NASA Technical Memorandum 101041

A Numerical Study of Viscous Vortex Rings Using a Spectral Method

S. K. Stanaway, B. J. Cantwell, Stanford University, Stanford, California
P. R. Spalart, Ames Research Center, Moffett Field, California

October 1988

NASA
National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

Abstract

Viscous, axisymmetric vortex rings are investigated numerically by solving the incompressible Navier-Stokes equations using a spectral method designed for this type of flow. The results presented are axisymmetric, but the method is developed to be naturally extended to three dimensions. The spectral method relies on divergence-free basis functions. The basis functions are formed in spherical coordinates using Vector Spherical Harmonics in the angular directions, and Jacobi polynomials together with a mapping in the radial direction. Simulations are performed of a single ring over a wide range of Reynolds numbers ($Re \equiv \Gamma/\nu$), $0.001 \leq Re \leq 1000$, and of two interacting rings. At large times, regardless of the early history of the vortex ring, it is observed that the flow approaches a Stokes solution that depends only on the total hydrodynamic impulse, which is conserved for all time. At small times, from an infinitely thin ring, the propagation speeds of vortex rings of varying Re are computed and comparisons are made with the asymptotic theory by Saffman. Our results are in agreement with the theory; furthermore, the error is found to be smaller than Saffman's own estimate by a factor $\sqrt{\nu t/R^2}$ (at least for $Re = 0$). The error also decreases with increasing Re at fixed core-to-ring radius ratio, and appears to be independent of Re as $Re \to \infty$. Following a single ring, with $Re = 500$, the vorticity contours indicate shedding of vorticity into the wake and a settling of an initially circular core to a more elliptical shape, similar to Norbury's steady inviscid vortices. Finally, we consider the case of "leapfrogging" vortex rings with $Re = 1000$. The results show severe straining of the inner vortex core in the first pass and merging of the two cores during the second pass.

Contents

Page

List of Figures

Page

List of Symbols

Roman Symbols

a Core radius.

$a_j(t)$ Family of scalar time-dependent coefficients.

$A_{n'n}(\ell)$ An $N \times N$ matrix which multiplies the time-dependent term (referred to as the mass matrix).

$B_{n'n}(\ell)$ An $N \times N$ matrix which multiplies the viscous term (referred to as the viscous matrix).

c_ℓ Constant which is dependent upon ℓ.

D Analytic drift defining the velocity of propagation of a Stokes vortex ring.

D_c Computed value of the drift of a Stokes vortex ring.

dS Surface element

dV Volume element.

E Total energy.

$f_{n\ell}(r)$ Family of radial basis functions (note that these are scalars).

$F_{\ell m}(r,t)$	Family of (scalars) radial basis functions dependent upon both time and r.
$GR(n,\ell,i)$	Radial basis functions used in computing u_r.
$GTH(n,\ell,i)$	Radial basis functions used in computing u_θ.
$GV(n,\ell,i)$	Radial basis functions used in computing ω_ϕ.
G_n^ℓ	Shifted Jacobi polynomials.
I_E	Percent error in impulse (defined by eqn. 4.2.1).
I/ρ	Impulse (with dimensions L^4/T).
ℓ	Index in polar direction, θ.
L	Number of modes in θ.
L_ℓ	Radial dependence of the Laplacian operator.
m	Index in azimuthal direction, ϕ.
$\mathbf{n}$	Unit normal.
n	Index in r.
N	Number of modes in r.
$N_c(\ell)$	Number of collocation points in r (which depends upon ℓ).
p	Static pressure.
P_ℓ^m	Legendre polynomials.
$P0(\ell,j)$	Polar basis function – Legendre polynomial with $m = 0$.
$P1(\ell,j)$	Polar basis function – Legendre polynomial with $m = 1$.
$\mathbf{q}$	The vector field given by $\mathbf{u} \times \omega$.
$Q_{n'}(\ell)$	One dimensional array arising from the nonlinear term.
r	Radial direction in spherical polar coordinates (see fig. 2.1).
R	Ring radius.
Re_D	Reynolds number based on diameter and ring speed, $\frac{2RU}{\nu}$.
Re_I	Reynolds number based on impulse, $\frac{(I/\rho)^{1/2}}{\nu t_I^{1/2}}$.
Re_Γ	Reynolds number based on circulation, $\frac{\Gamma}{\nu}$.
s, β	Polar coordinates (see fig. 3.1).

t Time.

t_I Time from a virtual origin when the flow was initially created by an impulsive force.

t_Γ Time from a virtual origin when the vorticity was concentrated at a point (in 2D) or on a circular line (in 3D).

$\mathbf{u}$ Velocity vector.

$\mathbf{u}_{mod}$ Modified velocity vector (see eqn. 2.1.11).

$\mathbf{U}$ Propagation speed of a vortex ring.

u_β Polar component of velocity in polar coordinates (see fig. 3.1).

$w(x)$ Weight function defining a quadrature rule in x.

x, y, z Cartesian coordinates (see fig. F.1).

$\mathbf{x}$ Vector originating from the origin of the coordinates.

$\mathbf{X}(t)$ Displacement of center of coordinate system.

$\mathbf{X}_{\ell m}$ Divergence-free member of vector spherical harmonic functions.

$\mathbf{V}_{\ell m}$ Second family of vector spherical harmonic functions.

$\mathbf{W}_{\ell m}$ Third family of vector spherical harmonic functions.

Greek Symbols

ε Dissipation.

η Self similar coordinate in the Stokes solution $(r/\sqrt{4\nu t})$.

θ Polar direction in spherical polar coordinates. (see fig. 2.1)

ν Kinematic viscosity.

ξ Mapped coordinate in the radial direction defined over [0,1].

ρ Density.

ϕ Azimuthal direction in spherical polar coordinates (see fig. 2.1)

ω Vector vorticity.

ω_{error} difference between exact and approximation of vorticity at a point.

ω_E global normalized error (defined by eqn. 4.1.2).

Γ Circulation in a meridional half plane.

Υ Vector potential.

Φ Dynamic pressure, $p/\rho + \mathbf{u} \cdot \mathbf{u}/2$.

Ψ_j Family of vector weight functions (also called test functions).

Other Symbols

$(\ )_o$ The value of ( ) at the start of the calculation.

$(\ )^j$ The quantity ( ) at time level j.

$(\ )_r$ Radial component of the function ( ).

$(\ )_\theta$ Polar component of the function ( ).

$(\ )_\phi$ Azimuthal component of the function ( ).

$(\ )_1, (\ )_2, (\ )_3$ Subscripts denoting the three independent radial functions needed to define a complete time-varying vector function in a volume.

$(\ )^+, (\ )^-$ Superscripts denoting the two independent families of functions describing a divergence-free velocity field (reduced from three to two by using continuity).

$(\ )$ Plain quantities (for u, ω, t, etc.) denote physical variables.

$\overline{(\ )}$ Dimensionless variables.

$\widetilde{(\ )}$ Numerical variables.

Chapter 1
Introduction

1.1 Review of Literature

Vortex structures are often observed in complex flows, and as Helmholtz vortex laws dictate for unbounded flow, a vortex line must close upon itself. Topologically, the most simple three-dimensional structure is a vortex ring. This problem has fascinated engineers, scientists, physicists and mathematicians alike for over one hundred years. Among the first theoretical studies were the works of Kelvin (Thomson 1867) and J.J. Thomson (1883) who investigated vortex rings as candidates for the fundamental structure of atoms. In an appendix in which Helmholtz' paper is translated to English, Kelvin (1867) gave (without proof) the well known formula for the velocity of translation of a thin vortex ring having uniform vorticity (ω/y = constant, where ω is the vorticity and y is the distance from the axis of symmetry). The propagation speed has a logarithmic dependence on core radius such that the ring speed is finite for a finite core radius, and in the limit of zero core radius, the ring speed is infinite. This result was later verified by Hicks (1885) and Gray (1914). It was then extended to a viscous vortex ring by Tung and Ting (1967) and derived by different means by Saffman (1970) (where he discovered a numerical error in the results of Tung and Ting). In the present work, Saffman's expression is verified and extended to higher order numerically. Furthermore, the effect of Reynolds number is quantified.

A classical view of vortex rings is given in Lamb (1945), Prandtl & Tietjens (1934), Sommerfeld (1950) and Batchelor (1967). Based on inviscid theory, a circular core of uniform vorticity travels under its own induced velocity as predicted by Kelvin. Furthermore, surrounding the vortex core and carried with it is a bubble of irrotational fluid which extends to the axis of symmetry unless the core-to-ring radius is smaller than about 0.01. The role which viscosity plays in the real flow was perhaps first observed by Reynolds (1876) where he states that "... they are continually adding to their bulk water taken up from that which surrounds them and with which their forward momentum has to be shared."

Vortex rings are experimentally generated by forcing a slug of fluid through a nozzle with a sharp lip (Maxworthy, 1977, Didden, 1979) or through a hole in a rigid wall (Glezer, 1981). Vorticity generated along the walls separates from the trailing edge and spirals into a vortex ring. By injecting either smoke into air, or dye into water at the lip, the ring is made visible through streaklines (1964 Magarvey & MacLatchy). This flow visualization technique became widely publicized through the huge smoke rings generated on the Camel cigarettes billboard in Times Square in New York. As pointed out by Magarvey & MacLatchy (1964) and others (Maxworthy, 1972), interpretation of the dye requires caution since the flow is viscous and the vorticity diffuses much more quickly than the dye. For example, a streakline can show a spiral structure, while the corresponding vorticity distribution is a smooth Gaussian.

A vortex ring is characterized by the ring radius, translational velocity, and the circulation. In an experiment, we must relate the parameters associated with the apparatus (the nozzle diameter, velocity history of the piston, and stroke length) to those of the ring. This subject is described by Didden (1979) where he examines detailed velocity measurements of the flow at the nozzle exit and relates this to the final circulation. Furthermore, Didden shows that the starting process is strongly dependent on secondary effects such as the vorticity of opposite sign which is generated on the outside of the nozzle.

Once the laminar sheet of vorticity leaves the nozzle, and spirals into a vortex ring, several situations are possible depending on the Reynolds number (Maxworthy, 1972). For Reynolds numbers less than around 600 (based on initial translation velocity and maximum bubble diameter), a stable, laminar vortex results. For Reynolds numbers greater than 600, azimuthal waves develop, as first demonstrated by Krutzsch (1939) and later observed by Widnall & Sullivan (1973), Liess & Didden (1976), and Maxworthy (1972, 1977) and numerically by Knio & Ghoniem (1988) with an inviscid vortex method. If the Reynolds number is greater than around 1000, the waves grow to amplitude where they break, resulting in turbulent flow. After some period of time a new stable ring emerges. This process was occasionally found to repeat itself as implied by Krutzsch, and then later reported by Brasseur (1985). Among the unanswered questions are the effects of viscosity and swirl on the instabilities (Saffman, 1978). Fully turbulent flow was investigated by Kovasznay, Fujita, Lee (1973) and later by Glezer (1981), who fully exploited the similarity of the flow resulting from Reynolds number invariance.

Three dimensional instabilities and the resulting turbulent vortex rings described above are beyond the scope of the studies considered here, as the results we will present are for axisymmetric rings. The numerical method, however, is naturally extended to three dimensions and will enable three dimensional effects to be studied at moderate Reynolds numbers in future work.

Because of the inherent unsteadiness of vortex rings, both temporally and spatially, quantitative measurement of the time-dependent vorticity and velocity fields has presented challenges to experimentalists. Widnall & Sullivan (1973) presented the first measurements of vorticity distributions at a single time in the evolution. Similar measurements were also shown by Maxworthy (1977). Other experimental results by Sallet & Widmayer (1974) give the time dependent ring velocity, ring diameter, core diameter and circulation. Maxworthy (1972) presents a model for the behavior of viscous stable rings. He proposes that, in contrast to the classical models, the vorticity is distributed throughout the bubble of fluid carried with the vortex core, and furthermore, the bubble shape grows in a self-similar fashion. The external irrotational fluid flows past the bubble, and through viscous diffusion a thin layer mixes with the vorticity in the bubble. The total pressure in the thin layer is reduced, and fluid is therefore entrained into the rear of the bubble. In order for the model to be dynamically consistent, he argues that vorticity is continually being shed into the wake. In our simulations, we show that there is indeed a wake of weak vorticity continually being shed.

It is natural to look for steady inviscid solutions (in the frame of reference traveling with the ring). A very early result by Hill (1894) gave such a solution with uniform vorticity (ω/y = constant) distributed in a sphere, known as Hill's spherical vortex. The existence of steady thin rings was proved by Fraenkel (1972), and Norbury (1973) studied rings of finite size. Norbury's family of steady rings range from thin rings at one end to Hill's spherical vortex at the other. For a viscous vortex, it is not possible to have a steady solution (because the energy decays). Our numerical viscous solutions, however, show a quasi-steady behavior (a nearly self-similar shape in a translating frame) with vorticity distributions reminiscent of Norbury's rings.

As the vortex ring propagates, fluid is entrained and the bubble of vorticity continues to grow. Meanwhile, vorticity is diffusing across the axis of symmetry and canceling with vorticity of the opposite sign, so that in the limit of large-time, the circulation goes to zero. The circulation is one measure of the Reynolds number. It follows that the Reynolds number is decreasing in time and asymptotically approaches zero.

For sufficiently large time, the convective term is negligible and we are left with the Stokes equations. The most slowly decaying solution to these equations is what is often termed the Stokes solution. This is an analytic solution first derived by Phillips (1956), which he points out is the final period of decay for any initial vorticity distribution in an infinite domain where the flow at infinity is at rest. Further studies of this regime were made by Kambe and Oshima (1975) where they experimentally verify the predicted $t^{-3/2}$ decay of velocity and claim to have extended the large time solution to second order through the method of matched asymptotic expansions. In the present work, it was shown numerically that for large times, the vorticity field returns to the drifting Stokes solution and that the only memory of the initial conditions that remains is the impulse and kinematic viscosity. Furthermore, we have numerically determined the proportionality constant relating the dimensionless propagation speed to $t^{-3/2}$. This constant was later found analytically by Rott & Cantwell (1987) and agrees with the numerical result to 4 significant digits, indicating that the numerical solution is very accurate.

In this work we show results for the propagation speed, shedding, and interactions, and ultimately the decay of axisymmetric viscous vortex rings from initial Reynolds numbers as high as 1000. A very accurate numerical method was developed for three-dimensional flows, and tested for axisymmetric rings. It was shown that the impulse, which is analytically conserved, is constant during a simulation to within 2% for even the most difficult case, and more typically to within fractions of a percent. Therefore, these results may be useful as a database to compare with other numerical methods under development such as discrete vortex methods which include the effects of viscosity.

With a numerical method, we are able to obtain *any* information from the flow, making this a very powerful tool (provided that the solutions are correct and accurate).

1.2 Numerical Method

The complete equations describing the physics are the incompressible Navier-Stokes equations. Because the Reynolds number is limited by the size of the smallest scale of the flow which can be resolved, it is desirable to use a numerical approach which is highly accurate for a given number of degrees of freedom. When properly formulated, spectral methods are known to have exponential convergence with respect to the number of degrees of freedom and to allow an excellent resolution of small scales. The objective, therefore, is to develop an efficient spectral method applicable to vortex ring calculations in an unbounded domain with a quiescent far field.

Two approaches were considered. The first approach, suggested by Leonard (1981), uses divergence-free basis functions tailored to inherently satisfy the boundary conditions of the problem. Because of this, the pressure does not appear in the final equations and the incompressibility condition allows the number of unknown velocity components to be reduced to two. The second approach considered uses a complete set of basis functions and enforces continuity (through the expansion coefficients) at each time step. In this case, there are four unknowns; three velocity components and the pressure. The first approach, although more elegant, relies on finding an appropriate divergence-free set of basis functions, which can be difficult.

The appropriate set of basis functions is strongly affected by the choice of coordinate system. Cartesian coordinates are mathematically the most simple to work with and therefore, the obvious first case to consider. An interesting approach for computing an infinite domain is to first map the domain $[-\infty, \infty]$ to $[-1, 1]$ using a tangent mapping and then to apply Fourier series expansions (Cain's mapping, Cain *et al.,* 1984). With the mapping, an infinite domain can be approximated with a periodic expansion since the image of the non-zero vorticity is infinitely far away from the flow of interest. An advantage is that Fourier series expansions can be applied, allowing FFT's (fast Fourier transforms) to be used, reducing the operation count from $O(N^2)$ to $O(N \log N)$. Cain's mapping has been successfully implemented on several occasions in one direction where one or both of the remaining two directions are assumed to be periodic (Cain *et al.,* 1984, Lowery, 1986). This approach results in a bandwidth of 5 for each direction for the mass matrix as well as a Poisson equation with bandwidth of 5 in each direction, each of which must be inverted to advance one time step. Because the domain of interest is infinite in all directions, the mapping would be applied three times, resulting in a fully coupled system of equations which would be extremely costly to invert.

The second coordinate system considered is spherical polar. The advantages here are that only one direction is infinite, and by holding the azimuthal direction, ϕ, fixed we can conveniently represent axisymmetric flow with only two dimensions. With only two dimensions we are able to study more cases. Furthermore, by computing exactly axisymmetric flow, we can compute the instabilities from this state more precisely. Another advantage here is that a set of functions which comprise a complete set for a vector field on the surface of a sphere are known, and furthermore their derivative relations are relatively simple. These are known as vector spherical harmonics (VSH, Hill 1953). With VSH, it is relatively straightforward to extract the divergence-free set of functions. Furthermore, because VSH are orthogonal, the angular directions completely decouple in the linear terms of the Navier-Stokes equations. To define vector functions for a divergence-free field in a volume, one must then choose appropriate radial functions. This is the approach taken.

The radial direction is handled by first mapping the semi-infinite domain, $0 \le r < \infty$, to a finite domain, $0 \le \xi < 1$, then expanding the velocity and vorticity in terms of Jacobi polynomials. Care is taken to ensure that in the far field, the velocity decays as $1/r^3$. Exponential decay in vorticity (occurring when the vorticity is initially zero outside a finite region) can be approximated, but individually the basis functions decay algebraically. The resulting matrices in the radial direction are banded, positive-definite and symmetric, with semi-bandwidths of 3 and 5 for the mass and viscous matrices, respectively. In addition, the functions are chosen such that the solution is smooth at the origin where there is a coordinate singularity.

The divergence-free method developed is similar to those of Leonard *et al.* (1982) for pipe flow, Moser *et al.* (1983) for channel flow, and Spalart (1986) for boundary layers. Special considerations which arise in applying this approach to the present problem are the unbounded domain, and the more complex geometry of the flow structure. Because of these complexities, two of the three directions do not allow the use of Fast Fourier Transforms in transforming between real and wave-number space.

The method is particularly well suited for flows where the activity is concentrated in a spherical domain. Vortex rings are representative of an important class of flows which are produced by a time-dependent point force (Cantwell, 1987). A delta-function forcing, for example, produces a vortex ring. The numerical method developed here for this particular flow could be extended to other members of this class, produced by more complex forcings.

1.3 Objectives

The objectives of this work, divided into numerics and fluid mechanics, are outlined below.

Objectives: Numerics

(i) To develop an accurate numerical method to study viscous vortex rings in three-dimensions.

(ii) To verify the method by implementing the axisymmetric case.

Objectives: Fluid Mechanics

(i) To study the large time behavior of axisymmetric rings; in particular, how does the asymptotic solution of the Navier-Stokes equations depend on the initial conditions.

(ii) To study the early time behavior when the ring is very thin and compare the propagation speed with the theory by Saffman.

(iii) To observe intermediate time development of vortex rings such as the deformation of the core, and shedding of vorticity into the wake.

(iv) To provide a database of an accurate solution of the Navier-Stokes equations representing vortex rings in unbounded domains.

(v) To observe leapfrogging of vortex rings.

Chapter 2

Design of the Numerical Method

In this chapter a spectral method is developed which solves the Navier-Stokes equations in an unbounded domain with a quiescent far field. There are no external forces acting on the flow, however including them would be a straightforward process. Using a weighted residual method, the momentum equation is projected onto specified weight functions. The flow field is represented as a summation of spatially-dependent functions, which form the basis for the solution space, multiplied by time-dependent coefficients.

In the method developed here, the weight functions and basis functions are the same (i.e. Galerkin approximation, cf., Gottlieb and Orszag, 1977). These functions are built using Vector Spherical Harmonics (VSH) for the angular dependence and an algebraic mapping together with Jacobi polynomials for the radial dependence. Using VSH, the set of basis functions is complete, and the divergence-free subset is extracted. Each basis function individually satisfies continuity, therefore, the continuity equation is implicitly satisfied in the formulation.

In the first section, we manipulate the conservation equations; first by non-dimensionalizing, then by transforming to an expanding and translating coordinate system. The transformation enables vortex rings to be computed over long time periods while having a minimum impact on the numerical algorithm. Next, we apply the weighted residual procedure. By using a Galerkin method with divergence-free basis functions, it is shown that the pressure drops from the equation. Finally, the basis functions are developed and substituted into the governing equations, leading to a set of coupled equations which are solved numerically. With this procedure, the spatial dependence is accounted for, and the Navier-Stokes equations reduce to a set of coupled ordinary differential equations which are integrated in time from an initial flow field.

2.1 Governing Equations

Using the invariant quantities, impulse (I/ρ) and kinematic viscosity (ν), the Navier-Stokes equations are first expressed in dimensionless form (identified by $\overline{(\ )}$). Next, a transformation is found, such that the boundary conditions at infinity are unchanged and the changes to the working equations are minimal. Because of this, the equations are cast in a hybrid coordinate system; the independent variables are referred to moving coordinates but the pressure and velocity are referred to non-moving coordinates. The computational variables are identified by $\widetilde{(\ )}$.

2.1.1 Physical and Non-dimensional Equations

The incompressible Navier-Stokes equations express conservation of mass,

$$\nabla \cdot \mathbf{u} = 0 \tag{2.1.1}$$

and conservation of momentum,

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} \tag{2.1.2}$$

where $\mathbf{u}$ is the velocity, p is the pressure, ρ is the density, and ν is the kinematic viscosity. Boldface quantities denote vectors. The domain of interest is infinite, with the vorticity confined to a finite region. It can be shown that, in this instance, the velocity decays like $1/r^3$ in the far field.

With the vorticity defined as $\omega = \nabla \times \mathbf{u}$ and the vector identity,

$$\mathbf{u} \cdot \nabla \mathbf{u} = \nabla(\mathbf{u} \cdot \mathbf{u}/2) - \mathbf{u} \times \omega \tag{2.1.3}$$

equation (2.1.2), can be written as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \Phi - \nu \nabla^2 \mathbf{u} = \mathbf{u} \times \omega, \tag{2.1.4}$$

where $\Phi \equiv p/\rho + \mathbf{u} \cdot \mathbf{u}/2$ is the total pressure.

The dimensions of the impulse (I/ρ), and the kinematic viscosity (ν), are L^4/T and L^2/T, respectively. By inspection, a length scale is formed, $(I/\rho)^{1/2}/\nu^{1/2}$, and a time scale, $(I/\rho)/\nu^2$. With this, the dimensionless variables are

$$\begin{aligned} \overline{\mathbf{x}} &\equiv \mathbf{x}\,\frac{\nu^{1/2}}{(I/\rho)^{1/2}} & \overline{\mathbf{u}} &\equiv \mathbf{u}\,\frac{(I/\rho)^{1/2}}{\nu^{3/2}} \\ & & \overline{\omega} &\equiv \omega\,\frac{I/\rho}{\nu^2} \\ \overline{t} &\equiv t\,\frac{\nu^2}{I/\rho} & \overline{p} &\equiv p/\rho\frac{I/\rho}{\nu^3}. \end{aligned} \tag{2.1.5}$$

Substituting equation (2.1.5) into equation (2.1.4) gives the dimensionless momentum equation,

$$\frac{\partial\overline{\mathbf{u}}}{\partial\overline{t}} + \overline{\nabla}\,\overline{\Phi} - \overline{\nabla}^2\overline{\mathbf{u}} = \overline{\mathbf{u}} \times \overline{\omega}, \tag{2.1.6}$$

where $\overline{\Phi} = \overline{p} + \overline{\mathbf{u}}^2/2$. Continuity becomes

$$\overline{\nabla}\cdot\overline{\mathbf{u}} = 0. \tag{2.1.7}$$

2.1.2 Transformed Equations

Since the domain of interest is infinite and we are studying vortex rings that are diffusing and translating in time, it is helpful to express the governing equations in a coordinate system which is also expanding and translating. In this way, we are able to efficiently compute flows from an initial state, such as a thin ring at high Reynolds number, for long periods of time.

A transformation is found which leaves the boundary conditions invariant and has a minimal impact on the algorithm. It is a hybrid transformation, where the velocity and pressure are referred to fixed coordinates while the independent variables are referred to a translating reference frame, given by

$$\begin{aligned} \tilde{\mathbf{x}} &\equiv (\overline{\mathbf{x}} - \overline{\mathbf{X}})\,\overline{t}^{\,-1/2}, & \tilde{\mathbf{u}} &\equiv \overline{\mathbf{u}}\,\overline{t}^{\,1/2}, \\ \tilde{t} &\equiv \ln\overline{t}, & \tilde{\omega} &\equiv \overline{\omega}\,\overline{t}, \\ & & \tilde{p} &\equiv \overline{p}\,\overline{t}. \end{aligned} \tag{2.1.8}$$

Here, $\overline{\mathbf{X}}$ is the displacement of the center of the coordinate system as a function of time. The expansion is prescribed proportional to $\overline{t}^{1/2}$ because this is the variation of the viscous length scale. However, the origin of $\overline{t}$ is arbitrary and will be chosen carefully in each case. Similarly, $\overline{\mathbf{X}}$ is left unspecified for now and will be adjusted to minimize the time dependence of the solution in the transformed frame. Note that the reference frame moving with $\overline{\mathbf{X}}$ does not have to be inertial.

Appendix A shows the details of the transformation. The final equations in the new coordinates express conservation of momentum,

$$\frac{\partial \widetilde{\mathbf{u}}}{\partial \widetilde{t}} + \widetilde{\nabla}\,\widetilde{\Phi} - \widetilde{\nabla}^2\,\widetilde{\mathbf{u}} = \widetilde{\mathbf{u}}_{mod} \times \widetilde{\omega} \tag{2.1.9}$$

and conservation of mass,

$$\widetilde{\nabla} \cdot \widetilde{\mathbf{u}} = 0. \tag{2.1.10}$$

Equation (2.1.9) has the same form as equation (2.1.4) except that the pressure and the non-linear terms are modified. They are given by

$$\begin{aligned}
\widetilde{\mathbf{u}}_{mod} &\equiv \widetilde{\mathbf{u}} - \frac{1}{2}\widetilde{\mathbf{x}} - \widetilde{\mathbf{U}} \\
\widetilde{\Phi} &\equiv \widetilde{p} + \widetilde{\mathbf{u}} \cdot \widetilde{\mathbf{u}}/2 - \frac{1}{2}\widetilde{\mathbf{x}} \cdot \widetilde{\mathbf{u}} - \widetilde{\mathbf{U}} \cdot \widetilde{\mathbf{u}} \\
\widetilde{\mathbf{U}} &\equiv \overline{t}^{\,1/2}\,\overline{\mathbf{U}} = \overline{t}^{\,1/2}\,\frac{d\overline{\mathbf{X}}}{d\overline{t}}.
\end{aligned} \tag{2.1.11}$$

For reasons to be described shortly, the pressure does not appear in the final equations, so that the only effect on the algorithm is due to the modified velocity in the non-linear term; $\overline{\mathbf{u}}$ becomes $\widetilde{\mathbf{u}}_{mod}$ each time the non-linear term is computed (see Chapter 3 for more detail). To simplify the notation, we will present the solution procedure of equation (2.1.4), even though equation (2.1.9) is solved in practice. The differences are minor: ν becomes 1 and the velocity in the convection term is replaced by $\mathbf{u}_{mod}$.

2.2 Weighted Residual Method

In a weighted residual method, the governing equations are multiplied by a set of weight functions, Ψ_j, and integrated over the domain of interest. This gives

$$< \frac{\partial \mathbf{u}}{\partial t}, \Psi_j > + < \nabla\Phi, \Psi_j > -\nu < \nabla^2\mathbf{u}, \Psi_j > \; = \; < \mathbf{u} \times \omega, \Psi_j > \tag{2.2.1}$$

where $< \mathbf{a}, \mathbf{b} >$, an inner product, denotes the integral of the dot product of two vectors, $\mathbf{a}$ and $\mathbf{b}$, over the volume. All of the quantities will decay fast enough at large distances for the integrals over the infinite domain to be finite.

By specifying certain constraints on the weight functions, Ψ_j, the pressure term will drop from the governing equations, greatly simplifying the numerical method. Using the product rule and Green's Theorem, the second term in equation (2.2.1) becomes

$$< \nabla\Phi, \Psi_j >= \int_S \Phi(\Psi_j \cdot \mathbf{n})dS - \int_V \Phi(\nabla \cdot \Psi_j)dV. \tag{2.2.2}$$

In Leonards' formulation (1982), the domain was bounded by pipe walls. The normal component of the basis functions is therefore zero at the boundaries, and the wall boundaries don't contribute to the first term in equation (2.2.2). The streamwise direction was periodic, and therefore had no contribution, hence the first term is zero. Since our domain is infinite, extra care is needed. The first term is zero when the integrand, $\Phi(\Psi_j \cdot \mathbf{n})$, decays fast enough such that the limit of the integral as S becomes infinitely large is zero. Since Φ decays like $1/r^2$ and Ψ_j decays like $1/r^3$, this is clearly the case. The second term is zero for weight functions which are divergence-free ($\nabla \cdot \Psi_j \equiv 0$). The weight functions, therefore, are chosen such that they satisfy continuity and the boundary conditions, leading to the so called *weak formulation* due to Leray (1934):

$$< \frac{\partial \mathbf{u}}{\partial t}, \Psi_j > -\nu < \nabla^2 \mathbf{u}, \Psi_j > = < \mathbf{u} \times \omega, \Psi_j > . \qquad (2.2.3)$$

For the Stokes equations, it can be shown (Moser *et al.,* 1984) that the solution of the strong form (eqn. 2.1.4) is also a solution of the weak form (eqn. 2.2.2) and that the solution of the weak form is unique. Therefore, it is valid to solve the weak form of the equations, and a solution of the strong form will be found (if it exists). It is also true that the weak form of the equations may have a solution when the strong form does not. But, there are no firm examples of such a behavior with the incompressible Navier-Stokes equations.

The next step in the formulation is to expand the velocity and vorticity in terms of unknown time-dependent coefficients, $a_j(t)$, multiplying known spatial-dependent basis functions, $\Psi_j(\mathbf{x})$:

$$\mathbf{u} = \sum_j a_j(t)\, \Psi_j(\mathbf{x}) \qquad (2.2.4)$$

where each basis function is divergence-free. Equation (2.2.4) is substituted into (2.2.3), the time-dependence is brought outside of the integrals, and inside the integral are expressions involving products of weight functions. In the integrand, there are two indices (j and j' for example) such that each integral is an element of a matrix. Furthermore, the integrals (or matrix elements) are dependent only on the spatial functions which are known a priori, and therefore only need to be computed once. The matrix multiplying the time-dependent term will be called the mass matrix and that originating from the diffusion term will be referred to as the viscous matrix.

If we constrain the basis functions and weight functions to span the same space (i.e. a Galerkin method), several benefits are realized. It can be shown that the solution gives a minimum in the L_2 error in the vorticity. Furthermore, the convergence properties of the numerical approximation to the differential equations are as good as the convergence of the series expansion to typical solutions (approximation theory) (Moser *et al.,* 1984). Also, when the weight functions and basis functions are the same, the mass and viscous matrices are positive-definite and symmetric,

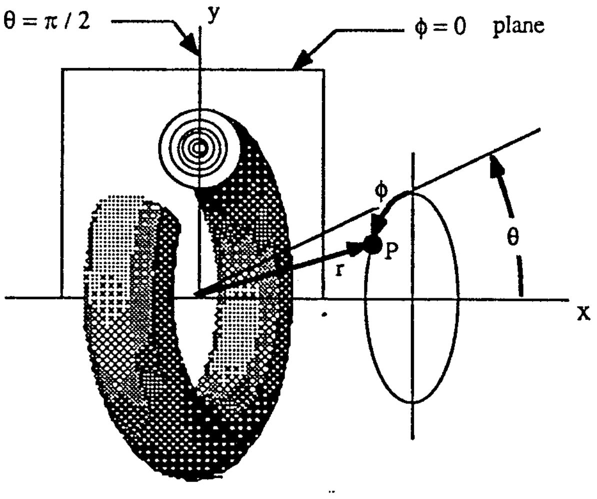


Figure 2.1. Schematic of a vortex ring is spherical coordinates. The domain is infinite, where $0 \leq r \leq \infty$, $0 \leq \theta \leq \pi$, and $0 \leq \phi \leq 2\pi$. The cross section indicates lines of constant vorticity.

leading to a more efficient numerical method. This approach, used by Spalart (1986) to simulate turbulent boundary layers, was also used here. Choosing basis functions which are appropriate for computing vortex rings and their interactions in an unbounded domain is the subject of the remainder of the Chapter.

2.3 Three-Dimensional Basis Functions

The basis functions must not only be complete for a given set of endpoint conditions and divergence-free, they should also lead to an efficient numerical method. One measure of the numerical efficiency is the sparseness and bandedness of the matrices resulting from the linear terms (on the left hand side of eqn. 2.2.3). Ideally, the basis functions would be orthogonal (bandwidth of one) in all three spatial coordinates. In practice, this is probably impossible to achieve simply by a judicious choice of functions.

Our problem is formulated in spherical polar coordinates (see Fig. 2.1) which offers several advantages. Vector spherical harmonics (VSH, cf., Hill, 1953), can be used as basis functions in the angular directions. They span the space of vector fields on the surface of a sphere. A complete set of functions in a volume is generated by multiplying the VSH by appropriate radial functions. Furthermore, the divergence-free subspace can be easily extracted using the properties of the VSH (Appendix C). Finally, because of the orthogonality of VSH, the different angular modes are completely decoupled in the linear terms of the Navier-Stokes equations. With these, the only remaining task is to choose the radial functions.

