HELICOPTER ROTOR BLADE LOADING CALCULATIONS USING AN AXISYMMETRIC VORTEX SHEET AND THE FREE WAKE METHOD

by

KEVIN HUH

B.S. Honors, The University of Illinois at Urbana-Champaign
(1985)

Submitted to the Department of
Aeronautics and Astronautics
in Partial Fulfillment of the
Requirement of the Degree of

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 1988

Signature of the Author _____
Department of Aeronautics and Astronautics, January 26, 1988

Certified by _____
Professor Sheila E. Widnall
Thesis Supervisor

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

HELICOPTER ROTOR BLADE LOADING CALCULATIONS USING AN AXISYMMETRIC VORTEX SHEET AND THE FREE WAKE METHOD

by

Kevin Huh

Submitted to the Department of Aeronautics and Astronautics
on January 26, 1988, in partial fulfillment of the
requirements of the Degree of Master of Science in
Aeronautics and Astronautics

Abstract

The wake geometry and the loading characteristics for a hovering helicopter are studied using an axisymmetric vortex sheet and the free wake analysis. The structure of the rolled-up vortex wake is calculated using a filament model as suggested by Kantelis & Widnall. The model represents the spiral portion of the sheet as an equivalent single vortex with higher order terms. The entire sheet is rediscretized at each time step to minimize errors due to uneven sheet stretching. The rollup produces distinct regions of high vorticity concentrations, which are then replaced with discrete vortices to simplify further calculations.

The lifting line theory is used to predict the rotor bound circulation distribution, and the extended Miller's free wake analysis is applied to the intermediate and the far wakes. Both the load distributions and the tip vortex locations show good agreements with experimental data. It appears using three vortices to represent amalgamated trailing vorticies in the intermediate wake is adequate.

Thesis Supervisor: Sheila E. Widnall

Title: Abby Rockerfeller Mauze Professor of
Aeronautics and Astronautics

ACKNOWLEDGEMENT

I would like to express my thanks to my research advisor, Professor Sheila E. Widnall, for giving me the opportunity to work on this interesting project under her guidance. Her support and expert guidance have been invaluable and are appreciated.

I would like to express my gratitude to Professor Miller. His experiences and insights were immensly helpful in all phases of the research.

I would like to thank my friends at MIT for sharing their knowledge and experiences with me. In particular, I would like to thank Fei Li who was always available to assist me.

I would like to thank my family members for their support and encouragement.

Finally, I would like to express my appreciation to Lynn, who was always only a phone call away. Without her emotional support this work would not have been possible.

Contents

List of Figures

NOMENCLATURE

Roman Letters

a	Pullin's scaling constant
b_n	Pullin's inner spiral constant
c	Core diameter
c	Chord lenth
C_l	Section lift coefficient
d_n	Pullin's inner spiral constant
E	Elliptic integral of the second kind
F	Elliptic integral constant
f_i	Interpolated function value for rediscretization
g_i	Rediscretization parabola
K	Elliptic integral of the first kind
K_1	Constant defined in L-L theory
K_2	Constant defined in L-L theory
k_0	Runge-Kutta constant
k_1	Runge-Kutta constant
k_2	Runge-Kutta constant
k_3	Runge-Kutta constant
l_0	Runge-Kutta constant
l_1	Runge-Kutta constant

l_2	Runge-Kutta constant
l_3	Runge-Kutta constant
L	Lift
m_o	Lift-curve slope
n	Number of intermediate wakes vorticies
N	Number of near wake vorticies
P	Impulse
r	Radius, radial direction
r_1	Least distance from (r, z) to the ring
r_2	Greatest distance from (r, z) to the ring
r_i	Radius of vortex i
r_i^{old}	Previous radius of vortex i
r_i^{new}	Current radius of vortex i
$\tilde{r}$	Radius of vortex inside spiral$i$
$\vec{r}$	Position vector
$\dot{r}$	Total velocity in the radial direction
R	Radius of rotor
s	Arc-length parameter
S_i	Shearing velocity at ring i
T	Kinetic energy
$\vec{s}$	Position vector
$\vec{t_i}$	Tangent vector
$\vec{u}$	Velocity vector
u_r	Velocity in the radial direction
u_ϕ	Velocity in the azimuthal direction
u_z	Velocity in the axial direction
$u_{z\,sind}$	Self-induced component of velocity
$u_{\tilde{x}}$	Velocity in the $\tilde{x}$ direction

$u_{\tilde{y}}$	Velocity in the $\tilde{y}$ direction
$u_{z\,ind}$	Self-induced component of velocity
U_{max}	Max velocity
$\vec{V}$	Velocity vector
w	Downwash
w_{wake}	Downwash due to the wake
w_{tv}	Downwash due to trailing vortex sheet
x_{max}	Maximum translation
$\vec{x}$	Position vector
$\vec{x}'$	Position vector used for Biot-Savart law
$\tilde{x}$	Local horizontal axis for the spiral
$\tilde{y}$	Local verticle axis for the spiral
z	Axial direction
z_i	Axial position of ring i
z_i^{old}	Previous axial position of vortex i
z_i^{new}	Current axial position of i
$\dot{z}$	Total velocity in the axial direction

Greek Letters

α	Angle of attack
α_{geomt}	Geometric angle of attack
α_{ind}	Induced angle of attack
α_{rot}	Rotation angle for spiral modelling
γ_i	Vorticity of vortex i
Γ	Circulation

Γ_i	Circulation of vortex i
ψ	Variable of integration for elliptic integrals
Ψ	Streamfunction
ω	Underrexation parameter, stream funtion constant
Ω	Blade rotation rate
Θ_{max}	Maximum spiral angle
Θ_s	Rotation angle defined by Kantelis
$\tilde{\theta}$	Rotation angle for coordinate transformation
η	Vortex ring calculation constant, radial direction
ϕ	Blade passing angle
τ	Pullin's inner spiral constant
τ_a	τ at A, constant position on the sheet
ρ	Fluid density
σ	Solidity
Λ	Initial portion to be lumped to tip

Chapter 1
Introduction

1.1 Brief History

An accurate prediction methodology for the aerodynamic flow over a lifting rotor has been a continuous problem to engineers for over forty years. The main difficulty lies in the complexity of the flow field about a rotating airfoil. In particular, due to the close interaction of the blade and the tip vortex of the previous blade, any methodology must be able to accurately predict the rollup schedule of the trailing vortex sheet. (Figure 1.1) A precursor to such a condition is the necessity of having an accurate wake model, since a trailing vortex sheet of a rotor convects within its own wake.

The early methods of prediction involved vortex methods originally developed for propellers and adapted to lifting rotors. These classical theories assumed that the wake may be treated as a rigid, non-contracting helical vortex cylinder. While this assumption is consistent for high speed propellers for which the induced velocities are small with respect to the axial translational speed, it is clearly wrong for a hovering helicopter for which the only velocities are the induced velocities. The momentum theory predicts that the induced velocity on the rotor blade is half of the velocity in the fully developed wake, and therefore the wake must contract.[14] Hence the fixed non-contracting wake theory is inapplicable to an analysis of a hovering rotor, and a more accurate description

of the vortex wake is required. There exists currently two distinct methods of modelling the helicopter vortex wake.

One such method is called the prescribed wake method [13]. These methods involve empirical formulas involving blade geometry and the thrust coefficients to obtain vortex wake geometries. They have proven to be quite efficient and accurate for conventional rotor configurations. However, the method's dependence on empiricism means that given an unconventional blade geometry, the results would become unreliable. The scheme's greatest fallacy is its inability to model the physics of vorticity transport.

A second scheme which is applicable given a general geometry is called the Free Wake Method. [20,14,15,21,19] These schemes are all based on the fact that vorticies move with the fluid. The wake is modelled as vortex sheets or filaments and allowed to convect in space according to the Biot-Savart relationship in a Lagrangian fashion until geometric and loading convergence is achieved. Although the free wake method is completely general, they are also computationally quite expensive owing to the large number of calculations required for vortex transportation. In addition, the question of numerical stability of these methods in three-dimensional flow has not been adequately addressed. The apparent convergences of current three-dimensional free-wake calculations are likely due to the coarse discretizations used because of limitations of current computers. There is a need for a test case which does not suffer from such three-dimensional instabilities, and thus can be used to study their convergence.

A simpler, yet physically viable approach to the rotor wake problem is the simplified free wake theory first proposed by Miller [14]. This scheme reduces

the number of vortex velocity calculations required by amalgamating the vortex sheet into two or three axisymmetric vorticies after 1/4 blade rotation. The amalgamation process is based on the Donaldson et al. [5] method first proposed for fixed wing aircrafts. Simply stated, Donaldson's rollup criteria for a complex wake states that separate vortices will be formed between the local minima of the curve $|d\Gamma/dr|$ vs r, and thus the tip vortex will be of maximum strength containing all of the negative vorticity in the region.[5] In addition, Miller simplified the vorticity representation in the far field by replacing the sequence of far-wake finite vorticies with distributed axisymmetric vortex sheets or cylinders. The calculations obtained seem to show good agreements with empirical data and required only a fraction of computations required of the full free wake technique.

Roberts & Murman [19] extended Miller's work by coupling the simplified free wake technique with an Euler solver in the near field of the rotor. This work offered the advantages of using an Euler scheme as opposed to potential methods, namely permitting distributed vorticity. This allows one to gain insights into the structure of the vortex core. In addition, Roberts & Murman were able to avoid excessive number of computations and vorticity dissipation by implementing a novel perturbation scheme. The calculations obtained showed good agreement with empirical data. However, Roberts used the same rollup model proposed earlier by Donaldson et al. [5], and questions remain as to its validity for hovering rotors.

1.2 Motivation for Study

The analysis included in the current report expands on the simplified free wake method to include the near wake rollup model for an axisymmetric vortex

sheet proposed by Kantelis & Widnall. [11] Previous works of rotor in hover have essentially ommitted a detailed analysis of the structure of rolled-up vorticity in the wake. Hence, the objectives of the study are:

- To formulate an accurate prediction method for the rotor blade bound circulation distribution and the tip vortex geometries.
- To offer physical insights of the transportation of vorticity in the wake, in particular, the amalgamation of inboard vorticity.
- To assess the accuracy of the rollup criteria proposed by Donaldson et al. [5] as used in other free wake techniques.

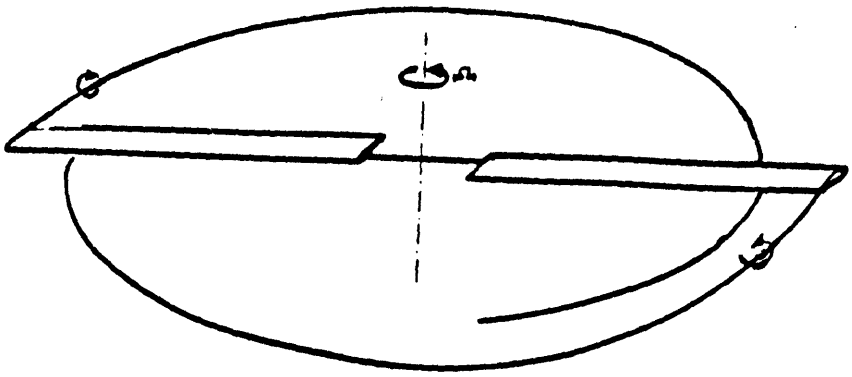


Figure 1.1: Blade-Vortex interaction

Chapter 2
Theoretical Development

2.1 Physical Model

The physical model in study here is one of an isolated rotor blade in hover position as illustrated in Figure 2.1; its associated wake is shown in Figure 2.2. The fluid is assumed to be steady, incompressible, and inviscid. There are two important consequences of these physical assumptions:

- Inviscidness and incompressibility of the fluid implies vorticity associated with a particular fluid element must remain constant.
- A rotor in hover implies no mean flow in the axial or radial directions, and hence all the perturbation velocities in the medium is generated by the vorticity in the fluid.

2.2 Wake Modelling

2.2.1 Miller's Simplified Free Wake Technique

A model which embodies the above two physical conditions is the simplified free wake technique, first used by Miller [14] and referred to earlier in the introduction. A detailed descriptions are available in other refrences [14,15,19,21] ,

and only a brief summary will be presented here.

The bound circulation is computed using lifting-line theory. The trailing vorticity from the blade is then transported into the wake using discrete vortex filaments according to Helmholtz's vortex theorems for a steady, inviscid flow; vortex lines must follow streamlines in the flow.[19] The vortex trajectories, therefore, must obey the following equations:

$$\frac{u_{ri}}{dr_i} = \frac{u_{\phi i}}{r_i d\phi} = \frac{u_{zi}}{dz_i} \tag{2.1}$$

where u_r, u_ϕ, and u_z are the radial, azimuthal, and axial velocity components in the frame of reference of the rotor (Figure(2.3)), and i is the filament index. The essence of the free wake analysis is the solution of the Equations 2.1 for the N discrete vorticies , where N represents the total number of vortex filaments in the wake. 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; the contributions of each blade cancel when averaged over the azimuth. [19] Next, the inclinations of the spiral wake are ignored. Miller [14] showed that the change in inclination angle only refines the downwash on the blade to the second-order. By ignoring the inclination effect of the wake, one may ignore the perturbations in the azimuthal direction, thus u_ϕ becomes,

$$u_{\phi i} = r_i \Omega \tag{2.2}$$

where Ω is the angular speed of the rotor. Substituting into Equations 2.1 yields,

$$dr_i = \frac{u_{ri}}{\Omega} d\phi \tag{2.3}$$

$$dz_i = \frac{u_{z\,i}}{\Omega} d\phi \tag{2.4}$$

Neglecting the azimuthal velocity perturbations is equivalent to treating the flow in a given azimuthal plane to be axisymmetric, and therefore, the helical vortex filaments may be replaced by vortex rings. A further simplification is made by computing the velocities and solving for the position of the vorticies only in the azimuthal plane containing the rotor blade, $\phi = 0$. Each vortex ring is constructed by taking π segments of the helical filaments from each blade and replacing them with vortex rings at their mean locations, as shown in Figure(2.5).

The total number of vorticies is reduced by dividing the wake into three distinct sections: Near wake attached to the blade, an intermediate wake, and a far wake as shown in Figure(2.4).

The near wake represents vorticity transported between blade passing time of $\phi = 0$ to $\phi = \frac{\pi}{2}$ and is approximated by a series of straight semi-infinite vortex filaments extending from the blade nodes.

It is assumed that the near wake rolls-up almost immediately according to Betz's theory of conservation of momentum into vortex rings. [4] The rollup criterion is based on the Donaldson et al. [5] approximation of dividing the vortex sheet along the local minima of the curve $|d\Gamma/dr|$ vs r. Hence, a tip vortex will form from the amalgated trailing vortex sheet starting at the tip to the maximum circulation position. A second vortex will form from this position to where $\frac{d\Gamma}{dr}$ again changes abruptly. The remaining circulation to the root is rolled-up into a third vortex.

The intermediate wake consists of four sets of vortex rings. Each ring is formed by taking π segments of the helical filaments as explained above. The first represents vorticity transported from blade passing time $\phi = \frac{\pi}{2}$ to $\phi = \frac{3\pi}{2}$, the second from time $\phi = \frac{3\pi}{2}$ to $\phi = \frac{5\pi}{2}$, the third from time $\phi = \frac{5\pi}{2}$ to $\phi = \frac{7\pi}{2}$, and the fourth from time $\phi = \frac{7\pi}{2}$ to $\phi = \frac{9\pi}{2}$. (Figure 2.5) The radii and vertical spacings are determined from the velocities induced by the wake, averaged over the spacing between vorticies.

To transport the vorticity from blade passing time $\phi = \frac{9\pi}{2}$ out to $\phi = \infty$, as per required for steady state hover conditions, two semi-infinite vortex cylinders are placed one vortex spacing below the last set of intermediate wake vorticies. The outboard cylinder is placed directly underneath the tip vortex, and its strength, $\frac{d\Gamma}{dz}$, is determined by dividing tip vortex circulation Γ by the spacing between the last two intermediate wake tip vorticies. To determine the location and strength of the inner vortex cylinder, first the centroids of the inner portion from the last two intermediate vortex sheets are computed, then the determination is completed as per tip vortex cylinder.

Since the problem is nonlinear, an iterative approach is used for its solution. At each iteration step the velocities induced at the intermediate wake vortex rings are computed from all the vorticity in the wake, including the rolled-up near wake. A new wake geometry followed by a new bound circulation of the blade is then computed. The new loading now determines the strengths of the vorticies in the wake, and the process is repeated until a suitable convergence criterion is satisfied for the bound circulation distribution.

2.2.2 Extended Simplified Free Wake Theory

The analysis given here expands on the ideas given by Miller by adding more detail to the near wake. As explained in the introduction, to correctly predict the blade loading distribution, it is essential that the analysis must accurately forecast the position and strength of the tip vortex at the first pass. Previous free wake methods relied upon Donaldson's [5] rollup model for the tip vortex determination, and it is felt that this rollup model may not be accurate for rotor wakes. Hence, a model which more closely resembles the fluid physics at the near wake seems necessary. The near wake in the current method is refined by direct calculation of the vortex wake rollup during the interval to the first blade passage ,and by more rigorous adhesion to vortex physics as explained by Betz[4]; namely conservation of circulation, conservation of impulse, and conservation of kinetic energy.

The near wake begins attached to the rotor blade and consists of n number of vortex rings, where n is typically 50 - 100. As in Miller's method, the attached near wake represents vorticity transport from time $\phi = 0$ to $\phi = \pi/2$. At the end of the time period, $\phi = \pi/2$, the near wake vorticies are allowed to convect freely in space until time $\phi = 3\pi/2$ at which time a new wake is formed. It is assumed that a new vortex ring is formed at each π rotation (for a two-bladed rotor), and hence the near wake represents vorticity transport from blade passing time $\phi = 0$ to $\phi = 3\pi/2$.(Figure 2.6) At the end of the transport, the vortex sheet is split into computationally determined number of intermediate wake vorticies according to a shear velocity criterion to be discussed.

As in Miller's method there are four levels of intermediate wakes. Thus the first level represents time $\phi = 3\pi/2$ to $\phi = 5\pi/2$, the second from $\phi = 5\pi/2$ to

$\phi = 7\pi/2$ and so on. The locations of the intermediate wake vortex rings are determined from velocity calculations of all the vorticies in the wake, and no averaging is performed.

The far wake position and strength are computed as in Miller's technique.

The problem still requires an iteration process. The positions of each vortex rings in the near and intermediate wakes are integrated in a Runge-Kutta fashion from blade passing time of $\phi = \pi/2$ to $\phi = 3\pi/2$; at which point a new bound circulation distribution, and a new vortex sheet would be formed on the rotor blade. The process is repeated until a convergence criterion is satisfied in the wake geometry and the bound circulation distribution. The iteration process will be discussed in more detail in the following chapter concerning numerical procedures.

2.2.3 Velocity Field

The vortex filament trajectories are computed by integrating the Equations 2.3 and 2.4. The perturbation velocities at the vortex locations are induced by all the discrete vorticies in the wake, and since vortex rings are curved this incldes the control vortex itself. The Biot-Savart law expresses the velocity field $\vec{V}(\vec{x})$ induced by a vortex line $\vec{\Gamma}'$ as, [3]

$$\vec{V}(\vec{x}) = -\frac{1}{4\pi}\int_{\sigma}\frac{\vec{s}\times\vec{\Gamma}'}{s^3}dA(\vec{x}') \tag{2.5}$$

where $\vec{s} = \vec{x} - \vec{x}'$, $\vec{\Gamma}'$ is the strength of the sheet at the point $\vec{x}'$, and σ extends over the entire sheet.

For the special case of axisymmetric vortex rings, Equation 2.5 is greately simplified. Given 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 2.7) are given by Lamb [12] to be:

$$u_r = \frac{\Gamma}{4\pi}\int_0^{2\pi} \frac{zr\cos\psi d\psi}{(\eta^2 + r^2 + z^2 - 2r\eta\cos\psi)^{\frac{3}{2}}} \tag{2.6}$$

$$u_z = \frac{\Gamma}{4\pi}\int_0^{2\pi} \frac{r(r - \eta\cos\psi)d\psi}{(\eta^2 + r^2 + z^2 - 2r\eta\cos\psi)^{\frac{3}{2}}} \tag{2.7}$$

The Equations 2.6 and 2.7 may further be simplified by expressing them in terms of complete elliptic integrals of the first and second kind,

$$K = \int_0^{\frac{\pi}{2}} \frac{d\psi}{(1 - k^2\sin^2\psi)^{\frac{1}{2}}} d\psi \tag{2.8}$$

$$E = \int_0^{\frac{\pi}{2}} (1 - k^2\sin^2\psi)^{\frac{1}{2}} d\psi \tag{2.9}$$

where

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

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

$$u_r = \frac{\Gamma}{4\pi}\frac{z}{2\eta}\sqrt{\frac{k^2}{r\eta}}\{E\frac{2-k^2}{1-k^2} - 2K\} \tag{2.10}$$

$$u_z = \frac{\Gamma}{4\pi}\sqrt{\frac{k^2}{r\eta}}\{K - E[\frac{1-\frac{k^2}{2}(1+\frac{r}{\eta})}{(1-k^2)}]\}. \tag{2.11}$$

In addition, the elliptic integrals K and E may be expressed in a series form.[19] These series are rapidly convergent, and only the first four terms are kept for computational purposes.

$$\begin{aligned} K = \; & F + \tfrac{1}{2}^2(F-1)(1-k^2) + (\tfrac{1\cdot3}{2\cdot4})^2(F-1-\tfrac{2}{3\cdot4})(1-k^2)^2 + \\ & (\tfrac{1\cdot3\cdot5\cdot}{2\cdot4\cdot6\cdot})^2(F-1-\tfrac{2}{3\cdot4}-\tfrac{2}{5\cdot6})(1-k^2)^3 + \cdots \end{aligned} \tag{2.12}$$

$$\begin{aligned} E = \; & 1 + \tfrac{1}{2}(F-\tfrac{1}{2})(1-k^2) + (\tfrac{1}{2})^2\tfrac{3}{4}(F-\tfrac{2}{2}-\tfrac{1}{3\cdot4})(1-k^2)^2 + \\ & (\tfrac{1\cdot3}{2\cdot4})^2\tfrac{5}{6}(F-\tfrac{2}{2}-\tfrac{2}{3\cdot4}-\tfrac{1}{5\cdot6})(1-k^2)^3 + \cdots \end{aligned} \tag{2.13}$$

where

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

The velocity induced by the far wake votex cylinders are given by:

$$u_{r\,cyl} = \frac{1}{4\pi}\frac{d\Gamma}{dz}\int_z^\infty\int_0^{2\pi}\frac{zr\cos\psi d\psi dz}{(\eta^2+r^2+z^2-2z\eta\cos\psi)^{\frac{3}{2}}} \tag{2.14}$$

$$u_{z\,cyl} = \frac{1}{4\pi}\frac{d\Gamma}{dz}\int_z^\infty\int_0^{2\pi}\frac{r(r-\eta\cos\psi)d\psi dz}{(\eta^2+r^2+z^2-2z\eta\cos\psi)^{\frac{3}{2}}} \tag{2.15}$$

Here, $\frac{d\Gamma}{dz}$ is the vortex cylinder strength, z is the axial distance from the end of the cylinder to the point at which the velocity is required, r is the cylinder radius, and η is the radius at the point of evaluation. By integrating the above equations over z we obtain,

$$u_{r\,cyl} = \frac{1}{4\pi}\frac{d\Gamma}{dz}\int_0^{2\pi} \frac{r\cos\psi d\psi}{(\eta^2 + r^2 + z^2 - 2r\eta\cos\psi)^{\frac{1}{2}}} \tag{2.16}$$

$$u_{z\,cyl} = \frac{1}{4\pi}\frac{d\Gamma}{dz}\int_0^{2\pi} \frac{r(r-\eta\cos\psi)}{\eta^2 + r^2 - 2r\eta\cos\psi}[1 - \frac{z}{(\eta^2 + r^2 + z^2 - 2r\eta\cos\psi)^{\frac{1}{2}}}]d\psi \tag{2.17}$$

The Equation 2.16 can be rewritten in terms of the complete integrals of the first and the second kind,

$$u_{r\,cyl} = \frac{1}{2\pi k}\frac{d\Gamma}{dz}\sqrt{\frac{r}{\eta}}[K(2-k^2) - 2E] \tag{2.18}$$

where k, K, and E are as defined previously. The equation for the axial component of velocity due to a vortex cylinder cannot be expressed solely in terms of the complete integrals of the first and second kind, but contains elliptic integrals of the third kind. Therefore, it is more convenient to evaluate the velocity by numerical integration of Equation 2.17 using the trapezoidal rule.[19] The approach used here is to use sixty integration steps in the ψ direction. As Roberts et al. points out, 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 ignored at the endpoints when $r = \psi$, and the expression is numerically evaluated for each of the remaining steps.

Since a curved vortex has a velocity influence upon itself, the vortex rings must have a self-induced velocity component. From Lamb [12] , a vortex ring of strength Γ, radius r, and a core radius of c has a self-induced velocity of,

$$u_{z\,sind} = \frac{\Gamma}{4\pi r}(\ln\frac{8r}{c} - \frac{1}{4}) \tag{2.19}$$

The core radius c is computed from the rolled-up near wake by conserving the kinetic energy of the vortex sheet; the details of the process will be discussed in the next section.

Using the above formulas for the induced velocities of each vortex ring and cylinder in the wake, the total induced velocity at each vortex location is determined. These velocities are integrated to solve for the new vortex locations at time $\phi = \phi + \triangle\phi$.

2.2.4 Near Wake Model

Kantelis & Widnall [11] discretized an isolated axisymmetric vortex sheet and tracked its rollup schedule. They showed that a simple straightforward application of the vortex filament model yields a chaotic result. As in previous two-dimensional cases, for example Fink & Soh [6,7], the vortex rings begin to rollup as expected, but quickly accumulate large errors as the rings begin to cross over each other. A typical example of this is shown in Figure (2.8) modelled by 20 rings. Kantelis et al. also showed that a simple increase in the number of vortex rings does not in itself alleviate the problem.

Kantelis et al. suggests that the cause of the errors are two-fold:

- The inablity of a finite number of discrete vorticies to model the singularity at the sheet edge.
- As the sheet stretches unevenly, the initially equally spaced vortex elements become unevenly spaced which can introduce large errors in the velocity calculations. This is to be expected since Fink & Soh [6,7]) re-

ported similiar errors for two-dimensinal vortex sheets. They suggested that the errors are due to the neglect of a logarithmic term in the solution of the Cauchy principle value integral representing the velocity. When the vortex is centered in the segment, the logarithmic term vanishes. As the vortex approaches the edges of the segment, however, the contribution of the logarithmic term becomes large and can not be neglected.

Hence, to model the time history of the near wake rollup correctly, a different method which reduces the above two errors must be introduced. A method which reduces the severity of the errors at the sheet edge in a discrete manner has been suggested by, among others, Hoeijmakers & Vaastra [8] and by Kantelis & Widnall [11]. The method replaces the inner portion of the spiral with a single tip vortex ring such that the total circulation and centroid of the system is preserved; amalgamation criterion used by Kantelis is to limit the spiral portion of the sheet to a fixed number of loops. (Figure(2.9)) In other words, if a vortex ring spiraled past the threshold loop number, it would be amalgamated into the tip vortex. The new amalgamated vortex strength is then computed by summing the two vorticies' circulation, and the location of the tip vortex is computed by conserving total impulse, i.e. the total impulse before amalgamation must equal the total impulse after amalgamation. For an axisymmetric vortex ring system the total impulse is given by Lamb [12] to be,

$$\frac{P}{\rho} = \pi \sum_{i=1}^{N} \Gamma_i r_i^2 \tag{2.20}$$

where P is the impulse, ρ is the density , N is the number of vorticies in the near wake, Γ_i is the circulation, and r_i is the radius of the vortex ring.

Similiarly, the core size of the newly amalgamated vortex is computed by conserving the kinetic energy of the system. Lamb [12] gives the the total kinetic energy of the flow to be,

$$T = -\pi\rho \int_0^\infty \int_{-\infty}^\infty \Psi\gamma dz dr \tag{2.21}$$

where γ is the vorticity and Ψ is the streamfunction. The streamfunction Ψ is again given by Lamb [12] to be,

$$\Psi(r,z) = -\frac{\Gamma}{2\pi}(r_1 + r_2)[K(\omega) - E(\omega)] \tag{2.22}$$

where:

r_1 = least distance from (r,z) to the ring
r_2 = greatest distance from (r,z) to the ring
$\omega = (r_1 - r_2)/(r_1 + r_2)$
K = Complete elliptic integral of the first kind
E = Complete elliptic integral of the second kind
Ψ, Γ as defined previously

Since the flow is irrotational everywhere except in the interior of the vortex rings, the energy can be expressed as a summation over all the rings

$$T = -\pi\rho \sum_{i=1}^{N} \int\int \Psi_i \gamma_i dz dr \tag{2.23}$$

where the integration is performed over the filament core sections. For each core section i, the streamfunction is written as the sum of the contributions of all of

the vortex rings including that of ring i itself.

$$\Psi_i = \sum_{j=1}^{N} \Psi_{i,j} \tag{2.24}$$

where $\Psi_{i,j}$ is the streamfunction at ring i due to ring j. Then the kinetic energy becomes,

$$T = -\pi\rho \sum_{i=1}^{N} [\int\int \Psi_{i,i}\gamma_i dzdr + \int\int \sum_{j=1, j\neq i} N\Psi_{i,j}\gamma_i dzdr] \tag{2.25}$$

where $\Psi_{i,i}$ is the streamfunction at ring i due to itself. The first term in the brackets is given by Lamb as,

$$\int\int \Psi_{i,i}\gamma_i dzdr = -\frac{k_i^2 R_i}{2\pi}[\ln \frac{8R_i}{c_i} - \frac{7}{4}] \tag{2.26}$$

where R_i is the radius and c_i is the core size of the vortex ring. The second term is integrated numerically assuming that $\Psi_{i,j}$, given by Equation 2.22, is constant over the core section of the ring.(Figure(2.10))

In addition to the tip amalgamation, extra higher-order velocity terms are required for both within and without the spiral since the spiral is asymmetric. Kantelis computes the asymmetric terms of the spiral using Pullin's [18] numerical integration of Kaden's [10] similarity solution for a parabolically loaded two-dimensional semi-infinite sheet. The assumption here is that a typical helicopter load corresponds closely to a parabolic load at the sheet edge, and based on previous calculations this assumption appears to be a reasonable one. For a detailed explanation of the higher-order terms consult Kantelis & Widnall [11] ; only the final result will be given here. The velocity at the tip vortex ring due to the higher-order asymmetric terms are,

$$u_{\tilde{z}} = -0.000332\tau_A^{-2.383} \tag{2.27}$$

$$u_{\tilde{y}} = -0.004885\tau_A^{-1.543} \tag{2.28}$$

where $(\tilde{x}, \tilde{y})$ is the rotating coordinate system fixed to the spiral center. τ_A corresponds to $\tau = a^4 t/\Gamma^3$ at position A; where A is the reference starting position of the spiral, and a is an arbitrary scaling factor used by Pullin[18]. (Figure(2.11)) To obtain the velocity components with respect to (r, z) rather than $(\tilde{x}, \tilde{y})$ a coordinate transformation is performed such that,

$$u_r = (u_{\tilde{x}} \cos \alpha_{rot} - u_{\tilde{y}} \sin \alpha_{rot}) \tag{2.29}$$

$$u_z = (u_{\tilde{x}} \sin \alpha_{rot} + u_{\tilde{y}} \cos \alpha_{rot}) \tag{2.30}$$

The angle α_{rot} is dependent on time and is the angle through which the $(\tilde{x}, \tilde{y})$ coordinate system must be rotated in order to match the point A with the actual sheet edge. (Figure(2.12)) The sheet edge position is determined during the rollup calculations.

