) + X(1,I+1,JMAX,K) + X(1,I,JMAX,K+1) +
     &       X(1,I+1,JMAX,K+1))*.25
            YC = (X(2,I,JMAX,K) + X(2,I+1,JMAX,K) + X(2,I,JMAX,K+1) +
     &       X(2,I+1,JMAX,K+1))*.25
            ZC = (X(3,I,JMAX,K) + X(3,I+1,JMAX,K) + X(3,I,JMAX,K+1) +
     &       X(3,I+1,JMAX,K+1))*.25
            RC = SQRT(YC**2 + XC**2)
            DDRV = SQRT((RC - RVOR)**2 + (ZC - ZVOR)**2)
C
C---  calculate the induced velocity field of the vortex for each cell
C     face, assuming a Lamb vortex core structure
C
            ROC = DDRV/CORE
            ROCSQ = ROC**2
            IF (ROC. LE ..01) THEN
             UIND = CIRC/(2.*PI*CORE)*(1. - (ROCSQ/2.)*(1. - (ROCSQ/3.)*
     &         (1. - ROCSQ/4.)))
             UAXI = -UIND*(RC - RVOR)/CORE + WSIN
             URAD =  UIND*(ZC - ZVOR)/CORE
            ELSE IF (ROC. GT ..01. AND .ROC. LE .2.) THEN
             UIND = CIRC/(2.*PI*DDRV)*(1. - EXP(-ROCSQ))
             UAXI = -UIND*(RC - RVOR)/DDRV + WSIN
             URAD =  UIND*(ZC - ZVOR)/DDRV
            ELSE IF (ROC. GT .2. AND .ROC. LE .4.) THEN
             UIND = CIRC/(2.*PI*DDRV)*(1. - EXP(-ROCSQ))
             UAXI = (-UIND*(RC - RVOR)/DDRV + WSIN)*(4. - ROC)/2.
             URAD = ( UIND*(ZC - ZVOR)/DDRV)*(4. - ROC)/2.
C
C---  in here, compute the elliptic integrals (approximately)
C
            ARG = 4.*RVOR*RC/((RVOR + RC)**2 + (ZVOR - ZC)**2)
            F = LOG(4./SQRT(1. - ARG))
            EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*(F -
     &       B12) + (1. - ARG)*C11*(F - C12)))
            EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*(F -
     &       B22) + (1. - ARG)*C21*(F - C22)))
C
C---  calculate the induced velocities
C
            DZ = ZC - ZVOR
            UAXI = (CIRC/(4.*PI)*SQRT(ARG/RVOR/RC)*(EL1 - EL2*(1. - .5*
     &       ARG*(1. + RVOR/RC))/(1. - ARG)))*(ROC - 2.)/2. + UAXI
            URAD = (CIRC*DZ/(8.*PI*RC)*SQRT(ARG/RC/RVOR)*(EL2*(2. -
     &       ARG)/(1. - ARG) - 2.*EL1))*(ROC - 2.)/2. + URAD
           ELSE
C
C---  in here, compute the elliptic integrals (approximately)
C
```

```
            ARG = 4.*RVOR*RC/((RVOR + RC)**2 + (ZVOR - ZC)**2)
            F = LOG(4./SQRT(1. - ARG))
            EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*(F -
     &       B12) + (1. - ARG)*C11*(F - C12)))
            EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*(F -
     &       B22) + (1. - ARG)*C21*(F - C22)))
C
C---  calculate the induced velocities
C
            DZ = ZC - ZVOR
            UAXI = CIRC/(4.*PI)*SQRT(ARG/RVOR/RC)*(EL1 - EL2*(1. - .5*
     &       ARG*(1. + RVOR/RC))/(1. - ARG))
            URAD = CIRC*DZ/(8.*PI*RC)*SQRT(ARG/RC/RVOR)*(EL2*(2. - ARG)
     &       /(1. - ARG) - 2.*EL1)
          END IF
C
C---  determine the entropy at the cell center and store it in Qfar-4
C
          IF (ROC. LE .4.) THEN
           QFAR(4,I,K) = SEVAL(ROC,SC,RHS,RCC,NSP)
          END IF
C
C---  store the induced velocities and the entropy in the state vector U
C
          QFAR(1,I,K) = QFAR(1,I,K) + XC*URAD/RC
          QFAR(2,I,K) = QFAR(2,I,K) + YC*URAD/RC
          QFAR(3,I,K) = QFAR(3,I,K) + UAXI
   70     CONTINUE
   80   CONTINUE
   90 CONTINUE
C
C---  now compute the contribution to the induced velocities at each
C     cell due to the far wake vortex cylinders
C
      DO 130 K = 1,KM
        DO 120 J = 1,JM
          DO 110 I = IL,IU
            XC = (X(1,I,J,K) + X(1,I+1,J,K) + X(1,I,J+1,K) +
     &       X(1,I+1,J+1,K) + X(1,I,J,K+1) + X(1,I+1,J,K+1) +
     &       X(1,I,J+1,K+1) + X(1,I+1,J+1,K+1))*.125
            YC = (X(2,I,J,K) + X(2,I+1,J,K) + X(2,I,J+1,K) +
     &       X(2,I+1,J+1,K) + X(2,I,J,K+1) + X(2,I+1,J,K+1) +
     &       X(2,I,J+1,K+1) + X(2,I+1,J+1,K+1))*.125
            ZC = (X(3,I,J,K) + X(3,I+1,J,K) + X(3,I,J+1,K) +
     &       X(3,I+1,J+1,K) + X(3,I,J,K+1) + X(3,I+1,J,K+1) +
     &       X(3,I,J+1,K+1) + X(3,I+1,J+1,K+1))*.125
            RC = SQRT(YC**2 + XC**2)
            DZI = ZC - ZCYLI
            ARG = 4.*RC*RCYLI/((RC + RCYLI)**2 + DZI**2)
            F = LOG(4./SQRT(1. - ARG))
            EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*(F -
     &       B12) + (1. - ARG)*C11*(F - C12)))
            EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*(F -
     &       B22) + (1. - ARG)*C21*(F - C22)))
            URAD = DGDZI/(2.*PI)*SQRT(RCYLI/ARG/RC)*(EL1*(2. -
     &     ARG) - 2.*EL2)
C
            DZT = ZC - ZCYLT
            ARG = 4.*RC*RCYLT/((RC + RCYLT)**2 + DZT**2)
            F = LOG(4./SQRT(1. - ARG))
            EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*(F -
```