The radial direction is mapped from the semi-infinite domain, $0 \leq r < \infty$, to a finite domain, $0 \leq \xi < 1$, then the velocity and vorticity are expanded in terms of Jacobi polynomials. Care is taken such that in the far field, the velocity decays as $1/r^3$. Exponential decay in vorticity can be approximated, but individually the basis functions decay algebraically. In addition, the functions must satisfy special conditions at the origin where there is a coordinate singularity.

2.3.1 Vector Spherical Harmonics

The coordinate system plays a vital role in the method. With this, the vector field (velocity or vorticity) is projected onto each of the coordinate directions and each component is expanded in terms of a family of complete polynomials. The coordinate system determines the boundary conditions in each direction which in turn determines the appropriate functions to use. Furthermore, the higher-resolution region of the approximating functions should coincide with the large gradients in the flow field. Finally, it is useful if a subset of the basis functions is consistent with the physics of the flow (i.e. by removing one coordinate the 2D or axisymmetric flow is recovered exactly).

For this problem, we chose spherical polar coordinates (see Fig. 1). By holding ϕ constant, the three-dimensional problem reduces exactly to the axisymmetric problem. This will be important for future studies of azimuthal instabilities and is a definite advantage over Cartesian coordinates. In addition, it is desirable that only one direction (radial) be infinite since special care is needed in an infinite interval, making this an advantage over cylindrical coordinates. Finally, the availability of VSH and their properties enables a complete, divergence-free set of basis functions to be defined analytically without excessive complexity. Furthermore, with VSH the matrices from the time-dependent and viscous terms in the Navier-Stokes equations are diagonal in the polar and azimuthal directions. In this section, these properties are illustrated and the basis functions are developed.

Since the vector spherical harmonics, $\mathbf{X}_{\ell m}$, $\mathbf{V}_{\ell m}$, and $\mathbf{W}_{\ell m}$, form a complete set on a sphere (Blatt & Weisskopf, 1952), an arbitrary, unsteady, three-dimensional vector field can be represented by

$$\begin{aligned}\mathbf{u}(r,\theta,\phi,t) = \sum_{\ell,m} \{&F_{1_{\ell m}}(r,t)\mathbf{X}_{\ell m}(\theta,\phi) \\ &+ F_{2_{\ell m}}(r,t)\mathbf{V}_{\ell m}(\theta,\phi) + F_{3_{\ell m}}(r,t)\mathbf{W}_{\ell m}(\theta,\phi)\}.\end{aligned} \tag{2.3.1}$$

The scalar functions, $F_{1_{\ell m}}$, $F_{2_{\ell m}}$, and $F_{3_{\ell m}}$, are arbitrary radial functions which vary in time. In numerical solutions, the series is truncated with $1 \le \ell \le L$ and $0 \le |m| \le \ell$ where the limit, L, is set according to the complexity of the flow field (i.e. the ratio of length scales which depends on the Reynolds number).

The divergence-free subspace is extracted from the complete space by substituting equation (2.3.1) into the continuity equation, and applying VSH properties (see Appendix B). This gives a velocity expansion requiring only two scalar functions, $F^-_{\ell m}(r,t)$ and $F^+_{\ell m}(r,t)$, instead of three:

$$\mathbf{u}(r,\theta,\phi,t) = \sum_{\ell,m} \left\{ F^-_{\ell m}(r,t)\mathrm{X}_{\ell m}(\theta,\phi) + \nabla \times [F^+_{\ell m}(r,t)\mathrm{X}_{\ell m}(\theta,\phi)] \right\}. \qquad (2.3.2)$$

The functions denoted by + and − are related to those denoted by 1, 2, and 3 as follows:

$$\begin{aligned} F_{1_{\ell m}} &\equiv F^-_{\ell m} \\ F_{2_{\ell m}} &\equiv k_1 \left[\frac{dF^+_{\ell m}}{dr} - \frac{\ell}{r} F^+_{\ell m} \right] \\ F_{3_{\ell m}} &\equiv k_2 \left[\frac{dF^+_{\ell m}}{dr} + \frac{(\ell+1)}{r} F^+_{\ell m} \right] \end{aligned} \qquad (2.3.3)$$

where $k_1 \equiv i(\frac{\ell}{2\ell+1})^{1/2}$, and $k_2 \equiv i(\frac{\ell+1}{2\ell+1})^{1/2}$. The + and − notation is that used by Leonard and Wray (1982). This procedure amounts to using a vector potential. That is, since a divergence-free vector can be written as the curl of another vector, the basis functions in equation (2.3.2) can be expressed as $\nabla \times \Upsilon^-$ and $\nabla \times \Upsilon^+$, where Υ^- and Υ^+ are the vector potentials. In two dimensions, the vector potential simply reduces to the stream function.

Until this point, the functions denoted by F included both the spatial variation in the radial direction and the time-dependence. These two dependences are now separated by expanding the radial functions in terms of polynomials. If these are properly chosen (solutions of a singular Sturm-Liouville problem) the expansion is complete and able to represent arbitrary behavior at the endpoints without exhibiting Gibbs phenomena (Gottlieb & Orszag, 1977). The particular choice of radial functions is described in the next section, but for the moment they are represented by $f^-_{n\ell}(r)$ and $f^+_{n\ell}(r)$ where n is the third index in our expansion. Note that these functions do not depend on the index m (thanks to some properties of the VSH) which simplified the method significantly. The time-dependence is expressed by unknown time-dependent coefficients, $a^-_{n\ell m}(t)$ and $a^+_{n\ell m}(t)$. The radial and time-dependence are then separated:

$$\begin{aligned} F^-_{\ell m}(r,t) &= \sum_{n=0}^{N(\ell)} a^-_{n\ell m}(t) f^-_{n\ell}(r) \\ F^+_{\ell m}(r,t) &= \sum_{n=0}^{N(\ell)} a^+_{n\ell m}(t) f^+_{n\ell}(r) \end{aligned} \qquad (2.3.4)$$

The expansion for the velocity becomes

$$\mathbf{u}(r,\theta,\phi,t) = \sum_{\ell,n,m} \left\{ a^-_{n\ell m}(t) f^-_{n\ell}(r) \mathrm{X}_{\ell m}(\theta,\phi) + a^+_{n\ell m}(t) \nabla \times \left[f^+_{n\ell}(r) \mathrm{X}_{\ell m}(\theta,\phi) \right] \right\}. \tag{2.3.5}$$

where $1 \leq \ell \leq L$, $0 \leq n \leq N(\ell)$, and $0 \leq m \leq \ell$. Furthermore, $N(\ell) = N(1) - \ell + 1$ and $N(1) \equiv N$ (see section 3.2). As discussed in the previous section, the weight functions and basis functions are the same. Therefore, the weight functions are simply,

$$\begin{aligned} \Psi^-_{n'\ell' m'} &= f^-_{n'\ell'} \mathrm{X}_{\ell' m'}, \\ \Psi^+_{n'\ell' m'} &= \nabla \times (f^+_{n'\ell'} \mathrm{X}_{\ell' m'}). \end{aligned} \tag{2.3.6}$$

We are now in a position to derive the final equations. Note that the radial functions are still arbitrary. The resulting equations will involve integrals of radial functions, and these functions will be chosen so that as few of these integrals as possible are nonzero.

Substituting equation (2.3.5) into equation (2.2.3), and using the orthogonality of VSH, we can show that the angular modes are orthogonal in the mass and viscous term (all the integrals are zero unless $\ell = \ell', m = m'$). In practice, this means that the two angular directions are completely decoupled for the linear terms in the Navier-Stokes equations. The result is two sets of ordinary differential equations for each of the indices, l and m, with dependent variables $a^-_{n\ell m}$ and $a^+_{n\ell m}$. In summation notation they read,

$$A^-_{n'n}(\ell) \frac{da^-_{n\ell m}}{dt} - \nu B^-_{n'n}(\ell) a^-_{n\ell m} = < \mathbf{u} \times \omega, f^-_{n'\ell} \mathrm{X}^*_{\ell m} > \tag{2.3.7}$$

$$A^+_{n'n}(\ell) \frac{da^+_{n\ell m}}{dt} - \nu B^+_{n'n}(\ell)\ a^+_{n\ell m} = < \mathbf{u} \times \omega, \nabla \times (f^+_{n'\ell} \mathrm{X}^*_{\ell m}) > \tag{2.3.8}$$

where $\mathrm{X}^*_{\ell m}$ is the complex conjugate of $\mathrm{X}_{\ell m}$. The matrix elements of $A_{n\,n'}$ and $B_{n\,n'}$, in terms of the radial basis functions, are given by

$$\begin{aligned} A^-_{n'n}(\ell) &\equiv \int_0^\infty f^-_{n\ell} f^-_{n'\ell}\, r^2 dr, \\ B^-_{n'n}(\ell) &\equiv \int_0^\infty L_\ell(f^-_{n\ell}) f^-_{n'\ell}\, r^2 dr, \\ A^+_{n'n}(\ell) &\equiv \int_0^\infty \left[f_{2_{n\ell}} f_{2_{n'\ell}} + f_{3_{n\ell}} f_{3_{n'\ell}} \right] r^2 dr, \\ B^+_{n'n}(\ell) &\equiv \int_0^\infty \left[L_{\ell+1}(f_{2_{n\ell}}) f_{2_{n'\ell}} + L_{\ell-1}(f_{3_{n\ell}}\ ) f_{3_{n'\ell}} \right] r^2 dr. \end{aligned} \tag{2.3.9}$$

They are real, independent of m, and symmetric. This is so because the Laplacian operator, $L_\ell(f) = \frac{d^2 f}{dr^2} + \frac{2}{r}\frac{df}{dr} - \frac{\ell(\ell+1)}{r^2} f$, (see VSH, Appendix C) is self-adjoint with respect to the measure $r^2\,dr$. Furthermore, $A_{n'n}$ is positive definite and $B_{n'n}$ is negative definite. The definitions of $f_{2_{n\ell}}$ and $f_{3_{n\ell}}$, in terms of a single function, $f_{n\ell}^{+}$, follow from equation (2.3.3):

$$f_{2_{n\ell}}(r) = i\left(\frac{\ell}{2\ell+1}\right)^{1/2}\left[\frac{d}{dr} - \frac{\ell}{r}\right]\; f_{n\ell}^{+}(r) \tag{2.3.10}$$

$$f_{3_{n\ell}}(r) = i\left(\frac{\ell+1}{2\ell+1}\right)^{1/2}\left[\frac{d}{dr} + \frac{(\ell+1)}{r}\right]\; f_{n\ell}^{+}(r) \tag{2.3.11}$$

Once the radial basis functions ($f_{n\ell}^{-}$ and $f_{n\ell}^{+}$) have been chosen, the matrix elements are evaluated. In the next section, these radial functions are chosen judiciously to minimize the radial coupling while still satisfying the boundary conditions.

2.3.2 Radial Functions

There are many constraints driving the choice of the radial functions. They must form a complete set and satisfy the boundary conditions. At the coordinate singularity, $r = 0$, the basis functions must be smooth (C_∞), implying in particular that they have the correct parity. The basis functions should be suitable for approximating typical functions, in other words, the collocation points should cluster in regions of large gradients. In order for three-dimensional flows to be studied, it is important that the number of operations per time-step be kept to a minimum. This is dependent primarily on the matrix structures (i.e. diagonal is optimal) and the availability of fast transforms (i.e. as FFT's). Because these constraints are not applied serially, rather, they are applied in parallel, it is easier to first state the result and then evaluate the merits.

An algebraic mapping is used, given by

$$r^2 \equiv \frac{r_1^2\,\xi}{1-\xi}, \tag{2.3.12}$$

combined with expansions of the form,

$$f_{n\ell}^{-} \equiv f_{n\ell}^{+} \equiv (1-\xi)^{(\ell+1)/2}\,\xi^{\ell/2}\,G_n^{\ell}(\xi), \tag{2.3.13}$$

where G_n^ℓ are Jacobi polynomials, defined in Appendix C, and $+$ and $-$ functions are the same (and called simply $f_{n\ell}$). Equation (2.3.12) maps the semi-infinite domain, $0 \le r \le \infty$, to the finite domain $0 \le \xi \le 1$. The mapping could be specified in terms of r^2 because the parity of each radial function is known. This mapping (eqn. 2.3.12) has the advantage of alleviating the unnecessary clustering of collocation points near the origin of the spherical coordinates. The constant, r_1, is a free parameter, chosen to minimize the error in the vorticity for a given initial condition. When the resolution is marginal, it is recommended that r_1 be varied by ±10% and the results compared to assure that the solution is not sensitive to this parameter.

Completeness is guaranteed by choosing a set of polynomials which are solutions of a Sturm-Liouville problem (Gottlieb & Orszag, 1977). Furthermore, if the eigenfunctions are solutions of a singular Sturm-Liouville problem, convergence is faster than any power of N (exponential) when approximating an infinitely differentiable function (C_∞) with arbitrary boundary conditions. Laguerre polynomials, defined in a semi-infinite domain, are a possible choice. They are solutions of a singular Sturm-Liouville problem, although their convergence to a function of given complexity requires roughly twice as many polynomials as Legendre and Chebychev polynomials (Gottlieb & Orszag, pg. 42). Therefore, it is better to map the radial coordinate to a finite domain and then, in the mapped domain, expand the solution in terms of Jacobi polynomials (similar to Legendre and Chebychev).

The far-field is the only *boundary* in the problem. The vorticity is assumed to be essentially confined to a finite domain, hence, in the far-field it decays exponentially. This corresponds to a $1/r^3$ decay in the velocity (Batchelor, 1967). This behavior is enforced by the factor $(1-\xi)^p$, through the exponent, p. To find the correct p, we first consider the + modes. The function, $f_{n\ell}$, is the radial dependence of the vector potential. Therefore, to determine the correct behavior of $f_{n\ell}$ at infinity, consider the Poisson equation relating the vector potential to the vorticity,

$$\nabla^2 \left(f_{n\ell}(r)\, \mathrm{X}_{\ell m}(\theta,\phi)\right) = -\omega^+ \tag{2.3.14}$$

Using VSH properties for the Laplacian, and assuming that the vorticity is of the form, $\omega \sim r^q\, \mathrm{X}_{\ell m}(\theta,\phi)$, where q is a large negative exponent (theoretically infinite), equation (2.3.14) becomes

$$L_\ell(f_{n\ell}) = \frac{1}{r^{-\ell+1}}\frac{d}{dr}\left[r^{-2\ell}\frac{d}{dr}(r^{\ell+1}f^+_{n\ell})\right] = O(r^q) \tag{2.3.15}$$

which has a solution of the form,

$$f_{n\ell} = C_1\, r^{-(\ell+1)} + C_2\, r^\ell + O(r^{q+2}) \tag{2.3.16}$$

where C_1 and C_2 are constants, determined by the boundary conditions. Since the solution is bounded at infinity, C_2 is zero. Therefore, the leading term at infinity is $C_1 r^{-(\ell+1)}$. At a large radius, $(1-\xi) \sim r^{-2}$, $\xi \sim 1$, and $G^\ell_n \sim 1$, therefore, for the correct decay

$$p = \frac{(\ell+1)}{2} \tag{2.3.17}$$

With this choice, each of the velocity basis functions decays like $1/r^3$ or faster. The vorticity basis functions also decay algebraically, like $1/r^5$ or faster, and collectively they will approximate exponential decay. A similar argument for the − modes shows that the same factor, $(1-\xi)^{(\ell+1)/2}$, must multiply the Jacobi polynomials in order to satisfy the far field boundary conditions.

At the origin, we must concern ourselves with the coordinate singularity associated with spherical polar coordinates. In spherical coordinates it is possible to specify the radial dependence of a function in such a way that it is a smooth function of r, θ, and ϕ, but not a smooth function of x, y, and z. This occurs because the metric coefficients of the mapping tend to infinity. Therefore, for smoothness, certain constraints are imposed on the radial basis functions. To illustrate this point, consider a simple example. Suppose we have a function, $g(r,\theta)$, in polar coordinates, which is a product:

$$g(r,\theta) = R(r)\,\Theta(\theta) \tag{2.3.18}$$

Now let $R = 1$ and $\Theta = \cos\theta$. It is clear that this function is discontinuous by plotting it along the x axis: if $x < 0$, then $g = 1$, and if $x > 0$, then $g = -1$. Now choose $R = r$, and the function, g, is smooth.

The appropriate constraints on the radial functions when using vector spherical harmonics, derived by Spalart (1988) and summarized in Appendix D, are given by

$$\begin{aligned} f_{1_{\ell m}}(r) &= r^{\ell}\, f_{X\ell m}(r^2) \\ f_{2_{\ell m}}(r) &= r^{\ell+1}\, f_{V\ell m}(r^2) \\ f_{3_{\ell m}}(r) &= r^{\ell-1}\, f_{W\ell m}(r^2) \end{aligned} \tag{2.3.19}$$

These are found by requiring that the vector function be infinitely differentiable near the origin and counting the associated degrees of freedom and constraints, thereby proving necessary and sufficient constraints shown in equation (2.3.19). Here the functions $f_{X\ell m}$, $f_{V\ell m}$, and $f_{W\ell m}$ are themselves infinitely differentiable and bounded for $[0,\infty]$. Note that (2.3.19) dictates both the parity of f_1, f_2, and f_3 and their rate of decay as $r \to 0$ (fast decay for large ℓ).

This analysis is reinforced by the earlier results derived by Cantwell (private communication) where the self-similar solutions of the three-dimensional Stokes equations for the vector vorticity, $\frac{\partial \omega}{\partial t} = \nabla^2 \omega$ (in spherical polar coordinates), are derived. The angular dependence is described by vector spherical harmonics, while the radial dependence involves associated Laguerre functions (Laguerre polynomials multiplied by decaying exponentials). Cantwell's radial functions give the following limiting behavior, as $r \to 0$, for each of the components of vector potential, $\boldsymbol{\Upsilon}$, velocity, and vorticity:

	$\boldsymbol{\Upsilon}^-$	$\boldsymbol{\Upsilon}^+$	$\mathbf{u}^-$	$\mathbf{u}^+$	ω^-	ω^+
$\hat{\imath}_r$	$r^{\ell-1}$	0	0	$r^{\ell-1}$	$r^{\ell-1}$	0
$\hat{\imath}_\theta$	$r^{\ell-1}$	r^{ℓ}	r^{ℓ}	$r^{\ell-1}$	$r^{\ell-1}$	r^{ℓ}
$\hat{\imath}_\phi$	$r^{\ell-1}$	r^{ℓ}	r^{ℓ}	$r^{\ell-1}$	$r^{\ell-1}$	r^{ℓ}

The general result (equation 2.3.19) is in agreement with the behavior of the Stokes solutions. In practice, the constraints at the origin are imposed by including a factor $\xi^{\ell/2}$ in the radial basis functions, $f_{n\ell}$, and choosing an algebraic mapping in terms of r^2. In doing this, the parity requirements are also satisfied.

In summary, the correct behavior in the far field and near the origin is represented with the algebraic mapping of equation (2.3.12) and the factors, $\xi^{\ell/2}$ and $(1-\xi)^{(\ell+1)/2}$ in equation (2.3.13). This form has the drawback that the radial functions depend on ℓ. A careful adjustment of the asymptotic behavior, however, ensures that all the degrees of freedom are useful in resolving the solution. An additional advantage is that more of the improper integrals (over the infinite domain, e.g. the impulse and kinetic energy) will be convergent and have meaningful, finite values.

Numerical efficiency depends on both the number of terms required to represent a typical solution, the length of the time steps that can be taken, and the number of operations required to advance the solution one time step. With VSH, the linear terms are completely uncoupled (orthogonal) in the two angular directions. When the basis functions and the weight functions are the same, the matrices resulting from the radial direction (eqn 2.3.9) are symmetric and positive-definite (section 3.4). With our choice of mapping and radial functions they are also banded, where the number of non-zero elements above the diagonal is 3 and 5, respectively (for every m and ℓ). Bandedness is important not only for efficiency in solving the linear terms in the final set of coupled ODE's but perhaps more importantly, to minimize the round-off error which could ultimately corrupt the solution when many terms are used. In the process of developing the method, several alternatives were considered where these matrices were full, which led to poorly conditioned matrices even at moderate values of L and N.

The azimuthal coordinate is the only direction employing the FFT, in contrast to similar spectral methods where typically two of the three directions are Fourier. The operation count in transforming to and from wave-space for large N is $O(N^4)$. In comparison to other three-dimensional spectral methods using FFT's in two directions and a "slow" transform in the third one (Leonard *et al.,* 1982, Spalart, 1986), the present approach is slower by a factor of two, (not an order of magnitude, as it might first appear). This is acceptable for three-dimensional computations.

Chapter 3

Numerical Procedure

In the previous chapter, the incompressible Navier-Stokes equations were cast as a set of coupled ordinary differential equations. By using spherical polar coordinates, and considering only the $m = 0$ modes, the three-dimensional problem reduces to the axisymmetric problem exactly. This axisymmetric problem was implemented in FORTRAN to run on the CRAY-XMP computer. This chapter discusses the practical aspects of solving these equations including time advancement, transforms to and from wave-space, Cholesky decomposition and its use, aliasing, and initial conditions.

The axisymmetric basis functions are extracted from the three-dimensional functions by setting $m = 0$ (axisymmetric flow) and imposing $u_\phi = 0$ (no swirl velocity) in the velocity expansion (eqn. 2.3.5). One finds that the axisymmetric solution is described by the + modes alone (eqn. 2.3.8). From this point forward, we will only discuss the axisymmetric problem, so the + symbol is dropped and m is set to zero, simplifying the notation (i.e. $a^+_{n\ell m} \equiv a_{n\ell}$, $f^+_{n\ell} \equiv f_{n\ell}$, $\Psi^+_{n'\ell m} \equiv \Psi_{n'\ell}$, etc.).

3.1 Time Advancement Schemes

In Chapter 2, the complete equations were derived for the three-dimensional problem. Assuming the flow is axisymmetric, these simplify to the equations governing the + modes:

$$A_{n'n}(\ell)\frac{da_{n\ell}}{dt} - \nu B_{n'n}(\ell)\; a_{n\ell} = Q_{n'}(\ell) \qquad (3.1.1)$$

where $Q_{n'}(\ell) = < \mathbf{u}\times\omega, \nabla\times(f^{+}_{n'\ell}\mathbf{X}^{*}_{\ell m}) >$. This is a set of $N(\ell)\times N(\ell)$ equations for each ℓ. Throughout this work, the matrices, $A_{n'n}(\ell)$ and $B_{n'n}(\ell)$, are referred to as the mass and viscous matrix, respectively. The vector representing the non-linear term, $Q_{n'}(\ell)$, acts as a forcing to the linear equation, and is integrated explicitly in time using Adams-Bashforth (AB), a second-order scheme.

Applying an explicit solver for the viscous term in a non-Fourier spectral method severely limits the maximum step size allowed by the stability criterion. An example of this is the solution of the heat equation between two walls using a Chebyshev-spectral method (Gottlieb and Orszag, pg. 115); for this problem, the stability limit gives $\Delta t = O(1/N^4)$ as $N \to \infty$. It is important, therefore, that the viscous term be treated implicitly for non-Fourier expansions. Because the semi-bandwidth of the mass matrix is 3 and that of the viscous matrix is 5, and the mass matrix needs to be inverted anyway, an implicit scheme for the viscous term brings only a moderate penalty over an explicit scheme (Leonard and Wray, 1982). The Crank-Nicolson scheme, used to integrate the viscous term, is also second-order accurate in time. Therefore, the overall time-integration scheme is second-order accurate.

The time-differenced form of equation (3.1.1) is thus:

$$A_{n'n}(\ell)\left(\frac{a^{j+1}_{n\ell} - a^{j}_{n\ell}}{\Delta t}\right) = B_{n'n}(\ell)\left(\frac{a^{j+1}_{n\ell} + a^{j}_{n\ell}}{2}\right) + \frac{3}{2}Q^{j}_{n'\ell} - \frac{1}{2}Q^{j-1}_{n'\ell} \qquad (3.1.2)$$

where j indicates the time level: $t = j\Delta t$. Defining $\Delta a^{j}_{n\ell} \equiv a^{j+1}_{n\ell} - a^{j}_{n\ell}$, equation (3.1.2) may be written in "delta form" as:

$$A^{\dagger}_{n'n}(\ell)\,\Delta a^{j}_{n\ell} = B^{\dagger}_{n'n}(\ell)a^{j}_{n\ell} - (3Q^{j}_{n'\ell} - Q^{j-1}_{n'\ell})\,\Delta t \qquad (3.1.3)$$

where $A^{\dagger}_{n'n}(\ell) = -[2\,A_{n'n}(\ell) - \Delta t\,B_{n'n}(\ell)]$ and $B^{\dagger}_{n'n}(\ell) = -2\,\Delta t\,B_{n'n}(\ell)$.

Two codes are used to solve these equations. The first code, *IC,* takes an initial condition, the velocity or vorticity field at an instant in time, and computes the coefficients of the spectral expansion. Using these coefficients, the second program, *NS*, integrates the coupled set of ODE's (eqn. 3.1.3) forward in time from the initial state.

In *NS*, the mass and viscous matrices are computed once and stored. With this, the matrices $A^{\dagger}_{n'n}(\ell)$ and $B^{\dagger}_{n'n}(\ell)$ are then computed and stored in place of $A_{n'n}(\ell)$ and $B_{n'n}(\ell)$. Next, the implicit matrix, $A^{\dagger}_{n'n}(\ell)$, is decomposed with the Cholesky method and replaced with its decomposed form. The matrices are symmetric and banded, requiring a total storage space of $6NL$ each. The time integration proceeds as follows:

(a) Using $a^j_{n\ell}$, compute $Q^j_{n'\ell}$ for all n' and ℓ (by a pseudospectral method)

(b) For each ℓ, compute $B^\dagger_{n'n}(\ell)a^j_{n\ell}$ and add it to $-(3Q^j_{n'\ell} - Q^{j-1}_{n'\ell})\Delta t$.

(c) For every ℓ, multiply through by $[A^\dagger_{n'n}(\ell)]^{-1}$ to obtain $\Delta a^j_{n\ell}$, hence $a^{j+1}_{n\ell}$.

(d) Save $Q^j_{n'\ell}$ in place of $Q^{j-1}_{n'\ell}$.

(e) Advance in time, save $a^{j+1}_{n\ell}$ in array $a^j_{n\ell}$, and return to (a).

The following sections more carefully describe these steps; transforming to and from wave space (section 3.3), forming the mass and viscous matrix elements (section 3.4), inverting $A^\dagger_{n'n}(\ell)$ using Cholesky decomposition (section 3.5), computing the non-linear term (section 3.6), and starting the initial conditions (section 3.7).

3.2 Axisymmetric Basis Functions

The velocity expansion for axisymmetric flow is given by

$$\begin{aligned}\mathbf{u} &= \sum_{\ell=1}^{L}\sum_{n=0}^{N(\ell)} a_{n\ell}\, c_\ell\, \nabla\times(f_{n\ell}\mathbf{X}_\ell) \\ &= \sum_{\ell=1}^{L}\sum_{n=0}^{N(\ell)} a_{n\ell}\, c_\ell (f_{2_{n\ell}}\mathbf{V}_\ell + f_{3_{n\ell}}\mathbf{W}_\ell) = \sum_{\ell=1}^{L}\sum_{n=0}^{N(\ell)} a_{n\ell}\,\mathbf{\Psi}_{n\ell}\end{aligned} \tag{3.2.1}$$

where a constant has been inserted for convenience, $c_\ell \equiv -i\left[\frac{4\pi\ell(\ell+1)}{2\ell+1}\right]^{1/2}$. The radial basis functions are specified through the mapping,

$$r^2 = \frac{r_1^2\,\xi}{1-\xi} \tag{3.2.2}$$

and the radial function,

$$f_{n\ell} = (1-\xi)^{(\ell+1)/2}\xi^{\ell/2}G^\ell_n(\xi). \tag{3.2.3}$$

Recall that $f_{2_{n\ell}}$ and $f_{3_{n\ell}}$ are related to $f_{n\ell}$ by

$$\begin{aligned} f_{2_{n\ell}} &= i\left(\frac{\ell}{2\ell+1}\right)^{1/2}\left[\frac{d}{dr} - \frac{\ell}{r}\right] f_{n\ell} \\ f_{3_{n\ell}} &= i\left(\frac{\ell+1}{2\ell+1}\right)^{1/2}\left[\frac{d}{dr} + \frac{(\ell+1)}{r}\right] f_{n\ell}.\end{aligned} \tag{3.2.4}$$

Jacobi polynomials, denoted by $G_n^\ell(\xi)$, are polynomials of order n. Their definition is modified from that of Abramowitz and Stegun (1972, eqn. 22.2.2) as follows: from the notation of A&S, start with shifted Jacobi polynomials, $G_n(p,q,x)$, normalize them such that $h_n = 1$ (i.e., orthonormal polynomials), and set p and q equal to $2\ell - 1$ and $\ell - \frac{1}{2}$ (note that p and q used here are not related to the notation of Chapter 2). The choices of p and q were determined by constraining the matrices, A and B, to be banded. Note that if we had a scalar field and did not apply the operators (2.3.10) and (2.3.11), and L_ℓ, the matrix would be diagonal.

The components of the basis functions for the velocity reduce to:

$$\mathbf{u} = \begin{bmatrix} u_r \\ u_\theta \\ u_\phi \end{bmatrix} = \sum_{\ell=1}^{L} \sum_{n=0}^{N(\ell)} a_{n\ell} \begin{bmatrix} \Psi_{n\ell r} \\ \Psi_{n\ell\theta} \\ \Psi_{n\ell\phi} \end{bmatrix} = \sum_{\ell=1}^{L} \sum_{n=0}^{N(\ell)} a_{n\ell}(t) \begin{bmatrix} \ell(\ell+1) \dfrac{f_{n\ell}(r)}{r} P_\ell^0(\mu) \\ \dfrac{1}{r}\dfrac{d}{dr}\left(r f_{n\ell}(r)\right) P_\ell^1(\mu) \\ 0 \end{bmatrix} \tag{3.2.5}$$

by substituting c_ℓ and the definitions for $\mathbf{V}_\ell$ and $\mathbf{W}_\ell$ (Appendix C) into equation (3.2.1), and simplifying. The functions $P_\ell^0(\mu)$ and $P_\ell^1(\mu)$ are the associated Legendre functions, $P_\ell^m(\mu)$ (Arfken, 1985), where $m = 0$ and $m = 1$, respectively, and $\mu = \cos\theta$. Their properties are summarized in Appendix C.

The vorticity only has its azimuthal component:

$$\omega_\phi = (\nabla \times \mathbf{u})_\phi = \sum_{\ell=1}^{L} \sum_{n=0}^{N(\ell)} a_{n\ell}\, c_\ell \; [\nabla \times \nabla \times (f_{n\ell}\mathbf{X}_\ell)]_\phi \tag{3.2.6}$$

Applying the definitions of c_ℓ and $\mathbf{X}_\ell$ (Appendix C), and assuming axisymmetric flow without swirl, equation (3.2.6) reduces to

$$\omega_\phi = \sum_{\ell=1}^{L} \sum_{n=0}^{N(\ell)} a_{n\ell}(t)\, P_\ell^1(\mu)\, L_\ell\left(f_{n\ell}(r)\right) \tag{3.2.7}$$

Recall that the definition of the operator, L_ℓ, from Chapter 2 is

$$L_\ell(f_{n\ell}) \equiv \left[\frac{d^2}{dr^2} + \frac{2}{r}\frac{d}{dr} - \frac{\ell(\ell+1)}{r^2}\right] f_{n\ell} \tag{3.2.8}$$

From equation (3.2.5), the radial dependence of the velocity components are given by $f_{n\ell}(r)/r$. This can be expressed as a polynomial in ξ of order $n + \ell + \frac{1}{2}$. For efficiency (i.e. so that we don't carrying useless degrees of freedom), we choose $N(\ell)$ so that each expansion is the same order in ξ for every ℓ. Therefore, by using a constant and sufficient number of collocation points, integrals of products of the functions (as in computing the nonlinear term) will be exact for every ℓ. The result is $N(1) + 1 + \frac{1}{2} = N(\ell) + \ell + \frac{1}{2}$, or $N(\ell) = N(1) - \ell + 1$.

3.3 Transforming To/From Wave-Space

Transforming from real-space to wave-space, given the coefficients of the basis functions, is done by evaluating summations at a finite number of grid or collocation points, r_i and θ_j, where $i = 1, \ldots, N_c$ and $j = 1, \ldots, L_c$. Transforming to wave-space from real-space involves numerical evaluation of definite integrals where the integrands are known at the collocation points. Two such examples are evaluating the non-linear terms, $Q_{n'}(\ell)$, and finding the expansion coefficients for an initial condition. This section discusses the theory and practice of transforming to and from wave-space for orthogonal functions.

Like the matrices, the basis functions (evaluated at each collocation point) are computed once and stored. The radial basis functions corresponding to u_r, u_θ, and ω_ϕ are

$$\begin{aligned} GR(n,\ell,i) &\equiv \ell(\ell+1)\,\frac{f_{n\ell}(\xi_i)}{r_i} \\ GTH(n,\ell,i) &\equiv \frac{1}{r_i}\,\frac{d}{dr}(r_i f_{n\ell}(\xi_i)) \\ GV(n,\ell,i) &\equiv L_\ell(f_{n\ell}(\xi_i)) \end{aligned} \tag{3.3.1}$$

where ξ_i is the i^{th} grid point in the mapped radial coordinate, ξ. The polar basis functions are

$$\begin{aligned} P0(\ell,j) &\equiv P_\ell^0(\mu_j) \\ P1(\ell,j) &\equiv P_\ell^1(\mu_j) \end{aligned} \tag{3.3.2}$$

and μ_j is the j^{th} grid point in the polar direction where $\mu = \cos\theta$.