The velocity at the remaining portion of the near wake due to the asymmetry in the spiral rollup is,

$$u_{\tilde{\theta}} = -\frac{\partial \Psi}{\partial \tilde{r}} + \frac{\Gamma}{2\pi \tilde{r}} \tag{2.31}$$

$$u_{\tilde{r}} = \frac{1}{r}\frac{\partial \Psi}{\partial \tilde{\theta}} \tag{2.32}$$

where$(\tilde{r}, \tilde{\theta})$ is the corrdinate system fixed to the center of the spiral as shown in Figure(2.12). The asymmetric stream function, Ψ, is computed numerically,

$$\Psi = \sum_{n=1}^{\infty} [b_n \tau^{m_n} \frac{\sin n\tilde{\theta}}{r^n} + d_n \tau^{l_n} \frac{\cos n\tilde{\theta}}{r^n}] \tag{2.33}$$

Computationally, only the first seven terms in the series are used and their values are,

n	b_n	m_n	d_n	l_n
1	0.005301	-1.386	0.0010070	-2.373
2	0.001310	-1.363	0.0001550	-2.282
3	0.000575	-1.345	0.0000657	-2.409
4	0.000319	-1.330	0.0000312	-2.361
5	0.000202	-1.316	0.0000157	-2.173
6	0.000139	-1.301	0.0000102	-2.221
7	0.000100	-1.283	0.0000115	-2.807

The velocity components in the usual (r, z) system are again computed by coordinate transformation

$$u_r = (u_{\tilde{r}} \cos\theta_s - u_{\tilde{\theta}} \sin\theta_s) \tag{2.34}$$

$$u_z = (u_{\tilde{r}} \sin\theta_s + u_{\tilde{\theta}} \cos\theta_s) \tag{2.35}$$

where $\theta_s = \tilde{\theta} - \pi/2 + \alpha_{rot}$ and is taken positive in the usual sense with $\theta_s = 0$ on the positive r axis.

The Pullin's numerical solution was also used as an initial condition such that the numerical simulation may be started at some time, $\phi = \pi/2 + \phi_o$, where $\phi_o > 0$. The outer portion of the sheet will already have rolled up into a logarithmic spiral which can be approximated using the Pullin model described earlier in the section. This approach eliminates the problem of attempting to model the initial motion of the tip singularity in a discrete manner with a series of vortex elements. [11]

The logarithmic spiral of the sheet edge at time $\phi = \pi/2 + \phi_o$ is replaced by a finite tip vortex at position $(r, z) = (a_o, b_o)$ with its associated circulation and higher order terms. The position vector (a_o, b_o) is,

$$a_o = 0.308(a\phi_o)^{\frac{2}{3}}$$

$$b_o = 0.489(a\phi_o)^{\frac{2}{3}}$$

where a is the Pullin scaling factor. The strength of the initial tip vortex is determined as,

$$\Gamma_{tip\,o} = -\Gamma_{r=1-a_0}(\phi = \pi/2)$$

The core size and the higher order terms are computed as described earlier. For simplicity the entire sheet, except the tip, is assumed to be flat.(Figure(2.13))

The large distortions in the vortex ring motion caused by the uneven sheet stretching is reduced by rediscretizing the near wake vortex sheet at every time step to ensure that the vortex elements remain equally spaced. Fink & Soh [6,7] have used this rediscretizing technique for a two-dimensional sheet with good results. The mechanics of such a procedure is as follows: (Ref.[11])

- After each time step a continuous model of the near wake is reconstructed by passing a smooth curve through all of the vortex elements.
- A functional representation of the circulation distribution along the sheet is obtained.
- The total arc length for the sheet is computed using the cubic spline technique and then divided by the number of vortex elements to yield a new

sheet segment length. The vortex elements are now placed at the centers of these segments.

- The vortex rings' strengths are determined from the cubic spline technique.

The cubic spline technique used here tends to bound the interpolated curve and reduces the appearance of wiggles seen in other types of splines.

The geometry and circulation of each portion of the sheet between any two vortex rings are described parametrically by an arc length parameter. For each ring interval i, the near wake vortex sheet is modelled as,

$$r_i = r_i(s) \tag{2.36}$$

$$z_i = z_i(s) \tag{2.37}$$

$$\Gamma_i = \Gamma_i(s) \tag{2.38}$$

where r_i and z_i represent radial and axial components of geometry respectively, Γ_i is the circulation, and s is the arc length along the sheet.

The spline is constructed for a sheet segment between vortex rings i and $i+1$ as shown in Figure(2.14). The first step is to construct a parabola, $g_i(s)$, through points $i-2, i-1$, and i and another parabola, $g_{i+1}(s)$, through the points $i-1, i$, and $i+1$. The interpolated function, $f_i(s)$, is then computed by linearly interpolating the two functions $g_i(s)$ and $g_{i+1}(s)$

$$f_i(s) = g_i(s)\frac{s_{i+1} - s}{s_{i+1} - s_i} + g_{i+1}(s)\frac{s - s_i}{s_{i+1} - s_i} \tag{2.39}$$

where s_i and s_{i+1} are the arc lengths at the points i and $i+1$ respectively. This generates a cubic function for f_i which has a slope at the end points that match the two parabolas g_i and g_{i+1}. The cubic function f_i may represent any of the three dependent variables r_i, z_i, or Γ_i. Repeated over all of the near wake vortex sheet, this procedure yields a continuous function over the entire sheet that has continuous first derivatives and is relatively free of the extraneous curvature seen in some other types of splines.[11]

Using the above technique, the near wake vortex sheet is allowed to convect freely from blade passing time $\phi = \pi/2$ to $\phi = 3\pi/2$. At the end of the time period, the effect of a typical near wake vortex on the rotor blade is small, and therefore, the near wake is amalgamated into n number of intermediate wake vorticies, where $n << N$.

2.2.5 Intermediate Wake Formation

From observations it is clear that the rotor vortex wake filaments convect nonuniformly in space. There are natural tendencies for the vorticies to split the vortex sheet and form distinct amalgamated vorticies. Therefore, at the end of the near wake time period the near wake is divided into n number of amalgamated intermediate vorticies by analyzing the shearing velocities along the sheet. The shearing velocity S_i along the sheet is defined to be,

$$S_i = \vec{u}_i \cdot \vec{t}_i \tag{2.40}$$

where $\vec{u}_i$ is the induced velocity and $\vec{t}_i$ is the unit tangent vector at the vortex location i. The tangent vector $\vec{t}_i$ is computed in a central-differencing fashion such that

$$\vec{t}_i = \frac{\vec{r}_{i+1} - \vec{r}_{i-1}}{|\vec{r}_{i+1} - \vec{r}_{i-1}|}$$

and $\vec{u}_i$ is computed as explained previously.

With this definition of the shearing velocity, it becomes apparent that the vortex sheet will divide itself along the positions where S_i changes signs in such a manner that the vortex sheet at that point is being stretched apart. Therefore, to determine if a point is a dividing point or not, one needs to consider:

- if $S_i < 0$ and $S_j > 0, j > i$, then the vorticies from i to j are travelling away from one another, and this is a point of sheet division.
- if $S_i > 0$ and $S_j < 0, j > i$, then the vorticies from i to j are travelling towards one another, and this is a point of sheet concentration.

During sheet rollup, these conditions do not remain true for all time at a particular point. However, after the rollup is substantially complete, the concentrations of vorticity are widely separated, and the condition for sheet division may be applied with some confidence. We have not made a study of how the sheet division condition varies with time as the wake rolls up. The aforementioned criteria was originally suggested by Yates[22], but Yates' application was to the initial wake of a wing rather than to a later rolled-up configuration. The intermediate wake vorticies are formed by amalgamating vorticies between the points where the vorticies are convecting *away* from each other. The procedure is identical to that of the tip vortex amalgamation; the conservation of circulation determines the strength, the conservation of impulse determines the location, and the conservation of kinetic energy determines the core size.

The vortex rings which constitute the near and intermediate wakes and the vortex cylinders which constitute the far wake are allowed to convect freely from time $\phi = \pi/2$ to $\phi = 3\pi/2$. Then a new bound circulation distribution along the rotor blade is computed. The next section will discuss the lifting-line theory employed here to calculate the blade loading.

2.3 Rotor Blade Load Modelling

Consider the Kutta-Zhukovsky theorem for a blade element

$$\vec{L}(r) = \rho\vec{U}(r) \times \vec{\Gamma}(r) \tag{2.41}$$

where $\vec{L}$ is the lift vector, $\vec{U}$ is the free-stream velocity, and $\vec{\Gamma}$ is the bound vortex.

From Thin-Airfoil theory we obtain,

$$L(r) = \frac{1}{2}\rho U(r)^2 c(r) C_l(r) \tag{2.42}$$

where $c(r)$ here is the section chord length and $C_l(r)$ is the section lift coefficient. The lifting-line theory assumes that at any station along the blade the local flow is two-dimensional, and hence the section lift coefficient may be expressed as,

$$C_l(r) = m_o(\alpha_{geomt}(r) - \alpha_{ind}(r))$$

Here m_o is the lift-curve slope, $\alpha_{geomt}(r)$ is the geometric angle of attack, and $\alpha_{ind}(r)$ is the induced angle of attack. In terms of these definitions, the bound circulation distribution may be expressed as,

$$\Gamma(r) = \frac{1}{2}U(r)c(r)m_o(\alpha_{geomt}(r) - \alpha_{ind}(r)) \tag{2.43}$$

where for a rotor blade,

$$U(r) = \Omega r$$

and Ω is the angular speed of the rotor blade.

For a small angle of attack $\alpha_{geomt}(r)$, the induced angle of attack $\alpha_{ind}(r)$ may be expressed as,

$$\alpha_{ind} = \frac{w(r)}{U(r)} \tag{2.44}$$

where $w(r)$ is the local downwash velocity with a positive sense in the $-z$ direction. Thus if the downwash velocity can be found, the lifting-line theory gives us the bound circulation distribution.

The downwash distribution, $w(r)$, is composed of two distinct components: the downwash contribution from the wake produced by previous blade passages, and the downwash contribution from the attached trailing vortex sheet. That is,

$$w(r) = w_{wake}(r) + w_{tv}(r) \tag{2.45}$$

where $w_{wake}(r)$ is the previous wake contribution and $w_{tv}(r)$ is the attached trailing vortex sheet contribution. $w_{wake}(r)$ is the induced axial component of velocity on the blade computed from the near wake, the intermediate wake, and the far wake; and $w_{tv}(r)$ is the induced axial component of velocity computed from the attached vortex sheet. The attached vortex sheet represents the vortex

transportation from time $\phi = 0$ to $\phi = \pi/2$, and thus consists of $1/2$ vortex rings. The half vortex rings are replaced with complete vortex rings of half circulation. [14] In addition, since the strength of the attached trailing vortex sheet is a function of the bound circulation, and hence is not known apriori, the final bound circulation distribution is computed in an iterative fashion. The details of this process is discussed in the next chapter concerning numerical procedures.

The validity of the current lifting-line theory was tested against an elliptically loaded airfoil. Cosine spacing technique with 250 angular divisions were used to increase the accuracy.[17] The downwash obtained agreed well with the theory; from $r = 0$ to $r = 0.985$, the deviations were less than 1%.

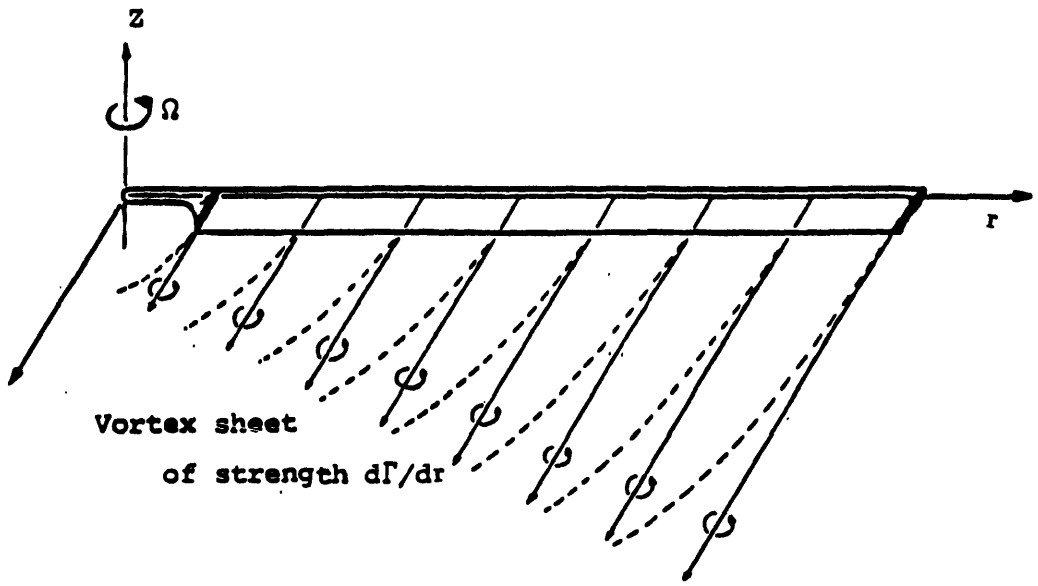


Figure 2.1: Isolated rotor and its trailing vortex sheet

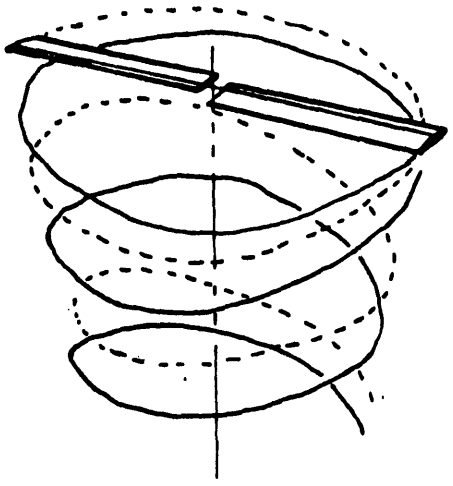


Figure 2.2: Vortex wake of a hovering rotor

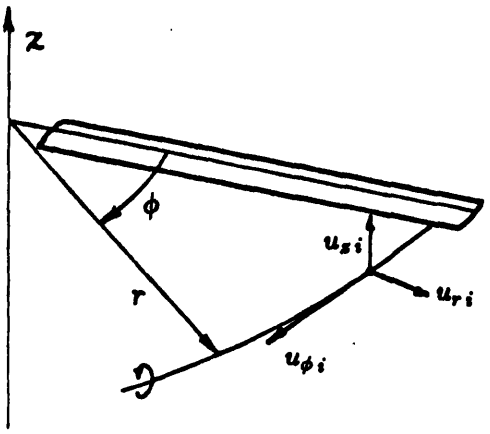


Figure 2.3: Vortex velocity components

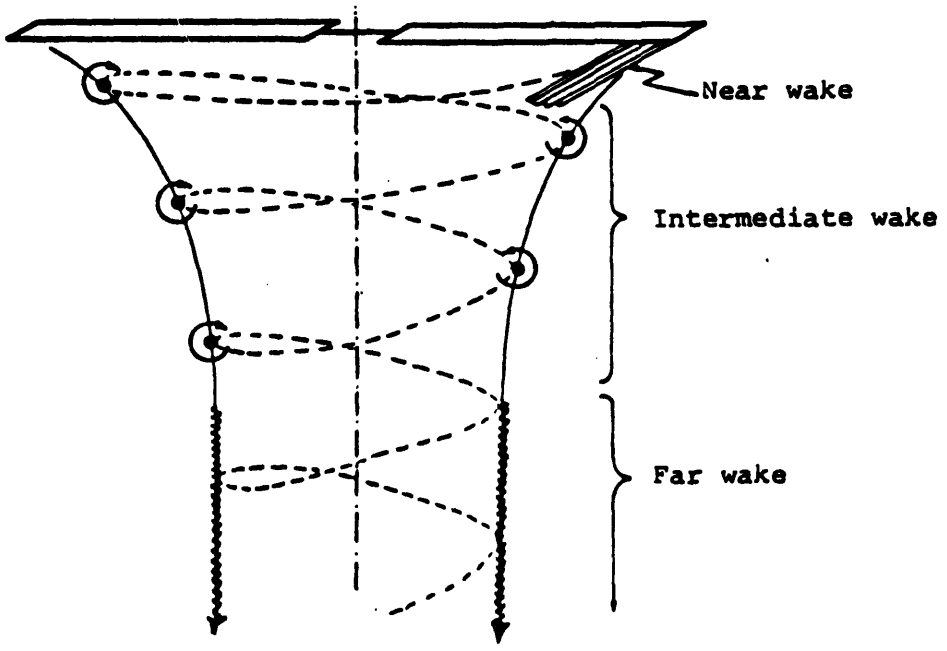


Figure 2.4: Wake divisions in Miller's free wake technique

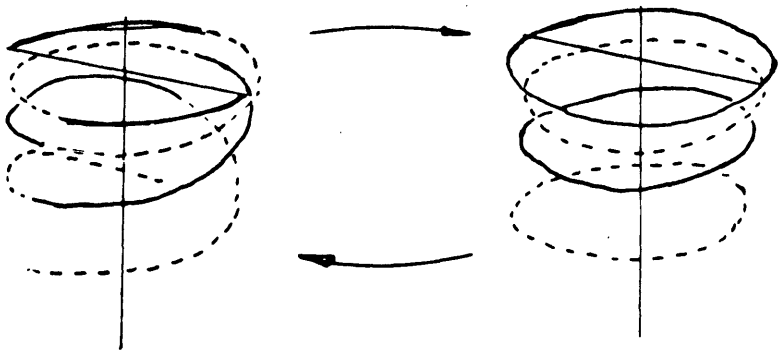


Figure 2.5: Formulation of vortex ring model

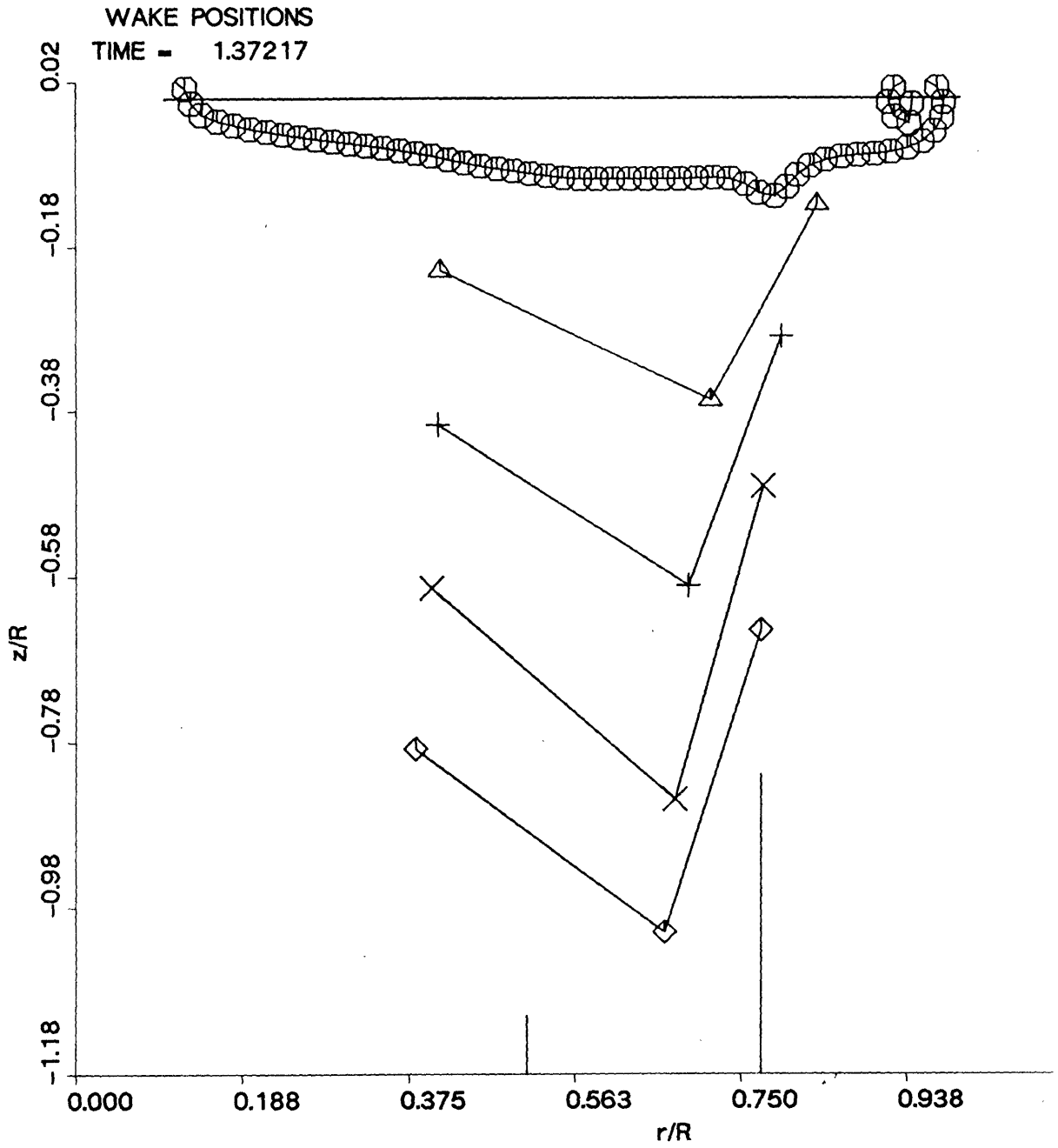


Figure 2.6: A typical wake geometry in motion

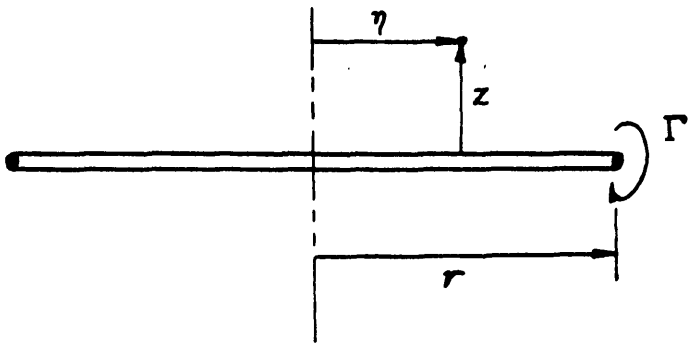


Figure 2.7: Coordinates for vortex ring velocity calculations

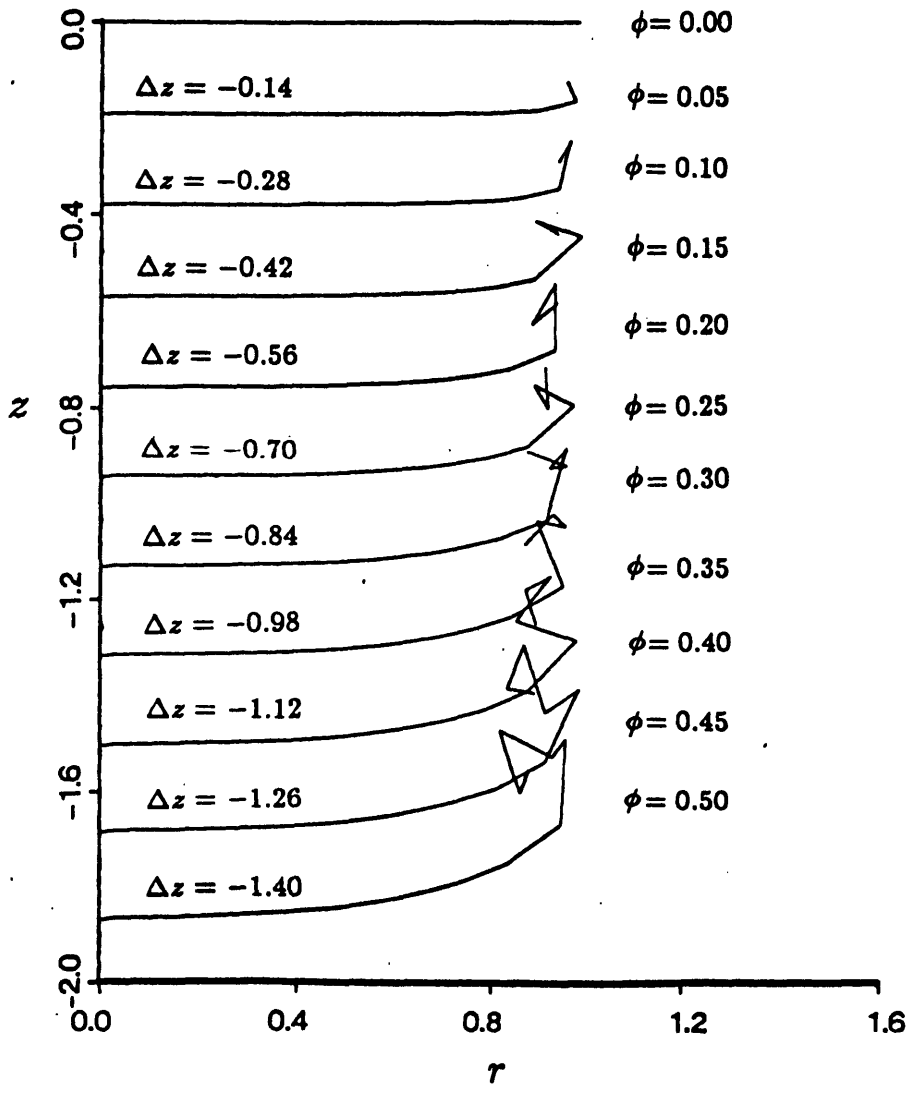


Figure 2.8: Straightforward Application of Discrete Model

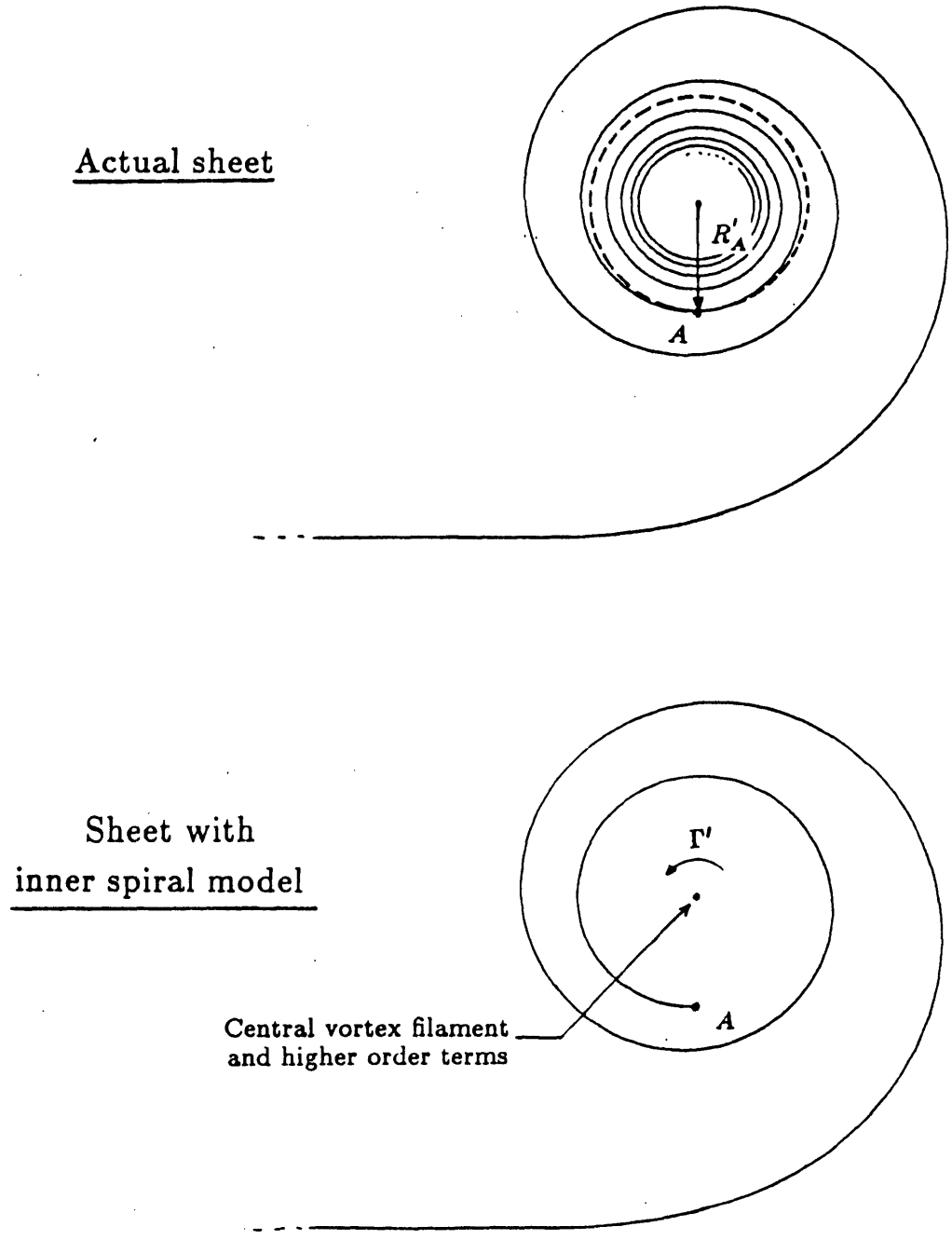
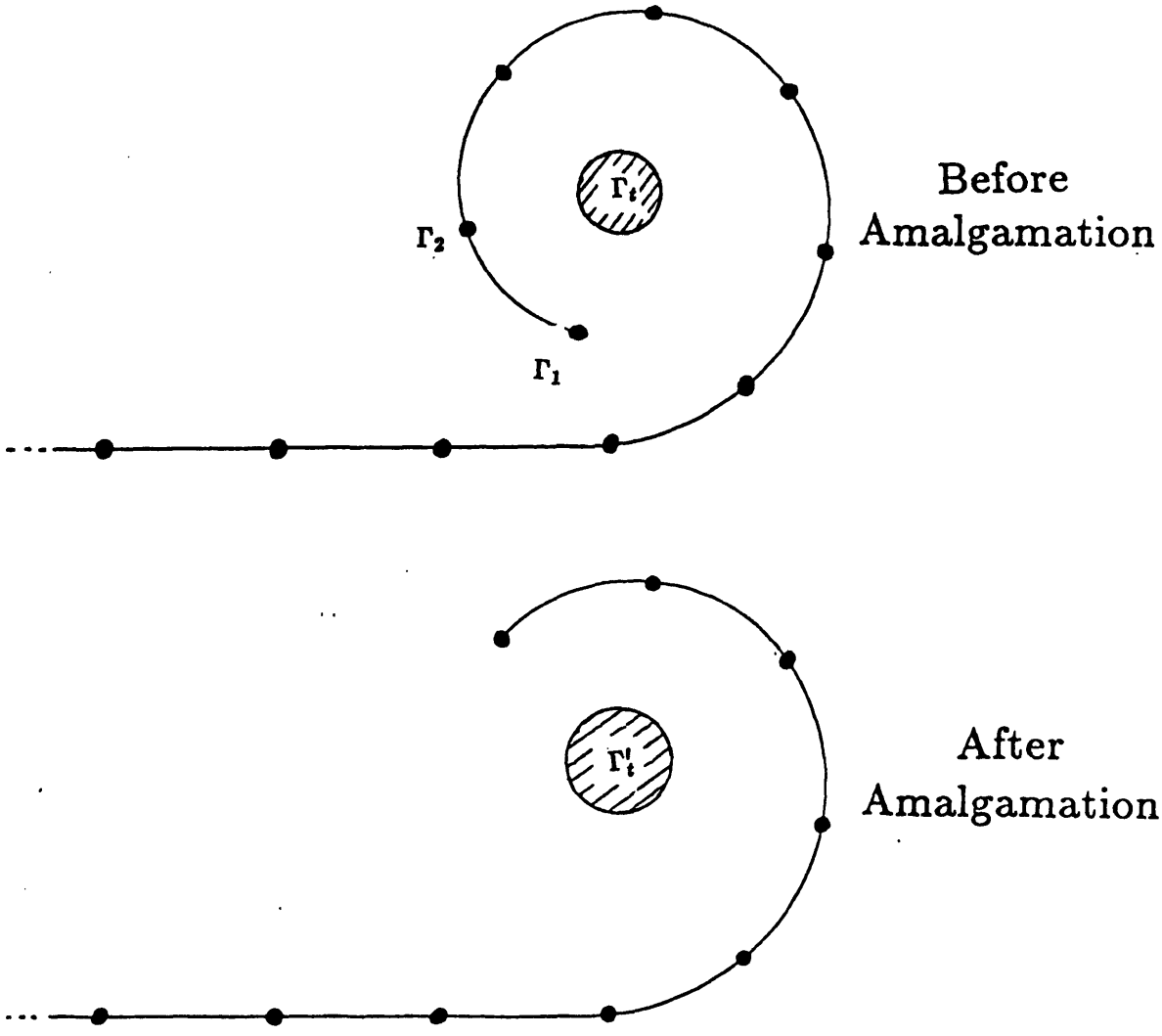


Figure 2.9: Inner spiral model



Γ_1, Γ_2, and Γ_t are amalgamated to form Γ'_t

1. Strength of tip filament found by conserving circulation
2. Location of tip filament found by conserving impulse
3. Tip filament core radius found by conserving kinetic energy

