Continuous Curved Near Wake Analysis for a Lifting Surface

by

Melinda Dee Godwin

B.S., Aeronautical Engineering, University of Maryland (1989)

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

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1991

Signature of Author _____
Department of Aeronautics and Astronautics
July 22, 1991

Certified by _____
Professor Mark Drela
Thesis Supervisor

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

CONTINUOUS CURVED NEAR WAKE ANALYSIS FOR A LIFTING SURFACE

BY

MELINDA DEE GODWIN

Submitted to the department of Aeronautics and Astronautics on July 22,1991, in partial fulfillment of the requirements for the Degree of Master of Science in Aeronautical Engineering.

Abstract

Accurate modeling of the near wake remains a significant problem in helicopter rotor aerodynamics. Motivated by the inability of present models to achieve close agreement with experimental near-wake geometries, an improved representation of the near-wake of a one-bladed rotor is developed. Although this problem has been addressed by others, close agreement with experimental wake geometries has not been achieved.

A major problem in matching experimental results using filamentary models involves specification of the proper vortex core size for a curved point vortex. This arbitrary core size is needed to eliminate the logarithmic induced velocity singularity and solutions have been shown to vary significantly depending on the chosen value. The present study uses a continuous vortex sheet rather than discrete filaments, avoiding the need for a finite core size. The model represents a one-bladed hovering rotor trailing a continuous vortex sheet. A free-wake analysis using the Biot-Savart law is then applied to determine the wake geometry in a fixed plane perpendicular to the initial position of the blade. The singularity associated with the circulation distribution at the tip is represented using a self-similar solution developed by Pullin. Several circulation distributions are studied, including an elliptical distribution and a distribution typical of a hovering rotor. For the latter, the intent is to study the character of the vortex roll-up and to determine if, counter to experimental evidence, a mid-span vortex occurs as it has using past models.

Results from the current study may be used in conjunction with discrete models to predict the proper core size to be used. In this manner, the greater computational efficiency of discrete models may be retained while obtaining the accuracy of a more rigorous near-wake

representation. This research could also be extended to forward flight and incorporated in an existing full-wake analysis code as the near-wake component.

Table of Contents

Acknowledgment

My sincerest thanks to my advisor, Mark Drela, for all the time and effort he spent assisting me in developing this thesis. His wisdom and patience was greatly appreciated.

I would also like to thank the NSF PYI Program, along with Earl Murman and NASA Langley (NASA Grant NAG-1-507) for making this research and my masters possible.

Finally, I would like to thank my husband Mark for his support, assistance, and understanding through the final stages of this thesis.

List of Figures

Nomenclature

d h	segment of integration strip which produces a velocity at a point in space.
l	radius vector from dh to a point in space
r	radial displacement corresponding to s
R	radial displacement of marker corresponding to s
s	location where circular integration strips are located
S	location along vortex sheet where induced velocity is calculated
z	vertical displacement of wake corresponding to s
Z	vertical displacement of marker corresponding to s
2δ	width of linear region
Ψ	rotor azimuthal angle, radians
Γ_b	bound circulation

1. Introduction

1.1 Historical Note

Modeling the aerodynamics of a rotor has always been a difficult task due to the geometric and aerodynamic complexity of the wake. In the early 1900's, Kutta and Joukowsky related lift to vorticity in the flow, providing the fundamental basis for vortex modeling of rotor wakes. This approach utilizes the Biot-Savart law which allows determination of the induced velocity at any position in the wake and modeling of the wake deformation and displacement in unsteady flows.

1.2 Previous Rotor Wake Research

Many different analyses and models evolved using vortex modeling, including the vortex sheet and filamentary models discussed below. The most recent development in vortex modeling has been free-wake analysis, defining the complete wake geometry using the calculated induced velocities. Free-wake analyses are physically more correct than previous models and, for that reason, many recent studies have been focusing on utilizing and optimizing this method of analysis.

Motivated by the inability of existing models to accurately predict near-wake geometry, this thesis focuses on an improved representation of the near-wake of a one-bladed rotor (the near wake is defined to extend from $\Psi = 0°$ to $\Psi = 45°$). Although there are a variety of near wake analysis programs using filamentary and vortex sheet models, the need to specify an arbitrary core size limits their general applicability. An example is an analysis by Miller[1] where the near wake is modeled

as a series of semi-infinite vortex elements which are then rolled up into two or three discrete vortices (before the first blade is encountered) according to Betz criteria.[2,3] Since the mid-span and root vortices which result have not been detected in any of the relevant experimental studies, Miller states that an "exact prediction of the wake structure" can not be determined from the results, and the roll-up technique should only be viewed as a "convenient computational technique to account for the effects of the inboard trailing vorticity".[4]

A vortex filament model developed by Brower[5] considers the entire rotor wake. Brower represents the near wake as straight vortex segments and includes a correction for viscous core size (causing the induced velocity to approach zero rather than infinity as the vortex element is approached). Since Brower uses straight filaments, he adds a correction factor for self-induced effects, since actual vortex filaments leaving the blade are curved and will induce a velocity on themselves. Brower found that the choice of core size had a significant effect on the induced velocities at nearby points, and that the choice of core size should be related to the computational mesh size. Most papers written on filamentary models note difficulties in selecting a proper core size due to the lack of a physically correct predictive model and an incomplete understanding of how core size is related to specific flows.

Vortex sheet models for near wake analysis using a free-wake approach have also been developed. An example is the model developed by Tanuwidjaja.[6] The near-wake is modeled as a vortex sheet with tip and root vortices. This method uses the Biot-Savart law and divides the blade and wake into nodes which define discrete segments, and sheet elements, which form the vortex sheet. The

induced velocities, however, are not calculated at these nodes, due to singularity problems at the discrete sheet elements. Instead, new locations between the nodes called "centers" are defined. The induced velocity is then calculated at these centers and linear interpolation is used to find the velocities at the nodes. Since, however, Tanuwidjaja uses discrete vortex sheet elements, the arbitrariness that stems from the need to specify a finite core size is not eliminated.

1.3 Outline of the Present Research

In this study, unlike the iterative analyses by Miller, the model is given a circulation distribution that is assumed correct, and it then calculates the induced velocities and resulting wake geometry by integrating across the wake using the Biot-Savart law. The induced velocities in the wake are calculated at sequential increments in time, so "snap-shots" of the resulting wake geometry are obtained downstream of the rotor. This process is repeated until an azimuthal angle of $\Psi = 45$ degrees is reached. This is a continuous model, therefore *no* finite core size (except for the tip vortex) is needed, which eliminates the arbitrariness in the solution due to core size specification.

As previously stated, the present research considers the near wake of a one bladed rotor using a free-wake analysis. The entire wake geometry is not considered, since the intermediate and far wakes are comparatively well understood (although computationally time consuming to analyze)[7]. Rather it is the intent to focus on a more physically correct representation of the near wake, yielding results which more closely match experimental data. By obtaining a more

accurate near wake geometry, this program may be utilized in the following ways.

1. To assist in selecting the appropriate core size to be used in filamentary models where core size is included as a parameter.
2. To test other wake geometry programs to investigate the accuracy of their solutions.
3. As the near-wake component in a full-wake analysis code.

Miller states, "It will never be possible to model exactly the dynamics and aerodynamics of a system as complex as a rotary wing vehicle. ....To this end the simplest forms of analysis, which retain the essence of the phenomena involved, are of greatest value during the crucial stages of design and flight evaluation."[8] The difficulty lies in knowing how to simplify a model while maintaining an acceptable level of accuracy. While the model outlined in this thesis may be physically more correct, it may not be practical to efficiently implement in free-wake codes modeling the entire wake. The model is more computationally time-consuming and mathematically difficult than models such as Miller's fast free wake analysis. The best alternative would be to use the knowledge gained from this research to improve the results of more simplistic models, thereby achieving good results while minimizing computational expense.

2. Description of the Model

The present axisymmetric model represents the near wake of a one bladed rotor as a vortex sheet rather than by a series of discrete

filaments. This sheet is composed of 500 integration intervals which contain the vorticity in the wake. In reality these integration strips follow helical paths downstream of the rotor. Miller has shown, however, that the effect of the inclination of these integration intervals on the induced velocity is negligible.[9] This model is therefore developed using integration intervals which are planar circles perpendicular to, and centered about the z-axis (see figure 3.1-1). This results in an axisymmetric flow pattern where the axial, radial, and tangential velocity components are independent of Ψ. Since the circular integration intervals remain perpendicular to the axis of symmetry as the wake convects downstream, the tangential component of the velocity will always be negligible. The purpose of this model is to investigate how the wake deforms downstream of the rotor blade, and to develop insight into the question of whether or not a discrete mid-span vortex is formed. First, a description of the coordinate system is provided, followed by an explanation of the primary non-dimensional parameters. The "linear region" concept used to eliminate the computational singularity in the induced velocity equations is then described. Next, the exponential stretching technique used to define integration strips (containing the vorticity in the wake) is discussed along with the similarity solution developed by Pullin to handle the tip singularity. Finally, a description of the vortex sheet will be given.

2.1 Coordinate System

The coordinate system for the wake geometry used in this analysis is three-dimensional, consisting of radial (r), azimuthal (Ψ) and vertical (z) components, as shown in figure 2.1-1.

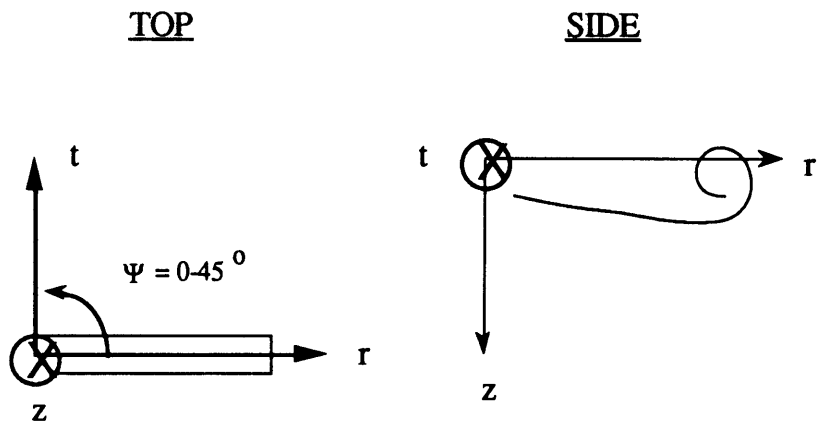


Figure 2.1-1 Basic Coordinate System

The actual wake deformation is viewed in two dimensions in the r-z plane (as shown in the side view above), producing two-dimensional, time dependent wake profiles. To designate locations along the profile, a sheet coordinate, "s", is introduced.

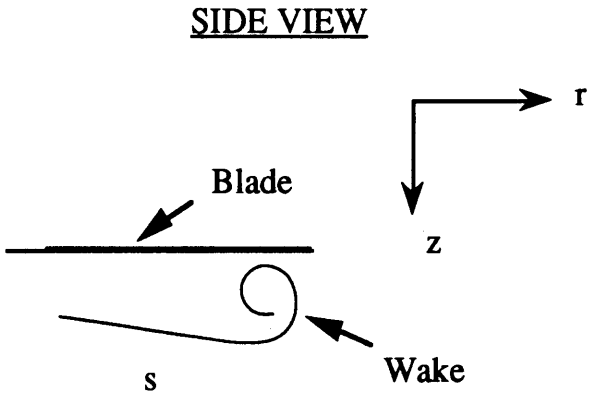


Figure 2.1-2 Definition of s Coordinate

To simplify the form of the equations, it is assumed that the two-dimensional wake is always viewed from a fixed position (Ψ=0°) with

the blade moving away at a positive angle. This approach eliminates additional angular velocity terms.

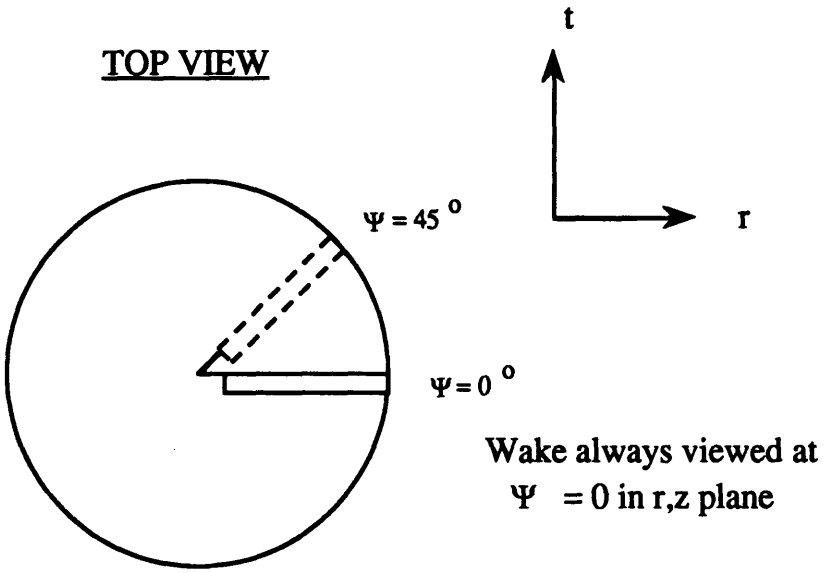


Figure 2.1-3 Top View of Rotor

2.2 Non-Dimensional Parameters

All of the parameters used in the equations are non-dimensionalized to insure flexibility in running test cases and generalizing results.

Length

The sheet geometry is parameterized in terms of its arc length s, non-dimensionalized by the blade radius. The sheet geometry is

therefore defined by the coordinates r(s) and z(s), which are also non-dimensionalized by the blade radius.

In the numerical scheme, the marker locations on the sheet where the velocity is calculated are denoted by uppercase symbols S, R, and Z, while the locations of the integration strips containing the vorticity are denoted by lowercase s, r, and z.

$$\mathbf{R} = \frac{\text{marker r-location}}{\text{blade radius}}, \mathbf{Z} = \frac{\text{marker z-location}}{\text{blade radius}}$$

Angle

The angle Ψ refers to the azimuthal angle (in radians) about the axis of rotation extending from $-\pi$ to π. This angle is shown in Figure 2.1-3.

Time

Time is non-dimensionalized by Ω, the rotational speed of the rotor (i.e. $t^* = t\,\Omega$). As an example, an azimuthal angle of $\pi/4$ radians corresponds to a non-dimensional time $t^*=\pi/4$, or more generally, $\Delta t^*=\Delta\Psi$. The near wake is broken into 45 increments, and therefore the outer time loop in the program moves through 45 one-degree time steps.

Velocity

The non-dimensional velocity is simply the ratio of the non-dimensional length and time described above:

$$V^* = l^*/t^* = \frac{l}{(\text{blade radius})(t)(\Omega)} = \frac{V}{(\Omega)(\text{blade radius})}$$

2.3 Definition of Linear Region

The linear region is a specified area on either side of a marker within which the equations are modified to analytically remove the singularity that occurs when an integration interval approaches a marker location.

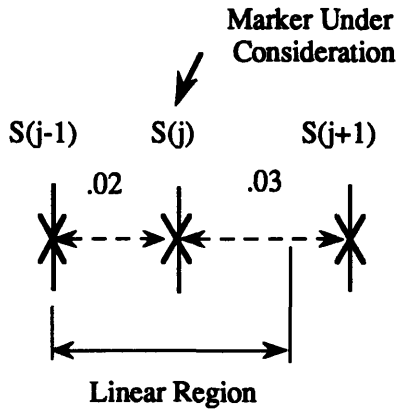
It is necessary to integrate equations 3.14 and 3.30 over the sheet length s, since the r and z terms are both functions of s. To facilitate this integration, r and z terms are linearized with respect to s. This is accomplished through the following relations:

$$R\text{-}r = K_1(S\text{-}s)$$
$$Z\text{-}z = K_2(S\text{-}s)$$

$$\text{where} \quad K_1 = \frac{dr}{ds}\Big|_{s=S} \qquad K_2 = \frac{dr}{ds}\Big|_{s=S}$$

These linear relations are then used to eliminate r and z in the equation, resulting in an equation dependent only on s. This allows the singularities in the integral to be analytically removed. Since these equations (i.e. 3.14, 3.30) can only be defined over small portions of the wake (because of the linearizations above), the modified form of the equation is only used when an integration interval is in close proximity to a marker (in the linear region) and the unmodified original equations are used elsewhere. The size of the linear region is a variable, and it

will be shown that the wake geometry is independent of the precise value chosen as long as the linearity assumption for r and s, and z and s remains valid. For the test cases shown, the lesser of the two distances from the marker being considered to the markers on either side is chosen as δ and the linear region is twice this distance (half on each side of the marker as shown in figure 2.3-1).



$$\delta = \text{MIN}(0.02,\ 0.03) = 0.02$$

$$S_{low} = S(j) - \delta$$

$$S_{high} = S(j) + \delta$$

$$\text{width of linear region} = 2\delta$$

Figure 2.3-1 Determination of Linear Region Size

In summary, the linear region represents a specified area about each marker within which the governing equations are modified to eliminate the singularity which would otherwise occur as an integration interval approaches a marker. The only difference between the

modified and unmodified equations is the elimination of higher order terms.

2.4 Exponential Stretching Technique

Layout of Markers

An exponential stretching technique is used to construct the initial lay-out of the markers (i.e. the locations where induced velocities are calculated). The stretching technique requires as input the desired number of markers and the desired distance between the first two and the last two markers. The routine then exponentially stretches between the second and the second to last markers. In general, a large distance between the first two points is specified and a small distance between the last two points. This results in a low population of markers near the root and a high density layout at the tip where, due to the tip vortex, the curvature of the wake is greatest (150 markers are generally used).

The vortex core, which is used as a filament *and* a marker, is not located through this technique, but through a method outlined in section 2.5.

Layout of Integration Strips Containing the Wake Circulation

The layout of the integration strips, unlike the layout for the markers, is modified at *each* time step and for *each* marker. At each time step, the sheet is broken into integration intervals to account for the vorticity in the wake. These intervals need to be denser near the tip (due to the strong vorticity and roll-up) and near the marker under consideration (due to the possibility of a singularity point being approached). To obtain a good distribution, a separate filament

distribution is established for each marker as it is considered. This layout is accomplished by utilizing an exponential stretching routine twice, once on each side of the marker. An example is pictured and described below.

x = marker

• = integration intervals

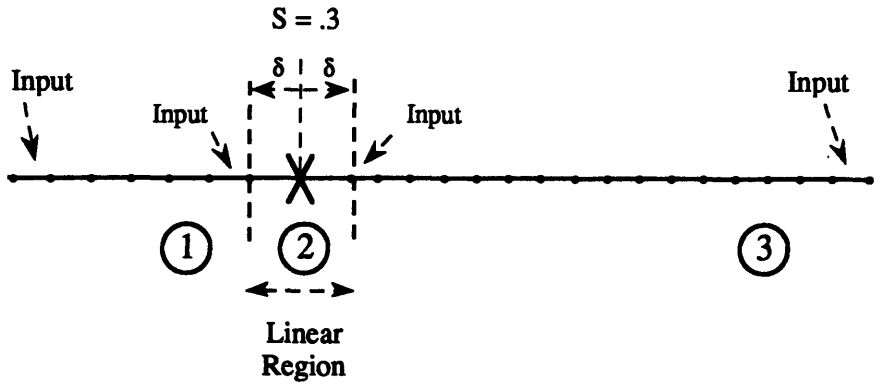


Figure 2.4-1 Filament Distribution

The lay-out of the filaments in region 1 are dictated by specifying the spacing between the first two and last two integration intervals and by specifying the number of intervals to be placed in this region (the filament lay-out in region 3 is obtained in an identical fashion). In the "linear region" (i.e. region 2), the layout of the filaments is slightly different. In this region, 6 points are always evenly distributed, keeping the number of integration strips constant in this area. This method assures that the number of strips within the linear region remains constant, or increases or decreases monotonically (see step 4 on the following page). If the number of integration intervals is allowed to

randomly fluctuate, instabilities occur in the wake due to truncation errors. These truncation errors arise due to the loss of higher-order terms, which occurs because the equations used in the linear region have been modified to handle the singularities which arise in this area. The lay-out of the filaments can be summarized as follows:

1. The linear region variable (δ) is added and subtracted from each side of the marker.

2. The area from the root to the left end of the linear region is stretched, and integration strips are located.

3. The area from the right end of the linear region to the tip is stretched and integration strips are located.

4. Six integration strips are located in the linear region, equal distances apart. At the root and the tip, where the linear region is only one-sided, 4 integration strips are used.

5. The midpoints between all these locations are then taken, starting from the root of the blade. These midpoints define the locations of the integration intervals.

6. This 5 step process is then repeated for *each* marker, at *every* time iteration.

2.5 Pullin Similarity Solution

Due to the singular roll-up at the outer edge of the sheet, a discrete tip vortex forms instantaneously. Applying the Biot-Savart law directly without accounting for this singularity leads to erroneous results. One way to resolve this singularity is to use the self-similar solution developed by Pullin.[10,11] Pullin considers a semi-infinite vortex sheet with $\Gamma=2a|R|^{1/2}$ (where R is the location along the blade, and "a" is a scaling parameter), and develops a self-similar solution which produces an initial wake geometry and circulation distribution past a certain matching point (in this case r/R=0.95) at time $\tau=0+$. Pullin derives a similarity solution where:

$$Z_0(\Gamma,\tau) = (at)^{2/3}\,\zeta(\lambda) \qquad (1)$$

and

$$\lambda = \frac{\Gamma}{(a^{4/3}\,t^{1/3})} \qquad (2)$$

ζ, λ: similarity parameters

a, t: scaling parameters

Z_0: dimensional self-similar shape function for the sheet.

Using Pullin's solution along with initial values for ζ and λ (obtained by choosing a matching point along the vortex sheet), the above two equations can be solved for t and a (the scaling parameters), yielding the initial circulation distribution and wake geometry beyond r=0.95 for

$\tau = 0+$. An example is given below for the circulation distribution $\Gamma(r) = .02\sqrt{1-r^2}$, where $\zeta_0 = -3.24\mathbf{R}$ and $\lambda_0=3.6$, and s=0.95 where s is the location along the sheet defined as the matching point.

(1) Substitute Γ and Z into (1) and (2):

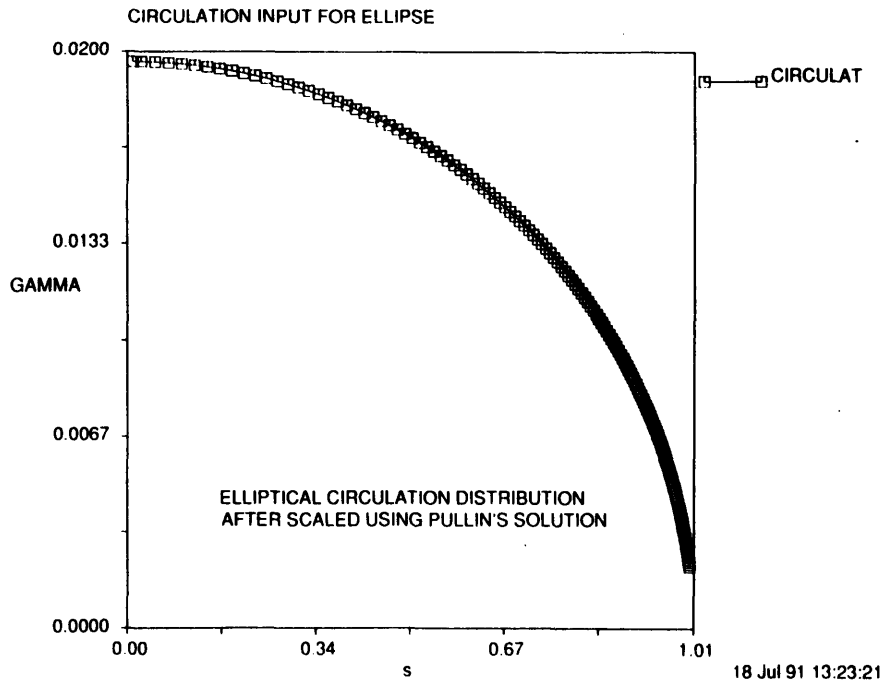
$$t = \frac{Z_0^{3/2}}{\zeta^{3/2}\ a} = \frac{[(1-s)]^{3/2}}{(3.24)^{3/2}\ a} \qquad (1a)$$

Note: (1-s) is necessary because Pullin has $\varsigma=0$ at the tip, and the coordinate system in this thesis is opposite.

$$t = \frac{\Gamma^3}{(\lambda^3\ a^4)} = \frac{\left(.02\ \sqrt{1-(1-s)^2}\right)^3}{(3.6)^3\ a^4} \qquad (1b)$$

(2) Solve(1a) and (1b) for t and a. In this case t = 0.1373 and a = 0.01396.

(3) Using t and a, the wake geometry and circulation distribution are defined beyond 95% blade radius by scaling Pullin's solution accordingly (see figure 2.5-1).