```
     &       B12) + (1. - ARG)*C11*(F - C12)))
             EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*(F -
     &       B22) + (1. - ARG)*C21*(F - C22)))
             URAD = URAD + DGDZT/(2.*PI)*SQRT(RCYLT/ARG/RC)*(EL1*(2. -
     &       ARG) - 2.*EL2)
C
C---  for the axial component of the induced velocity, integrate the
C     expression numerically
C
            DTT = PI/40.
            TR = DTT
            ETA = RC
            RSPESI = RCYLI**2 + ETA**2
            RSPEST = RCYLT**2 + ETA**2
            ZISQ = DZI**2
            ZTSQ = DZT**2
            DENI = RSPESI - 2.*RCYLI*ETA
            DENT = RSPEST - 2.*RCYLT*ETA
            IF (ABS(DENI). LE .1.E-7) THEN
             FRACI = .5
            ELSE
             FRACI = RCYLI*(RCYLI - ETA)/DENI
            END IF
            IF (ABS(DENT). LE .1.E-7) THEN
             FRACT = .5
            ELSE
             FRACT = RCYLT*(RCYLT - ETA)/DENT
            END IF
            FI = (1. - DZI/SQRT(DENI + ZISQ))*FRACI
            FT = (1. - DZT/SQRT(DENT + ZTSQ))*FRACT
            UZ2 = DGDZI*FI + DGDZT*FT
            DO 100 M = 1,79
              COSR = COS(TR)
              DENI = RSPESI - 2.*RCYLI*ETA*COSR
              FRACI = RCYLI*(RCYLI - ETA*COSR)/DENI
              FI = (1. - DZI/SQRT(DENI + ZISQ))*FRACI
              DENT = RSPEST - 2.*RCYLT*ETA*COSR
              FRACT = RCYLT*(RCYLT - ETA*COSR)/DENT
              FT = (1. - DZT/SQRT(DENT + ZTSQ))*FRACT
              UZ2 = UZ2 + DGDZI*FI + DGDZT*FT
              TR = TR + DTT
  100       CONTINUE
            UAXI = UZ2*DTT/(4.*PI)
            U(2,I,J,K) = U(2,I,J,K) + XC*URAD/RC
            U(3,I,J,K) = U(3,I,J,K) + YC*URAD/RC
            U(4,I,J,K) = U(4,I,J,K) + UAXI
  110     CONTINUE
  120   CONTINUE
  130 CONTINUE
C
C---  find the density and pressure at each cell assuming homenthalpic
C     flow; determine the state vector at each cell
C
      DO 160 K = 1,KM
        DO 150 J = 1,JM
          DO 140 I = IL,IU
            S = U(5,I,J,K)
            POR = (GAM - 1.)/GAM*(HIN - .5*(U(2,I,J,K)**2 +
     &       U(3,I,J,K)**2 + U(4,I,J,K)**2))
            RHO = (POR/S)**(1./(GAM - 1.))
            PR = RHO**GAM*S
```

```
            U(1,I,J,K) = RHO
            U(2,I,J,K) = RHO*U(2,I,J,K)
            U(3,I,J,K) = RHO*U(3,I,J,K)
            U(4,I,J,K) = RHO*U(4,I,J,K)
            U(5,I,J,K) = RHO*HIN - PR
  140     CONTINUE
  150   CONTINUE
  160 CONTINUE
C
C---  now compute the contribution to the induced velocities at the far
C     field boundaries from the far wake vortex cylinders
C
      DO 190 K = 1,KM
        DO 180 I = 1,IM
          XC = (X(1,I,JMAX,K) + X(1,I+1,JMAX,K) + X(1,I,JMAX,K+1) +
     &     X(1,I+1,JMAX,K+1))*.25
          YC = (X(2,I,JMAX,K) + X(2,I+1,JMAX,K) + X(2,I,JMAX,K+1) +
     &     X(2,I+1,JMAX,K+1))*.25
          ZC = (X(3,I,JMAX,K) + X(3,I+1,JMAX,K) + X(3,I,JMAX,K+1) +
     &     X(3,I+1,JMAX,K+1))*.25
          RC = SQRT(YC**2 + XC**2)
          DZI = ZC - ZCYLI
          ARG = 4.*RC*RCYLI/((RC + RCYLI)**2 + DZI**2)
          F = LOG(4./SQRT(1. - ARG))
          EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*(F -
     &     B12) + (1. - ARG)*C11*(F - C12)))
          EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*(F -
     &     B22) + (1. - ARG)*C21*(F - C22)))
          URAD = DGDZI/(2.*PI)*SQRT(RCYLI/ARG/RC)*(EL1*(2. -
     &     ARG) - 2.*EL2)
C
          DZT = ZC - ZCYLT
          ARG = 4.*RC*RCYLT/((RC + RCYLT)**2 + DZT**2)
          F = LOG(4./SQRT(1. - ARG))
          EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*(F -
     &     B12) + (1. - ARG)*C11*(F - C12)))
          EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*(F -
     &     B22) + (1. - ARG)*C21*(F - C22)))
          URAD = URAD + DGDZT/(2.*PI)*SQRT(RCYLT/ARG/RC)*(EL1*(2. -
     &     ARG) - 2.*EL2)
C
C---  for the axial component of the induced velocity, integrate the
C     expression numerically
C
          DTT = PI/40.
          TR = DTT
          ETA = RC
          RSPESI = RCYLI**2 + ETA**2
          RSPEST = RCYLT**2 + ETA**2
          ZISQ = DZI**2
          ZTSQ = DZT**2
          DENI = RSPESI - 2.*RCYLI*ETA
          DENT = RSPEST - 2.*RCYLT*ETA
          IF (ABS(DENI). LE .1.E-7) THEN
           FRACI = .5
          ELSE
           FRACI = RCYLI*(RCYLI - ETA)/DENI
          END IF
          IF (ABS(DENT). LE .1.E-7) THEN
           FRACT = .5
          ELSE
```