Figure 2.10: Tip vortex amalgamation procedure

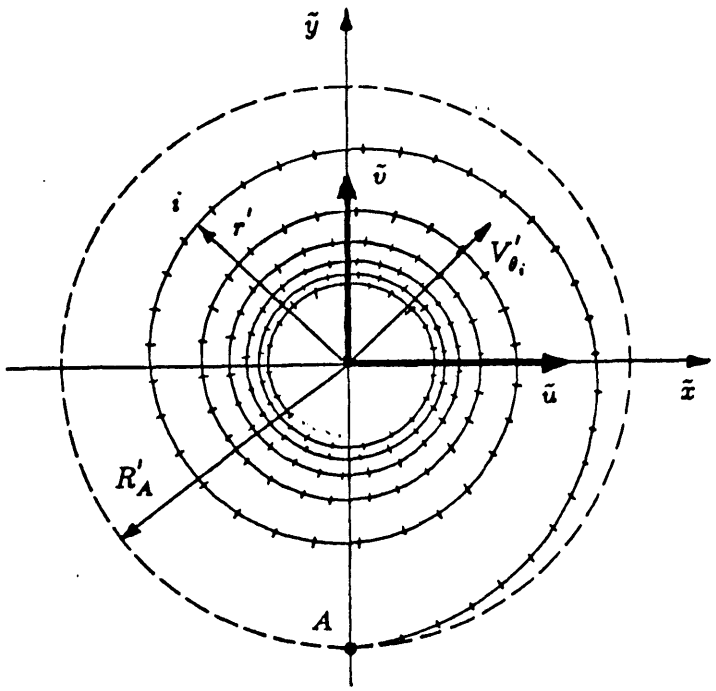


Figure 2.11:Numerical integration of inner spiral model

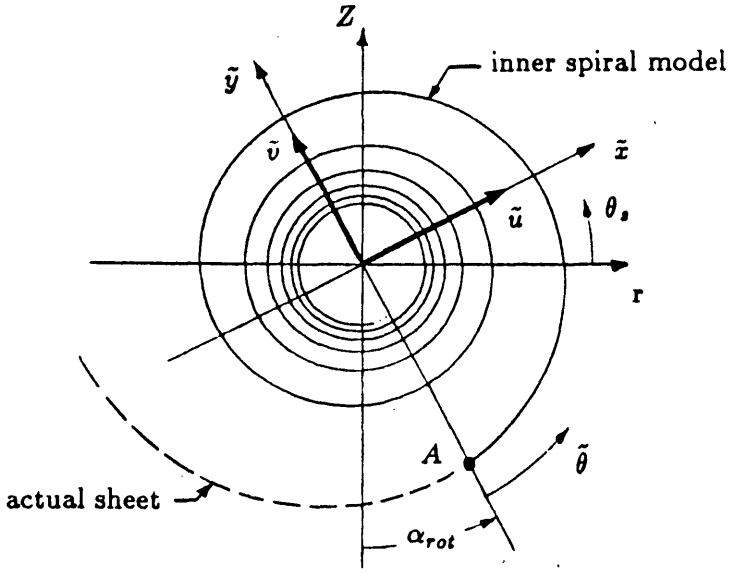
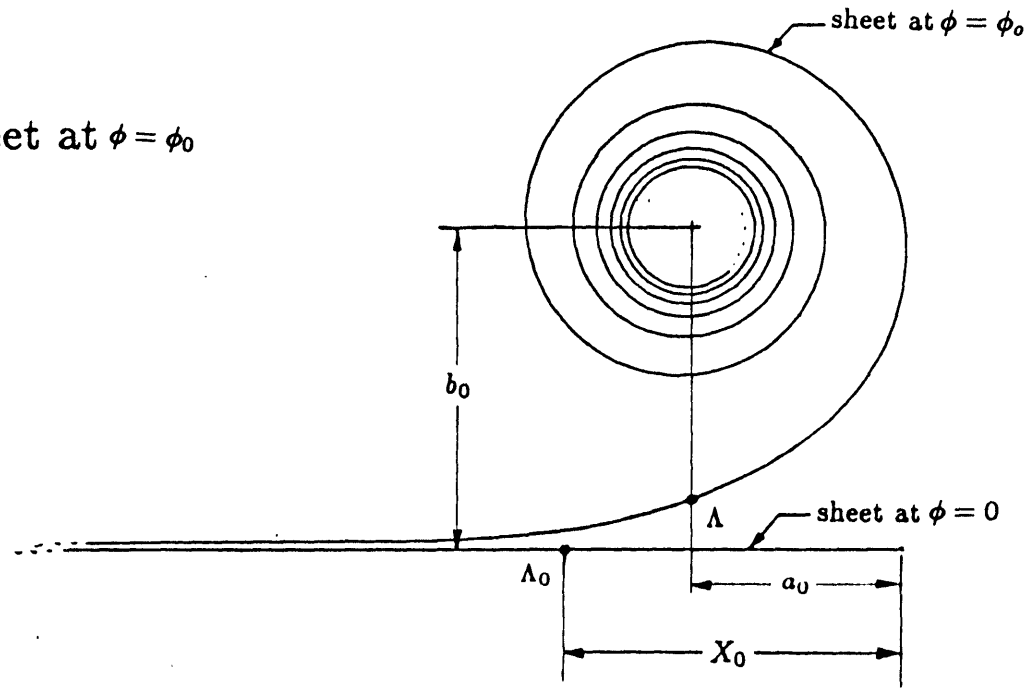
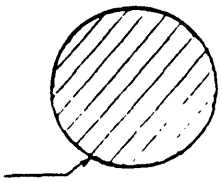


Figure 2.12: Inner spiral model rotation

Sheet at $\phi = \phi_0$

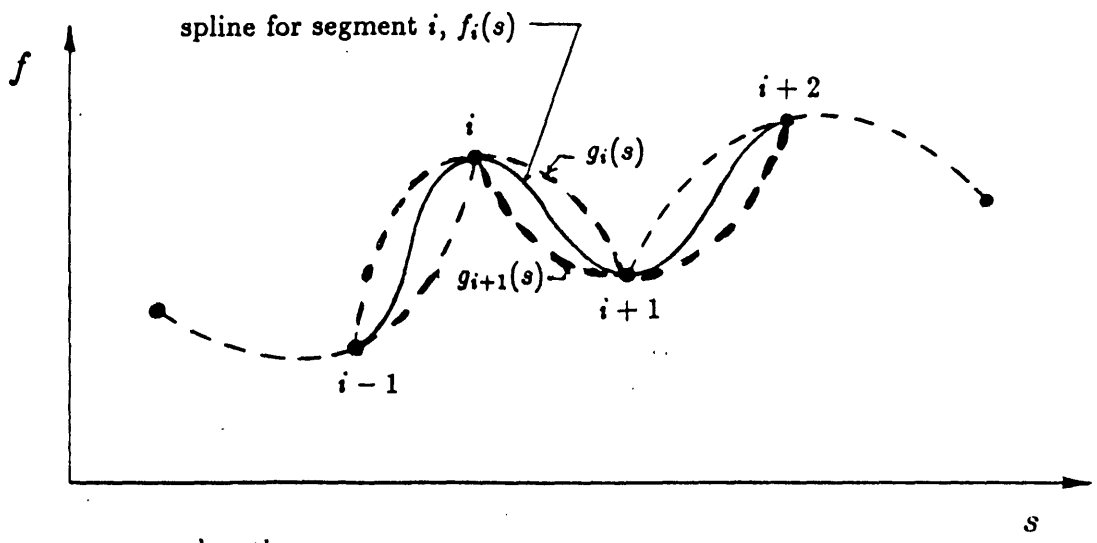


is modelled as:



tip vortex ring with finite core
and higher order terms

Figure 2.13: Initial representation of the near wake



s = arc length

f_i = function being splined, either r_i, z_i, or Γ_i

$$f_i(s) = g_i(s)\frac{s_{i+1} - s}{s_{i+1} - s_i} + g_{i+1}(s)\frac{s - s_i}{s_{i+1} - s_i}$$

$$\begin{aligned} g_i(s) &= \text{parabola through points } i-1,\ i,\ i+1 \\ &= a_0 + a_1(s - s_i) + a_2(s - s_i)^2 \end{aligned}$$

$$\begin{aligned} g_{i+1}(s) &= \text{parabola through points } i,\ i+1,\ i+2 \\ &= b_0 + b_1(s - s_{i+1}) + b_2(s - s_{i+1})^2 \end{aligned}$$

Figure 2.14: Cubic spline for sheet rediscretization

Chapter 3
Numerical Methods

3.1 Time Integration

The new positions of the vortex rings in the near and intermediate wakes are calculated by numerical integrations of the velocity vectors shown in the previous chapter. For each vortex ring i a total induced velocity from all the vorticies, including itself, is computed:

$$\dot{r}_i = \sum_{j=1}^{N} u_{r\,j} \tag{3.1}$$

$$\dot{z}_i = \sum_{j=1}^{N} u_{z\,j} \tag{3.2}$$

The new r_i and z_i position components are computed by integrating the above two equations. Two different types of the integration procedures were considered. The first is just the simple Euler scheme, i.e. a first order approximation to a Taylor Series.

$$r_i^{n+1} = r_i^n + \dot{r}_i^n \, \Delta\phi \tag{3.3}$$

$$z_i^{n+1} = z_i^n + \dot{z}_i^n \, \Delta\phi \tag{3.4}$$

where $\Delta\phi$ is the time step. In addition, the four stage Runge-Kutta time stepping sheme was also considered, [11]

$$r_i^{n+1} = r_i^n + \frac{k_0 + 2k_1 + 2k_2 + k_3}{6} \tag{3.5}$$

$$z_i^{n+1} = z_i^n + \frac{l_0 + 2l_1 + 2l_2 + l_3}{6} \tag{3.6}$$

where

$$k_0 = \Delta\phi \dot{r}_i(r_i^n, z_i^n)$$

$$k_1 = \Delta\phi \dot{r}_i(r_i^n + k_0/2, z_i^n + l_0/2)$$

$$k_2 = \Delta\phi \dot{r}_i(r_i^n + k_1/2, z_i^n + l_1/2)$$

$$k_3 = \Delta\phi \dot{r}_i(r_i^n + k_2, z_i^n + l_2)$$

and

$$l_0 = \Delta\phi \dot{z}_i(r_i^n, z_i^n)$$

$$l_1 = \Delta\phi \dot{z}_i(r_i^n + k_0/2, z_i^n + l_0/2)$$

$$l_2 = \Delta\phi \dot{z}_i(r_i^n + k_1/2, z_i^n + l_1/2)$$

$$l_3 = \Delta\phi \dot{z}_i(r_i^n + k_2, z_i^n + l_2)$$

For an accurate representation of the near wake it is essential that the numerical integration methodolgy conserve the three invariants: circulation, impulse,

and kinetic energy. The Euler scheme is attractive because of its simplicity, however, it is also quite inaccurate for large time steps. Figure (3.1) and Figure (3.2) show the change in fluid impulse with time as predicted by the two methods. The Euler scheme requires a time step of $O(0.001)$ to acheive the same accuracy of conservation of impulse as the Runge-Kutta scheme does in a time step of $O(0.01)$. Since the Runge-Kutta scheme requires four times as many calculations per time step as the Euler scheme, but is more accurate by $O(10)$, it was decided that the Runge-Kutta scheme was more efficient; all time integrations are done in a Runge-Kutta fashion.

3.2 Iteration Procedure

The convergence criteria must be met for both rotor blade loading and its related wake geometry. Furthermore, since the flow is steady, given a particular blade bound circulation distribution, there must exist a unique force-free wake geometry associated with it. To avoid numerical difficulties the two quantities are converged separately. That is, the wake rollup is converged before the blade loading is updated to respond to the changed blade downwash.

The vortex ring trajectories are calculated from time $\phi = \pi/2$ to $\phi = 3\pi/2$, at which point a new wake is created. The convergence criteria is now applied to the wake geometry; if the wake geometry has converged, a new rotor blade loading and its associated attached trailing vortex sheet is computed, if not, then no bound circulation computations are done, and the old trailing vortex sheet is again used. The vortex rings are now allowed to convect until time $\phi = 5\pi/2$ at which point the convergence criteria for the wake geometry is again applied. The procedure continues similiarly until a convergence in both the rotor blade

loading and its wake is achieved.

Thus, there are a total of three distinct iterations performed,

- Lifting-line theory and matching the total downwash distribution with the blade loading
- Bound circulation
- Wake geometry

3.2.1 L-L theory and $\Gamma \longleftrightarrow w$ matching

From section(2.3) the bound circulation $\Gamma(r)$ is given as,

$$\Gamma(r) = K_1(r)\,(K_2(r) - \frac{w(r)}{U(r)}) \tag{3.7}$$

where

$$K_1(r) = \frac{1}{2}U(r)c(r)m_o$$

and

$$K_2(r) = \alpha_{geomt}(r)$$

But $w(r) = w_{wake}(r) + w_{tv}(r)$ and is a function of $\Gamma(r)$. Thus $\Gamma(r)$ cannot be solved directly and an underrelaxation method is employed to iterate between $\Gamma(r)$ and $w(r)$. At each iteration (except the first) $w_{tv}(r)$ is computed from the attached trailing vortex sheet, a new $\Gamma(r)$ is computed, followed by the trailing

vortex sheet strength; the process is then repeated until a convergence criterion is satisfied, i.e.

$$\gamma^k(r) \longrightarrow w^{k+1}(r)$$

$$w^{k+1}(r) \longrightarrow \Gamma^{k+1}(r)$$

$$\Gamma^{k+1'}(r) = \Gamma^{k'}(r) + \omega(\Gamma^{k+1}(r) - \Gamma^{k'}(r))$$

$$\Gamma^{k+1'}(r) \longrightarrow \gamma^{k+1}(r)$$

where $\gamma(r)$ is the strength of the trailing vortex sheet, k is the iteration level and $\omega = 0.4$ is the relaxation parameter. The iteration begins with $\gamma^0(r) = w^0_{iv}(r) = 0$ and ends when

$$\frac{\Gamma^{k+1'}(r) - \Gamma^{k'}(r)}{\Gamma^{k+1'}(r)} \leq 0.001 \tag{3.8}$$

3.2.2 Bound Circulation Iteration

Miller [14] showed that to eliminate large oscillations in the bound circulation computations, it is necessary to employ an underrelaxation scheme. That is,

$$\Gamma^{new}(r) = \Gamma^{old}(r) + \omega(\Gamma'(r) - \Gamma^{old}(r)) \tag{3.9}$$

where $\Gamma'(r)$ is the value computed from the lifting-line theory just shown in the previous section, and $\omega = 0.1$ is the relaxation parameter. The iteration ends when

$$\frac{\Gamma^{new}(r) - \Gamma^{old}(r)}{\Gamma^{new}(r)} \leq 0.001 \tag{3.10}$$

3.2.3 Wake Geometry Iteration

It became apparent during computations that without some sort of dampening added to the vortex trajectories, the wake geometries would become chaotic with a large number of vortex pairings. To counteract this tendency an underrelaxation technique was added to the geometry calculations for the intermediate vortex rings,

$$r_i^{new} = r_i^{old} + \omega(r_i' - r_i^{old}) \tag{3.11}$$

$$z_i^{new} = z_i^{old} + \omega(z_i' - z_i^{old}) \tag{3.12}$$

where r_i' and z_i' are the integrated values and $\omega = 0.1$ is the relaxation parameter. Again the iteration ends when

$$\frac{r_i^{new} - r_i^{old}}{r_i^{new}} \leq 0.001 \tag{3.13}$$

and

$$\frac{z_i^{new} - z_i^{old}}{z_i^{new}} \leq 0.001 \tag{3.14}$$

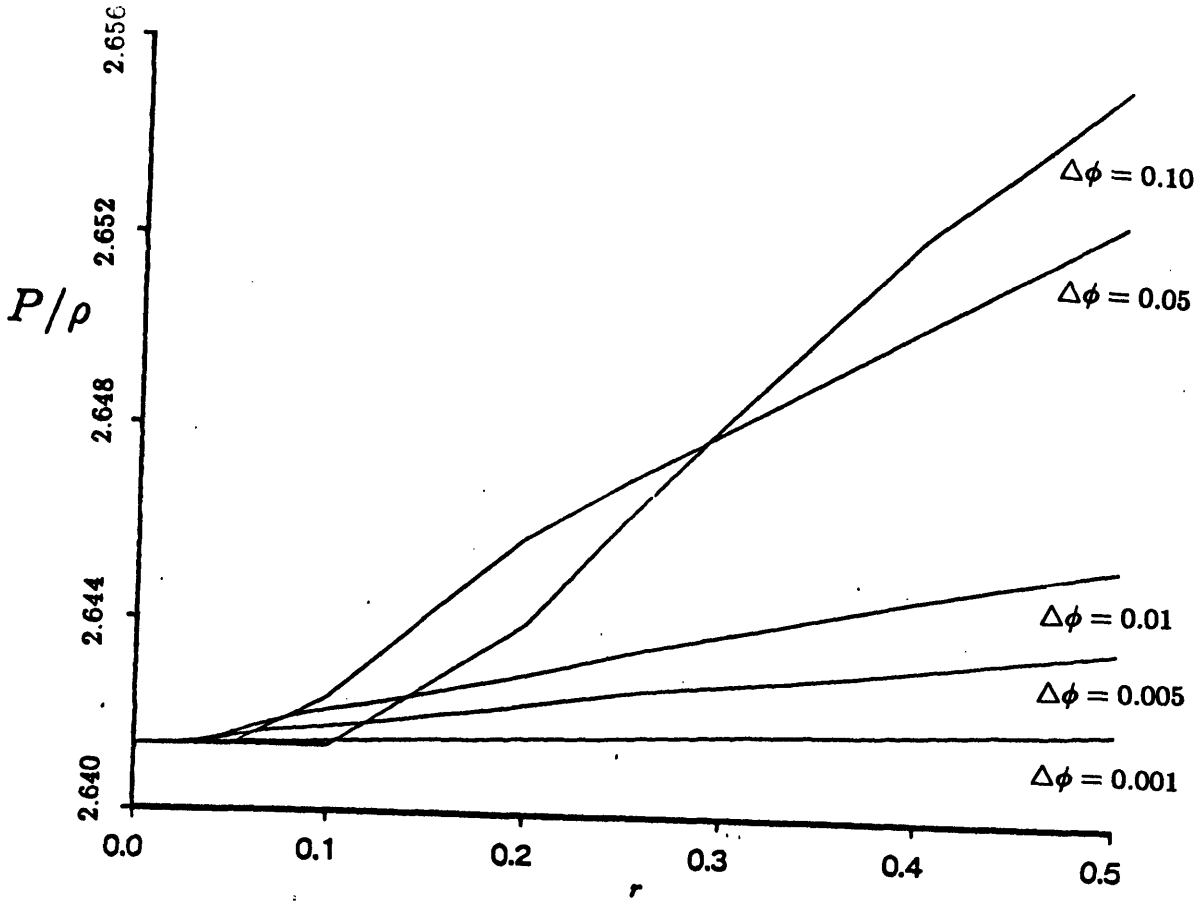


Figure 3.1: Fluid Impulse for the Euler Scheme

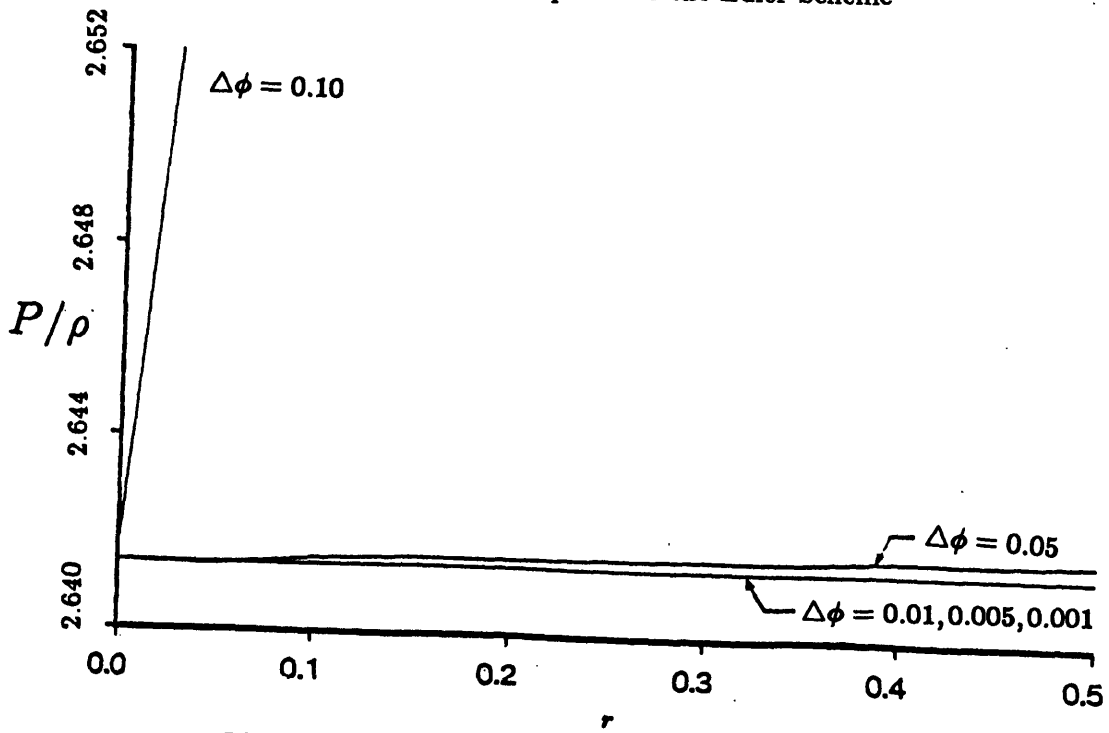


Figure 3.2: Fluid Impulse for the Runge-Kutta Scheme

Chapter 4
Application

4.1 Rotor Blade Geometry

The validity of the method is demonstrated on a straight two-bladed rotor as described below:[9]

Blade radius, R	1.045m
Angular speed, Ω	73.3 rad/sec
Chord, c/R	0.0729
Solidity, σ	0.0464
Twist	-11^o

where solidity is

$$\sigma = \frac{2c}{\pi R}$$

and c is the dimensional chord length.

The distances and velocities were nondimensionalized as follows,

$$\vec{x} = \frac{\vec{x}}{R}$$

$$\vec{u} = \frac{\vec{u}}{\Omega R}$$

and thus nondimensional time is the blade passing angle ϕ.

4.2 Initial Wake Placement and Input Values

Since the flow is steady, the simulation at $\phi = \pi/2 + \phi_o$ must start within a prescribed wake. The approach here is to use Miller's [14] final blade bound circulation distribution and wake positions as the initial conditions. See Figure(4.1) and Figure(4.2).

The input parameters for the simulation were as follows:

$\Delta x_{max} = 0.01$	Maximum translation in one time step
$\Theta_{max} = 2\pi$	Near wake amalgamation angle
$N = 60$	Number of near wake vorticies
$\Lambda_o = 0.01$	Outer portion to be lumped at $\phi = \pi/2 + \phi_o$

At each time step maximum velocity, U_{max}, is computed; the incremental time $\Delta\phi$ then follows as,

$$\Delta\phi = \frac{U_{max}}{\Delta x_{max}}$$

The choice of $\Delta x_{max} = 0.01$ yields an average value of $\Delta\phi_{avg} \approx 0.05$, which is about the maximum time step allowed while conserving impulse/kinetic energy.

Θ_{max} of 2π is equivalent to one spiral turn amalgamation; more spiral turns unnecessarily increase the computational time. The choice of $N = 60$ and $\Lambda_o = 0.01$ was reached at after some numerical experimentation. It is felt that the given choices would allow a sufficient resolution without excessive sacrifices in computational speed.

One percent of the outer portion of the near wake rolled into the tip vortex filament results in a tip filament strength of about 30% of the total near wake strength (of one sign). The large value is a result of the high concentration of vorticity near the edge of the sheet. As determined using the methods described in section(2.2.4) the initial tip vortex location is found to be $r = 0.987$ and $z = 0.011$ with a corresponding time $\phi_o = 0.00963$.

4.3 Results

The computed bound circulation distribution and its related wake are compared with Ballard's [2] published results in Figure(4.3) and Figure(4.4). The computed circulation distribution compares relatively well with the experimental values, although the peak value seems a bit excessive and outboard of the empirical result. The cause of the discrepancies near the rotor tip cannot be ascertained exactly, but three possible causes are:

- The three-dimensionality of the flow around the tip caused a breakdown in the lifting-line theory.
- The non-linear interactions of the tip vortex formation on the blade prior to the trailing edge.

- Relatively large viscous effects due to large velocity gradients present near the tip affected the experimental results.

One possible solution for the first cause would be to use a lifting-surface technique.

The vortex positions in the wake also compare reasonably well with the experimental values. The flagged values shown in Figure(4.4) are the experimentally observed tip vortex locations. It should be noted that the inboard vorticies have not yet been observed. The computed tip vortex locations are outboard of the experimental values, and this is to be expected since the peak in the loading is also outboard of the observed values.

The current method required 191 iterations to converge. To demonstrate the process of the intermediate wake formation, the vorticies were convected for one additional iteration from time $\phi = 191\pi$ to time $\phi = 192\pi$. The initial conditions for this iteration are shown in Table(4.1) and Figure(4.5) and the final positions are shown in Figure(4.4). At this point, the shear velocity, S_i, were computed along the near wake to determine the dividing points for the intermediate wake. From Figure(4.6) and Figure(4.7) one can see that the sheet divides itself at three different positions. Vorticies from arc length ≈ 0 to arc length ≈ 0.6 form a root vortex, vorticies from arc length ≈ 0.6 to arc length ≈ 1.1 form a center vortex, and remaining vorticies out to arc length ≈ 1.4 form the tip vortex. This formation of the first level of intermediate wake is shown in Figure(4.8), where the amalgamation process was previously described in Chapter 2 to conserve the three invariants: circulation, impulse, and kinetic energy.

By applying Donaldson's rollup criteria to the loading curve one should conclude that three vorticies are necessary to transport the trailing vorticity. The direct rollup calculations also show a formation of three vorticies. The tip vortex strength shows an amalgamation of primarily negative vorticity. Donaldson's tip vortex strength is computed by summing only the negative vorticity in Table(4.1) which leads to $\Gamma_{tip} = -0.023428$. The current analysis indicates a tip vortex strength of $\Gamma_{tip} = -0.022220$. There is a difference of approximately 5% between the values, indicating that the tip vortex is somewhat weaker than Donaldson's model would indicate due to the rollup of counter-signed vorticity.

Figure(4.9) shows the distribution of vorticity vs. arc length. From arc length ≈ 0.9 to arc length ≈ 1.4 there is little or no vorticity, and one can conclude that the strong tip vortex has essentially separated from the remainder of the sheet. In addition, the figure shows two distinct areas of vorticity concentrations inboard of the tip. The tip and the two inboard vorticies combined would, infact, represent approximately 85% of the total vorticity (of both signs) in the sheet. Therefore, replacing the vortex sheet by a small number of concentrated vorticies should be an accurate representation.

4.4 Computer Usuage

All calculations were done on the MIT-CFD Alliant FX/8 vector computer. In a full vector/concurrency mode each iteration required approximately 10 minutes of CPU time for completion. The present method required 191 iterations for convergence, i.e. about 32 hours of CPU time. Such a large requirement in computer time is a deterrent for the use of the present method, especially since there are other methods [14,15,13,21] which require far fewer hours of CPU

usuage with comparable results in bound circulation distribution. However, the methodolgy used here was developed to assess the effect of free wake rollup on rotor load predictions and also to validate some of the simpler models of wake geometry as Miller's[14,15] method rather than as an engineering tool. It should be noted that all attempts were made to make the computer code as efficient as possible, including considerable manual vectorization.