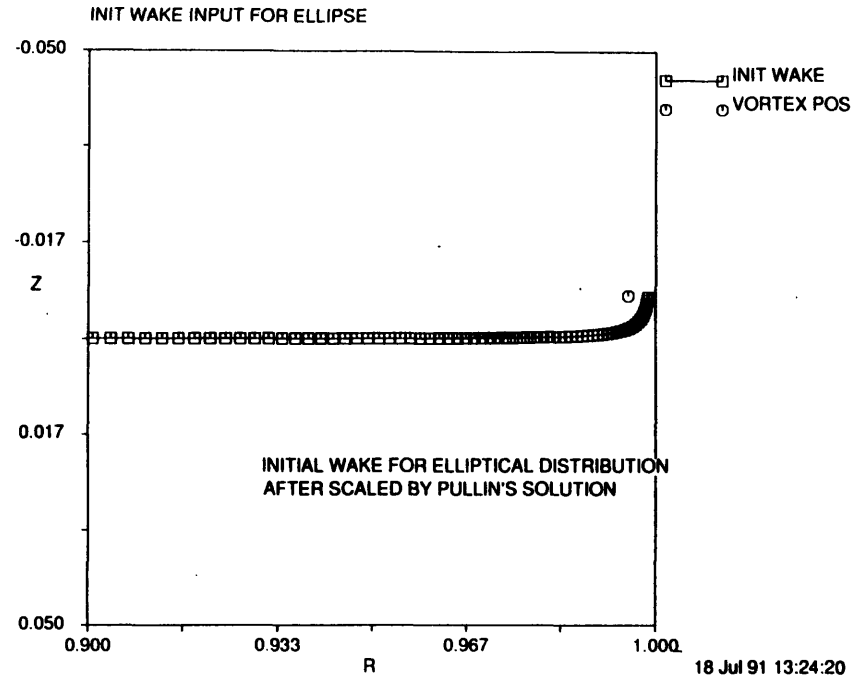


Figure 2.5-1 Pullin's Solution Applied to an Elliptically Loaded Blade

(4) Determine the tip strength and location. Pullin gives the total value of λ for the four spirals and the core as 1.342. By dumping this circulation into the core and re-locating the core (although in this case the amount the core is moved is negligible), the core strength is found to be λ=1.342 and its location is $\zeta = -.308e_r$, $\eta = .498e_z$, whose values can be scaled to Γ, z and r values.

One problem encountered when using this solution is in step 3 where Pullin's solution is scaled for the appropriate test case. Since Pullin's wake and circulation distribution were not available in tabular form, several values were estimated from the plots in references 10 and 11 and then splined. Due to unavoidable errors in transcribing this data, however, small perturbations would amplify and the wake would rapidly go unstable. To resolve this problem, Pullin's solution is applied to a specific case study (in this instance the ellipse), and then the resulting wake and circulation distributions are fitted with equations. By using the data from these equations (which contains the Pullin solution), the singularity due to $d\Gamma/dr$ approaching infinity at the blade tip is eliminated and an initial wake geometry and circulation distribution are defined.

2.6 Vortex Sheet Model

The basic wake model consists of a continuous vortex sheet with a discrete tip vortex. The wake is defined by two independent parameters, R and Z, where markers are placed. These markers are also the locations where the induced velocities are calculated, and are

therefore the points tracked downstream of the rotor blade which define the wake evolution in time. The number of markers used to represent the wake is a variable. Two other important parameters used throughout the analysis, r and z, are variables of integration, and represent the location of the integration strips containing the vorticity in the wake. The remaining two sheet geometry parameters are S and s, both referring to the sheet coordinate. The parameter S is the actual location of the markers along the sheet, while s is the location of the integration strips along the sheet. The number of integration strips is also a variable. These integration strips are modeled as circles from $\Psi=-\pi$ to $\Psi=\pi$ as shown in figure 3.1-1. Relating these six parameters:

$$\Delta S^2 = \Delta R^2 + \Delta Z^2$$
$$\Delta s^2 = \Delta r^2 + \Delta z^2$$

3. Analytical Method

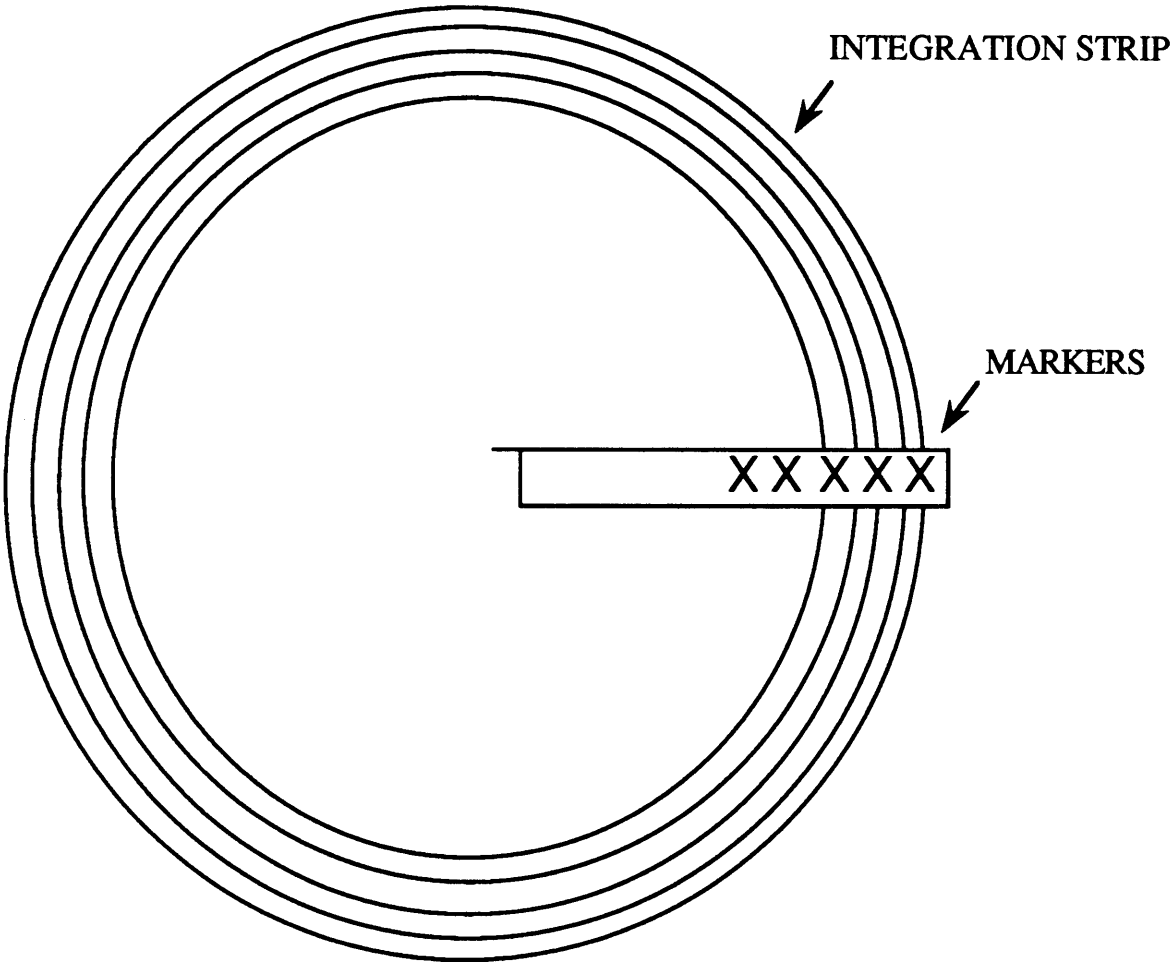
3.1 Application of the Biot-Savart Law

The two-dimensional near-wake geometry is obtained using a free-wake analysis technique. Using this approach, the wake assumes an equilibrium (or "force-free") position by moving according to the induced velocities at each marker location. These velocities are obtained by applying the Biot-Savart law at each marker and integrating along the sheet.

The wake is composed of circular integration intervals containing the circulation in the wake. These intervals are of infinitesimal strength and create a continuous vortex sheet. They are termed *integration*

intervals rather than *vortex filaments*, since they do not require a finite core size, as is the case with conventional filamentary models. In this continuous sheet model, the coordinate s is defined positive outward along the sheet. The location of the circular integration strips along the sheet are defined by lower case s, while the locations of the markers, where the induced velocities are calculated, are defined by S (see figure 3.1-1).

TOP VIEW



SIDE VIEW

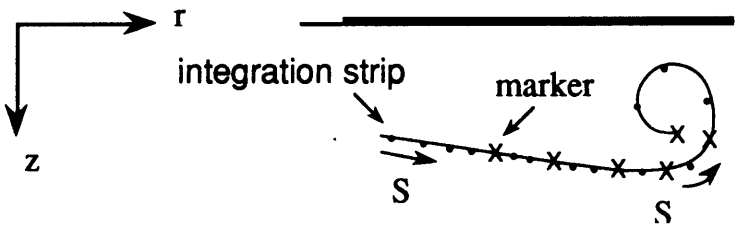


Figure 3.1-1 Problem Set-Up

To calculate the induced velocity at any point in the wake due to a filament of strength Γ, the Biot-Savart law can be applied as follows:[1,2]

$$V_i = \int \frac{\Gamma}{4\pi} \left\{ \frac{d\mathbf{h} \times \boldsymbol{l}}{|\boldsymbol{l}|^3} \right\} d\Psi \qquad (3.1)$$

Γ: strength of the vortex filament.

$d\mathbf{h}$: segment of the filament which induces a velocity at a point in space.

$\boldsymbol{l}$: radius vector from $d\mathbf{h}$ to any point in space.

Application of the Biot-Savart Law

For a vortex sheet composed of circular integration intervals initially in the plane of the rotor, this equation becomes (see figure 3.1-2):

$$V_i = \int \int \frac{d\Gamma}{ds} \frac{1}{4\pi} \frac{d\mathbf{h} \times \boldsymbol{l}}{|\boldsymbol{l}|^3} d\Psi \, ds \qquad (3.2)$$

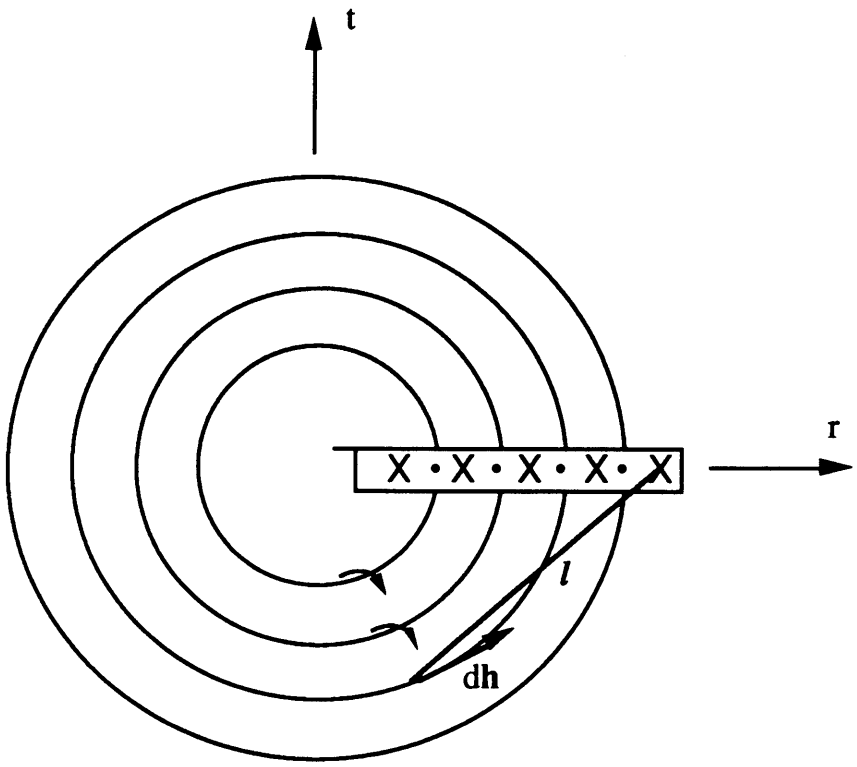


Figure 3.1-2 Application of Biot-Savart law

Now referring to figures 3.1-1 and 3.1-2 and realizing there is an additional component of the l vector in the z-plane at all times after $\tau=0$, it is clear that:

$$\boldsymbol{l} = \{R-r\cos\Psi\}\mathbf{r} - \{r\sin\Psi\}\mathbf{t} + \{Z-z\}\mathbf{z} \qquad (3.3)$$

Since the dh vector in equation 3.2 is tangent to the circular integration strips (which remain parallel to the rotor plane at all times downstream), it is only composed of components in the **r-t** plane. This vector can be represented as:

$$\mathbf{dh} = (r\sin\Psi d\Psi)\mathbf{r} - (r\cos\Psi d\Psi)\mathbf{t} \qquad (3.4)$$

Therefore, after computing the cross-product $d\mathbf{h} \times \boldsymbol{l}$ and substituting back into the Biot-Savart law (equation 3.2) the following equations are obtained:

$$V_a = \text{axial induced velocity} = \frac{1}{4\pi} \int_0^{s_{max}} \frac{d\Gamma}{ds} r \int_{-\pi}^{\pi} \frac{R\cos\Psi - r}{|l|^3} d\Psi\, ds \qquad (3.5)$$

$$V_r = \text{radial induced velocity} = -\frac{1}{4\pi} \int_0^{s_{max}} \frac{d\Gamma}{ds} \int_{-\pi}^{\pi} r \frac{\cos\Psi(Z-z)}{|l|^3} d\Psi\, ds \qquad (3.6)$$

$$V_t = \text{tangential induced velocity} = -\frac{1}{4\pi} \int_0^{s_{max}} \frac{d\Gamma}{ds} \int_{-\pi}^{\pi} r \frac{\sin\Psi(Z-z)}{|l|^3} d\Psi\, ds \qquad (3.7)$$

These equations represent the radial, axial, and tangential induced velocities calculated at each marker location.

3.2 Axial Velocity Component

As shown above, the equation derived for the axial velocity using the Biot-Savart law is:

$$4\pi V_a = \int \frac{d\Gamma}{ds} r \int \frac{R\cos(\Psi) - r}{|l|^3} d\Psi\, ds \qquad (3.8)$$

where:

$$\ell^3 = \left\{ R^2 - 2Rr\cos\Psi + r^2 + (Z-z)^2 \right\}^{3/2} \tag{3.9}$$

Now defining F as:

$$F = \frac{R\cos(\Psi) - r}{|\ell|^3} \tag{3.10}$$

and substituting $|\ell|^3$ into this expression, F becomes:

$$F = \frac{R\cos\Psi - r}{\left[R^2 - 2Rr\cos\Psi + r^2 + (Z-z)^2\right]^{3/2}} \tag{3.11}$$

It is apparent that a numerical singularity is encountered when $\cos\Psi$ approaches 1, and r approaches R (when r=R, z=Z). To resolve this singularity, $\cos\Psi = 1 - \frac{\Psi^2}{2}$ may be substituted into 3.11 to yield the factor which must be added and subtracted to F so the equation is well-behaved. Upon completing this substitution, it is apparent that the factor H (defined below) when used as shown in 3.13, eliminates the singularity in 3.11.

$$H = \frac{R-r}{\left[(R-r)^2 + Rr\Psi^2 + (Z-z)^2\right]^{3/2}} \tag{3.12}$$

$$4\pi V_a = \int \frac{d\Gamma}{ds} r \int \left\{ \left[F(r,R,\Psi) - H(r,R,\Psi)\right] \right\} d\Psi ds + \int \frac{d\Gamma}{ds} r \int H(r,R,\Psi) d\Psi ds \tag{3.13}$$

(1) (2) (3)

In this manner, as $\cos\Psi$ approaches 1, and r approaches R, the numerator and denominator of the first two terms in 3.13 approach zero at the same rate. The third term in equation 3.13 is now integrated analytically with respect to Ψ from $-\pi$ to π and defined as "A" where:

$$A = \int H(r,R,\Psi)d\Psi = \frac{\frac{d\Gamma}{ds} r(R-r)2\pi}{[(R-r)^2 + (Z-z)^2]\,[(R-r)^2 + (Z-z)^2 + Rr\pi^2]^{1/2}}$$

(3.14)

This analytical solution (3.14) now must be integrated over s numerically. Once again, however, the denominator goes to zero faster than the numerator as r approaches R (as an integration strip approaches a marker). The elimination of this singularity is slightly more complex because the integral is over s, and the relationships between r and s, and z and s are known only in spline form. To solve this problem, the r and z terms are linearized with respect to s as outlined in section 2.3. In this manner, r and z can be represented as simple linear functions of s, and the integration of equation 3.14 with respect to s becomes straightforward. This linearization and the modified form of 3.14 that follows will only be used over the linear region. At all other locations along the vortex sheet, equation 3.14 will be integrated numerically in its original form.

The linear relations defined in equation 2.3.1 will now be substituted into 3.14 for integration over s, resulting in:

$$A_{lin} = 2\pi \int \frac{\frac{d\Gamma}{ds} K_1[R-K_1(S-s)]}{C(S-s)[C(S-s)^2+R^2\pi^2-RK_1(S-s)\pi^2]^{1/2}} \, ds \qquad (3.15)$$

where:

$$C = K_1^2 + K_2^2 = 1$$

$$K_1 = \frac{dr}{ds}\Big|_{s=S}$$

$$K_2 = \frac{dz}{ds}\Big|_{s=S}$$

Before the integration over s can be completed, however, it is apparent that an additional term B (below) needs to be added and subtracted from A_{lin} to avoid the singularity which occurs as s approaches S:

$$B = \frac{2K_1 \frac{d\Gamma}{ds}}{C(S-s)} \qquad (3.16)$$

Equation 3.15 can now be integrated over s from 0 to s_{max}, and the third term in equation (3.13) results in:

$$r \frac{d\Gamma}{ds} \int Ads = \int \frac{\frac{d\Gamma}{ds} r(R-r)2\pi}{[(R-r)^2+(Z-z)^2]\,[(R-r)^2+(Z-z)^2+Rr(\pi)^2]^{1/2}} \, ds \qquad (3.17)$$

outside the linear region and:

3.15 + 3.16 - 3.16 =

$$2\pi\int\left\{\frac{\frac{d\Gamma}{ds}K_1[R-K_1(S-s)]}{C(S-s)[C(S-s)^2+R^2\pi^2-RK_1(S-s)\pi^2]^{1/2}}\right.$$

$$\left.-\frac{2K_1\frac{d\Gamma}{ds}}{C(S-s)}\right\}ds+\frac{2K_1\frac{d\Gamma}{ds}}{C}\log\left|\frac{S-s_{\delta 1}}{S-s_{\delta 2}}\right| \qquad (3.18)$$

within the linear region.

In the last term of 3.18, $s_{\delta 1}$ refers to the first integration strip encountered inside the linear region, and $s_{\delta 2}$ refers to the last integration strip encountered inside the linear region.

When considering a specific marker and stepping through the integration strips (i.e. integrating along the sheet), 3.17 is used until s comes within + or - δ (which defines the linear region- see figure 2.4-1) of the marker. At this point, 3.18 is used until this linear region is exited, and then use of 3.17 is resumed.

Now reviewing the first two terms in 3.13 (F-H) there is one more singularity to be eliminated. This is again resolved by substituting $\cos\Psi = 1-(1/2)\Psi^2$ in the numerator and denominator of F (as shown in 3.19) to yield the additional term needed to eliminate the singular behavior of equation 3.19.

$$F\text{-}H=\frac{R-\frac{R\Psi^2}{2}-r}{[R^2+r^2-2Rr+Rr\Psi^2+(Z-z)^2]^{3/2}}-\frac{R-r}{[R^2+r^2-2Rr+Rr\Psi^2+(Z-z)^2]^{3/2}}$$

(3.19)

The two terms in 3.19 will not cancel unless $R\Psi^2/2$ is included in the numerator of the second term. This factor must also be subtracted from 3.19 so the equation remains unchanged. Therefore:

$$F\text{-}H = \frac{R\cos(\Psi) - r}{[R^2 - 2Rr\cos\Psi + r^2 + (Z-z)^2]^{3/2}} - \frac{R - r - \frac{R\Psi^2}{2}}{[R^2 + r^2 - 2Rr + Rr\Psi^2 + (Z-z)^2]^{3/2}} - \frac{\frac{R\Psi^2}{2}}{[R^2 + r^2 - 2Rr + Rr\Psi^2 + (Z-z)^2]^{3/2}} \qquad (3.20)$$

The first two terms of equation 3.20 will now be renamed F-H'. The third term of equation 3.20 is integrated analytically over Ψ and becomes:

$$\frac{\pi}{r[(R-r)^2 + (Z-z)^2 + Rr(\pi)^2]^{1/2}} - \frac{1}{2r\sqrt{Rr}}\left\{ \log\left[\pi + \sqrt{\pi^2 + \frac{(R-r)^2+(Z-z)^2}{Rr}}\right] - \log\left[-\pi + \sqrt{\pi^2 + \frac{(R-r)^2+(Z-z)^2}{Rr}}\right] \right\} \qquad (3.21)$$

All terms in 3.21 will be well-behaved when integrated along the sheet except the third term which exhibits singular behavior as s approaches S. This singularity is analytically resolved by using a

Taylor's Series expansion, which reveals the additional term to be added and subtracted from this function so it becomes well-behaved. The resulting third term in 3.21 is:

$$\frac{1}{2r\sqrt{b}}\left\{\log\left[\frac{-\pi+\sqrt{\pi^2+\frac{(S-s)^2}{R[R-k_1(S-s)]}}}{\frac{(S-s)^2}{2\pi R^2}}\right]+\log\left[\frac{(S-s)^2}{2\pi R^2}\right]\right\} \quad (3.22)$$

Because of finite precision arithmetic, (3.22) exhibits noisy behavior when:

$$\frac{K1(S-s)}{R} < 0.01$$

It is therefore necessary to do an expansion about 1 to the first log term in 3.22, which will only be used when $\frac{K_1(S-s)}{R} < 0.01$. This first term becomes:

$$\frac{1}{2r\sqrt{Rr}}\log\left[\frac{1}{1-\frac{K_1(S-s)}{R}}\right]. \quad (3.23)$$

Defining the total component of axial velocity (including the additional terms so the equation remains well-behaved over the linear region):

$$V_a=\frac{1}{4\pi}\int\frac{d\Gamma}{ds}r\left\{\int(F-H')d\Psi+\frac{\pi}{r\sqrt{a+b(\pi)^2}}-\right.$$

$$\frac{1}{2r\sqrt{b}}\left[\log\left(\pi+\sqrt{\pi^2+\frac{a}{b}}\right) - A^*\right]\Bigg\} ds + \frac{d\Gamma}{ds}\Big|_{s=S} \frac{8}{\pi\sqrt{b}} \int \log\left[\frac{(S-s)^2}{2\pi R^2}\right] ds$$

$$+ (3.18) \qquad (3.24)$$

where:

$$A^* = \log\left(\frac{-\pi + \sqrt{\pi^2 + \frac{(S-s)^2}{R[R-K_1(S-s)]}}}{\frac{(S-s)^2}{2\pi R^2}}\right) \quad \text{when} \quad \frac{K_1(S-s)}{R} < 0.01$$

$$A^* = \log\left[\frac{1}{1 - \frac{K_1(S-s)}{R}}\right] \quad \text{when} \quad \frac{K_1(S-s)}{R} \geq 0.01$$

Outside the linear region, however, this equation simplifies to:

$$V_a = \frac{1}{4\pi}\int \frac{d\Gamma}{ds} r\Bigg\{ \int (F-H')d\Psi + \frac{\pi}{r\sqrt{a+b(\pi)^2}} -$$

$$\frac{1}{2r\sqrt{b}}\left[\log\left(\pi+\sqrt{\pi^2+\frac{a}{b}}\right) - \log\left(-\pi+\sqrt{\pi^2+\frac{a}{b}}\right)\right]\Bigg\} ds +$$

$$2\pi \int \frac{rG'(s)(R-r)}{a\left[a+b\pi^2\right]^{1/2}} ds \qquad (3.25)$$

where:

$a = (R-r)^2 + (Z-z)^2$

$b = Rr$

3.3 Radial Velocity Component

The radial velocity component of the induced velocity is derived in a similar manner. First, from the Biot-Savart law, the general form of the radial velocity is:

$$V_r = -\int \frac{d\Gamma}{ds} \int \frac{r \cos\Psi (Z-z)}{|l|^3} d\Psi \; ds \tag{3.26}$$

where J is defined as:

$$J = \frac{\cos\Psi \; (Z-z)}{[R^2 - 2Rr\cos\Psi + r^2 + (Z-z)^2]^{3/2}} \tag{3.27}$$

The singularities in J are analytically removed using equations 3.28 and 3.29.

$$K = \frac{Z-z}{[(R-r)^2 + Rr\Psi^2 + (Z-z)^2]^{3/2}} \tag{3.28}$$

$$4\pi V_r = \int \frac{d\Gamma}{ds} r \left\{ \underbrace{\int [J(r,R,\Psi) - K(r,R,\Psi)] d\Psi}_{(1)} + \underbrace{\int K(r,R,\Psi)\, d\Psi}_{(2)} \right\} ds \qquad \underbrace{\qquad}_{(3)}$$

(3.29)

The third term in equation 3.29 is now integrated analytically over Ψ and is found to be:

$$A1 = \frac{\frac{d\Gamma}{ds} 2\pi r(Z-z)}{\left[(R-r)^2 + (Z-z)^2\right]\left[(R-r)^2 + (Z-z)^2 + Rr\pi^2\right]^{1/2}} \tag{3.30}$$

This equation must now be integrated over s, but singularities are encountered identical to those in equation 3.14 in the previous section. This singular behavior is handled in a similar manner, transforming equation 3.30 into:

$$2\pi \int \left\{ \frac{\frac{d\Gamma}{ds} K_2\{R-K_1(S-s)\}}{C(S-s)[C(S-s)^2 + R^2\pi^2 - RK_1(S-s)\pi^2]^{1/2}} - \frac{\frac{d\Gamma}{ds} 2K_2}{C(S-s)} \right\} ds$$

$$+ \frac{\frac{d\Gamma}{ds} 2K_2}{C} \log\left|\frac{S-s_{\delta 1}}{S-s_{\delta 2}}\right| \tag{3.31}$$

$$C = K_1^2 + K_2^2 = 1$$

within the linear region and

$$2\pi \int \frac{\frac{d\Gamma}{ds} r2\pi(Z-z)}{[(R-r)^2 + (Z-z)^2]\ \{[(R-r)^2 + (Z-z)^2 + Rr(\pi)^2]\}^{1/2}} \tag{3.32}$$

outside the linear region.