Before evaluating equations (3.3.1), the derivatives are replaced with sums of the neighboring Jacobi polynomials (i.e. G_{n+1}^ℓ and G_{n-1}^ℓ, cf., Appendix B). The derivatives in equation (3.3.1) could either be evaluated numerically or computed analytically. It is preferable to compute them analytically in order to reduce the round-off error (see section 3.5). Due to the mapping, the second derivatives in L_ℓ, and the complicated recurrence relations of the two parameter family of Jacobi polynomials, the algebra is quite involved. By hand, it would be extremely tedious and time consuming. With the aid of a symbolic algebra program, MACSYMA, this task is greatly simplified, but is still a major effort. MACSYMA is also used to analytically compute the elements of the mass and viscous matrices in equations (3.4.1) and (3.4.3).

The transforms in vorticity and velocity in terms of the stored arrays are given by

$$\begin{aligned} u_r(r_i,\theta_j) &= \sum_{\ell=1}^{L}\sum_{n=1}^{N(\ell)} a_{n\ell}\, GR(n,\ell,i)\, P0(\ell,j) \\ u_\theta(r_i,\theta_j) &= \sum_{\ell=1}^{L}\sum_{n=1}^{N(\ell)} a_{n\ell}\, GTH(n,\ell,i)\, P1(\ell,j) \\ \omega_\phi(r_i,\theta_j) &= \sum_{\ell=1}^{L}\sum_{n=1}^{N(\ell)} a_{n\ell} GV(n,\ell,i)\, P1(\ell,j) \end{aligned} \tag{3.3.3}$$

Assuming $L_c = N_c$, it appears at a first glance, that $O(N_c^4)$ operations are required to evaluate equations (3.3.3). The operation count is reduced to $O(N_c^3)$ because the G functions do not depend on j and the P functions do not depend on i, as shown in figure G.3. Ordering the loops in this way, also turns out to be well suited for vector processing on the CRAY-XMP.

We wish to evaluate the integral of a function, $g(x)$ over the interval $[a,b]$ using Gauss quadrature. We begin by writing $g(x)$ as the product of a new function, $P(x)$, and a specified weight function, $w(x)$ (determined by the interval $[a,b]$). For example, in Gauss-Laguerre quadrature, the weight function is e^{-x} and the interval is $[0,\infty]$. The definite integral is then approximated as a discrete sum by

$$\int_a^b g(x)\,dx \equiv \int_a^b P(x)\,w(x)\,dx \approx \sum_{i=1}^{N_c} P(x_i)\,A_i \tag{3.3.4}$$

where A_i are the weights (not to be confused with the weight functions, $w(x)$), and x_i are the collocation points. The degrees of freedom are A_i, and x_i, where $i = 1,\ldots,N_c$, totaling $2\,N_c$ (Appendix E). If the function $P(x)$ in equation (3.3.4) is a polynomial of order $2N_c - 1$ (the space of such polynomials also has dimension $2N_c$), then the discrete sum describing the integral is exact (to within machine precision) with N_c collocation points.

The power of Gauss quadrature together with spectral methods, is that we can often integrate expressions exactly by using Gauss-type quadrature rules which are compatible with our expansion functions.

3.4 Forming the Mass and Viscous Matrices

The mass and viscous matrices result from the coupling of the radial modes. Because they are not time-dependent, they can be computed once and stored. This section will discuss the properties of the matrices and the method used to compute them.

In Chapter 2, we developed the mass and viscous matrices,

$$A_{n'n}(\ell) = \int_0^\infty \left[f_{2_{n\ell}} f_{2_{n'\ell}} + f_{3_{n\ell}} f_{3_{n'\ell}}\right] r^2 dr \tag{3.4.1}$$

$$B_{n'n}(\ell) = \int_0^\infty \left[L_{\ell+1}(f_{2_{n\ell}}) f_{2_{n'\ell}} + L_{\ell-1}(f_{3_{n\ell}}\ ) f_{3_{n'\ell}}\right] r^2 dr \tag{3.4.2}$$

which are functions of the radial functions, $f_{n\ell}$, and mapping, $\xi(r)$. These were chosen such that A and B have semi-bandwidths of 3 and 5, respectively. With banded matrices, the operations count per time step is much lower; there is also less round-off error (see section 3.5).

It is clear from equation (3.4.1) that A is symmetric and positive-definite, however, this is not as clear for equation (3.4.2). Before evaluating the viscous matrix, it is first simplified by integrating by parts. The result,

$$\begin{aligned} B_{n'n}(\ell) = -\Bigg\{ &\int_0^\infty \frac{df_{2_{n\ell}}}{dr} \frac{df_{2_{n'\ell}}}{dr} r^2 dr + (\ell+1)(\ell+2) \int_0^\infty f_{2_{n\ell}} f_{2_{n'\ell}} dr \\ &+ \int_0^\infty \frac{df_{3_{n\ell}}}{dr} \frac{df_{3_{n'\ell}}}{dr} r^2 dr + \ell(\ell-1) \int_0^\infty f_{3_{n\ell}} f_{3_{n'\ell}} dr \Bigg\} \end{aligned} \tag{3.4.3}$$

is clearly symmetric and negative-definite. As mentioned earlier, L_ℓ is self-adjoint. This form (eqn. 3.4.3) is preferred from a computational standpoint since the order of the derivatives is reduced by one. Again, MACSYMA, was used here.

MACSYMA is a high level programming language. It symbolically manipulates expressions, and has an extensive library of algorithms which, among other things, evaluates derivatives and integrals. One can either interactively enter commands to the program, or submit a list of commands in the form of a batch job. The program, written in LISP, is recursive in nature, and therefore, even for moderate problems can quickly use very large fractions of the disk space of a VAX 11-780. Much of the difficulty encountered was due to running out of disk space.

The end result, the diagonal and off-diagonal expressions as a function on n and ℓ, are written directly in FORTRAN (by MACSYMA). A single expression is up to 15 lines long, giving an idea of the complexity of the algebra. The MACSYMA codes are described and listed in Appendix G.

3.5 A Few Words About Stiffness

In a viscous flow, the ratio of largest to smallest length scales is a function of the Reynolds number of the flow. This, in turn, is reflected in the ratio of the largest to the smallest eigenvalues of the matrices. As the ratio of eigenvalues becomes large, the time integration becomes increasingly difficult. This behavior is what we call stiffness. By using implicit time advancement for the viscous term, we have helped to alleviate some of the problem. The stability limit for the time step is not as severe as it otherwise would be, although we still must contend with round-off errors. Round-off error comes from several sources – from forming the matrix elements, and from advancing the solution in time, in particular it is a function of the number of operations per time step. Several steps are taken to keep these errors to a minimum.

The number of operations per time step is determined largely by the matrix structure of the system being solved. For example, a full matrix requires $O(N^3)$ operations initially and $O(N^2)$ at each time step, while a banded matrix requires $O(m^2N)$ operations, where m is the number of non-zero elements above the diagonal. For the problem at hand, the matrix structure is even more specialized – it's banded, symmetric and positive-definite – leading to even greater savings. The algorithm which exploits this particular matrix form is called Cholesky decomposition (see Golub & van Loan, 1984). The round-off error of this algorithm has been studied rigorously by Wilkinson (1968) and shown to be very low.

An important consideration, therefore, in forming a numerical method is to consider the conditioning of the system of equations which are solved. Minimizing the number of operations not only helps the efficiency but also the round-off error. By choosing weight functions to be the same as the basis functions, the resulting matrices are positive-definite and symmetric. It is also very helpful to minimize the bandwidth of the matrices by careful choice of the basis functions.

Round-off error also enters the problem in computing the matrix elements. Again, using MACSYMA, the analytic expressions for the elements were found. The result is an expression for the diagonal, and off-diagonals, in terms of n and ℓ. These expressions are evaluated once and stored, so it is not a large penalty to evaluate them in double precision, and store them in single precision, thus obtaining the correct result to every significant digit.

3.6 Computing the Non-Linear Term

From equation (2.3.8), the non-linear term is

$$Q_{n'}(\ell) = < \mathbf{u} \times \omega, \Psi_{n'\ell} > \frac{1}{c_\ell^2} \tag{3.6.1}$$

One way to compute this term is spectrally. By substituting the expansions for velocity and vorticity, the time-dependence is separated in the usual way from the spatial dependence giving integrals of triple products of the basis functions. Each time step would then involve a convolution sum. Even if the integrals of basis functions could be evaluated analytically, a convolution sum is more expensive than a pseudospectral approach where the coefficients are transformed to real space, the non-linear product is formed, and the result is transformed back to wave space. This led us to adopt the pseudospectral method.

To compute $Q_{n'}(\ell)$ pseudospectrally, we first write equation (3.6.1) in component form,

$$Q_{n'}(\ell) = \frac{2\pi}{c_\ell^2} \int_0^\infty \int_0^\pi \left[(u_\theta \, \omega_\phi) \, \Psi_{n'\ell r} - (u_r \, \omega_\phi) \, \Psi_{n'\ell\theta} \right] r^2 \, \sin\theta dr d\theta \qquad (3.6.2)$$

then equation (3.6.2) is expressed as successive transformations in each direction (note that this was also the case for eqn. 3.3.3). With $\mathbf{q} = \mathbf{u} \times \omega$, the nonlinear term can be written as

$$Q_{n'}(\ell) = \frac{2\pi}{c_\ell^2} \left(\widehat{\widehat{q}}_r(n',\ell) - \widehat{\widehat{q}}_\theta(n',\ell) \right) \qquad (3.6.3)$$

where the polar transform is

$$\begin{aligned} \widehat{q}_r(r,\ell) &\equiv \int_{-1}^{1} (u_\theta \omega_\phi) \, P_\ell^0(\mu) \, d\mu \\ \widehat{q}_\theta(r,\ell) &\equiv \int_{-1}^{1} (u_r \omega_\phi) \, P_\ell^1(\mu) \, d\mu \end{aligned} \qquad (3.6.4)$$

and the radial transform is

$$\begin{aligned} \widehat{\widehat{q}}_r(n',\ell) &\equiv \int_0^\infty \widehat{q}_r(r,\ell) \, \ell(\ell+1) \, \frac{f_{n\ell}}{r} \, r^2 dr \\ \widehat{\widehat{q}}_\theta(n',\ell) &\equiv \int_0^\infty \widehat{q}_\theta(r,\ell) \, \frac{1}{r} \frac{d}{dr}(r f_{n\ell}) \, r^2 dr \end{aligned} \qquad (3.6.5)$$

The integrals are then cast as sums using Gauss-Legendre and Gauss-Jacobi quadrature (Appendix E).

When using a pseudospectral method, we must concern ourselves with aliasing – the phenomena of higher frequencies "masquerading" as lower frequencies because a continuous function is sampled only at discrete points and described by a Fourier series. With polynomials, errors creep in for the same reason, but you cannot say that "one polynomial is mistaken for another." Typically in a pseudospectral method, the number of collocation points is equal to the order of the approximating polynomials. In forming a triple product of the basis functions (as in the nonlinear term above), the result will have aliasing errors. This occurs because the product of functions produces a new function with frequencies that are higher than the original functions. If the original function is barely resolved, then the polynomial approximation will not resolve the product and the higher frequencies will appear as lower frequencies. With Fourier series, one way to remove this error (or dealias) is to pack the coefficients in spectral space with zeros, transform to real space with a larger number of points, perform the non-linear product at the sample points, transform back to wave-space, and discarding the higher coefficients (previously added). The number of points added and removed is $(1/2)\,N$, giving a total of $(3/2)N$. The equivalent process for other polynomial expansions is done by choosing $N_c = (3/2)N$. In this way, the integral expression, involving products of three polynomials of up to order N (as in eqn. 3.6.2), is integrated exactly with Gauss quadrature. If the flow is smooth and well resolved, however, dealiasing may not be necessary since the higher modes which are interpreted as lower modes have a very small contribution. To avoid aliasing errors, therefore, either the flow must be over-resolved or we must use $3/2\,N$ collocation points.

3.7 Initial Conditions

To start a calculation, only one scalar component of the vorticity needs to be specified over the infinite domain. One could also start from a velocity field, but specifying such a field requires two components and a constraint. Thus, in practice a vorticity-based initialization is much more powerful. Two initial distributions have been used – the Stokes vortex ring, and a thin ring with a Gaussian vorticity distribution in the core.

3.7.1 Stokes Vortex Ring

Exact solutions to the diffusion equation for vorticity in spherical polar coordinates are developed by Cantwell (1986) for several different forcings. The solution of interest here is that resulting from an impulsive point force, leading to what we will refer to as the Stokes vortex ring. This is a self-similar solution in time with similarity variable, $\eta = r/\sqrt{4\nu t}$. This solution is a convenient starting point in our calculations for several reasons.

Given such an exact solution, we are able to validate the Stokes part of the Navier-Stokes solver (see section (4.2.1)). In solving the Stokes equations, the angular direction is exactly represented by the first mode in θ $(\ell = 1)$; the radial modes are the only non-trivial terms in the expansion.

The vorticity distribution for a Stokes vortex ring is

$$\omega(\eta,\theta) = \frac{I/\rho}{8\pi^{3/2}(\nu\, t)^2}\ \sin\theta\,\eta\,\exp^{-\eta^2} \tag{3.7.1}$$

and the radial and polar components of velocity are

$$u_r(\eta,\theta) = \frac{-I/\rho}{4\pi(\nu\, t)^{3/2}}\ \cos\theta\ \frac{1}{\eta}\left[\frac{1}{2\sqrt{\pi}\,\eta}\,e^{-\eta^2} - \frac{1}{4\eta^2}\mathrm{erf}(\eta)\right] \tag{3.7.2}$$

$$u_\theta(\eta,\theta) = \frac{-I/\rho}{8\pi(\nu\, t)^{3/2}}\ \sin\theta\ \frac{1}{\eta}\left[\frac{1}{\sqrt{\pi}}\left(\frac{1}{2\eta}+\eta\right)e^{-\eta^2} - \frac{1}{4\eta^2}\,\mathrm{erf}(\eta)\right] \tag{3.7.3}$$

As Cantwell points out, the Stokes solution is uniformly valid over the full range $0 \le \eta < \infty$.

The Stokes equations are a limiting form of the Navier-Stokes equations for a Reynolds number approaching zero, where the appropriate Reynolds number is

$$Re_I = \frac{(I/\rho)^{1/2}}{\nu {t_I}^{1/2}} \tag{3.7.4}$$

Note that the Reynolds number is inversely proportional to time. Starting at $t_I = 0$, an impulsive force forms a turbulent puff (with $Re_I = \infty$) of the type studied by Glezer (1981), and after a very large time ($t_I \to \infty$), the vorticity and velocity fields approach the Stokes distributions given by equations (3.7.1)-(3.7.3).

The free parameters are found by transforming the vorticity to computational variables using equations (2.1.5) and (2.2.8),

$$\tilde{\omega} = \frac{1}{8\pi^{3/2}\,\bar{t}_I} \sin\tilde{\theta}\,\tilde{r}\exp^{-\tilde{r}^2} \tag{3.7.5}$$

From this, we see that the time, $\bar{t}_I$, is the only parameter in the solution. Also, the time in the code, $\bar{t}$, is set to $\bar{t}_I$ (or Re_I).

There is nothing that prevents us from specifying, for our initial state, a Stokes solution which is at a Reynolds number outside of the range of validity of the Stokes equations ($Re_I > 1$). This is in fact what is done in the results section (5.2.1) where the Navier-Stokes equations are solved with an initial Stokes distribution of Reynolds numbers 30 and 40. Therefore, the starting solution contains only the $\ell = 1$ mode, and the other modes are excited through the non-linear terms. As time progresses, the solution decays and returns to the Stokes solution. This is the simplest initial condition, since it is characterized by a single parameter, Re_I.

Note from equation (3.7.5) that contours of $\tilde{\omega}\bar{t}$ in expanding coordinates, $\tilde{r}$, are steady. This is convenient in studying the long time behavior; we can study how the non-linear solution approaches the Stokes solution.

3.7.2 Thin Ring

Note that any function, $\omega_\phi(r,\theta)$ is a valid *initial* condition since in the unsteady case the time derivative, $\frac{\partial \omega_\phi}{\partial t}$, is free. In this section, we describe a vorticity distribution in the core which is easy to specify – a Gaussian. It has the additional advantage that for a thin viscous ring, it is the first order solution to the vorticity distribution in the core for a solution which has been evolving according to the Navier-Stokes equations. Therefore, it is reasonable as a first guess.

In the core of a thin ring, where the core radius is small compared to the ring radius, the solution is locally two-dimensional. Furthermore, to a good approximation near the core the streamlines are circular. Applying these assumptions to the Navier-Stokes equations in polar coordinates, s and β, the momentum equations become

$$\frac{\rho u_\beta^2}{s} \approx \frac{\partial p}{\partial s} \tag{3.7.6}$$

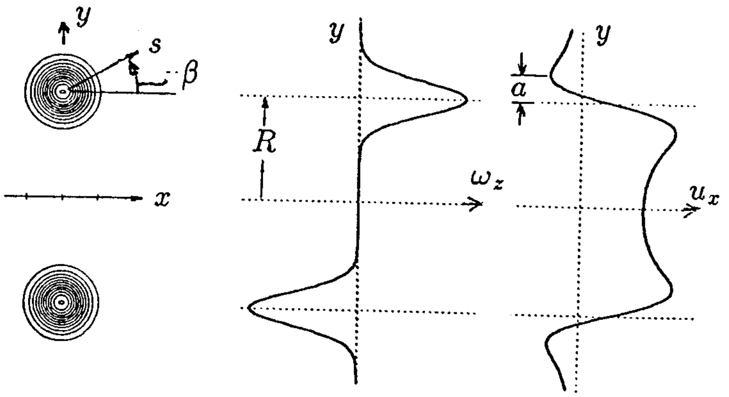


Figure 3.1. Schematic of a thin ring with Gaussian vorticity distribution

$$\frac{\partial u_\beta}{\partial t} \approx \nu \left(\frac{\partial^2 u_\beta}{\partial s^2} + \frac{1}{s}\frac{\partial u_\beta}{\partial s} - \frac{u_\beta}{s^2} \right) \tag{3.7.7}$$

where s is the distance from the vortex center. Equation (3.7.6) is a constraint of the radial pressure gradient which must be maintained to have circular streamlines. Rewriting equation (3.7.7) with the vorticity as the dependent variable, we have

$$\frac{\partial \omega}{\partial t} = \nu \left(\frac{\partial^2 \omega}{\partial s^2} + \frac{1}{s}\frac{\partial \omega}{\partial s} \right) \tag{3.7.8}$$

giving the classical Oseen solution (1910):

$$\omega(s, t_\Gamma) = \frac{\Gamma}{4\pi\, \nu t_\Gamma} \exp\left(\frac{-s^2}{4\, \nu t_\Gamma} \right) \tag{3.7.9}$$

where Γ is the circulation in a meridional half-plane. It follows that the tangential velocity distribution is

$$u_\beta(s, t_\Gamma) = \frac{1}{s}\int_0^s \omega s ds = \frac{\Gamma}{2\pi\, s}\left\{ 1 - \exp\left(-\frac{s^2}{4\, \nu t_\Gamma} \right) \right\} \tag{3.7.10}$$

At the origin of time, $t_\Gamma = 0$, the vorticity is concentrated on a line, $s = 0$, and as it spreads it takes the shape of a Gaussian. The core radius, a, is defined as the distance from the center of the vortex to the peak radial velocity. This is found from equation (3.7.10) by setting $\frac{du_\beta}{ds} = 0$, and solving for a, giving $a = 2.24182\sqrt{\nu t}$.

To describe a thin ring, the two-dimensional vortex is offset a distance R from the axis of symmetry as shown in figure 3.1. The distance from the core is therefore $s = (R^2 + r^2 - 2\, R r\, \sin\theta)^{1/2}$, and the total vorticity distribution is

$$\omega = \frac{K}{\pi}\frac{\Gamma}{a^2} \exp\left\{ -K \left(\frac{R^2}{a^2} + \frac{r^2}{a^2} - \frac{2R}{a}\frac{r}{a}\sin\theta \right) \right\} \tag{3.7.11}$$

where $K = (2.24182)^2/4$. To assure that the initial distribution was smooth, an image ring was placed across the axis of symmetry so that $\omega_\phi \equiv 0$ on the axis. In computational coordinates this becomes

$$\widetilde{\omega} = \frac{K}{\pi}\Gamma/\nu \frac{1}{\widetilde{R}^2\,(a/R)^2} \exp\left\{ -\frac{K}{(a/R)^2} \left(1 + \frac{\widetilde{r}^2}{\widetilde{R}^2} - \frac{2\,\widetilde{r}\,\sin\widetilde{\theta}}{\widetilde{R}} \right) \right\} \tag{3.7.12}$$

This initial condition is characterized by three parameters, $Re_\Gamma \equiv (\Gamma/\nu)_o$, $(a/R)_o$, and the ring radius location in computational variables, $\widetilde{R}_o$. For thin rings, $\widetilde{R}_o$ is inconsequential in the description of the initial condition in physical coordinates. Furthermore, given $\widetilde{R}_o$, the collocation points are clustered about this radius. In describing the Stokes solution, however, this is not a free parameter. Therefore, for computing flows for long periods of time, we pick $\widetilde{R}_o$ which can resolve the limiting solution well, the Stokes vortex ring. In the Stokes solution, the ring radius is not well defined; the peak of vorticity ($\widetilde{R} \approx 2$) and the point where $u_\theta = 0$ ($\widetilde{R} = 3$) are not the same. Experience showed that the latter gave a more accurate approximation of the solution, therefore, for all simulations, $\widetilde{R}_o = 3$.

The appropriate choice of the initial value of time, $\bar{t}_o$, for the thin ring initial condition must be consistent with $\widetilde{R} = 3$ and is found from the expression for impulse

$$I/\rho = \pi \int_0^\pi \int_0^\infty \omega_\phi (r \sin\theta)^2 \, r \, dr d\theta \tag{3.7.13}$$

Rewriting equation (3.7.13) in computational variables, gives

$$1 = \bar{t}\, \pi \int_0^\pi \int_0^\infty \widetilde{\omega}_\phi (\widetilde{r} \sin\widetilde{\theta})^2 \, \widetilde{r} \, d\widetilde{r} d\widetilde{\theta} \tag{3.7.14}$$

With equation (3.7.12), the vorticity ($\widetilde{\omega}_\phi$) is specified, therefore, from equation (3.7.14) $\bar{t}$ is determined.

3.8 Reynolds Numbers and Virtual Origins

There are many ways to define a Reynolds number for a vortex ring. Experimentalists often use $\frac{UD}{\nu} \equiv Re_D$, where U is the propagation speed, D is the diameter of the ring, and ν is the kinematic viscosity. Alternatively one might use $\frac{(I/\rho)^{1/2}}{\nu t_I^{1/2}} \equiv Re_I$, where I/ρ is impulse, and $t_I \equiv 0$ is a virtual origin in time when the ring was created by an impulse force (Cantwell, 1987). Still another measure is $\Gamma/\nu \equiv Re_\Gamma$. Because the flow is viscous, Γ decreases with time. For rings with infinitesimal cores however, the cancellation of vorticity across the axis is exponentially small and for a time, $\Gamma \approx$ constant. Locally, in the core, the vorticity is diffusing like a two-dimensional Oseen vortex. The origin in time for this case, $t_\Gamma \equiv 0$, refers to the concentration of vorticity along a circular line.

Each of these Reynolds numbers is more appropriate in different circumstances. Expressions can be derived giving approximate relationships of these Reynolds numbers to one another. For example, using $I/\rho \approx \pi \Gamma(t_\Gamma) R(t_\Gamma)^2$ (which assumes that the core is thin), and $a \approx 2.24(\nu t_\Gamma)^{1/2}$ (from an Oseen solution), we get

$$Re_I \approx 2.24\sqrt{\pi} \left(\frac{\Gamma}{\nu}\right)^{1/2} \left(\frac{R}{a}\right) \tag{3.8.1}$$

Note that in equation (3.8.1), t_I was replaced by t_Γ. It gives an approximate relationship between Re_I and Re_Γ (together with a/R).

Next, we express $Re_D(Re_\Gamma, a/R)$ by using the expression for the propagation speed of viscous vortex rings (Saffman, 1970, i.e. equation (5.2.10)), and again, $a \approx 2.24(\nu t_\Gamma)^{1/2}$, giving

$$\frac{UR}{\Gamma} \approx \frac{1}{4\pi}\left[\ln\left(4(2.24)\frac{R}{a}\right) - 0.558\right] \tag{3.8.2}$$

Rearranging (3.8.2) gives

$$Re_D \equiv \frac{UD}{\nu} \approx \frac{\Gamma}{\nu}\cdot\frac{1}{2\pi}\left[\ln\left(8.96\frac{R}{a}\right) - 0.558\right]. \tag{3.8.3}$$

In the results, we have scaled the time in two different ways. This final paragraph discussed these scalings and their relationships. The scaling more frequently used in this work is $\frac{t\nu^2}{I/\rho} \equiv \bar{t}$. When there isn't a subscript, $(\;)_I$ or $(\;)_\Gamma$, the origin in time is determined by the method described in section 3.7.2. The second scaling, $\frac{t\,\nu}{R^2}$, is used in section 5.2 to study thin rings, and the virtual origin is chosen in order to compare the calculation with theory, giving $\frac{t_\Gamma \nu}{R^2}$. In order to relate t_Γ to the time in the calculation, we must find $\frac{\nu t_\Gamma}{R^2}(\bar{t}, \bar{t}_o, \widetilde{R}_o)$.

From (2.1.5), we can write

$$\frac{\nu t}{R^2} = \frac{\bar{t}}{\overline{R}^2}. \tag{3.8.4}$$

Since $\overline{R} \approx$ constant, $\overline{R} \approx \widetilde{R}_o \bar{t}_o^{1/2}$,

$$\frac{\nu t}{R^2} = \frac{\bar{t}}{\bar{t}_o}\cdot\frac{1}{\widetilde{R}_o^2}. \tag{3.8.5}$$

Next, we shift the origin in time to get t_Γ by

$$\frac{\nu t_\Gamma}{R^2} = \frac{\nu t}{R^2} + \frac{\nu t_{shift}}{R^2} \tag{3.8.6}$$

and we solve for the initial time in the calculation, t_o from

$$\frac{\nu t_{shift}}{R^2} = \frac{\nu t_\Gamma}{R^2} - \frac{\nu t}{R^2} = \left(\frac{(a/R)_o}{2.24182}\right)^{1/2} - \frac{1}{\widetilde{R}_o^2}. \tag{3.8.7}$$

Combining equations (3.8.5) and (3.8.7) gives

$$\frac{\nu t_\Gamma}{R^2} = \frac{1}{\widetilde{R}_o^2}\left(\frac{\bar{t}}{\bar{t}_o} - 1\right) + \left(\frac{(a/R)_o}{2.24182}\right)^2. \tag{3.8.8}$$

In the results, Re_Γ and Re_I are used when referring to a specific calculation. When the distinction is not necessary, we simply use Re to denote the Reynolds number.

Chapter 4

Harmonic Convergence

In this chapter, we examine convergence properties of the numerical method. As mentioned in section 2.2, it can be shown for linear equations, that the convergence of a spectral approximation to a typical velocity field (by least squares) determines the convergence of the spectral solution to the exact solution of the differential equation. Therefore, we begin by examining the convergence of a spectral expansion to some typical solutions – Stokes rings and thin rings.

Next, by using the Stokes vortex ring solution, an analytic solution to the Stokes equations, a substantial part of the code is tested. This test is easily implemented by setting the convective terms to zero. Note that the polar dependence is exact using the $\ell = 1$ mode, so that only expansions in the radial direction are tested.

Because the Navier-Stokes equations are nonlinear, there is no guarantee that the convergence of the time-dependent solution is related to that of the basis functions, however, this is what normally happens in practice. By computing the impulse (which is conserved in the exact solution) as a function of time, we are able to test the global convergence and accuracy as the time step is decreased and the resolution is increased. From this, we show exponential convergence with resolution and second order convergence with the time step, as expected. Furthermore, when the two limits are taken simultaneously, the error in impulse goes to zero, showing that the solution is globally accurate.

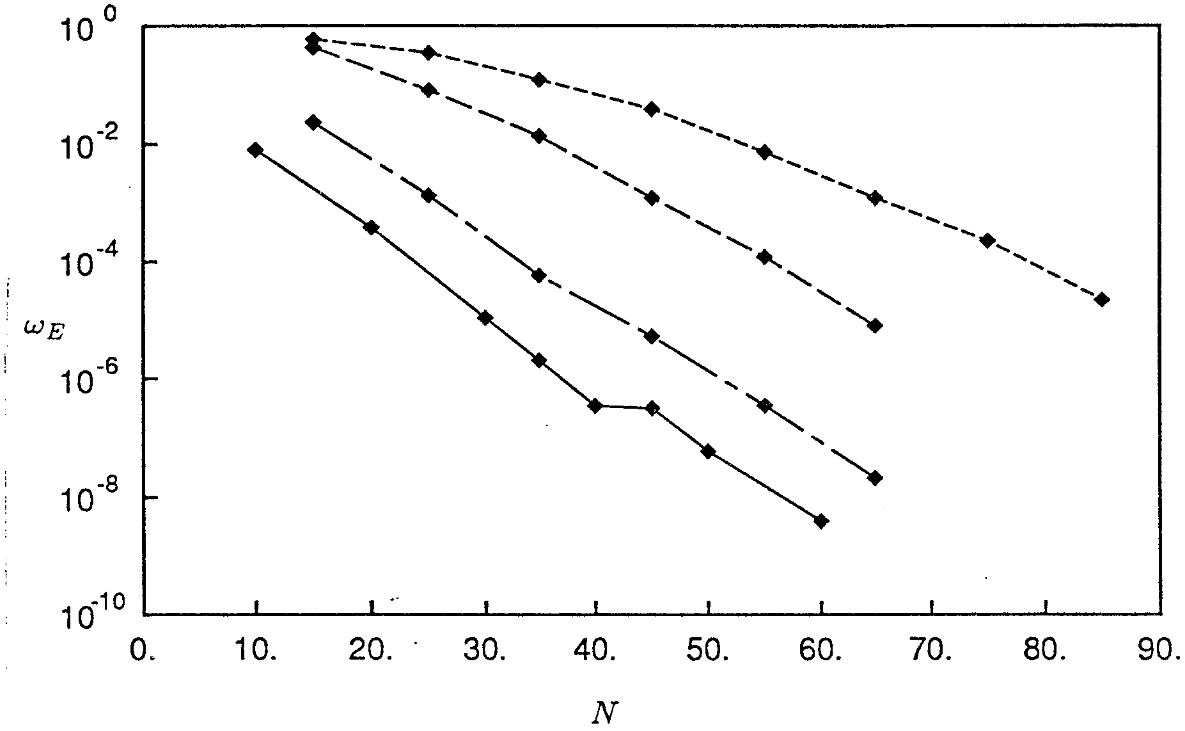


Figure 4.1. Convergence of the spectral expansion to typical initial conditions: ——, the Stokes solution; — – —, $a/R = 0.35$; — —, $a/R = 0.15$; - - -, $a/R = 0.1$.

4.1 Initial Conditions

As shown in Gottlieb and Orszag (1977), the beauty of spectral methods is their exponential convergence as the number of modes, N, is increased. If the basis functions are chosen incorrectly, however, the convergence can be much slower, or the numerical solution may even converge to the wrong solution. An important test is conducted in this section showing the convergence properties of the spectral expansion to typical solutions. Of interest in particular, is the number of terms that are required to represent a given vector field, and how the error varies with the number of modes. First, we must define a measure of the error.

Only one vorticity component (ω_ϕ) is nonzero in axisymmetric flow. If the exact vorticity is given by ω, then the local error,

$$\omega_{error} = \omega - \sum_{n\ell} a_{n\ell}\, \Psi_{n\ell\omega}(r,\theta) \tag{4.1.1}$$

is exponentially small. From this we define a global, normalized error:

$$\omega_E \equiv \left[\frac{\int_A \omega_{error}^2\, dA}{\int_A \omega^2\, dA}\right]^{1/2} . \tag{4.1.2}$$

It follows that we expect the global error to also be exponentially small. Therefore, $\ln \omega_E$ versus N should be linear.

In figure 4.1, the global error is plotted on a logarithmic scale as a function of the number of modes, N, where $L = N$. For four different initial conditions – a Stokes solution, and three thin rings with $a/R = 0.35$, 0.15, and 0.1 – the curves are indeed essentially linear, at least for large enough N. Because the Stokes vortex ring involves only the $\ell = 1$ mode and has a relatively smooth vorticity distribution, it is the easiest to represent. With only 10 modes, the global error is $O(10^{-2})$. Between 40 and 45 modes there is a jog in the curve, for which we do not have a definitive explanation. The number of collocation points used in all cases studied in figure 4.1 is $N+5$. For this stray point ($N = 45$), an additional case was run with $\frac{3}{2}N$, giving the same result to within plotting accuracy. From this, we conclude that integration error is not the cause. Perhaps it is due to round off errors since, with this number of modes, the errors are very small.

As expected, approximations of rings with thinner cores have larger errors than thicker cores for a given number of modes. Notice also, that the slope is decreasing slightly for thinner cores. With a sufficient number of modes, however, all of the cases considered indicate exponential convergence. When N and L are equal, the number of collocation points in the core in each direction (r and θ) was roughly equal. Therefore, for all of the thin ring cases, L is set equal to N, simplifying the input.

The thinnest ring which was resolved sufficiently was $a/R = 0.1$. Because the spacing between collocation points in θ is equal (Appendix C), to resolve a ring with half the core size, we need twice as many modes. At some point, therefore, the payback is small compared to the cost. In practice, to determine if a given problem is resolved sufficiently, it is best to show that the quantities of interest are insensitive to variations in the free parameters, N, N_c, and r_1, by $\pm 10\%$.

In solving the Navier-Stokes equations, we can start with any vorticity distribution which satisfies the boundary conditions. In general, the gradients in the solution will decrease overall (since the Re decreases in time), however, this is not necessarily true locally (because Re is a global parameter). Although we expect the solution at later times to require fewer terms than the initial condition, time-dependent diagnostics are desirable. Furthermore, such diagnostics are needed to monitor the accuracy of the time integration.