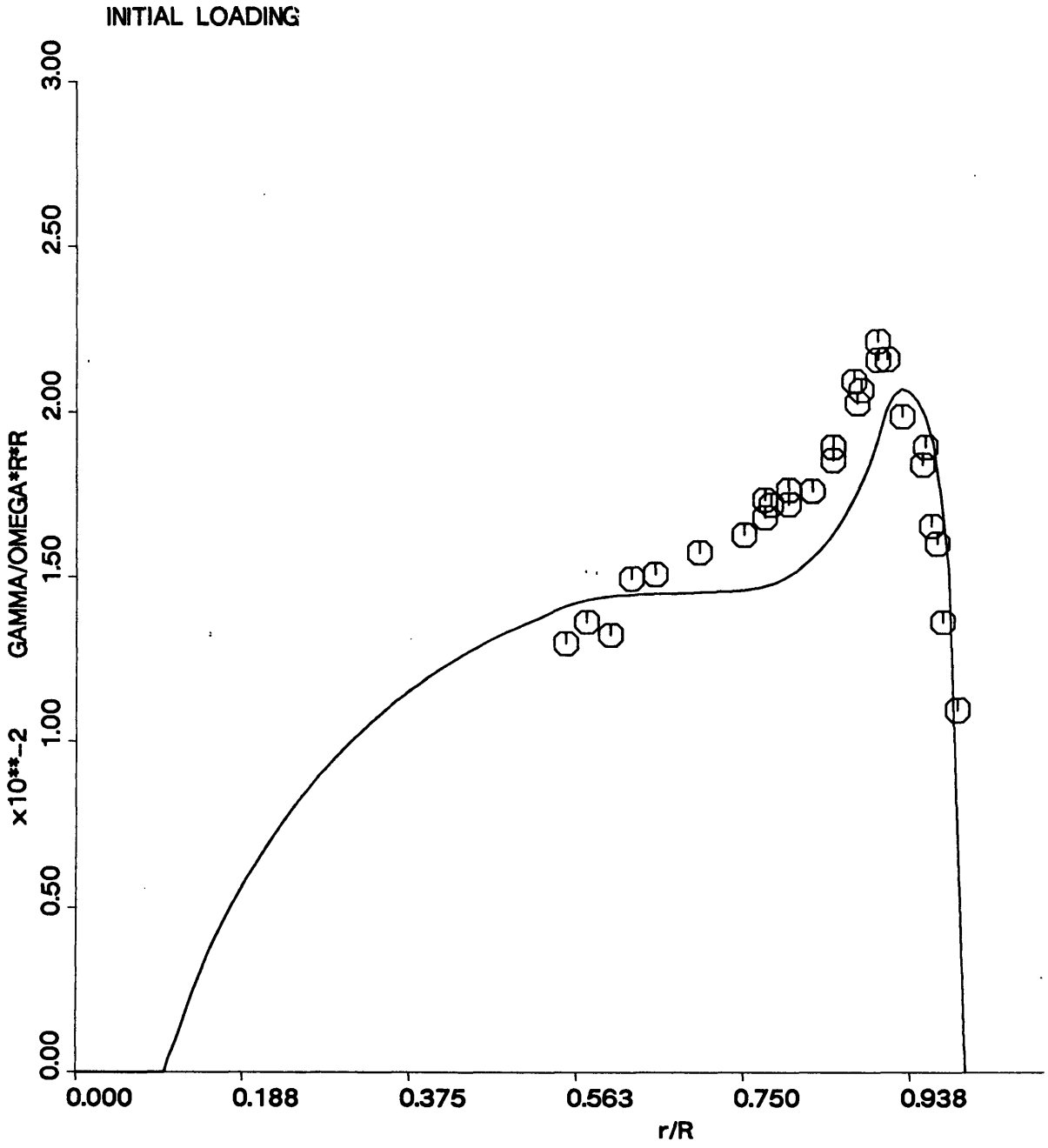


Figure 4.1: Initial load distribution from Miller

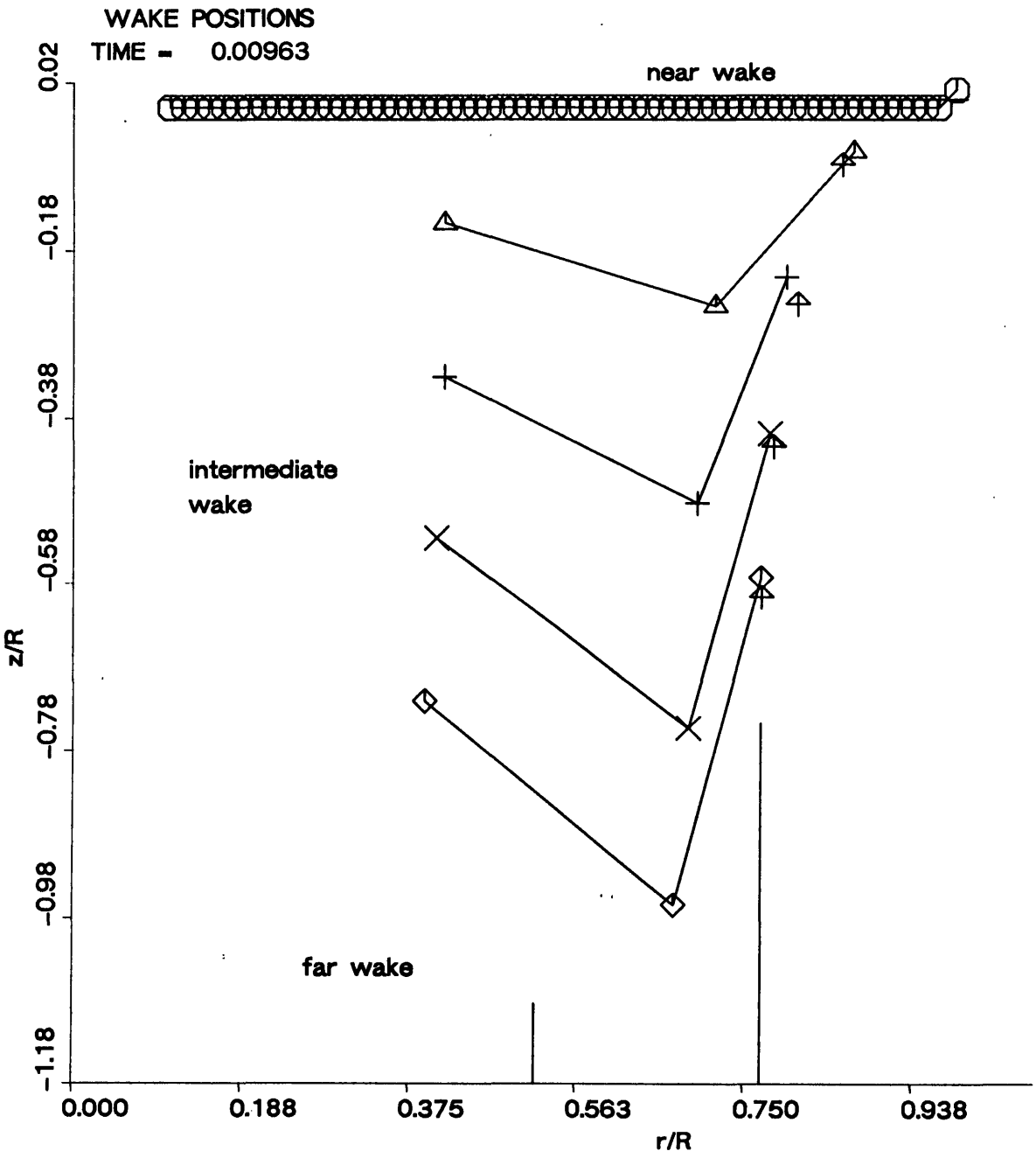


Figure 4.2: Initial wake geometry

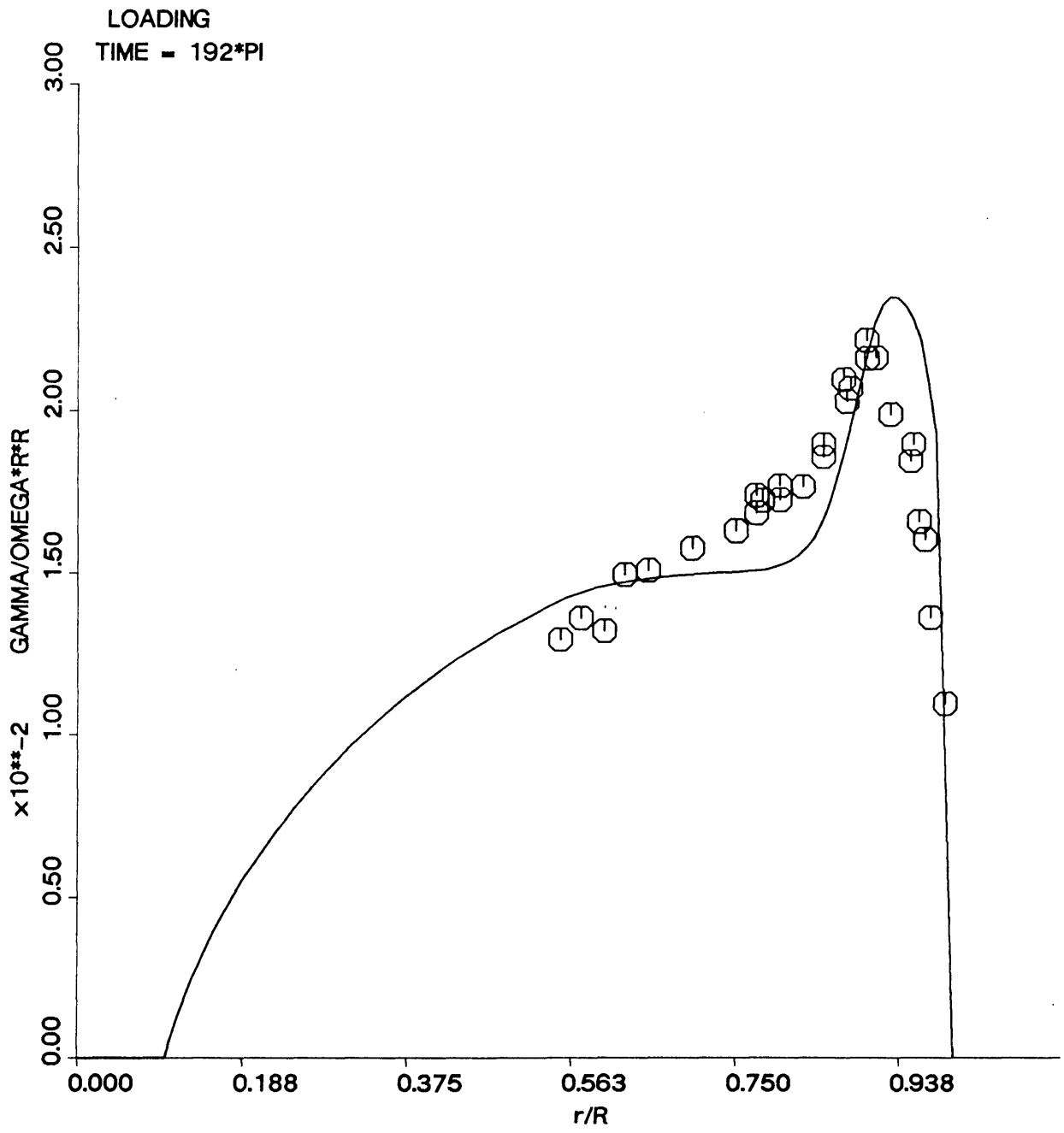


Figure 4.3: Final load distribution

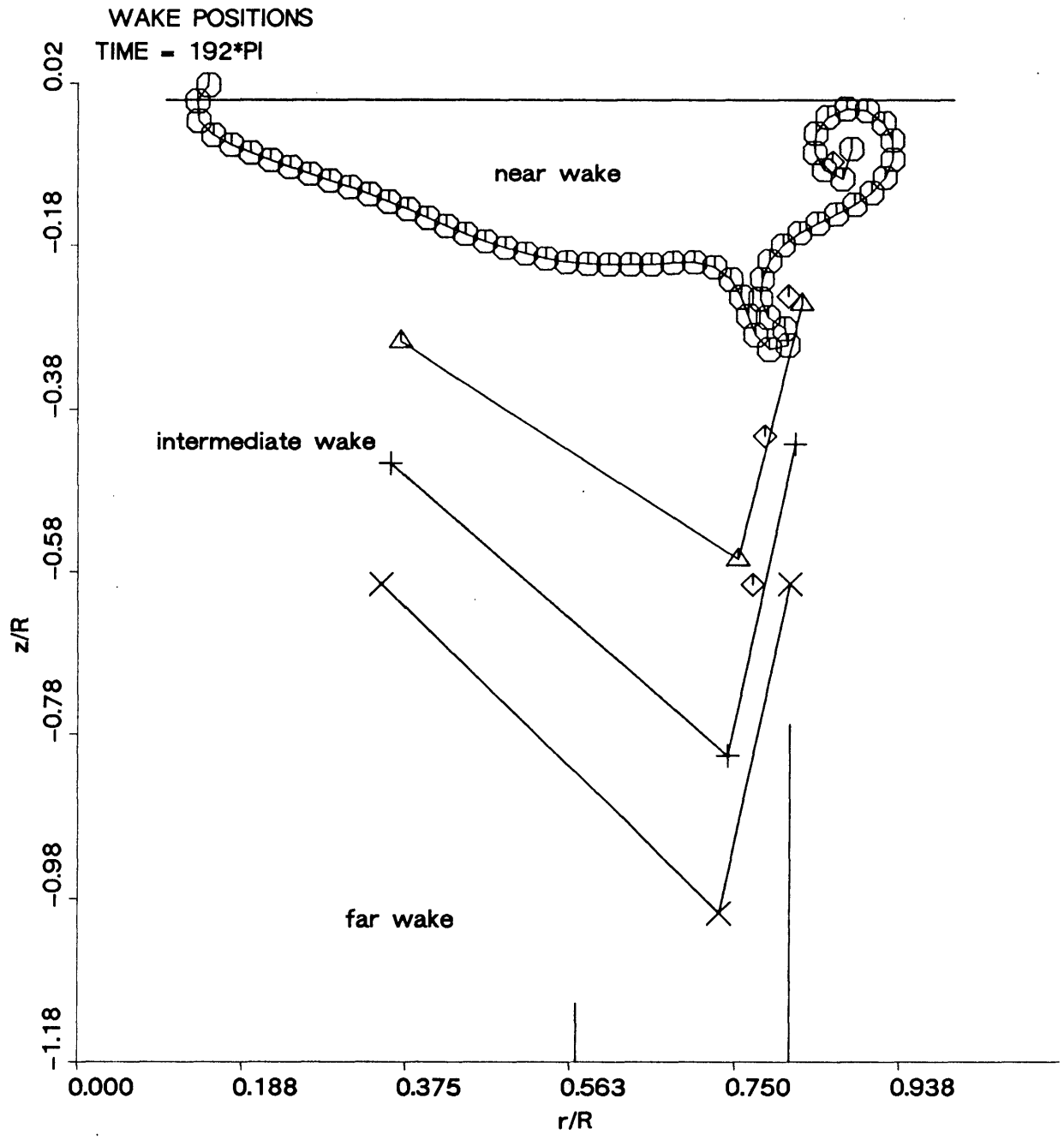


Figure 4.4: Final wake geometry

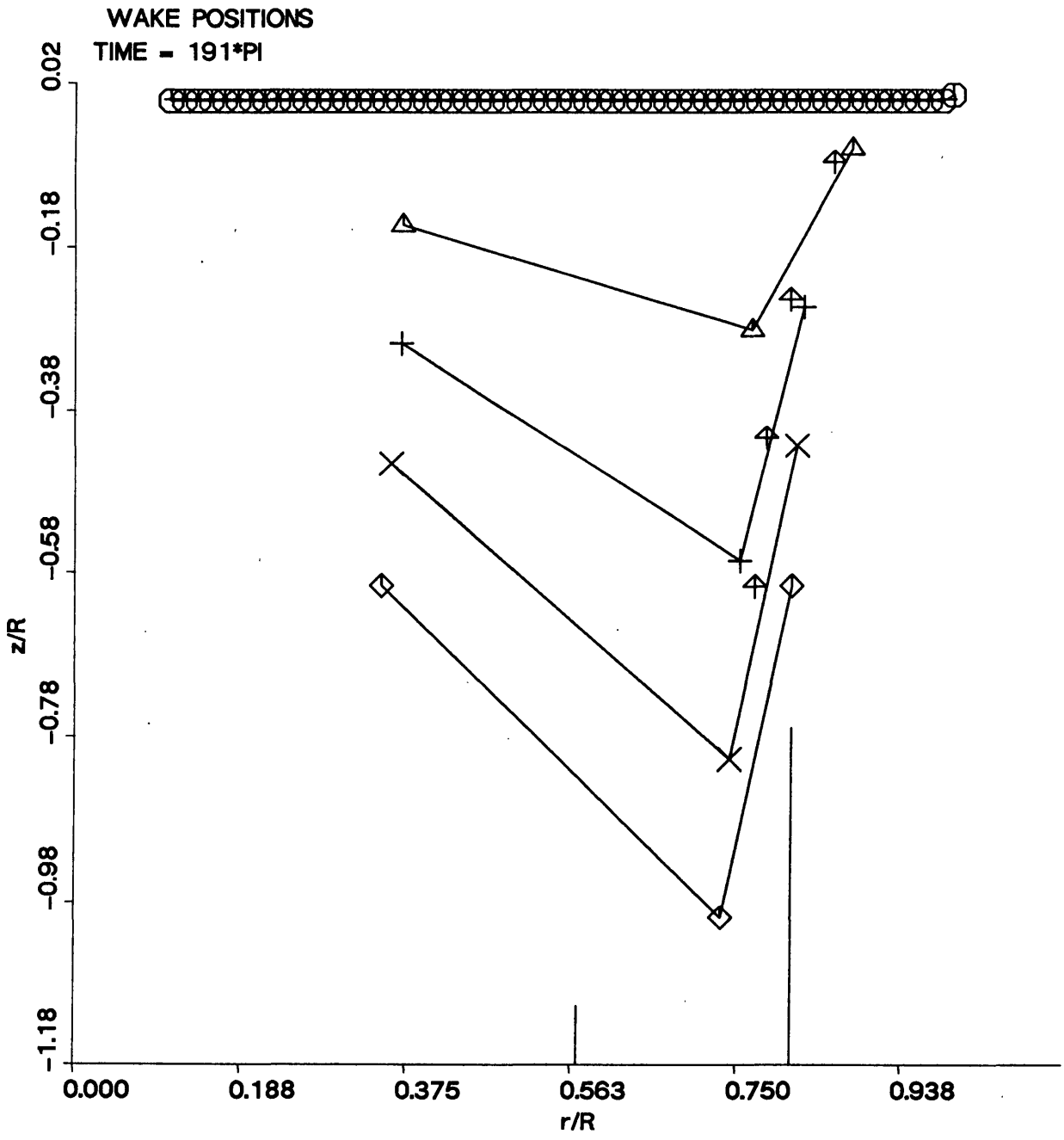


Figure 4.5: Initial wake geometry for final iteration

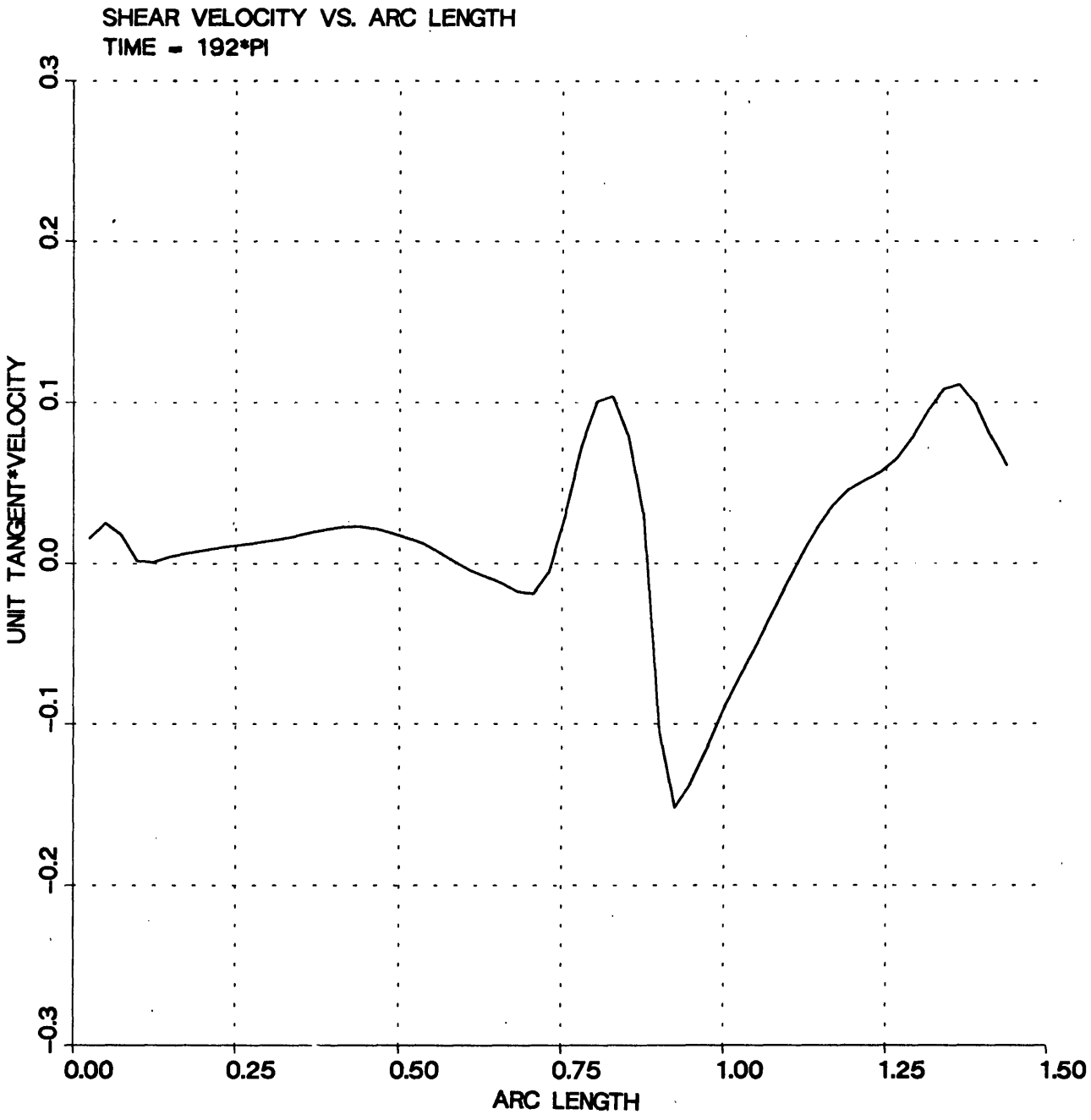


Figure 4.6: Shear strength distribution for near wake

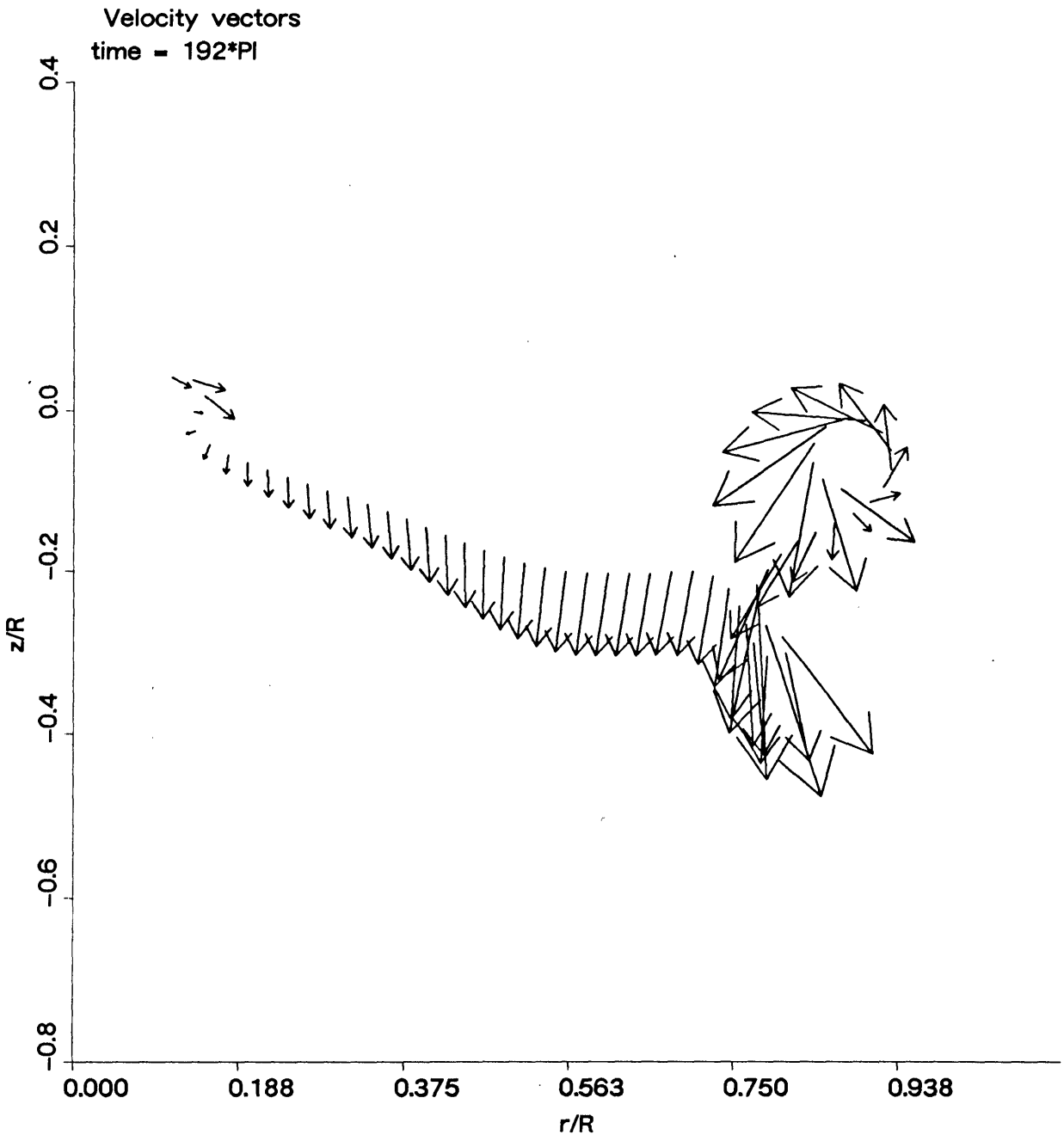


Figure 4.7: Velocity vectors for near wake

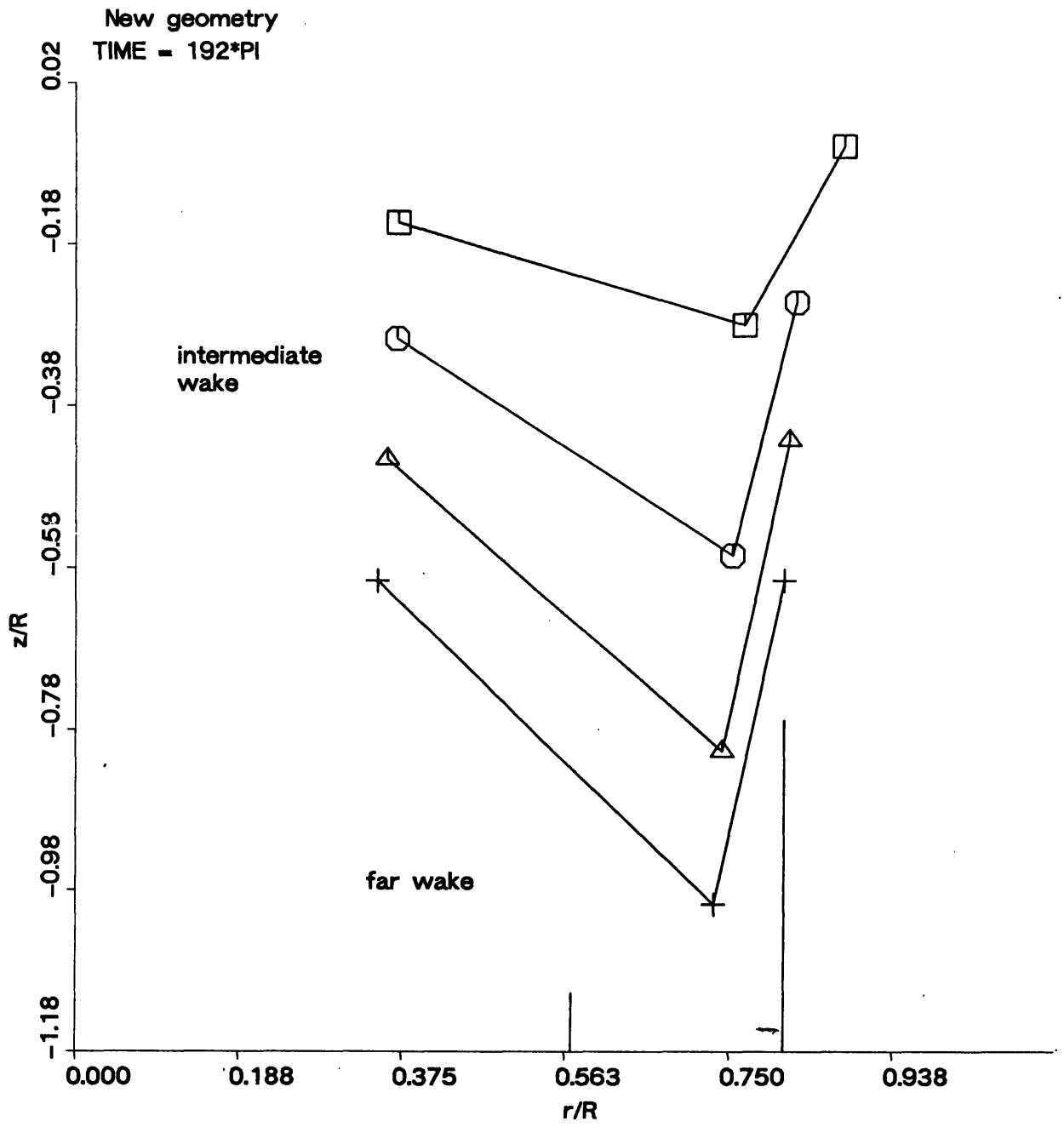


Figure 4.8: New intermediate wake formation

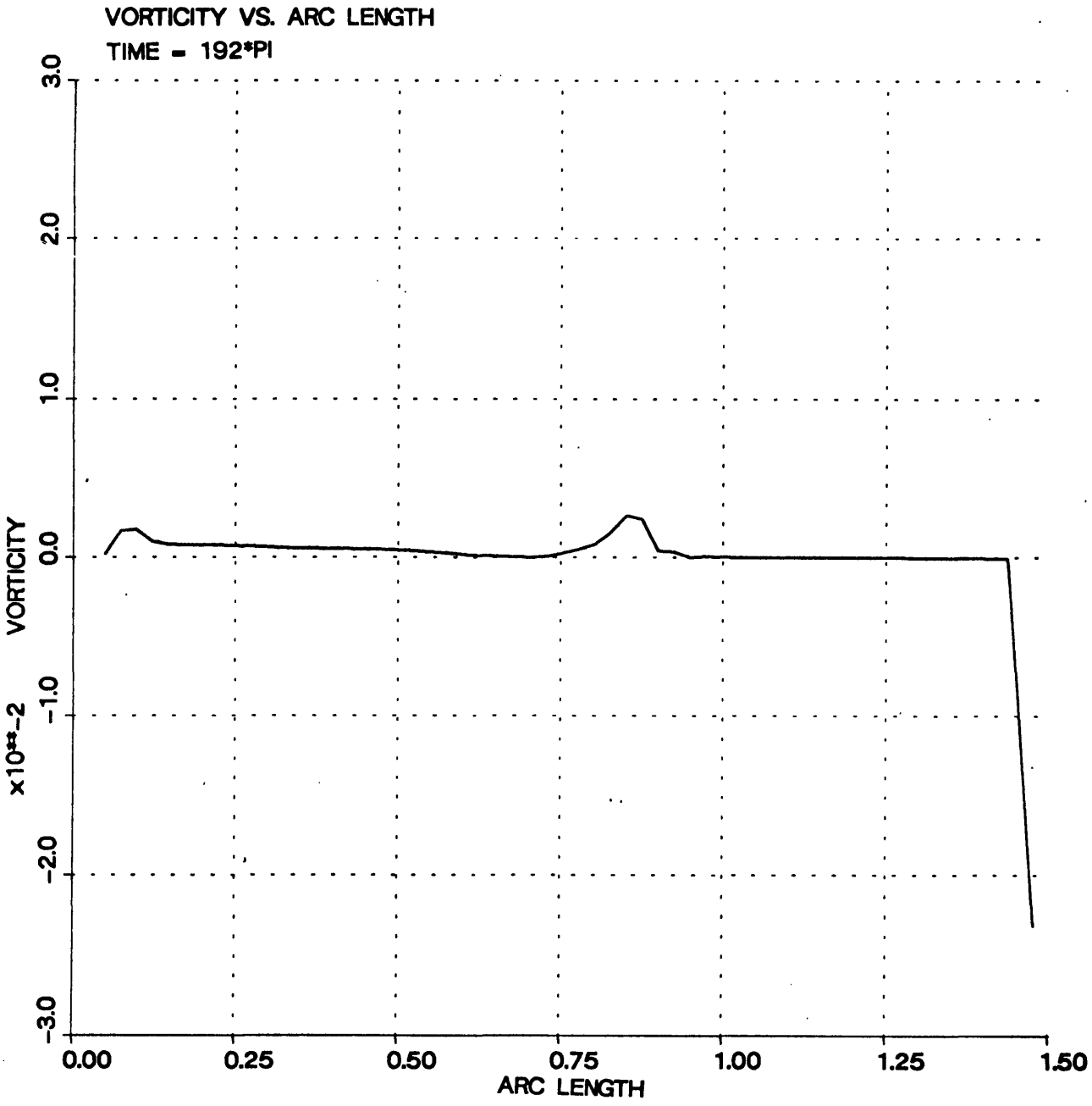


Figure 4.9: Vorticity distribution in near wake

TIME = 191*PI

INITIAL VALUES

I	RADIUS	HEIGHT	GAMMA
0	0.100000E+00	-0.190906E-02	0.000000E+00
1	0.107555E+00	-0.190906E-02	0.132221E-02
2	0.122665E+00	-0.190906E-02	0.110944E-02
3	0.137775E+00	-0.190906E-02	0.969350E-03
4	0.152886E+00	-0.190906E-02	0.839926E-03
5	0.167996E+00	-0.190906E-02	0.742824E-03
6	0.183106E+00	-0.190906E-02	0.660581E-03
7	0.198216E+00	-0.190906E-02	0.603929E-03
8	0.213326E+00	-0.190906E-02	0.568352E-03
9	0.228436E+00	-0.190906E-02	0.539236E-03
10	0.243547E+00	-0.190906E-02	0.513880E-03
11	0.258657E+00	-0.190906E-02	0.488983E-03
12	0.273767E+00	-0.190906E-02	0.458082E-03
13	0.288877E+00	-0.190906E-02	0.432115E-03
14	0.303987E+00	-0.190906E-02	0.414475E-03
15	0.319097E+00	-0.190906E-02	0.397019E-03
16	0.334208E+00	-0.190906E-02	0.375556E-03
17	0.349318E+00	-0.190906E-02	0.356263E-03
18	0.364428E+00	-0.190906E-02	0.336615E-03
19	0.379538E+00	-0.190906E-02	0.322777E-03
20	0.394648E+00	-0.190906E-02	0.310860E-03
21	0.409758E+00	-0.190906E-02	0.298928E-03
22	0.424869E+00	-0.190906E-02	0.285679E-03
23	0.439979E+00	-0.190906E-02	0.264327E-03
24	0.455089E+00	-0.190906E-02	0.252877E-03
25	0.470199E+00	-0.190906E-02	0.242041E-03
26	0.485309E+00	-0.190906E-02	0.226967E-03
27	0.500419E+00	-0.190906E-02	0.223110E-03
28	0.515530E+00	-0.190906E-02	0.221499E-03
29	0.530640E+00	-0.190906E-02	0.218379E-03
30	0.545750E+00	-0.190906E-02	0.190482E-03
31	0.560860E+00	-0.190906E-02	0.157134E-03
32	0.575970E+00	-0.190906E-02	0.129644E-03
33	0.591081E+00	-0.190906E-02	0.106640E-03
34	0.606191E+00	-0.190906E-02	0.858419E-04
35	0.621301E+00	-0.190906E-02	0.671865E-04
36	0.636411E+00	-0.190906E-02	0.587981E-04
37	0.651521E+00	-0.190906E-02	0.484763E-04
38	0.666631E+00	-0.190906E-02	0.425968E-04
39	0.681742E+00	-0.190906E-02	0.393866E-04
40	0.696852E+00	-0.190906E-02	0.324324E-04
41	0.711962E+00	-0.190906E-02	0.243736E-04
42	0.727072E+00	-0.190906E-02	0.162050E-04
43	0.742182E+00	-0.190906E-02	0.149012E-04
44	0.757292E+00	-0.190906E-02	0.286410E-04
45	0.772403E+00	-0.190906E-02	0.441615E-04
46	0.787513E+00	-0.190906E-02	0.921236E-04
47	0.802623E+00	-0.190906E-02	0.156075E-03
48	0.817733E+00	-0.190906E-02	0.293590E-03
49	0.832843E+00	-0.190906E-02	0.488207E-03
50	0.847953E+00	-0.190906E-02	0.854593E-03
51	0.863064E+00	-0.190906E-02	0.138038E-02
52	0.878174E+00	-0.190906E-02	0.171804E-02
53	0.893284E+00	-0.190906E-02	0.172476E-02
54	0.908394E+00	-0.190906E-02	0.131142E-02
55	0.923504E+00	-0.190906E-02	0.325283E-03
56	0.938614E+00	-0.190906E-02	-0.343490E-03
57	0.953725E+00	-0.190906E-02	-0.911472E-03
58	0.968835E+00	-0.190906E-02	-0.234807E-02
59	0.983945E+00	-0.190906E-02	-0.118641E-01
60	0.995580E+00	0.510344E-02	-0.796055E-02

Table 4.1: Initial near wake values for final iteration

Chapter 5
Conclusions

A free wake analysis using a lifting-line technique is combined with the Kantelis & Widnall's model of a vortex sheet rollup to analyze the flow-field about a hovering rotor. The computational results are compared with experimental data for circulation distribution and wake geometry.