Returning to the first two terms in equation 3.29, J and K, there is one additional singularity which is removed by adding and subtracting 3.33 to J-K which results in equation 3.34.

$$\frac{(Z-z)\frac{\Psi^2}{2}}{[(R-r)^2 + Rr\Psi^2 + (Z-z)^2]^{3/2}} \tag{3.33}$$

$$J\text{-}K = \frac{\cos(\Psi)\,(Z-z)}{[R^2 - 2Rr\cos\Psi + r^2 + (Z-z)^2]^{3/2}} -$$

$$\frac{(Z-z) - (Z-z)\frac{\Psi^2}{2}}{[R^2 + r^2 - 2Rr + Rr\Psi^2 + (Z-z)^2]^{3/2}} -$$

$$\frac{(Z-z)\frac{\Psi^2}{2}}{[R^2 + r^2 - 2Rr + Rr\Psi^2 + (Z-z)^2]^{3/2}} \tag{3.34}$$

The first two terms of 3.34 will now be renamed J-K'. The third term, denoted D1, is then integrated analytically to yield:

$$D1 = \frac{del\ \pi}{b\left[(R-r)^2+del^2+b(\pi)^2\right]^{1/2}} -$$

$$\frac{del}{2b\sqrt{b}}\left[\log\left(\pi+\sqrt{\pi^2+\frac{a}{b}}\right) - \log\left(-\pi+\sqrt{\pi^2+\frac{a}{b}}\right)\right] \tag{3.35}$$

The third term in 3.35 exhibits singular behavior identical to the third term in equation 3.21, and is handled in a similar manner (which translates into the B^* terms in 3.36). The only difference is in the numerically integrated term which is approximated as zero due to its symmetry about a marker location.

Defining the total component of radial velocity (including the additional terms to eliminate the singularity):

$$V_r = \frac{1}{4\pi} \int \frac{d\Gamma}{ds} \left[r \int (J-K')d\Psi + \frac{(Z-z)\,\pi}{R\sqrt{a+Rr(\pi)^2}} - \right.$$

$$\left. \frac{1}{2R\sqrt{b}} \left\{ (Z-z)\log\left(\pi + \sqrt{\pi^2 + \frac{a}{b}}\right) - B^* \right\} \right] \quad ds \quad + \quad (3.31)$$

(3.36)

where:

$$B^* = K_2(S-s)\log\left[\frac{-\pi + \sqrt{\pi^2 + \frac{(S-s)^2}{R[R-K_1(S-s)]}}}{\frac{(S-s)^2}{2\pi R^2}} \right] \text{ when } \frac{K_1(S-s)}{R} < 0.01$$

$$B^* = K_2(S-s)\log\left[\frac{1}{1 - \frac{K_1(S-s)}{R}} \right] \text{ when } \frac{K_1(S-s)}{R} \geq 0.01$$

If, however, the program is not in the linear region, this equation simplifies to:

$$V_r = \frac{1}{4\pi} \int \frac{d\Gamma}{ds} r \left[\int (J-Kbar)d\Psi + \frac{(Z-z)\,\pi}{b\sqrt{a + b(\pi)^2}} - \right.$$

$$\frac{(Z-z)}{2b\sqrt{b}}\left\{\log\left(\pi+\sqrt{\pi^2+\frac{a}{b}}\right)-\log\left(-\pi+\sqrt{\pi^2+\frac{a}{b}}\right)\right\}\Bigg]ds + 2\pi\int\frac{\frac{d\Gamma}{ds}r(Z-z)}{a[a+b\pi^2]^{1/2}}ds$$

where $a=(R-r)^2+(Z-z)^2$ $b=Rr$ (3.37)

3.4 Tangential Velocity Component

The general form of the tangential induced velocity, developed using the Biot-Savart Law, is defined as:

$$V_t = -\int\frac{d\Gamma}{ds}\int\frac{r\sin\Psi(Z-z)}{|l|^3}\,d\Psi\,ds \qquad (3.38)$$

This term has negligible effect on the wake geometry for the following reasons. The denominator must be small (or the numerator very large) to obtain a significant component of V_t. Since l (the denominator) is the distance between the interval of integration and the marker position, l is only small when Ψ is near $-\pi$ or π (see figure 3.1-2). When Ψ is near $-\pi$ or π, however, $\sin\Psi$ is approximately zero, so the tangential induced velocity is always small and has no significant effect on the wake geometry and is therefore ignored. This is one of the bases of the axisymmetry assumption made in this thesis.

3.5 Self-Induced Velocity of the Tip Vortex

Since the tip vortex has a finite strength and is curved, a self-induced velocity term is included. This self-induced velocity is found to be:

$$w = \frac{\Gamma}{4\pi R}\left\{\ln\frac{8R}{a} - \frac{1}{4}\right\} \tag{3.39}$$

where R is equal to the radial location of the vortex and "a" is its core radius.[13]

3.6 Smoothing

Under certain conditions, minor instabilities occur in the wake beyond 90% radius. If these instabilities are not eliminated, they are carried into the vortex (as the sheet is stretched) and the vortex roll-up becomes unstable. For this reason, a smoothing routine was introduced. This routine, called *filter*, uses a second order difference scheme to smooth certain points in the wake.

Filter solves:

$$-t^2_{fil}\frac{d^2r}{ds^2} + r = r_o$$

which smooths $r_o(s)$ into r(s). This is done in an identical manner for z(s). It is therefore evident that if $t_{fil} = 0$, then $r=r_o$ and there is no smoothing, alternatively if t_{fil} approaches infinity, then $\frac{d^2r}{ds^2}=0$ (i.e. r(s) is linear in s, and the wake sheet is "smoothed" straight).

The degree of smoothing is therefore controlled locally by t_{fil}. T_{fil} is set to a specific value (usually between 0.01 and 0.02) if the marker being considered is in need of smoothing and 0 if no smoothing is necessary.

Although a general criterion was followed in defining t_{fil} as either zero or some finite value, modifications were necessary for each test case as defined below.

Circular Disk Distribution

For this fundamental test case, which looks at the downwash on a translating disk, no smoothing was necessary.

Elliptical Circulation Distribution

Since the elliptical distribution is monotonically decreasing, geometric inflection points in the wake are used to enact the smoothing routine. The markers in need of smoothing are located by considering subsequent sets of four points (forming 3 vectors), and multiplying the cross product of the first and second vectors times the cross product of the second and third vectors. If this product is found to be negative, an inflection point has been located and t_{fil} is set to 0.01 for the marker being considered, otherwise t_{fil} is set to 0.

Miller Distribution

For this distribution there is a natural inflection point in the wake at approximately 95% radius.[14] To avoid erroneous smoothing in this area, the smoothing routine used for the elliptical distribution was only applied from 0.96-1.0 R. In addition, a stronger smoothing parameter (0.015) was used.

4. An Elliptically Loaded Blade

For an elliptically loaded blade at τ=0:

$$\Gamma(r) = .02\sqrt{1-(r)^2}$$

r->[0,1]; z=0

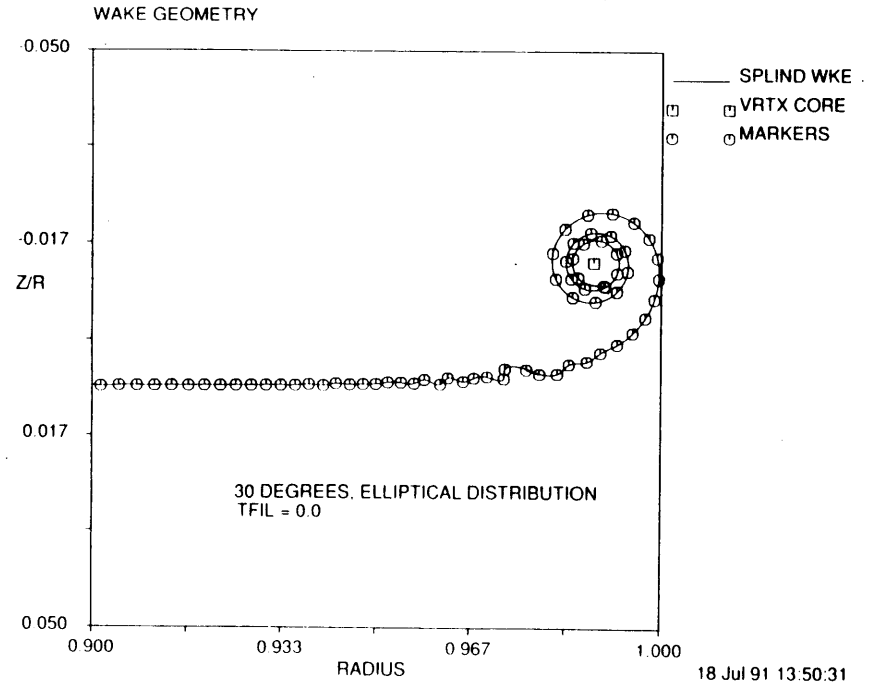
Due to the singularity at the tip in the initial circulation distribution, a tip vortex is instantaneously formed. Pullin's self-similar solution is applied here, as discussed in Section 2.5. By following the steps outlined in that section, the scaling parameters a and t are found to be 0.01396 and 0.1373. These parameters are then used to scale the outer portion of the circulation distribution and wake geometry appropriately, while also giving the strength, location and radius of the tip vortex. The resulting circulation distribution and initial wake geometry are shown in Figure 2.5-1.

For this classical case, several issues are investigated as the blade passes through 45 degrees and the wake convects downstream.

(1) effect of smoothing on wake geometry
(2) effect of time-step on wake geometry
(3) effect of linear region size on wake geometry
(4) evolution of the wake from $\Psi = 0$ to $\Psi = 45$ degrees

4.1 Effect of Smoothing

As previously stated, the smoothing parameter (t_{fil}) used for this distribution was 0.01. In Figure 4.1-1, the wake at Ψ=30 degrees is shown before and after smoothing.



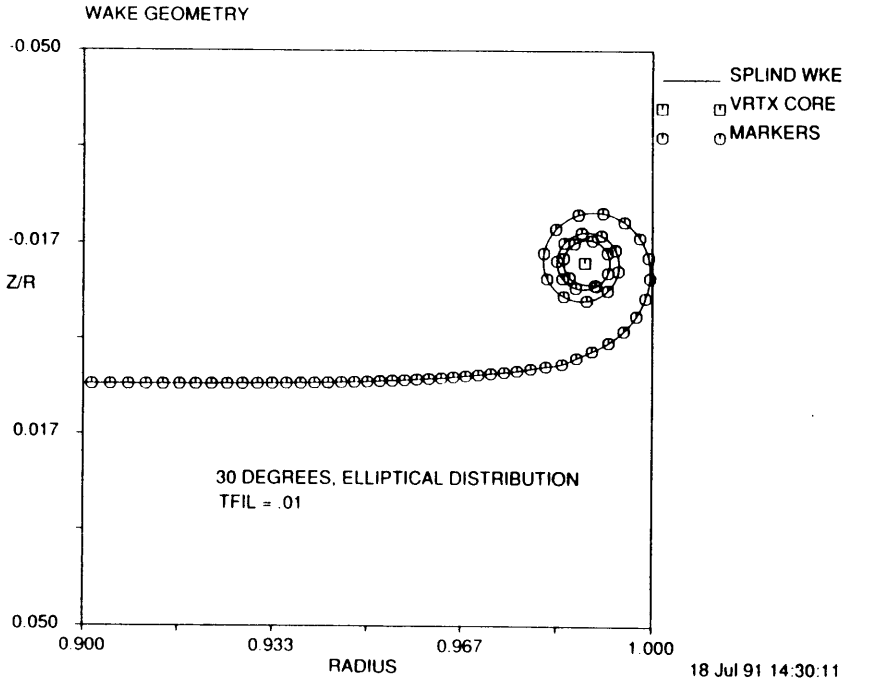
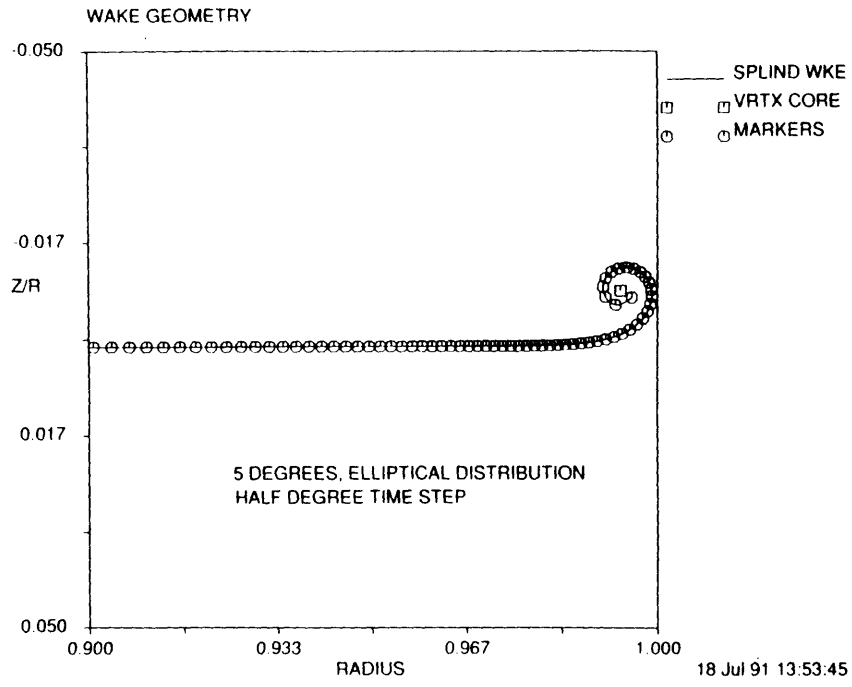


Figure 4.1-1 Effect of Smoothing Wake Geometry

It is important to note that while all inflections are eliminated, the curvature of the wake and the extent of vortex roll-up are not changed significantly. Originally, a smoothing routine was adapted which smoothed every marker along the sheet, rather than only those near an inflection point. The result was a significant loss of curvature in the wake beyond 95% radius along with a large decrease in the diameter of the vortex.

4.2 Dependence of Wake Solution on Time Step

A Predictor-Corrector difference method was utilized for the time-step scheme, providing second-order accuracy. The wake was normally set to convect in one degree increments downstream of the blade. To validate convergence, however, half degree time steps were also tested on the elliptical distribution. The resulting wake geometries at various time steps downstream were compared with wake geometries using one degree increments. Figure 4.2-1 shows one of these wake comparisons at Ψ=5 degrees. The slight difference seen between the two roll-up's indicates the coarseness of the time step is borderline.



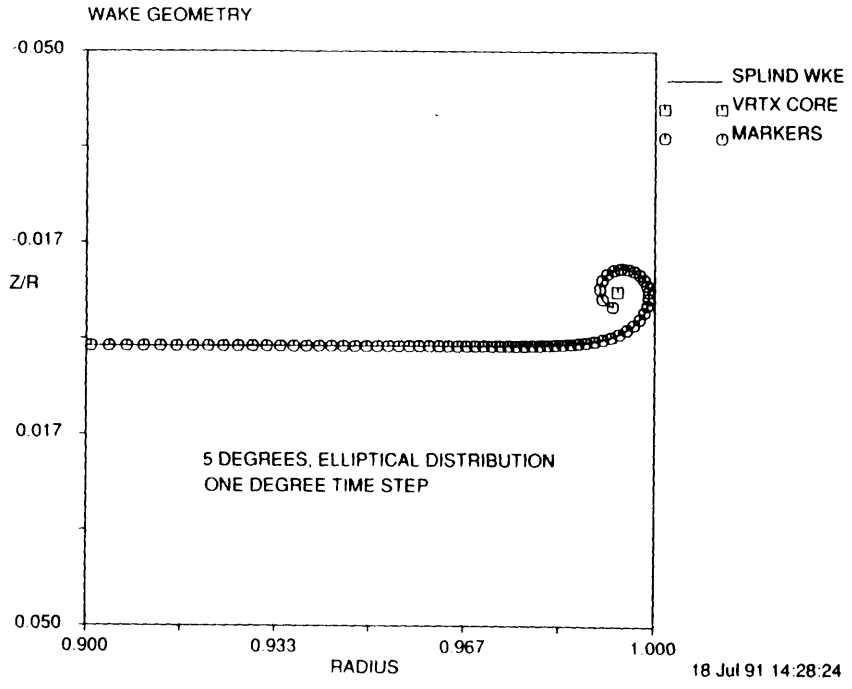
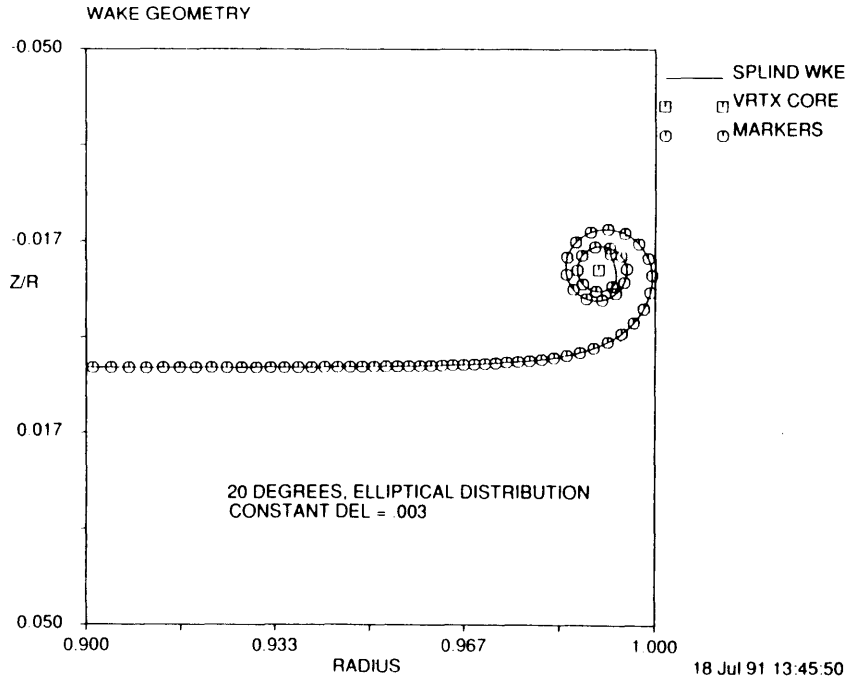


Figure 4.2-1 Effect of Time Step on Wake Geometry

4.3 Dependence of Solution on Linear Region Size

The linear region, as explained in Section 2.3, is a region where the equations for the induced velocity are modified to eliminate singularities that arise as a variable of integration approaches a marker location. The difference between the modified and unmodified equations involves only higher order terms. These terms should have negligible effect on the resulting induced velocities, and therefore the solution should be independent of the size of the linear region (as long as the linear relations in section 2.3 are preserved). Test cases were run for several linear region sizes, and the resulting wake geometries confirmed that the solution is in fact independent of linear region size. Figure 4.3-1 shows a comparison between a constant linear region size of $\delta = 0.003$ versus a variable linear region size (depending on marker location) that was normally used.



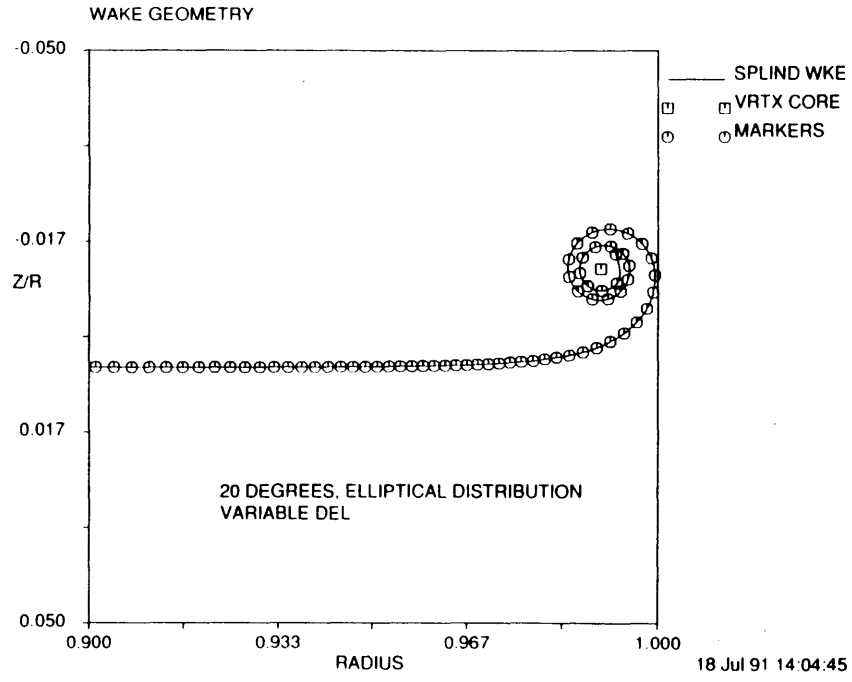
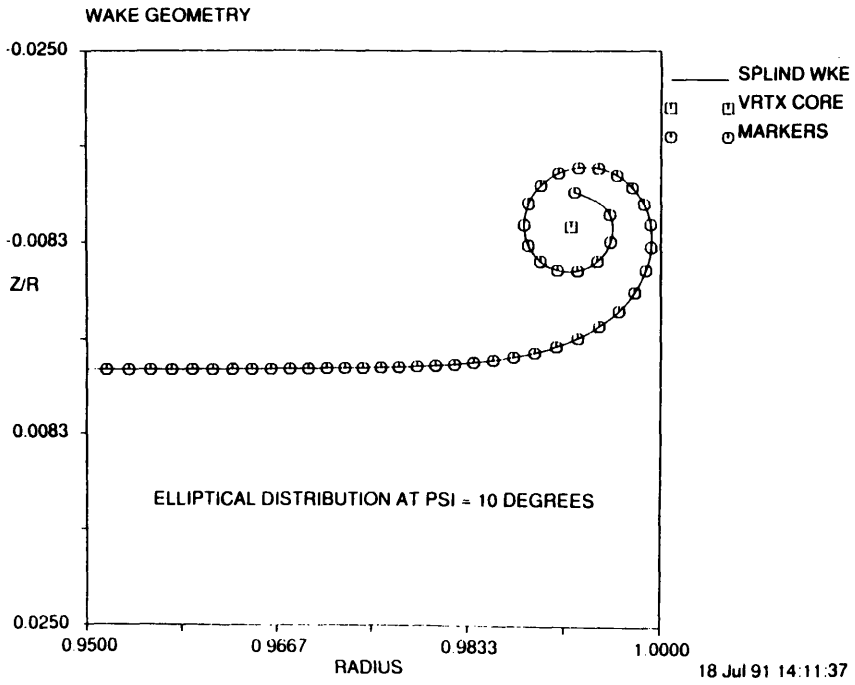


Figure 4.3-1 Effect of Linear Region Size on Wake Solution

4.4 Calculated Wake Geometry

In Figures 4.4-1, 4.4-2, and 4.4-3 the wake is shown convecting downstream as the blade moves from 0 to 45 degrees. As expected, the tip vortex convects downstream slower than the rest of the vortex sheet. There is a significant roll-up in the near-wake, but the vortex remains well-behaved (assisted by the smoothing routine). Only the tip is shown in the following figures since the wake remains relatively flat inboard.