```
            FRACT = RCYLT*(RCYLT - ETA)/DENT
           END IF
           FI = (1. - DZI/SQRT(DENI + ZISQ))*FRACI
           FT = (1. - DZT/SQRT(DENT + ZTSQ))*FRACT
           UZ2 = DGDZI*FI + DGDZT*FT
           DO 170 M = 1,79
             COSR = COS(TR)
             DENI = RSPESI - 2.*RCYLI*ETA*COSR
             FRACI = RCYLI*(RCYLI - ETA*COSR)/DENI
             FI = (1. - DZI/SQRT(DENI + ZISQ))*FRACI
             DENT = RSPEST - 2.*RCYLT*ETA*COSR
             FRACT = RCYLT*(RCYLT - ETA*COSR)/DENT
             FT = (1. - DZT/SQRT(DENT + ZTSQ))*FRACT
             UZ2 = UZ2 + DGDZI*FI + DGDZT*FT
             TR = TR + DTT
  170      CONTINUE
           UAXI = UZ2*DTT/(4.*PI)
           QFAR(1,I,K) = QFAR(1,I,K) + XC*URAD/RC
           QFAR(2,I,K) = QFAR(2,I,K) + YC*URAD/RC
           QFAR(3,I,K) = QFAR(3,I,K) + UAXI
  180    CONTINUE
  190 CONTINUE
C
C---  calculate the fluxes, dissipation terms, and time steps of the
C     prescribed vortical flow field
C
      CALL BNDRYC
      CALL DTSIZE
      CALL FLUX
      CALL DISSIP
C
C---  copy the original values of the state vector into U
C
  191 CONTINUE
      IF (ITER. NE .1) THEN
       DO 220 K = 1,KM
         DO 210 J = 1,JM
           DO 200 I = 1,IM
             U(1,I,J,K) = UTEMP(1,I,J,K)
             U(2,I,J,K) = UTEMP(2,I,J,K)
             U(3,I,J,K) = UTEMP(3,I,J,K)
             U(4,I,J,K) = UTEMP(4,I,J,K)
             U(5,I,J,K) = UTEMP(5,I,J,K)
  200      CONTINUE
  210    CONTINUE
  220  CONTINUE
      END IF
C
C---  determine set values of RO array
C
      DO 250 K = 1,KM
        DO 240 J = 1,JM
          DO 230 I = 1,IM
            RO(1,I,J,K) = 0.
            RO(2,I,J,K) = 0.
            RO(3,I,J,K) = 0.
            RO(4,I,J,K) = 0.
            RO(5,I,J,K) = 0.
  230     CONTINUE
  240   CONTINUE
  250 CONTINUE
```

```
C      IF (ITER. EQ .1. OR .NSTART. GE .NEND) RETURN
       DO 280 K = 3,KM
          DO 270 J = 3,JM
             DO 260 I = IRL,IRU
                RO(1,I,J,K) = R(1,I,J,K) + D(1,I,J,K)
                RO(2,I,J,K) = R(2,I,J,K) + D(2,I,J,K)
                RO(3,I,J,K) = R(3,I,J,K) + D(3,I,J,K)
                RO(4,I,J,K) = R(4,I,J,K) + D(4,I,J,K)
                RO(5,I,J,K) = R(5,I,J,K) + D(5,I,J,K)
  260        CONTINUE
  270      CONTINUE
  280   CONTINUE
C
C---   to complete the set-up, the contribution of the remaining
C      intermediate wake vortices to the induced velocities at the far
C      field boundary must be computed
C
       DO 310 NNN = NEND+1,NVORT*NWAKE
         CIRC = GAMV(NNN)
         RVOR = RV(NNN)
         ZVOR = ZV(NNN)
         WSIN = CIRC/(4.*PI*RVOR)*(LOG(8.*RVOR/CORE) - .25)
         CALL CORENT(SC,RHS,RCC,NSP,CORE,CIRC)
         CALL SPLINE(SC,RHS,RCC,NSP)
         DO 300 K = 1,KM
           DO 290 I = 1,IM
             XC = (X(1,I,JMAX,K) + X(1,I+1,JMAX,K) + X(1,I,JMAX,K+1) +
     &        X(1,I+1,JMAX,K+1))*.25
             YC = (X(2,I,JMAX,K) + X(2,I+1,JMAX,K) + X(2,I,JMAX,K+1) +
     &        X(2,I+1,JMAX,K+1))*.25
             ZC = (X(3,I,JMAX,K) + X(3,I+1,JMAX,K) + X(3,I,JMAX,K+1) +
     &        X(3,I+1,JMAX,K+1))*.25
             RC = SQRT(YC**2 + XC**2)
             DDRV = SQRT((RC - RVOR)**2 + (ZC - ZVOR)**2)
C
C---   calculate the induced velocity field of the vortex for each cell
C      face, assuming a Lamb vortex core structure
C
             ROC = DDRV/CORE
             ROCSQ = ROC**2
             IF (ROC. LE ..01) THEN
              UIND = CIRC/(2.*PI*CORE)*(1. - (ROCSQ/2.)*(1. - (ROCSQ/3.)*
     &         (1. - ROCSQ/4.)))
              UAXI = -UIND*(RC - RVOR)/CORE + WSIN
              URAD =  UIND*(ZC - ZVOR)/CORE
             ELSE IF (ROC. GT ..01. AND .ROC. LE .2.) THEN
              UIND = CIRC/(2.*PI*DDRV)*(1. - EXP(-ROCSQ))
              UAXI = -UIND*(RC - RVOR)/DDRV + WSIN
              URAD =  UIND*(ZC - ZVOR)/DDRV
             ELSE IF (ROC. GT .2. AND .ROC. LE .4.) THEN
              UIND = CIRC/(2.*PI*DDRV)*(1. - EXP(-ROCSQ))
              UAXI = (-UIND*(RC - RVOR)/DDRV + WSIN)*(4. - ROC)/2.
              URAD = ( UIND*(ZC - ZVOR)/DDRV)*(4. - ROC)/2.
C
C---   in here, compute the elliptic integrals (approximately)
C
              ARG = 4.*RVOR*RC/((RVOR + RC)**2 + (ZVOR - ZC)**2)
              F = LOG(4./SQRT(1. - ARG))
              EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*(F -
     &         B12) + (1. - ARG)*C11*(F - C12)))
              EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*(F -
```