Miller's free wake technique is extended by using a high resolution near wake. A naive application of the vortex sheet convection was shown to lead to large numerical errors due to 1) uneven vortex sheet stretching and 2) inaccurate modelling of the singularity at the tip. The errors due to vortex sheet stretching are reduced by rediscretizing the sheet at every time step to ensure that the vortex elements remain equally spaced. The singularity at the sheet edge is resolved by using a special model at the tip. The model replaces the outermost portion of the sheet with a single vortex of equivalent circulation, impulse and kinetic energy. The model also includes some higher order terms to account for the asymmetry of the spiral.

The intermediate wake is formed from an analysis of the shearing forces on the near wake. The shearing force distribution was calculated to find the division points along the sheet; the sheet was then amalgamated to form the intermediate wake vorticies. The amalgamtion process was equivalent to that of the near wake tip vortex, i.e. conserve the three invariants, circulation, impulse

and kinetic energy.

The validity of the model was tested by comparing it with known experimental values. The circulation distribution showed an overall good agreement with some discrepancies arising near the tip. The errors may be due to the breakdown of the model because of 1) three-dimensionality of the flow, 2) non-linearities near the tip, or 3) viscous effects.

The comparison of direct rollup calculation with those given by Donaldson [5] yielded mixed results. The strength of the tip vortex is somewhat less than that predicted by Donaldson's criterion due to the incorporation of counter-signed vorticity during the rollup process. The analysis of the shearing velocity distribution indicated the same number of amalgamated vorticies as that predicted by Donaldson.

Bibliography

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

[2] 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.

[3] G.K. Batchelor, *An Introduction to Fluid Dynamics*. Cambridge University Press, 1967.

[4] A. Betz. Behaviour of vortex systems. Tech. Memo 713, NACA, June, 1933.

[5] C. duP. Donaldson, R.S. Snedeker, and R.D. sullivan. Calculation of the wakes of three aircraft in holding, takeoff, and landing configurations, and comparison with experimental measurements. AFOSR-TR-73-1594, A.R.A.P. Report No. 190, A.R.A.P. New Jersey, 1973.

[6] P.T. Fink and W.K. Soh. Calculation of vortex sheets in unsteady flow and applications in ship hydrodynamics. *Proc. 10th Symp. Naval Hydrodynamics,* pages 463-488, Cambridge, Mass. 1974.

[7] P.T. Fink and W.K. Soh. A new approach to rollup calculations of vortex sheets. *Proc.R.Soc.London Ser.A,* 362:195-209, 1978.

[8] H.W.M. Hoeijmakers and Vaatstra. A higher-order Panel method applied to vortex sheet rollup. AIAA Paper 82-0096, 1982.

[9] W. Johnson. Comparison of calculated and measured model rotor loading and wake geometry. NASA TM 81189, 1980.

[10] H.Kaden. *Ing. Arch.,* 2:140-168, 1931.

[11] J.P. Kantelis and S.E. Widnall. *Calculations of Axisymmetric Vortex Sheet Rollup Using a Panel and a Filament Model.* Master's thesis, Massachusetts Institute of Technology, 1986.

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

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

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

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

[16] R.H. Miller. private communication.

[17] J. Moran. *An Introduction to Theoretical and Computational Aerodynamics.* John Wiley & Sons, 1984.

[18] D.I. Pullin. On a generalization of Kaden's Problem. *J. Fluid Mech.,* 104:45-53, 1981.

[19] T.W. Roberts. *Euler Equation computations for the Flow Over a Hovering Helicopter Rotor.* PhD thesis, Massachusetts Institute of Technology, 1986.

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

[21] T. Westberg and S.E. Widnall. Helicopter Aerodynamics: Predition Method for Blade Loading and Wake Vortex Locations. FDRL Report 83-2, 1983.

[22] J.E. Yates. Calculation of initial vortex rollup in aircraft wakes. *J. of Aircraft,* Vol.11, No. 7, 1974.