4.2 Time-Dependent Solutions

Next, the convergence and accuracy of the time-dependent solutions is examined. The Stokes part of the Navier-Stokes code is verified to be working correctly by comparing vorticity profiles at several times with the analytic time-dependent solutions. To study the accuracy and convergence of the numerical solutions to the Navier-Stokes equations, since there are no useful exact solutions, we rely on

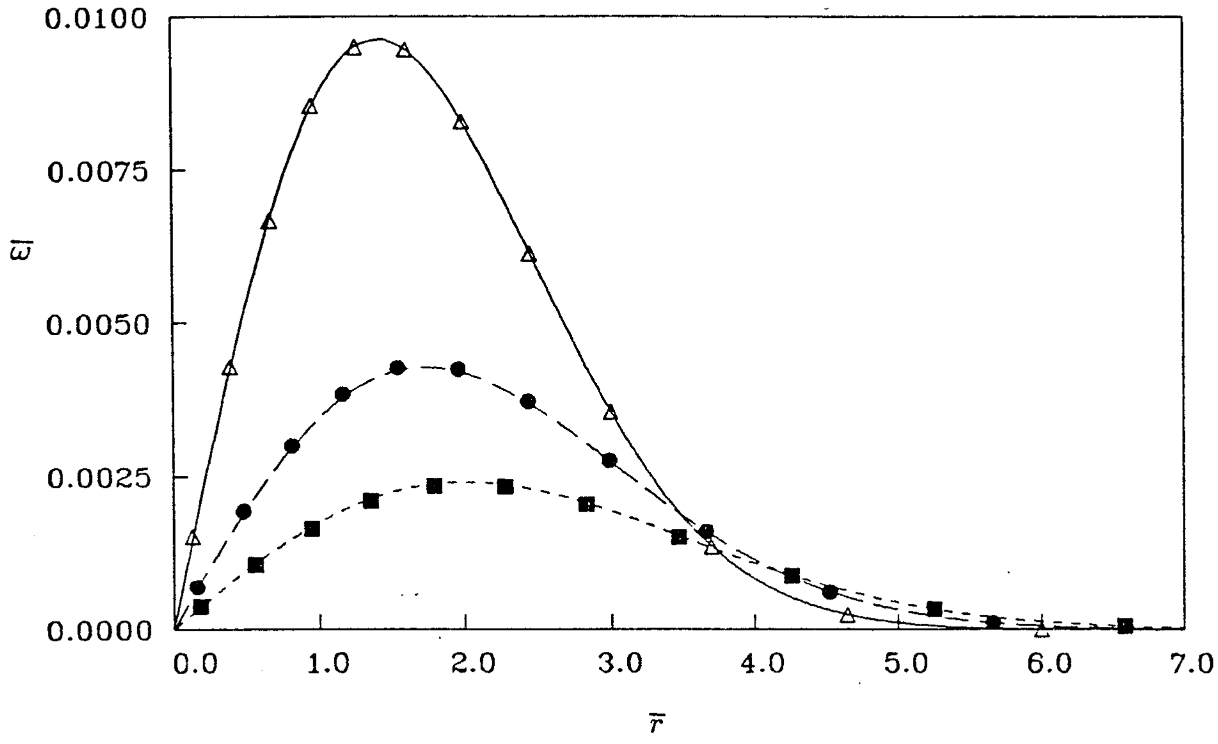


Figure 4.2. Vorticity along $\theta = \pi/2$ computed from the Stokes equation (shown as symbols at the collocation points) compared to the analytic Stokes solution: —, $t = 1.0$; — —, $t = 1.5$; – – –, $t = 2.0$).

diagnostics such as impulse, circulation, and the comparison of the rate-of-change of energy with dissipation. The impulse is especially useful since it is fundamentally conserved for unforced flow in an unbounded domain. The impulse of time-dependent solutions is computed for various step sizes and resolutions and shows that the solutions converge (with decreasing step size and increasing N) and that they are accurate.

4.2.1 Stokes Equations

The convergence and accuracy of the time-dependent solution of the Stokes equations are tested using the exact solution of the Stokes equations, the Stokes vortex ring (eqn. 3.7.1). This test is easily implemented by setting the convection term equal to zero in the Navier-Stokes code. The polar direction is exact with only the $\ell = 1$ mode (the θ dependence for both the Stokes vorticity and the first polar vorticity function, P_1^1, is $\sin\theta$), therefore, it is sufficient to consider only the radial direction, for which there are 10 modes and 15 collocation points.

The initial condition is given by the solid line in figure 4.2 ($\bar{t} = 1$) and the approximate representation of the function at the collocation points is shown by triangles. Integrating the Stokes equations in time (40 time steps) gives the result shown by circles ($\bar{t} = 1.5$). The exact solution at that time, shown by the long-dashed line, is in close agreement. Integrating once again (30 time steps) we get the square symbols. Again, the results are in very close agreement with the exact solution at that time ($\bar{t} = 2.0$) represented by the shortdashed line.

Notice that the solution is very accurate with only 10 terms – the initial condition, at $\bar{t} = 1$, corresponds to ω_E of $O(10^{-2})$ in figure 4.1 – and the accuracy is maintained at the later times. Also note that the peak vorticity moves outward with time and the collocation points (because of the transformation – eqn. 2.1.8) follow suit.

4.2.2 Navier-Stokes Equations

Since exact solutions of the full nonlinear equations are not available, other methods are needed to verify that the code converges and that it is accurate. One indication is the degree to which global invariants, such as impulse, are conserved. Also computed are the rate of change of vorticity, circulation, the momentum, and the dissipation and rate of change of energy. Another means of verifying the accuracy is to compute limiting solutions and compare them with asymptotic theories (see Chapter 5).

A single case ($(\Gamma/\nu)_o = 100$ and $(a/R)_o = 0.35$) is run: first, with a fixed resolution and varying the time step, and second, with a fixed time step and varying resolution. The baseline case, from which these variations are made, uses 55 × 55 modes, 400 time steps, and $(\Delta\tilde{t})_b = 0.0025$. At each time step, we compute the percent error in impulse, defined by

$$I_E \equiv \frac{I - I_{exact}}{I_{exact}} \times 100. \qquad (4.2.1)$$

Recall that $\bar{t}$ refers to the dimensionless time and $\tilde{t}$ refers to the time in the calculation, where $\tilde{t} \equiv \ln \bar{t}$.

In figure 4.3, the time history of I_E is plotted for four different step sizes: $\frac{1}{2}(\Delta\tilde{t})_b$, $(\Delta\tilde{t})_b$, $2(\Delta\tilde{t})_b$, $4(\Delta\tilde{t})_b$, using 800, 400, 200, and 100 time steps, respectively, where the baseline case is shown by the chaindashed curve. As expected, the error becomes smaller with smaller time steps. Furthermore, the magnitude of the error in this figure is very small – ≈ 0.001% (accurate to 5 significant digits). We

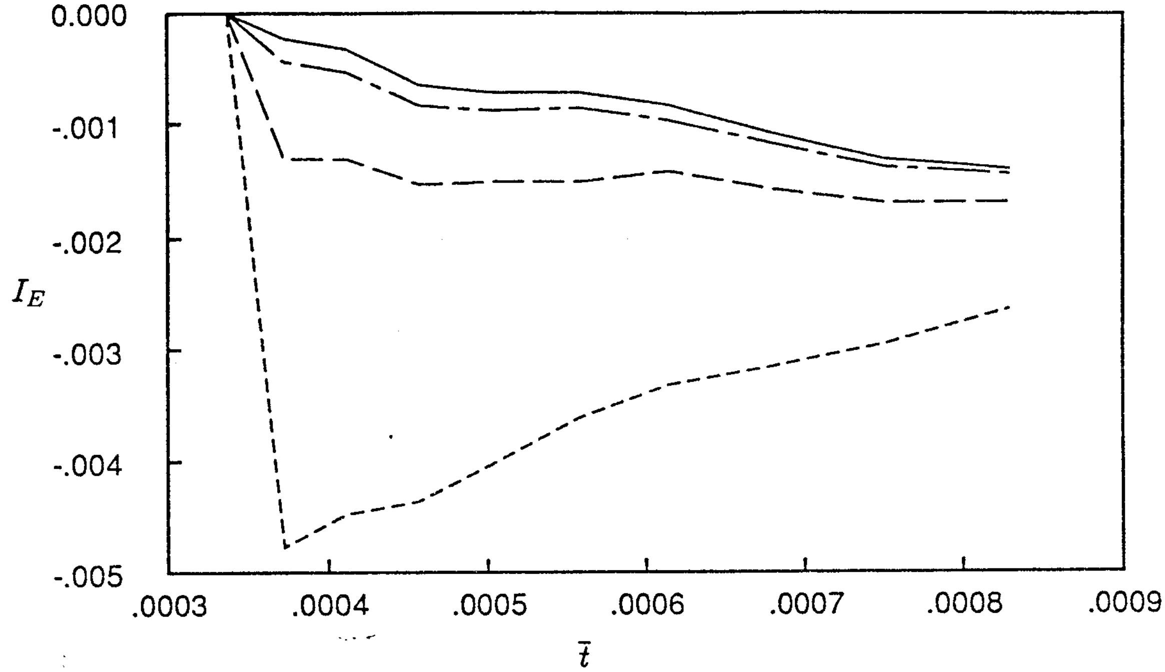


Figure 4.3. Navier-Stokes calculations with initial conditions $(a/R)_o = 0.35$, $(\Gamma/\nu)_o = 100$, and $N = L = 55$, and different step sizes: ——, $\frac{1}{2}(\Delta\widetilde{t})_b$; — – —, $(\Delta\widetilde{t})_b$; — —, $2(\Delta\widetilde{t})_b$; – – –, $4(\Delta\widetilde{t})_b$.

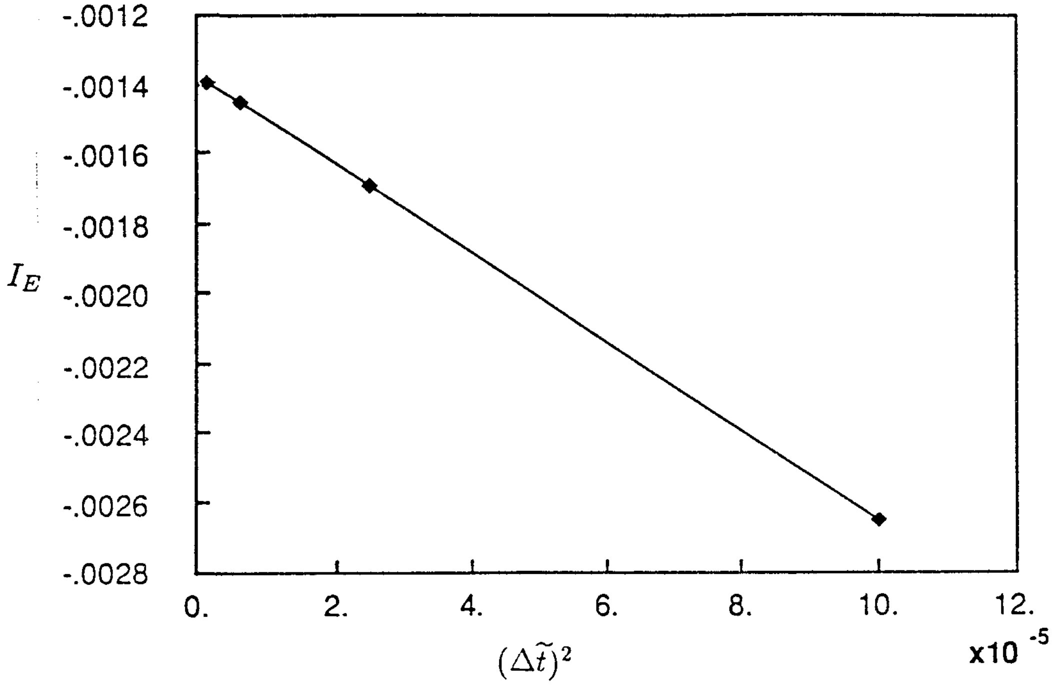


Figure 4.4. The percent error in impulse versus $(\Delta\widetilde{t})^2$ at the final time in figure 4.3.

found that the largest step size, $4(\Delta\tilde{t})_b$, is very near to the stability limit (with larger values of $\Delta\tilde{t}$, the solution blew up) indicating that stability rather than accuracy, constrains the maximum step size. At the final time, $\bar{t} = 0.00083$, I_E is plotted as a function of $(\Delta\tilde{t})^2$ (figure 4.4). As expected, the curve is linear, since the time differencing scheme is second order accurate (see Chapter 3). This test is useful for detecting subtle coding errors.

Note in figure 4.3 that the curves are converging to a finite error in impulse. We immediately suspect that the spatial resolution is responsible for the remaining error. To test this, the resolution is varied (45, 55, and 65 modes) from the baseline. The result, figure 4.5, shows that this is indeed the case – the dashed line is the more coarse grid and the solid line is the fine grid, showing that the error is now going to zero. Furthermore, plotting the logarithm of the error at the final time as a function of the number of modes (fig. 4.6), gives a relatively straight line. Therefore, we have shown that the time-dependent solution converges exponentially with the number of modes, N. This is interesting since a rigorous proof for this behavior does not exist for the Navier-Stokes equations.

At each time step in the calculation, several other diagnostics are computed in addition to the impulse. The circulation, given by

$$\Gamma = \int_{-\infty}^{\infty} \int_{0}^{\infty} \omega \, dy \, dx \tag{4.2.2}$$

is compared to the input value for thin rings, and monitored over time. As the time becomes large, the circulation approaches zero due to diffusion of vorticity across the axis of symmetry, which cancels with that of the opposite sign. However, at early times the loss of circulation is exponentially small.

As shown by Cantwell (1986) the initial forcing on the flow in the form of an impulse in space and time, transfers 2/3 of the impulse to the velocity field and 1/3 to the pressure field. It is a simple matter to compute the integral of the velocity over the domain,

$$H = \int_V u \, dV = 2\pi \int_A u \, y \, dA \tag{4.2.3}$$

where V is the volume of an infinite domain, and A is the area of the meridional half plane. Evaluating equation (4.2.3) does indeed give 2/3 of the impulse. This is primarily a check of the behavior of the velocity field at large distances.

The kinetic energy was computed and its rate of decay compared very well with the dissipation, which was computed separately from the velocity derivatives (see figures 5.1k and 5.11k). This showed that the viscous term was resolved and that the numerical dissipation (of either sign) introduced by time-integration errors was much smaller than the true viscous dissipation. This again indicates that the stability criterion is more stringent than the accuracy concern. One advantage of this behavior, is that any solution which is stable is also very accurate.

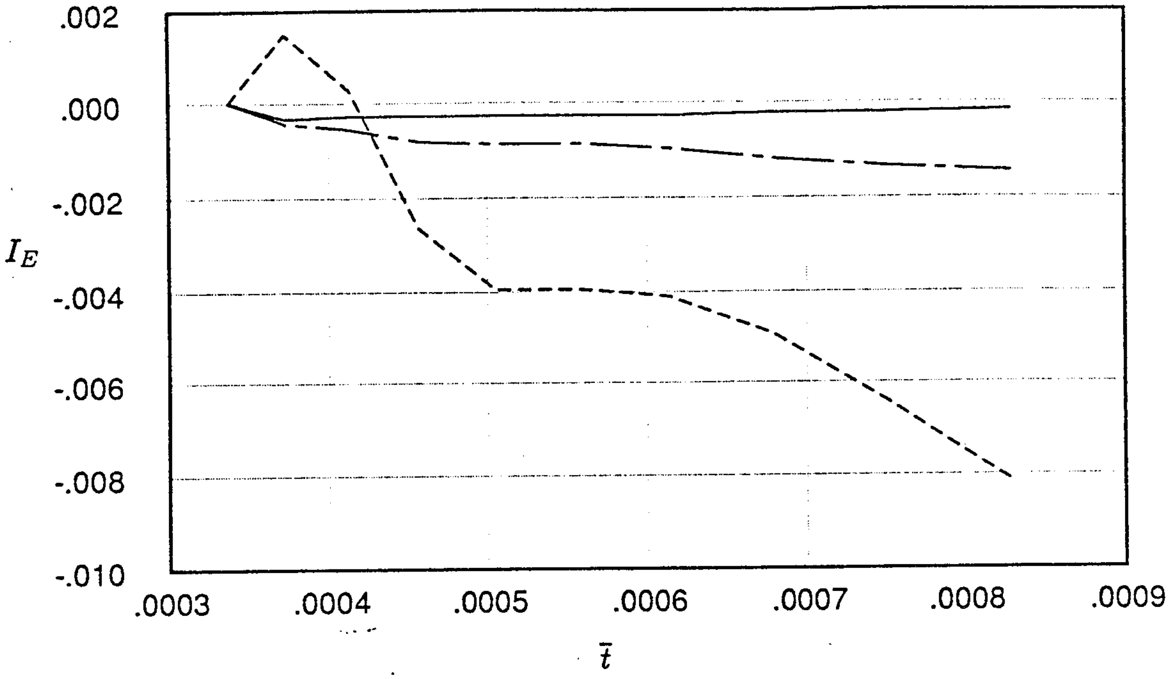


Figure 4.5. Navier-Stokes calculations with initial conditions $(a/R)_o =$ 0.35, $(\Gamma/\nu)_o = 100$, and $\Delta\tilde{t} = 0.0025$, and variable number of modes where $L = N$: ——, $N = 65$, — — —, $N = 55$; - - -, $N = 45$.

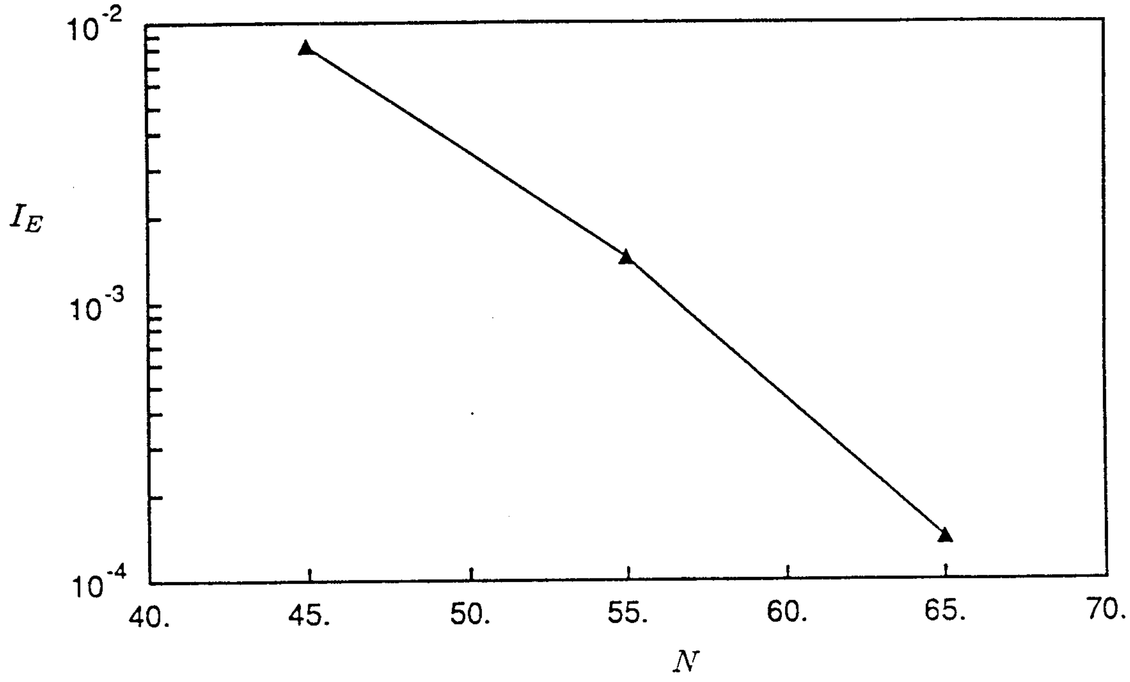


Figure 4.6. The logarithm of percent error in impulse versus N at the final time ($\bar{t} = 0.00083$) in figure 4.5.

Chapter 5

Results

In the previous chapter we showed that for the numerically computed flow fields, impulse is conserved to high accuracy, and that it converges exponentially in space and second order in time. Furthermore, by comparing computations of the Stokes equations with the exact solution, we gained confidence that the method is formulated and implemented correctly. In this chapter, we will build upon that confidence by showing that the calculations agree with asymptotic theories for large time and small time motion: the Stokes limit and the limit of infinitesimal cores. Because the full nonlinear equations are solved, we observe the way in which the solutions approach the asymptotic solutions.

To orient the reader, we will begin by showing a typical calculation of a single vortex ring at a Reynolds number, Re_Γ, of 500. Vorticity contours and streamlines are plotted, showing details of core shape, shedding, and transport of vorticity with the core. Next, the asymptotic cases are discussed. The final section describes the computation of interacting vortex rings with Reynolds numbers of 1000 each. The vorticity contours illustrate the severe straining of the inner vortex through the first pass and merging of the two vortices during the second passage.

5.1 Evolution of a Single Vortex Ring

In experiments, vortex rings are generated by impulsively forcing a column of fluid through a nozzle. The vortex ring forms from the rolling-up of the vortex sheets created by the boundary layer along the walls of the nozzle (Didden, 1979). After the initial transient behavior associated with this formation process is over, the dominant parameters of the flow are Γ/ν, and a/R. Although we are not equipped to calculate such a complex starting process, we can start with any smooth vorticity distribution (which satisfies the boundary conditions) and integrate the Navier-Stokes equations forward in time. As a first guess, we use a Gaussian distribution through the core, since locally (for a sufficiently thin core) it is the leading term in the Navier-Stokes equations (see section 3.5.2).

Figures 5.1a. through 5.1g show vorticity contours for a typical solution with the initial conditions, $a/R = 0.35$ and $Re_\Gamma = 500$. The contour levels are the same for each figure. The solid lines denote high levels and the dashed lines are low levels (the difference between the solid lines is a factor of 10 larger than between the dashed ones; the outermost dashed contour is $1/100^{th}$ of the innermost solid contour in figure 5.1a). Two families of contours were needed to display the behavior in the core and in the tail on the same plot. In figures 5.1h and 5.1i, time-histories of the global quantities, ring speed and circulation, are shown. The solid dots in figure 5.1h correspond to figures 5.1a through 5.1g. Figures 5.1j and 5.1k are the time-dependent histories of the diagnostics, impulse, and diffusion shown together with the rate-of-change of energy.

The vorticity is positive (counterclockwise), therefore, the ring convects to the right under its own induced velocity. Note that the peak vorticity decreases rapidly from the initial condition in the first few frames. The vorticity is diffusing outward from the core, as seen from the lowest contour levels. When the vorticity reaches the axis of symmetry, it cancels with vorticity of opposite sign across the axis, and circulation decreases. This is reflected in figure 5.1i, where the initially constant circulation begins to drop off rapidly at the time corresponding to figure 5.1c.

The core starts out circular (fig 5.1a) and after traveling a short distance, adjusts to a more elliptical shape (fig 5.1b) through its own self-induced strain. Evidence of this transient behavior is seen in the ring speed history of figure 5.1h, where the ring initially speeds up before it begins to decelerate. If the initial core size were made sufficiently small, this transient effect would no longer appear because the Gaussian core would be very close to the natural solution. Because of this initial unsteady behavior, the wake of vorticity is swept upward. For later times, the vorticity contours show a quasi-steady distribution (nearly steady in a translating frame) with a shape that is reminiscent of Norbury's family of steady, inviscid vortex rings.

Figures 5.1a′ through 5.1g′ show instantaneous streamlines (in a translating reference frame) on top of the vorticity contours shown previously. Note the dividing streamline separating the fluid which is carried with the core of the vortex and the fluid which is left behind at a given instant. Because the flow is unsteady, the streamlines differ from the pathlines and streaklines, and care must be used in discussing the time-dependence given an instantaneous picture. The observations seem to be in agreement with Maxworthy's heuristic model (1972) where vorticity diffuses across the dividing streamline and is carried downstream to form a wake. The wake however, does not extend very far behind the ring before it is annihilated by viscous diffusion. Including a passive scalar in the calculation would be useful to clearify this process.

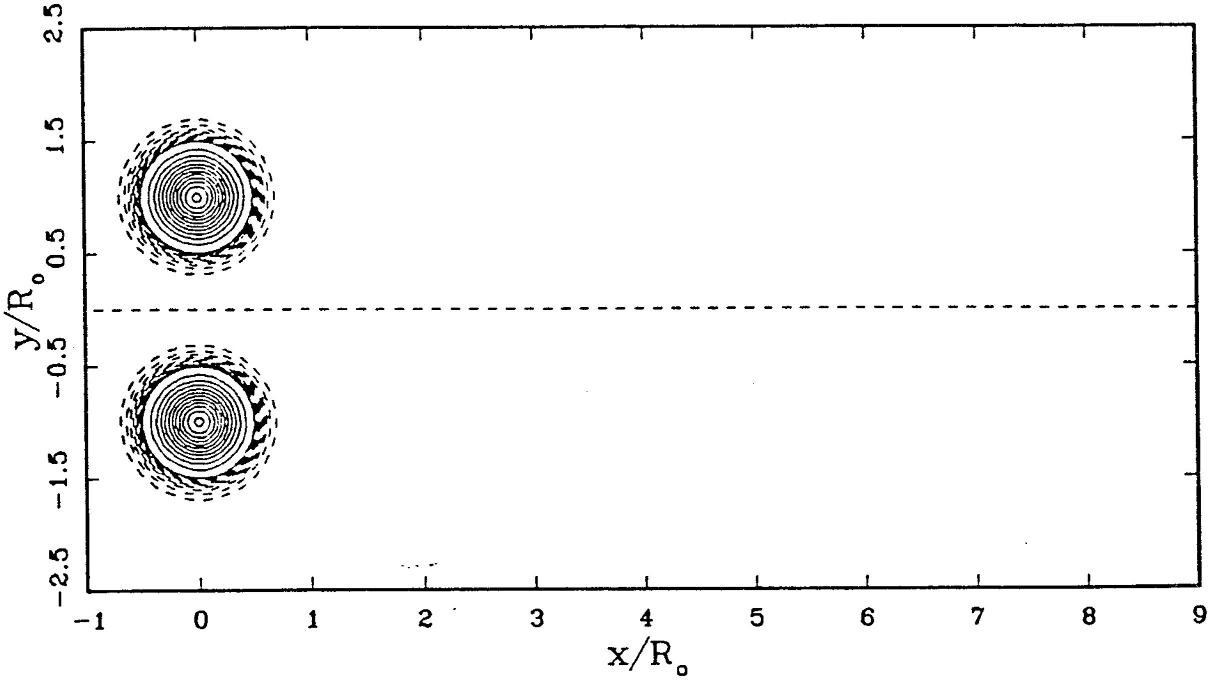


Figure 5.1a.

Figure 5.1. The evolution of a single vortex ring starting from initial conditions of $a/R = 0.35$, $\Gamma/\nu = 500$. (a)-(g): Vorticity contours at several instants in time. The change in vorticity between solid contours is a factor of 10 larger than between dashed contours. For lines of the same type, the vorticity varies linearly. (a′)-(g′): Instantaneous streamlines in a translating reference frame plotted on top of the vorticity contours of figures 5.1a-g. (h) Ring speed versus time. (i) Circulation versus time. (j) Impulse versus time (k) Rate of change of energy and dissipation versus time.

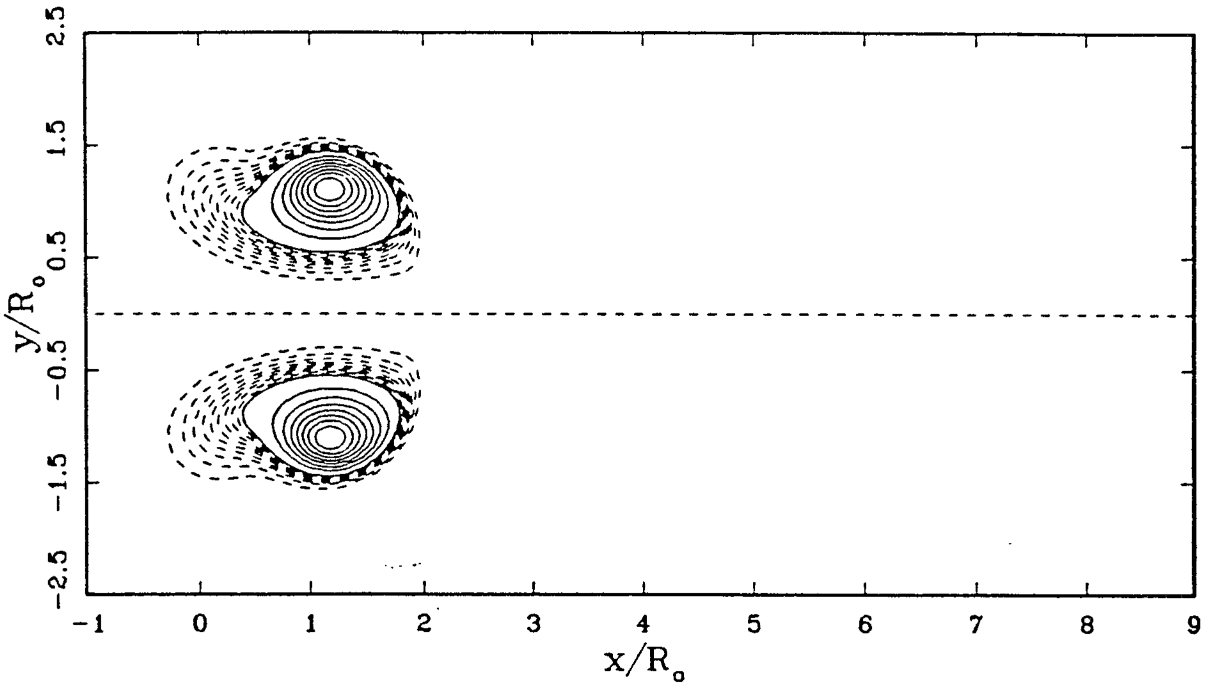


Figure 5.1b.

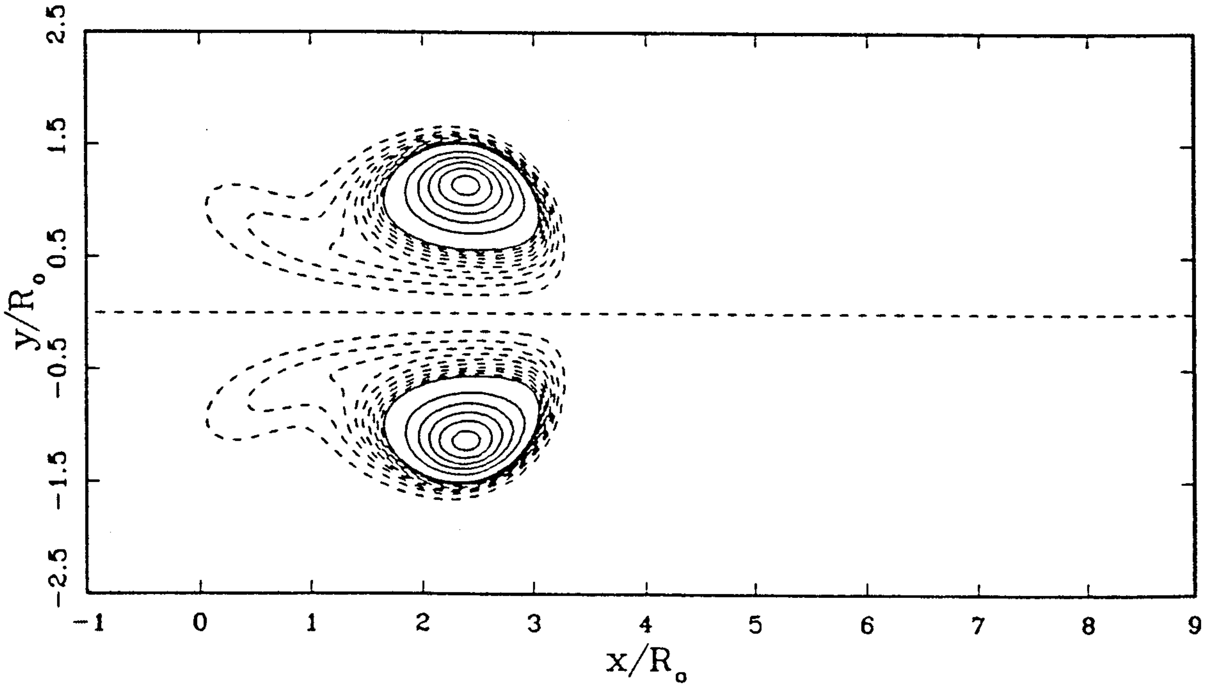


Figure 5.1c.

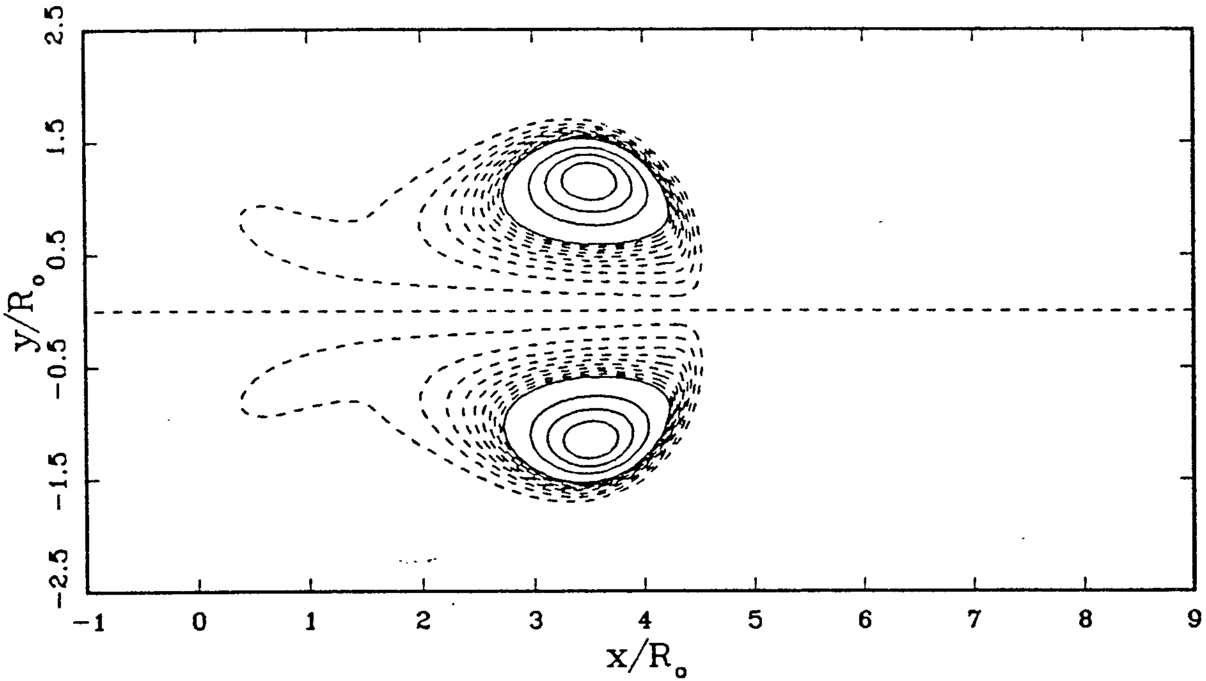


Figure 5.1d.