```
     &        B22) + (1. - ARG)*C21*(F - C22)))
C
C---  calculate the induced velocities
C
             DZ = ZC - ZVOR
             UAXI = (CIRC/(4.*PI)*SQRT(ARG/RVOR/RC)*(EL1 - EL2*(1. - .5*
     &        ARG*(1. + RVOR/RC))/(1. - ARG)))*(ROC - 2.)/2. + UAXI
             URAD = (CIRC*DZ/(8.*PI*RC)*SQRT(ARG/RC/RVOR)*(EL2*(2. -
     &        ARG)/(1. - ARG) - 2.*EL1))*(ROC - 2.)/2. + URAD
           ELSE
C
C---  in here, compute the elliptic integrals (approximately)
C
             ARG = 4.*RVOR*RC/((RVOR + RC)**2 + (ZVOR - ZC)**2)
             F = LOG(4./SQRT(1. - ARG))
             EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*(F -
     &        B12) + (1. - ARG)*C11*(F - C12)))
             EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*(F -
     &        B22) + (1. - ARG)*C21*(F - C22)))
C
C---  calculate the induced velocities
C
             DZ = ZC - ZVOR
             UAXI = CIRC/(4.*PI)*SQRT(ARG/RVOR/RC)*(EL1 - EL2*(1. - .5*
     &        ARG*(1. + RVOR/RC))/(1. - ARG))
             URAD = CIRC*DZ/(8.*PI*RC)*SQRT(ARG/RC/RVOR)*(EL2*(2. - ARG)
     &        /(1. - ARG) - 2.*EL1)
           END IF
C
C---  determine the entropy at the cell center and store it in Qfar-4
C
           IF (ROC. LE .4.) THEN
            QFAR(4,I,K) = SEVAL(ROC,SC,RHS,RCC,NSP)
           END IF
C
C---  store the induced velocities and the entropy in the state vector U
C
           QFAR(1,I,K) = QFAR(1,I,K) + XC*URAD/RC
           QFAR(2,I,K) = QFAR(2,I,K) + YC*URAD/RC
           QFAR(3,I,K) = QFAR(3,I,K) + UAXI
  290     CONTINUE
  300   CONTINUE
  310 CONTINUE
      RETURN
      END ! PERT
```

```
C
C
C  SUBROUTINE CORENT:  computes the entropy distribution through the
C  core of an isolated 2-D vortex.  A Lamb vortex core structure is
C  assumed in order to get the velocity and vorticity distribution
C  and Crocco's relation is integrated numerically to obtain the
C  entropy in the core.
C
      SUBROUTINE CORENT(SC,RHS,RC,NSP,CORE,CIRC)
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      DIMENSION SC(1),RHS(1),RC(1)
      REAL MACH, KAPPA
      KAPPA = CIRC/PI
      CORESQ = CORE**2
      DR = 4./FLOAT(NSP-1)
      R = 0.
      FL = 0.
      RC(1) = 0.
      SC(1) = 0.
      DO 10 N = 2,NSP
        R = R + DR
        EXPR = EXP(-R**2)
        UT = KAPPA/(2.*R*CORE)*(1. - EXPR)
        OM = KAPPA/CORESQ*EXPR
        T = 1./(GAM - 1.) - UT**2/(2.*GAM)
        FR = -UT*OM/T
        SC(N) = SC(N-1) + .5*(FL + FR)*DR*CORE
        RC(N) = R
        FL = FR
   10 CONTINUE
      SINF = SC(NSP)
      DO 20 N = 1,NSP
        SC(N) = EXP(SC(N) - SINF)
   20 CONTINUE
      RETURN
      END ! CORENT
```