APPENDIX

Program listings

```
      program hover
c
c  A program to determine rotor load distributrion and wake
c  geometry, using an axisymmetric vortex sheet and the free
c  wake method.
c
c  Written by Kevin Huh (1987) and John Kantelis (1986)
c
      include '/user/huh/rollup/common.f'
       common /th/ theta(0:900),thmax
      COMMON /E1/ TAUO,BDUM,CDUM,DDUM,TAU2LOOP
      COMMON /E2/ DUM1,DUM2,DUM3,DUM4,DUM5,AE2
      DIMENSION SGAM(0:900),DDGM(0:900)
      REAL*4 LAMBDA,K
      CHARACTER PLTITL*80,TITLE*80
       character*60 gtitle,ltitle
       integer indexvort(10)
       logical lgo,lrite,lgo2,lrite2
       real tmp(2)
       data lgo,lrite,lgo2,lrite2/.true.,.false.,.true.,.false./
       data iaxi,ipanel,iredisc,itrs,intkod,iload/1,0,1,0,1,0/
C
C  NRING   = number of rings in the near wake (counting starts at 0)
C  NSMALL  = number of vortex rings which form the sheet, NSMALL=NRING-1
C
C  Ring Indicies ------------------------------------------------------
C
C    Index     Meaning
C
C      0       Degenerate ring on the central axis of the ring system.
C              This ring has zero circulation and is used only to
C              define the inner boundary of the sheet.
C
C  1 - NSMALL  These are the "real" vortex rings, i.e., they have non-
```

```
C  (inclusive) zero circulation. These all lie on the sheet.
C
C     NSMALL   This ring forms the outer boundary of the sheet.
C
C     NRING    This is the index of the very last ring. It is the
C              concentrated core tip ring. It does not lie on the sheet.
C              Initially, this ring has circulation determined by FRAC.
C              As the sheet rolls up, the circulation from the outer
C              portions, which is chopped off, gets put into the core.
C
      PI=ACOS(-1.0)
      TIME(1)=0.0
C
      READ(5,*)IWR
      read(5,*) inew
      READ(5,*)DXMAX,nitrs,nload
      READ(5,*)THMAX
      READ(5,*)FRAC
      READ(5,*)NU2,nu22
      if(inew .eq. 0)then
         READ(5,*)NRING
         READ(5,*)LINEUP
         READ(5,*)MINSECT
         READ(5,*)MAXSEGS
         NSMALL=NRING-1
      endif
      TIMEAB=9.0E20
      DISTIN=9.0E5
      DISTMX=DISTIN**2
c
c   open files for output
C
        open(unit=15,file='loading.dat',form='formatted',status='new')
        open(unit=9,file='time.dat',status='new',form='formatted')
        open(unit=12,file='geomt.dat',form='formatted',status='new')
```

```
      open(unit=26,file='velvec.dat',form='formatted',status='new')
      open(unit=19,file='newgeomt.dat',form='formatted',status='new')
      open(unit=7,form='unformatted',status='new',file='sheettemp.dat')
      call seta
      TIME(1)=((2.0*PI**2)/(9.0*AE2))*frac**(1.5)
      call experiment
C
c if inew = 0 then start new else restart
C
      if(inew .eq. 0)then
c
c   set circulation from Miller's method
c
         call initwake
         call initcirc(frac)
         call ritecirc
          tint = pi
         istep = 1
          TIMENOW=TIME(1)
      else
         open(unit=16,file='restart.dat',form='unformatted',status='old')
         call readinput(tint,itpre)
         istep = 2
         timenow = time(2)
         close(16)
      endif
       CALL movem(istep)
      call riteinput
C
C--Write out the input data
C
C
C-----------------------------------------------------------------
C-------- MAIN TIME LOOP -----------------------------------------
C-----------------------------------------------------------------
```

```
C
 234       continue
       if(.not. lgo)goto 99
       if ((timenow-tint) .ge. 0.)goto99
       istep = istep + 1
C
C--Compute velocities at step ISTEP which
C  corresponds to time (ISTEP-1)*"DT" and also update positions
C
        CALL MOVEM(ISTEP)
        WVTIP=WVEL(NRING)
        UVTIP=UVEL(NRING)
        CTIP=CORE(NRING)
        NIN=NSMALL
        NOUT=NSMALL
        GMM=GAMMA(NRING)
        CALL REDISC(RADIUS,HEIGHT,GAMMA,NIN,NOUT,GMM,SUML,DELTAL)
        WVEL(NRING)=WVTIP
        UVEL(NRING)=UVTIP
        CORE(NRING)=CTIP
        WVTIP=WVEL(NRING)
        UVTIP=UVEL(NRING)
        CTIP=CORE(NRING)
        CALL LUMPS
        WVEL(NRING)=WVTIP
        UVEL(NRING)=UVTIP
        CORE(NRING)=CTIP
C
C--Now update the core size, since the sheet has stretched
C
        DO 691 II=1,NSMALL
  691   CORE(II)=WPAN(II)/2.0
C
       if(istep .ge. 50000)then
          type*,'istep .gt. 50000',istep,timenow
```

```
         goto 57
       endif
        goto 234
   99 CONTINUE
C
C-----------------------------------------------------------------
C---------- END OF MAIN TIME LOOP -------------------------------
C-----------------------------------------------------------------
C
       itrs = itrs + 1
       write(9,*) ' '
       write(9,17)'pi itr# = ',itrs,'   time = ',timenow
       write(9,*) ' '
 17       format(a,i6,a,f10.5)
       tint = tint + pi
c
c   write out results
c
       if(mod((itrs),nu2) .eq. 0 .or. itrs .eq. nitrs)then
          lrite = .true.
        endif
c
c   redivide the sheet, calculate new loading and wake geometry
c
       call seta
       call divide(indexvort,nvort)
       call newwake(indexvort,nvort,lrite,lgo)
c
c  check if wake converged
c
       if(lgo .and. itrs .lt. nitrs)then
          call oldsheet(istep)
           call riterst(tint,itrs+itpre)
          goto 234
       endif
```

```
      if(itrs .ge. nitrs)goto57
      iload = iload + 1
      write(9,*) ' '
      write(9,18)'load itr # = ',iload
      write(9,*) ' '
 18      format(a,i6)
      if(mod(iload,nu22) .eq. 0)lrite2=.true.
c
c   if wake converged, recalculate new loading on blade
c
      call circulation(lgo2,lrite2)
      call makesheet(frac,istep)
      call riteinput
c
c   check if loading has converged
c
      if(lgo2 .and. iload .lt. nload)then
         lgo = .true.
          call riterst(tint,itrs+itpre)
         goto 234
      endif
 57      continue
      type*,'Total # of pi iterations = ',itrs+itpre
      type*,'Total # of load iterations = ',iload
       call riterst(tint,itrs+itpre)
            END



c
c   this is the include file of common blocks
c
      COMMON /B1/ RADIUS(0:900),HEIGHT(0:900),WVEL(0:900),UVEL(0:900),
     $            GAMMA(0:900),CORE(0:900),TIME(0:50000),
     $            TENERG(0:50000),TIMPUL(0:50000)
```

```
      COMMON /B2/ NRING,PI,DXMAX,INTKOD,NSMALL,VOLTIP,IAXI
      COMMON /B9/ IWR
      COMMON /PANEL/ IPANEL,DRPAN(0:900),DZPAN(0:900),WPAN(0:900)
      COMMON /C1/ LINEUP,MAXSEGS,MINSECT
      COMMON /D1/ DISTMX
      COMMON /SEGS/ AXLSEG(0:900)
      COMMON /F1/ ALPLAST,GMTO
      COMMON /ZZ/ ZXO,ZYO
      COMMON /G1/ TIPVAX
      COMMON /NOW/ TIMENOW,BIGRREF
       common /wake/zinter(10,5),rinter(10,5),ginter(10,5),cinter(10,5)
     &                   ,zfar(10),rfar(10),gfar(10),uveli(10,5),wveli(10,5)
     &                   ,ninter(5),nfar,level,volinter(10,5)
       common /blade/pitch(900),chord(900),span(900),aslope,sigma,nspan
      COMMON /RK/ RADO(0:900),HEIO(0:900),RK(0:900,4),HK(0:900,4)
       common /rkwake/ zintO(10,5),rintO(10,5),rki(10,5,4),hki(10,5,4)
       common /loading/circold(900),circ(900)
       common /oldwake/roldi(10,5),zoldi(10,5),goldi(10,5)
       common /exper/expx(28),expy(28)




      SUBROUTINE ANGLES(RS,ZS,NTOTSUB,THSUB)
C
C--Get angles relative to tip for each subinterval
C
       include '/user/huh/rollup/common.f'
       real r(0:900),z(0:900)
       equivalence (r,radius)
       equivalence (z,height)
      DIMENSION RS(0:8100),ZS(0:8100),THSUB(0:8100)
      REAL*4 MAG
C
      RT=R(NRING)     !tip values of radius
```

```
      ZT=Z(NRING)     !          and height
C
      A=-1.0
      B=0.0
      C=RS(0)-RT
      D=ZS(0)-ZT
      DOT=A*C+B*D
      CROSS=A*D-B*C
      MAG=SQRT((A*A+B*B)*(C*C+D*D))
      ARG=DOT/MAG
      IF (ABS(ARG) .GT. 1.0) ARG=SIGN(1.0,ARG)
      THSUB(0)=SIGN(1.0,CROSS)*ACOS(ARG)
C
      DO 100 I=1,NTOTSUB
      A=RS(I-1)-RT
      B=ZS(I-1)-ZT
      C=RS(I)-RT
      D=ZS(I)-ZT
      DOT=A*C+B*D
      CROSS=A*D-B*C
      MAG=SQRT((A*A+B*B)*(C*C+D*D))
      ARG=DOT/MAG
      IF (ABS(ARG) .GT. 1.0) ARG=SIGN(1.0,ARG)
  100 THSUB(I)=THSUB(I-1)+SIGN(1.0,CROSS)*ACOS(ARG)
C
      RETURN
      END



      SUBROUTINE CENSPIR(VXP,VYP)
C--Calculates the velocity induced at the spiral center due to the
C  inner spiral region
       include '/user/huh/rollup/common.f'
```

```
      COMMON /E1/ TAUO,B,C,D,TAU2LOOP
      COMMON /E2/ AO,A1,A2,A3,RHOSTAR,A
C
      TAUOLOOP=0.24016
      BGROLOOP=0.32094
      BIGRPRM=BGROLOOP*(A*TIME(1))**(2.0/3.0)
      CC=A**(4.0/3.0)*TIMENOW**(1.0/3.0)/BIGRPRM
      ALFR=alplast-pi/2.
      TAUA=TAUOLOOP+ALFR*(0.60702015)/(2.0*PI)
C
      ALF=alplast-pi/2.
      IF (ALF .LT. 4.0*PI) THEN
         ZX=ZXO
         ZY=ZYO
      ELSE IF (ALF .GT. 6.0*PI) THEN
         ZX=0.0
         ZY=0.0
      ELSE
         ZX=ZXO-ZXO*(ALF-4.0*PI)/(2.0*PI)
         ZY=ZYO-ZYO*(ALF-4.0*PI)/(2.0*PI)
      ALF=alplast-pi/2
      ENDIF
C
      VXTILD=-0.00029091*ZX*CC*TAUA**(-2.30525)
      VYTILD=-0.00424515*ZY*CC*TAUA**(-1.45439)
      VXP=VXTILD*COS(ALFR)-VYTILD*SIN(ALFR)
      VYP=VXTILD*SIN(ALFR)+VYTILD*COS(ALFR)
C     WRITE(6,*)'NRING,UVEL(NRING),WVEL(NRING)'
C     WRITE(6,*) NRING,UVEL(NRING),WVEL(NRING)
C     WRITE(6,*)'TAUOLOOP,BGROLOOP,A,TIME(1),TIMENOW,BIGRPRM'
C     WRITE(6,*) TAUOLOOP,BGROLOOP,A,TIME(1),TIMENOW,BIGRPRM
C     WRITE(6,*)'CC,ALFR,TAUA,ZX,ZY,VXTILD,VYTILD,VXP,VYP'
C     WRITE(6,*) CC,ALFR,TAUA,ZX,ZY,VXTILD,VYTILD,VXP,VYP
C
      RETURN
```

```
      END



      SUBROUTINE circulation(lgo,lrite)
C
C--This routine calculates new loading on the blade based on the
C  lifting-line theory.
C
       include '/user/huh/rollup/common.f'
      REAL*4 LAMBDA,K,k1
       logical lgo,lrite
       real wvnear(900),wvint(10,5),wvfar(10),wvelt(103),circt(103)
       character*60 gtitle,title
       common /temp1/a0,a1,a2,a3,a4,b0,b1,b2,b3,b4
       common /temp2/c0,c1,c2,c3,c4,d0,d1,d2,d3,d4
       data omega,epsilon/0.1,0.001/
CVD$R ASSOC
C
C--Loop over all rings, except center ring, get induced velocity at each
C
      DO 10 I=2,nspan
        W=span(i)
        Z=0.
        DO 20 J=1,nring
          WP=RADIUS(J)
          ZP=HEIGHT(J)
         if(wp .le. 0.18 .or. zp .gt. 0.)goto20
          R1=SQRT((Z-ZP)**2+(W-WP)**2)
          R2=SQRT((Z-ZP)**2+(W+WP)**2)
          LAMBDA=(R2-R1)/(R2+R1)
          FK=1.0-LAMBDA**2
          K = A0+FK*(A1+FK*(A2+FK*(A3+FK*A4)))
     $         +(B0+FK*(B1+FK*(B2+FK*(B3+FK*B4))))*ALOG(1.0/FK)
```

```
          E = C0+FK*(C1+FK*(C2+FK*(C3+FK*C4)))
     $         +(D0+FK*(D1+FK*(D2+FK*(D3+FK*D4))))*ALOG(1.0/FK)
          F1=((W-WP)/R1+(W+WP)/R2)*(K-E)
          F2=2.0*E*LAMBDA/(1.0-LAMBDA**2)
          F3=(W*(R1**2-R2**2)+WP*(R1**2+R2**2))/(R1*R2*(R1+R2))
          WV=GAMMA(J)*(F1+F2*F3)/(2.0*PI*W)
          WVnear(j)=WV
   20    CONTINUE
c
c    calculate influence of intermediate wake
c
        do 50 j1 = 2,level
           do 50 i1 = 1,ninter(j1)
          wp=rinter(i1,j1)
          zp=zinter(i1,j1)
          r1=sqrt((z-zp)**2+(w-wp)**2)
          r2=sqrt((z-zp)**2+(w+wp)**2)
           lambda=(r2-r1)/(r2+r1)
           fk=1.0-lambda**2
           k = a0+fk*(a1+fk*(a2+fk*(a3+fk*a4)))
     $         +(b0+fk*(b1+fk*(b2+fk*(b3+fk*b4))))*alog(1.0/fk)
           e = c0+fk*(c1+fk*(c2+fk*(c3+fk*c4)))
     $         +(d0+fk*(d1+fk*(d2+fk*(d3+fk*d4))))*alog(1.0/fk)
           f1=((w-wp)/r1+(w+wp)/r2)*(k-e)
           f2=2.0*e*lambda/(1.0-lambda**2)
           f3=(w*(r1**2-r2**2)+wp*(r1**2+r2**2))/(r1*r2*(r1+r2))
           wv=ginter(i1,j1)*(f1+f2*f3)/(2.0*pi*w)
           wvint(i1,j1)=wv
 50        continue
c
c    calculate influence of far wake cylinder
c
        dphi = pi/20.       !40 integration steps
        nphi = nint(2.*pi/dphi) - 1
        do 671 ifar = 1,nfar
```

```
      zp = abs(z - zfar(ifar))
      wsum = 0.
      do 601 iphi = 1,nphi
         phi = float(iphi)*dphi
         x3 = rfar(ifar)*rfar(ifar)+w*w+zp*zp-
     &              2.*rfar(ifar)*w*cos(phi)
         dwfar = dphi*rfar(ifar)*(rfar(ifar)-w*cos(phi))/
     &                (x3-zp*zp)*(1.-zp/sqrt(x3))
         wsum = wsum + dwfar
 601      continue
      x3 = rfar(ifar)*rfar(ifar)+w*w+zp*zp-2.*rfar(ifar)*w
      if((x3-zp*zp) .ne. 0.)then
         dwfar = dphi*rfar(ifar)*(rfar(ifar)-w)/(x3-zp*zp)*(1.-zp/sqrt(x3))
         wsum = wsum + dwfar
      endif
      wfar = wsum/4./pi*gfar(ifar)
      wvfar(ifar)=  wfar
 671      continue
      wvelt(i) = 0.
      do i1 = 1,nring
         wvelt(i) = wvelt(i) + wvnear(i1)
      enddo
      do j1 = 2,level
         wvelt(i) = wvelt(i) + wvint(1,j1)+wvint(2,j1)+wvint(3,j1)
      enddo
      wvelt(i) = wvelt(i) + wvfar(1) + wvfar(2)
C
C--get velocity of higher order terms in tip region
C
         CALL DOUBLET(W,Z,VXPRM,VYPRM,VRPRM,VTHPRM)
         WVELt(i)=WVELt(i)+VYPRM
   10 CONTINUE
      call itercirc(wvelt,circt)
      do i = 1,nspan
        if(span(i) .le. 0.11 .or. span(i) .ge. 0.985)then
```

```
            circ(i) = circold(i)
          else
            circ(i) = circold(i) + omega*(circt(i)-circold(i))
          endif
        enddo
        lgo = .false.
        do 200 i = 50,nspan
           if(circ(i) .eq. 0.)goto200
           dchange =  abs((circ(i)-circold(i))/circ(i))
           if(dchange .gt. epsilon)then
               lgo = .true.
               goto 201
           endif
 200        continue
        lrite = .true.
        type*,'Load convergence at time = ',timenow
c
 201        continue
c
c    reset old values
c
        do i = 1,nspan
           circold(i) = circ(i)
        enddo
        if(lrite)then
c
c    write out loading values
c
        gtitle='LOADING'
        title='RADIUS~GAMMA~TIME = '
        write(15,*)gtitle
        write(15,*)2
        write(15,*)52
        write(15,2)title,timenow
 2        format(a20,f9.5)
```

```
       write(15,*)1
       write(15,*)0.,1.,0.,3.e-02
       write(15,*)2
       write(15,*)nspan
CVD$ NOCONCUR
CVD$ NOVECTOR
       do 90 i = 1,nspan
            write(15,*) span(i),circ(i)
 90       continue
       write(15,*)4
       write(15,*)28
CVD$ NOCONCUR
CVD$ NOVECTOR
       do i = 1,28
            write(15,*) expx(i),expy(i)
       enddo
       lrite = .false.
       endif
C
C--done
C
      RETURN
      END




      SUBROUTINE divide(indexvort,nvort)
C
C--This routine calculates the division positions of the near wake
C
       include '/user/huh/rollup/common.f'
       common /oldgeom/radiusold(0:900),heightold(0:900)
         COMMON /TH/ THETA(0:900),THMAX
       real tanr(900),tanz(900)
```

```
      character*60 gtitle,title
      integer indexvort(10)
CVD$R ASSOC
c
      do i = 1,nring
         radiusold(i) = radius(i)
         heightold(i) = height(i)
      enddo
c
      do 5 i = 1,10
         indexvort(i) = 0
 5      continue
c
c   calculate tangent vector to the sheet, central differencing
c
      do 2 i = 1,nsmall
         tanr(i) = radius(i+1) - radius(i-1)
         tanz(i) = height(i+1) - height(i-1)
         str = sqrt(tanr(i)**2 + tanz(i)**2)
         tanr(i) = tanr(i)/str
         tanz(i) = tanz(i)/str
 2      continue
c
c   calculate the inner product of tangent vector and the velocity vector
c   where they change the sign is the dividing point
c
      nvort  = 1
c      gtitle = 'SHEAR STRENGTH VS. ARC LENGTH'
c      write(2,*)gtitle
c      write(2,*) 1
c      write(2,*) 60
c      write(2,*) 'x~UNIT TANGENT*VELOCITY~'
c      WRITE(2,*) 1
c      WRITE(2,*) 0,1.5,-.3,.3
c      WRITE(2,*) 2
```

```
c        WRITE(2,*) NSMALL
        predot = tanr(1)*uvel(1) + tanz(1)*wvel(1)
c        write(2,*)axlseg(1),predot
        do 90 i = 2,nsmall
           curdot = tanr(i)*uvel(i) + tanz(i)*wvel(i)
c           write(2,*)axlseg(i),curdot
           if((curdot * predot) .lt. 0. .and. curdot .lt. 0.)then
               indexvort(nvort) = i
               nvort = nvort + 1
           endif
           predot = curdot
 90        continue
        indexvort(nvort) = nring + 1
c        dr = radius(nring)-radius(nsmall)
c           dz = height(nring)-height(nsmall)
c        axlseg(nring) = sqrt(dr*dr+dz*dz)+axlseg(nsmall)
c        gtitle = 'VORTICITY VS. ARC LENGTH'
c        write(2,*)gtitle
cj        write(2,*) 1
c        write(2,*) 60
c        write(2,*) 'x~VORTICITY~'
c        WRITE(2,*) 1
c        WRITE(2,*) 0,1.5,-.03,.03
c        WRITE(2,*) 2
c        WRITE(2,*) Nring
ch        do i = 2,nring
c           write(3,*)axlseg(i),gamma(i)
c        enddo
      RETURN
      eND



      SUBROUTINE DOUBLET(W,Z,VXPRM,VYPRM,VRPRM,VTHPRM)
```

```
C
C--Calculates the higher order velocity components in the spiral region
C
C  W       radius of control point
C  Z       height of control point
C  VXPRM   V sub x prime in the notes, radial velocity component
C  VYPRM   V sub y prime in the notes, axial velocity component
C  GMTO    strength of tip at start of run
C
       include '/user/huh/rollup/common.f'
      COMMON /E1/ TAUO,BDUM,CDUM,DDUM,TAU2LOOP
      COMMON /E2/ AO,A1,A2,A3,RHOSTAR,A
CVD$R ASSOC
      DIMENSION B(7,2),D(7,2)     !B(coef., exponent)
      DATA B /.0053006800, .0013100600, .000574976 , .000319442 ,
     $        .0002020870, .0001387070, .000100337 ,
     $        -1.38557    , -1.36294    , -1.34587    , -1.33015    ,
     $        -1.31621    , -1.30109    , -1.28252                  /
      DATA D /.0010074100, .0001546680, .0000656961, .0000312179,
     $        .0000157322, .0000101938, .0000115224,
     $        -2.37327    , -2.28212    , -2.40883    , -2.36051    ,
     $        -2.17250    , -2.22138    , -2.80713                  /
      BIGRREF=0.0966063
      NCOEF=7
C
      BIGRPRM=BIGRREF*(A*TIME(1))**(2.0/3.0)  !constant in time
      RPRIME=SQRT((W-RADIUS(NRING))**2+(Z-HEIGHT(NRING))**2)
      R=RPRIME/BIGRPRM
      SINTST=(Z-HEIGHT(NRING))/RPRIME
      COSTST=(W-RADIUS(NRING))/RPRIME
      THETAST=ATAN2(SINTST,COSTST)
      VRPRM=0.0
      VTHPRM=0.0
C     WRITE(6,*)' ALPLAST,ALPROT,BIGRREF,BIGRPRM,A,TIME(1)'
C     WRITE(6,*)  ALPLAST,ALPROT,BIGRREF,BIGRPRM,A,TIME(1)
```

```
C     WRITE(6,*)' CC,W,RADIUS(NRING),Z,HEIGHT(NRING),NRING'
C     WRITE(6,*)  CC,W,RADIUS(NRING),Z,HEIGHT(NRING),NRING
C     WRITE(6,*)' RPRIME,R,SINTST,COSTST,THETAST,GAMMA(NRING),GMTO'
C     WRITE(6,*)  RPRIME,R,SINTST,COSTST,THETAST,GAMMA(NRING),GMTO
c***
       if(gmt0 .eq. 0.)gmt0=gamma(nring)
      DO 10 N=1,NCOEF
       if(gmt0 .eq. 0.)then
          rrr = 1.
       else
           RRR=GAMMA(NRING)/GMTO            !adjust for varying tip strength
       endif
      TAUA=TAU2LOOP+(alplast-pi/2.)*(0.60702015)/(2.0*PI)
      BN=B(N,1)*TAUA**B(N,2)
      DN=D(N,1)*TAUA**D(N,2)
      DDVR= (BN*COS(N*(THETAST+PI/2.0-(alplast-pi/2.)))
     $      -DN*SIN(N*(THETAST+PI/2.0-(alplast-pi/2.))))
     $      *(N/R**(N+1))*RRR*(A**(4.0/3.0)*TIMENOW**(1.0/3.0)/BIGRPRM)
      DDVTH=(BN*SIN(N*(THETAST+PI/2.0-(alplast-pi/2.)))
     $      +DN*COS(N*(THETAST+PI/2.0-(alplast-pi/2.))))
     $      *(N/R**(N+1))*RRR*(A**(4.0/3.0)*TIMENOW**(1.0/3.0)/BIGRPRM)
C     WRITE(6,*)' I,BETA(I),RRR,RR,DDVR,DDVTH'
C     WRITE(6,*)  I,BETA(I),RRR,RR,DDVR,DDVTH
      VRPRM =VRPRM+DDVR
   10 VTHPRM=VTHPRM+DDVTH
      VXPRM=VRPRM*COSTST-VTHPRM*SINTST
      VYPRM=VRPRM*SINTST+VTHPRM*COSTST
C
      ALF=alplast-pi/2.
      IF (ALF .LT. 4.0*PI) THEN
         ZX=ZX0
         ZY=ZY0
      ELSE IF (ALF .GT. 6.0*PI) THEN
         ZX=0.0
         ZY=0.0
```

```
      ELSE
         ZX=ZXO-ZXO*(ALF-4.O*PI)/(2.O*PI)
         ZY=ZYO-ZYO*(ALF-4.O*PI)/(2.O*PI)
      ALF=alplast-pi/2.
      ENDIF
      VXPRM=VXPRM*ZX
      VYPRM=VYPRM*ZY
C     IF (IWR .LE. 9) THEN
C       WRITE(6,*)' VXPRM,VYPRM',VXPRM,VYPRM
C       WRITE(IWR,*)' TIMENOW,VXPRM,VYPRM',TIMENOW,VXPRM,VYPRM
C     ENDIF
C
      RETURN
      END




      SUBROUTINE ELLIP(RK,SUMK,SUME)
       common /temp1/a0,a1,a2,a3,a4,b0,b1,b2,b3,b4
       Common /temp2/c0,c1,c2,c3,c4,d0,d1,d2,d3,d4
C
C--Computes complete elliptic integrals, function of RK.
C  Uses a four term series approximation.
C    SUMK   = K(RK) - FIRST KIND
C    SUME   = E(RK) - SECOND KIND
C
C
      FK=1.O-RK**2
      SUMK = AO+FK*(A1+FK*(A2+FK*(A3+FK*A4)))
     $     +(BO+FK*(B1+FK*(B2+FK*(B3+FK*B4))))*ALOG(1.O/FK)
      SUME = CO+FK*(C1+FK*(C2+FK*(C3+FK*C4)))
     $     +(DO+FK*(D1+FK*(D2+FK*(D3+FK*D4))))*ALOG(1.O/FK)
C
      RETURN
```

```
      END



      subroutine experiment
c
c   a subroutine to set the experimental loading positions
C
      include '/user/huh/rollup/common.f'
      data expx/.99,.9733334,.9666666,.96,.95,.9533334,.926666,.91,.9,
     &                .8993333,.8813334,.8733333,.8766666,.85,.8499667,.8266,
     &               .8,.8003334,.78,.773334,.7731334,.75,.7,.65,.623333,.6,
     &           .5733333,.55/
      data expy/1.093333,1.36,1.6,1.653333,1.84,1.893333,1.986666,2.16,
     &               2.1573335,2.213333,2.066666,2.0933334,2.0266669,1.8933,
     &               1.8533333,1.76,1.72,1.762669,1.7186668,1.68,1.733333,
     &               1.6266668,1.5733335,1.506668,1.493335,1.32,1.36,1.2933/
      do i = 1,28
         expy(i) = expy(i) * .01
      enddo
      return
      end



      SUBROUTINE GETANG
C
C--This routine computes the angles of the segments relative to
C  the horizontal axis
C
C
      include '/user/huh/rollup/common.f'
      COMMON /TH/ THETA(0:900),THMAX
```

```
      DIMENSION DT(0:900)
C
C--Get increment for first angle
C
      DT(0)=ATAN((HEIGHT(1)-HEIGHT(0))/(RADIUS(1)-RADIUS(0)))
      THETA(0)=DT(0)
C
C--Get THETA for the rest
C
      DO 10 I=1,NSMALL-1
      XM=RADIUS(I-1)
      XI=RADIUS(I)
      XP=RADIUS(I+1)
      YM=HEIGHT(I-1)
      YI=HEIGHT(I)
      YP=HEIGHT(I+1)
C
C--Get sign of cross product of Vi and Vi+1
C
      SCROSS=SIGN(1.0,(XI-XM)*(YP-YI)-(XP-XI)*(YI-YM))
      VMAGS=SQRT(((XI-XM)**2+(YI-YM)**2)*((XP-XI)**2+(YP-YI)**2))
      DOT=(XI-XM)*(XP-XI)+(YI-YM)*(YP-YI)
C
C--Watch out for numerical inaccuracy in argument of arccos
C
      DT(I)=0.0
C     WRITE(6,*)' I, DOT/VMAGS:',I,DOT/VMAGS
      IF (ABS(DOT/VMAGS) .LE. 1.0) DT(I)=SCROSS*ACOS(DOT/VMAGS)
      THETA(I)=THETA(I-1)+DT(I)
   10 CONTINUE
C     DO 22 I=0,NSMALL-1
C  22 WRITE(6,*)' I,DT(I),THETA(I):',I,DT(I),THETA(I)
C
      RETURN
      END
```

```
      SUBROUTINE GETDT(DT)
       include '/user/huh/rollup/common.f'
C
C--Find max velocity and DT, DT=(ABS(DXMAX) if DXMAX .LT. 0.0)
C
      IF (DXMAX .LT. 0.0) THEN
        DT=ABS(DXMAX)
      ELSE
        VMAX=UVEL(0)**2+WVEL(0)**2
        DO 401 I=1,NRING
          VRING=UVEL(I)**2+(WVEL(I))**2
          IF (VRING .GT. VMAX) VMAX=VRING
 401    CONTINUE
        VMAX=SQRT(VMAX)
        DT=DXMAX/VMAX
      ENDIF
      RETURN
      END




      SUBROUTINE GETK(N)
       include '/user/huh/rollup/common.f'
C
C--Computes the 4 K parameters used in the Runge-Kutta routine.
C  The K's are just the velocity components.
C
      DO 10 I=0,NRING
        RK(I,N)=UVEL(I)
        HK(I,N)=WVEL(I)
```

```
   10 CONTINUE
       do 20 j = 1,level
          do 20 i =1,ninter(j)
              rki(i,j,n) = uveli(i,j)
              hki(i,j,n) = wveli(i,j)
 20       continue
C
      RETURN
      END
      FUNCTION GETRAD(ILOC,R1,R2)
C
C--This function determines the location of the vortex ring within
C  a segment, if ILOC=1 then the ring is centered, if ILOC=2 then the
C  ring is positioned such that the total impulse is maintained
C
      IF (ILOC .EQ. 1) THEN
        GETRAD=(R1+R2)/2.0
      ELSE
        F1=((SQRT(1.0-R1**2))**3-(SQRT(1.0-R2**2))**3)*8.0/3.0
        F2=4.0*R1**2*SQRT(1.0-R1**2)
        F3=4.0*R2**2*SQRT(1.0-R2**2)
        F4=4.0*(SQRT(1.0-R1**2)-SQRT(1.0-R2**2))
        GETRAD=SQRT((F1+F2-F3)/F4)
      ENDIF
C
      RETURN
      END




      SUBROUTINE IMPULS(TIMP)
C
C--Computes total impulse of the vortex filament system for
C  either axi or 2-D.  (Note: only computing the "x" or "radial"
```

```
C  component of the 2-D impulse.
C
       include '/user/huh/rollup/common.f'
CVD$R ASSOC
C
        TIMP=0.0
        DO 10 I=1,NRING
   10   TIMP=TIMP+PI*GAMMA(I)*RADIUS(I)**2
C
      RETURN
      END




      SUBROUTINE INDVEL
C
C  AXISYMMETRIC FILAMENTS  (RINGS)
C--This routine calculates the induced velocity at each ring position
C  due to all the other rings and due to itself.
c--For all wake vorticies. (near,intermediate and far)
C
       include '/user/huh/rollup/common.f'
      REAL*4 LAMBDA,K,k1
       real uvnear(900),wvnear(900),uvint(10,5),wvint(10,5)
       real uvfar(10),wvfar(10)
       common /temp1/a0,a1,a2,a3,a4,b0,b1,b2,b3,b4
       common /temp2/c0,c1,c2,c3,c4,d0,d1,d2,d3,d4
      DATA A0,A1,A2,A3,A4,B0,B1,B2,B3,B4/
     $  1.38629436112, .09666344259, .03590092383,
     $    .03742563713, .01451196212, .5          ,
     $    .12498593597, .06880248576, .03328355346,
     $    .00441787012/
      DATA C0,C1,C2,C3,C4,D0,D1,D2,D3,D4/
     $  1.0          , .44325141463, .06260601220,
```

```
     $    .04757383546, .01736506451, .0          ,
     $    .24998368310, .09200180037, .04069697526,
     $    .00526449639/
CVD$R ASSOC
C
C--Loop over all rings, except center ring, get induced velocity at each
C
      DO 10 I=1,NRING
        W=RADIUS(I)
        Z=HEIGHT(I)
       do i1 = 1,nring
          uvnear(i1) = 0.
          wvnear(i1) = 0.
       enddo
C
C--Loop over all rings again (except control ring) and determine the
C  contribution of each to the induced velocity at ring I.
C
        DO 20 J=1,i-1
          WP=RADIUS(J)
          ZP=HEIGHT(J)
          R1=SQRT((Z-ZP)**2+(W-WP)**2)
          R2=SQRT((Z-ZP)**2+(W+WP)**2)
          LAMBDA=(R2-R1)/(R2+R1)
          FK=1.0-LAMBDA**2
          K = AO+FK*(A1+FK*(A2+FK*(A3+FK*A4)))
     $          +(BO+FK*(B1+FK*(B2+FK*(B3+FK*B4))))*ALOG(1.0/FK)
          E = CO+FK*(C1+FK*(C2+FK*(C3+FK*C4)))
     $          +(DO+FK*(D1+FK*(D2+FK*(D3+FK*D4))))*ALOG(1.0/FK)
          F1=((W-WP)/R1+(W+WP)/R2)*(K-E)
          F2=2.0*E*LAMBDA/(1.0-LAMBDA**2)
          F3=(W*(R1**2-R2**2)+WP*(R1**2+R2**2))/(R1*R2*(R1+R2))
          WV=GAMMA(J)*(F1+F2*F3)/(2.0*PI*W)
          WVnear(j)=WV
          F4=(Z-ZP)*(R1+R2)/(R1*R2)
```

```
        F5=K-E*(1.0+LAMBDA**2)/(1.0-LAMBDA**2)
        UV=GAMMA(J)*F4*F5/(2.0*PI*W)
        UVnear(j)=-UV
   20   CONTINUE
        DO 40 J=i+1,nring
        WP=RADIUS(J)
        ZP=HEIGHT(J)
        R1=SQRT((Z-ZP)**2+(W-WP)**2)
        R2=SQRT((Z-ZP)**2+(W+WP)**2)
        LAMBDA=(R2-R1)/(R2+R1)
        FK=1.0-LAMBDA**2
        K = AO+FK*(A1+FK*(A2+FK*(A3+FK*A4)))
     $          +(BO+FK*(B1+FK*(B2+FK*(B3+FK*B4))))*ALOG(1.0/FK)
        E = CO+FK*(C1+FK*(C2+FK*(C3+FK*C4)))
     $          +(DO+FK*(D1+FK*(D2+FK*(D3+FK*D4))))*ALOG(1.0/FK)
        F1=((W-WP)/R1+(W+WP)/R2)*(K-E)
        F2=2.0*E*LAMBDA/(1.0-LAMBDA**2)
        F3=(W*(R1**2-R2**2)+WP*(R1**2+R2**2))/(R1*R2*(R1+R2))
        WV=GAMMA(J)*(F1+F2*F3)/(2.0*PI*W)
        WVnear(j)=WV
        F4=(Z-ZP)*(R1+R2)/(R1*R2)
        F5=K-E*(1.0+LAMBDA**2)/(1.0-LAMBDA**2)
        UV=GAMMA(J)*F4*F5/(2.0*PI*W)
        UVnear(j)=-UV
   40   CONTINUE
c
c   calculate influence of intermediate wake
c
       do 50 j1 = 1,level
          do 50 i1 = 1,ninter(j1)
          wp=rinter(i1,j1)
          zp=zinter(i1,j1)
          r1=sqrt((z-zp)**2+(w-wp)**2)
          r2=sqrt((z-zp)**2+(w+wp)**2)
          lambda=(r2-r1)/(r2+r1)
```

```
          fk=1.0-lambda**2
          k = a0+fk*(a1+fk*(a2+fk*(a3+fk*a4)))
     $         +(b0+fk*(b1+fk*(b2+fk*(b3+fk*b4))))*alog(1.0/fk)
          e = c0+fk*(c1+fk*(c2+fk*(c3+fk*c4)))
     $         +(d0+fk*(d1+fk*(d2+fk*(d3+fk*d4))))*alog(1.0/fk)
          f1=((w-wp)/r1+(w+wp)/r2)*(k-e)
          f2=2.0*e*lambda/(1.0-lambda**2)
          f3=(w*(r1**2-r2**2)+wp*(r1**2+r2**2))/(r1*r2*(r1+r2))
          wv=ginter(i1,j1)*(f1+f2*f3)/(2.0*pi*w)
          wvint(i1,j1)=wv
          f4=(z-zp)*(r1+r2)/(r1*r2)
          f5=k-e*(1.0+lambda**2)/(1.0-lambda**2)
          uv=ginter(i1,j1)*f4*f5/(2.0*pi*w)
          uvint(i1,j1)=-uv
 50       continue
c
c   calculate influence of far wake cylinder
c
       dphi = pi/20.       !40 integration steps
       nphi = nint(2.*pi/dphi) - 1
       do 67 ifar = 1,nfar
       zp = abs(z - zfar(ifar))
       wsum = 0.
       do 60 iphi = 1,nphi
          phi = float(iphi)*dphi
          x3 = rfar(ifar)*rfar(ifar)+w*w+zp*zp-
     &             2.*rfar(ifar)*w*cos(phi)
          dwfar = dphi*rfar(ifar)*(rfar(ifar)-w*cos(phi))/
     &                (x3-zp*zp)*(1.-zp/sqrt(x3))
          wsum = wsum + dwfar
 60       continue
       x3 = rfar(ifar)*rfar(ifar)+w*w+zp*zp-2.*rfar(ifar)*w
       if((x3-zp*zp) .ne. 0.)then
          dwfar = dphi*rfar(ifar)*(rfar(ifar)-w)/(x3-zp*zp)*(1.-zp/sqrt(x3))
          wsum = wsum + dwfar
```

```
      endif
      wfar = wsum/4./pi*gfar(ifar)
      wvfar(ifar) = wfar
      k1 = 4.*w*rfar(ifar)/((w+rfar(ifar))**2+zp*zp)
      if (kl .eq. 1.)then
         k  = 10.
         e  = 1.
         goto 62
      endif
      f1 = alog(4./sqrt(1.-k1))
      e  = 1. + .5*(f1-.5)*(1.-k1)+3./16.*(f1-1.-1./12.)*(1.-k1)**2
     &           +0.1171875*(f1-1.-1./6.-1./30.)*(1.-k1)**3
      k  = f1 + .25*(f1-1.)*(1.-k1) + 9./64*(f1-1.-1./6.)*(1.-k1)**2
     &           +0.09765625*(f1-1.-1./6.-1./15.)*(1.-k1)**3
62       continue
      ufar = (sqrt(rfar(ifar)/w/k1))*(k*(2.-k1)-2.*e)*.5/pi
      ufar = abs(ufar)*gfar(ifar)
      uvfar(ifar) = ufar
67       continue
      uvel(i) = 0.
      wvel(i) = 0.
      do i1 = 1,nring
         uvel(i) = uvel(i) + uvnear(i1)
         wvel(i) = wvel(i) + wvnear(i1)
      enddo
      do j1 = 1,level
         uvel(i) = uvel(i) + uvint(1,j1)+uvint(2,j1)+uvint(3,j1)
           wvel(i) = wvel(i) + wvint(1,j1)+wvint(2,j1)+wvint(3,j1)
      enddo
      uvel(i) = uvel(i) + uvfar(1) + uvfar(2) +uvfar(3)
      wvel(i) = wvel(i) + wvfar(1) + wvfar(2) +wvfar(3)
c
C--Calculate final velocity including self-induced component
C
       WV=GAMMA(I)*(ALOG(8.0*W/CORE(I))-.25)/(4.0*PI*W)
```

```
        WVEL(I)=WVEL(I)+WV