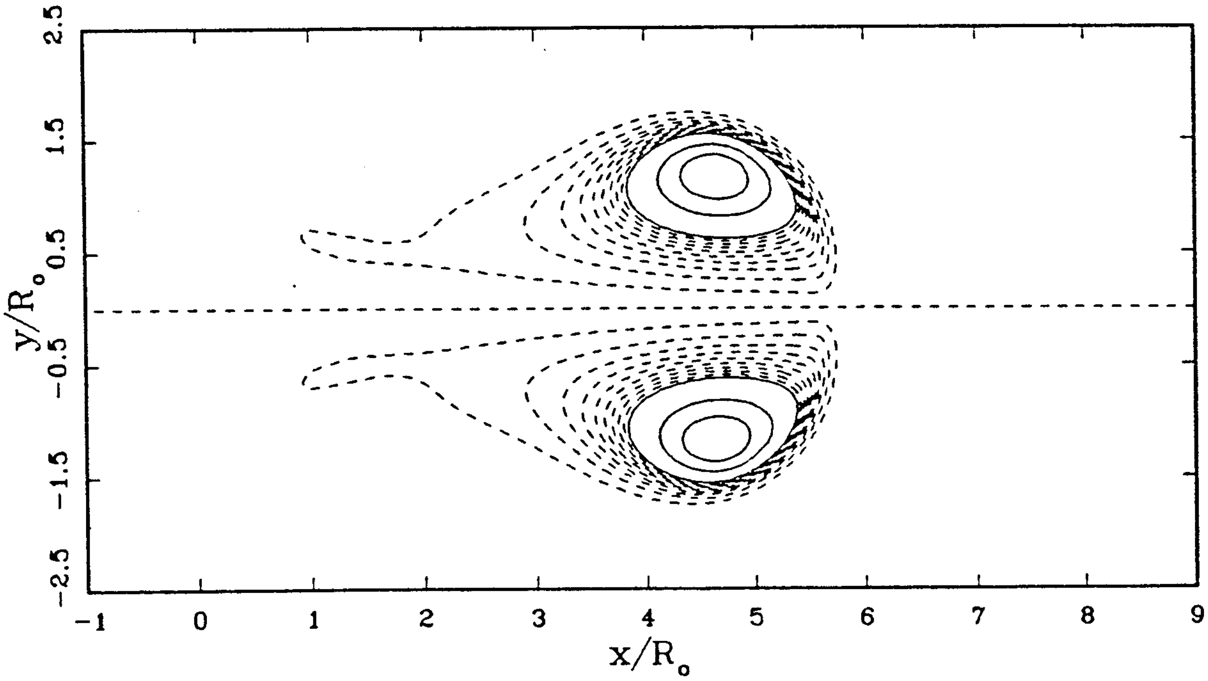


Figure 5.1e.

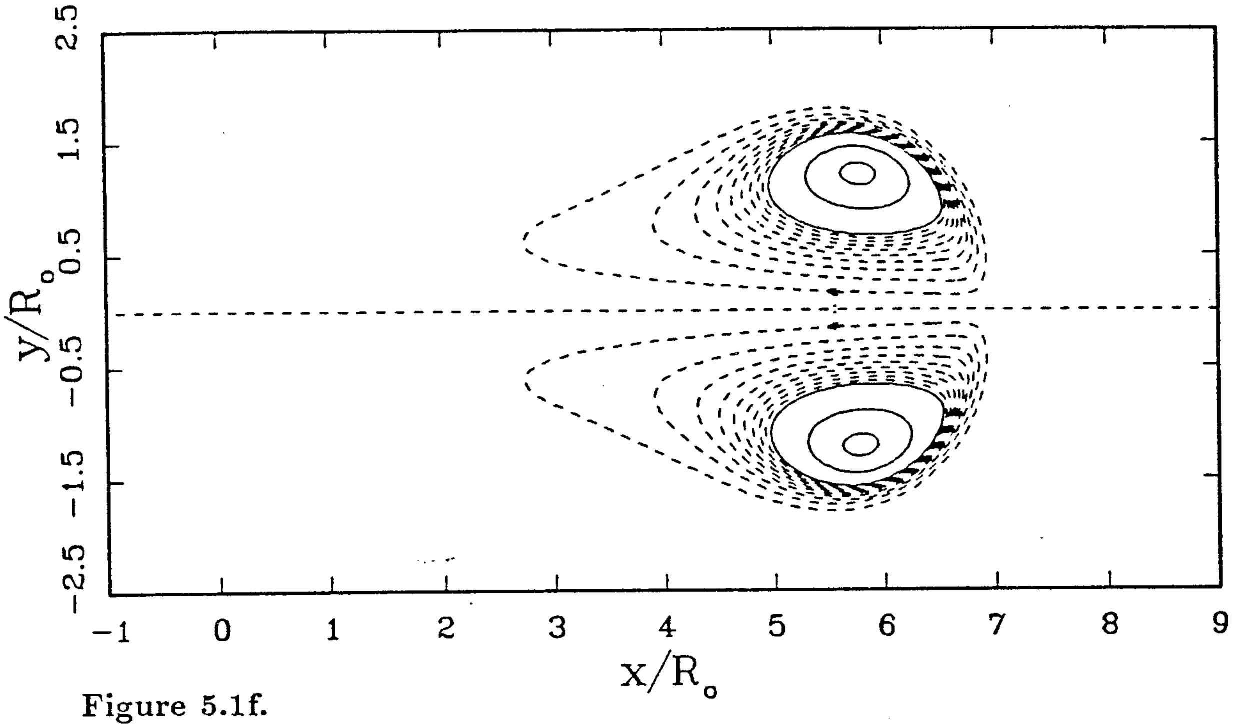


Figure 5.1f.

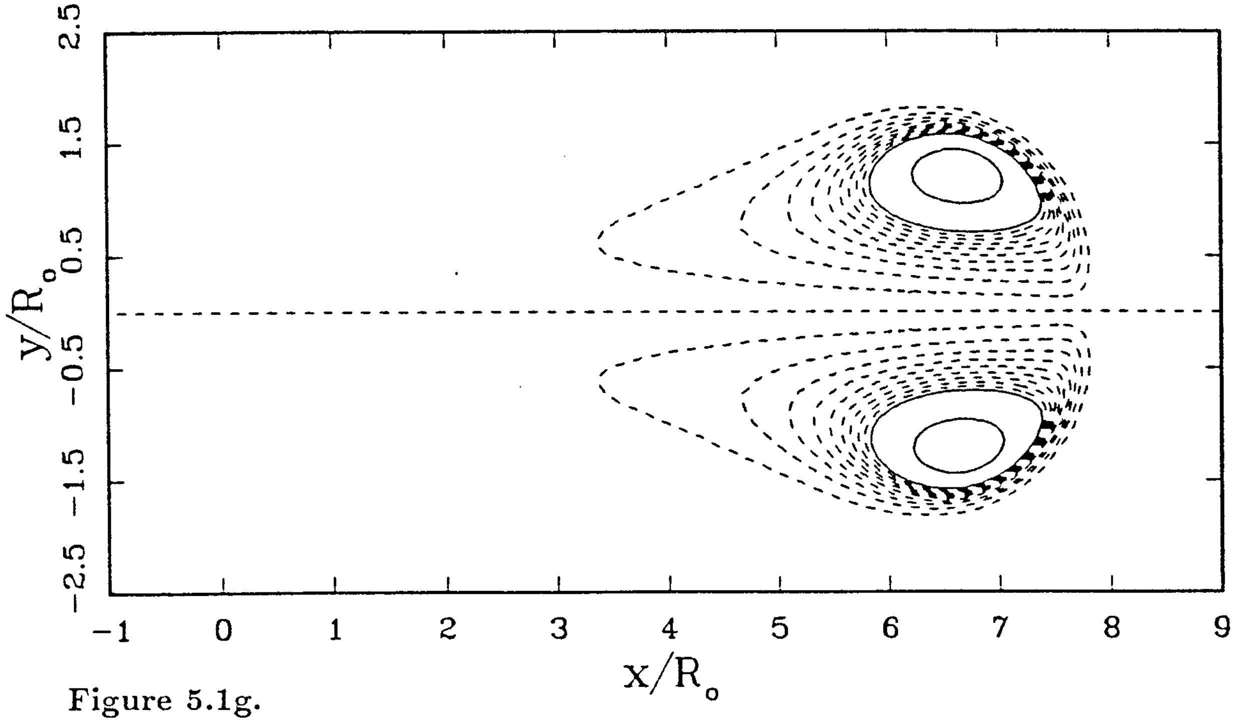


Figure 5.1g.

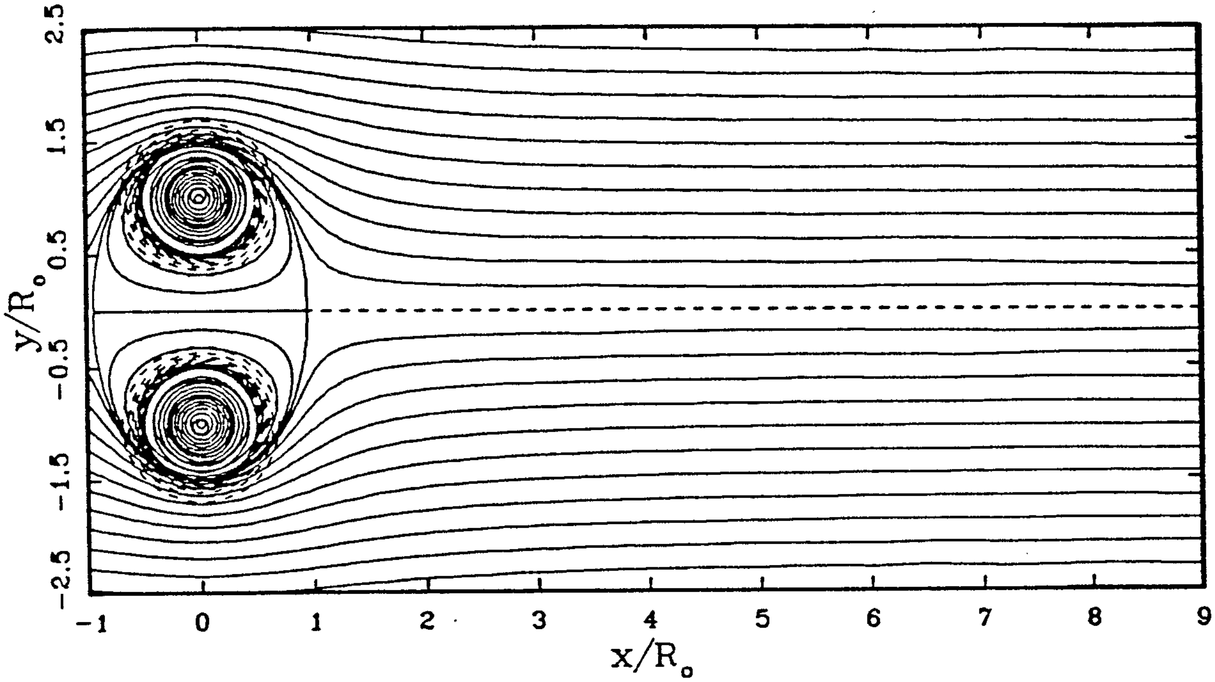


Figure 5.a′.

Figure 5.1a′-g′. Instantaneous streamlines in a translating reference frame plotted on top of the vorticity contours of figures 5.1a-g.

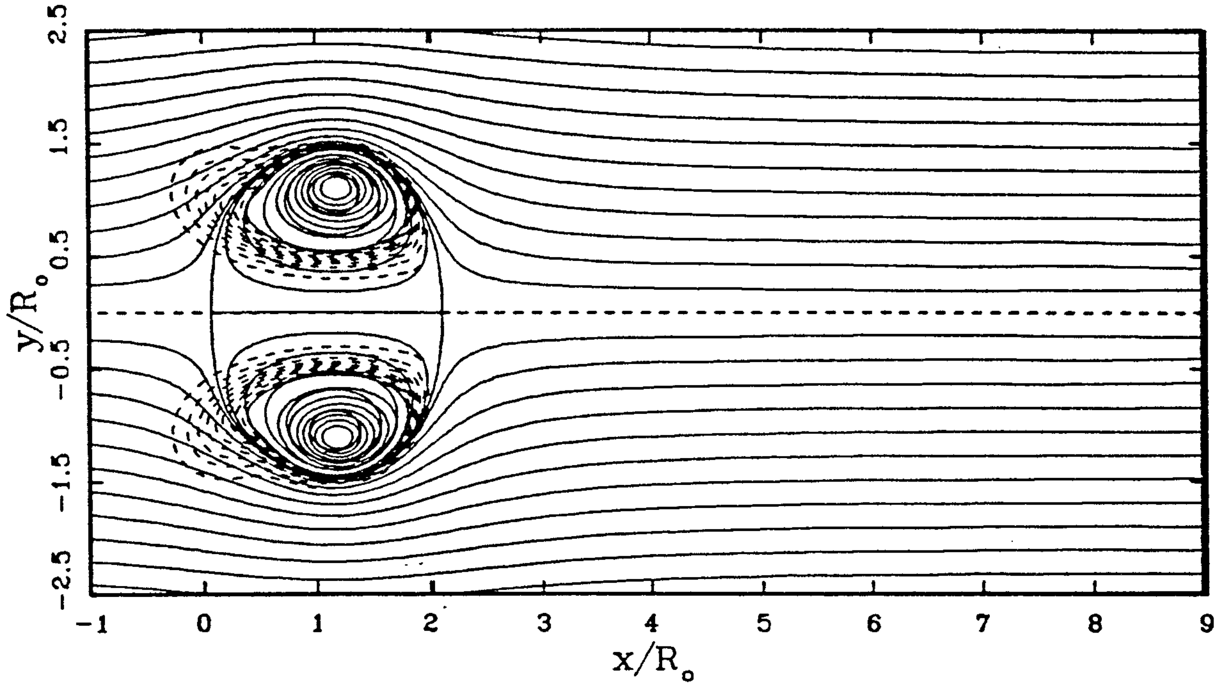


Figure 5.1b′.

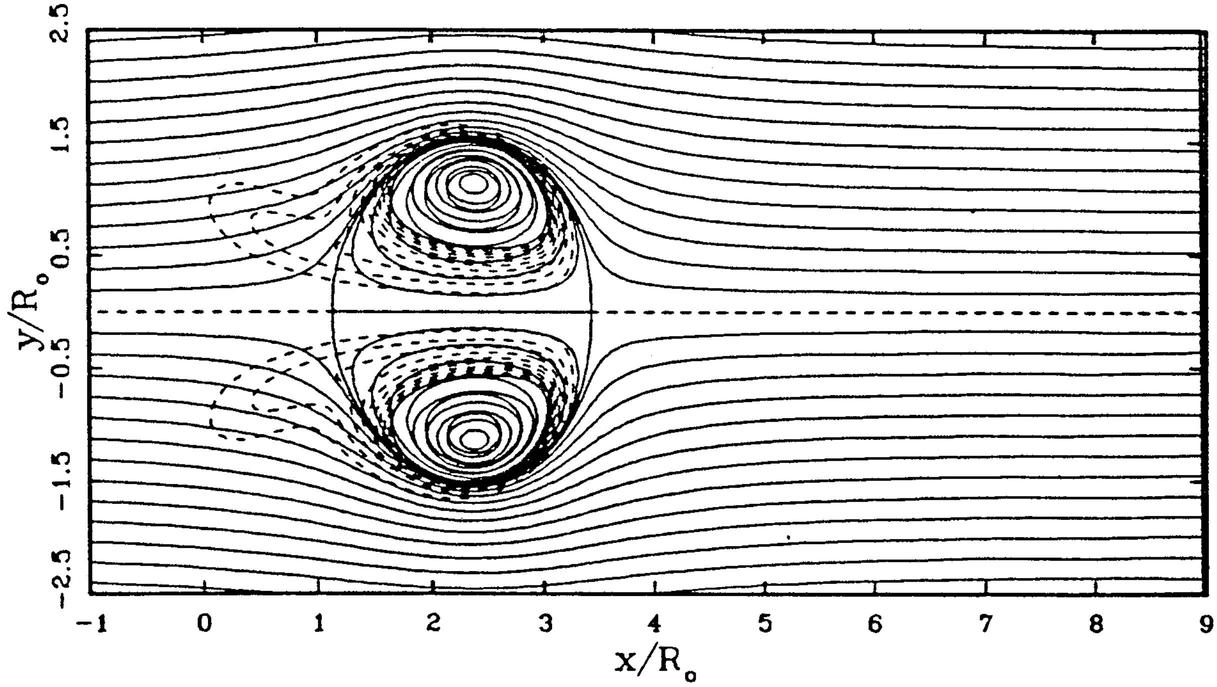


Figure 5.1c′.

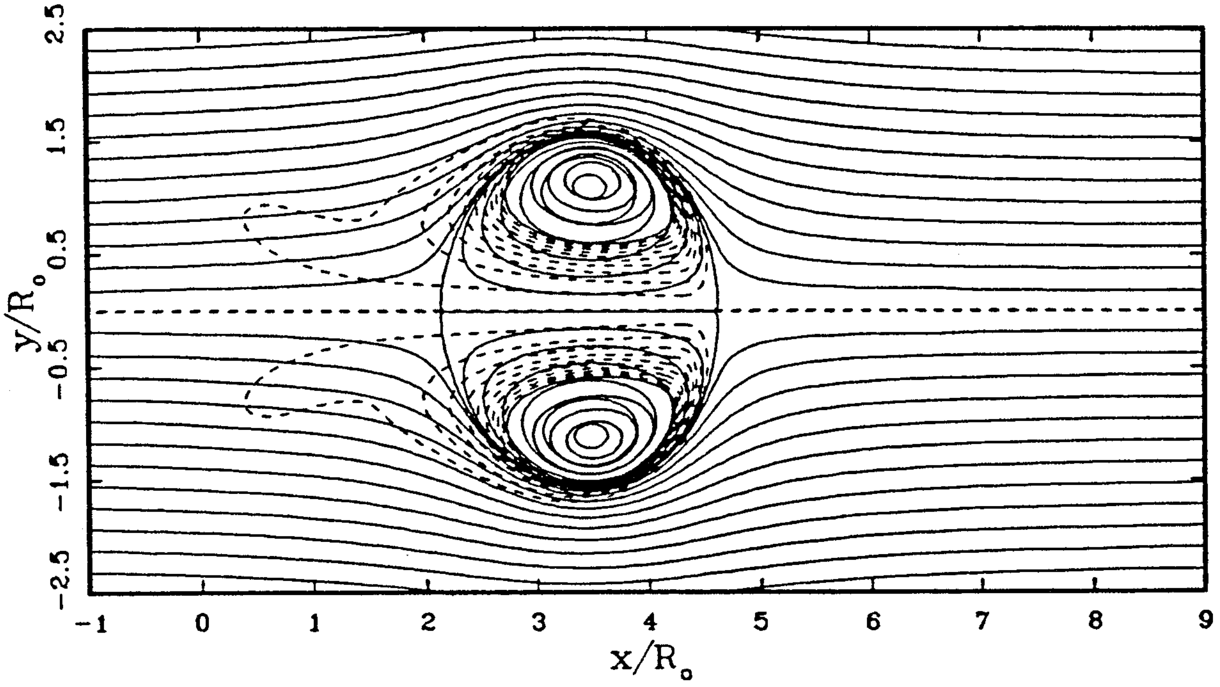


Figure 5.1d′.

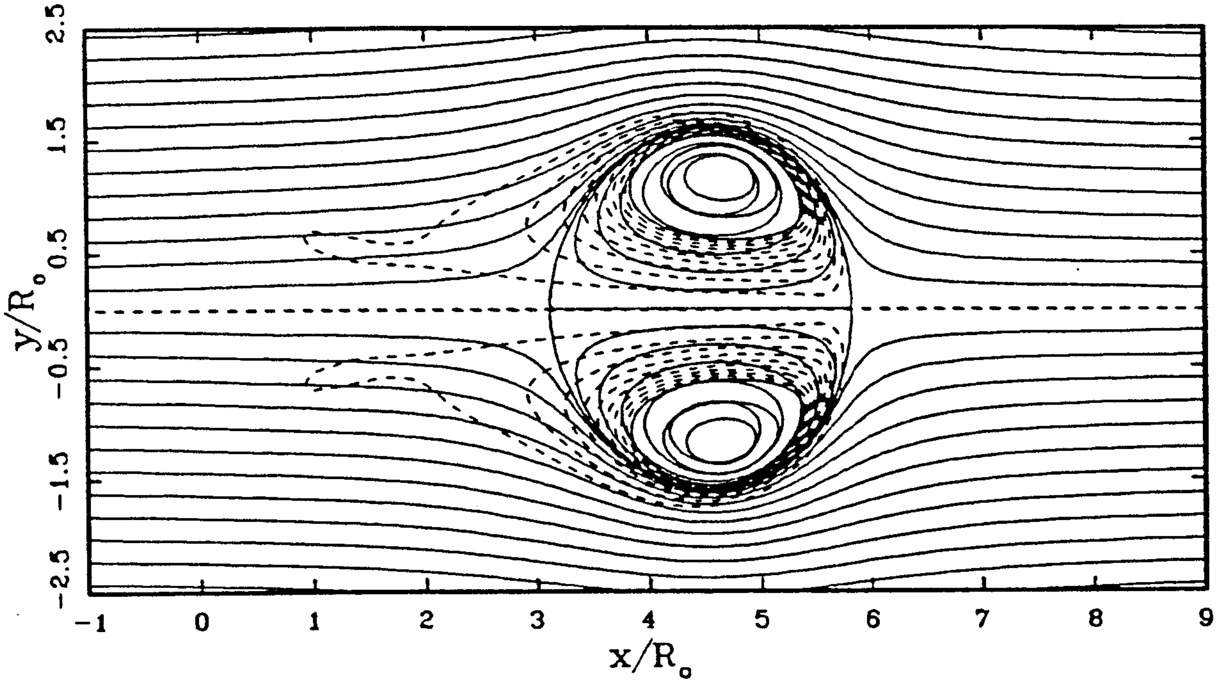


Figure 5.1e′.

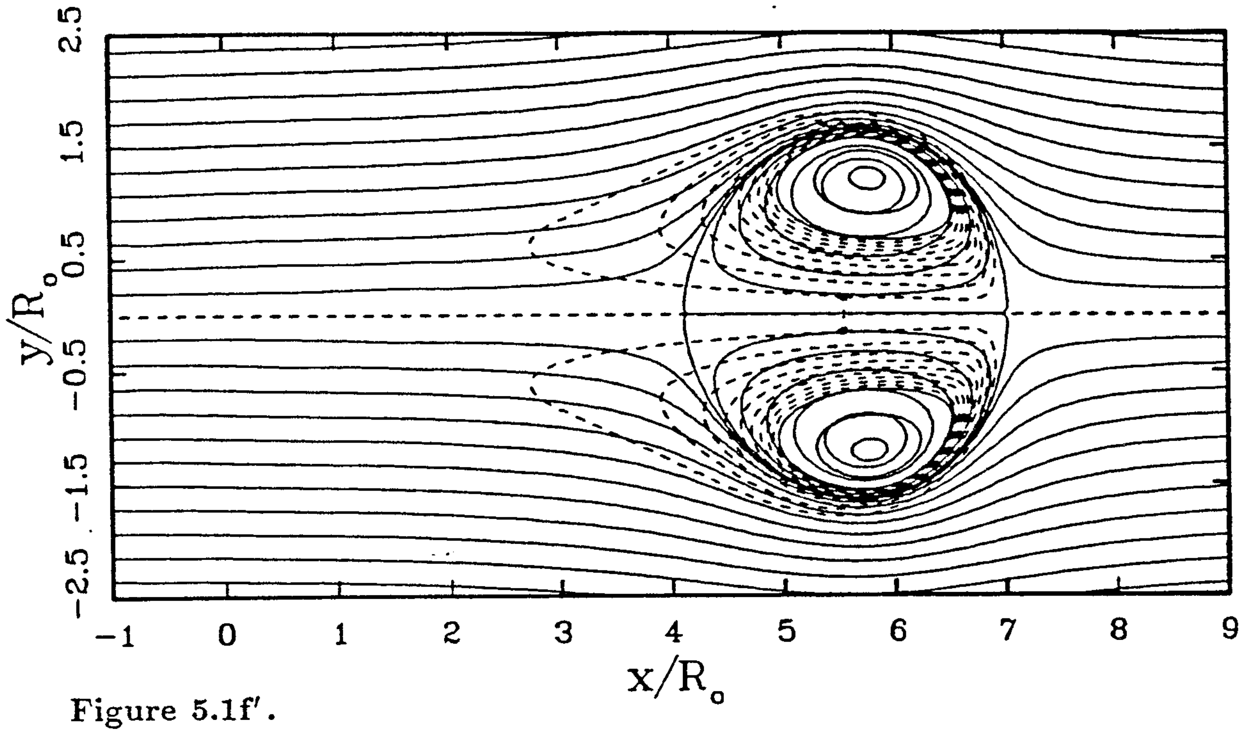


Figure 5.1f′.

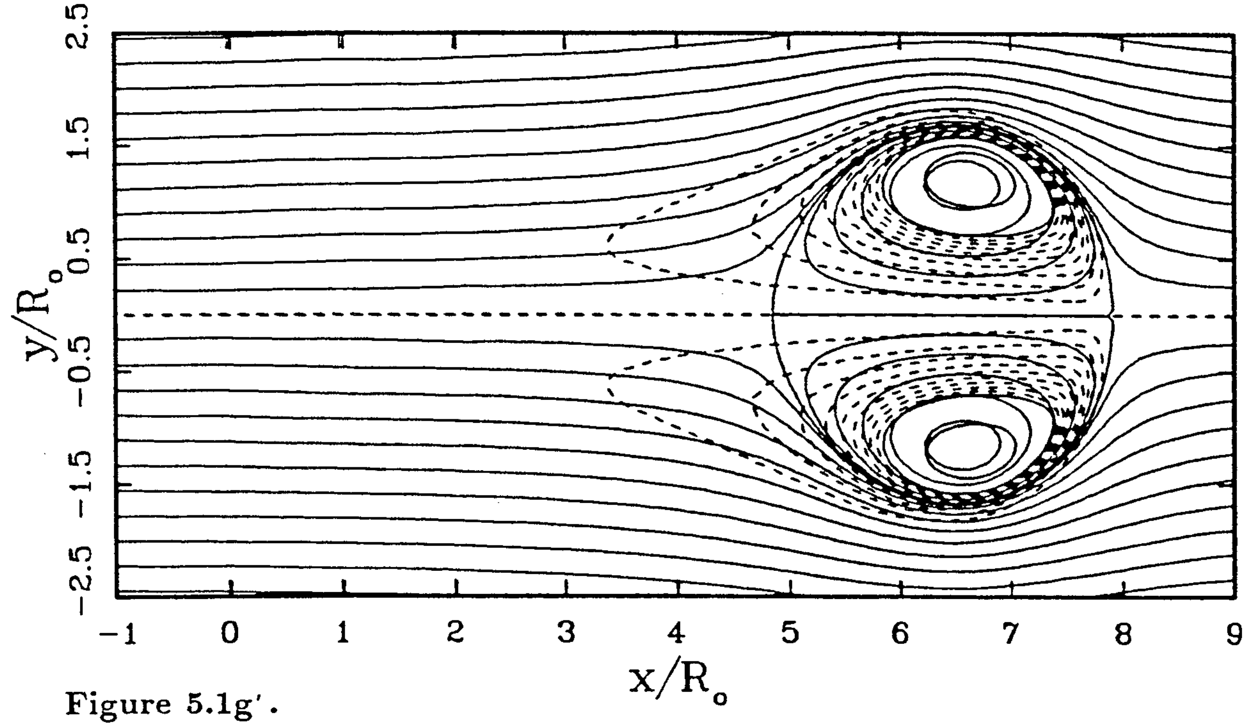


Figure 5.1g′.

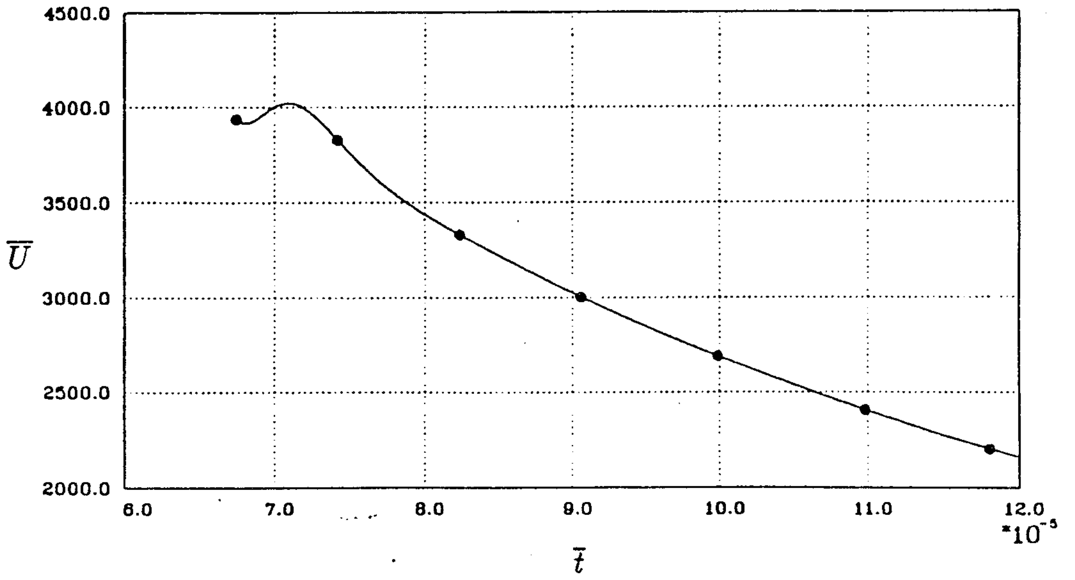


Figure 5.1h. Speed of the ring versus time. Dots correspond to vorticity plots (a)-(g).

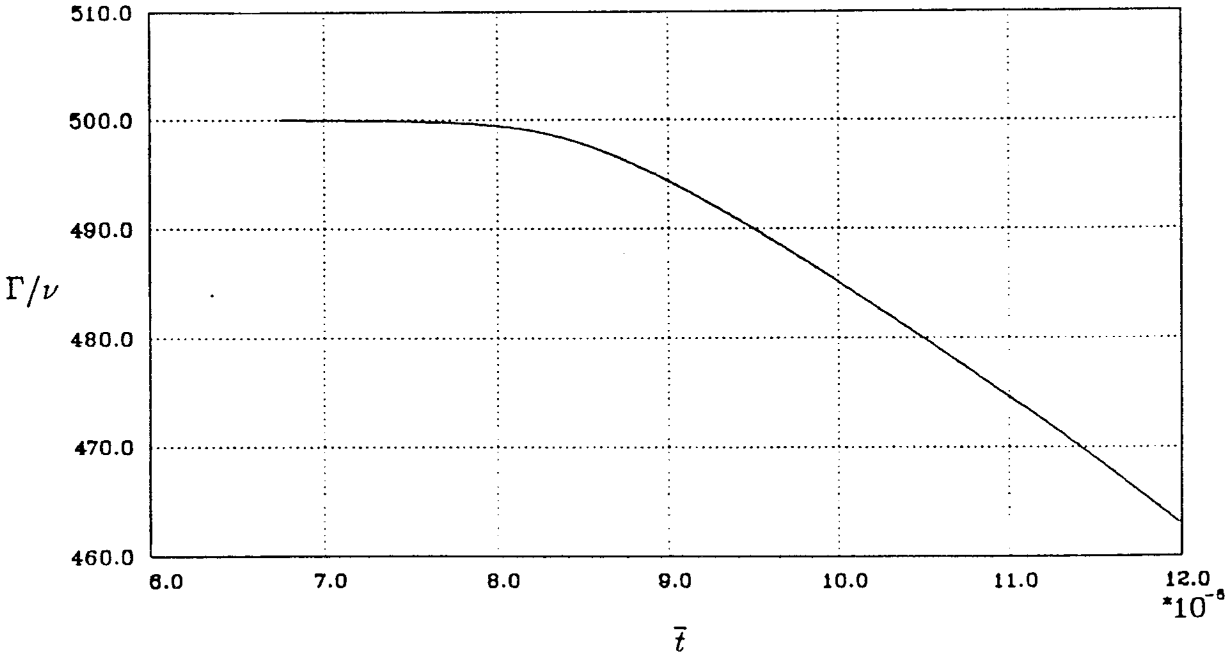


Figure 5.1i. Circulation versus time.