```
C*************************************************C
C                                                 C
C   Spline routine swiped from...                 C
C                  ... M. Giles and M. Drela      C
C                                                 C
C*************************************************C
      SUBROUTINE SPLINE(X,XP,S,II)
C
      DIMENSION X(1),XP(1),S(1)
      DIMENSION A(480),B(480),C(480)
C
      IF(II.GT.480) STOP 'SPLINE: Array overflow'
      DO 1 I = 2, II-1
C
        IO = I
        IM = I - 1
        IP = I + 1
        DSM = S(IO) - S(IM)
        DSP = S(IP) - S(IO)
C
        B(IO) = DSP
        A(IO) = 2. * ( DSM + DSP )
        C(IO) = DSM
        XP(IO) = 3.*(DSM*(X(IP)-X(IO))/DSP + DSP*(X(IO)-X(IM))/DSM)
C
  1   CONTINUE
C
      A(1) = 2.
      C(1) = 1.
      XP(1) = 3. * (X(2)-X(1)) / (S(2)-S(1))
      B(II) = 1.
      A(II) = 2.
      XP(II) = 3. * (X(II)-X(II-1)) / (S(II)-S(II-1))
C
      CALL TRISOL(A,B,C,XP,II)
C
      RETURN
      END ! SPLINE
C
C
C
      SUBROUTINE TRISOL(A,B,C,D,KK)
C
      DIMENSION A(1),B(1),C(1),D(1)
C
      DO 1 K = 2, KK
        KM = K - 1
        C(KM) = C(KM) / A(KM)
        D(KM) = D(KM) / A(KM)
        A(K) = A(K) - B(K) * C(KM)
        D(K) = D(K) - B(K) * D(KM)
  1   CONTINUE
C
      D(KK) = D(KK) / A(KK)
C
      DO 2 K = KK-1, 1, -1
        D(K) = D(K) - C(K) * D(K+1)
  2   CONTINUE
C
      RETURN
      END ! TRISOL
```

```
C
C
C
      FUNCTION SEVAL(SS,X,XP,S,N)
      REAL X(1), XP(1), S(1)
C
      ILOW = 1
      I = N
C
   10 IF(I-ILOW .LE. 1) GO TO 11
C
      IMID = (I+ILOW)/2
      IF(SS .LT. S(IMID)) THEN
       I = IMID
      ELSE
       ILOW = IMID
      ENDIF
      GO TO 10
C
   11 DS = S(I) - S(I-1)
      T = (SS-S(I-1)) / DS
      CX1 = DS*XP(I-1) - X(I) + X(I-1)
      CX2 = DS*XP(I)   - X(I) + X(I-1)
      SEVAL = T*X(I) + (1.0-T)*X(I-1) + (T-T*T)*((1.0-T)*CX1 - T*CX2)
      RETURN
      END ! SEVAL
C
C
C
      FUNCTION DEVAL(SS,X,XP,S,N)
      REAL X(1), XP(1), S(1)
C
      ILOW = 1
      I = N
C
   10 IF(I-ILOW .LE. 1) GO TO 11
C
      IMID = (I+ILOW)/2
      IF(SS .LT. S(IMID)) THEN
       I = IMID
      ELSE
       ILOW = IMID
      ENDIF
      GO TO 10
C
   11 DS = S(I) - S(I-1)
      T = (SS-S(I-1)) / DS
      CX1 = DS*XP(I-1) - X(I) + X(I-1)
      CX2 = DS*XP(I)   - X(I) + X(I-1)
      DEVAL = (X(I)-X(I-1) + (1.-4.*T+3.*T*T)*CX1 + T*(3.*T-2.)*CX2)/DS
      RETURN
      END ! DEVAL
```

```
C
C
C     SUBROUTINE FRWAKE:  A routine for solving for the positions of the
C     wake vortices of a helicopter using the fast free wake method of
C     Miller.  (Coded 12/85 by Tom Roberts.)
C
      SUBROUTINE FRWAKE
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      COMMON/FWAK/ RV(50),ZV(50),DRV(45),DZV(45),URV(50),UZV(50),
     &             GAMV(50),CORE,NWAKE,NVORT,URP,RCYLT,RCYLI,
     &             ZCYLT,ZCYLI,DGDZT,DGDZI,NSTART,NEND,ITWAKE,IRL,IRU
      COMMON/ELCO/ A11,A12,B11,B12,C11,C12,A21,A22,B21,B22,C21,C22
      REAL MACH
C
C---  start free wake iteration procedure
C
      DO 110 ITER = 1,ITWAKE
C
C---  calculate the position of the far wake cylinders
C
        RCYLT = RV(NWAKE*NVORT)
        ZCYLT = 2.*ZV(NWAKE*NVORT) - ZV((NWAKE-1)*NVORT)
        DGDZT = GAMV(NWAKE*NVORT)/(ZV((NWAKE-1)*NVORT) - ZV(NWAKE*
     &   NVORT))
        DO 20 N = 1,2
          GAMT = 0.
          RGAT = 0.
          ZRGT = 0.
          DO 10 I = 1,NVORT-1
            GT = GAMV((NWAKE-N)*NVORT + I)
            RT = RV((NWAKE-N)*NVORT + I)**2*GT
            ZT = ZV((NWAKE-N)*NVORT + I)*RT
            GAMT = GAMT + GT
            RGAT = RGAT + RT
            ZRGT = ZRGT + ZT
   10     CONTINUE
          IF (N. EQ .1) THEN
           RCYLI = SQRT(RGAT/GAMT)
           ZCYLI = ZRGT/RGAT
          ELSE
           DZ = ZCYLI - ZRGT/RGAT
           ZCYLI = ZCYLI + DZ
           DGDZI = -GAMT/DZ
          END IF
   20   CONTINUE
C
C---  calculate the velocity induced at each vortex location by all
C     the other vortices in the wake
C
        DO 30 N = 1,NWAKE*NVORT
          URV(N) = 0.
          UZV(N) = 0.
   30   CONTINUE
        DO 50 N = 1,NWAKE*NVORT-1
          DO 40 I = N+1,NWAKE*NVORT
C
C---  in here, compute the elliptic integrals (approximately)
C
            ARG = 4.*RV(N)*RV(I)/((RV(N) + RV(I))**2 + (ZV(N) -
     &       ZV(I))**2)
            F = LOG(4./SQRT(1. - ARG))
```