C
C--get velocity of higher order terms in tip region, if not tip ring now
C
        IF (I .NE. NRING) THEN
          CALL DOUBLET(W,Z,VXPRM,VYPRM,VRPRM,VTHPRM)
          WVEL(I)=WVEL(I)+VYPRM
          UVEL(I)=UVEL(I)+VXPRM
        ENDIF
C
   10 CONTINUE
C
C--velocity at the intermediate and far wakes
C
      DO 101 I=1,level
      do 101 ii = 1,ninter(i)
        W=rinter(ii,i)
        Z=zinter(ii,i)
       do j1 = 1,level
          do i1 = 1,ninter(j1)
              uvint(i1,j1) = 0.
              wvint(i1,j1) = 0.
          enddo
       enddo
C
C--Loop over all rings again (except control ring) and determine the
C  contribution of each to the induced velocity at intermediate wake ii,i
C
        DO 201 J=1,nring
          WP=RADIUS(J)
          ZP=HEIGHT(J)
          R1=SQRT((Z-ZP)**2+(W-WP)**2)
          R2=SQRT((Z-ZP)**2+(W+WP)**2)
          LAMBDA=(R2-R1)/(R2+R1)
          FK=1.0-LAMBDA**2
```

```
      K = AO+FK*(A1+FK*(A2+FK*(A3+FK*A4)))
     $      +(BO+FK*(B1+FK*(B2+FK*(B3+FK*B4))))*ALOG(1.0/FK)
      E = CO+FK*(C1+FK*(C2+FK*(C3+FK*C4)))
     $      +(DO+FK*(D1+FK*(D2+FK*(D3+FK*D4))))*ALOG(1.0/FK)
      F1=((W-WP)/R1+(W+WP)/R2)*(K-E)
      F2=2.0*E*LAMBDA/(1.0-LAMBDA**2)
      F3=(W*(R1**2-R2**2)+WP*(R1**2+R2**2))/(R1*R2*(R1+R2))
      WV=GAMMA(J)*(F1+F2*F3)/(2.0*PI*W)
      WVnear(j)=WV
      F4=(Z-ZP)*(R1+R2)/(R1*R2)
      F5=K-E*(1.0+LAMBDA**2)/(1.0-LAMBDA**2)
      UV=GAMMA(J)*F4*F5/(2.0*PI*W)
      UVnear(j)=-UV
  201 CONTINUE
c
c   calculate influence of intermediate wake
c
      do 501 j1 = 1,level
         do 501 i1 = 1,ninter(j1)
            if(i1 .eq. ii .and. j1 .eq. i)goto501
              wp=rinter(i1,j1)
               zp=zinter(i1,j1)
               r1=sqrt((z-zp)**2+(w-wp)**2)
               r2=sqrt((z-zp)**2+(w+wp)**2)
               lambda=(r2-r1)/(r2+r1)
               fk=1.0-lambda**2
               k = a0+fk*(a1+fk*(a2+fk*(a3+fk*a4)))
     $        +(b0+fk*(b1+fk*(b2+fk*(b3+fk*b4))))*alog(1.0/fk)
               e = c0+fk*(c1+fk*(c2+fk*(c3+fk*c4)))
     $        +(d0+fk*(d1+fk*(d2+fk*(d3+fk*d4))))*alog(1.0/fk)
                f1=((w-wp)/r1+(w+wp)/r2)*(k-e)
               f2=2.0*e*lambda/(1.0-lambda**2)
               f3=(w*(r1**2-r2**2)+wp*(r1**2+r2**2))/(r1*r2*(r1+r2))
               wv=ginter(i1,j1)*(f1+f2*f3)/(2.0*pi*w)
               wvint(i1,j1)=wv
```

```
                f4=(z-zp)*(r1+r2)/(r1*r2)
                f5=k-e*(1.0+lambda**2)/(1.0-lambda**2)
                uv=ginter(i1,j1)*f4*f5/(2.0*pi*w)
                uvint(i1,j1)=-uv
 501        continue
c
c   calculate influence of far wake cylinder
c
       dphi = pi/20.       !40 integration steps
       nphi = nint(2.*pi/dphi) - 1
       do 671 ifar = 1,nfar
       zp = abs(z - zfar(ifar))
       wsum = 0.
       do 601 iphi = 1,nphi
          phi = float(iphi)*dphi
          x3 = rfar(ifar)*rfar(ifar)+w*w+zp*zp-
     &              2.*rfar(ifar)*w*cos(phi)
          dwfar = dphi*rfar(ifar)*(rfar(ifar)-w*cos(phi))/
     &                 (x3-zp*zp)*(1.-zp/sqrt(x3))
          wsum = wsum + dwfar
 601       continue
       x3 = rfar(ifar)*rfar(ifar)+w*w+zp*zp-2.*rfar(ifar)*w
       if((x3-zp*zp) .ne. 0.)then
          dwfar = dphi*rfar(ifar)*(rfar(ifar)-w)/(x3-zp*zp)*(1.-zp/sqrt(x3))
          wsum = wsum + dwfar
       endif
       wfar = wsum/4./pi*gfar(ifar)
       wvfar(ifar) = wfar
       k1 = 4.*w*rfar(ifar)/((w+rfar(ifar))**2+zp*zp)
       if (k1 .eq. 1.)then
          k  = 10.
          e  = 1.
          goto 621
       endif
       f1 = alog(4./sqrt(1.-k1))
```

```
      e  = 1. + .5*(f1-.5)*(1.-k1)+3./16.*(f1-1.-1./12.)*(1.-k1)**2
     &           +0.1171875*(f1-1.-1./6.-1./30.)*(1.-k1)**3
      k  = f1 + .25*(f1-1.)*(1.-k1) + 9./64*(f1-1.-1./6.)*(1.-k1)**2
     &           +0.09765625*(f1-1.-1./6.-1./15.)*(1.-k1)**3
 621      continue
      ufar = (sqrt(rfar(ifar)/w/k1))*(k*(2.-k1)-2.*e)*.5/pi
      ufar = abs(ufar)*gfar(ifar)
      uvfar(ifar) = ufar
 671      continue
      uveli(ii,i) = 0.
      wveli(ii,i) = 0.
      do i1 = 1,nring
         uveli(ii,i) = uveli(ii,i) + uvnear(i1)
         wveli(ii,i) = wveli(ii,i) + wvnear(i1)
      enddo
      do j1 = 1,level
         uveli(ii,i)=uveli(ii,i)+uvint(1,j1)+uvint(2,j1)+uvint(3,j1)
         wveli(ii,i)=wveli(ii,i)+wvint(1,j1)+wvint(2,j1)+wvint(3,j1)
      enddo
      uveli(ii,i) = uveli(ii,i) + uvfar(1) + uvfar(2) +uvfar(3)
      wveli(ii,i) = wveli(ii,i) + wvfar(1) + wvfar(2) +wvfar(3)
C
C--Calculate final velocity including self-induced component
C
       WV=Ginter(ii,I)*(ALOG(8.0*W/Cinter(ii,I))-.25)/(4.0*PI*W)
       WVELi(ii,I)=WVELi(ii,I)+WV
C
C--get velocity of higher order terms in tip region, if not tip ring now
C
       CALL DOUBLET(W,Z,VXPRM,VYPRM,VRPRM,VTHPRM)
       WVELi(ii,I)=WVELi(ii,I)+VYPRM
       UVELi(ii,I)=UVELi(ii,I)+VXPRM
C
  101 CONTINUE
C
```

```
C--Get velocity contribution at tip due to inner spiral
C
      CALL CENSPIR(VXP,VYP)
      UVEL(NRING)=UVEL(NRING)+VXP
      WVEL(NRING)=WVEL(NRING)+VYP
C
C--Get induced velocity at center by extrapolating the velocities of
C  points 1 and 2.  Fit a parabola (for the velocity) through points
C  0, 1, and 2 by matching the values at points 1 and 2 and setting
C  the slope=0 at r=0.  Then the velocity at r=0 (i.e., the center
C  point) is just the constant part of the parabola.
C
      UVEL(0)=0.0
      WVEL(0)=(WVEL(1)*RADIUS(2)**2-WVEL(2)*RADIUS(1)**2)
     $        /(RADIUS(2)**2-RADIUS(1)**2)
C
C--done
C
      RETURN
      END
```

```
      subroutine initcirc(frac)
c
c   a subroutine to determine the initial circulation distribution
c   from Miller
c   also sets blade geometry.
C
      include '/user/huh/rollup/common.f'
      real circt(103),r(103)
      real lambda,k
      data aslope,sigma/6.15,.0464/
CVD$R ASSOC
c
      nspan = 103
      open(unit=21,file='initload.dat',status='old',form='formatted')
      do i = 1,103
         read(21,*)r(i),circt(i)
         span(i) = r(i)
      enddo
      close(21)
      do 10 i = 1,nspan
         pitch(i) = 0.123 + 0.19*(1.-span(i))
         chord(i) = .07288495
 10      continue
c
      nsmall = nring
      call initsheet(0.)
      dr = .9/float(nsmall)
      do 40 i = 1,nsmall
         r1 = float(i-1)*dr+.1
         r2 = float(i)*dr+.1
         circ1 = spl1(103,r,circt,r1)
         circ2 = spl1(103,r,circt,r2)
         gamma(i) = circ2 - circ1
 40      continue
```

```
      call toteng(tenerg1)
c
c   calculate initial vortex sheet strength
c
      nsmall = nsmall -1
      call initsheet(frac)
      dr = (.9-frac)/float(nsmall)
      do 50 i = 1,nsmall
         r1 = float(i-1)*dr+.1
         r2 = float(i)*dr+.1
         circ1 = spl1(103,r,circt,r1)
         circ2 = spl1(103,r,circt,r2)
         gamma(i) = circ2 - circ1
 50      continue
       HEIGHT(NRING)=0.825*frac
      redg = 1.-frac
      radius(nring) = 1.-(0.52*frac)
      gamma(nring) = -spl1(103,r,circt,redg)
      call rke(tenerg2,nring,nring)
      sum3 = 0.
     DO 80 I=1,NSMALL
      R1=SQRT((HEIGHT(I)-height(nring))**2+
     &    (RADIUS(I)-Radius(nring))**2)
      R2=SQRT((HEIGHT(I)-Height(nring))**2+
     &    (RADIUS(I)+Radius(nring))**2)
      LAMBDA=(R2-R1)/(R2+R1)
      CALL ELLIP(LAMBDA,K,E)
      sum3=sum3+GAMMA(I)*gamma(nring)*(R1+R2)*(K-E)/2.0
   80 CONTINUE
      senergy = tenerg1 - tenerg2-2.*sum3
      czzz = exp(2.*senergy/(gamma(nring)**2*radius(nring))+1.75)
      core(nring) = 8.*radius(nring)/czzz
C
C--Get volume of tip core
C
```

```
        VOLTIP=2.0*PI*PI*RADIUS(NRING)*CORE(NRING)**2
C
C--Center ring
C
        RADIUS(0)=0.1
        HEIGHT(0)=0.0
        DO 8315 I=0,NRING
 8315   HEIGHT(I)=HEIGHT(I)-1.0*TIME(1)
        GMTO=GAMMA(NRING)
       do 19 i = 1,nspan
          circold(i) = spl1(103,r,circt,span(i))
 19       continue
       do i = 1,nspan
          circ(i) = circold(i)
       enddo
       write(7)(radius(i),i=0,nring)
       write(7)(height(i),i=0,nring)
       write(7)(core(i),i=0,nring)
       write(7)(gamma(i),i=0,nring)
       write(7)(axlseg(i),i=0,nring)
       return
       end




       subroutine initsheet(frac)
c
c   sets the strength and the location of the trailing vortex sheet
C
       include '/user/huh/rollup/common.f'
      COMMON /E2/ DUM1,DUM2,DUM3,DUM4,DUM5,AE2
       data iloc/1/
      XO=FRAC
      AO=0.520*XO
```

```
      BO=0.825*XO
C
C--Set core size
C
      CCORE=(1.0/NSMALL)/4.0
C
C--Compute radius, height, and ring circulation for constant spacing
C
      DR=(.9-FRAC)/float(NSMALL)
       expand = 1.
      DO 10 I=1,NSMALL
        HEIGHT(I)=0.0
        CORE(I)=CCORE
        R1=(I-1)*DR+.1
        R2=I*DR+.1
        RADIUS(I)=EXPAND*GETRAD(ILOC,R1,R2)
   10 CONTINUE
C
C--Initialize the arc length array, AXLSEG(I) is the total arc length
C  to the end of segment I.  (Note: segment I is the segment which
C  contains ring or panel I)
C
      AXLSEG(0)=0.0
      DO 11 I=1,NSMALL-1
        AXLSEG(I)=AXLSEG(I-1)
     $            +SQRT( (RADIUS(I+1)-RADIUS(I))**2
     $                 +(HEIGHT(I+1)-HEIGHT(I))**2 )
   11 CONTINUE
      AXLSEG(NSMALL)=AXLSEG(NSMALL-1)
     $              +SQRT( (RADIUS(NSMALL)-RADIUS(NSMALL-1))**2
     $                   +(HEIGHT(NSMALL)-HEIGHT(NSMALL-1))**2)
       return
       end
```

```
      subroutine initwake
c
c   sets the initial intermediate and far wake from Miller's output
C
      include '/user/huh/rollup/common.f'
CVD$R ASSOC
      level = 4
      nfar = 2
       do 134 i = 1,level
         ninter(i) = 3
 134      continue
c
c   root vortex
      zinter(1,1) = -0.147389
      zinter(1,2) = -0.329837
      zinter(1,3) = -0.524966
      zinter(1,4) = -0.720314
      rinter(1,1) = 0.414289
      rinter(1,2) = 0.41414
      rinter(1,3) = 0.405735
      rinter(1,4) = 0.392395
      ginter(1,1:4) = 0.00970935
c
c   center vortex
c
      zinter(2,1) = -0.245905
      zinter(2,2) = -0.482224
      zinter(2,3) = -0.7527273
      zinter(2,4) = -0.9638
      rinter(2,1) = 0.718209
      rinter(2,2) = 0.6980455
      rinter(2,3) = 0.688636
      rinter(2,4) = 0.672146
```

```
      ginter(2,1:4) = 0.0059792
c
c   tip vortex
c
      zinter(3,1) = -0.0612717
      zinter(3,2) = -0.211619
      zinter(3,3) = -0.3981818
      zinter(3,4) = -0.572313
      rinter(3,1) = 0.872495
      rinter(3,2) = 0.797164
      rinter(3,3) = 0.7789773
      rinter(3,4) = 0.769287
      ginter(3,1:4) = -0.020662807
c
c   calculate far wake positions
c
      zdis = abs(zinter(ninter(level),level)-
     &             zinter(ninter(level-1),level-1))
      rfar(nfar) = rinter(ninter(level),level)
      zfar(nfar) = zinter(ninter(level),level)-zdis
      gfar(nfar) = ginter(ninter(level),level)/zdis
      hsum1 = 0.
      zsum1 = 0.
      do 60 i = 1,ninter(level-1)-1
          hsum1 = hsum1 +
     &      ginter(i,level-1)*rinter(i,level-1)*rinter(i,level-1)
          zsum1 = zsum1 +
     &      ginter(i,level-1)*zinter(i,level-1)*rinter(i,level-1)**2
 60       continue
      z1 = zsum1/hsum1
      hsum2 = 0.
      zsum2 = 0.
      gsum2 = 0.
      do 70 i = 1,ninter(level)-1
          hsum2 = hsum2 +
```

```
     &            ginter(i,level)*rinter(i,level)*rinter(i,level)
            zsum2 = zsum2 +
     &            ginter(i,level)*zinter(i,level)*rinter(i,level)**2
            gsum2 = gsum2 + ginter(i,level)
 70       continue
       z2 = zsum2/hsum2
       rfar(nfar-1) = sqrt(hsum2/gsum2)
       zdif    = abs(z2-z1)
       zfar(nfar-1) = z2 - zdif
       gfar(nfar-1) = gsum2/zdif
c
c   set core sizes
c
       do i = 1,3
          cinter(i,1) = rinter(i,1)*.025
          volinter(i,1) = 2.*pi*pi*rinter(i,1)*cinter(i,1)**2
       enddo
       do j = 2,level
          do  i = 1,ninter(j)
             volinter(i,j) = volinter(i,1)
             cinter(i,j) = sqrt(volinter(i,j)/(2.*rinter(i,j)))/pi
          enddo
       enddo
       do j = 1,level
          do i = 1,ninter(j)
               roldi(i,j) = rinter(i,j)
               zoldi(i,j) = zinter(i,j)
               goldi(i,j) = ginter(i,j)
          enddo
       enddo
       return
       end
```

```
      subroutine itercirc(wwake,circt)
c
c   a subroutine to iterate for bound circulation
C
      parameter(maxitrs = 5000)
      include '/user/huh/rollup/common.f'
      common /temp1/a0,a1,a2,a3,a4,b0,b1,b2,b3,b4
      common /temp2/c0,c1,c2,c3,c4,d0,d1,d2,d3,d4
      real wwake(250),circt(250)
      real g(250),r(250),wtotal(250),cold(250),cnew(250)
      logical go
      data epsilon,omega/0.001,0.4/
      data go/.true./
CVD$R ASSOC
      do i = 1,nspan
         if(span(i) .lt. 0.11 .or. span(i) .gt. 0.985)then
             cold(i) = circold(i)
         else
             alphaeff = pitch(i) + wwake(i)/span(i)
             cold(i) = 0.5*chord(i)*aslope*span(i)*alphaeff
         endif
      enddo
      do i = 1,nspan-1
         r(i) = 0.5*(span(i) + span(i+1))
         g(i) = cold(i+1) - cold(i)
      enddo
      itrs = 1
      do while(go .and. itrs .lt. maxitrs)
         go = .false.
          DO I=2,nspan
             W=span(i)
             Z=0.
            wvelt = 0.
             DO J=2,nspan-1
                 WP=r(J)
```

```
            ZP=0.
            R1=SQRT((Z-ZP)**2+(W-WP)**2)
            R2=SQRT((Z-ZP)**2+(W+WP)**2)
             LAMBDA=(R2-R1)/(R2+R1)
            FK=1.0-LAMBDA**2
            K = AO+FK*(A1+FK*(A2+FK*(A3+FK*A4)))
     $     +(BO+FK*(B1+FK*(B2+FK*(B3+FK*B4))))*ALOG(1.0/FK)
             E = CO+FK*(C1+FK*(C2+FK*(C3+FK*C4)))
     $     +(DO+FK*(D1+FK*(D2+FK*(D3+FK*D4))))*ALOG(1.0/FK)
            F1=((W-WP)/R1+(W+WP)/R2)*(K-E)
             F2=2.0*E*LAMBDA/(1.0-LAMBDA**2)
             F3=(W*(R1**2-R2**2)+WP*(R1**2+R2**2))/(R1*R2*(R1+R2))
                 WV=G(J)*(F1+F2*F3)/(2.0*PI*W)
          wvelt = wvelt + wv
        enddo
        wvelt = 0.5*wvelt
        wtotal(i) = wwake(i) + wvelt
      enddo
      do i = 2,nspan
        if(span(i) .le. 0.11 .or. span(i) .ge. 0.985)then
          cnew(i) = cold(i)
        else
          alphaeff = pitch(i) + wtotal(i)/span(i)
          ctemp = 0.5*chord(i)*aslope*span(i)*alphaeff
          cnew(i) = cold(i) + omega*(ctemp-cold(i))
        endif
      enddo
      ispan = 1
      do while((.not. go) .and. ispan .lt. nspan)
        if(cnew(ispan) .gt. 1.e-05)then
                test = (cnew(ispan)-cold(ispan))/cnew(ispan)
           if(abs(test) .ge. epsilon)go = .true.
        endif
        ispan = ispan + 1
      enddo
```

```
            do i = 1,nspan-1
               g(i) = cnew(i+1) - cnew(i)
               cold(i) = cnew(i)
            enddo
            cold(nspan) = cnew(nspan)
            itrs = itrs + 1
         enddo
         do i = 1,nspan
               circt(i) = cnew(i)
         enddo
         return
         end



      SUBROUTINE LUMPS
C
C--This subroutine looks at the rolled up part of the sheet and
C  determines if part of it should be truncated and lumped into
C  the tip vortex ring.
C
C
       include '/user/huh/rollup/common.f'
      COMMON /TH/ THETA(0:900),THMAX
      REAL*4 LAMBDA,K
       real en2i(900),en2(900),en3(900)
CVD$R ASSOC
CVD$R CNCALL
C
C-- Get segment angles relative to horizintal axis
C
      CALL GETANG
C
C--See if any rings exceed the cut-off angle THMAX
C
```

```
      M=0           ! M will contain the # of rings to be lumped
      I=NSMALL-1    ! Highest segment number
   20 IF (THETA(I) .LT. THMAX) GO TO 10
      M=M+1
      I=I-1
      GO TO 20
C
C--Now M contains the # of rings to be lumped into the core
C
   10 IF (M .EQ. 0) GO TO 999           !Return
C
      IF (IWR .EQ. 11) THEN
        IF (IAXI .EQ. 1) CALL TOTENG(TTKE)
        CALL IMPULS(TTIM)
        WRITE(11,4387)TTKE,TTIM
 4387   FORMAT(' START LUMPS: TTKE,TTIM',2F12.6)
                     ENDIF
C
C--Have M rings to lump, first find new position of tip by conserving
C  impulse.
C
C   GTIP  =  new tip circulation
C   RTIP  =  new tip radius
C   HTIP  =  new tip height
C
      GTIP=0.0
      DO 30 I=NSMALL-M+1,NRING
   30 GTIP=GTIP+GAMMA(I)
C
        RT2=0.0
        DO 40 I=NSMALL-M+1,NRING
   40   RT2=RT2+GAMMA(I)*RADIUS(I)**2
        RTIP=SQRT(RT2/GTIP)
        HTIP=0.0
        DO 50 I=NSMALL-M+1,NRING
```

```
   50   HTIP=HTIP+GAMMA(I)*HEIGHT(I)*RADIUS(I)**2
        HTIP=HTIP/RT2
      CTIP=0.0
C
C  Only get new core size for the axi case
C--Get new tip core size by conserving kinetic energy.
C  First compute total kinetic energy of system.
C
      CALL TOTENG(TKE)
C-----------------------------------------------------------------!
C---DEBUG-------------------------------------------------------- !
c      WRITE(6,*)' M,GTIP,RTIP,HTIP,TKE',M,GTIP,RTIP,HTIP,TKE       !
C-----------------------------------------------------------------!
C
C--Now get the various sums needed to compute TTBAR
C
      SUM1=0.0
      DO 60 I=1,NSMALL-M
      TIBAR=GAMMA(I)**2*RADIUS(I)*(ALOG(8.0*RADIUS(I)/CORE(I))
     $      -1.75)/2.0
   60 SUM1=SUM1+TIBAR
C
      DO 70 I=1,NSMALL-M
       do j = 1,nsmall-m
          en2(j) = 0.
       enddo
       if(i .eq. 1)goto73
      DO 72 J=1,i-1
        R1=SQRT((HEIGHT(I)-HEIGHT(J))**2+(RADIUS(I)-RADIUS(J))**2)
        R2=SQRT((HEIGHT(I)-HEIGHT(J))**2+(RADIUS(I)+RADIUS(J))**2)
        LAMBDA=(R2-R1)/(R2+R1)
        CALL ELLIP(LAMBDA,K,E)
        en2(j)=GAMMA(I)*GAMMA(J)*(R1+R2)*(K-E)/2.0
   72 CONTINUE
 73       continue
```

```
        if(i .eq. (nsmall-m))goto75
      DO 74 J=i+1,NSMALL-M
        R1=SQRT((HEIGHT(I)-HEIGHT(J))**2+(RADIUS(I)-RADIUS(J))**2)
        R2=SQRT((HEIGHT(I)-HEIGHT(J))**2+(RADIUS(I)+RADIUS(J))**2)
        LAMBDA=(R2-R1)/(R2+R1)
        CALL ELLIP(LAMBDA,K,E)
        en2(j)=GAMMA(I)*GAMMA(J)*(R1+R2)*(K-E)/2.0
   74 CONTINUE
 75       continue
       en2i(i) = 0.
       do j = 1,nsmall-m
          en2i(i) = en2i(i) + en2(j)
       enddo
 70       continue
      SUM2=0.0
       do i = 1,nsmall-m
          sum2 = sum2 + en2i(i)
       enddo
C
      DO 80 I=1,NSMALL-M
       R1=SQRT((HEIGHT(I)-HTIP)**2+(RADIUS(I)-RTIP)**2)
       R2=SQRT((HEIGHT(I)-HTIP)**2+(RADIUS(I)+RTIP)**2)
       LAMBDA=(R2-R1)/(R2+R1)
       CALL ELLIP(LAMBDA,K,E)
       en3(i)=GAMMA(I)*GTIP*(R1+R2)*(K-E)/2.0
   80 CONTINUE
      SUM3=0.0
       do i = 1,nsmall-m
          sum3 = sum3 + en3(i)
       enddo
C
      SUM4=SUM3
C
C--Get energy contribution of tip.
C
```

```
      TTBAR=TKE-SUM1-SUM2-SUM3-SUM4
C
C--Now get core size corrosponding to this energy.
C
      CZZZ=EXP(2.0*TTBAR/(GTIP**2*RTIP)+1.75)
      CTIP=8.0*RTIP/CZZZ
C
C--Put tip variables into main arrays
C
      RADIUS(NRING)=RTIP
      HEIGHT(NRING)=HTIP
      GAMMA(NRING)=GTIP
      CORE(NRING)=CTIP
C
C--Compute new volume of tip ring
C
      VOLTIP=2.0*PI*PI*RADIUS(NRING)*CORE(NRING)**2
C
C--Now rediscretize remaining portion of sheet.
C
      IF (IWR .EQ. 11) THEN          !this only good for M=1
        RADIUS(NRING-1)=RADIUS(NRING)
        HEIGHT(NRING-1)=HEIGHT(NRING)
        GAMMA(NRING-1)=GAMMA(NRING)
        CORE(NRING-1)=CORE(NRING)
        NRING=NRING-1
        CALL TOTENG(TTKE)
        CALL IMPULS(TTIM)
        NRING=NRING+1
        WRITE(11,4388)TTKE,TTIM
 4388   FORMAT(' LUMPS BEFORE REDISC, TTKE,TTIM',2F12.6)
      ENDIF
      NIN=NSMALL-M
      NOUT=NSMALL
c      GMM=GMAX-GAMMA(NRING)
```

```
      GMM=GAMMA(NRING)
      CALL REDISC(RADIUS,HEIGHT,GAMMA,NIN,NOUT,GMM,SUML,DELTAL)
      IF (IWR .EQ. 11) THEN
        IF (IAXI .EQ. 1) CALL TOTENG(TTKE)
        CALL IMPULS(TTIM)
        WRITE(11,4389)TTKE,TTIM
 4389   FORMAT(' LUMPS AFTER REDISC , TTKE,TTIM',2F12.6)
      ENDIF
C
C--done
C
  999 RETURN
      END




       subroutine makesheet(frac,istep)
c
c    a subroutine to create a new near wake
C
       include '/user/huh/rollup/common.f'
       real lambda,k
CVD$R ASSOC
cCVD$R CNCALL
       nsmall = nring
       call initsheet(0.)
       dr = .9/float(nsmall)
       do 40 i = 1,nsmall
          r1 = float(i-1)*dr+.1
          r2 = float(i)*dr+.1
          circ1 = spl1(nspan,span,circ,r1)
          circ2 = spl1(nspan,span,circ,r2)
          gamma(i) = circ2 - circ1
 40       continue
```

```
      call toteng(tenerg1)
      nsmall = nsmall -1
      call initsheet(frac)
      dr = (.9-frac)/float(nsmall)
      do 50 i = 1,nsmall
         r1 = float(i-1)*dr+.1
         r2 = float(i)*dr+.1
         circ1 = spl1(nspan,span,circ,r1)
         circ2 = spl1(nspan,span,circ,r2)
         gamma(i) = circ2 - circ1
50       continue
       HEIGHT(NRING)= 0.825*frac
      redg = 1.-frac
      radius(nring) = 1.-0.52*frac
      gamma(nring) = -spl1(nspan,span,circ,redg)
      call rke(tenerg2,nring,nring)
      sum3 = 0.
     DO 80 I=1,NSMALL
      R1=SQRT((HEIGHT(I)-height(nring))**2+
     &    (RADIUS(I)-Radius(nring))**2)
      R2=SQRT((HEIGHT(I)-Height(nring))**2+
     &    (RADIUS(I)+Radius(nring))**2)
      LAMBDA=(R2-R1)/(R2+R1)
      CALL ELLIP(LAMBDA,K,E)
      sum3=sum3+GAMMA(I)*gamma(nring)*(R1+R2)*(K-E)/2.0
   80 CONTINUE
      senergy = tenerg1 - tenerg2-2.*sum3
      czzz = exp(2.*senergy/(gamma(nring)**2*radius(nring))+1.75)
      core(nring) = 8.*radius(nring)/czzz
C
C--Get volume of tip core
C
       VOLTIP=2.0*PI*PI*RADIUS(NRING)*CORE(NRING)**2
C
C--Center ring
```

```
C
        RADIUS(0)=0.1
        HEIGHT(0)=0.0
        DO 8315 I=0,NRING
 8315   HEIGHT(I)=HEIGHT(I)-1.0*TIME(1)
        GMTO=GAMMA(NRING)
       time(istep) = time(istep) + time(1)
       close(7)
       open(unit=7,file='sheettemp.dat',status='new',form='unformatted')
       write(7)(radius(i),i=0,nring)
       write(7)(height(i),i=0,nring)
       write(7)(core(i),i=0,nring)
       write(7)(gamma(i),i=0,nring)
       write(7)(axlseg(i),i=0,nring)
       return
       end




       subroutine movem(istep)
       include '/user/huh/rollup/common.f'
CVD$R ASSOC
C
C--Computes the new wake positions using the
C  Runge-Kutta time stepping
C
C--First save original data
C
      ALPO=ALPLAST
      DO 20 I=0,NRING
        RADO(I)=RADIUS(I)
        HEIO(I)=HEIGHT(I)
   20 CONTINUE
       do 25 j = 1,level
```

```
          do 25 i = 1,ninter(j)
              rint0(i,j)=rinter(i,j)
              zint0(i,j)=zinter(i,j)
 25       continue
C
C--Now check for time step method
C
C
        CALL PULANG(NRING-1,BEFORE) !angle at last panel center at start
        TIMENOW=TIME(ISTEP)
        CALL indvel
        CALL GETDT(DT)
        CALL GETK(1)
        CALL UPDATE(DT/2.0,1)
        TIPVAX=WVEL(NRING)
C
        CALL PULANG(NRING-1,ANOW)  !get new angle and compute the change
        ALPLAST=ALPO+(ANOW-BEFORE) !and add it to the angle of the edge
        TIMENOW=TIME(ISTEP)+DT/2.0
        CALL indvel
        CALL GETK(2)
        CALL UPDATE(DT/2.0,2)
C
        CALL PULANG(NRING-1,ANOW)
        ALPLAST=ALPO+(ANOW-BEFORE)
        TIMENOW=TIME(ISTEP)+DT/2.0
        CALL indvel
        CALL GETK(3)
        CALL UPDATE(DT,3)
C
        CALL PULANG(NRING-1,ANOW)
        ALPLAST=ALPO+(ANOW-BEFORE)
        TIMENOW=TIME(ISTEP)+DT
        CALL indvel
        CALL GETK(4)
```

```
C
C--Update positions for Runge-Kutta
C
        DO 10 I=0,NRING
          RADIUS(I)=RADO(I)+(DT/6.0)*(RK(I,1)+2.0*RK(I,2)+
     $              2.0*RK(I,3)+RK(I,4))
          HEIGHT(I)=HEIO(I)+(DT/6.0)*(HK(I,1)+2.0*HK(I,2)+
     $              2.0*HK(I,3)+HK(I,4))
   10   CONTINUE
        DO 101 j = 1,level
       do 101 i = 1,ninter(j)
          Rinter(I,j)=RintO(I,j)+(DT/6.0)*(RKi(I,j,1)+2.0*RKi(I,j,2)+
     $              2.0*RKi(I,j,3)+RKi(I,j,4))
          zinter(I,j)=zintO(I,j)+(DT/6.0)*(HKi(I,j,1)+2.0*HKi(I,j,2)+
     $              2.0*HKi(I,j,3)+HKi(I,j,4))
  101   CONTINUE
c
c   far wake
c
       rfar(2) = rinter(ninter(4),4)
       zdis = abs(zinter(ninter(4),4)-zinter(ninter(3),3))
       zfar(2) = zinter(ninter(4),4)-zdis
       gfar(2) = ginter(ninter(4),4)/zdis
       hsum1 = 0.
       zsum1 = 0.
       do 60 i = 1,ninter(3)-1
          hsum1 = hsum1 + ginter(i,3)*rinter(i,3)*rinter(i,3)
          zsum1 = zsum1 + ginter(i,3)*zinter(i,3)*rinter(i,3)**2
 60       continue
       z1 = zsum1/hsum1
       hsum2 = 0.
       zsum2 = 0.
       gsum2 = 0.
       do 70 i = 1,ninter(4)-1
          hsum2 = hsum2 + ginter(i,4)*rinter(i,4)*rinter(i,4)
```

```
          zsum2 = zsum2 + ginter(i,4)*zinter(i,4)*rinter(i,4)**2
          gsum2 = gsum2 + ginter(i,4)
 70       continue
       z2 = zsum2/hsum2
       rfar(1) = sqrt(hsum2/gsum2)
       zdif    = abs(z2-z1)
       zfar(1) = z2 - zdif
       gfar(1) = gsum2/zdif
c
c--compute new core radii for intermediate vorticies
c
       do 27 j = 1,level
          do 27 i = 1,ninter(j)
              cinter(i,j) =SQRT(VOLinter(i,j)/
     &                          (2.0*Rinter(i,j)))/PI
 27       continue
C
C--Compute the new core radius for the tip ring and update time
C
      CORE(NRING)=SQRT(VOLTIP/(2.0*RADIUS(NRING)))/PI
      TIME(ISTEP+1)=TIME(ISTEP)+DT
C
      RETURN
      END




       subroutine newwake(indexvort,nvort,lrite,lgo)
c
c   subroutine to form new 1st level intermediate wake from rolled-up
c   near wake
c
       include '/user/huh/rollup/common.f'
       integer indexvort(10)
```

```
      integer ivstart(10)
      character*60 gtitle,title
      real rtemp(10),ztemp(10),gtemp(10)
      real rfirst(3),zfirst(3),gfirst(3),lambda,k
      logical lrite,lgo,lstart
      data omegaw,epsilon/1.0,0.001/
      data rmin/0.2/
CVD$R ASSOC
      title = 'r/R~z/R~time = '
      istart = 1
      do 10 i = 1,nvort
         gtemp(i) = 0.
         hsum = 0.
         zsum = 0.
         lstart = .false.
         do 20 j = istart,indexvort(i)-1
              if(radius(j) .ge. rmin)then
                if(.not. lstart)then
                   ivstart(i) = j
                   lstart = .true.
                endif
                gtemp(i) = gtemp(i) + gamma(j)
                hsum = hsum + gamma(j)*radius(j)*radius(j)
                zsum = zsum + gamma(j)*height(j)*radius(j)*radius(j)
              endif
 20           continue
         if(     abs(gtemp(i)) .le. 1.e-05
     &             .or. abs(hsum)     .le. 1.e-05)then
              ztemp(i) = 1.
              goto10
         endif
         rtemp(i) = sqrt(hsum/gtemp(i))
         ztemp(i) = zsum/hsum
         istart = indexvort(i)
 10      continue
```

```
c
c   discard vorticies if they are above the plane of the rotor
c   or if they are too close to the blade
c
        i = 1
 15       continue
       if(ztemp(i) .ge. 0. .or. rtemp(i) .le. rmin)then
          do 17 j = i,nvort-1
             rtemp(j) = rtemp(j+1)
             ztemp(j) = ztemp(j+1)
             gtemp(j) = gtemp(j+1)
             ivstart(j) = ivstart(j+1)
 17          continue
          nvort = nvort -1
       else
          i = i+1
       endif
       if(i .ne. nvort)then
          goto 15
       endif
c
c   repack arrays
c
       do 40 j = 4,2,-1
          do 40 i = 1,ninter(j-1)
             rinter(i,j) = roldi(i,j) + omegaw*(rinter(i,j-1)-roldi(i,j))
             zinter(i,j) = zoldi(i,j) + omegaw*(zinter(i,j-1)-zoldi(i,j))
             ninter(j)   = ninter(j-1)
 40       continue
c
c   determine 1st level of intermediate wake from rolled-up near wake
c
       call toteng(tke)
       groot = gtemp(1)
       hroot = gtemp(1)*rtemp(1)**2
```

```
  zroot = gtemp(1)*ztemp(1)*rtemp(1)**2
  rfirst(1) = sqrt(hroot/groot)
  zfirst(1) = zroot/hroot
  gfirst(1) = groot
  call rke(eroot,ivstart(1),ivstart(2)-1)
  sum3 = 0.
        DO I=1,ivstart(1)-1
           R1=SQRT((HEIGHT(I)-zfirst(1))**2+
&         (RADIUS(I)-rfirst(1))**2)
           R2=SQRT((HEIGHT(I)-zfirst(1))**2+
&         (RADIUS(I)+rfirst(1))**2)
           LAMBDA=(R2-R1)/(R2+R1)
       CALL ELLIP(LAMBDA,K,E)
       sum3=sum3+GAMMA(I)*gfirst(1)*(R1+R2)*(K-E)/2.0
  enddo
   DO I=ivstart(2),nring
       R1=SQRT((HEIGHT(I)-zfirst(1))**2+
&         (RADIUS(I)-rfirst(1))**2)
       R2=SQRT((HEIGHT(I)-zfirst(1))**2+
&         (RADIUS(I)+rfirst(1))**2)
       LAMBDA=(R2-R1)/(R2+R1)
       CALL ELLIP(LAMBDA,K,E)
       sum3=sum3+GAMMA(I)*gfirst(1)*(R1+R2)*(K-E)/2.0
  enddo
  senergy = tke - eroot - 2.*sum3
  czzz = exp(2.*senergy/(gfirst(1)**2*rfirst(1))+1.75)
  croot = 8.*rfirst(1)/czzz
  volinter(1,1) = 2.*pi*pi*rfirst(1)*croot**2
  gcent = 0.
  hcent = 0.
  zcent = 0.
  do i = 2,nvort-1
     gcent = gcent + gtemp(i)
     hcent = hcent + gtemp(i)*rtemp(i)**2
     zcent = zcent + gtemp(i)*ztemp(i)*rtemp(i)**2
```

```
  enddo
  rfirst(2) = sqrt(hcent/gcent)
  zfirst(2) = zcent/hcent
  gfirst(2) = gcent
  call rke(ecent,ivstart(2),nsmall)
  sum3 = 0.
        DO I=1,ivstart(2)-1
            R1=SQRT((HEIGHT(I)-zfirst(2))**2+
&        (RADIUS(I)-rfirst(2))**2)
            R2=SQRT((HEIGHT(I)-zfirst(2))**2+
&        (RADIUS(I)+rfirst(2))**2)
            LAMBDA=(R2-R1)/(R2+R1)
       CALL ELLIP(LAMBDA,K,E)
       sum3=sum3+GAMMA(I)*gfirst(2)*(R1+R2)*(K-E)/2.0
  enddo
   R1=SQRT((HEIGHT(nring)-zfirst(2))**2+
&     (RADIUS(nring)-rfirst(2))**2)
   R2=SQRT((HEIGHT(nring)-zfirst(2))**2+
&     (RADIUS(nring)+rfirst(2))**2)
   LAMBDA=(R2-R1)/(R2+R1)
   CALL ELLIP(LAMBDA,K,E)
   sum3=sum3+GAMMA(nring)*gfirst(2)*(R1+R2)*(K-E)/2.0
  senergy = tke - ecent - 2.*sum3
  czzz = exp(2.*senergy/(gfirst(2)**2*rfirst(2))+1.75)
  ccent = 8.*rfirst(2)/czzz
  volinter(2,1) = 2.*pi*pi*rfirst(2)*ccent**2
  rfirst(3) = rtemp(nvort)
  zfirst(3) = ztemp(nvort)
  gfirst(3) = gtemp(nvort)
  volinter(3,1) = voltip
  do i = 1,3
     rinter(i,1) = roldi(i,1) + omegaw*(rfirst(i) - roldi(i,1))
     zinter(i,1) = zoldi(i,1) + omegaw*(zfirst(i) - zoldi(i,1))
     ginter(i,1) = gfirst(i)
  enddo
```

```
      ninter(1) = 3
c
c   set vortex strengths equal to each other
c
      do j = 2,level
         do i = 1,ninter(j)
            ginter(i,j) = ginter(i,1)
            volinter(i,j) = volinter(i,1)
         enddo
      enddo
c
c   set new core sizes
c
      do j = 1,level
         do i = 1,ninter(j)
            cinter(i,j) = sqrt(volinter(i,j)/(2.*rinter(i,j)))/pi
         enddo
      enddo
c
c   check for convergence of tip vortex
c
      lgo = .false.