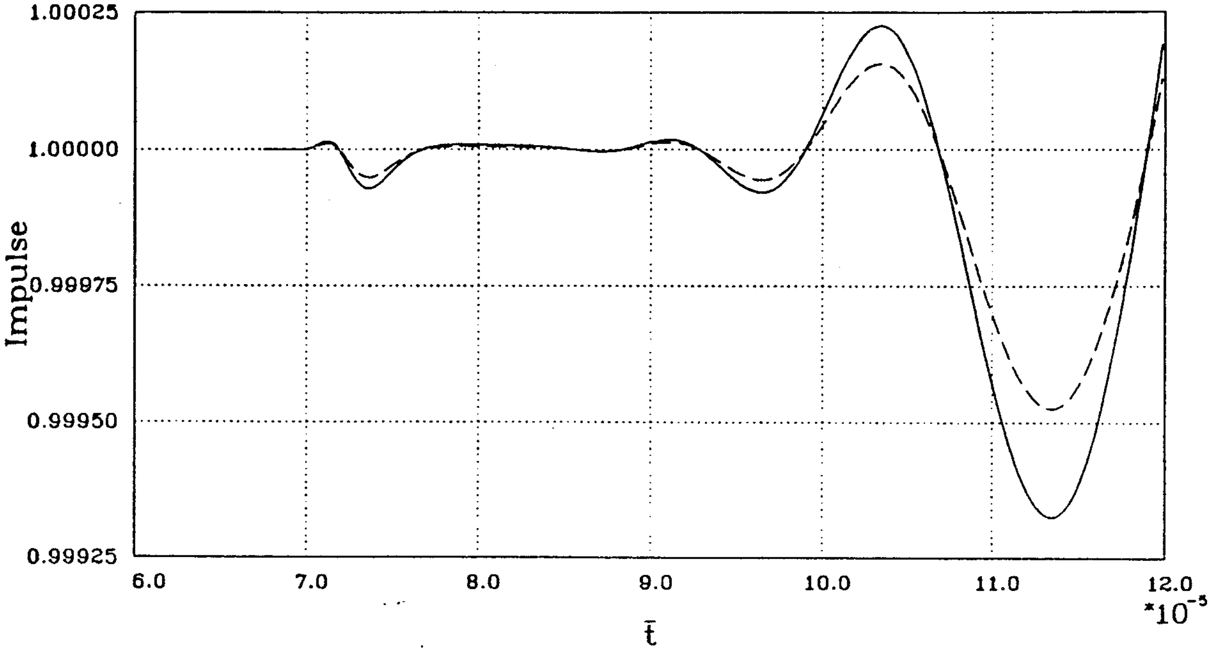


Figure 5.1j. Impulse versus time, computed by a multipole expansion (——) and computed using Gauss Quadrature (- - -).

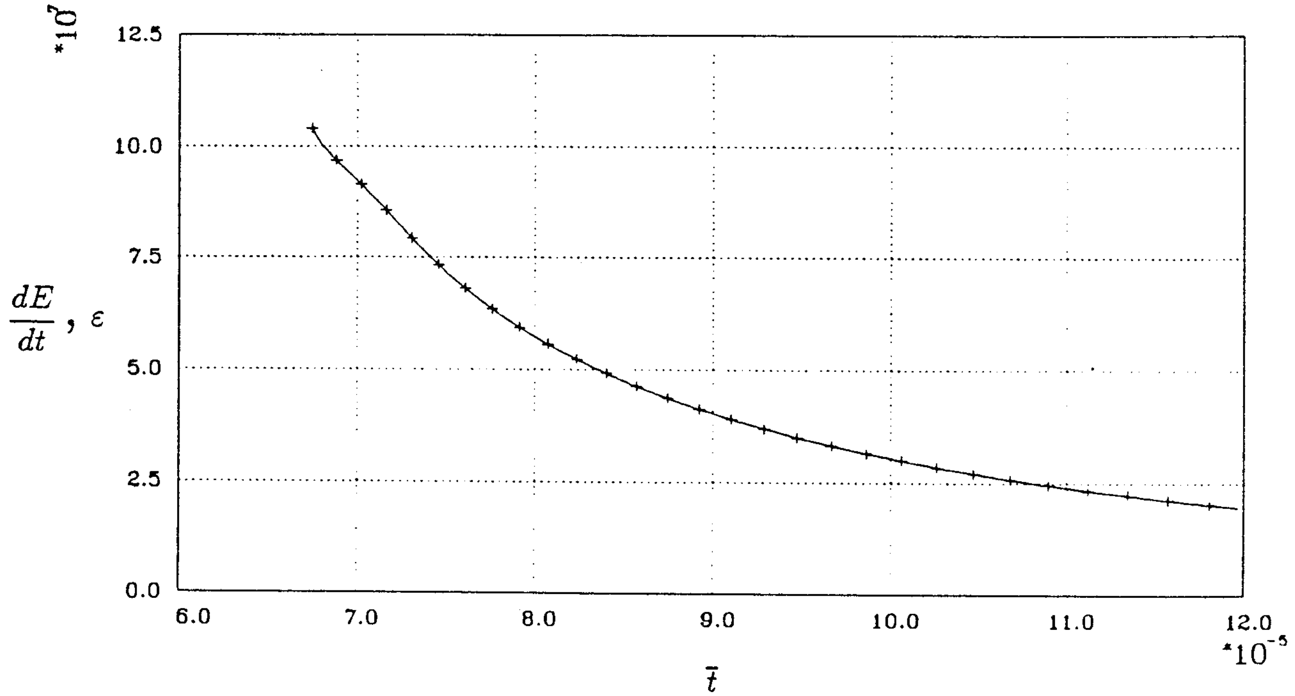


Figure 5.1k. Rate of change of energy (——) and dissipation (+) versus time.

5.2 Asymptotic Results

In this next section, asymptotic solutions are considered. Irrespective of the initial condition, the flow will come back to rest after a long period of time (since the kinetic energy necessarily decays). We are interested in the rate at which this happens, and whether or not it depends on the history of the flow (i.e. such as the initial conditions). We are also concerned with the asymptotic solution as the core radius becomes infinitesimal. In particular, by solving the full nonlinear equations we want to determine the effect of core size and Reynolds number on the vortex ring propagation speed.

5.2.1 Large Time

As described above, a bubble of vorticity surrounds the vortex core and travels with it. Due to viscous diffusion, irrotational fluid is entrained into the bubble as it propagates, the bubble grows, and the ring slows down. The vortex ring asymptotically comes to rest, and the vorticity spreads to the far field with the viscous length scale, $\sqrt{\nu t}$. At this point, the viscous effects have overtaken the nonlinear convection effects. The way in which this occurs and its dependence on the initial conditions is the subject of this section.

The appropriate Reynolds number for this flow is

$$Re_I \equiv \frac{(I/\rho)^{1/2}}{\nu\, t_I^{1/2}} = \bar{t}_I^{-1/2}. \tag{5.2.1}$$

where t_I corresponds to the virtual time when the ring was started from an impulsive, point force (Cantwell, 1986). Note that as the time becomes larger, the Reynolds number becomes smaller.

The limiting behavior of the Navier-Stokes equations as $t \to \infty$ is found by substituting the similarity form of the Stokes variables

$$\begin{aligned} \eta_i &= \frac{x_i}{\sqrt{4\nu t}} \\ u_i &= \frac{I/\rho}{(4\nu)^{3/2} t^{3/2}} U_i(\eta_i) \\ p/\rho &= \frac{I/\rho}{(4\nu) t^2} P(\eta_i) \end{aligned} \tag{5.2.2}$$

into the Navier-Stokes equations, simplifying, and grouping the remaining parameters to form a Reynolds number. Then, taking the limit as $Re \to 0$, gives the self similar form of the Stokes equations:

$$\frac{-3}{2} U_i - \frac{\eta_j}{2} \frac{\partial U_i}{\partial \eta_j} + \frac{\partial P}{\partial \eta_i} - \frac{\partial^2 U_i}{\partial \eta_j \partial \eta_j} = 0 \tag{5.2.3}$$

(Cantwell, 1986). The solution of equation (5.2.3) is the Stokes vortex ring given by equation (3.7.1). The vorticity distribution of this solution is fixed in space.

By dimensional arguments, we expect that the Stokes vortex ring is drifting like $t^{-3/2}$. Indeed this behavior was experimentally observed by Kambe and Oshima (1975). In this paper, they also attempted (without success) to extend the Stokes solution to second order. In a successful attempt to extend the Stokes solution to second order, Rott & Cantwell (1987) found that it was necessary to add a drift to the Stokes solution in the form of a modified independent variable: η_1,

$$\eta_1 = \frac{x - D\,\frac{I/\rho}{\nu}\,\frac{2}{\sqrt{\nu t}}}{\sqrt{4\nu t}} \qquad (5.2.4)$$

where the constant, D, is required to be

$$D = \frac{7}{240\pi^{3/2}2^{1/2}} \qquad (5.2.5)$$

in order for the second order solution to exist. The drift velocity is therefore given by

$$\overline{U} = D\,\overline{t}^{\,-3/2} \qquad (5.2.6)$$

Note that the length scale associated with the drift is of lower order than the length scale of the vorticity due to viscous diffusion.

We are now in a position to compute the flow field and observe the effects of the initial conditions on the limiting behavior. The simplest initial condition, the Stokes solution, has only one independent parameter, the Reynolds number (or the elapsed time, t_I). Unless the Reynolds number is much smaller than one (i.e. very large t_I), the solution is outside the range of validity of the Stokes equations and the nonlinear terms affect the solution. As a result, higher modes are excited (than the dipole). After a sufficient passage of time, the viscosity damps these higher modes and the solution returns to the Stokes distribution of vorticity. Because the Stokes solution is self-similar in time, there is a frame of reference in which it is steady – contours of $\varpi\overline{t}^2$ with the coordinates $r/\sqrt{\nu t}$, in a frame of reference which is translating with the speed of the ring. This is convenient for observing the departure from the Stokes solution and the subsequent return to it. Unless otherwise mentioned in this section, these are the contours and axes plotted. In addition, for each case we plot the drift, $\overline{U}\overline{t}^{3/2}$, as a function of time.

The first case considered is the Stokes solution, at an initial Reynolds number, Re_{I_o}, of 30. Vorticity contours ($\varpi\overline{t}^2$) in the meridional half plane are shown in figures 5.2a through 5.2d, and the time dependent drift is shown in figure 5.2e, with symbols corresponding to the vorticity plots. In the first frame we see the Stokes solution (fig. 5.2a), which is symmetric fore and aft. By solving the Navier-Stokes equations, we get the solution at a later time, figure 5.2b. This figure shows that the peak has moved faster than the surrounding vorticity, and through the nonlinear term, the symmetry has been disturbed. Futhermore, the quasi-elliptic core is tilted to an angle with respect to the axis of symmetry. At a later time (fig. 5.2c) the vorticity distribution is becoming more Stokes-like and finally (fig 5.2d) it returns to the Stokes solution. The corresponding drift starts out at the constant D_c, and quickly deviates. By the time the solution appears to have returned to the Stokes solution (fig. 5.2d), the drift has returned to D_c to within plotting accuracy.

In the next case, Re_{I_o} is 40. The vorticity contour plots (fig 5.3a through 5.3d) are qualitatively similar but the initially higher Re ring initially moves faster, and the vorticity gradients are larger. Again, the solution asymptotically returns to the Stokes vortex ring (fig 5.3d). From figure 5.3e, we see that $\overline{U}\overline{t}^{3/2}$ initially deviates dramatically from D_c, and then begins to return. In contrast to the previous case, $\overline{U}\overline{t}^{3/2}$ overshoots, and returns to D_c asymptotically from below. An analogy can be drawn between this and a damped spring-mass system where increasing the Reynolds number is analogous to decreasing the damping.

Next, we begin with two Stokes distributions at $Re = 2$ each (fig. 5.4). The two rings merge with very little influence from the nonlinear term (no noticeable asymmetry). For this case the initial drift, $\overline{U}\overline{t}^{3/2}$, is different from D_c. (Here, there is some arbitrariness in the choice of the origin of time. However, for large enough times, this effect becomes insignificant.) After some time, the flow goes to the single Stokes vortex ring with the same Stokes drift, D_c, as the previous cases (the impulse used is that of the two rings combined).

Figure 5.5 shows a starting condition of a single thin ring with an initial Reynolds number, Re_{Γ_o}, of 150 and $a/R = 0.25$. Figures 5.5a through 5.5d show the usual contours in the Stokes coordinates. For comparison, figures 5.5a' through 5.5d', show vorticity contours at the same times in stationary physical coordinates. Note that the ring travels a fixed distance, and at large times, the flow at the origin is uniform. The drift at large times also asymptotes to the same value as the above cases, D_c.

The value of D_c found by computations is 0.003703 which matches to four significant digits the value in equation (5.2.5) determined by Rott.

In summary, the results for the large time behavior show that flows with several different initial vorticity distributions approach the same steady solution for $\overline{\omega}\overline{t}^2$ in a frame of reference which is translating at the ring speed and expanding with viscous diffusion, $r/\sqrt{\nu t}$. The asymptotic solution (in this translating frame) is the Stokes solution. Furthermore, the centroid of the vorticity drifts at a speed, $\overline{U} = 0.003703\ \overline{t}^{-3/2}$ regardless of the initial condition. Finally, the accuracy of the computations is further confirmed by the agreement to 4 significant digits between the computed drift and that obtained from theory by Rott.

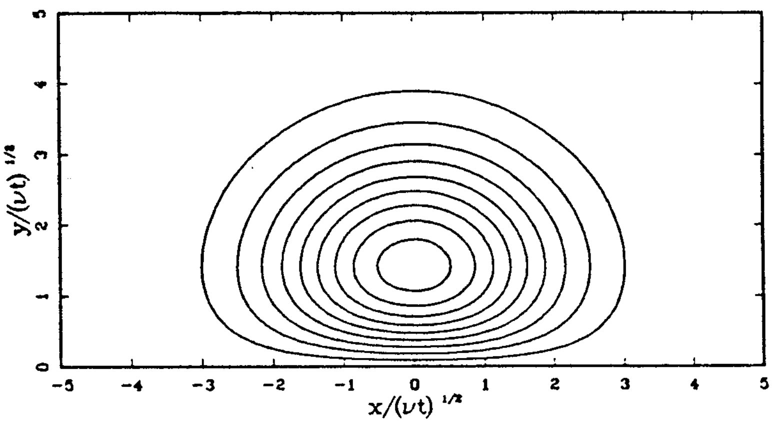


Figure 5.2a.

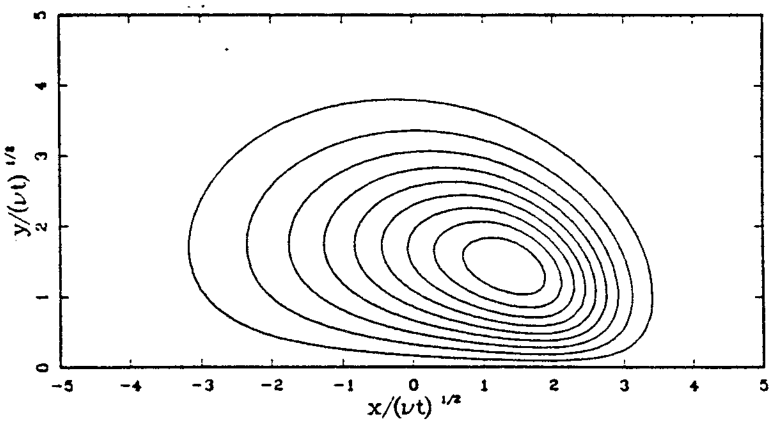


Figure 5.2b.

Figure 5.2. Navier-Stokes calculation starting with a Stokes vorticity distribution at a Reynolds number of 30. (a)-(d) Contours of $\bar{\omega}\ \bar{t}^2$ in a frame of reference translating with the ring speed which is also expanding. (e) Time history of ring speed where dots correspond to the times shown in figures (a)-(d).

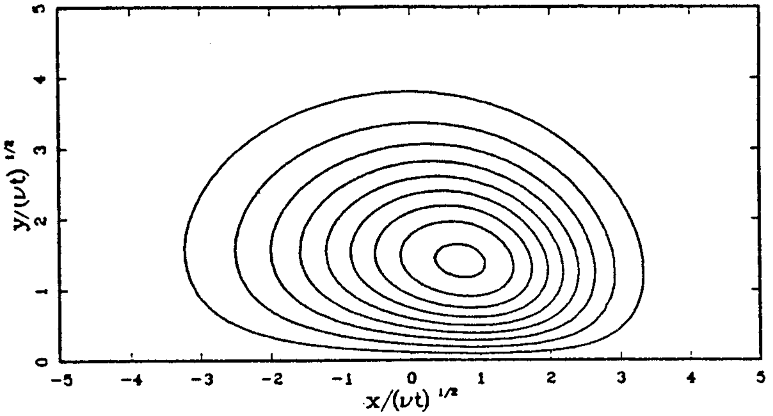


Figure 5.2c.

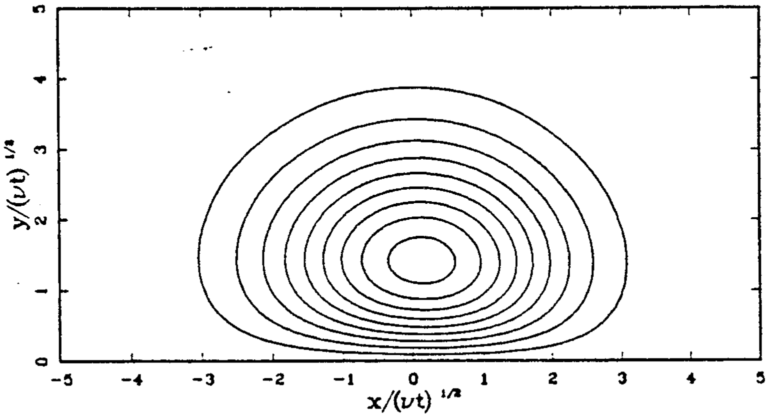


Figure 5.2d.

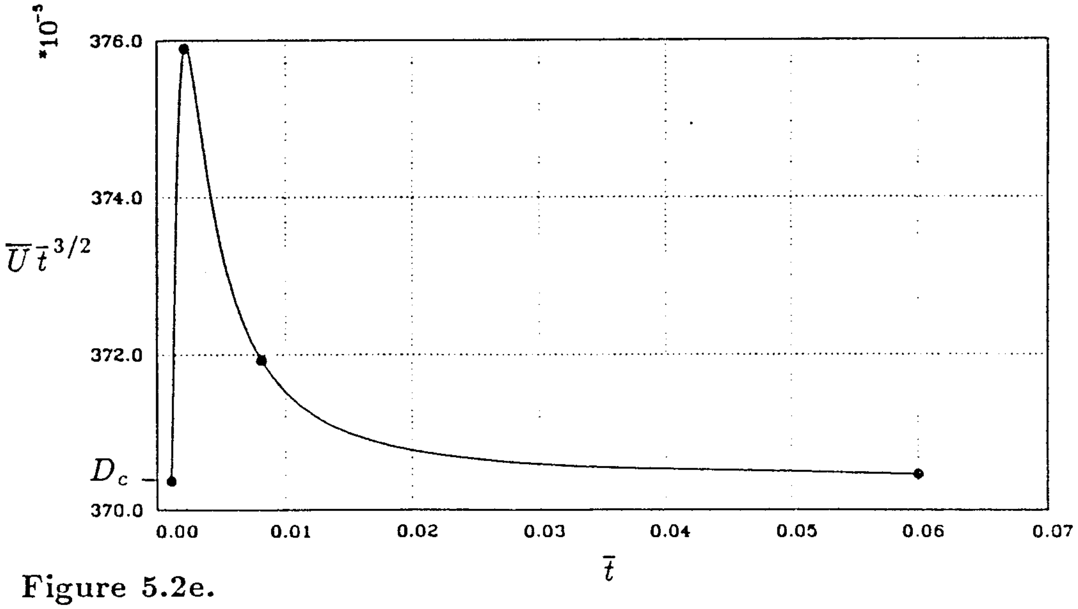


Figure 5.2e.

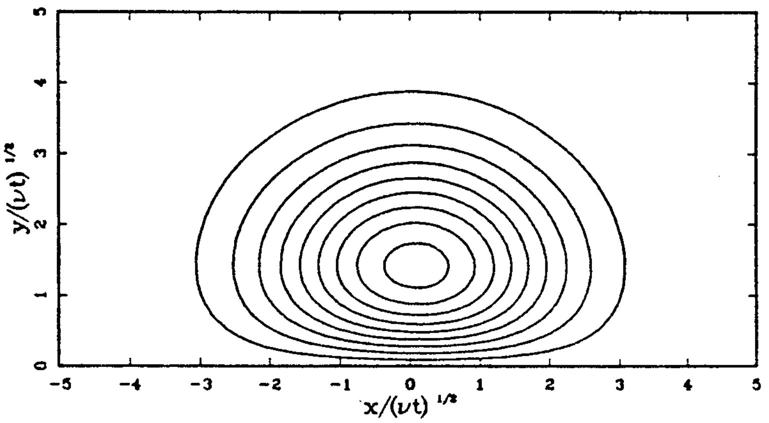


Figure 5.3a.

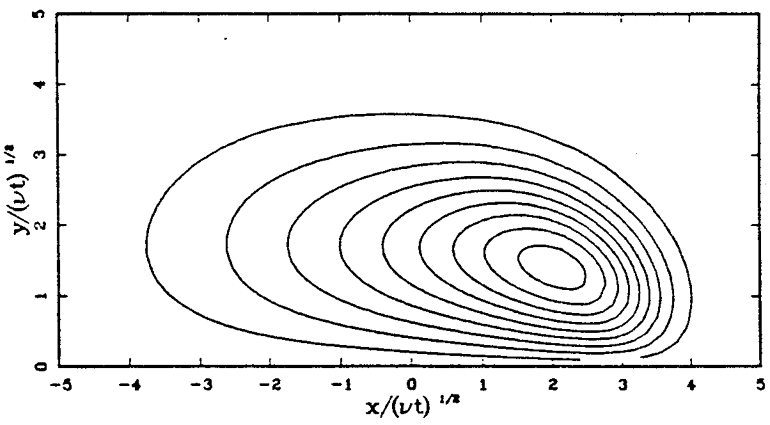


Figure 5.3b.

Figure 5.3. Navier-Stokes calculation starting with a Stokes vorticity distribution at a Reynolds number of 40. (a)-(d) Contours of $\overline{\omega}\ \overline{t}^{2}$ in a frame of reference translating with the ring speed which is also expanding. (e) Time history of ring speed where dots correspond to the times shown in figures (a)-(d).

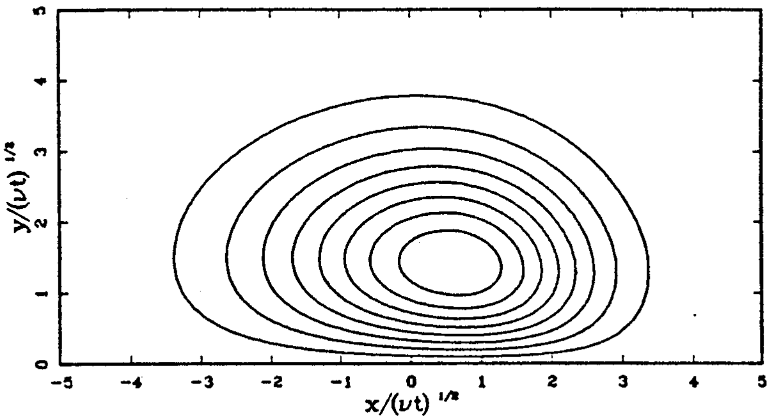


Figure 5.3c.

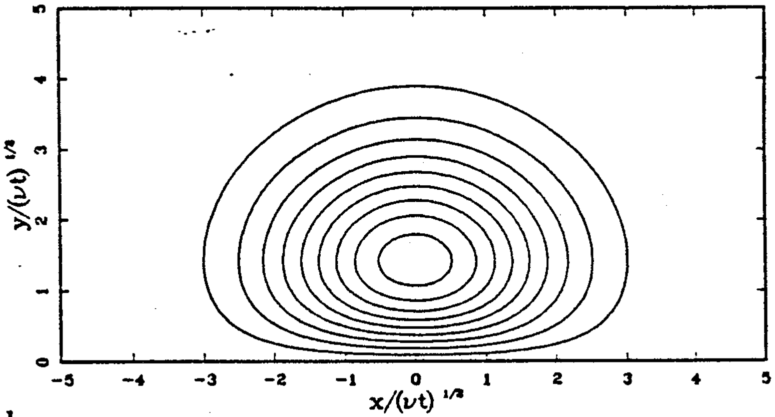


Figure 5.3d.

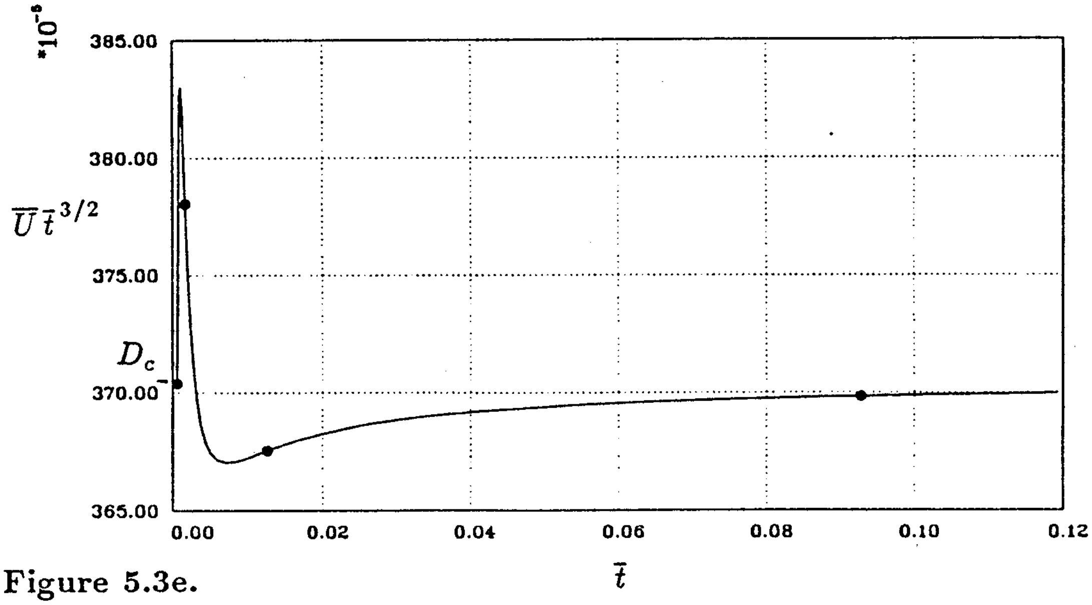


Figure 5.3e.

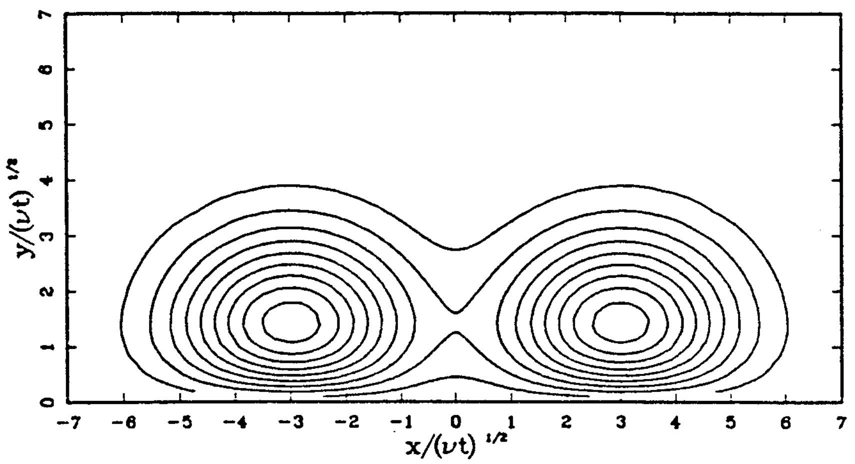


Figure 5.4a.

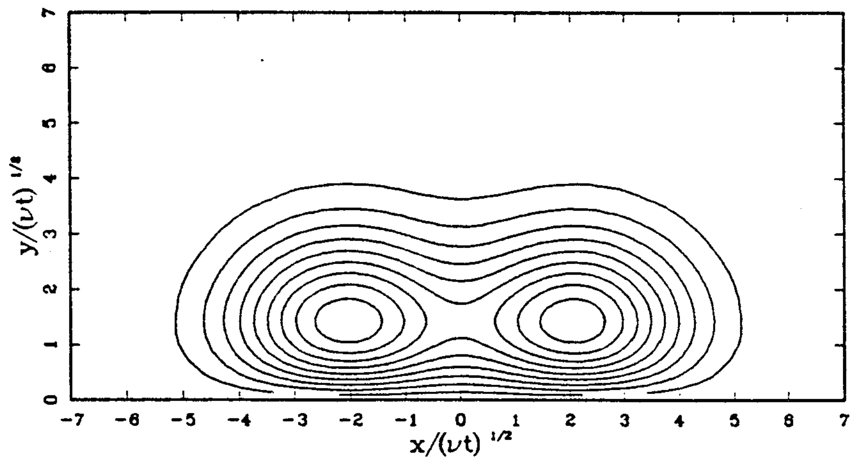


Figure 5.4b.

Figure 5.4. Navier-Stokes calculation starting with an initial condition of two Stokes rings at a Reynolds numbers of 4 each. (a)-(d) Contours of $\overline{\omega}\ \overline{t}^2$ in a frame of reference translating with the ring speed which is also expanding. (e) Time history of ring speed where dots correspond to the times shown in figures (a)-(d).

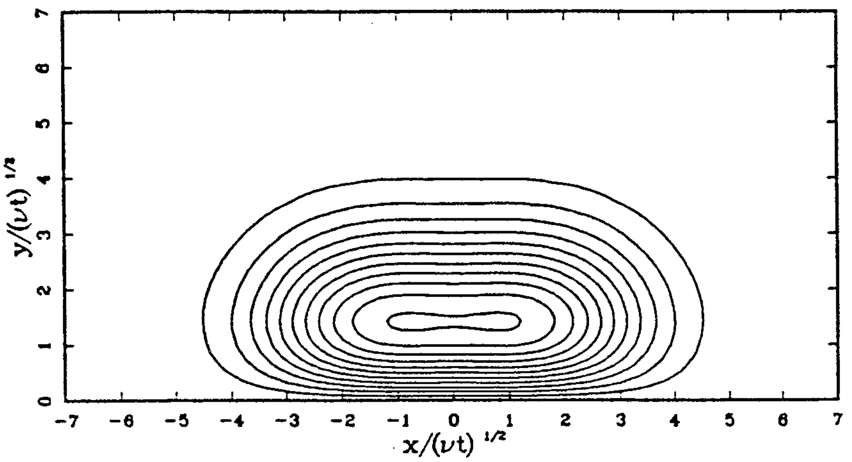


Figure 5.4c.

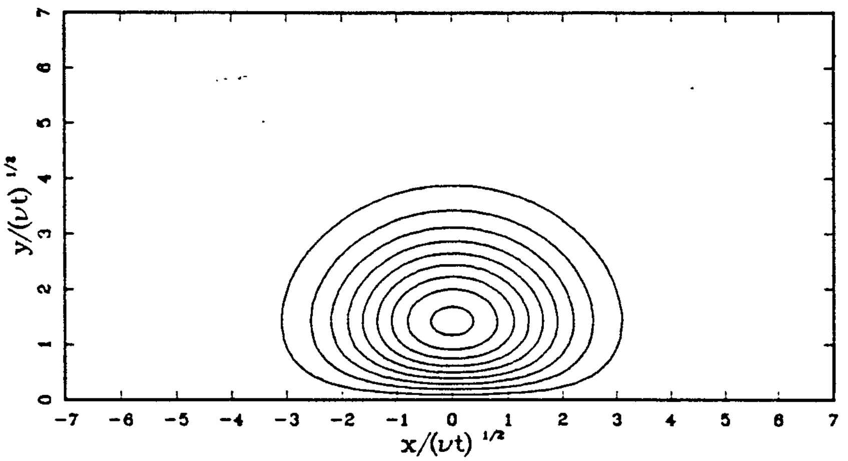


Figure 5.4d.

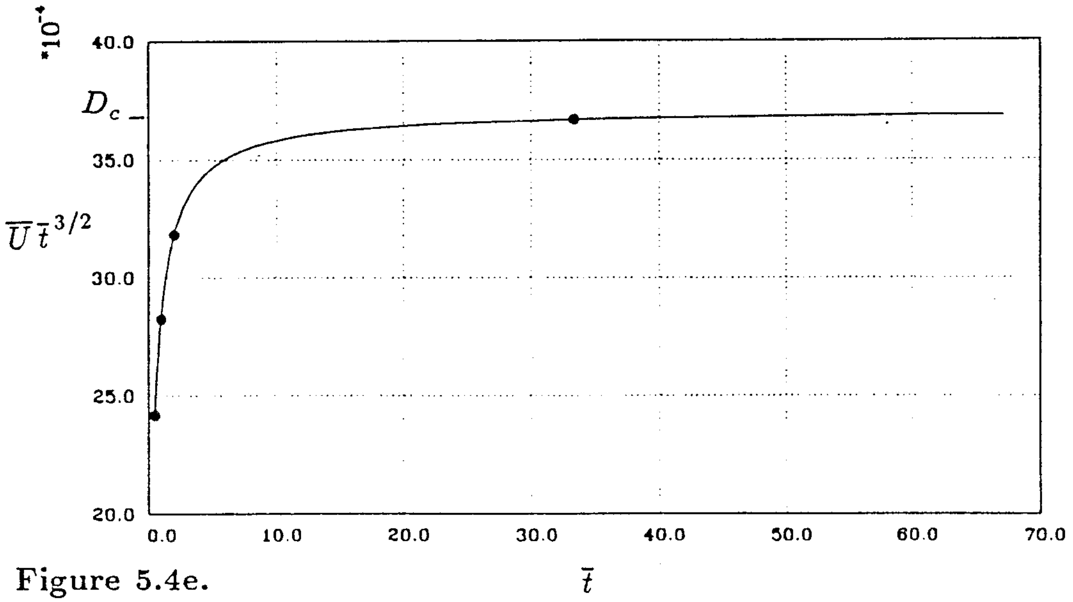


Figure 5.4e.