```
            EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*(F -
     &       B12) + (1. - ARG)*C11*(F - C12)))
            EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*(F -
     &       B22) + (1. - ARG)*C21*(F - C22)))
C
C---  calculate the induced velocities
C
            DZ = ZV(I) - ZV(N)
            UZV(I) = UZV(I) + GAMV(N)/(4.*PI)*SQRT(ARG/RV(N)/RV(I))*
     &       (EL1 - EL2*(1. - .5*ARG*(1. + RV(N)/RV(I)))/(1. - ARG))
            UZV(N) = UZV(N) + GAMV(I)/(4.*PI)*SQRT(ARG/RV(N)/RV(I))*
     &       (EL1 - EL2*(1. - .5*ARG*(1. + RV(I)/RV(N)))/(1. - ARG))
            URV(I) = URV(I) + GAMV(N)*DZ/(8.*PI*RV(I))*SQRT(ARG/RV(I)/
     &       RV(N))*(EL2*(2. - ARG)/(1. - ARG) - 2.*EL1)
            URV(N) = URV(N) - GAMV(I)*DZ/(8.*PI*RV(N))*SQRT(ARG/RV(I)/
     &       RV(N))*(EL2*(2. - ARG)/(1. - ARG) - 2.*EL1)
   40     CONTINUE
   50   CONTINUE
C
C---  compute the contribution to the induced velocities at each
C     vortex from the far wake vortex cylinders
C
        DO 60 N = 1,NWAKE*NVORT
          DZI = ZV(N) - ZCYLI
          ARG = 4.*RV(N)*RCYLI/((RV(N) + RCYLI)**2 + DZI**2)
          F = LOG(4./SQRT(1. - ARG))
          EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*(F -
     &     B12) + (1. - ARG)*C11*(F - C12)))
          EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*(F -
     &     B22) + (1. - ARG)*C21*(F - C22)))
          URV(N) = URV(N) + DGDZI/(2.*PI)*SQRT(RCYLI/ARG/RV(N))*(EL1*
     &     (2. - ARG) - 2.*EL2)
C
          DZT = ZV(N) - ZCYLT
          ARG = 4.*RV(N)*RCYLT/((RV(N) + RCYLT)**2 + DZT**2)
          F = LOG(4./SQRT(1. - ARG))
          EL1 = F + (1. - ARG)*(A11*(F - A12) + (1. - ARG)*(B11*(F -
     &     B12) + (1. - ARG)*C11*(F - C12)))
          EL2 = 1. + (1. - ARG)*(A21*(F - A22) + (1. - ARG)*(B21*(F -
     &     B22) + (1. - ARG)*C21*(F - C22)))
          URV(N) = URV(N) + DGDZT/(2.*PI)*SQRT(RCYLT/ARG/RV(N))*(EL1*
     &     (2. - ARG) - 2.*EL2)
C
C---  for the axial component of the induced velocity, integrate the
C     expression numerically
C
          DT = PI/40.
          TR = DT
          ETA = RV(N)
          RSPESI = RCYLI**2 + ETA**2
          RSPEST = RCYLT**2 + ETA**2
          ZISQ = DZI**2
          ZTSQ = DZT**2
          DENI = RSPESI - 2.*RCYLI*ETA
          DENT = RSPEST - 2.*RCYLT*ETA
          IF (ABS(DENI). LE .1.E-7) THEN
           FRACI = .5
          ELSE
           FRACI = RCYLI*(RCYLI - ETA)/DENI
          END IF
          IF (ABS(DENT). LE .1.E-7) THEN
```

```
            FRACT = .5
           ELSE
            FRACT = RCYLT*(RCYLT - ETA)/DENT
           END IF
           FI = (1. - DZI/SQRT(DENI + ZISQ))*FRACI
           FT = (1. - DZT/SQRT(DENT + ZTSQ))*FRACT
           UZ2 = DGDZI*FI + DGDZT*FT
           DO 55 J = 1,79
             COSR = COS(TR)
             DENI = RSPESI - 2.*RCYLI*ETA*COSR
             FRACI = RCYLI*(RCYLI - ETA*COSR)/DENI
             FI = (1. - DZI/SQRT(DENI + ZISQ))*FRACI
             DENT = RSPEST - 2.*RCYLT*ETA*COSR
             FRACT = RCYLT*(RCYLT - ETA*COSR)/DENT
             FT = (1. - DZT/SQRT(DENT + ZTSQ))*FRACT
             UZ2 = UZ2 + DGDZI*FI + DGDZT*FT
             TR = TR + DT
  55       CONTINUE
           UZV(N) = UZV(N) + UZ2*DT/(4.*PI)
  60     CONTINUE
C
C---   finally, compute the self induced velocity of each vortex
C
         DO 70 N = 1,NWAKE*NVORT
           UZV(N) = UZV(N) + GAMV(N)/(4.*PI*RV(N))*(LOG(8.*RV(N)/CORE) -
     &       .25)
  70     CONTINUE
C
C---   find the average change in position of each vortex element
C
         DO 80 N = 1,(NWAKE-1)*NVORT
C          DRV(N) = .5*(URV(N) + URV(N+NVORT))*PI/OMEGA
C          DZV(N) = .5*(UZV(N) + UZV(N+NVORT))*PI/OMEGA
           DRV(N) = URP*(.5*(URV(N) + URV(N+NVORT))*PI/OMEGA -
     &       RV(N+NVORT) + RV(N))
           DZV(N) = URP*(.5*(UZV(N) + UZV(N+NVORT))*PI/OMEGA -
     &       ZV(N+NVORT) + ZV(N))
  80     CONTINUE
C
C---   now, update the vortex positions using an underrelaxation
C      procedure
C
         DO 90 N = NVORT+1,NWAKE*NVORT
C          RV(N) = (1. - URP)*RV(N) + URP*(RV(N-NVORT) + DRV(N-NVORT))
C          ZV(N) = (1. - URP)*ZV(N) + URP*(ZV(N-NVORT) + DZV(N-NVORT))
           RV(N) = RV(N) + DRV(N-NVORT)
           ZV(N) = ZV(N) + DZV(N-NVORT)
  90     CONTINUE
C
C---   compute the rms and maximum delta position
C
         IF (MOD(ITER,20). EQ .0. OR .ITER. EQ .ITWAKE) THEN
          DXMAX = 0.
          DXRMS = 0.
          DO 100 N = 1,(NWAKE-1)*NVORT
            DXSQ = DRV(N)**2 + DZV(N)**2
            DXRMS = DXRMS + DXSQ
            IF (DXMAX. LE .DXSQ)  DXMAX = DXSQ
 100      CONTINUE
          DXRMS = SQRT(DXRMS/FLOAT((NWAKE-1)*NVORT))
          DXMAX = SQRT(DXMAX)
```