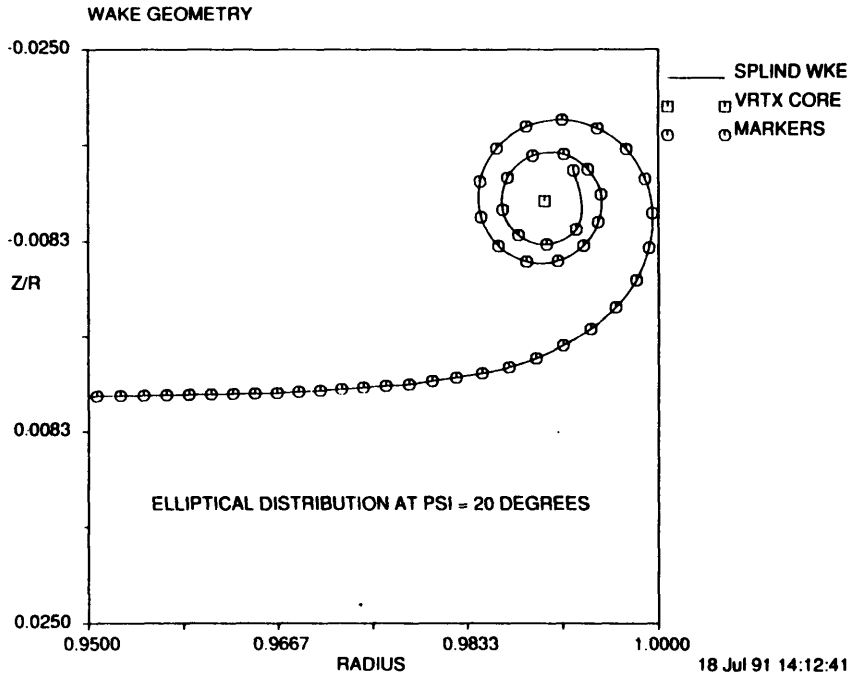
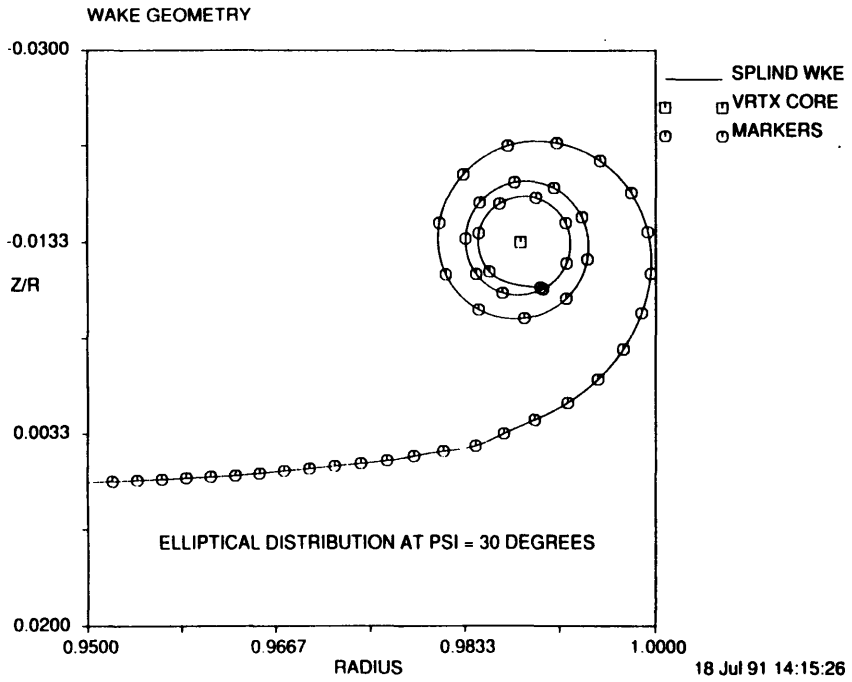


Figure 4.4-1 Wake Profiles for the Elliptically Loaded Blade



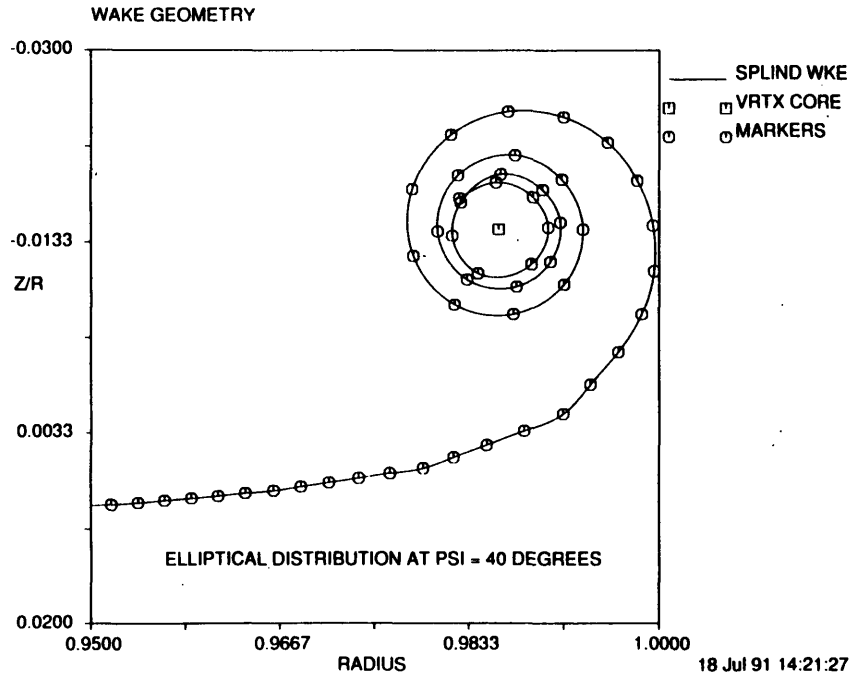
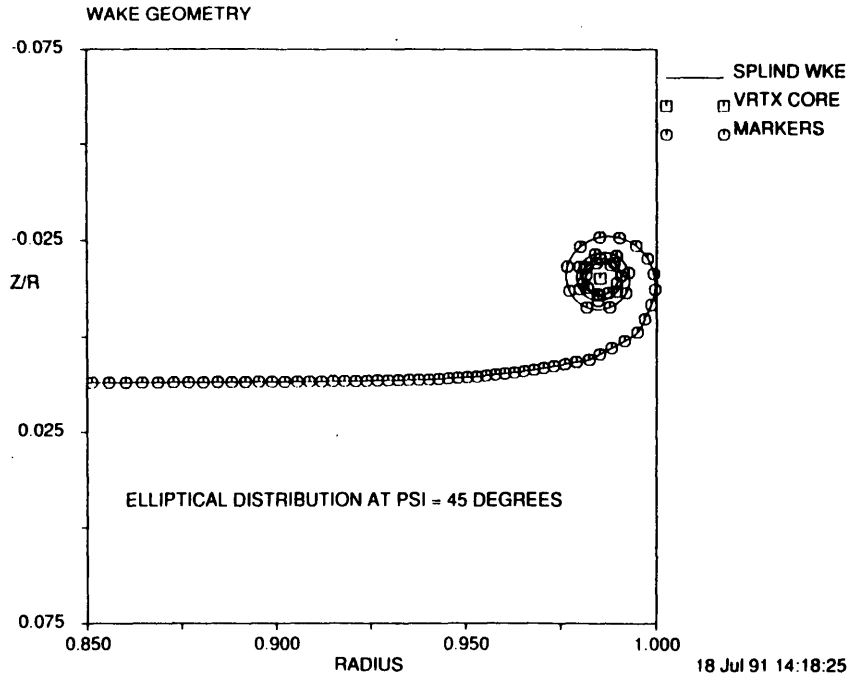


Figure 4.4-2 Wake Profiles for the Elliptically Loaded Blade



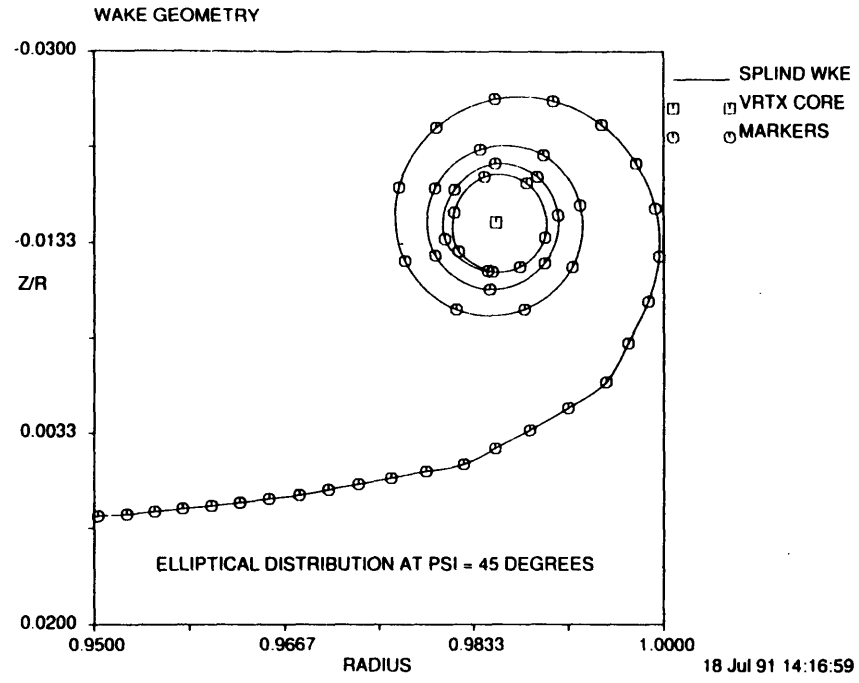


Figure 4.4-3 Wake Profile for the Elliptically Loaded Blade at $\Psi = 45^{o}$

If the sheet is allowed to convect further downstream, the extent of roll-up may lead to instabilities which could be handled using a dumping technique outlined by Hoeijmakers.[15] Using this approach, the vortex sheet is only allowed four revolutions, and any vorticity past that point is fed into the vortex core. The core is then re-located to preserve the center of vorticity. The amount of roll-up is checked at each iteration and the process is repeated.

5. Translating Circular Disk

As a check case, a circulation distribution was developed for an oblate ellipsoid (with minor axis b and major axis a), moving parallel to its axis of rotation b, in the limit as its minor axis goes to zero.[16] For this case, an irrotational flow due to a circular disk with radius "a" results as shown in Figure 5-1. Since the boundary condition states that no flow can pass through this disk, all particles on the disk must descend at the same rate as the disk, producing a constant downwash condition. The intent of this test case is to validate the basic analytic theory developed in this thesis and its application in the present code.

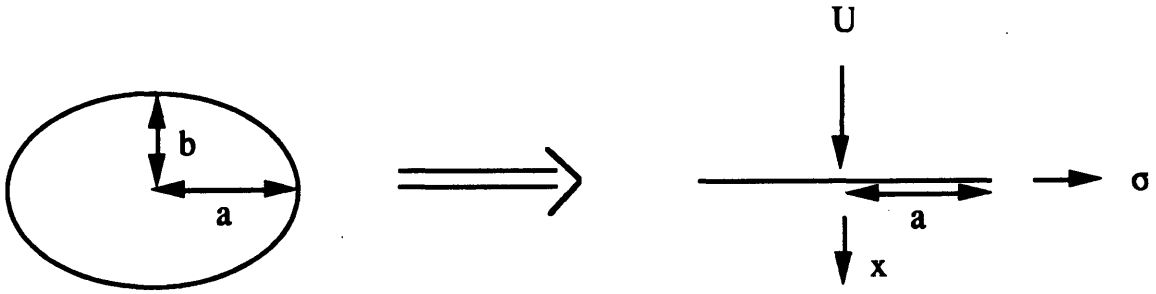


Figure 5-1 Oblate Spheroid Collapsed to Circular Disc

In Batchelor, the velocity potential for this flow is given as:

$$\Phi = -\frac{2aU}{\pi}\cos\eta$$

where ζ and η are the elliptical coordinates obtained through the transformation:

$$x + i\sigma = (a^2-b^2)^{1/2}\sinh(\zeta+i\eta)$$

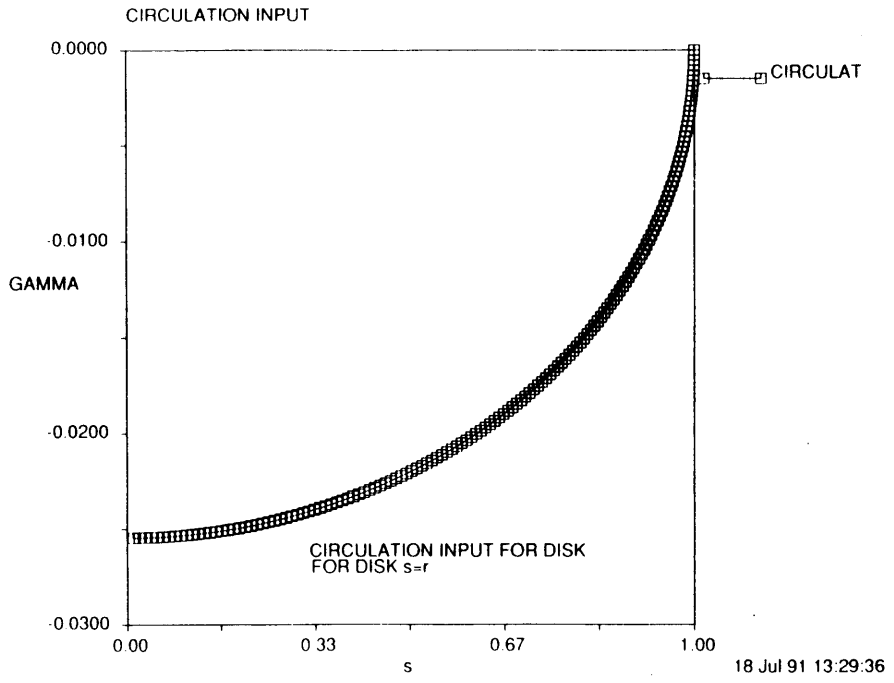
For the circular disk, using the coordinate system shown in figure 5-1, it is apparent x and b equal zero while a equals unity. This leaves the transformation $\sigma = \sin(\eta)$, where ϕ can be written as:

$$\phi = \frac{-2aU}{\pi}(1-\sigma^2)^{1/2}$$

The circulation distribution along the plate is simply the change in velocity potential across the plate giving:

$$\Gamma(s) = \frac{-4aU}{\pi}(1-r^2)^{1/2}$$

Using this distribution (see figure 5-2), a flat wake exhibiting constant downwash should result. Figure 5-3 shows the resulting downwash distribution for this test case. The circulation distribution was derived assuming a non-dimensional downwash of 0.02 and the code accurately reproduces this value.



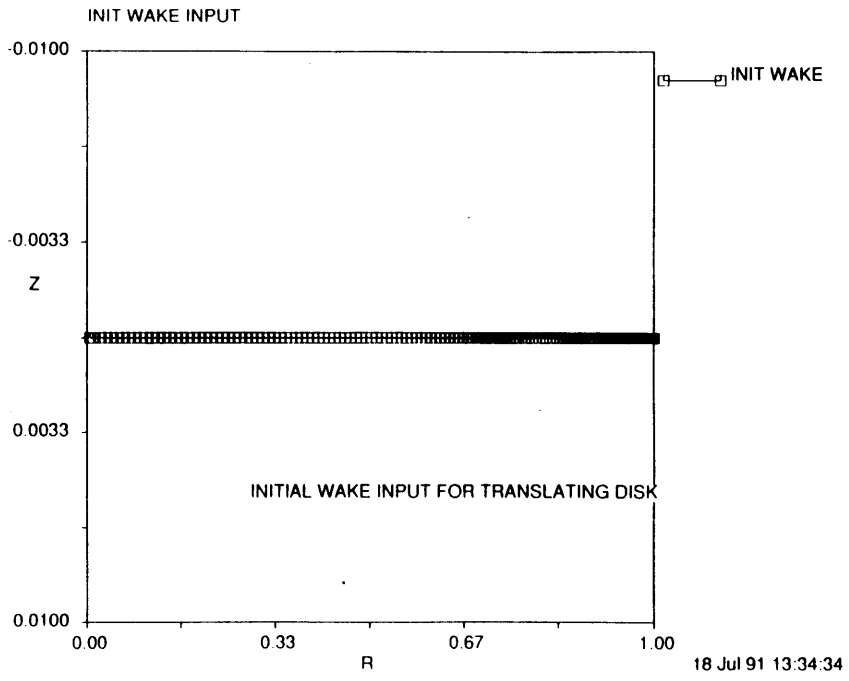


Figure 5-2 Circulation and Wake Distribution for the Oblate Spheroid

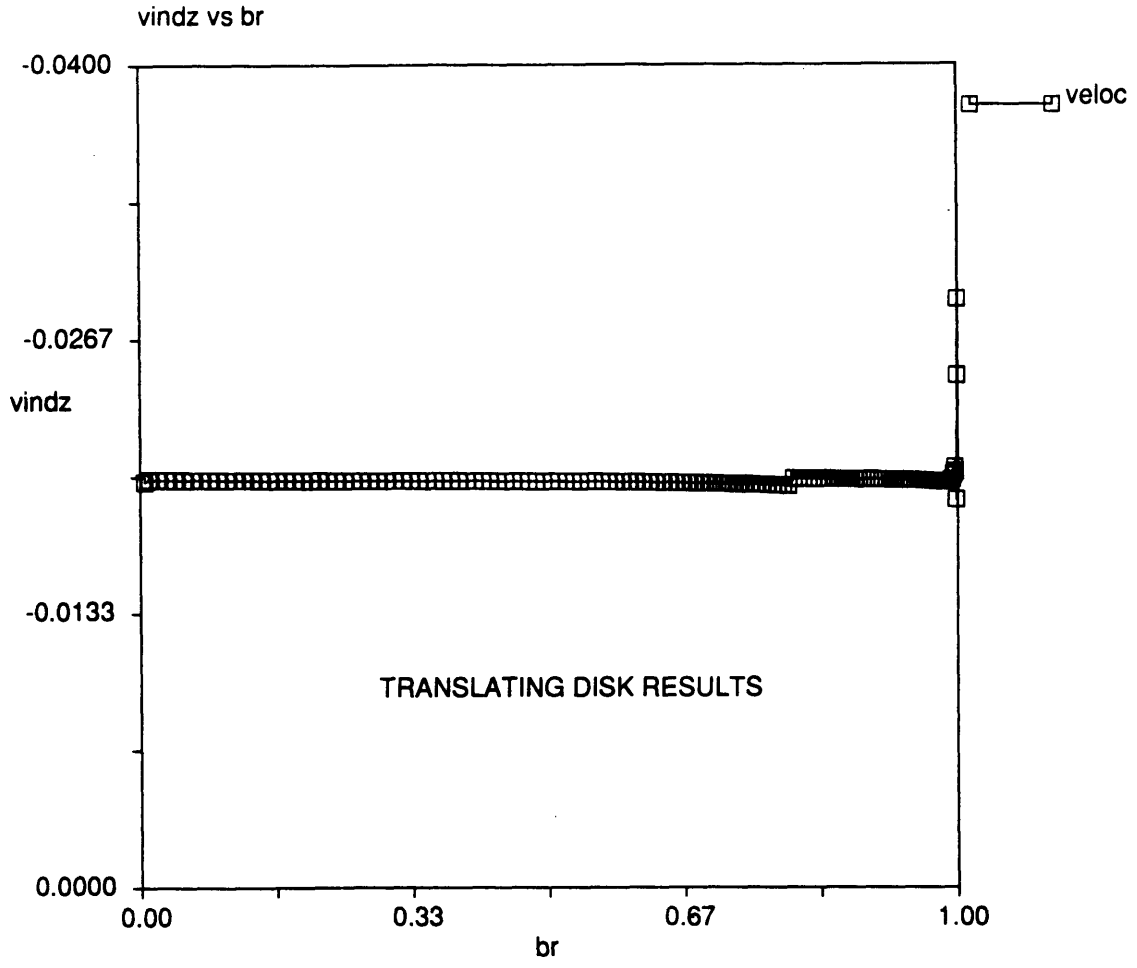
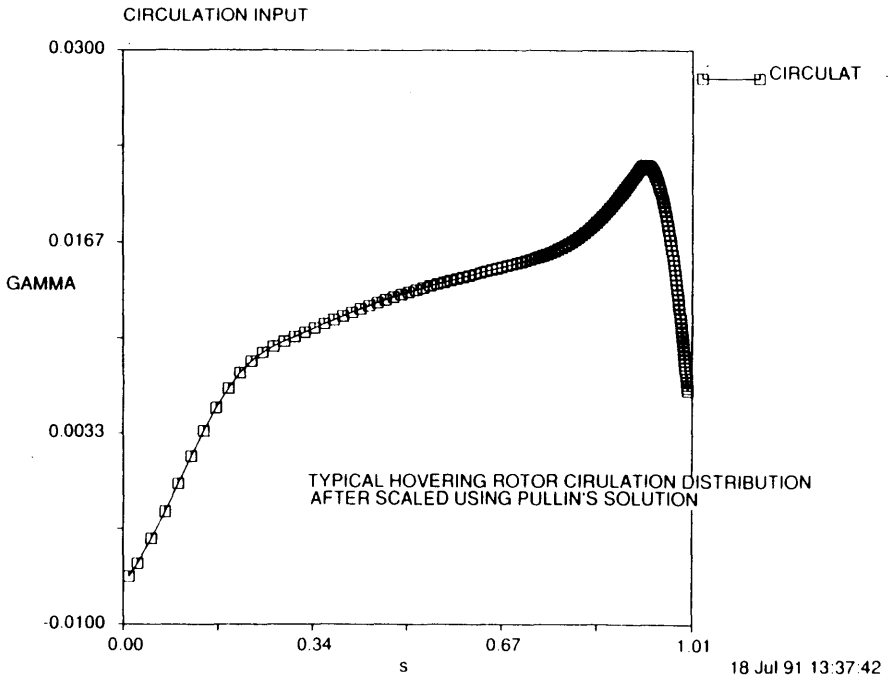


Figure 5-3 Downwash Distribution for the Oblate Spheroid

Two abnormalities are seen in this plot, one at 80% radius and one at the tip. The slight increase in accuracy seen at .8 radius is due to a higher density of integration strips placed outboard of this location. This is necessary to correctly account for the vorticity in this area where the extent of curvature in the circulation distribution is significantly increased. At the tip spikes are seen on the outer 0.02%. These are due to the inability of the model handle $\frac{d\Gamma}{ds}$ approaching infinity at the tip.

6. Typical Hovering Rotor Loading

The circulation distribution shown in figure 6-1 was chosen as the final test case because there has been considerable debate over the resulting wake geometry. This circulation distribution was obtained iteratively by Miller[17] using the entire wake, but has been used in discrete near-wake analyses to try to understand the resulting wake behavior.[18] The main point in question is whether or not a mid-span vortex exists. Full-wake discrete models[19,20] have shown the formation of a mid-span vortex, however this contradicts experimental results. Since the continuous vortex sheet representation of the present approach is physically more correct, it was hoped that the character of the wake roll-up would more closely match experimental evidence. Figures 6-2 through 6-5 show the roll-up of the wake as the blade advances through 45 degrees.