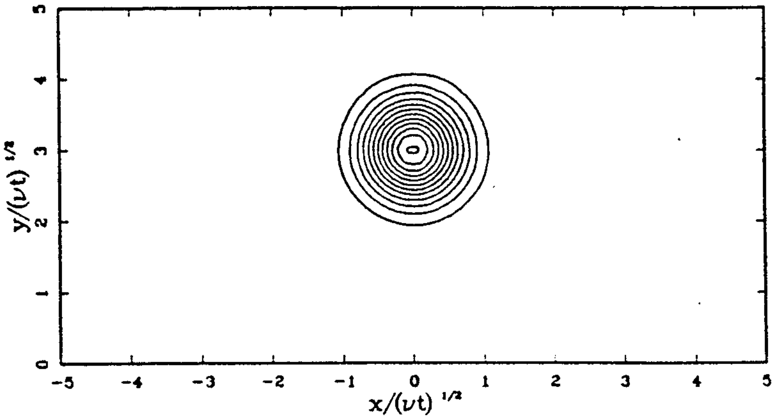


Figure 5.5a.

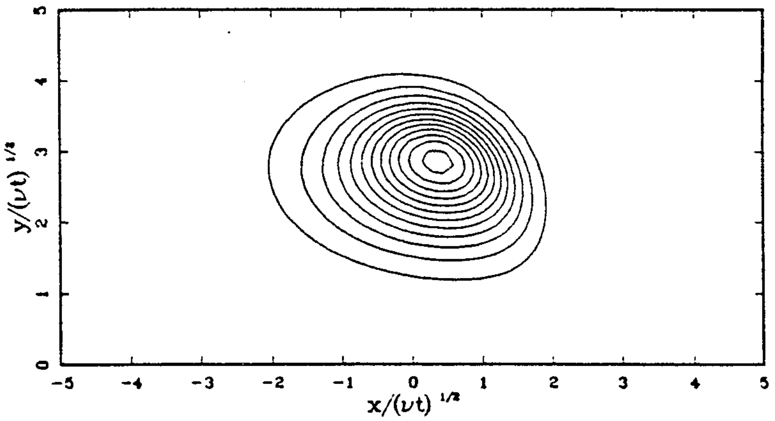


Figure 5.5b.

Figure 5.5. Navier-Stokes calculation starting with an initial condition of a thin ring, $a/R = 0.25$ and $\Gamma/\nu = 150$. (a)-(d) Contours of $\overline{\omega}\ \overline{t}^2$ in a frame of reference translating with the ring speed which is also expanding. (a')-(d') Contours of $\overline{\omega}$ in a fixed frame of reference, corresponding to figures (a)-(d). (e) Time history of ring speed where dots correspond to the times shown in figures (a)-(d) and (a')-(d').

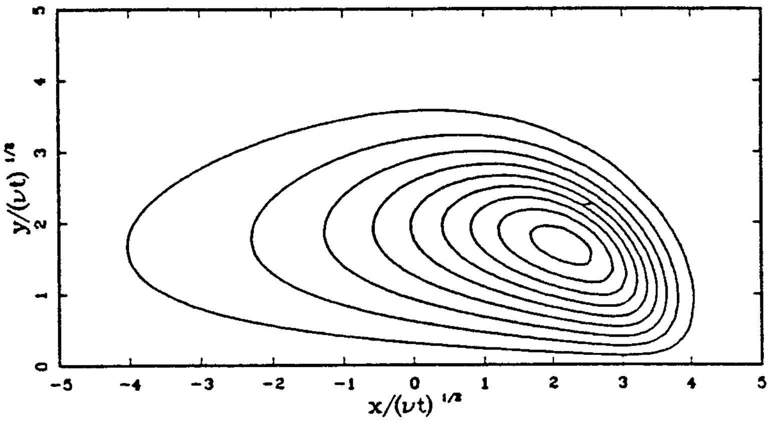


Figure 5.5c.

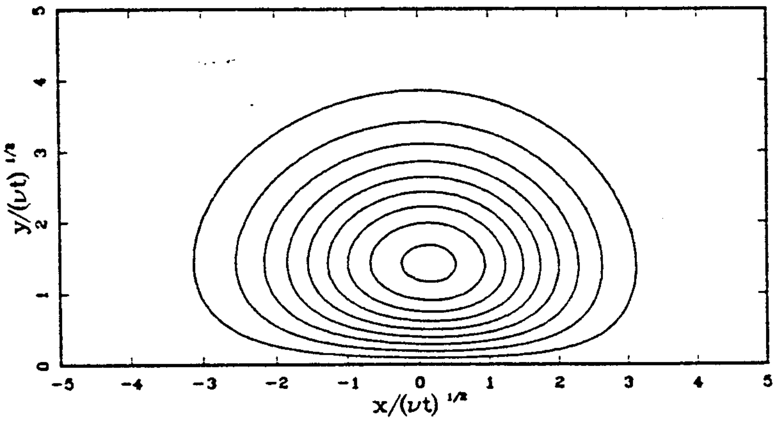


Figure 5.5d.

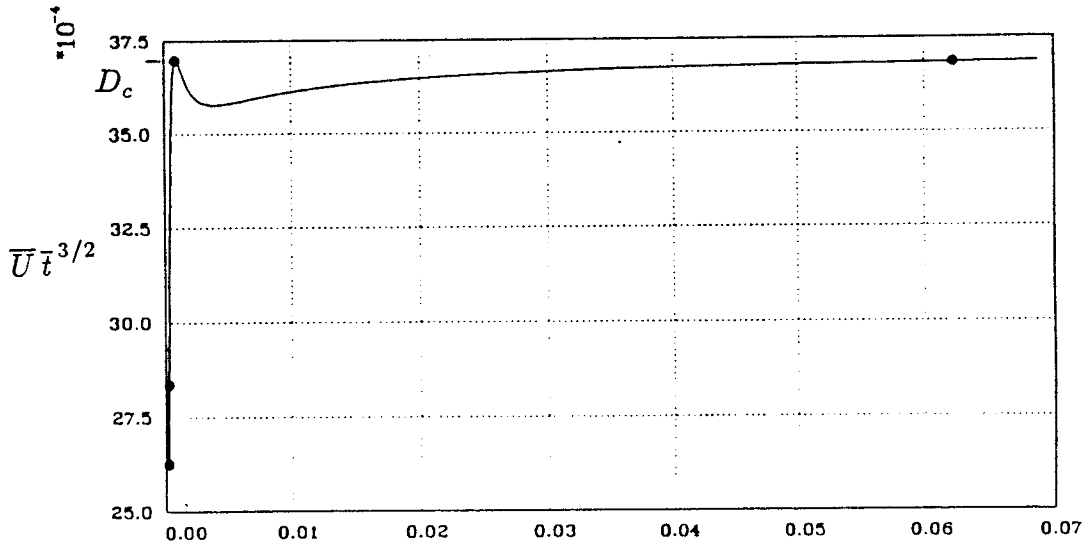


Figure 5.5e. $\bar{t}$

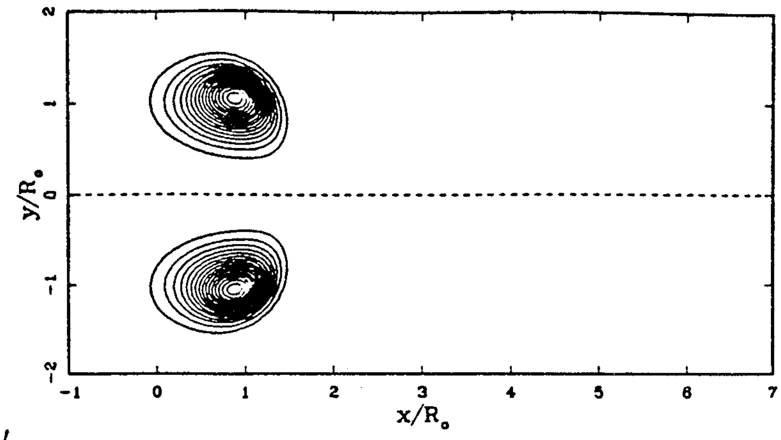


Figure 5.5a′.

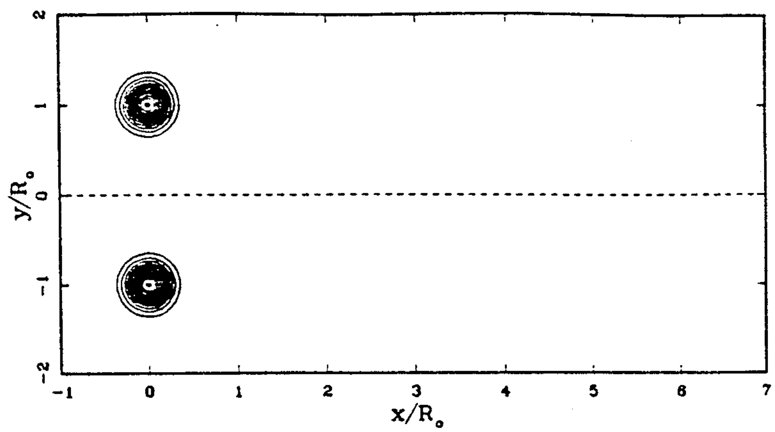


Figure 5.5b′.

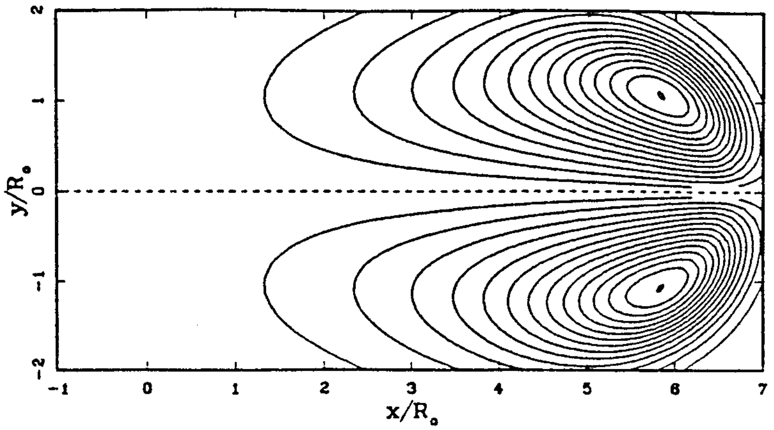


Figure 5.5c′.

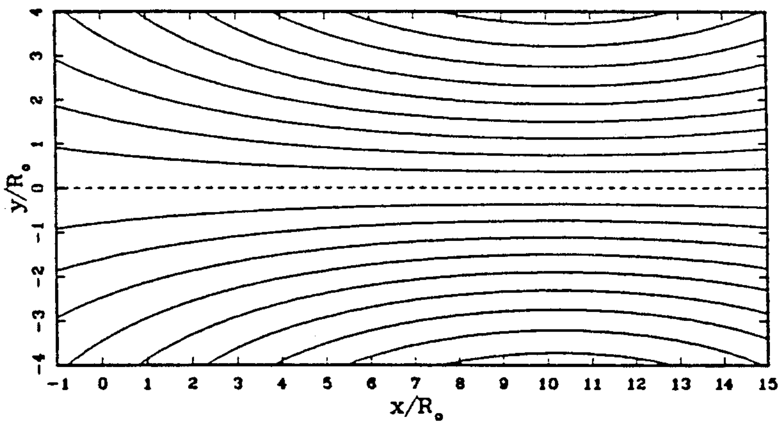


Figure 5.5d′.

5.2.2 Small Time

In 1970 Saffman extended the theory for the propagation speed of steady inviscid vortex rings (Fraenkel, 1970) to that of unsteady vorticity distributions, resulting from the presence of viscosity. The underlying assumption in Saffman's theory is that the core radius is vanishingly small ($a^2 << R^2$). In addition, he suggests a general definition of ring speed in a three-dimensional unsteady flow, which has many merits (and it is a vital step in the theory). In this section, computed solutions of the full Navier-Stokes equations in the thin ring regime are compared to the theory. Motivations for this study are: to provide a check to the code, to assess the range of validity of the theory, and to determine the next order correction to the theory. First, we describe the theory, next we expound the interesting issues, and then discuss the results.

Saffman's Theory

Since the flow is not steady in any reference frame, it is necessary to define what is meant by the velocity of the ring. For three-dimensional unsteady flow, Saffman used a centroid defined by

$$\mathbf{X} = \frac{1}{2}\int_V \frac{\mathbf{r}\times\omega\cdot\mathbf{I}}{I^2}\,\mathbf{r}\,dV \tag{5.2.7}$$

where $\mathbf{r}$ is the position vector relative to some fixed point. The ring speed is therefore

$$\mathbf{U} = \frac{d\mathbf{X}}{dt}. \tag{5.2.8}$$

Saffman's formula is easily checked for several special cases where (5.2.7) reduces to the obvious centroid: a pair of rectilinear line vortices, a vortex ring with an infinitesimal cross section, and Hill's spherical vortex. Also note that the denominator, impulse, is the fundamental conserved quantity. Therefore, $\mathbf{X}$ is the centroid of impulse elements. For axisymmetric flow, equation (5.2.7) reduces to

$$X = \frac{\int \omega\, x\, y^2\, dA}{\int \omega\, y^2\, dA} = \frac{\int \omega\, x\, y^2\, dA}{I} \tag{5.2.9}$$

where the position of the point in the flow defined by X is independent of the reference point defining the origin.

Saffman showed that the vorticity distribution, to first order, is Gaussian in the limit of small cores. Furthermore, the length scale associated with the core is $(4\nu t_\Gamma)^{1/2}$ where ν is the kinematic viscosity and t_Γ is the time since the ring was concentrated on a circular line. With this, he found that the propagation speed of a viscous vortex ring, valid in the limit of $\nu t_\Gamma/R^2 \to 0$, is

$$U = \frac{\Gamma}{4\pi R}\left(\ln\Big(\frac{8R}{\sqrt{4\nu t_\Gamma}}\Big) - 0.558 + O\Big[\Big(\frac{\nu t_\Gamma}{R^2}\Big)^{1/2}\ln\Big(\frac{\nu t_\Gamma}{R^2}\Big)\Big]\right) \tag{5.2.10}$$

It is interesting to compare this result with the propagation speed of a thin core with uniform vorticity in an ideal fluid,

$$U = \frac{\Gamma}{4\pi R}\left[\ln\frac{8R}{a} - \frac{1}{4} + O\left(\frac{a}{R}\right)\right]. \tag{5.2.11}$$

known as Kelvin's formula (Lamb, 1945). The viscosity does not affect the logarithmic leading term, and the core radius, a, is replaced by a time dependent length which measures the size of the viscous core, $(4\nu t_\Gamma)^{1/2}$.

Issues

As mentioned above, an assumption in the theory is that the vorticity distribution in the core is locally, a two-dimensional Gaussian. This is affected by the ring curvature. To be precise, the viscous term expressed in Cartesian coordinates is

$$\nu\nabla^2\omega_\phi = \frac{\nu}{y}\left[\frac{\partial}{\partial x^2} + \frac{\partial}{\partial y^2} - \frac{1}{y}\frac{\partial}{\partial y}\right]\omega_\phi y \tag{5.2.12}$$

A local solution is a two dimensional Gaussian when the curvature term (last term on the right hand side of eqn. 5.2.12) is neglected. We are interested in quantifying the effect on the propagation speed with this term included, as well as the curvature in the convective term.

Although it is not discussed by Saffman, there exists a one-parameter family of solutions even with vanishing initial core radius. The parameter is the Reynolds number, $Re_\Gamma \equiv \Gamma/\nu$, which is independent of core size. The question arises therefore, as to whether or not equation (5.2.10) is uniformly valid and how the ring speed depends on Re_Γ. For clarity, equation (5.2.10) is written as

$$\text{if } \frac{\nu t_\Gamma}{R^2} < A, \quad \text{then } \left|U_{diff}\right| < B \times \left|\left(\frac{\nu t_\Gamma}{R^2}\right)^{1/2}\ln\left(\frac{\nu t_\Gamma}{R^2}\right)\right|. \tag{5.2.13}$$

Here U_{diff} is the difference between equation (5.2.10) and the ring speed defined by equation (5.2.7) and (5.2.8) and computed in the Navier-Stokes code. A is, roughly speaking, the range of validity of the approximation, and B is the proportionality constant. They are not unique (one can trade between A and B), but for a given A there is a minimum value B_{min}. The most relevant quantity is the limit of $B_{min}(A)$ as $A \to 0$. If equation (5.2.13) holds, this limit is finite. Furthermore, A and B depend on Re_Γ. If B is finite over the range $0 \leq Re_\Gamma < \infty$ then the error estimate is uniform.

There are several limits being taken at once. We are interested in the limit as $\nu t_\Gamma/R^2 \to 0$, while $Re_\Gamma \to 0$ and $Re_\Gamma \to \infty$. Computing flows in the limit $Re_\Gamma \to 0$ is easy, the other two limits are not. Unfortunately, we are not able to compute the flow at $t_\Gamma = 0$ when the vorticity is concentrated on a line, preventing us from strictly observing the behavior in the limit as $\nu t_\Gamma/R^2 \to 0$. This is not a serious problem, however, if we can start with sufficiently thin rings. As Saffman argued, after a sufficient amount of time has passed (from the initial condition), the only parameters remembered are the ring size, R, and the circulation, Γ. By starting with several initial core sizes, $(t_\Gamma)_o$, the time dependent flow should collapse to the solution which would be followed if the initial core were infinitesimal. The limit as $Re_\Gamma \to \infty$ is also not possible with this code because of the time step restrictions related to the explicit scheme applied to the nonlinear term. Still, a sufficient number of cases are considered at fairly high Re_Γ to give a good indication of the high Re_Γ behavior.

Results

We are looking for small differences, so diagnostics are very important. The usual diagnostics monitored are impulse, circulation, energy and dissipation. The most difficult case is the thinnest ring at the highest Re_Γ, giving an upper bound for the errors. The impulse was accurate to within 0.02% for the duration of these computations. Because the theory applies only for thin rings, it is assumed that the loss of circulation is exponentially small. Computations are stopped, therefore, when the circulation is 99% of the initial value, giving a conservative upper bound. The rate of decay of kinetic energy compared very well with dissipation, which was computed separately from the velocity derivatives. This shows that the viscous term was resolved and that the numerical dissipation introduced by the time integration errors was much smaller than the viscous dissipation. In addition, the "effective radius," defined by

$$R_{eff} = \frac{\int\int y\,\omega\,dx\,dy}{\int\int \omega\,dx\,dy} \tag{5.2.14}$$

was shown both analytically and numerically (to leading order) to vary like $dR_{eff}^2/dt = -2\nu$. This is related to the spreading of the core and, like the kinetic energy, is a fine measure of the effect of viscosity.

Figures 5.6 through 5.9 correspond to Reynolds numbers of 0.01, 50, 100, and 200. In each case, two plots are shown. The first plot, labeled (a) displays the asymptotic theory (eqn. 5.2.10) and four different initial core sizes (a/R = 0.12, 0.15, 0.25, 0.35). The initial conditions are indicated by solid dots. Keep in mind that each initial condition is a Gaussian vorticity distribution and that the core is circular. The lines represent the time history of the ring speed.

In figure 5.6, Re_Γ is nearly zero ($Re_\Gamma = 0.01$). Therefore, the effect of the nonlinear term is very weak and the vorticity remains nearly Gaussian as it spreads (as seen in the results of the previous section). As mentioned above, however, the axisymmetric vorticity is not exactly Gaussian due to the curvature term in equation (5.2.12). The ring speed indicates that the vorticity is nearly Gaussian as predicted by Saffman, since the time histories of the initially thinner rings pass right through the symbols corresponding to two-dimensional Gaussian initial distributions with larger cores. In other words, the curvature effect is very small.

By plotting the difference between the velocity given by the asymptotic theory and that from the computations, U_{diff}, the ordinate is greatly expanded. Figure 5.6b shows the same data (as shown in figure 5.6a) of U_{diff} and an abscissa of $-(\nu t_\Gamma/R^2)\ln(\nu t_\Gamma/R^2)^{1/2}$ (the reason for this form is discussed later). Now the lines are distinguishable, and we can see that the curves merge at later times, however, the differences between the time histories of various initial core sizes is small. At higher Re_Γ, this will no longer be the case.

Figure 5.7 is a similar case except that $Re_\Gamma = 50$. In figure 5.7a, the solid dots show the intial conditions and the lines leaving them are the time histories of the ring speed from the initial condition. Again, the asymptotic theory is shown for comparison. On this scale, the differences are difficult to distinguish, so we again plot the results in the expanded view of figure 5.7b. The (lighter) solid line corresponds to the $Re_\Gamma = 0.01$ case for comparison, which starts from the initial core size, $a/R = 0.12$. Because the velocity is normalized by Γ, the initial condition does not change from the previous case (fig. 5.6). From this figure, we can test the hypothesis that the solution at some time does not depend on the initial conditions. The curves corresponding to the four different initial core sizes form an envelope, which represents the correct solution if we were able to start with an infinitesimal core. This claim is further substantiated by figures 5.8b and 5.9b where the Reynolds numbers are 100 and 200. In the $Re_\Gamma = 200$ case we see that the envelope formed can be approached from either side. To reiterate, by starting with the thinnest possible core, the transient associated with imposing the Gaussian core as an initial condition is smallest (since in the limit of $a/R \to 0$, the Gaussian is to first order correct). By considering several different initial core sizes, we are able to isolate the effect of this transient and see at what point the envelope of the curves forms. This envelope corresponds to the solution which would result if we were able to start with an infinitesimal core.

The results of figures 5.6 through 5.9 are summarized in the next three figures, along with two cases for higher Re_Γ (400 and 800). Figure 5.10 shows a plot of ring speed as a function of time for several different Reynolds numbers, compared to the asymptotic theory. The initial core sizes for all of these cases are $a/R = 0.12$, and because the ring speeds are normalized by Γ, the initial velocities collapse. Bracketing the results is the asymptotic theory on the top which is labeled as such, and on the bottom is the nearly zero Re_Γ case shown by a thick solid line. The $Re_\Gamma = 50$ case is distinctly shown by a long dash line, and the rest of the Re_Γ cases are more or less on top of one another. This suggests that the zero Re_Γ case has the largest disparity from the asymptotic theory at finite cores and that the normalized ring speed at high Re_Γ is independent of Re_Γ. Again it is helpful to expand the picture, so that small differences are more apparent. In figure 5.10b, we show U_{diff} plotted versus Saffman's error estimate, $(\nu t_\Gamma/R^2)^{1/2}\ \ln(\nu t_\Gamma/R^2)^{1/2}$ for each of the cases shown in figure 5.10. Note all of these curves are tending to zero faster than linearly, indicating that Saffman's error estimate was conservative. By trial and error, we came up with an improved error estimate, $(\nu t_\Gamma/R^2)\ \ln(\nu t_\Gamma/R^2)^{1/2}$. Plotting the curves versus this function of time (fig. 5.10c) we see that at zero

Re_Γ, U_{diff} is nearly linear. From the slope of 0.42, we then have a correction to the asymptotic theory of (5.2.10) which shows the effect of a finite core size and of curvature. For higher Re_Γ, the curves are no longer linear, but this previous correction gives an upper bound. The transient behavior is also more apparent in this coordinate system.

In summary, the numerical results are in full agreement with Saffman's theory. Furthermore the error at $Re = 0$, and probably at all Reynolds numbers, is smaller than Saffman's own estimate by a factor $\sqrt{\nu t_\Gamma / R^2}$. The new error estimate holds until the diffusion of viscosity across the axis becomes significant. There is an indication that as Re approaches either 0 or ∞, the velocity approaches a limit which is only a function of the time in viscous units $(\nu t_\Gamma / R^2)$. Finally, the procedure of starting the simulation at a finite time $(\nu t_\Gamma / R^2)_0$ and taking the limit $(\nu t_\Gamma / R^2)_0 \rightarrow 0$ is valid, and our method allowed us to start with thin enough rings to observe the asymptotic behavior.

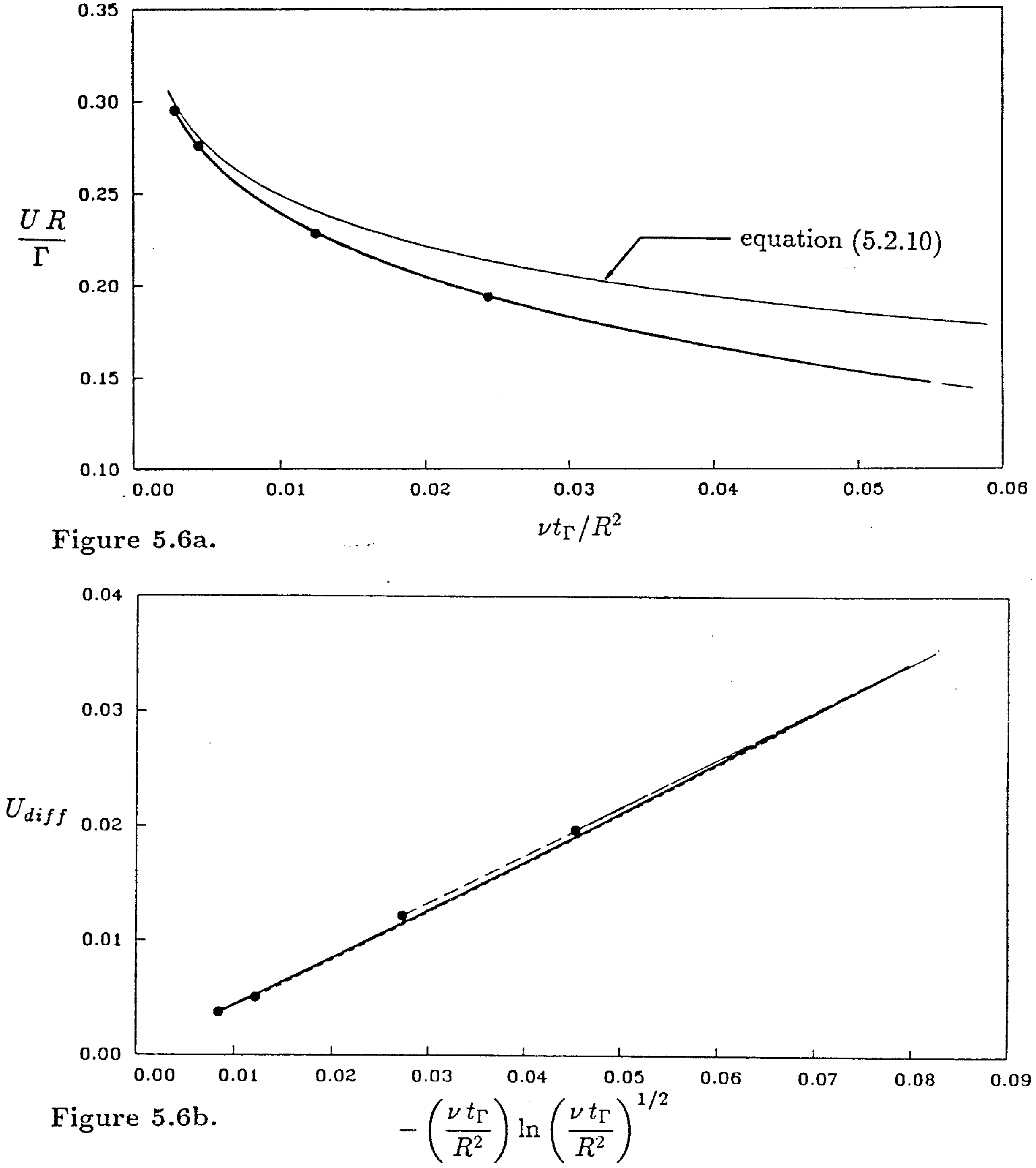


Figure 5.6a.

Figure 5.6b.

Figure 5.6. Propagation speed versus time for vortex rings of strength, $Re_\Gamma = 0.01$, initial core sizes vary: ——, $(a/R)_o = 0.12$; - - -, $(a/R)_o = 0.15$; — — —, $(a/R)_o = 0.25$; — — —, $(a/R)_o = 0.35$. The solid dots are the initial conditions. (a) Comparing computation with theory, versus time, $\nu t_\Gamma/R^2$. (b) Difference between computation and theory, versus $(\nu t_\Gamma/R^2)\ \ln(\nu t_\Gamma/R^2)^{1/2}$.

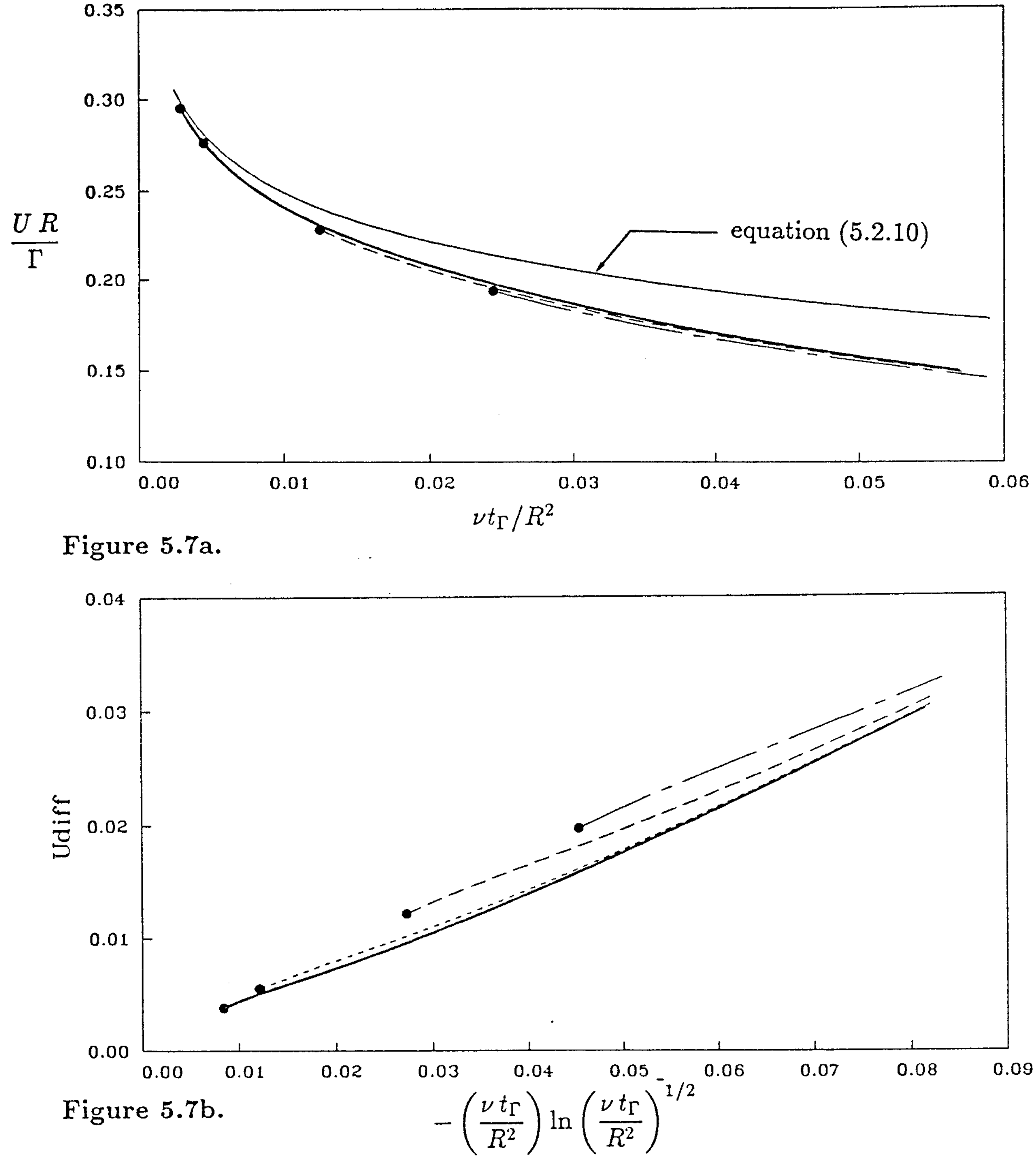


Figure 5.7a.

Figure 5.7b.

Figure 5.7. Propagation speed versus time for vortex rings of strength, $Re_\Gamma = 50$, initial core sizes vary: ——, $(a/R)_o = 0.12$; - - -, $(a/R)_o = 0.15$; – – –, $(a/R)_o = 0.25$; — — —, $(a/R)_o = 0.35$. The solid dots are the initial conditions. (a) Comparing computation with theory, versus time, $\nu t_\Gamma / R^2$. (b) Difference between computation and theory, versus $(\nu t_\Gamma / R^2)\ \ln(\nu t_\Gamma / R^2)^{1/2}$.