```
        WRITE (6,*) 'Iteration:',ITER,' dx-rms:',DXRMS,' dx-max:',DXMAX
      END IF
  110 CONTINUE
      NTOT = NWAKE*NVORT
      OPEN (UNIT=7,STATUS='OLD',FILE='WAKE.PLT')
      WRITE (7,1001) NTOT
      WRITE (7,1001) NVORT
      WRITE (7,1001) ITER
      DO 120 N = 1,NWAKE*NVORT
        RAD = RV(N)/ASRA
        AXI = ZV(N)/ASRA
        WRITE (7,1002) RAD,AXI
  120 CONTINUE
      CLOSE (7)
 1001 FORMAT(I5)
 1002 FORMAT(2F7.4)
      RETURN
      END ! FRWAKE
```

```
C
C
C---  SUBROUTINE BCIRC:  computes the bound circulation distribution
C     on the rotor and determines the distribution of vorticity in
C     the wake
C
      SUBROUTINE BCIRC(ITCOUP)
      COMMON/VAR / U(5,-1:50,-1:12,-1:18),P(-1:50,-1:12,-1:18),
     &             R(5,49,11,17),IMAX,JMAX,KMAX,IM,JM,KM,NCELLS,
     &             RO(5,48,10,16),X(3,49,11,17)
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      COMMON/FWAK/ RV(50),ZV(50),DRV(45),DZV(45),URV(50),UZV(50),
     &             GAMV(50),CORE,NWAKE,NVORT,URP,RCYLT,RCYLI,
     &             ZCYLT,ZCYLI,DGDZT,DGDZI,NSTART,NEND,ITWAKE,IRL,IRU
      COMMON/BGAM/ ROR(0:17),THET(0:17),BOUN(0:17),OMCIRC
      DIMENSION BOUNP(0:17), TBL(6)
      REAL MACH
      OMG = OMCIRC
C
C---  first, compute the bound circulation at all the span locations on
C     the rotor blade
C
      BOUN(0) = 0.
      DO 20 K = 1,KM
        CIRB = 0.
        DO 10 I = 1,IM
          DXDXI = .25*(X(1,I+1,1,K) + X(1,I+1,1,K+1) + X(1,I+1,2,K)
     &      + X(1,I+1,2,K+1) - X(1,I,1,K) - X(1,I,1,K+1) -
     &      X(1,I,2,K) - X(1,I,2,K+1))
          DYDXI = .25*(X(2,I+1,1,K) + X(2,I+1,1,K+1) + X(2,I+1,2,K)
     &      + X(2,I+1,2,K+1) - X(2,I,1,K) - X(2,I,1,K+1) -
     &      X(2,I,2,K) - X(2,I,2,K+1))
          DZDXI = .25*(X(3,I+1,1,K) + X(3,I+1,1,K+1) + X(3,I+1,2,K)
     &      + X(3,I+1,2,K+1) - X(3,I,1,K) - X(3,I,1,K+1) -
     &      X(3,I,2,K) - X(3,I,2,K+1))
          CIRB = CIRB + (U(2,I,1,K)*DXDXI + U(3,I,1,K)*DYDXI +
     &      U(4,I,1,K)*DZDXI)/U(1,I,1,K)
   10   CONTINUE
        BOUN(K) = BOUN(K) + OMG*(CIRB - BOUN(K))
   20 CONTINUE
      BOUN(KMAX) = 0.
C
C---  next, spline the bound vorticity distribution
C
      II = KMAX + 1
      CALL SPLINE (BOUN,BOUNP,THET,II)
      TBL(NVORT+1) = PI
C
C---  find the location of the peak bound circulation, and roll-up the
C     tip vortex from that point
C
      BCMAX = 0.
      DTHET = .01*PI
      DO 30 N = 100,1,-1
        THETA = PI - DTHET*FLOAT(101 - N)
        BC = SEVAL(THETA,BOUN,BOUNP,THET,II)
        IF (BC. LT .BCMAX. AND .BC. GT .0.) THEN     ! haven't found the
         TBL(NVORT) = THETA + DTHET                  ! peak yet
         GO TO 31
        ELSE
         BCMAX = BC
```