      do j = 1,level
         disnew = rinter(3,j)**2 + zinter(3,j)**2
         disold = roldi(3,j)**2 + zoldi(3,j)**2
         test  = abs((disnew-disold)/disold)
         if(test  .gt. epsilon )then
            lgo = .true.
            goto 33
            endif
      enddo
      lrite = .true.
      type*,'wake convergence at time =',timenow
      title = 'r/R~z/R~CONVERGED WAKE time = '
 33      continue
```

```
c
c   calculate new far wake positions
c
       rfar(2) = rinter(ninter(4),4)
       zdis = abs(zinter(ninter(4),4)-zinter(ninter(3),3))
       zfar(2) = zinter(ninter(4),4)-zdis
       gfar(2) = ginter(ninter(4),4)/zdis
       hsum1 = 0.
       zsum1 = 0.
       do 60 i = 1,ninter(3)-1
          hsum1 = hsum1 + ginter(i,3)*rinter(i,3)*rinter(i,3)
          zsum1 = zsum1 + ginter(i,3)*zinter(i,3)*rinter(i,3)**2
 60       continue
       z1 = zsum1/hsum1
       hsum2 = 0.
       zsum2 = 0.
       gsum2 = 0.
       do 70 i = 1,ninter(4)-1
          hsum2 = hsum2 + ginter(i,4)*rinter(i,4)*rinter(i,4)
          zsum2 = zsum2 + ginter(i,4)*zinter(i,4)*rinter(i,4)**2
          gsum2 = gsum2 + ginter(i,4)
 70       continue
       z2 = zsum2/hsum2
       rfar(1) = sqrt(hsum2/gsum2)
       zdif    = abs(z2-z1)
       zfar(1) = z2 - zdif
       gfar(1) = gsum2/zdif
c
c   set old wake values
c
       do j = 1,level
          do i = 1,ninter(j)
              roldi(i,j) = rinter(i,j)
              zoldi(i,j) = zinter(i,j)
              goldi(i,j) = ginter(i,j)
```

```
         enddo
      enddo
      if(lrite)then
      call prntout
      gtitle = 'New geometry'
      write(19,*) gtitle
      write(19,*) level
      write(19,*) 52
      write(19,12) title,timenow
12       format(a30,f9.5)
      write(19,*) 1
      write(19,*) 0.,1.5,-1.18,.02
      write(17,*)'New geometry time = ',timenow
      write(17,*)'level = ',level
      do 30 j = 1,level
         write(19,*) 6
         write(17,*)' '
         write(17,*)'for level#',j
         write(17,*)'# of vorticies = ',ninter(j)
         write(19,*) ninter(j)
         do 30 i = 1,ninter(j)
             write(19,*) rinter(i,j),zinter(i,j)
             write(17,*)i,rinter(i,j),zinter(i,j),ginter(i,j)
     &                          ,cinter(i,j)
30       continue
      write(17,*)' '
      write(17,*)' far wake'
      write(17,*)'nfar =',nfar
      do 99 i = 1,nfar
         write(17,*)i,rfar(i),zfar(i),gfar(i)
99       continue
      lrite = .false.
      endif
      return
      end
```

```
      subroutine oldsheet(istep)
c
c   a subroutine to read in old near wake values if the geometry has
c   not converged
C
      include '/user/huh/rollup/common.f'
      rewind(7)
      read(7)(radius(i),i=0,nring)
      read(7)(height(i),i=0,nring)
      read(7)(core(i),i=0,nring)
      read(7)(gamma(i),i=0,nring)
      read(7)(axlseg(i),i=0,nring)
      gmt0=gamma(nring)
      time(istep) = time(istep)+time(1)
      voltip = 2.*pi*pi*radius(nring)*core(nring)**2
      return
      end




      subroutine prntout
c
c   a subroutine to printout the final results
c

      include '/user/huh/rollup/common.f'
      common /oldgeom/radiusold(0:900),heightold(0:900)
      character*60 gtitle,ltitle,title
      real expwx(4),expwy(4)
      data expwx/.77,.783334,.81,.86/
```

```
      data expwy/-.5966666,-.4133333,-.243333,-7.6666594e-02/
c
c   write out wake geometry
c
      gtitle = 'WAKE POSITIONS'
      lTITLE = 'RADIUS~HEIGHT~TIME = '
      write(12,*) gtitle
      write(12,*) level+5
      write(12,*) 52
      write(12,12) lTITLE,timenow
 12      format(a21,f9.5)
      WRITE(12,*) 1
      write(12,*) 0.,1.5,-1.18,.02
      write(12,*) 2
      write(12,*) 2
      write(12,*) 0.1,0.0
      write(12,*) 1.0,0.0
      write(12,*) 6
      write(12,*)nring
CVD$ NOVECTOR
CVD$ NOCONCUR
      do 151 i = 1,nring
         write(12,*)radius(i),height(i)
 151      continue
CVD$ NOVECTOR
CVD$ NOCONCUR
      do 163 j = 1,level
        write(12,*)6
        write(12,*)ninter(j)
CVD$ NOVECTOR
CVD$ NOCONCUR
        do 183 i = 1,ninter(j)
             write(12,*) rinter(i,j),zinter(i,j)
 183         continue
 163      continue
```

```
      data expwy/-.5966666,-.4133333,-.243333,-7.6666594e-02/
c
c   write out wake geometry
c
      gtitle = 'WAKE POSITIONS'
      lTITLE = 'RADIUS~HEIGHT~TIME = '
      write(12,*) gtitle
      write(12,*) level+5
      write(12,*) 52
      write(12,12) lTITLE,timenow
 12      format(a21,f9.5)
      WRITE(12,*) 1
      write(12,*) 0.,1.5,-1.18,.02
      write(12,*) 2
      write(12,*) 2
      write(12,*) 0.1,0.0
      write(12,*) 1.0,0.0
      write(12,*) 6
      write(12,*)nring
CVD$ NOVECTOR
CVD$ NOCONCUR
      do 151 i = 1,nring
         write(12,*)radius(i),height(i)
 151      continue
CVD$ NOVECTOR
CVD$ NOCONCUR
      do 163 j = 1,level
        write(12,*)6
        write(12,*)ninter(j)
CVD$ NOVECTOR
CVD$ NOCONCUR
        do 183 i = 1,ninter(j)
             write(12,*) rinter(i,j),zinter(i,j)
 183         continue
 163      continue
```

```
       write(12,*)4
       write(12,*)4
CVD$ NOVECTOR
CVD$ NOCONCUR
       do i = 1,4
          write(12,*)expwx(i),expwy(i)
       enddo
       write(12,*)2
       write(12,*)2
       write(12,*)rfar(1),zfar(1)
       write(12,*)rfar(1),-1.18
       write(12,*)2
       write(12,*)2
       write(12,*)rfar(2),zfar(2)
       write(12,*)rfar(2),-1.18
c
c   write out velocity vectors
c
       gtitle='Velocity vectors'
       title='r/R~z/R~time = '
       write(26,*) gtitle
       write(26,*) 52
       write(26,13) title,timenow
 13       format(a15,f9.5)
       write(26,*) 1
       write(26,*) 0.,1.,-.6,.15
       write(26,*) nsmall
       write(26,*) 2
CVD$ NOVECTOR
CVD$ NOCONCUR
       do 27 i = 1,nsmall
          write(26,*) radius(i),height(i),uvel(i),wvel(i)
 27       continue
CVD$ NOVECTOR
CVD$ NOCONCUR
```

```
      do 271 i = 1,nsmall
         write(26,*) radius(i),height(i),0.,0.
271       continue
      return
      end




      SUBROUTINE PULANG(NIN,THNIN)
      include '/user/huh/rollup/common.f'
      real r(0:900),z(0:900)
      equivalence(r,radius)
      equivalence(z,height)
      COMMON /TEMP/ BETA(0:900)
      REAL*4 MAG
C
      RT=R(NRING)
      ZT=Z(NRING)
C
      A=-1.0
      B=0.0
      C=R(0)-RT
      D=Z(0)-ZT
      DOT=A*C+B*D
      CROSS=A*D-B*C
      MAG=SQRT((A*A+B*B)*(C*C+D*D))
      ARG=DOT/MAG
      IF (ABS(ARG) .GT. 1.0) ARG=SIGN(1.0,ARG)
      BETA(0)=SIGN(1.0,CROSS)*ACOS(ARG)
      btemp = beta(0)
C
      DO 100 I=1,NIN
      A=R(I-1)-RT
      B=Z(I-1)-ZT
```

```
      C=R(I)-RT
      D=Z(I)-ZT
      DOT=A*C+B*D
      CROSS=A*D-B*C
      MAG=SQRT((A*A+B*B)*(C*C+D*D))
      ARG=DOT/MAG
      IF (ABS(ARG) .GT. 1.0) ARG=SIGN(1.0,ARG)
      BETA(I)=btemp+SIGN(1.0,CROSS)*ACOS(ARG)
       btemp = beta(i)
 100      continue
C
      THNIN=BETA(NIN)
C
      RETURN
      END
```

```
      subroutine readinput(tint,itrs)
c
c   a subroutine to read in initial values for restart
C
      include '/user/huh/rollup/common.f'
CVD$R ASSOC
c
      read(16) nring,lineup,minsect,maxsegs
      read(16) time(2),itrs
      read(16) tint
      nsmall = nring-1
      read(16) level,nfar
      read(16) (ninter(i),i=1,level)
      do i = 1,3
         do j = 1,4
             read(16)rinter(i,j)
             read(16)zinter(i,j)
             read(16)ginter(i,j)
             read(16)cinter(i,j)
         enddo
      enddo
      do i = 1,3
         volinter(i,1) = 2.*pi*pi*rinter(i,1)*cinter(i,1)**2
         do j = 2,4
             volinter(i,j) = volinter(i,1)
         enddo
      enddo
      read(16)(rfar(i),i=1,nfar)
      read(16)(zfar(i),i=1,nfar)
      read(16)(gfar(i),i=1,nfar)
      do i = 1,3
         do j = 1,4
             roldi(i,j) = rinter(i,j)
             zoldi(i,j) = zinter(i,j)
```

```
            goldi(i,j) = ginter(i,j)
         enddo
      enddo
      read(16)aslope,sigma
      read(16)nspan
      read(16)(span(i),i=1,nspan)
      read(16)(pitch(i),i=1,nspan)
      read(16)(chord(i),i=1,nspan)
      read(16)(circold(i),i=1,nspan)
      read(16)(radius(i),i=0,nring)
      read(16)(height(i),i=0,nring)
      read(16)(gamma(i),i=0,nring)
      read(16)(core(i),i=0,nring)
      read(16)(axlseg(i),i=0,nsmall)
      voltip = 2.*pi*pi*radius(nring)*core(nring)**2
      gmt0 = gamma(nring)
      write(7)(radius(i),i=0,nring)
      write(7)(height(i),i=0,nring)
      write(7)(core(i),i=0,nring)
      write(7)(gamma(i),i=0,nring)
      write(7)(axlseg(i),i=0,nring)
      return
      end



      SUBROUTINE REDISC(R,Z,DGAM,NIN,NOUT,GMAX,SUML,DELTAL)
C
C--This does curve fitting using blended parabolas on R, Z, DGAM
C
C
       include  '/user/huh/rollup/common.f'
      COMMON/STAR/ICIRC(10),FCIRC(10),SSTAR(10),GSTAR(10),NSTAR
      DIMENSION R(0:900),Z(0:900),DGAM(0:900),S(0:900),QR(0:900),
     $          QZ(0:900),A0(0:900),A1(0:900),A2(0:900),A3(0:900),
```

```
     $          B0(0:900),B1(0:900),B2(0:900),B3(0:900),C0(0:900),
     $          C1(0:900),C2(0:900),C3(0:900),GMID(0:900),SMID(0:900),
     $          Q(0:900),RS(0:8100),ZS(0:8100),AL(0:8100),
     $          THSUB(0:8100),ARCG(0:900),GPL(0:900)
      DIMENSION ROLD(0:900),ZOLD(0:900),DGAMOLD(0:900)
      LOGICAL LINEAR          !LINEAR = .TRUE. for linear circ. fit
      DATA LINEAR/.FALSE./    !LINEAR = .FALSE. for cubic circ. fit
      DATA NSUB/9/
C
CVD$R ASSOC
CVD$R CNCALL
      ILOOPER=0
C
      DO 1 I=0,NRING
      ROLD(I)=R(I)
      ZOLD(I)=Z(I)
    1 DGAMOLD(I)=DGAM(I)
C
C--Get straight line distances between points, S(I) is the total
C   distance along the straight lines to node I
C
      S(0)=0.0
      DO 20 I=1,NIN
   20 S(I)=S(I-1)+SQRT((Z(I)-Z(I-1))**2+(R(I)-R(I-1))**2)
C
C--Get value of s for the trailing extended segment, this value
C  of S is out at the presumed end of the sheet.
C
      S(NIN+1)=S(NIN)+0.5*(S(NIN)-S(NIN-1))
C
C--Get slopes at each interior node, QR(I)=D(R)/DS at node I
C
      DO 30 I=1,NIN-1
      S1=(S(I+1)-S(I))/(S(I-1)-S(I))
      S2=1.0/S1
```

```
      S3=1.0/(S(I+1)-S(I-1))
      QR(I)=((R(I-1)-R(I))*S1-(R(I+1)-R(I))*S2)*S3
   30 QZ(I)=((Z(I-1)-Z(I))*S1-(Z(I+1)-Z(I))*S2)*S3
C
C--Coef's for interior segments
C
      DO 40 I=1,NIN-2
      S1=1.0/(S(I+1)-S(I))
      AO(I)=R(I)
      A1(I)=QR(I)
      A2(I)=S1*(-QR(I+1)-2*QR(I)+3*S1*(R(I+1)-R(I)))
      A3(I)=S1*S1*(QR(I+1)+QR(I)-2*S1*(R(I+1)-R(I)))
      BO(I)=Z(I)
      B1(I)=QZ(I)
      B2(I)=S1*(-QZ(I+1)-2*QZ(I)+3*S1*(Z(I+1)-Z(I)))
   40 B3(I)=S1*S1*(QZ(I+1)+QZ(I)-2*S1*(Z(I+1)-Z(I)))
C
C--Coef's for first segment
C
      S1=S(1)-S(0)
      S2=S(2)-S(0)
      S3=1.0/(S(2)-S(1))
      AO(0)=R(0)
      A1(0)=((R(1)-R(0))*S2/S1-(R(2)-R(0))*S1/S2)*S3
      A2(0)=((R(2)-R(0))/S2-(R(1)-R(0))/S1)*S3
      A3(0)=0.0
      BO(0)=Z(0)
      B1(0)=((Z(1)-Z(0))*S2/S1-(Z(2)-Z(0))*S1/S2)*S3
      B2(0)=((Z(2)-Z(0))/S2-(Z(1)-Z(0))/S1)*S3
      B3(0)=0.0
C
C--Coef's for last segment
C
      N=NIN
      S1=S(N)-S(N-1)
```

```
      S2=S(N-2)-S(N-1)
      S3=1.0/(S(N-2)-S(N))
      A0(N-1)=R(N-1)
      A1(N-1)=((R(N)-R(N-1))*S2/S1-(R(N-2)-R(N-1))*S1/S2)*S3
      A2(N-1)=((R(N-2)-R(N-1))/S2-(R(N)-R(N-1))/S1)*S3
      A3(N-1)=0.0
      B0(N-1)=Z(N-1)
      B1(N-1)=((Z(N)-Z(N-1))*S2/S1-(Z(N-2)-Z(N-1))*S1/S2)*S3
      B2(N-1)=((Z(N-2)-Z(N-1))/S2-(Z(N)-Z(N-1))/S1)*S3
      B3(N-1)=0.0
C
C--Get the angle of the ring with index NSMALL wrt the tip core
C
      CALL PULANG(NIN,THNIN)
C
C--Get the "A" coefficient for the Kadin spiral
C
      RNU=XNU(THNIN)
      IF (ABS(SIN(THNIN)) .GE. 0.5) THEN
        A=(Z(NRING)-Z(NIN))/(THNIN**(-RNU)*SIN(THNIN))
                                    ELSE
        A=(R(NRING)-R(NIN))/(THNIN**(-RNU)*COS(THNIN))
                                    ENDIF
C
C--Get array of R and Z at the subinterval locations, and compute
C   the total length of the curve
C
C--First do the blended parabola segments
C
      RS(0)=A0(0)
      ZS(0)=B0(0)
      AL(0)=0.0
      DO 50 I=0,NIN-1
      DO 50 KSUB=1,NSUB
      J=I*NSUB+KSUB
```

```
      T=(S(I+1)-S(I))*FLOAT(KSUB)/FLOAT(NSUB)
      RS(J)=AO(I)+T*(A1(I)+T*(A2(I)+T*A3(I)))
      ZS(J)=BO(I)+T*(B1(I)+T*(B2(I)+T*B3(I)))
   50 AL(J)=AL(J-1)+SQRT((RS(J)-RS(J-1))**2+(ZS(J)-ZS(J-1))**2)
C
C--The next few sections find the value of theta at the end of the
C  trailing extended segment, i.e., gets the theta corrosponding
C  to arclength S(NIN+1)
C
C--First, before doing the numerical integration, need to get a
C  value to use for delta theta
C
      DS=S(NIN+1)-S(NIN)
      RNU=XNU(THNIN)
      DTG=DS/(SQRT(RNU**2+THNIN**2)*A*THNIN**(-RNU-1.0))
      DTG=DTG/2.0
C
C--Now have delta theta (i.e., DTG) which should yield several
C  steps in the numerical integration from S(NIN) to S(NIN+1).
C
C--Now do the numerical integration and search for theta(NIN+1)
C
      ITERS=0
      SEND=S(NIN+1)-S(NIN)
      STH=0.0
      TH=THNIN
   64 ITERS=ITERS+1
      IF (ITERS .GT. 10000) STOP 'ITERS .GT. 10000 LOOP 63'
      RNU=XNU(TH)
      R1=R(NRING)-A*COS(TH)*TH**(-RNU)
      Z1=Z(NRING)-A*SIN(TH)*TH**(-RNU)
      RNU=XNU(TH+DTG)
      R2=R(NRING)-A*COS(TH+DTG)*(TH+DTG)**(-RNU)
      Z2=Z(NRING)-A*SIN(TH+DTG)*(TH+DTG)**(-RNU)
      DS=SQRT((R2-R1)**2+(Z2-Z1)**2)
```

```
      IF (STH+DS .GE. SEND) GO TO 63
      STH=STH+DS
      TH=TH+DTG
      GO TO 64
   63 FRAC=(SEND-STH)/DS
      THNIN1=TH+FRAC*DTG
C
C--Now can get the sub-interval locations over the extended spiral
C  segment.
C
      DTH=(THNIN1-THNIN)/NSUB
      DO 65 KSUB=1,NSUB
      THETA=THNIN+KSUB*DTH
      J=NIN*NSUB+KSUB
      RNU=XNU(THETA)
      RS(J)=R(NRING)-A*THETA**(-RNU)*COS(THETA)
      ZS(J)=Z(NRING)-A*THETA**(-RNU)*SIN(THETA)
      AL(J)=AL(J-1)+SQRT((RS(J)-RS(J-1))**2+(ZS(J)-ZS(J-1))**2)
   65 CONTINUE
C
C--Total length of curve is SUML
C
      SUML=AL((NIN+1)*NSUB)
C
  689 CONTINUE
C
C--Check for lineup of spiral sheet segments
C
      IF (LINEUP .EQ. 1) THEN
        GO TO 2010            !line up the segments
      ELSE
        GO TO 2020            !for equal sized segments
      ENDIF
C
C--This section for equal sized segments
```

```
C
C--Get new R and Z points at distance DELTAL apart, these are the
C  centroids
C
 2020 DELTAL=SUML/NOUT
      XLSEG=DELTAL/2.0
      AXLSEG(0)=0.0
      I=1
      DO 60 J=1,NSUB*(NIN+1)
   80 IF (XLSEG .LT. AL(J-1) .OR. XLSEG .GT. AL(J)) GO TO 60
      FRAC=(XLSEG-AL(J-1))/(AL(J)-AL(J-1))
      R(I)=abs(FRAC*(RS(J)-RS(J-1))+RS(J-1))
      Z(I)=FRAC*(ZS(J)-ZS(J-1))+ZS(J-1)
      AXLSEG(I)=XLSEG+DELTAL/2.0  !running total of arc length to
                                  !  segment edge.
      WPAN(I)=DELTAL/2.0          !panel half-width
      I=I+1
      XLSEG=XLSEG+DELTAL
      IF (I .GT. NOUT) GO TO 70
      GO TO 80
   60 CONTINUE
   70 CONTINUE
      GO TO 2030
C
C--This part is for lined up segments in the spiral region
C
 2010 NOUT1=NOUT
      NTOTSUB=NSUB*(NIN+1)
      CALL ANGLES(RS,ZS,NTOTSUB,THSUB)
      ALPLAST=THSUB(NTOTSUB) !save angle at edge of sheet for DOUBLET
      IF (THSUB(NTOTSUB) .LE. 9.0*PI/2.0) THEN !ALFSPIR is the angle
        ALFSPIR=THSUB(NTOTSUB)-3.0*PI/2.0      !at which the "spiral"
      ELSE                                     !part of the sheet begins
        ALFSPIR=3.0*PI                         !(i.e., the equal angular
      ENDIF                                    !segments)
```

```
      IF (THSUB(NTOTSUB) .LT. 7.0*PI/2.0) GO TO 2020 !check for at
                                                      !least 1.5 loop
      STOTAL=SUML                                  !total length of sheet
      I=1                                          !find point on sheet where
 3513 IF (THSUB(I) .GT. ALFSPIR) GO TO 3512        !the spiral section is
      I=I+1                                        !assumed to start, at
      GO TO 3513                                   !theta=ALFSPIR
 3512 SFLAT=AL(I)
      SSPIRAL=STOTAL-SFLAT                !length of spiral part of sheet
      R3PIQ2=SQRT((RS(I)-ROLD(NRING))**2+(ZS(I)-ZOLD(NRING))**2)
      RAVERAGE=0.0
      DO 3511 J=I,NTOTSUB
 3511 RAVERAGE=RAVERAGE
     $      +SQRT((RS(J)-ROLD(NRING))**2+(ZS(J)-ZOLD(NRING))**2)
      RAVERAGE=RAVERAGE/(NTOTSUB-I+1)
 7126 DENOM=1.0+(SSPIRAL*R3PIQ2)/(SFLAT*RAVERAGE)
      NSEG=NOUT1                          !# of sheet segments to be output
      NSEGFLAT=INT(NSEG/DENOM+0.5)        !estimate the # of segments on
      NSEGSPIR=NSEG-NSEGFLAT              !the flat and spiral parts
      RNLOOP=(THSUB(NTOTSUB)-ALFSPIR)/(2.0*PI)  !# of loops in spiral
      NTHSECT=INT((FLOAT(NSEGSPIR)/RNLOOP)+0.5) !# of theta sections
      IF (NOUT1 .LT. MAXSEGS) THEN
        IF (NTHSECT .LT. MINSECT) THEN
          ILOOPER=ILOOPER+1
          IF (ILOOPER .GT. MAXSEGS) STOP ' ILOOPER .GT. MAXSEGS'
          NOUT1=NOUT1+1
          GO TO 7126
        ENDIF
      ENDIF
c       if(nthsect .eq. 0)return
      DTHETA=2.0*PI/NTHSECT          !theta increment of a theta section
      I=1                            !segment counter for spiral region
      ISEG=NSEG+1-I                  !segment index, we are going backwards
      THEDGE=THSUB(NTOTSUB)-DTHETA  !edge of current segment (small theta)
      IF (THEDGE .LT. 7.0*PI/2.0) GO TO 2020  !check for at least 1.5 loop
```

```
      THMID=THEDGE+DTHETA/2.0       !theta for centroid of current segment
      AXLSEG(ISEG)=SUML
      DO 3060 J=NTOTSUB,1,-1
 3080 IF (THMID .GT. THSUB(J) .OR. THMID .LT. THSUB(J-1)) GO TO 3050
      FRAC=(THMID-THSUB(J-1))/(THSUB(J)-THSUB(J-1))
      R(ISEG)=abs(FRAC*(RS(J)-RS(J-1))+RS(J-1))
      Z(ISEG)=FRAC*(ZS(J)-ZS(J-1))+ZS(J-1)
      THMID=THMID-DTHETA
 3050 IF (THEDGE .GT. THSUB(J) .OR. THEDGE .LT. THSUB(J-1)) GO TO 3060
      FRAC=(THEDGE-THSUB(J-1))/(THSUB(J)-THSUB(J-1))
      AXLSEG(ISEG-1)=FRAC*(AL(J)-AL(J-1))+AL(J-1)
      THEDGE=THEDGE-DTHETA
      IF (THEDGE .LT. ALFSPIR) GO TO 3070         !done with spiral part
      I=I+1
      ISEG=NSEG+1-I
      GO TO 3080
 3060 CONTINUE
 3070 CONTINUE     !I contains number of segments in the spiral region
      NSEGSPIR=I                !exact value
      NSEGFLAT=NSEG-NSEGSPIR
C
C  now find segments on flat part of sheet
C
C--get segment stretching parameters
C
      SUMI=0.0
      DO 3761 I=1,NSEGFLAT-1
 3761 SUMI=SUMI+I
      DSLAST=AXLSEG(ISEG)-AXLSEG(ISEG-1)
      HHH=(AXLSEG(ISEG-1)-NSEGFLAT*DSLAST)
     $                          /(SUMI-NSEGFLAT*(NSEGFLAT-1))
      DSFIX=DSLAST-HHH*(NSEGFLAT-1)
C
      AXLSEG(0)=0.0
      I=1
```

```
      DSI=DSFIX+HHH*(I-1)
      XLSEG=DSI/2.0              !arc length to filament
      EDGE=DSI                   !arc length to outboard edge of segment
      DO 3061 J=1,NTOTSUB
 3081 IF (XLSEG .LT. AL(J-1) .OR. XLSEG .GT. AL(J)) GO TO 3061
      FRAC=(XLSEG-AL(J-1))/(AL(J)-AL(J-1))
      R(I)=abs(FRAC*(RS(J)-RS(J-1))+RS(J-1))
      Z(I)=FRAC*(ZS(J)-ZS(J-1))+ZS(J-1)
      AXLSEG(I)=EDGE
      I=I+1
      DSI=DSFIX+HHH*(I-1)
      XLSEG=EDGE+DSI/2.0
      EDGE=EDGE+DSI
      IF (I .GT. NSEGFLAT) GO TO 3071
      GO TO 3081
 3061 CONTINUE
 3071 CONTINUE
C--set WPAN for all segments
      DO 3110 ISEG=1,NSEG
      WPAN(ISEG)=(AXLSEG(ISEG)-AXLSEG(ISEG-1))/2.0
 3110 CONTINUE
      R(NOUT1+1)=ROLD(NRING)
      Z(NOUT1+1)=ZOLD(NRING)
      DGAM(NOUT1+1)=DGAMOLD(NRING)
      NOUT=NOUT1
      NRING=NOUT1+1
      NSMALL=NRING-1
      GO TO 2030
C
 2030 CONTINUE
C
C--If panel case, estimate inclinations (only nec. for R-K)
C
      IF (IPANEL .EQ. 1) THEN
        DO 920 I=1,NSMALL-1
```

```
          SL=-SQRT((R(I)-R(I-1))**2+(Z(I)-Z(I-1))**2)
          SR= SQRT((R(I)-R(I+1))**2+(Z(I)-Z(I+1))**2)
          A1P=((SL*SR)/(SL-SR))
     $                *((R(I+1)-R(I))/SR**2-(R(I-1)-R(I))/SL**2)
          B1P=((SL*SR)/(SL-SR))
     $                *((Z(I+1)-Z(I))/SR**2-(Z(I-1)-Z(I))/SL**2)
          DRPAN(I)=A1P
          DZPAN(I)=B1P
  920   CONTINUE
        A2P=((R(I+1)-R(I))/SR-(R(I-1)-R(I))/SL)/(SR-SL)
        B2P=((Z(I+1)-Z(I))/SR-(Z(I-1)-Z(I))/SL)/(SR-SL)
        DRPAN(NSMALL)=A1P+A2P*SR
        DZPAN(NSMALL)=B1P+B2P*SR
      ENDIF
C
C--Get values of S at the midpoints of the original ring locations
C
      SMID(0)=0.0
      SMID(NIN)=SUML
      DO 90 I=1,NIN-1
   90 SMID(I)=(AL(NSUB*(I+1))+AL(NSUB*I))/2.0
C
C--Get values of circulation at the midpoints of the original
C   ring locations
C
      GMID(0)=GMAX
      DO 100 I=1,NIN
  100 GMID(I)=GMID(I-1)+DGAM(I)
C
C--Curve fit for circulation, test for linear or cubic fit
C
      IF (LINEAR) THEN
C
C--The following block is for a linear fit of circulation
C
```

```
        DO 411 I=0,NIN-1
        C0(I)=GMID(I)
        C1(I)=(GMID(I+1)-GMID(I))/(SMID(I+1)-SMID(I))
        C2(I)=0.0
  411   C3(I)=0.0
C
      ELSE
C
C--The following block is for a blended parabola fit of circulation
C
C--Get slopes at each interior point
C
        DO 110 I=1,NIN-1
        S1=(SMID(I+1)-SMID(I))/(SMID(I-1)-SMID(I))
  110   Q(I)=((GMID(I-1)-GMID(I))*S1-(GMID(I+1)-GMID(I))/S1)
     $       /(SMID(I+1)-SMID(I-1))
C
C--Coef's for interior points
C
        DO 120 I=1,NIN-2
        S1=1.0/(SMID(I+1)-SMID(I))
        G1=(GMID(I+1)-GMID(I))*S1
        C0(I)=GMID(I)
        C1(I)=Q(I)
        C2(I)=S1*(-Q(I+1)-2*Q(I)+3*G1)
  120   C3(I)=S1*S1*(Q(I+1)+Q(I)-2*G1)
C
C--Coef's for first segment
C
        S1=SMID(1)-SMID(0)
        S2=SMID(2)-SMID(0)
        S3=1.0/(SMID(2)-SMID(1))
        C0(0)=GMID(0)
        C1(0)=S3*((GMID(1)-GMID(0))*S2/S1-(GMID(2)-GMID(0))*S1/S2)
        C2(0)=S3*((GMID(2)-GMID(0))/S2-(GMID(1)-GMID(0))/S1)
```

```
        C3(0)=0.0
C
C--Coef's for last segment
C
        S1=SMID(NIN-2)-SMID(NIN-1)
        S2=SMID(NIN)-SMID(NIN-1)
        S3=1.0/(SMID(NIN-2)-SMID(NIN))
        G1=GMID(NIN-2)-GMID(NIN-1)
        G2=GMID(NIN)-GMID(NIN-1)
        CO(NIN-1)=GMID(NIN-1)
        C1(NIN-1)=S3*(G2*S1/S2-G1*S2/S1)
        C2(NIN-1)=S3*(G1/S1-G2/S2)
        C3(NIN-1)=0.0
C
      ENDIF
C
  651 CONTINUE
C
C--Interpolate for new delta circulation values.
C  XLSEG will now be an array containing the arc length to the
C  edge of segment I, call it AXLSEG(I)
C
      I=1
      GPREV=GMAX
      DO 130 J=1,NIN
  150 IF(AXLSEG(I) .LT. SMID(J-1) .OR. AXLSEG(I) .GT. SMID(J))GO TO 130
      T=AXLSEG(I)-SMID(J-1)
      GNEW=CO(J-1)+T*(C1(J-1)+T*(C2(J-1)+T*C3(J-1)))
      DGAM(I)=GNEW-GPREV
      GPREV=GNEW
      I=I+1
      IF (I .GT. NOUT-1) GO TO 140
      GO TO 150
  130 CONTINUE
  140 CONTINUE
```

```
      T=SMID(NIN)-SMID(NIN-1)
      GNEW=CO(NIN-1)+T*(C1(NIN-1)+T*(C2(NIN-1)+T*C3(NIN-1)))
      DGAM(NOUT)=GNEW-GPREV
C
C--Done
C
      RETURN
      END




       subroutine ritecirc
c
c   a subroutine to output final load distribution
C
       include '/user/huh/rollup/common.f'
       real circt(103),r(103)
       character*60 gtitle,title
       open(unit=21,file='initload.dat',status='old',form='formatted')
       do i = 1,103
          read(21,*)r(i),circt(i)
       enddo
       close(21)
       open(unit=8,file='initsheet.dat',form='formatted',status='new')
       gtitle='INIT-SHEET'
       title = 'RADIUS  VORT'
       write(8,*) gtitle
       write(8,*) 1
       write(8,*) 52
       write(8,*) title
       write(8,*) 1
       write(8,*) 0.,1.,-5.e-03,1.e-03
       write(8,*) 2
       write(8,*) nring
```

```
      do 65 i = 1,nring
         write(8,*) radius(i),gamma(i)
65       continue
      close(8)
      gtitle='MILLER LOADING'
      title='RADIUS  GAMMA'
      write(15,*)gtitle
      write(15,*)2
      write(15,*)52
      write(15,*)title
      write(15,*)1
      write(15,*)0.,1.,0.,3.e-02
       write(15,*)2
      write(15,*)103
       do 133 i = 1,103
         write(15,*)r(i),circt(i)
133       continue
      write(15,*)4
      write(15,*)28
      do i = 1,28
         write(15,*)expx(i),expy(i)
      enddo
      return
      end




      subroutine riteinput
c
c   a subroutine to write out initial near wake values
C
CVD$R ASSOC
      include '/user/huh/rollup/common.f'
       WRITE(IWR,*)'    '
```

```
       WRITE(IWR,*)'   '
       WRITE(IWR,*)'   '
      write(iwr,*)'TIME = ',timenow
       WRITE(IWR,*)'   '
       WRITE(IWR,*)' -------------- INITIAL VALUES ---------------'
       WRITE(IWR,*)'   '
       WRITE(IWR,3243)
3243   FORMAT('   I      RADIUS       HEIGHT       GAMMA     ',
    $     '       CORE       UVEL        WVEL           AXLSEG    ')
       WRITE(IWR,3244)(I,RADIUS(I),HEIGHT(I),GAMMA(I),i=0,nring)
3244   FORMAT(1X,I3,3x,e14.6,3x,e14.6,3x,e14.6)
      sum = 0.
      do i = 1,nring
      sum = sum+gamma(i)
      enddo
      write(iwr,*) 'total circulation = ',sum
      return
      end




      subroutine riterst(tint,itrs)
c
c   a subroutine to write out values for future restart runs
C
      include '/user/huh/rollup/common.f'
      open(unit=16,file='restart.dat',form='unformatted',status='new')
      write(16) nring,lineup,minsect,maxsegs
      write(16) timenow,itrs
      write(16) tint
      write(16) level,nfar
      write(16) ninter(1:level)
      do i = 1,3
         do j = 1,4
```

```
            write(16) rinter(i,j)
            write(16) zinter(i,j)
            write(16) ginter(i,j)
            write(16) cinter(i,j)
         enddo
      enddo
      write(16)rfar(1:nfar)
      write(16)zfar(1:nfar)
      write(16)gfar(1:nfar)
      write(16)aslope,sigma
      write(16)nspan
      write(16)span(1:nspan)
      write(16)pitch(1:nspan)
      write(16)chord(1:nspan)
      write(16)circ(1:nspan)
      write(16)radius(0:nring)
      write(16)height(0:nring)
      write(16)gamma(0:nring)
      write(16)core(0:nring)
      write(16)axlseg(0:nsmall)
      close(16)
      return
      end



      SUBROUTINE rke(ENERG,istart,iend)
C
C--Computes total energy of the vortex ring system
C
       include '/user/huh/rollup/common.f'
CVD$R ASSOC
      REAL*4 LAMBDA,K
C
```

```
      SUM1=0.0
      SUM2=0.0
       if(istart .eq. 1)goto40
      DO 200 I=1,istart-1
        W=RADIUS(I)
        Z=HEIGHT(I)
        DO 20 J=1,istart-1
           if(i .eq. j)goto20
            WP=RADIUS(J)
            ZP=HEIGHT(J)
            R1=SQRT((Z-ZP)**2+(W-WP)**2)
            R2=SQRT((Z-ZP)**2+(W+WP)**2)
            LAMBDA=(R2-R1)/(R2+R1)
            CALL ELLIP(LAMBDA,K,E)
            SUM1=SUM1+GAMMA(I)*GAMMA(J)*(R1+R2)*(K-E)/(2.0*PI)
 20    CONTINUE
       DO 200 J=iend+1,nring
            WP=RADIUS(J)
            ZP=HEIGHT(J)
            R1=SQRT((Z-ZP)**2+(W-WP)**2)
            R2=SQRT((Z-ZP)**2+(W+WP)**2)
            LAMBDA=(R2-R1)/(R2+R1)
            CALL ELLIP(LAMBDA,K,E)
            SUM1=SUM1+GAMMA(I)*GAMMA(J)*(R1+R2)*(K-E)/(2.0*PI)
 200   CONTINUE
 40      continue
       DO 300 I=iend+1,nring
       W=RADIUS(I)
       Z=HEIGHT(I)
      if(istart .eq. 1)goto60
       DO 70 J=1,istart-1
            WP=RADIUS(J)
            ZP=HEIGHT(J)
            R1=SQRT((Z-ZP)**2+(W-WP)**2)
            R2=SQRT((Z-ZP)**2+(W+WP)**2)
```

```
            LAMBDA=(R2-R1)/(R2+R1)
            CALL ELLIP(LAMBDA,K,E)
            SUM1=SUM1+GAMMA(I)*GAMMA(J)*(R1+R2)*(K-E)/(2.0*PI)
   70    CONTINUE
 60        continue
         DO 80 J=iend+1,nring
            if(i .eq. j)goto 80
            WP=RADIUS(J)
            ZP=HEIGHT(J)
            R1=SQRT((Z-ZP)**2+(W-WP)**2)
            R2=SQRT((Z-ZP)**2+(W+WP)**2)
            LAMBDA=(R2-R1)/(R2+R1)
            CALL ELLIP(LAMBDA,K,E)
            SUM1=SUM1+GAMMA(I)*GAMMA(J)*(R1+R2)*(K-E)/(2.0*PI)
  80     CONTINUE
 300        continue
      SUM1=SUM1*PI
C
       if(istart .eq. 1)goto 800
      DO 700 I=1,istart-1
        W=RADIUS(I)
 700   SUM2=SUM2+GAMMA(I)**2*W*(ALOG(8.0*W/CORE(I))-1.75)/(2.0*PI)
 800        continue
      DO 900 I=iend+1,nring
        W=RADIUS(I)
 900   SUM2=SUM2+GAMMA(I)**2*W*(ALOG(8.0*W/CORE(I))-1.75)/(2.0*PI)
      SUM2=SUM2*PI
C
      ENERG=SUM2+SUM1
C
      RETURN
      END
```

```
      SUBROUTINE SETA
       include '/user/huh/rollup/common.f'
      COMMON /E2/ AO,A1,A2,A3,RHOSTAR,A
      COMMON /E1/ TAUO,BDUM,CDUM,DDUM,TAU2LOOP
C
      ALPLAST=PI/2.0
      TAUO=0.045
      BIGRREF=0.0966063
      TAU2LOOP=1.4542
      A=2.0*SQRT(2.0)/PI
C
      RETURN
      END




      SUBROUTINE TOTENG(ENERG)
C
C--Computes total energy of the vortex ring system
C
       include '/user/huh/rollup/common.f'
CVD$R ASSOC
      REAL*4 LAMBDA,K
C
      SUM1=0.0
      SUM2=0.0
      DO 20 I=1,NRING
        W=RADIUS(I)
        Z=HEIGHT(I)
        DO 20 J=1,NRING
          IF (I .NE. J) THEN
            WP=RADIUS(J)
            ZP=HEIGHT(J)
```

```
            R1=SQRT((Z-ZP)**2+(W-WP)**2)
            R2=SQRT((Z-ZP)**2+(W+WP)**2)
            LAMBDA=(R2-R1)/(R2+R1)
            CALL ELLIP(LAMBDA,K,E)
            SUM1=SUM1+GAMMA(I)*GAMMA(J)*(R1+R2)*(K-E)/(2.0*PI)
          ENDIF
   20   CONTINUE
      SUM1=SUM1*PI
C
      DO 30 I=1,NRING
        W=RADIUS(I)
      SUM2=SUM2+GAMMA(I)**2*W*(ALOG(8.0*W/CORE(I))-1.75)/(2.0*PI)
 30       continue
      SUM2=SUM2*PI
C
      ENERG=SUM2+SUM1
C
      RETURN
      END




      SUBROUTINE UPDATE(H,N)
       include '/user/huh/rollup/common.f'
C
C--Computes new positions of the rings based on the time step H and
C  the velocity contained in RK and HK.
C
      DO 10 I=0,NRING
        RADIUS(I)=RADO(I)+H*RK(I,N)
        HEIGHT(I)=HEIO(I)+H*HK(I,N)
   10 CONTINUE
       do 15 j = 1,level
          do 15 i = 1,ninter(j)
```

```
              rinter(i,j) = rint0(i,j) + h*rki(i,j,n)
              zinter(i,j) = zint0(i,j) + h*hki(i,j,n)
 15        continue
C
C--Compute the new core radius for the tip ring
C
      CORE(NRING)=SQRT(VOLTIP/(2.0*RADIUS(NRING)))/PI
C
C--Compute the new core radius for intermediate wake
C
       do 17 j = 1,level
          do 17 i = 1,ninter(j)
               Cinter(i,j)=SQRT(VOLinter(i,j)/(2.0*Rinter(i,j)))/PI
 17        continue
C
      RETURN
      END
      FUNCTION XNU(THETA)
      DATA PI/3.141592654/
      XNU=2.0/3.0
      IF (THETA .LT. 2.0*PI) THEN
        XNU=(1.0/(2.0*PI-1.0))-(1.0/(THETA-1.0))+2.0/3.0
                             ENDIF
      RETURN
      END
```