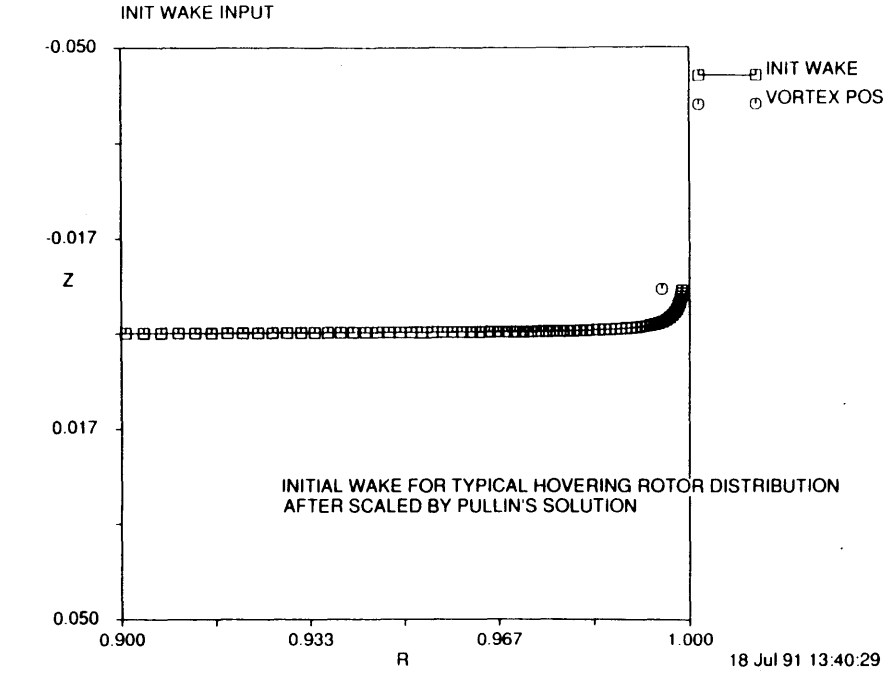
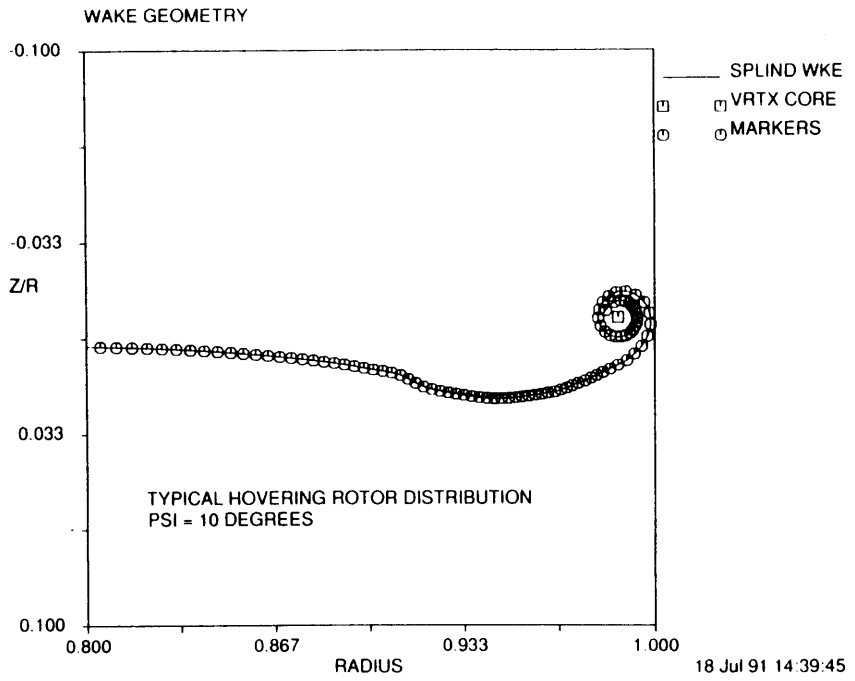


Figure 6-1 Typical Hovering Rotor Circulation & Wake Distribution



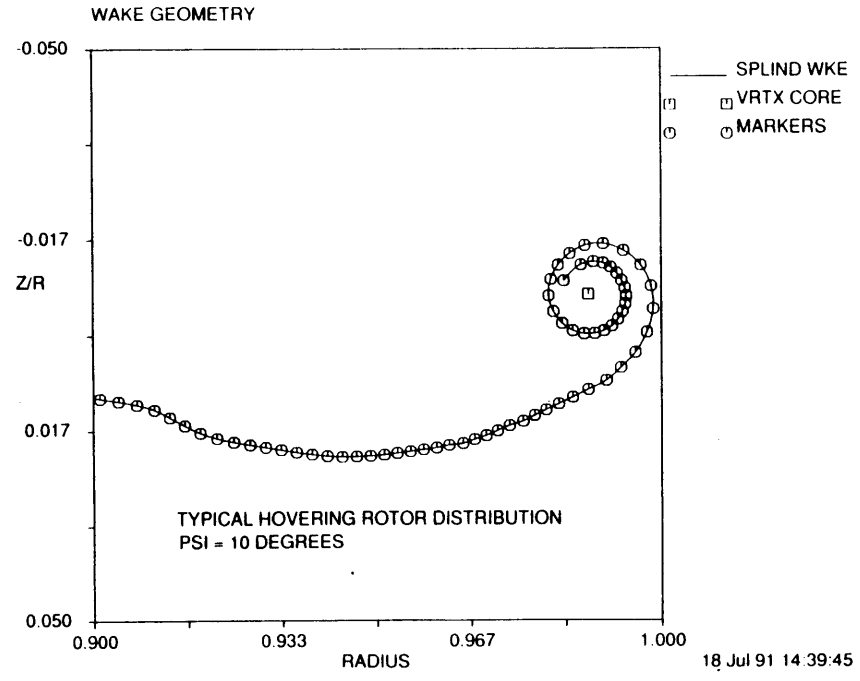
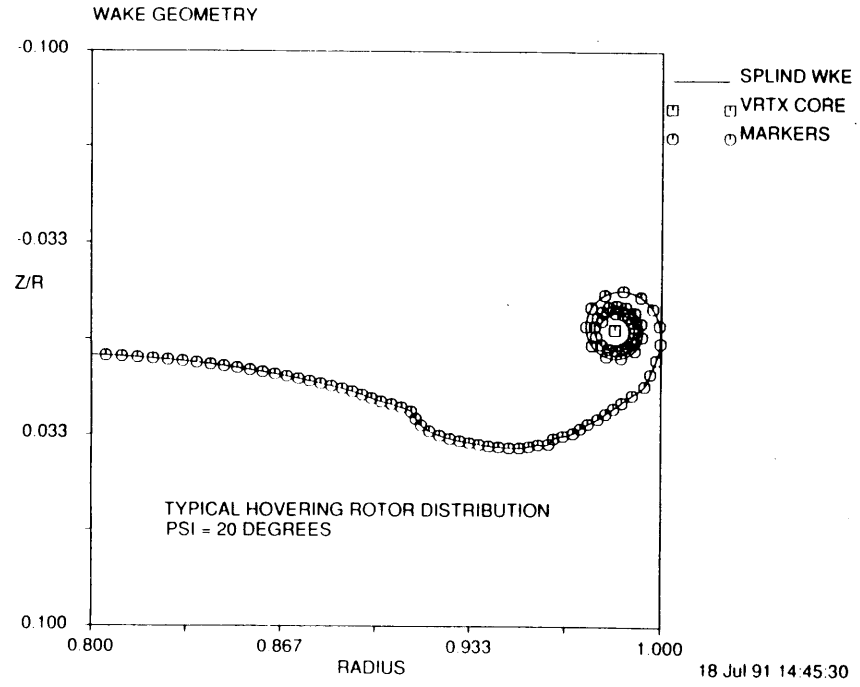


Figure 6-2 Wake Profile for Typical Hover Circulation, $\Psi = 10^{o}$



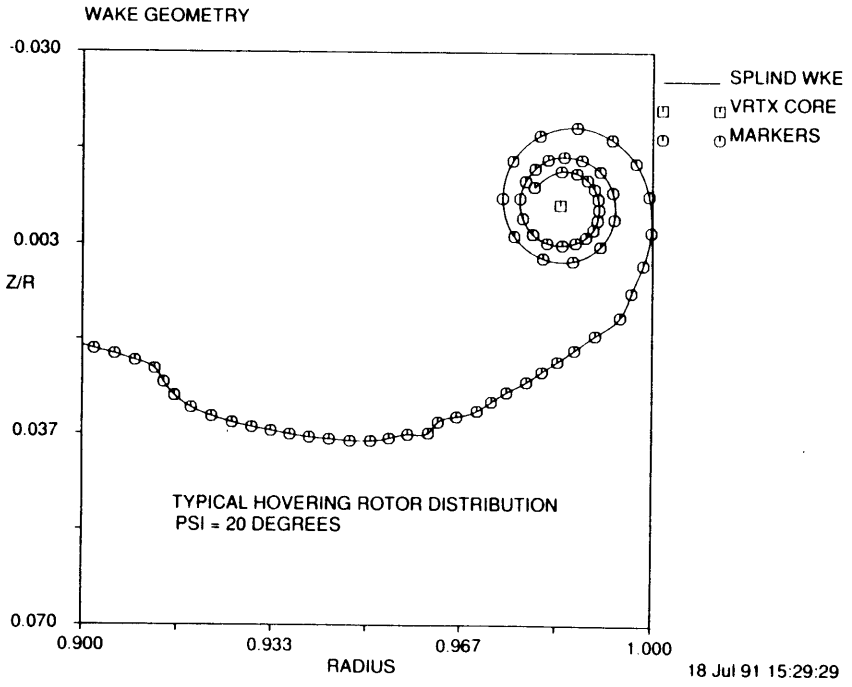
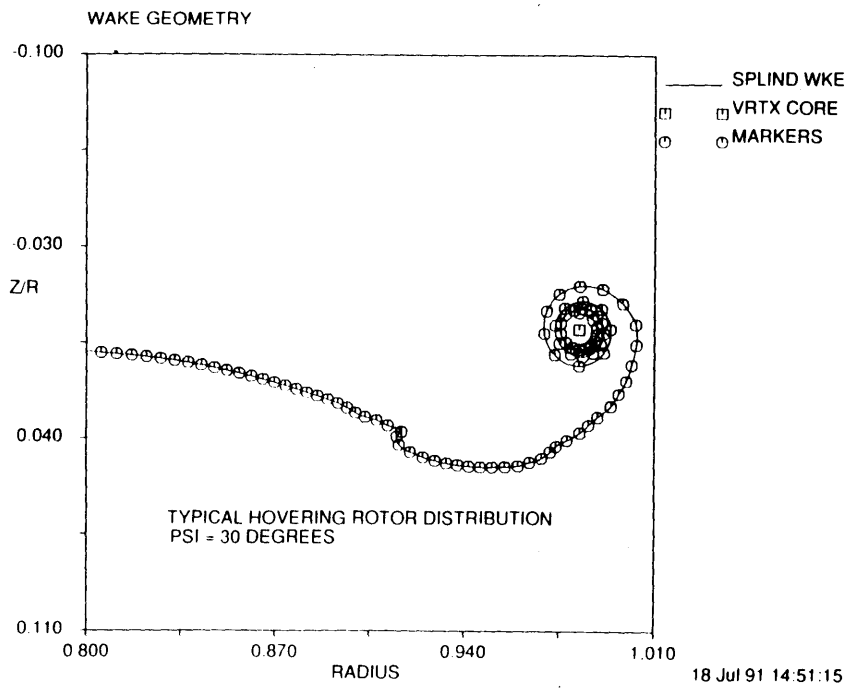


Figure 6-3 Wake Profile for Typical Hover Circulation, $\Psi = 20^{o}$



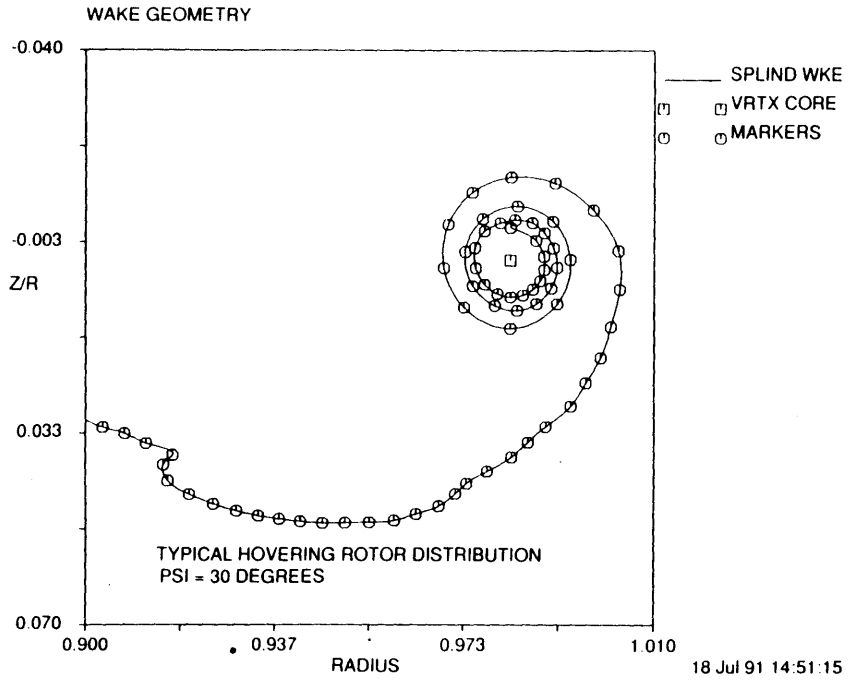
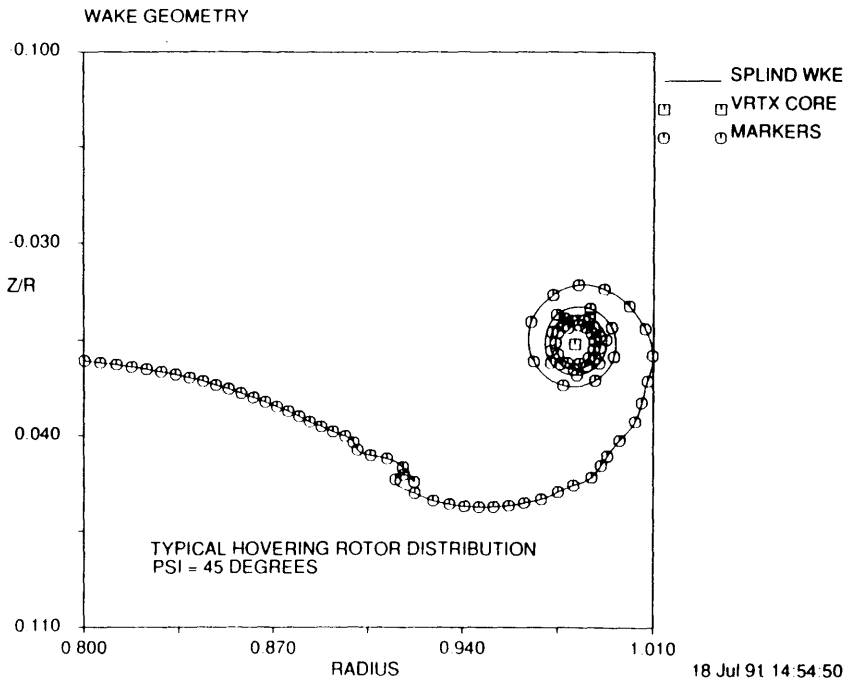


Figure 6-4 Wake Profile for Typical Hover Circulation, Ψ = 30°



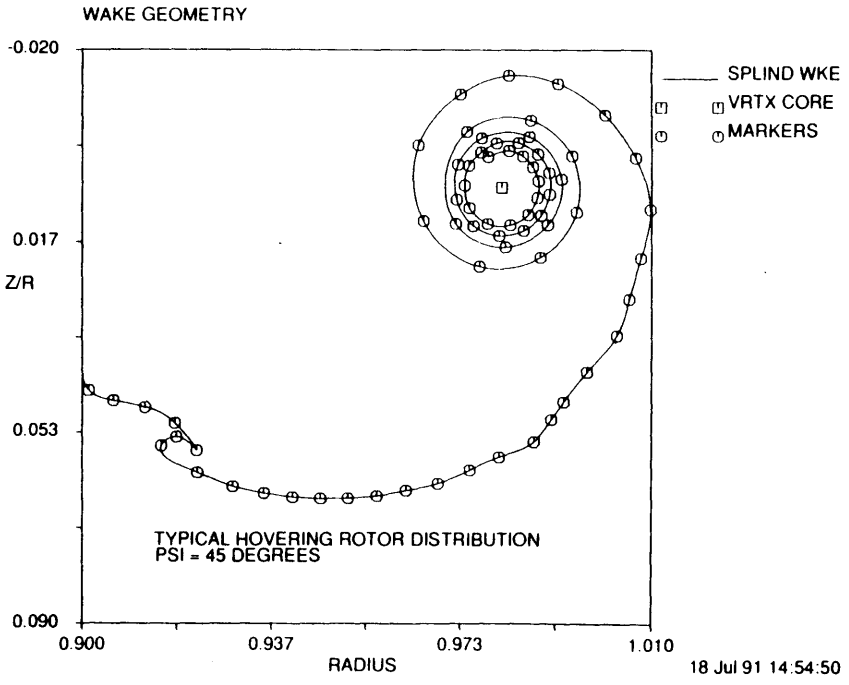


Figure 6-5 Wake Profile for Typical Hover Circulation, $\Psi = 45^{o}$

It is seen in figure 6-3, that at Ψ=20 degrees an additional vortex starts to appear. It is induced at approximately 92% blade radius which corresponds with the peak in circulation shown in figure 6-1 (remember only smooth past 96% radius). This roll-up is opposite to that seen at the tip as expected, and becomes more pronounced as the blade convects downstream.

By comparing the elliptical and typical hover circulation distributions, it appears the the outer portion of the wake for the typical hover distribution should convect faster than the elliptical case, and the inner portion slower, which is confirmed by comparing the resulting wake geometries.

8. Conclusions and Future Recommendations

The program resulting from this research can be utilized to assist in resolving the arbitrariness of the finite core size in discrete models, and as the near-wake component of a hovering or forward flight full-wake code.

First, understanding more completely the physical behavior of the near-wake may give insight to why current computer models are not currently matching experimental results.

There seems to be a consensus that representing the wake as a vortex sheet rather than a series of discrete filaments leads to a more realistic model, but at the expense of theoretical and computational

simplicity. Since one methods advantage is the other's weak point, it seems a study comparing the two models would be helpful in determining a fairly simplistic but more physically correct model. It was hoped that a comparison between discrete versus continuous models could be done in this thesis, but was impossible due to the unavailability of a hovering, discrete, near-wake code. By creating a code similar to the one contained herein (but using a discrete model), a solid analysis could be done. By comparing the results between these two models, an appropriate core size may become apparent for the discrete model, making its solution much less arbitrary while keeping its simplicity in place.

It does not appear however, that doing this alone will result in the agreement of experimental and computational results, since the more complex continuous model still revealed a mid-span vortex which has yet to be documented experimentally. The reason for this occurrence is still unclear and will possibly only be resolved when viscous terms are included in the wake analysis.

The other primary use for this code, implementation into hovering and forward flight full-wake codes, can be accomplished with minimal changes.

For hovering cases, as outlined in reference 13, the integration strips are defined as semi-circles from $\Psi=0$ to $\Psi=\frac{\pi}{2}$. By reviewing equations 3.5, 3.6, and 3.7, it is apparent this modification can be accomplished by simply dividing these equations by 2.

For forward flight, however, not only the limits of integration over Ψ changes, but the actual wake geometry changes as well. The upper limit of Ψ in this case defines the near wake, with the intermediate and far wakes following. Since the wake geometry is different for forward flight due to the additional forward flight velocity term, the location of the integration intervals will have to be modified. This can be accomplished by appropriately modifying the l vector in equation 3.3 and carrying these modified terms throughout the equations. It should be emphasized that except for the l vector, the overall numerical Biot-Savart analysis developed in this thesis remains unchanged.

[1] Miller, R. H., *ASimplified Approach to the Free Wake Analysis of a Hovering Rotor,* Vertica, 6, 1982, pp. 89 - 91.

[2] Donaldson, C., Snedeker, R.S. and Sullivan, R.D., *Calculation of the Wakes of Three Transport Aircraft in Holding, Take-off and Landing Configurations, and Comparison with Experimental Measurements*, AFOSR-TR-73-1594, 1973.

[3] Rossow, V., *On The Inviscid Rolled-Up Structure of Lift Generated Vortices*, Journal of Aircraft, 1973.

[4] Miller, R. H., *Methods for Rotor Aerodynamic and Dynamic Analysis*, Progress in Aerospace Sciences, 22(2), 1985, p. 124.

[5] Brower, M., *Free Wake Techniques for Rotor Aerodynamic Analysis, Volume III: Vortex Filament Models*, ASRL TR 199-3, Massachusetts Institute of Technology, 1982.

[6] Tanuwidjaja, A., *Free Wake Techniques for Rotor Aerodynamic Analysis, Volume II, Vortex Sheet Models*, ASRL-TR-199-2, Massachusetts Institute of Technology, 1982.

[7] Miller, R. H., *Methods for Rotor Aerodynamic and Dynamic Analysis*, Progress in Aerospace Sciences, 22(2), 1985, p. 120.

[8] Miller, R. H., *Methods for Rotor Aerodynamic and Dynamic Analysis*, Progress in Aerospace Sciences, 22(2), 1985, pp. 116.

[9] Miller, R.H., *Simplified Free Wake Analyses for Rotors.*, ASRL-TR-194-3, Massachusetts Institute of Technology, 1981.

[10] Pullin, D.I., *The Large Scale Structure of Unsteady Self-Similar Rolled-Up Vortex Sheets*, Journal of Fluid Mechanics, 88, 1978, pp. 401-408.

[11] Hoeijmakers, H.W.M., *An Approximate Method for Computing Inviscid Vortex Wake Roll-Up*, NLR TR 85149 U18-26, National Aerospace Laboratory NLR, The Netherlands, 1985.

[12] Anderson, J. D., Fundamentals of Aerodynamics. New York, McGraw-Hill Book Company, 1984, pp. 234-241.

[13] Roberts, T. W., *Computation of Potential Flows With Embedded Vortex Rings and Applications to Helicopter Rotor Wakes.*, CFDL-TR-83-5, Massachusetts Institute of Technology,1983.

[14] Miller, R. H., *Methods for Rotor Aerodynamic and Dynamic Analysis*, Progress in Aerospace Sciences, 22(2), 1985, p. 123.

[15] Hoeijmakers, H.W.M., *An Approximate Method for Computing Inviscid Vortex Wake Roll-Up*, NLR TR 85149 U18-26, National Aerospace Laboratory NLR, The Netherlands, 1985.

[16] Batchelor, G.K. An Introduction to Fluid Dynamics. Cambridge University Press, Cambridge, 1988, pp. 456-458.

[17] Miller, R. H., *Methods for Rotor Aerodynamic and Dynamic Analysis*, Progress in Aerospace Sciences, 22(2), 1985, p. 123.

[18] Miller, R.H., Ellis,S.C. and Dadone, L., *The Effects of Wake Migration During Roll-Up On Blade Airloads,* Vertica,13(1), 1989, pp. 3-6.

[19] Miller, R. H., *Methods for Rotor Aerodynamic and Dynamic Analysis*, Progress in Aerospace Sciences, 22(2), 1985, p. 123.

[20] Miller, R.H., Ellis,S.C. and Dadone, L., *The Effects of Wake Migration During Roll-Up On Blade Airloads,* Vertica, 13(1), 1989, p.5.