```
        END IF
   30 CONTINUE
C
C---  divide the space from the maximum bound circulation point to the
C     root into equal radial (r) increments
C
   31 RCMAX = .5*ASRA*(1. - COS(TBL(NVORT)))
      RCMIN = ROR(2)    ! take the root to be the last airfoil section
      DELR = (RCMAX - RCMIN)/FLOAT(NVORT-1)  ! on the rotor blade
      DO 50 N = 1,NVORT-1
        RBL = RCMIN + DELR*FLOAT(N-1)
        TBL(N) = ACOS(1. - 2.*RBL/ASRA)
   50 CONTINUE
C
C---  now compute the centroid of each of the vortices
C
      DO 70 N = 1,NVORT
        DTHET = .05*(TBL(N+1) - TBL(N))
        THETA = TBL(N)
        BCL = SEVAL(THETA,BOUN,BOUNP,THET,II)
        FL = .5*ASRA*BCL*SIN(THETA)
        FRL = .5*FL*ASRA*(1. - COS(THETA))
        GCEN = 0.
        GRCEN = 0.
        DO 60 NN = 1,20
          THETA = THETA + DTHET
          BCR = SEVAL(THETA,BOUN,BOUNP,THET,II)
          FR = .5*ASRA*BCR*SIN(THETA)
          FRR = .5*FR*ASRA*(1. - COS(THETA))
          GCEN = GCEN + .5*(FL + FR)*DTHET
          GRCEN = GRCEN + .5*(FRL + FRR)*DTHET
          FL = FR
          FRL = FRR
   60   CONTINUE
        GAMV(N) = .5*(BCR - BCL)
C        GAMV(N) = GAMV(N) + OMG*(.5*(BCR - BCL) - GAMV(N))  ! underrelax
        RV(N) = GRCEN/GCEN                  ! the wake circulation update
        ZV(N) = 0.
   70 CONTINUE
C
C---  assign the correct circulation to each vortex in the wake
C
      DO 90 N = 1,NVORT
        DO 80 NN = 2,NWAKE
          NNN = (NN - 1)*NVORT + N
          GAMV(NNN) = 2.*GAMV(N)
   80   CONTINUE
   90 CONTINUE
C
C---  write out the new vortex strengths and centroids, normalized
C     by (omega * R**2) and R, respectively
C
      WRITE (6,1000)
      DO 100 N = 1,NVORT
        RAD = RV(N)/ASRA
        CIRC = 2.*GAMV(N)/(SQRT(GAM)*MACH*ASRA)
        WRITE (6,1001) N,RAD,CIRC
  100 CONTINUE
      WRITE (6,1002)
C
 1000 FORMAT(/4X,'N',5X,'r/R',6X,'gamma'/3X,'---',2X,7('-'),3X,7('-'))
```

```
1001 FORMAT(I5,2F10.5)
1002 FORMAT(1X)
     RETURN
     END ! BCIRC
```

```
C
C
C  SUBROUTINE BCINIT:  initializes the bound circulation distribution
C  from combined momentum and blade element theory
C
      SUBROUTINE BCINIT(DCLDA,SIGMA,THET75,TWIST)
      COMMON/VAR / U(5,-1:50,-1:12,-1:18),P(-1:50,-1:12,-1:18),
     &             R(5,49,11,17),IMAX,JMAX,KMAX,IM,JM,KM,NCELLS,
     &             RO(5,48,10,16),X(3,49,11,17)
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      COMMON/BGAM/ ROR(0:17),THET(0:17),BOUN(0:17),OMCIRC
      REAL MACH
      THET75 = PI*THET75/180.
      TWIST = PI*TWIST/180.
C
C---  determine the bound circulation distribution from combined
C     momentum/blade element theory; the blade is assumed to have a
C     linear twist and a constant chord
C
      BOUN(0) = 0.
      DO 10 K = 1,KM
        PITCH = THET75 - (ROR(K)/ASRA - .75)*TWIST
        WOOR = .0625*SIGMA*DCLDA*ASRA/ROR(K)*(-1. +
     &   SQRT(1. + 32.*ROR(K)*PITCH/(ASRA*SIGMA*DCLDA)))
        BOUN(K) = 4.*OMEGA*(WOOR*ROR(K))**2/(SIGMA*ASRA)
   10 CONTINUE
      BOUN(KMAX) = 0.
      RETURN
      END ! BCINIT
```

```
C
C
C  SUBROUTINE SURFCO:  writes the coordinates of the wing surface
C  to be used in plotting the pressure coefficients
C
      SUBROUTINE SURFCO
      COMMON/VAR / U(5,-1:50,-1:12,-1:18),P(-1:50,-1:12,-1:18),
     &             R(5,49,11,17),IMAX,JMAX,KMAX,IM,JM,KM,NCELLS,
     &             RO(5,48,10,16),X(3,49,11,17)
      COMMON/PARA/ MACH,GAM,OMEGA,HIN,ASRA,CFL,AENTH,PI,ILE
      WRITE (3) IM,KM
      DO 20 K = 1,KM
C
C---  find coordinate of spanwise station
C
        YAV = .5*((X(2,1,1,K) + X(2,1,1,K+1)))/ASRA
        WRITE (3) YAV
C
C---  find chord at span station K
C
        XTE = .5*(X(1,1,1,K) + X(1,1,1,K+1))
        XLE = .5*(X(1,ILE+1,1,K) + X(1,ILE+1,1,K+1))
        CHORD = XTE - XLE
C
C---  write chordwise coordinates at span station K
C
        DO 10 I = 1,IM
          XAV = .25*(X(1,I,1,K) + X(1,I+1,1,K) + X(1,I,1,K+1) +
     &     X(1,I+1,1,K+1))
          XAV = (XAV - XLE)/CHORD
          WRITE (3) XAV
   10   CONTINUE
   20 CONTINUE
      RETURN
      END ! SURFCO
```