Appendix - Computer Programs

```
C ************************VORTEX.F************************************

c     Melinda godwin, september 28,1990


C************NUMBERED EQUATIONS REFERRED TO IN THE COMMENTED
C************SECTIONS CAN BE FOUND IN THESIS.



c  I have two things defined as "a" in this program
c  but they do not interact.



c
c This program is to calculate the two components of the induced
c velocity, Vz and Vr.  This will be done using the Biot Savart Law.
c The induced velocity will be calculated at various positions
c downstream of the blade along the vortex sheet.  The resulting
c wake will then be plotted. The sheet is not flat, but from a
c side view is curved and rolls up toward the tip where a tip vortex
c is located.
c
c
      dimension delrsq(301),     delzsq(301),       vrold(301),
     &          ls(2000),        bs(301),           lr(2000),
     &          dgambs(2000),    x(301),            gam(301),
     &          bz(301),         bzold(301),        dels(301),
     &          dgamls(2000),    vindr(301),        br(301),
     &          vindz(301),      brold(301),        gb(2000),
     &          lsdan(2000),     lsdana(2000),      lz(2000),
     &          gbs(301),        rbs(301),          zbs(301),
     &          zbr(301),        s(2000),           expls(2000),
     &          gbbtbs(2000),    tbs(2000),         probx3(2000),
     &          proby3(2000),    yaxis(2000),       xaxis(2000),
     &          pltbs(2000),     pltbr(2000),       pltbz(2000),
     &          vzold(301)


      dimension nper(2),         ilin(2),           isym(2)

      real ls,      lr,       lz,     kdphi,  lsdan,  lsdana
      real integ2,  integ1,   k1,     k2

      data ilin/1,0/
      data isym/1,2/

      character * 30 titl(2)
      character * 30 pltitl
```

```
      pi = 4.*atan(1.)

c     (pi/4)/50 ~.016~1 deg ... t* = t(omega)
c     when t*=1, t=time to go one radian, therefore vary t*=0. to pi/4


      delta = .01745

      TIME=0.


      WRITE(*,76)
 76   FORMAT(1X,'WHAT IS NSTOP? - 99 if prog stopped')
      READ(5,77) NSTOP
 77   FORMAT(I2)

      write(*,885)
 885  format(1x,'WHAT CASE ARE YOU RUNNING? 1-DISK, 2-ELL,3-TYP HOV')
      READ(*,886) NCASE
 886  FORMAT(I2)
      write(*,886) ncase

c tfil is the parameter which controls the strength of the local
c smoothing.

      write(*,80)
 80   format(1x,'WHAT IS TFIL?')
      READ(*,81) SMINC
 81   FORMAT(F10.4)

c
cThe following is for fildis.f
            write(*,90)
 90         format(1x,'Enter amark: ellip- .9, disk- .8, Typ Hover-.9')
            read(5,*) amark
            write(*,91) amark
 91         format(1x,'amark equals',1x,f10.4)
            write(*,922)
 922        format(1x,'Enter vardint:',/,
     &      'ellip- .0002, disk- .0002, typical hover- .002')
            read(5,*) vardint
            write(*,93) vardint
 93         format(1x,'vardint equals',1x,f10.4)
            write(*,94)
 94         format(1x,'****amark and vardint are found in fildis.f')



C******THIS IS MAIN SECTION TO MAKE PROGRAM ABLE TO STOP AND START
C      SEE END OF CODE TO SEE WHAT IS WRITTEN TO THESE FILES.

      IF(NSTOP.EQ.99) THEN
         write(*,854)
 854     format(1x,'RESTARTING VORTEX')
```

```
         open(unit=63, file='wakestore.dat',status='old')
         open(unit=64, file='gamma.dat',status='old')
         open(unit=65, file='miscstore.dat',status='old')

         READ(65,*) NUMBR,TIME,II,RCORE,ZCORE
         II=II+1

            nbr=numbr
            nvortx=numbr+1
            if (ncase.eq.1) then
               nmark = nbr
            else
               nmark = nvortex
            endif

            do 78 i=1,nmark
            READ(63,*) BR(I),BZ(I)
            if(i.ne.nvortx) then
            READ(64,*) GAM(I)
            endif
 78         continue

         close(unit=63)
         close(unit=64)
         close(unit=65)

      ENDIF

      IF(NSTOP.EQ.99) GOTO 150

c *****END MOST ADDITIONS TO MAKE PROGRAM ABLE TO STOP AND START


c for translating disk case
      if(ncase.eq.1) then
         call disk(BR,BZ,BS,BSMAX,GB,TBS,NUM4,NUMBR)
         a=1
         t=1
      endif

c for elliptical distribution case
         if(ncase.eq.2) then
            write(*,*) ncase
           call ellip(BR,BZ,BS,BSMAX,GB,TBS,NUM4,NUMBR,a,t)
         endif

c for typical hovering rotor case
            if(ncase.eq.3) then
             call typhov(BR,BZ,BS,BSMAX,GB,TBS,NUM4,NUMBR,A,T)
            endif

      write(*,101) bsmax
 101  format(1x,'bsmax=',f10.4)
```

```
c nbr is number of markers along sheet
      nbr=numbr
c nvortx is number of markers along sheet plus tip vortex
      nvortx=numbr+1

c splining gamma versus sheet location
      CALL SPLIND(GB,GBBTBS,TBS,NUM4,999.,999.)


         open(unit=64, file='gamma.dat',status='unknown')

c loop to define gamma at every marker location, the gamma for a
c specific marker will NOT change since the markers are moving
c along a streamline.
         do 92 j = 1,nbr
            gam(j) = seval(bs(j),gb,gbbtbs,tbs,num4)
            write(64,*) gam(j)
 92      continue
         close(unit=64)


c ******************plotting*********************
c call plotting routine for gamma vs marker position
       nper(1) = nbr
       titl(1) = 'gam vs. bs'
       pltitl = '~bs ~gam ~'
       call grinit(5,6,'gam vs bs')
       call grklin(ilin,isym,nper,titl,1,bs,gam,pltitl,599)
c ****************end plotting*******************

c now for location of vortex core.
c the location for the core for typical hover and elliptical
c distribution is nearly identical, so approximated as
c that here.

         RCORE  = .9952
         ZCORE  = -.007684


c ******************plotting*********************
c call plotting routine for initial wake geometry
       nvortx = numbr + 1
       br(nvortx) = .9952
       bz(nvortx) = -.007684
       nper(1) = nbr
       nper(2) = 1
       titl(1) = 'init wake'
       titl(2) = 'vortex pos'
       pltitl = '~br ~bz ~'
       call grinit(5,6,'bz vs br ')
       call grklin(ilin,isym,nper,titl,2,br,bz,pltitl,599)
c ****************end plotting*******************
```

```
150     continue


      nvortx = numbr + 1
      NBR = NUMBR



c VORTEX CORE STRENGTH

         GAMCOR = -1.342*(a**(4./3.))*(t**(1./3.))

c VORTEX CORE RADIUS

         CORAD  = .058*((a*t)**(2./3.))

         IF(NSTOP.EQ.99) goto 54


c there will be 2 outer main loops in this program where
c ii => loop 15 outer time loop
c jj => loop 2007 outer loop for predictor corrector
c now loops for axial velocity
c j  => loop 12 which goes through marker stations (for vz component)
c k  => loop 13 which goes through integration strips (for vz compont)
c iii=> loop 14 which goes through azimuthal stations (for vz compont)
c now loops for radial velocity
c j  => loop 121 which goes through marker stations (for vr component)
c k  => loop 131 which goes through integration strips (for vr compont)
c iii=> loop 141 which goes through azimuthal stations (for vr compont)


       do 15 ii=1,45

 54       CONTINUE


          DO 2007 JJ = 1,2

          write(*,*) ii

c ALREADY FOUND BS'S IN PULLIN SUBROUTINE WHEN II = 1 SO SKIP TO
C 25 CONTINUE
          IF (II.EQ.1.and.jj.eq.1) GO TO 25

           BS(1) = BR(1)

c THIS NBR NOT NVORTEX CAUSE VORTEX DOESNT HAVE BS, NOT PART OF SHEET
c finding bs locations of markers by using pythag. theorem on br and bz
      do 10 i = 2,nbr

        delrsq(i) = (br(i) - br(i-1))**2.
```

```
        delzsq(i) = (bz(i) - bz(i-1))**2.
        dels(i)   = sqrt(delrsq(i) + delzsq(i))
        bs(i) = dels(i) + bs(i-1)
 10    continue
       BSMAX = BS(NBR)

 25    continue



c smooting routine which locally smoothes points
       call smooth(br,bz,bs,nbr,bsmax,ii,SMINC)

       write(*,1053) bsmax
 1053  format(1x,'bsmax after smooth=',f10.4)


c can spline gam and bs cause gam at br = gam at bs originally
c and gamma stays same at any marker location since
c moving along streamline. (marker locations are tracked by bs.)


       call splind(gam,gbs,bs,nbr,999.,999.)
       call splind(br,rbs,bs,nbr,999.,999.)
       call splind(bz,zbs,bs,nbr,999.,999.)
       call splind(bz,zbr,br,nbr,999.,999.)


       if(jj.eq.1) then
          time = delta
       endif

       nvortx = nbr + 1

       self = 0.0

c if running translating disk case no vortex core
       if(ncase.eq.1) then
          nmark = nbr
          else
             nmark = nvortx
             endif

       do 12 j = 1,nmark

          if(j.eq.nvortx) then
             br(j) = rcore
             bz(j) = zcore
             dgambs(j) = gamcor
             go to 160
          endif

c if not at vortex core get the deriv of gamma wrt bs through
c spline of gamma vs bs.
```

```
         dgambs(j) = deval(bs(j),gam,gbs,bs,nbr)

c*******lay-out of filaments and definition of danger zone in fildis

         CALL FILDIs(bs,bsmax,numbr,expls,xaxis,yaxis,npoin,klm,j,n6r,
     &                bslow,bshigh,dan,amark,vardint)


 160      continue

         singf3 = 0.
         f3ext = 0.0
         fhfunc = 0.
         HP = 0.
         klm = 0
         lkm = 0
         MARK = 0
         NPT = 0
         nflag = 0
         analt1 = 0.

c n6r due to expls(K+1)
c n6r is number of filaments along sheet
c nvor is number of filaments along sheet plus vortex core filament.
         nvor = n6r+1

c For translating disk case do not have vortex core (which is a
c filament and a marker for other cases) therefore no last filament.

         if(ncase.eq.1) then
            nfilmt = n6r
            else
               nfilmt = nvor
               endif

         DO 13 K = 1,nfilmt

            SING = 0.0
            ALT = 0.0
            f3dan = 0.0
            f3else = 0.0

c when bs = vortex core and ls = vortex core
            if(k.eq.nvor.and.j.eq.nvortx) go to 185

c when ls = vortex core need to give location and strength of filament
            if(k.eq.nvor) then

               lr(k) = rcore
               lz(k) = zcore
               hls = 1.0
               dgamls(k) = gamcor

             else
```

```
C TAKE MIDPOINT TO OBTAIN LS(K) (WHERE FILAMENTS ARE
C LOCATED), AND USE MIDPOINT METHOD FOR INTEGRATION ALONG SHEET.

              LS(K) = (EXPLS(K) + EXPLS(K+1))/2.
              HLS = EXPLS(K+1) - EXPLS(K)
              dgamls(k) = deval(ls(k),gam,gbs,bs,nbr)
              lr(k) = seval(ls(k),br,rbs,bs,nbr)
              lz(k) = seval(ls(k),bz,zbs,bs,nbr)

            endif

            nn = 72
            phil = -pi
            phiu = pi

            fh = 0.0
            hphi = (phiu - phil)/float(nn)

c loop for integration about azimuthal angle.

         do 14 iii = 1,nn

c x = azimuthal angle (psi)

            x(iii) = phil + (float(iii) - .5)*hphi

            onetop = br(j)*cos(x(iii)) - lr(k)
            pt1 = br(j)**2 + lr(k)**2
            pt2 = -2.0*br(j)*lr(k)*cos(x(iii)) + (bz(j) - lz(k))**2.
            onebot = (pt1 + pt2)**1.5
            twotop = br(j) - lr(k) - (.5*br(j)*(x(iii)**2.))
            pt1a = (br(j) - lr(k))**2. + br(j)*lr(k)*(x(iii)**2)
            pt2a = (bz(j) - lz(k))**2
            twobot = (pt1a + pt2a)**1.5


c  this is the integral f-h where h has a bar over it.
            fhbar = onetop/onebot - twotop/twobot
            fh = fh + fhbar

 14      continue

         fminh = hphi * fh

c now add in the analytical part that comes from hbar
c that was integrated analytically over psi (eqtn 3.21- not
c including third term which must be handled in linear zone).

         a = (br(j) - lr(k))**2. + (bz(j) - lz(k))**2
         f1 = (pi/lr(k))*(1./((a+br(j)*lr(k)*(-pi)**2)**.5))
         f2a = 1./(2*lr(k)*(br(j)*lr(k))**.5)
         f2b = alog(pi + ((pi)**2 + a/(br(j)*lr(k)))**.5)
         f2 = f2a*f2b
```

```
        func = f1 - f2

        FINFHF = (LR(K)*DGAMLS(K)*(FMINH + FUNC))



c now add on eqtns. 3.18 and eqtns. 3.22 and 3.23 when in linear zone
c and their complimentary parts when not in linear zone.  These
c equations were integrated analytically over phi but not s.

c bs and ls not defined when j = nvortx, and should never enter
c danger area when j = nvortx since vortex core is not along sheet.
        if(k.eq.nvor.or.j.eq.nvortx) goto 167


           bslow  = bs(j) - dan

           if(j.eq.numbr) then
              bshigh = bs(j)
           else
              bshigh = bs(j) + dan
           endif

           if(ls(k).gt.bslow.and.ls(k).lt.bshigh) then



c ****these lines are for plotting purposes and for two analytical
c ****integrals over s.

              klm = klm + 1

              IF (KLM.EQ.1) THEN
                 NPT = KLM + LKM
                 ELSE
                    NPT = NPT + 1
                    ENDIF

                    MARK = 1

c if nflag is never set to one, then you are never inside danger zone
c and you skip over analytical integrals over s.

              nflag = 1

              lsdan(klm) = ls(k)

c **********end lines pertaining to above  ** comment*****************

c br(nvortx) and bz(nvortx) will not go into deval cause nbr specified.

              k1 = deval(bs(j),br,rbs,bs,nbr)
              k2 = deval(bs(j),bz,zbs,bs,nbr)
```

```
               a1 = (pi**2)*(br(j)**2)
               b1 = -(pi**2)*br(j)*k1
               c1 = k1**2 + k2**2

c c1 should always be one (sometimes came out to be .9999999, so just
c made 1.0).

               if(c1.ne.1) then
                  c1 = 1
               endif


               bsls = bs(j) - ls(k)

c ***the additional terms that were integrated analytically wrt s, for
c ***equation 3.18, 3.22 and 3.23, are found after 13 continue.
c ***They are analt1 for 3.18 and f3ext for 3.22 and 3.23.


c equation 3.18
               chktop = k1*2*pi*dgamls(k)*(br(j)-k1*bsls)

               chkbot = c1*bsls*sqrt(a1+b1*bsls+c1*(bsls)**2)

               chktpa = (-2.*k1*dgambs(j))/(c1*bsls)

               ALT = (chktop*bsls-2*k1*dgambs(j)*chkbot)/(bsls*chkbot)


c for equation 3.22

      DANCHK = (K1*BSLS)/BR(J)

      IF (DANCHK.GT..01) THEN

      f3dan = (dgamls(k)/(2*(br(j)*lr(k))**.5))*
     &alog(abs((-pi+ sqrt((-pi)**2+(bsls**2/(br(j)**2-br(j)*k1*bsls))))
     &/((bsls**2)/(2.*(br(j)**2)*pi))))

      ELSE

c equation 3.23

      f3dan =(dgamls(k)/(2*(br(j)*lr(k))**.5))*
     &alog(abs(1/(1-(K1*BSLS/BR(J)))))

      ENDIF


c now if not in linear zone do "else" for ALT and f3dan.

            else
```

```
 167          continue

c ***lines for plotting
              lkm = lkm + 1

              IF (MARK.NE.1) THEN
                 NPT = LKM
                 ELSE
                    NPT = NPT + 1
                    ENDIF
c ***end lines for plotting

c equation 3.17
              topa = lr(k)*dgamls(k)*2.*pi*(br(j) - lr(k))

              a = (br(j) - lr(k))**2 + (bz(j) - lz(k))**2

              bota = a*(a + br(j)*lr(k)*((pi)**2.))**.5

              SING = TOPA/BOTA


c third term in 3.21

        f3el1 = (1./(2*lr(k)*(br(j)*lr(k))**.5))
        f3el2 = alog(abs((-pi + ((-pi)**2 + a/(br(j)*lr(k)))**.5)))
        f3else = (lr(k)*dgamls(k))*f3el1*f3el2


            endif

            FHFUNC = FHFUNC + HLS*FINFHF

            singf3 = singf3 + hls*(f3dan + f3else)


            HP = HP + HLS*(SING+ALT)

c 13 is loop over integration strips
 13         continue



      npoin = lkm + klm


c if never entered linear zone, then do not do analytical integrals
c that go with danger zone.

      if(nflag.eq.0) go to 878

c analytical integral coming from linear zone for equation 3.18
      ex = bs(j) - lsdan(klm)
```

```
      ex1 = bs(j)-lsdan(1)

      integ1 = 2*k1*dgambs(j)*alog(abs(ex1/ex))

      analt1 = integ1

C ANALYTICAL INTEGRAL COMING FROM LINEAR ZONE FOR 3.22 AND 3.23

      F3EXT1 = -2*((BS(J) - LSDAN(KLM))*ALOG(ABS(BS(J)-LSDAN(KLM)))
     &+ LSDAN(KLM)*(ALOG(ABS((SQRT(2.*PI))*BR(J))) + 1))

      F3EXT2 = 2*((BS(J)-LSDAN(1))*ALOG(ABS(BS(J)-LSDAN(1)))
     &+ LSDAN(1)*(ALOG(ABS((SQRT(2.*PI))*BR(J))) + 1))


      F3EXT = (dgambs(j)/(2*(br(j)*br(j))**.5))*(F3EXT1 + F3EXT2)

 878         continue
 185         continue

c for the induced velocity in z direction

      if (j.eq.nvortx) then
C include self induced velocity of vortex core on itself.

C GAMCOR NEEDS A NEGATIVE SO IT CONVECTS DOWN BY MAKING "SELF"
C POSITIVE.  GAMCOR BY ITSELF IS A NEGATIVE NUMBER.

c self is velocity core induces on itself.
          self = (-gamcor/(4.*pi*br(j)))*(alog(abs((8*br(j))/
     &            (corad) - (1./4.))))
          vindz(j) = ((hp + fhfunc + analt1)/(4*pi)) + self

      else

          vindz(j) = (hp + fhfunc + analt1 + singf3 + f3ext)/(4.*pi)

      endif


c 12 is loop which goes through markers
 12       continue




c ***************NOW FOR RADIAL VELOCITY COMPONENT*****************

c start loops for integration for radial component


      nvortx = nbr + 1
```

```
c for translating disk no vortex core, therefore one less marker.
      if(ncase.eq.1) then
         nmark = nbr
         else
            nmark = nvortx
            endif

c loop through markers
      do 121 j = 1,nmark


c define location and strength of core
         if(j.eq.nvortx) then
            br(j) = rcore
            bz(j) = zcore
            dgambs(j) = gamcor
            go to 175
         endif

         dgambs(j) = deval(bs(j),gam,gbs,bs,nbr)



c use integration strip distribution from bsmax for integration strip
c distribution for vortex core.
         call fildis(bs,bsmax,numbr,expls,xaxis,yaxis,npoin,klm,j,n6r,
     &               bslow,bshigh,dan,amark,vardint)


 175      continue

          TP5EX = 0.0
          SNGTP5 = 0.0
          FKTOT  = 0.0
          SUMKDP = 0.0
          klma   = 0
          lkma = 0
          mark1 = 0
          npt = 0
          analt2 = 0.
          mflag  = 0
          e = 0.

c add extra filament to include vortex core
c vortex core is NOT part of sheet
          nvor = n6r + 1

c no vortex core for case 1, translating disk
          if(ncase.eq.1) then
             nfilmt = n6r
             else
                nfilmt = nvor
                endif
```

```
c now integrate along sheet by using Midpoint Method, visiting each
c integration interval.
         do 131 k =1,nfilmt

            ALTA = 0.0
            TP5DAN = 0.0
            TP5ELS = 0.0


            KDPHI = 0.0

c when j = nvortx,  and k = nvor
            if (j.eq.nvortx.and.k.eq.nvor) goto 186

c when k = nvor (at vortex core) define location and strength
            if (k.eq.nvor) then
               lr(k) = rcore
               lz(k) = zcore
               dgamls(k) = gamcor
               hls = 1.0

            else

c must redefine ls for radial velocity because ls distribution is
c dependent on each marker location, and therefore gets overwritten
c each time.
               ls(k) = (expls(k) + expls(k+1))/2.
               hls = expls(k+1)-expls(k)
               dgamls(k) = deval(ls(k),gam,gbs,bs,nbr)
               lr(k) = seval(ls(k),br,rbs,bs,nbr)
               lz(k) = seval(ls(k),bz,zbs,bs,nbr)

            endif

            delz = bz(j)-lz(k)
            PHIL = -PI
            PHIU = PI
            NN = 72
            fk = 0.0
c hphi is distance between integration strips
            hphi = (phiu - phil)/float(nn)

      do 141 iii = 1,nn

         x(iii) = phil + (float(iii) - .5)*hphi

c now do the integration over j-kbar

         top1 = cos(x(iii))*delz
         bot1 = (br(j)**2 + lr(k)**2-2.*br(j)*lr(k)*
     &           cos(x(iii)) + delz**2)**1.5
         top2 = delz - (delz*x(iii)**2)/2.
         bot2 = ((br(j)-lr(k))**2+br(j)*lr(k)*x(iii)**2 +
```

```
     &          delz**2)**1.5


          fkbar = top1/bot1 - top2/bot2
          fk = fk + fkbar


 141      continue


          fkint = hphi*fk


c now add in the analytical part that comes from kbar
c that was integrated analytically over psi (equation 3.35 -
c not including third term which is handled later).

          dum = (br(j) - lr(k))**2 + delz**2
          top3 = delz*pi
          bot3 = br(j)*lr(k)*(dum+br(j)*lr(k)*(pi)**2)**.5
          top4 = delz*(alog(pi+sqrt((pi)**2 + dum/(br(j)*lr(k)))))
          bot4 = 2.*br(j)*lr(k)*(br(j)*lr(k))**.5

          analkb = top3/bot3 - top4/bot4

          fkdone = (lr(k)*dgamls(k)*(fkint +
     &             analkb))


c now add on 3.31, and third term in 3.35 that were integrated
c analytically over phi, but contained singularities when integrated
c over s, and therefore needed to be dealt with in the linear zone.
c Bs not defined when j=nvortx; and should never be in danger zone
c when at core location since it does not lie along sheet.

          if (k.eq.nvor.or.j.eq.nvortx) goto 164

             bslowa = bs(j) - dan
             bshgha = bs(j) + dan

          if(ls(k).gt.bslowa.and.ls(k).lt.bshgha) then

c ***these lines are for plotting and analytical integration that
c ***is necessary for 3.31, and third term in 3.35 when integrated
c ***over linear zone.
                klma = klma + 1

                IF (KLMA.EQ.1) THEN
                   NPT1 = KLMA + LKMA
                   ELSE
                      NPT1 = NPT1 + 1
                      ENDIF

                      MARK1 = 1
```

```
                mflag = 1
c ***
                lsdana(klma) = ls(k)

                k1 = deval(bs(j),br,rbs,bs,nbr)
                k2 = deval(bs(j),bz,zbs,bs,nbr)

                a1 = (pi**2)*(br(j)**2)
                b1 = -(pi**2)*br(j)*k1
                c1 = k1**2 + k2**2

                if(c1.ne.1) then
                   c1 = 1
                endif

                bsls = bs(j) - ls(k)

c the additional analytically integrated terms for 3.31 and the third
c term in 3.35 are found following 131 continue.

c equation 3.31
                cktop = k2*2*pi*dgamls(k)*(br(j)-k1*bsls)

                ckbot = c1*bsls*sqrt(a1+b1*bsls+c1*(bsls)**2)

                ALTA = (cktop*bsls-2.*k2*dgambs(j)*ckbot)/(bsls*ckbot)

c third term in 3.35, after it is handled like third term in equation
c 3.21 for axial velocity.

       tp5chk = (k1*bsls)/br(j)
       if (tp5chk.gt..01) then

       tp5dan = ((bz(j)-lz(k))/br(j))*
      &(dgamls(k)/(2*(br(j)*lr(k))**.5))*
      &alog(abs((-pi+ sqrt((-pi)**2+(bsls**2/(br(j)**2-br(j)*k1*bsls))))
      & /((bsls**2)/(2.*(br(j)**2)*pi))))

       else

c when ls(k) really close to bs(j) use singularity fix plus additional
c expansion about 1.

       tp5dan = ((bz(j)-lz(k))/br(j))*
      &(dgamls(k)/(2*(br(j)*lr(k))**.5))*
      &alog(abs(1/(1-(K1*BSLS/BR(J)))))

       endif

             else

 164           continue
```

```
c ***lines for plotting
               lkma = lkma + 1

               IF (MARK1.NE.1) THEN
                  NPT1 = LKMA
                  ELSE
                     NPT1 = NPT1 + 1
                     ENDIF
c ***end lines for plotting

c equation 3.31 when not in linear zone
               top6 = lr(k)*dgamls(k)*2*pi*delz

               bot6 = dum*(dum+br(j)*lr(k)*((pi)**2))**.5

               kdphi = top6/bot6

c third term in 3.35 when not in linear zone
               TP5EL1 = ((BZ(J)-LZ(K))/BR(J))*
     &                    (1./(2*lr(k)*(br(j)*lr(k))**.5))
               TP5EL2 =
     &         alog(abs((-pi + ((-pi)**2 + dum/(br(j)*lr(k)))**.5)))
               TP5ELS = LR(K)*DGAMLS(K)*tp5el1*tp5el2

            endif


c now doing midpoint method numerical scheme
               fktot = fktot + hls*fkdone

               SNGTP5 = SNGTP5 + HLS*(TP5DAN + TP5ELS)


               sumkdp = sumkdp + hls*(kdphi+alta)


 131        continue


c summing up integration intervals inside and outside of linear zone.
            npoin1 = lkma + klma

c if mflag = 0 never entered linear zone and do not do analytical
c integration that comes from this zone.
               if(mflag.eq.0) go to 879

c now for analytical terms coming from linear zone for equation 3.31
               ex2 = bs(j) - lsdana(klma)
               ex3 = bs(j)-lsdana(1)
               integ2 = 2.*k2*dgambs(j)*alog(abs(ex3/ex2))
               analt2 = integ2
```

```
c analytical term stemming from third term in 3.35.  Integral
c approximated as zero due to symmetry of graph of equation about
c marker location (graph not perfectly symmetrical
c that's why this is an approximation).
              TP5EX = 0.0

 879          continue
 186          continue


c for the induced velocity in the r direction
c negative sign in next two cases is for change in coordinate

             vindr(j) = -(sumkdp+fktot+analt2+SNGTP5+TP5EX)/(4.*pi)



 121       continue


c PREDICTOR-CORRECTOR TIME STEP
            if(ncase.eq.1) then
               nmark = nbr
               else
                  nmark = nvortx
                  endif
            IF(JJ.EQ.1) THEN
               DO 911 J=1,NMARK
                  VZOLD(J) = VINDZ(J)
                  VROLD(J) = VINDR(J)
 911              CONTINUE
                  ENDIF

                  IF(JJ.EQ.2) THEN
                     DO 912 J=1,NMARK
                        VINDZ(J)= (VZOLD(J) + VINDZ(J))/2.
                        VINDR(J)= (VROLD(J) + VINDR(J))/2.
 912                    CONTINUE
                        ENDIF
C END ADDITION FOR PREDICTOR-CORRECTOR

c now for the distance the particle is moved due to velocity in
c radial direction.
            do 909 j = 1,nmark

               IF(JJ.EQ.1) THEN
                  OLDCRE = RCORE
                  brold(j) = br(j)
                  ENDIF

               br(j) = vindr(j) * time
 909           continue
```

```
c now for the distance the particle is moved due to velocity in
c axial direction.

            DO 999 J = 1,nmark

               IF(JJ.EQ.1) THEN
                  OLDCZE = ZCORE
                  bzold(j) = bz(j)
                  ENDIF

               bz(j) = vindz(j)*time
 999        CONTINUE


C NOW WRITING RESULTS TO FILE INCASE CODE NEEDS TO BE STOPPED AND
C STARTED.  ONLY WRITE WHEN JJ=2, CAUSE THAT'S THE SECOND ITERATION
C FOR PREDICTOR- CORRECTOR WHICH IS WHAT IS USED ON NEXT II ITERATION
C FOR LOOP 15.

            if(ii.eq.1.and.jj.eq.2) then
               open(unit=62, file='wakegeom.out',status='unknown')
            endif

            if(ii.ne.1.and.jj.eq.2) then
               open(unit=62, file='wakegeom.out',status='unknown',
     &              access='append')
            endif

            if(jj.eq.2) then
            open(unit=63, file='wakestore.dat',status='unknown')
            open(unit=65, file='miscstore.dat',status='unknown')
            endif

            if(jj.eq.2) then
            write(62,*) ii
            endif

C END SAVING RESULTS

c     PUTTING THE MARKERS AT ITS NEW LOCATION

             do 957 i = 1,nbr
                br(i) = br(i) + brold(i)
                bz(i) = bz(i) + bzold(i)
            if(jj.eq.2) then
                write(62,*) br(i),bz(i)
                write(63,*) br(i),bz(i)
            endif
 957         continue

c     PUTTING THE CORE AT ITS NEW LOCATION

             rcore = br(nvortx) + OLDCRE
             zcore = bz(nvortx) + OLDCZE
```

```
            br(nvortx) = rcore
            bz(nvortx) = zcore
C SAVING RESULTS AGAIN FOR STOP/START ABILITY
           if(jj.eq.2) then
            write(62,*) br(nvortx),bz(nvortx)
            close(unit=62)
            write(63,*) br(nvortx),bz(nvortx)
            write(65,*) numbr,time,ii,rcore,zcore
            close(unit=63)
            close(unit=65)
           endif
C END SAVING RESULTS


C *********USE THIS PLOTTING FOR DISK CASE BECAUSE
C *********WANT TO SEE RESULTS ON FIRST ITERATION
C *********OF PREDICTOR-CORRECTOR.  AT SECOND
C *********ITERATION WILL SCREW-UP DUE TO SPIKES
C *********AT TIP.

c *****************plotting*********************
c call plotting routine for vindz vs. bz
c       vindz(nbr) = 0.
       nper(1) = nmark
       titl(1) = 'veloc'
       pltitl = '~br ~vindz ~'
       call grinit(5,6,'vindz vs br ')
c       call grklin(ilin,isym,nper,titl,1,br,vindz,pltitl,599)
c************************************************
c *****************plotting*********************
c call plotting routine for vindr vs. br
c       vindz(nbr) = 0.
       nper(1) = nmark
       titl(1) = 'veloc'
       pltitl = '~br ~vindr ~'
       call grinit(5,6,'vindr vs br ')
c       call grklin(ilin,isym,nper,titl,1,br,vindr,pltitl,599)
c************************************************

C 2007 LOOP FOR SECOND ITERATION OF PREDICTOR CORRECTOR
 2007        CONTINUE



C WAKE GEOMETRY PLOT AT EACH TIME STEP AFTER CONVERGED
C USING PREDICTOR-CORRECTOR.
c *****************plotting*********************
c call plotting routine for br vs. bz
       nper(1) = nbr
       nper(2) = 1
       titl(1) = 'wake disp'
       titl(2) = 'vortex pos'
       pltitl = '~br ~bz ~'
```

```
       call grinit(5,6,'bz vs br ')
c       call grklin(ilin,isym,nper,titl,2,br,bz,pltitl,599)
c ****************end plotting******************


c       15 is outer time loop
 15         continue

            go to 995

 676        write(*,677)
 677        format(1x,'e equals zero!!')
 678        write(*,679)
 679        format(1x,'q equals zero!!')
 680        write(*,681)
 681        format(1x,'q is positive !!')
 682        write(*,683)
 683        format(1x,'c is negative !!')

 995        continue

            stop
            end
```

```
      FUNCTION DEVAL(SS,X,XS,S,N)
      DIMENSION X(1), XS(1), S(1)
C-------------------------------------------------
C     Calculates dX/dS(SS)                        |
C     XS array must have been calculated by SPLINE |
C     WRITTEN BY PROFESSOR MARK DRELA, M.I.T.     |
C-------------------------------------------------
      ILOW = 1
      I = N
C
   10 IF(I-ILOW .LE. 1) GO TO 11
C
      IMID = (I+ILOW)/2
      IF(SS .LT. S(IMID)) THEN
       I = IMID
      ELSE
       ILOW = IMID
      ENDIF
      GO TO 10
C
   11 DS = S(I) - S(I-1)
      T = (SS - S(I-1)) / DS
```

```
      CX1 = DS*XS(I-1) - X(I) + X(I-1)
      CX2 = DS*XS(I)   - X(I) + X(I-1)
      DEVAL = X(I) - X(I-1) + (1.-4.0*T+3.0*T*T)*CX1 + T*(3.0*T-2.)*CX2
      DEVAL = DEVAL/DS
      RETURN
      END ! DEVAL
      SUBROUTINE DISK(BR,BZ,BS,BSMAX,GB,TBS,ndis1,NUMBR,a,t)



C!!!!!DONT FORGET TO DIMENSION VARIABLES IN MAIN PROGRAM

      DIMENSION DELRSQ(2001), TBS(2001), DELZSQ(2001), DELS(2001),
     *          GB(2001),     BR(2001),  BZ(2001),     BS(2001),
     *          gam(100),     R(2001),   Z(2001),      zbs(2000),
     *          gbbtbs(2001), rbs(2000), s(2000)

      DIMENSION NPER(2),ILIN(2),ISYM(2)
      DATA ILIN/1,2/
      DATA ISYM/1,2/
      CHARACTER * 10 TITL(2)
      CHARACTER * 30 PLTITL

      pi = 4.*atan(1.)

C     ********FOR STRAIGHT LINE DISTRIBUTION**********

      num = 150
      ntotal = num
      numbr = 150
      do 65 j = 2,ntotal
         r(1) = .0053
         r(ntotal) = sin((pi/2)*(149.5-1)/(num-1))
         r(j) = sin((pi/2)*(j-1)/(num-1))
         z(j) = 0.0
 65   continue

C     **********END STRAIGHT LINE DISTRIBUTION*********


C NOW TURN R AND Z VALUES INTO S VALUES.

         bs(1) = r(1)
         do 71 i = 2,ntotal
            delrsq(i) = (r(i) - r(i-1))**2
            delzsq(i) = (z(i)-z(i-1))**2
            dels(i) = sqrt(delrsq(i) + delzsq(i))
            bs(i) = dels(i) + bs(i-1)
 71      continue
         smax = bs(ntotal)

         write(*,87) bs(ntotal)
 87      format(1x,'smax=',f10.7)
```

```
        CALL SPLIND(Z,ZBS,BS,NTOTAL,999.,999.)
        CALL SPLIND(R,RBS,BS,NTOTAL,999.,999.)


c define br(1) = .01 cause otherwise setexp calls br(1) = 0.
c and setexp in vortex cannot deal with this.

        bs(1) = .01
        bsmax = bs(numbr)

        do 66 j=1,numbr

           br(j) = seval(bs(j),r,rbs,bs,ntotal)
           bz(j) = seval(bs(j),z,zbs,bs,ntotal)

 66     continue

C CHANGED NVORTX=NUMBR+1 TO NVORTX=NUMBR FOR CIRCULAR DISK
C SINCE THERE IS NO TIP VORTEX

        nvortx = numbr

        NPER(1) = numbr
        TITL(1) ='INIT WAKE'
        PLTITL = '~RADIUS ~Z/R ~'
        CALL GRINIT(5,6,'PLOT OF INITIAL DISK WAKE')
        CALL GRKLIN(ILIN,ISYM,NPER,TITL,1,BR,BZ,PLTITL,599)


        u = .02
        ndis1 = numbr
        do 27 i = 1,numbr
          tbs(i) = sin((pi/2)*(i-1)/(num-1))
          gb(i) = (-(4*u)/pi)*sqrt(1.-tbs(i)**2)
 27       continue


        NPER(1) = ndis1
        TITL(1) ='DISK CIRCULATION DISTRIBUTION'
        PLTITL = '~TBS ~GB ~'
        CALL GRINIT(5,6,'PLOT OF GB VS S')
c        CALL GRKLIN(ILIN,ISYM,NPER,TITL,1,tbs,gb,PLTITL,599)

        CALL SPLIND(GB,GBBTBS,tbs,Ndis1,999.,999.)

        do 92 j=1,numbr
           gam(j) = seval(bs(j),gb,gbbtbs,br,ndis1)
 92     continue

        NPER(1) = numbr
        TITL(1) ='DISK CIRCULATION DISTRIBUTION'
        PLTITL = '~BS ~GAM ~'
        CALL GRINIT(5,6,'PLOT OF GAM VS BS')
```

```
c          CALL GRKLIN(ILIN,ISYM,NPER,TITL,1,bs,gam,PLTITL,599)



         return
         end ! disk
c         stop
c         end
```

```
c The part of this  routine that uses Pullin circulation distribution
c works except since the points are hand-picked
c off a plot, waves in the wake soon come into play.  If
c I could curve fit the data for the gamma distribution
c from .95 out that would be a solution, but cant find
c appropriate curve fit.  Instead shifted ellipse over
c and up (using an equation) to try to closely
c represent pullin distribution.  Thisworks very well.

      SUBROUTINE ELLIP(BR,BZ,BS,BSMAX,GB,TBS,ndis1,NUMBR,a,t)

C!!!!!DONT FORGET TO DIMENSION VARIABLES IN MAIN CODE

      DIMENSION DELRSQ(2001), lampul(2001), DELZSQ(2001), gbbtbs(2001),
     *          DELS(2001),   GB(2001),     yplt(2001),   Z(2001),
     *          R(2001),      ebz(2000),    xplt(2000),   yyplt(2000),
     *          rbs(2000),    P(301),       PBS(301),     TBS(2001),
     *          zbs(2000),    s(2000),      ZETA(2000),   ETA(2000),
     *          BR(2001),     BZ(2001),     BS(2001)

      DIMENSION NPER(2),ILIN(2),ISYM(2)
      DATA ILIN/1,2/
      DATA ISYM/1,2/
      CHARACTER * 10 TITL(2)
      CHARACTER * 30 PLTITL

      real lampul

      NUM = 19
      ZETA(1) =  64.7997
```

```
      ETA (1)  =  0.0
      ZETA(2) =  45.0
      ETA (2)  =  0.0
      ZETA(3) =  25.9200
      ETA (3)  =  0.0
      ZETA(4)  = 6.48
      ETA (4)   = 0.0
      ZETA(5) =   3.2400
      ETA (5)  =  0.0
      ZETA(6) =   2.0600
      ETA (6)  =  0.0
      ZETA(7) =   1.7250
      ETA (7)  =   .0063
      ZETA(8) =   1.4438
      ETA (8)  =   .0125
      ZETA(9) =   1.0863
      ETA (9)  =   .0281
      ZETA(10) =    .8875
      ETA (10)  =   .0438
      ZETA(11) =   .7125
      ETA (11)  =  .0613
      ZETA(12) =   .5375
      ETA (12)  =  .08625
      ZETA(13) =   .3625
      ETA (13)  =  .1375
      ZETA(14) =   .2250
      ETA (14)  =  .2163
      ZETA(15) =   .1781
      ETA (15)  =  .2531
      ZETA(16) =   .1375
      ETA (16)  =  .3
      ZETA(17) =   .10625
      ETA (17)  =  .3625
      ZETA(18) = .09
      ETA(18) = .4125
      ZETA(19) = .0828
      ETA(19) = .475

      do 51 i = 1,num
            xplt(i) = zeta(i)
            yyplt(i) = eta(i)
 51   continue

c Now the zeta and eta values above (which are pulled directly
c from Pullin's plots) will be curve-fitted and overwritten
c so instabilities do not occur due to errors in transcribing
c Pullin's data.

      int1 = ((65-10)/5) + .1
      int2 = ((10-3)/.05) + .1
      int3 = ((3-1)/.01) + .1
      int4 = ((1-.083)/.001) + 1.1
      dec2 = int1 + int2
      dec3 = dec2 + int3
```

```
      ntotal = int1 + int2 + int3 + int4
      write(*,*) ntotal
      zet = 65
      ainc = 0.0
      do 61 i = 1,ntotal
         zet = zet - ainc
         zeta(i) = zet
         eta(i) = .04*((1/zeta(i)) - (1./64.7997))
         ind = num + i
         xplt(ind) = zeta(i)
         yyplt(ind) = eta(i)
         if(i.le.int1) then
            ainc = 5
         endif
         if(i.gt.int1.and.i.le.dec2) then
            ainc = .05
         endif
         if(i.gt.dec2.and.i.le.dec3) then
            ainc = .01
         endif
         if(i.gt.dec3) then
            ainc = .001
         endif
 61   continue


C PLOT WITH NEW CURVE FIT VALUES WHERE PULLIN ZETA AND ETA HAVE
C ALREADY BEEN OVERWRITTEN IN LOOP 61.
      NPER(1) = ntotal
      TITL(1) ='PULLIN'
      PLTITL = '~ZETA ~ETA ~'
      CALL GRINIT(5,6,'PLOT OF ETA VS ZETA')
c      CALL GRKLIN(ILIN,ISYM,NPER,TITL,1,zeta,eta,PLTITL,599)
c

      CALL SPLIND(ETA,EBZ,ZETA,Ntotal,999.,999.)


      A = .01396
      T = .1373
      DO 1 I = 1,NTOTAL

c since old coordinate system was zero at tip,
c 1.- ..... is necessary.

         R(I) = 1. - ((A*T)**(2./3.)*ZETA(I))

c  negative is because Pullin has z pos upward.

         Z(I) = -(A*T)**(2./3.)*ETA(I)

 1       CONTINUE
```

```
C NOW TURN R AND Z VALUES INTO S VALUES.

        s(1) = r(1)
        do 71 i = 2,ntotal
           delrsq(i) = (r(i) - r(i-1))**2
           delzsq(i) = (z(i)-z(i-1))**2
           dels(i) = sqrt(delrsq(i) + delzsq(i))
           s(i) = dels(i) + s(i-1)
 71     continue
        write(*,87) s(ntotal)
 87     format(1x,'smax=',f10.7)

        CALL SPLIND(Z,ZBS,S,NTOTAL,999.,999.)
        CALL SPLIND(R,RBS,S,NTOTAL,999.,999.)



c Setext lays-out the markers through an exponential
c stretching technique.

        DS1 = .025
        DSN = .0005
        NUMBR = 150
        SMAX = S(NTOTAL)
        CALL SETEX2(BS,DS1,DSN,SMAX,NUMBR)



c define br(1) = .01 cause otherwise setexp calls br(1) = 0.
c and setexp in vortex cannot deal with this.

        bs(1) = .01
        bsmax = bs(numbr)
        do 66 j=1,numbr

           br(j) = seval(bs(j),r,rbs,s,ntotal)
           bz(j) = seval(bs(j),z,zbs,s,ntotal)

 66     continue
        nvortx = numbr + 1
        br(nvortx) = .9952
        bz(nvortx) = -.007684

        NPER(1) = numbr
        nper(2) = 1
        TITL(1) ='INIT WAKE'
        TITL(2) ='CORE LOC'
        PLTITL = '~RADIUS ~Z/R ~'
        CALL GRINIT(5,6,'PLOT OF INITIAL PULLIN WAKE')
c        CALL GRKLIN(ILIN,ISYM,NPER,TITL,2,br,bz,PLTITL,599)
```

```
C
C NOW FOR THE CIRCULATION DISTRIBUTION THAT IS TAKEN DIRECTLY
C FROM PULLIN'S PLOT (ALL THE LINES COMMENTED OUT) .
C THIS IS NEVER USED, BUT OVERWRITTEN
C BY A SOLUTION THAT CLOSELY APPROXIMATES THIS SOLUTION IMMEDIATELY
C FOLLOWING ALL THE COMMENTS.
c         NUM2 = 15
c         TBS(1)  =  .9532
c         P  (1)  =  .3063
c         TBS(2)  =  .9585
c         P  (2)  =  .2878
c         TBS(3)  =  .9629
c         P  (3)  =  .2709
c         TBS(4)  =  .9684
c         P  (4)  =  .2532
c         TBS(5)  =  .9734
c         P  (5)  =  .2313
c         TBS(6)  =  .9776
c         P  (6)  =  .2101
c         TBS(7)  =  .9815
c         P  (7)  =  .1930
c         TBS(8)  =  .9847
c         P  (8)  =  .1775
c         TBS(9)  =  .9877
c         P  (9)  =  .1637
c         TBS(10)  =  .9903
c         P  (10)  =  .1539
c         TBS(11) =  .9931
c         P  (11) =  .1409
c         TBS(12) =  .9956
c         P  (12) =  .1311
c         TBS(13) =  .9977
c         P  (13) =  .1238
c         TBS(14) = .9998
c         P  (14) =  .1197
c         TBS(15) = 1.0018
c         P  (15) =  .1124

c         CALL SPLIND(P,PBS,TBS,NUM2,999.,999.)


c      DO 5 I = 1,NUM2

c         HA = .70741
c         HT = .00271
c         if(i.eq.num2) then
c         LAMPUL(I) = p(num2)/((HA**(4/3))*(HT**(1/3)))
c         else
c         endif
c          LAMPUL(I) = p(i)/((HA**(4/3))*(HT**(1/3)))

c          GB(I) = LAMPUL(I)*(A**(4/3))*(T**(1/3))
```

```
c
c          xplt(i) = tbs(i)
c          yplt(i) = gb(i)

c 5    CONTINUE

c         NPER(1) = NUM2
c         TITL(1) ='Hoej'
c         PLTITL = '~tbs ~gb ~'
c         CALL GRINIT(5,6,'for tbs gt .95')
c         CALL GRKLIN(ILIN,ISYM,NPER,TITL,1,TBS,gb,PLTITL,599)




c      AINC = .01
c      NDIS = .95/aINC + .01

c      nend = ndis + 1
c      NUM3 = Nend + 1
c      NUM4 = Nend + NUM2 - 1
c      DO 4 I = NUM3,NUM4


c             NUM5 = I-NDIS
c             GB(I) = GB(NUM5)
c             TBS(I) = TBS(NUM5)

c 4    CONTINUE

c          rad = -ainc
c          ind = nend
c      DO 6 I = 1,Nend
c           rad = rad + ainc
c           tbs(i) = rad
c           tbs(1) = 0.
c           tbs(nend) = .95
c           gb(i) = .02*((1.-tbs(i)**2))**.5
c           if(i.le.num2) then
c           ind = ind + 1
c           tbs(ind) = xplt(i)
c           gb(ind) = yplt(i)
c           endif

c 6     continue

c       npts = nend + num2
c         NPER(1) = Npts
c         TITL(1) ='actual'
c         PLTITL = '~TBS ~GB ~'
c         CALL GRINIT(5,6,'Pullin Circ')
c         CALL GRKLIN(ILIN,ISYM,NPER,TITL,1,TBS,GB,PLTITL,599)
C*********ONLY LEAVE NDIS1=NUM4, WHEN NOT USING SHIFTED ELLIPSE
```

```
c       ndis1 = num4

C*******************************************************

c**********************just use shifted ellipse*********************
c THIS VERY CLOSELY APPROXIMATES PULLIN'S SOLUTION.
         ainc = .01
         ndis = 1./ainc + 1.01
         rad = -ainc
         ndis1 = ndis + 1
      DO 27 I = 1,NDIS1
           rad = rad + ainc
           tbs(i) = rad
           tbs(1) = 0.
           tbs(ndis) = 1.0
           tbs(ndis1) = 1.0018
           gb(i) = .02*((1.-(tbs(i)-.01)**2))**.5 - (.02*
     &             sqrt(1.-(1.0018 - .01)**2)) + .0022

 27     continue

         CALL SPLIND(GB,GBBTBS,TBS,Ndis,999.,999.)
         NPER(1) = Ndis1
         TITL(1) = 'Approx Pullin'
         PLTITL = '~TBS ~GB ~'
         CALL GRINIT(5,6,'PLOT OF GB VS S')
         CALL GRKLIN(ILIN,ISYM,NPER,TITL,1,TBS,GB,PLTITL,599)
c********end shifted ellipse


         return
         end ! pullin
c         stop
c         end









      SUBROUTINE FILDIS(BS,bsmax,NUMBR,expls,XAXIS,YAXIS,NPOIN,klm,j,
     &                  n6r,bslow,bshigh,dan,amark,vardint)

c Melinda Godwin March 16,1991
```

```
      dimension s(2000),     expls(2000), yplt(2000),  temps(2000),
     &          bs(301),     xplt(2000),  xplt1(2000), yplt1(2000),
     &          xaxis(2000), xplt3(2000),
     &          yplt3(2000), yaxis(2000)

      dimension nper(2),ilin(2),isym(2)
      real ls
      data ilin/0,1/
      data isym/1,2/
      character * 20 titl(2)
      character * 25 pltitl


c ***set-up for linear zone********


          NFLAG = 0

c setting del (half linear region size)
         danbac = bs(j) - bs(j-1)
         danfor = bs(j+1) - bs(j)

         if (j.eq.1) then
            danbac = 1000
         endif


         if (j.eq.numbr) then
            danfor = 1000
         endif

         if (danbac.lt.danfor) then
               dan = danbac
         else
               dan = danfor
         endif



         nxy = numbr - 1
         if (j.eq.nxy) then
            dan = danfor
            nflag = 1
            endif


          bshigh = bs(j) + dan
          bslow =  bs(j) - dan

          if (j.eq.1) then
             bslow = bs(1)
             endif
          if (j.eq.numbr) then
```

```
            bshigh = bs(numbr)
            endif



          danzne = bshigh - bslow

c nspac = 5 gives 5 spacings in linear zone, therefore 4 fil

          if (j.eq.numbr.or.j.eq.1) then
c use only half the number of integration intervals plus one
c when the linear zone is one sided.
c This puts extra filament at end so linear
c area is treated like all other linear zones.
c
             nspac = 3
          else
             nspac = 6
          endif

          filspc = danzne/nspac



          IF (NFLAG.NE.1) THEN
             nfil = 5
          ELSE
c For second to last marker need an extra integration interval since
c there is no non-linear zone to right of this marker (there is only
c the linear zone).  This extra integration interval is necessary
c so later the midpoint can be taken between the second to last and
c last (now added) integration interval.
             NFIL = 6
          ENDIF

          if (j.eq.numbr.OR.J.EQ.1) then
             nfil = 3
          endif

c ****************end set-up for linear zone***********


c start set-up for how integration intervals are layed-out to the
c left of the linear zone.

      dint = .002

      if (j.eq.1) then

         nnl=0
         goto 299

         else
```

```
      nnl = (bslow/dint) + 1

      endif

      if (nnl.le.2) then
         nnl = 3
      endif


      if (nnl.le.3) then
         ds1 = .0005
         dsn = .0005
      else
         ds1 = .0005
         dsn = .001
      endif


      smax = bslow

      call setex2(s,ds1,dsn,smax,nnl)



      do 21 i = 1,nnl
         k = nnl-i+1
         expls(i) = s(i)
         xplt(i) = expls(i)
         yplt(i) = 0.
21       continue
      expls(1) = 0.
      xplt(1) = 0.



 299        continue

c end set-up for integration intervals to left of linear zone.

c begin set-up for integration intervals in linear zone.

      klm = 0
      nnew = nnl + 1
      nnew1 = nnl + nfil
      if (j.eq.1) then
         updat = bslow - filspc
         else
            updat = bslow
            endif


      do 23 i = nnew,nnew1
         klm = klm + 1
         updat = updat + filspc
```

```
         expls(i) = updat
         xplt1(klm) = expls(i)
         yplt1(klm) = 0.0
 23    continue

c end set-up for linear zone

c begin set-up for integration intervals to right of linear zone


      if (bshigh.ge.bsmax) then
         nnr = 0
         else
c The following is done to get more integration strips out near the
c tip for markers near the tip because of the high curvature in this
c area.  It was necessary for the translating disk case to keep
c accuracy high, but was too computationally time consuming to
c apply to the ellip distrib in full form (because of stretching),
c and not applied to the typical hovering case at all because in this
c case the sheet is stretched even more than for the elliptical case.
c It is important to note that as the sheet stretches bsmax increases
c and since this dictates the number of integration intervals the
c number of integration intervals increases as well (more integ.
c interv. means increase in computational time).
c The values given to amark and vardint can be found in main code.
            if(bs(j).gt.amark) then
               dint = vardint
               endif
            if(bs(j).le..9) then
                  dint = .002
                  endif
         nnr = (bsmax-bshigh)/dint + 1
      endif

      dint = .002

      if (nnr.le.3.and.nnr.ne.0) then
         nnr = 4
      endif

      n5r = nnr + nnl + nfil
      n6r = nnr + nnl + nfil - 1
      nlpd = nnl + nfil + 1


      if(nnr.eq.0) goto 160

      if (nnr.le.4) then
         ds1 = (bsmax - bshigh)/5
         dsn = ds1
      else
         ds1 = .001
         dsn = .001
      endif
```

```
      if(bs(j).gt..8) then
         ds1 = (bsmax - bshigh)/nnr
         dsn = ds1
         endif

      smax = bsmax-bshigh
      call setex2(s,ds1,dsn,smax,nnr)

      ind = nnl
      do 22 i = nlpd,n5r
         m = i-nlpd+1
         ind = ind + 1
         temps(i) = s(m)
         expls(i) = bshigh+temps(i)
         xplt(ind) = expls(i)
         yplt(ind) = 0.0
         xplt3(m) = expls(i)
         yplt3(m) = 0.0
22       continue
         expls(n5r) = bsmax


160      continue

c end set-up for integration intervals to right of linear zone.

c ***************************PLOTTING*****************************



         npoin = nnr + nnl + nfil
         do 150 i = 1,npoin

            if(i.le.klm) then
               xaxis(i) = xplt1(i)
               yaxis(i) = yplt1(i)
            else
               index = i - klm
               xaxis(i) = xplt(index)
               yaxis(i) = yplt(index)
            endif

150      continue

         if (bs(j).ge..995.and.bs(j).le..997) then
         nx = npoin - klm
         nper(1) = klm
         nper(2) = nx
         titl(1) = 'fildis'
         titl(2) = 'fildis'
         pltitl = '~expls ~0 ~'
         call grinit(5,6,'exp stretched ls')
```

```
c            call grklin(ilin,isym,nper,titl,2,xaxis,yaxis,pltitl,599)
            endif
c ***************************************************************


            return
            end ! fildis


C
      SUBROUTINE FILTER(T,Y,N,TFIL)
      PARAMETER (NX=1000)
      REAL T(N), Y(N), tfil(n)
      REAL AA(NX), BB(NX), CC(NX)
C
C     WRITTEN BY PROFESSOR MARK DRELA, MIT
C
      IF(N.GT.NX) STOP 'FILTER:  Array overflow'
C
C
      CC(1) = 0.
      AA(1) = 1.0
      DO 10 I=2, N-1
        TFSQ = TFIL(i)**2
        DTM = T(I) - T(I-1)
        DTP = T(I+1) - T(I)
        DT = 0.5*(T(I+1) - T(I-1))
        CC(I) = -TFSQ / (DTP*DT)
        AA(I) =  TFSQ * (1.0/DTP + 1.0/DTM)/DT  +  1.0
        BB(I) = -TFSQ / (DTM*DT)
   10 CONTINUE
      BB(N) = 0.
      AA(N) = 1.0
C
      CALL TRISOL(AA,BB,CC,Y,N)
C
      RETURN
      END ! FILTER

      SUBROUTINE SETEXP(S,DS1,SMAX,NN)
C.....................................................
C     Sets geometrically stretched array S:
C
C     WRITTEN BY PROFESSOR MARK DRELA, M.I.T.
C
C       S(i+1) - S(i)  =  r * [S(i) - S(i-1)]
C
C       S     (output)  array to be set
C       DS1   (input)   first S increment:  S(2) - S(1)
C       SMAX  (input)   final S value:      S(NN)
C       NN    (input)   number of points
C.....................................................
      REAL S(1)
C
```

```
      SIGMA = SMAX/DS1
      NEX = NN-1
      RNEX = FLOAT(NEX)
      RNI = 1.0/RNEX
C
C---- solve quadratic for initial geometric ratio guess
      AAA = RNEX*(RNEX-1.0)*(RNEX-2.0) / 6.0
      BBB = RNEX*(RNEX-1.0) / 2.0
      CCC = RNEX - SIGMA
C
      DISC = BBB**2 - 4.0*AAA*CCC
      DISC = AMAX1( 0.0 , DISC )
C
      IF(NEX.LE.1) THEN
       STOP 'SETEXP: Cannot fill array.  N too small.'
      ELSE IF(NEX.EQ.2) THEN
       RATIO = -CCC/BBB  +  1.0
      ELSE
       RATIO = (-BBB + SQRT(DISC))/(2.0*AAA)  +  1.0
      ENDIF
C
      IF(RATIO.EQ.1.0) GO TO 11
C
C---- Newton iteration for actual geometric ratio
      DO 1 ITER=1, 100
        SIGMAN = (RATIO**NEX - 1.0) / (RATIO - 1.0)
        RES = SIGMAN**RNI - SIGMA**RNI
        DRESDR = RNI*SIGMAN**RNI
     &         * (RNEX*RATIO**(NEX-1) - SIGMAN) / (RATIO**NEX - 1.0)
C
        DRATIO = -RES/DRESDR
        RATIO = RATIO + DRATIO
C
        IF(ABS(DRATIO) .LT. 1.0E-5) GO TO 11
C
    1 CONTINUE
      WRITE(6,*) 'SETEXP: Convergence failed.  Continuing anyway ...'
C
C---- set up stretched array using converged geometric ratio
   11 S(1) = 0.0
      DS = DS1
      DO 2 N=2, NN
        S(N) = S(N-1) + DS
        DS = DS*RATIO
    2 CONTINUE
C
      RETURN
      END ! SETEXP


      SUBROUTINE SETEX2(S,DS1,DSN,SMAX,N)
      DIMENSION S(N)
C.....................................................................
C     Sets array S stretched so that a prescribed spacing is
```

```
C     obtained at each end.  The interior spacing is a blend
C     of two geometric stretchings "shot" from each end.
C
C     WRITTEN BY PROFESSOR MARK DRELA, M.I.T.
C
C       S     (output)  array to be set
C       DS1   (input)   approximate first S increment:  S(2) - S(1)
C       DSN   (input)   approximate last  S increment:  S(N) - S(N-1)
C       SMAX  (input)   final S value:      S(N)
C       N     (input)   number of points
C.....................................................................
C
      PARAMETER (NDIM=2000)
      REAL S1(NDIM), SN(NDIM)
C
C---- with bigger FEND, the actual end increments will get closer to DS1 & DSN,
C     but the interior spacing might get screwy.
      DATA FEND / 0.5 /
C
      IF(N.GT.NDIM) STOP 'SETEX2:  Array overflow.'
C
C---- calculate spacing arrays each having the prescribed end increment
      CALL SETEXP(S1,DS1,SMAX,N)
      CALL SETEXP(SN,DSN,SMAX,N)
C
C---- blend spacing arrays with power-function weights
      DO 10 I=1, N
        IN = N-I+1
        SS1 = S1(I)
        SSN = SMAX - SN(IN)
C
C------ power function of integer index
        WT1 = FLOAT(N-I)**FEND
        WTN = FLOAT(I-1)**FEND
C
C------ power function of coordinate
CCC     WT1 = (1.0 - SSN/SMAX)**FEND
CCC     WTN = (      SS1/SMAX)**FEND
C
        S(I) = (SS1*WT1 + SSN*WTN) / (WT1 + WTN)
   10 CONTINUE
C
C---- check for monotonicity
      DO 20 I=2, N
        IF(S(I) .LE. S(I-1)) THEN
         WRITE(6,*) 'SETEX2: Warning. Returned array not monotonic.'
         RETURN
        ENDIF
   20 CONTINUE
C
      RETURN
      END ! SETEX2

      FUNCTION SEVAL(SS,X,XP,S,N)
```

```
      REAL X(1),XP(1),S(1)
C
C     WRITTEN BY PROFESSOR MARK DRELA, M.I.T.
C
      IF( (SS-S(N)) .GT. 0.1*(S(N)-S(N-1)) .OR.
     &    (SS-S(1)) .LT. 0.1*(S(1)-S(2)) ) then
c         write(*,172)
c 172     format('seval:outside range')
        endif
C
      ILOW = 1
      I = N
C
   10 IF(I-ILOW .LE. 1) GO TO 11
C
      IMID = (I+ILOW)/2
      IF(SS .LT. S(IMID)) THEN
       I = IMID
      ELSE
       ILOW = IMID
      ENDIF
      GO TO 10
C
   11 DS = S(I) - S(I-1)
      T = (SS-S(I-1)) / DS
      CX1 = DS*XP(I-1) - X(I) + X(I-1)
      CX2 = DS*XP(I)   - X(I) + X(I-1)
      SEVAL = T*X(I) + (1.0-T)*X(I-1) + (T-T*T)*((1.0-T)*CX1 - T*CX2)
      RETURN
      END ! SEVAL

      SUBROUTINE SMOOTH(BR,BZ,BS,NBR,bsmax,ii,sminc)



c MELINDA GODWIN
C MAY 23,1991

      DIMENSION BR(301),   BZ(301), BS(301), DELRSQ(301), DELZSQ(301),
     &          DELS(301), TFIL(2000)


      character * 12 titl(3)
      character * 16 pltitl


      write(*,877) sminc
 877  format(1x,'tfil=',f10.4)


        if(ii.eq.1) then

      write(*,47)
 47   format(1x,'INPUT CASE NUMBER, 1-DISK, 2-ELLIP, 3-TYP HOV ROTOR',
```

```
     &/,/)
      read(5,*) mcase

      write(*,48) mcase
48    format(1x,'YOU CHOSE CASE',1x,i2,1x,
     &           'ENTER 1 IF TRUE, 2 IF FALSE',/,/)
      read(5,*) nchoice

      if(nchoice.eq.2) then
      write(*,49)
49    format(1x,'INPUT CASE NUMBER, 1-DISK, 2-ELLIP, 3-TYP HOV ROTOR')
      read(5,*) mcase

      write(*,50) mcase
50    format(1x,'YOU ARE RUNNING CASE',1X,I2)
      endif

         endif



      nend = nbr-2

      do 2003 i = 2,nend

         delr1=br(i)-br(i-1)
         delz1=bz(i)-bz(i-1)
         delr2=br(i+1)-br(i)
         delz2=bz(i+1)-bz(i)
         delr3=br(i+2)-br(i+1)
         delz3=bz(i+2)-bz(i+1)
         cross1 = delr1*delz2 - delz1*delr2
         cross2 = delr2*delz3 - delz2*delr3

         flag=cross1*cross2

         if (mcase.eq.1.or.mcase.eq.2) then
              start = 0.0
            else
              start = .96
         endif


         if(flag.lt.0..and.br(i).gt.start) then

               tfil(i) = sminc
            else
               tfil(i)=0.0

         endif


         if(flag.lt.0.and.i.eq.nend) then
             if(sminc.eq.0.) then
```

```
              tfil(nend+1)=0.0
              tfil(nend+2)=0.0
            else
              tfil(nend+1)=sminc
              tfil(nend+2)=sminc
            endif
          endif

 2003 continue


       nolast = nbr

       call filter(bs,br,nolast,tfil)
       call filter(bs,bz,nolast,tfil)

      bs(1) = BR(1)
      do 18 i = 2,nbr
        delrsq(i) = (br(i) - br(i-1))**2.
        delzsq(i) = (bz(i) - bz(i-1))**2.
        dels(i)   = sqrt(delrsq(i) + delzsq(i))
        bs(i) = dels(i) + bs(i-1)
 18   continue
      BSMAX = BS(NBR)


       return
       end


      SUBROUTINE SPLIND(X,XS,S,N,XS1,XS2)
      DIMENSION X(1),XS(1),S(1)
      PARAMETER (NMAX=2000)
      DIMENSION A(NMAX),B(NMAX),C(NMAX)
C-----------------------------------------------------
C     WRITTEN BY PROFESSOR MARK DRELA, M.I.T.          |
C                                                      |
C     Calculates spline coefficients for X(S).         |
C     Specified 1st derivative and/or usual zero 2nd   |
C     derivative end conditions are used.              |
C     To evaluate the spline at some value of S,       |
C     use SEVAL and/or DEVAL.                          |
C                                                      |
C     S        independent variable array (input)      |
C     X        dependent variable array   (input)      |
C     XS       dX/dS array                (calculated) |
C     N        number of points           (input)      |
C     XS1,XS2  endpoint derivatives       (input)      |
C              If = 999.0, then usual zero second      |
C              derivative end condition(s) are used    |
C                                                      |
C-----------------------------------------------------
      IF(N.GT.NMAX) STOP 'SPLIND: array overflow, increase NMAX'
C
```

```
      DO 1 I=2, N-1
        DSM = S(I) - S(I-1)
        DSP = S(I+1) - S(I)
        B(I) = DSP
        A(I) = 2.0*(DSM+DSP)
        C(I) = DSM
        XS(I) = 3.0*((X(I+1)-X(I))*DSM/DSP + (X(I)-X(I-1))*DSP/DSM)
    1 CONTINUE
C
      IF(XS1.EQ.999.0) THEN
C----- set zero second derivative end condition
       A(1) = 2.0
       C(1) = 1.0
       XS(1) = 3.0*(X(2)-X(1)) / (S(2)-S(1))
      else if (xs1.eq.-999.) then
       A(1) = 1.0
       C(1) = 1.0
       XS(1) = 2.0*(X(2)-X(1)) / (S(2)-S(1))
      ELSE
C----- set specified first derivative end condition
       A(1) = 1.0
       C(1) = 0.
       XS(1) = XS1
      ENDIF
C
      IF(XS2.EQ.999.0) THEN
       B(N) = 1.0
       A(N) = 2.0
       XS(N) = 3.0*(X(N)-X(N-1)) / (S(N)-S(N-1))
      else if (xs2.eq.-999.0) then
       A(N) = 1.0
       C(N) = 1.0
       XS(N) = 2.0*(X(N)-X(N-1)) / (S(N)-S(N-1))
      ELSE
       A(N) = 1.0
       B(N) = 0.
       XS(N) = XS2
      ENDIF
C
C---- solve for derivative array XS
      CALL TRISOL(A,B,C,XS,N)
C
      RETURN
      END ! SPLIND

      SUBROUTINE TRISOL(A,B,C,D,KK)
      DIMENSION A(1),B(1),C(1),D(1)
C-----------------------------------------
C     Solves KK long, tri-diagonal system |
C                                         |
C             A C         D               |
C             B A C       D               |
C               B A .     .               |
C                 . . C   .               |
```

```
C                   B A    D              |
C                                         |
C     The righthand side D is replaced by |
C     the solution.  A, C are destroyed.  |
C-----------------------------------------
C
      DO 1 K=2, KK
        KM = K-1
        C(KM) = C(KM) / A(KM)
        D(KM) = D(KM) / A(KM)
        A(K) = A(K) - B(K)*C(KM)
        D(K) = D(K) - B(K)*D(KM)
    1 CONTINUE
C
      D(KK) = D(KK)/A(KK)
C
      DO 2 K=KK-1, 1, -1
        D(K) = D(K) - C(K)*D(K+1)
    2 CONTINUE
C
      RETURN
      END ! TRISOL
      SUBROUTINE TYPHOV(BR,BZ,BS,BSMAX,GB,TBS,ndis1,NUMBR,A,T)



c Uses Pullin self-similar solution with typical hover gamma distrib.

C!!!!!DONT FORGET TO DIMENSION VARIABLES IN MAIN PROGRAM

      DIMENSION DELRSQ(2001), lampul(2001), gbbtbs(2001), yyplt(2000),
     *          ZETA(2000),   ETA(2000),    BR(2001),     BZ(2001),
     *          P(301),       PBS(301),     TBS(2001),    BS(2001),
     *          DELZSQ(2001), DELS(2001),   GB(2001),     R(2001),
     *          Z(2001),      yplt(2001),   ebz(2000),    xplt(2000),
     *          rbs(2000),    tempbs(2000), tempgb(2000), tgbtbs(2000),
     *          zbs(2000),    s(2000)

      DIMENSION NPER(2),ILIN(2),ISYM(2)
      DATA ILIN/1,2/
      DATA ISYM/1,2/
      CHARACTER * 10 TITL(2)
      CHARACTER * 50 PLTITL

      real lampul

      NUM = 19
      ZETA(1) =  64.7997
      ETA (1)  =  0.0
      ZETA(2) =  45.0
      ETA (2)  =  0.0
      ZETA(3) =  25.9200
      ETA (3)  =  0.0
      ZETA(4)  = 6.48
```

```
      ETA (4)    = 0.0
      ZETA(5) =   3.2400
      ETA (5)  =  0.0
      ZETA(6) =   2.0600
      ETA (6)  =  0.0
      ZETA(7) =   1.7250
      ETA (7)  =   .0063
      ZETA(8) =   1.4438
      ETA (8)  =   .0125
      ZETA(9) =   1.0863
      ETA (9)  =   .0281
      ZETA(10) =    .8875
      ETA (10)  =   .0438
      ZETA(11) =   .7125
      ETA (11)  =  .0613
      ZETA(12) =   .5375
      ETA (12)  =  .08625
      ZETA(13) =   .3625
      ETA (13)  =  .1375
      ZETA(14) =   .2250
      ETA (14)  =  .2163
      ZETA(15) =   .1781
      ETA (15)  =  .2531
      ZETA(16) =   .1375
      ETA (16)  =  .3
      ZETA(17) =   .10625
      ETA (17)  =  .3625
      zeta(18) = .09
      eta(18) = .4125
      zeta(19) = .0828
      eta(19) = .475

      do 51 i = 1,num
            xplt(i) = zeta(i)
            yyplt(i) = eta(i)
 51   continue


c Now the zeta and eta values above (which are pulled directly
c from Pullin's plots) will be curve-fitted and overwritten
c so instabilities do not occur due to errors in transcribing
c Pullin's data.

      int1 = ((65-10)/5) + .1
      int2 = ((10-3)/.05) + .1
      int3 = ((3-1)/.01) + .1
      int4 = ((1-.083)/.001) + 1.1
      dec2 = int1 + int2
      dec3 = dec2 + int3
      ntotal = int1 + int2 + int3 + int4
      write(*,*) ntotal
      zet = 65
      ainc = 0.0
      do 61 i = 1,ntotal
```

```
         zet = zet - ainc
         zeta(i) = zet
         eta(i) = .04*((1/zeta(i)) - (1./64.7997))
         ind = num + i
         xplt(ind) = zeta(i)
         yyplt(ind) = eta(i)
         if(i.le.int1) then
            ainc = 5
         endif
         if(i.gt.int1.and.i.le.dec2) then
            ainc = .05
         endif
         if(i.gt.dec2.and.i.le.dec3) then
            ainc = .01
         endif
         if(i.gt.dec3) then
            ainc = .001
         endif
 61   continue


C PLOT WITH NEW CURVE FIT VALUES WHERE PULLIN ZETA AND ETA HAVE
C ALREADY BEEN OVERWRITTEN IN LOOP 61.
      NPER(1) = ntotal
      TITL(1) ='PULLIN'
      PLTITL = ''ZETA ~ETA ~'
      CALL GRINIT(5,6,'PLOT OF ETA VS ZETA')
c      CALL GRKLIN(ILIN,ISYM,NPER,TITL,1,zeta,eta,PLTITL,599)
c

      CALL SPLIND(ETA,EBZ,ZETA,Ntotal,999.,999.)


      A = .0418
      T = .0458
      DO 1 I = 1,NTOTAL

c since old coordinate system was zero at tip,
c 1.- ..... is necessary.
         R(I) = 1. - ((A*T)**(2./3.)*ZETA(I))


c  Negative is because Pullin has z pos upward.

         Z(I) = -(A*T)**(2./3.)*ETA(I)

 1       CONTINUE



C NOW TURN R AND Z VALUES INTO S VALUES.

         s(i) = r(1)
```

```
      do 71 i = 2,ntotal
         delrsq(i) = (r(i) - r(i-1))**2
         delzsq(i) = (z(i)-z(i-1))**2
         dels(i) = sqrt(delrsq(i) + delzsq(i))
         s(i) = dels(i) + s(i-1)
71    continue
      write(*,87) s(ntotal)
87    format(1x,'smax=',f10.7)

      CALL SPLIND(Z,ZBS,S,NTOTAL,999.,999.)
      CALL SPLIND(R,RBS,S,NTOTAL,999.,999.)


c Now s values will be stretched.

      DS1 = .025
      DSN = .0005
      NUMBR = 150
      SMAX = S(NTOTAL)
      CALL SETEX2(BS,DS1,DSN,SMAX,NUMBR)



c define br(1) = .01 cause otherwise setexp calls br(1) = 0.
c and setexp in vortex cannot deal with this.

      bs(1) = .01
      bsmax = bs(numbr)
      do 66 j=1,numbr

         br(j) = seval(bs(j),r,rbs,s,ntotal)
         bz(j) = seval(bs(j),z,zbs,s,ntotal)

66    continue

      nvortx = numbr + 1
      super = 2./3.
      scale = (a*t)**super
      br(nvortx) = 1.-.308*scale
      bz(nvortx) = -.498*scale

      NPER(1) = numbr
      nper(2) = 1
      TITL(1) ='INIT WAKE'
      TITL(2) ='CORE LOC'
      PLTITL = '~RADIUS ~Z/R ~'
      CALL GRINIT(5,6,'INITIAL WAKE FOR PRACTICAL DISTRIBUTION')
c      CALL GRKLIN(ILIN,ISYM,NPER,TITL,2,br,bz,PLTITL,599)



C
C NOW FOR THE CIRCULATION DISTRIBUTION EXACTLY FROM PULLIN'S
C SOLUTION.
```

```
      NUM2 = 15
      TBS(1)  =  .9532
      P  (1)  =  .3063
      TBS(2)  =  .9585
      P  (2)  =  .2878
      TBS(3)  =  .9629
      P  (3)  =  .2709
      TBS(4)  =  .9684
      P  (4)  =  .2532
      TBS(5)  =  .9734
      P  (5)  =  .2313
      TBS(6)  =  .9776
      P  (6)  =  .2101
      TBS(7)  =  .9815
      P  (7)  =  .1930
      TBS(8)  =  .9847
      P  (8)  =  .1775
      TBS(9)  =  .9877
      P  (9)  =  .1637
      TBS(10)  =  .9903
      P  (10)  =  .1539
      TBS(11) =  .9931
      P  (11) =  .1409
      TBS(12) =  .9956
      P  (12) =  .1311
      TBS(13) =  .9977
      P  (13) =  .1238
      TBS(14) = .9998
      P  (14) =  .1197
      TBS(15) = 1.0018
      P  (15) =  .1124


      CALL SPLIND(P,PBS,TBS,NUM2,999.,999.)


    DO 5 I = 1,NUM2

      HA = .70741
      HT = .00271

      LAMPUL(I) = p(i)/((HA**(4/3))*(HT**(1/3)))
      GB(I) = LAMPUL(I)*(A**(4/3))*(T**(1/3))
      xplt(i) = tbs(i)
      yplt(i) = gb(i)

5     CONTINUE

      NPER(1) = NUM2
      TITL(1) ='Hoej'
      PLTITL = '~tbs ~gb ~'
      CALL GRINIT(5,6,'for tbs gt .95')
c      CALL GRKLIN(ILIN,ISYM,NPER,TITL,1,TBS,gb,PLTITL,599)
```

```
          num99 = 9
c This is the part of the circulation distribution < .93, which
c was taken from plots referenced in thesis.  Pullin's solution
c is not applied to this.  It is only layed-out here so that
c the tip circulation before and after the curve fit can be plotted
c and compared.
          tempbs(1) = .1
          tempgb(1) = .0
          tempbs(2) = .2
          tempgb(2) = .0072
          tempbs(3) = .3
          tempgb(3) = .01
          tempbs(4) = .42
          tempgb(4) = .012
          tempbs(5) = .7
          tempgb(5) = .0152
          tempbs(6) = .84
          tempgb(6) = .018
          tempbs(7) = .915
          tempgb(7) = .0216
          tempbs(8) = .94
          tempgb(8) = .0208
          tempbs(9) = .95
          tempgb(9) = .0188


      CALL  SPLIND(TEMPGB,TGBTBS,TEMPBS,NUM99,999.,999.)

           ainc = .01
           disend = .94
           NDIS = disend/aINC + .01
           nend = ndis + 1
           rad = -ainc
           ind = nend

      DO 662 I = 1,Nend

           rad = rad + ainc
           tbs(i) = rad
           tbs(1) = 0.
           tbs(nend) = disend
           gb(i) = seval(tbs(i),tempgb,tgbtbs,tempbs,num99)

           if(i.le.num2) then
              ind = ind+1
              tbs(ind) = xplt(i)
              gb(ind) = yplt(i)
              endif

 662       continue
```

```
         npts = num2 + nend
         NPER(1) = Npts
         TITL(1) ='actual pullin solution'
         PLTITL = '~tbs ~gb ~'
         CALL GRINIT(5,6,'for tbs gt .95')
c         CALL GRKLIN(ILIN,ISYM,NPER,TITL,1,TBS,gb,PLTITL,599)

C END PULLIN'S EXACT SOLUTION

C NOW THE ABOVE CIRCULATION DISTRIBUTION IS APPROXIAMATED
C BY AN EQUATION AND OVERWRITTEN TO INSURE SMOOTHNESS IN THE
C CIRCULATION DISTRIBUTION.

         deltbs = .919
         num2 = 8
         do 785 i = 1,num2
            tbs(i) = deltbs + .01
            gb(i) = .022-2.95*(tbs(i)-.929)**2
            xplt(i) = tbs(i)
            yplt(i) = gb(i)
            deltbs = tbs(i)
 785        continue

         NPER(1) = num2
         TITL(1) ='approx soltn'
         PLTITL = '~tbs ~gb ~'
         CALL GRINIT(5,6,'for tbs gt .95')
         CALL GRKLIN(ILIN,ISYM,NPER,TITL,1,TBS,gb,PLTITL,599)



c This is the part of the circulation distribution < .93, which
c was taken from plots referenced in thesis.  Pullin's solution
c is not applied to this.
          num99 = 7
          tempbs(1) = .1
          tempgb(1) = .0
          tempbs(2) = .2
          tempgb(2) = .0072
          tempbs(3) = .3
          tempgb(3) = .01
          tempbs(4) = .42
          tempgb(4) = .012
          tempbs(5) = .7
          tempgb(5) = .0152
          tempbs(6) = .84
          tempgb(6) = .018
          tempbs(7) = .915
          tempgb(7) = .0216


      CALL  SPLIND(TEMPGB,TGBTBS,TEMPBS,NUM99,999.,999.)
```

```
      ainc = .01
      rad = -ainc
      disend = .92
      NDIS = disend/aINC + .01
      nend = ndis + 1
      ind = nend

      DO 6 I = 1,Nend


           rad = rad + ainc
           tbs(i) = rad
           tbs(1) = 0.
           tbs(nend) = disend
           gb(i) = seval(tbs(i),tempgb,tgbtbs,tempbs,num99)

           if(i.le.num2) then
              ind = ind + 1
              tbs(ind) = xplt(i)
              gb(ind) = yplt(i)
              endif

 6     continue

         ndis1 = npts

         npts = nend + num2
         CALL SPLIND(GB,GBBTBS,TBS,Ndis,999.,999.)
         NPER(1) = Npts
         TITL(1) ='GAMMA DIS'
         PLTITL = '~S ~GAMMA ~'
         CALL GRINIT(5,6,'CIRCULATION  DISTRIBUTION  VS. SHEET LENGTH')
         CALL GRKLIN(ILIN,ISYM,NPER,TITL,1,TBS,GB,PLTITL,599)



C*******************************************************


         return
         end ! pullin
c         stop
c         end
```