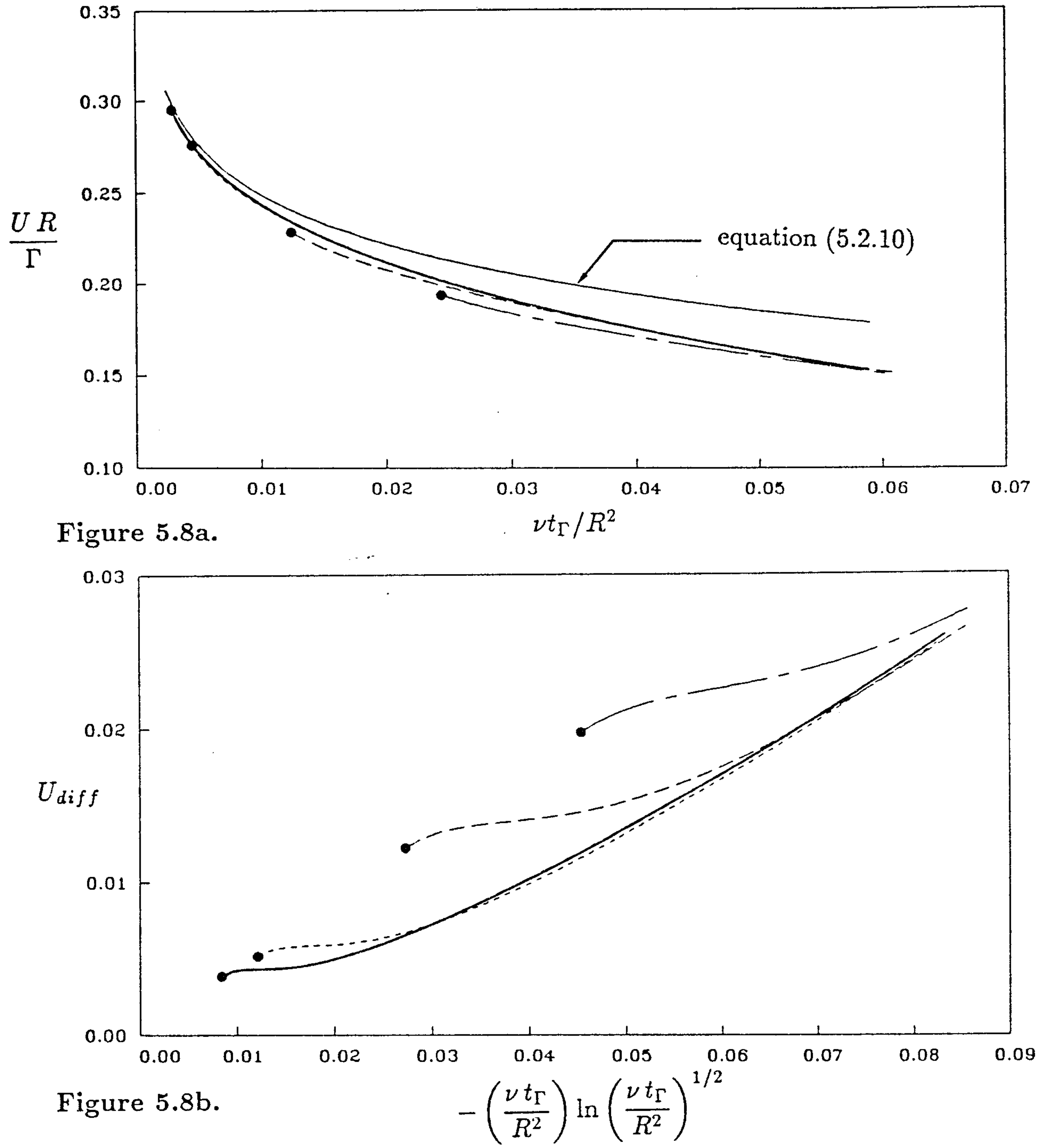


Figure 5.8a.

Figure 5.8b.

Figure 5.8. Propagation speed versus time for vortex rings of strength. $Re_\Gamma = 100$, initial core sizes vary: ——, $(a/R)_o = 0.12$; - - -, $(a/R)_o = 0.15$; – – –, $(a/R)_o = 0.25$; — — —, $(a/R)_o = 0.35$. The solid dots are the initial conditions. (a) Comparing computation with theory, versus time, $\nu t_\Gamma / R^2$. (b) Difference between computation and theory, versus $(\nu t_\Gamma / R^2) \ln(\nu t_\Gamma / R^2)^{1/2}$.

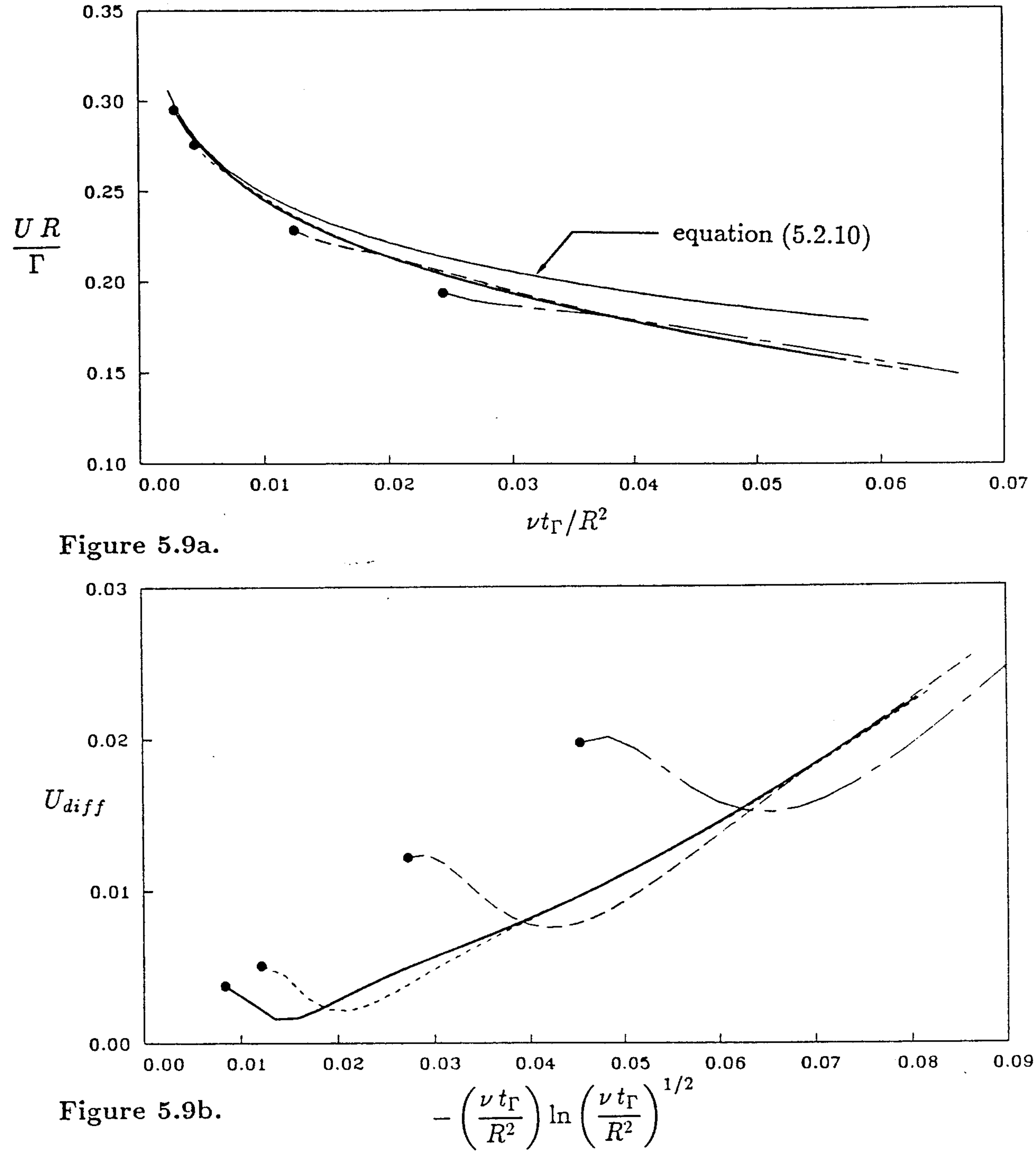


Figure 5.9a.

Figure 5.9b.

Figure 5.9. Propagation speed versus time for vortex rings of strength, $Re_\Gamma = 200$, initial core sizes vary: ——, $(a/R)_o = 0.12$; - - -, $(a/R)_o = 0.15$; – – –, $(a/R)_o = 0.25$; — — —, $(a/R)_o = 0.35$. The solid dots are the initial conditions. (a) Comparing computation with theory, versus time, $\nu t_\Gamma / R^2$. (b) Difference between computation and theory, versus $(\nu t_\Gamma / R^2) \ln(\nu t_\Gamma / R^2)^{1/2}$.

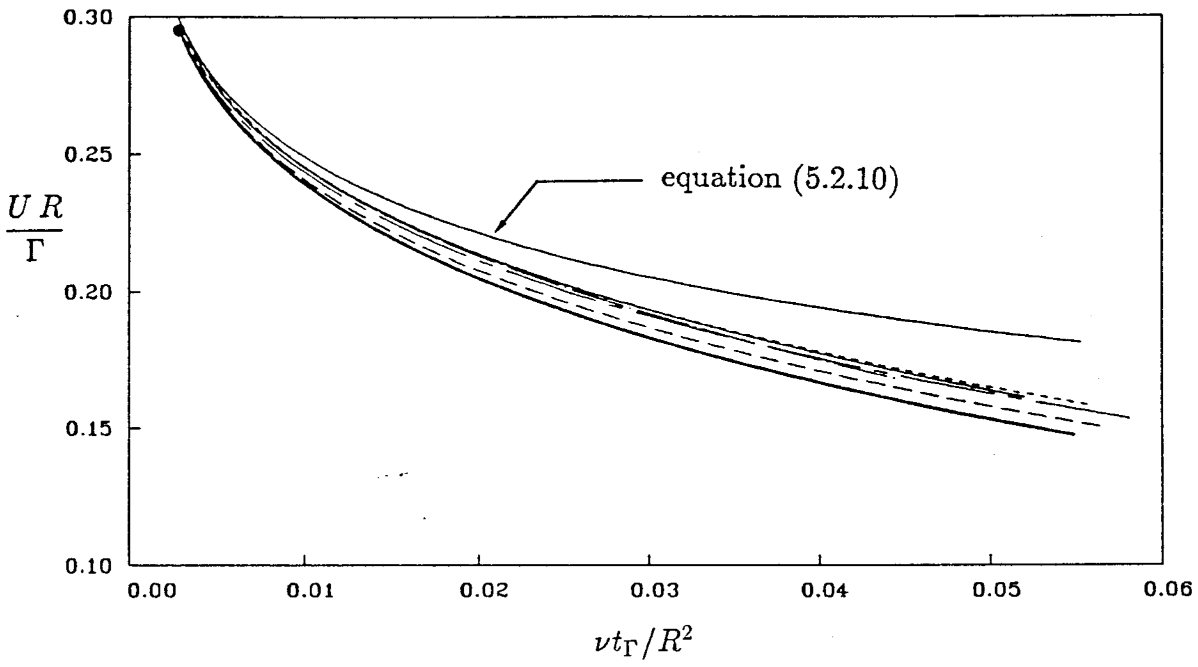


Figure 5.10a.

Figure 5.10. Evolution of vortex rings with $(a/R)_o = 0.12$ and differing Re_Γ: ——, $Re_\Gamma = 0.01$; – – –, $Re_\Gamma = 50$; — — —, $Re_\Gamma = 100$; - - -, $Re_\Gamma = 200$; ——, $Re_\Gamma = 400$; —·—, $Re_\Gamma = 800$. The solids dot are the initial conditions. (a) Comparison of computed ring speed and theory (eqn. 5.2.10), which assumes $(a/R)^2 << 1$. (b) Difference between computed ring speed and theory versus $(\nu t_\Gamma/R^2)^{1/2} \ln(\nu t_\Gamma/R^2)^{1/2}$. (c) Difference between computed ring speed and theory versus $(\nu t_\Gamma/R^2) \ln(\nu t_\Gamma/R^2)^{1/2}$.

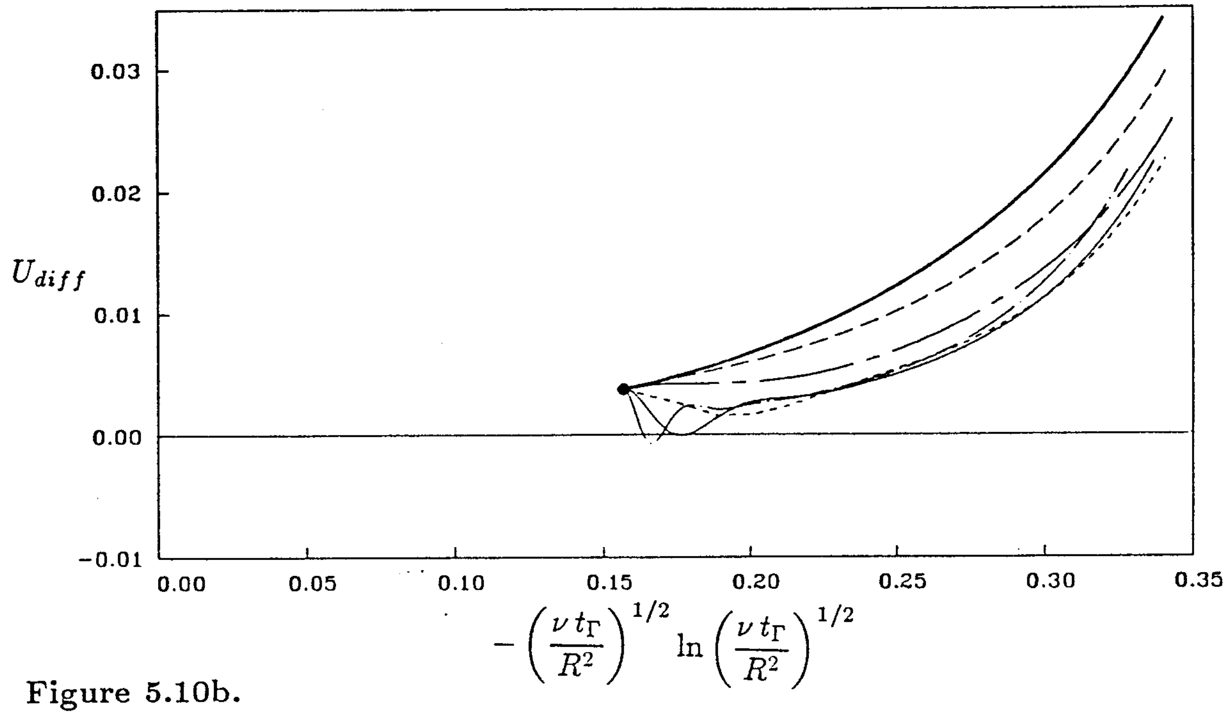


Figure 5.10b.

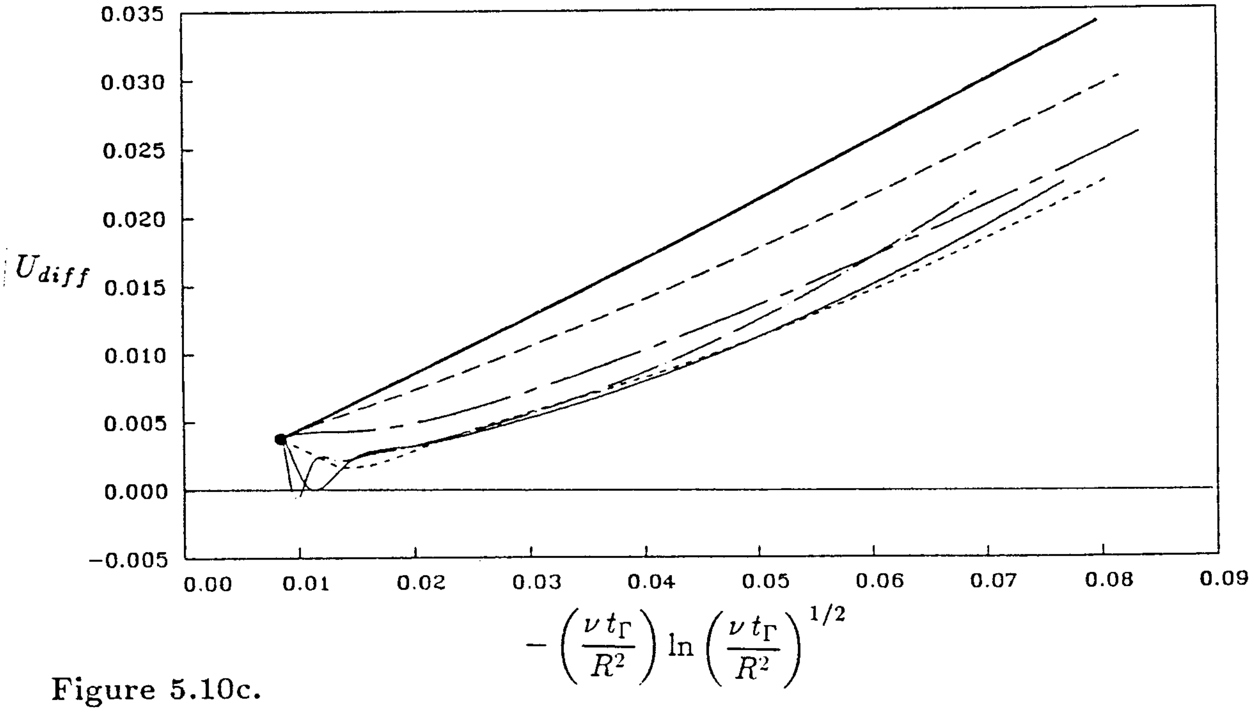


Figure 5.10c.

5.3 Leapfrogging of Two Vortex Rings

There has been some debate in the literature about the nature of interacting vortex rings, in particular, whether or not the classical leapfrogging of inviscid rings could be realized in viscous flows before the vorticity merged due to viscous effects (Oshima *et al.*, 1975, Maxworthy, 1979, Yamada & Matsui, 1978, 1979). Using a smoke wire stretched across the exit of a vortex generator, Yamada & Matsui (1978, 1979), showed the passage of smoke marking the fluid and a subsequent merging of the smoke during the apparent second passage. Furthermore, they stated that three or four slip-throughs were observed by Oshima *et al.* (1977). Because vorticity diffuses more rapidly than smoke, however, it is quite possible that the smoke can indicate a successful passage while the vorticity simply merges (Oshima *et al.*, 1975). Below, vorticity contours are plotted for a Navier-Stokes calculation of two rings with an initial separation distance of one ring radius. The results do show a successful passage before merging. The effects of convection and diffusion on the interactions are distinguished through a qualitative comparison of the Navier-Stokes calculation with an Euler calculations of a similar case by Shariff (1987).

With vorticity of the same sign and with the initial conditions, $Re_\Gamma = 1000$ and $a/R = 0.1$ each, two rings are seen leapfrogging in figure 5.11. In figures 5.11a through 5.11g, the contours are of $\overline{\omega}$, and figure 5.11h shows the propagation velocity of the total vorticity centroid (defined by eqn. 5.2.7 and 5.2.8) as a function of time. As in figure 5.1, the contour levels are the same throughout the simulation, where the higher vorticity levels are shown by thick lines and the lower vorticity levels are shown by thinner lines. As before, the difference in vorticity between the thick lines is a factor of 10 larger than the difference between the thinner lines. For ease of discussion we will name the rings: the ring which is initially on the right is referred to as R1, and the ring initially on the left is R2.

Because cores are very thin, the contours in figure 5.11a are indistinguishable. The outermost contour is $1/100^{th}$ of the initial peak vorticity (there are 10 thick and 10 thin contours lines). In the second frame (Fig. 5.11b), the peak vorticity has decreased roughly 60% and 70% of the initial value for R1 and R2, respectively (i.e. 4 thick lines remain for R1, and 3 for R2).

At the initial time of the calculation (Fig. 5.11a), R1 and R2 induce velocities on each other perpendicular to the axis of symmetry; R1 stretches while R2 contracts. Considering inviscid flow, by Helmholtz' laws, $D(\omega/R)/Dt = 0$ or $D(\omega S)/Dt = 0$, where S is the cross sectional area of a vortex tube. From this, we would expect that $\omega \sim R$ and $S \sim 1/R$. Figure 5.11b shows that we indeed see the correct trend – R1 has a higher peak than R2 and a smaller core. The interactions continue and R2 catches up with R1 in figure 5.11c and has passed it in figure 5.11d. Note that R2 shows straining from the passage: it develops a tail-like structure (Fig. 5.11e) which then pinches off (Fig 5.11f). Starting from figure 5.11e, the process repeats from the beginning where R2 is now the leading ring and R1 is trailing. Because of the first passage and the viscous effects, the rings are significantly fatter and their cores are closer together than they were initially and the second passage is unsuccessful (R1 merges with R2).

It is interesting to note that the velocity of the vorticity centroid oscillates as the rings pass, with the maximum speed occurring when the rings have the same radius and a minimum speed occurring when their centroids are at the same x-location (see figure 5.11h).

The qualitative behavior is similar to the smoke visualizations of Yamada & Matsui (1978), however, it isn't very useful to compare the results in detail since the smoke is not marking the vorticity (especially since smoke is not only ejected into the vorticity layer but is across the entire jet diameter). A passive scalar is needed in the calculation in order to make a comparison with this type of experiment useful. It is quite interesting, however, to compare the viscous calculations with similar inviscid results of Shariff (1987). Shariff solves the Euler equations using contour dynamics for two rings of the same initial separation distance. The vorticity distribution in the core for the two calculations are necessarily different. For the contour dynamics formulation, it is assumed that the vorticity is uniform (constant ω/y) and that it is zero outside the core radius. In the viscous calculation we start with the usual Gaussian vorticity distribution. Furthermore, the initial core-to-ring radius for the contour dynamics case is 0.18. The comparisons, therefore, are qualitative. Figure 12 shows vorticity contours for the two calculations at similar times. The figures are shown on the same scale. In figures 12a and 12a$'$, the core shapes are very similar. As the passage progresses the straining of the ring which passes inside the first shows remarkable agreement for the viscous and inviscid calculations. From the viscous calculation however, the cores are diffusing and the cores are closer together after the passage than the initial separation. The result is that the second passage of the viscous calculation shows merging or pairing.

In summary, leapfrogging was observed for two rings with initial Re_Γ of 1000 each, an initial separation distance of one radius, and initial a/R's of 0.1. Through the first passage, the trailing ring was severly strained due to the inviscid straining field, and a tail forms which eventually pinches off. Because of the viscous effects, the ring which was initially in front merged with the other in an attempted second passage.

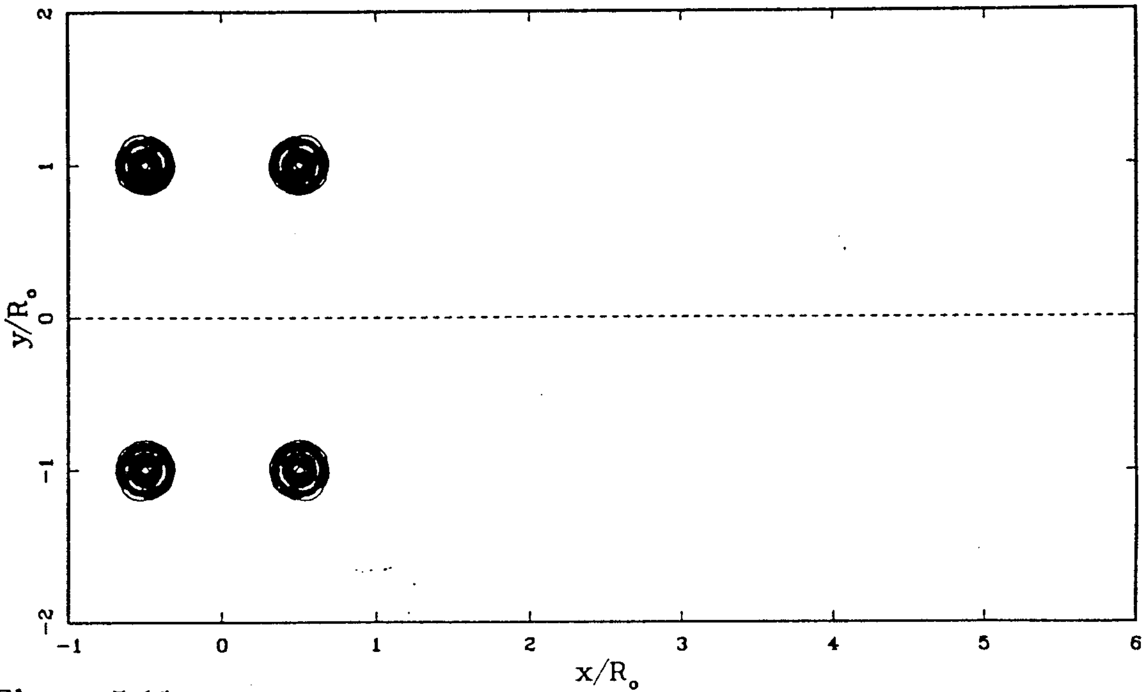


Figure 5.11a.

Figure 5.11. Evolution of two vortex rings of the same sign, each with $Re_{\Gamma_o} = 1000$ and $(a/R)_o = 0.1$. (a)-(g) Vorticity fields at several instants in time. The change in vorticity between the thick lines is a factor of 10 larger than between the thinner lines. Between lines of the same type, the vorticity varies linearly. (h) Ring speed versus time. (i) Circulation versus time. (j) Impulse versus time (k) Rate of change of energy and dissipation versus time.

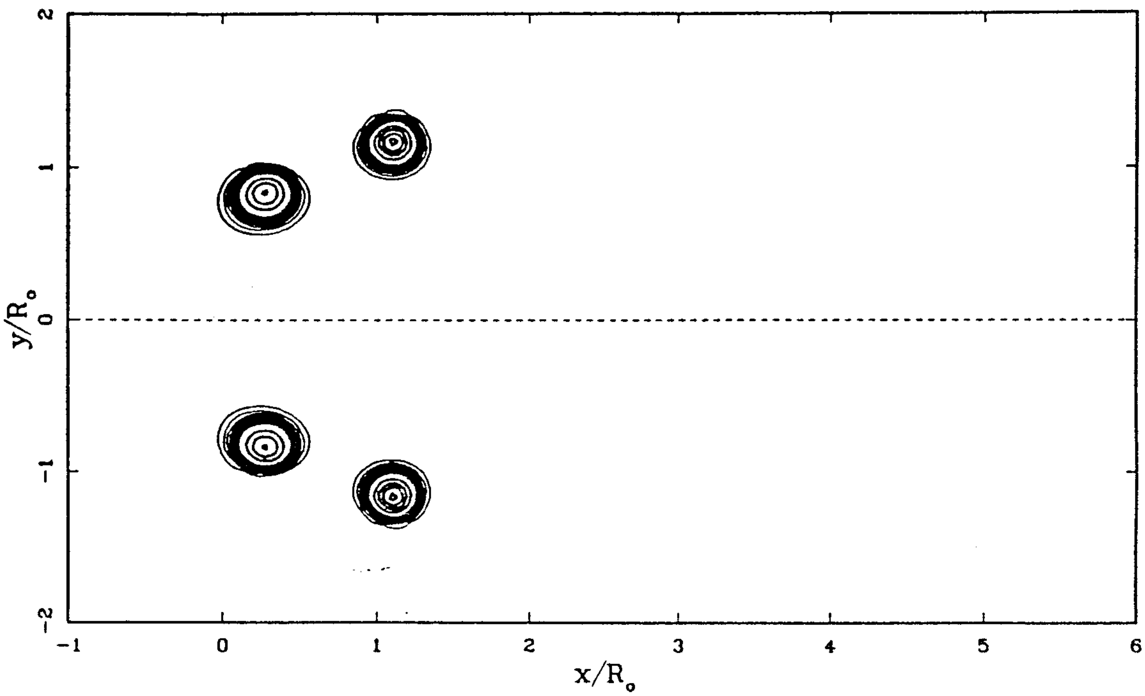


Figure 5.11b.

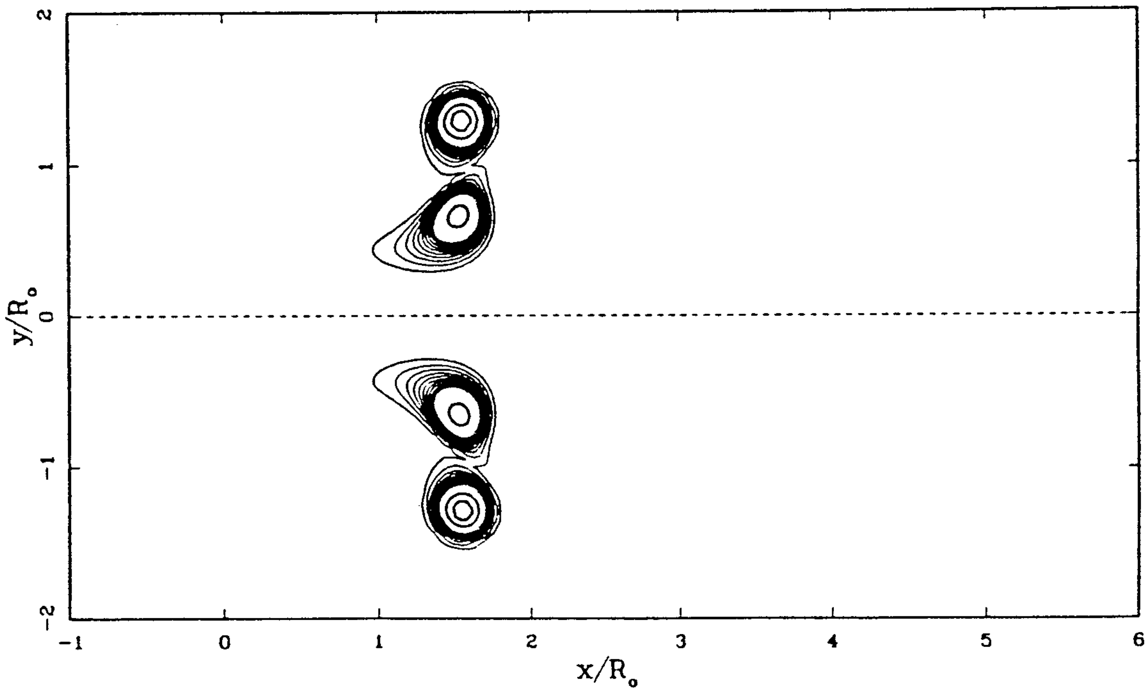


Figure 5.11c.

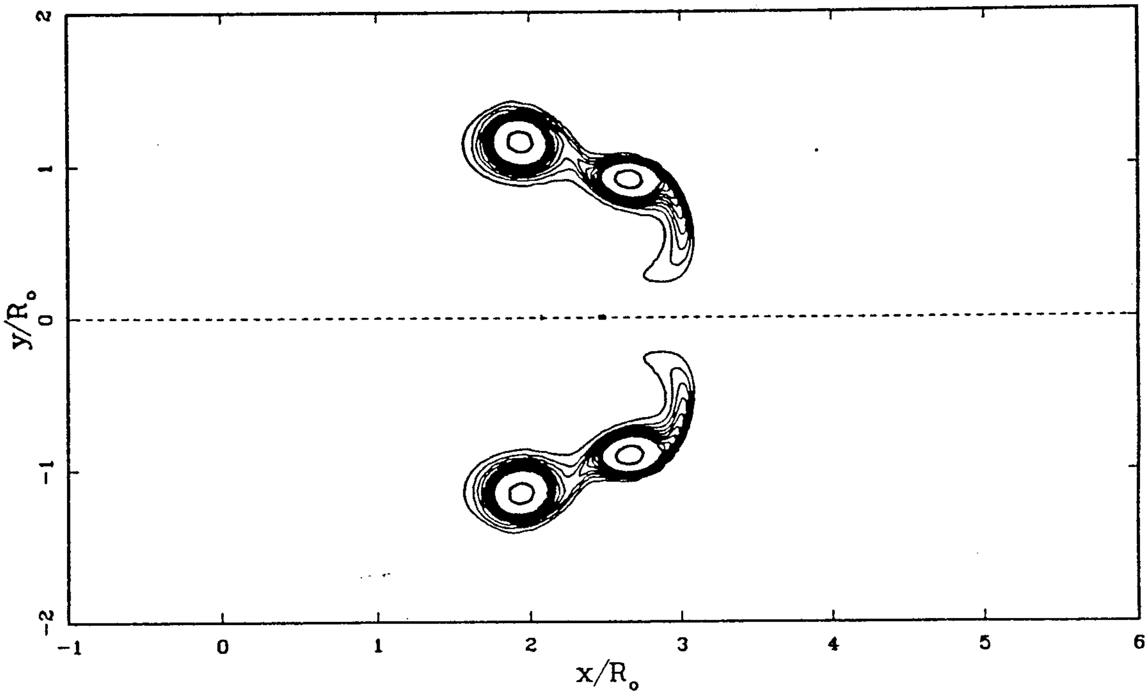


Figure 5.11d.

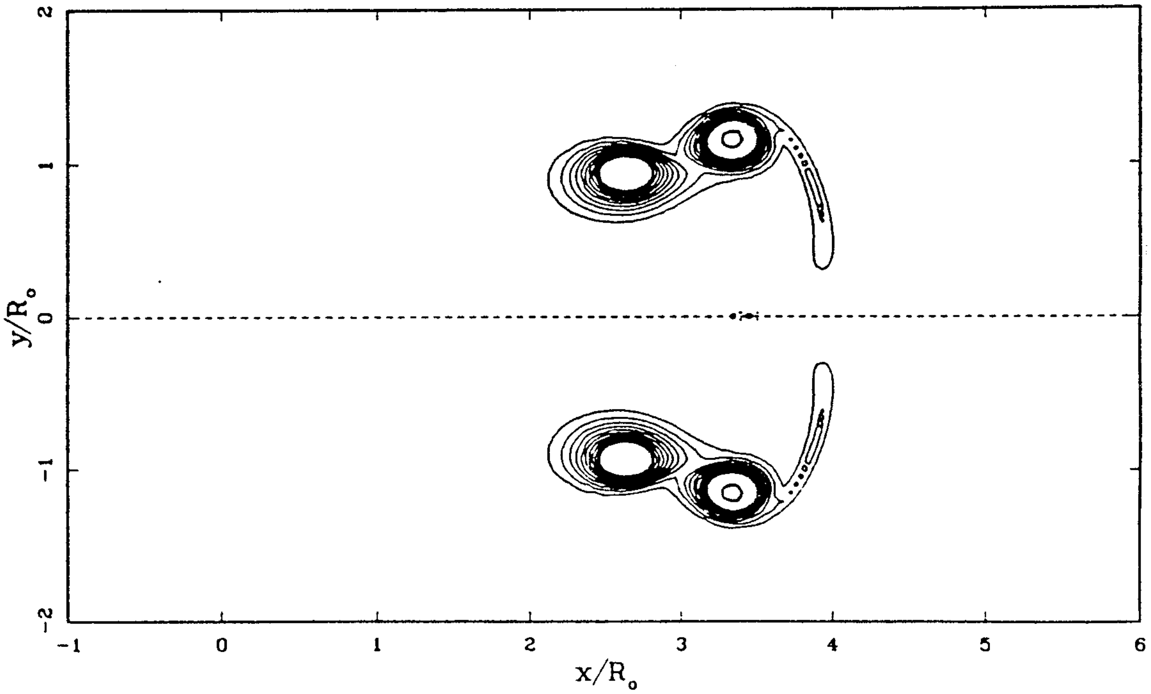


Figure 5.11e.

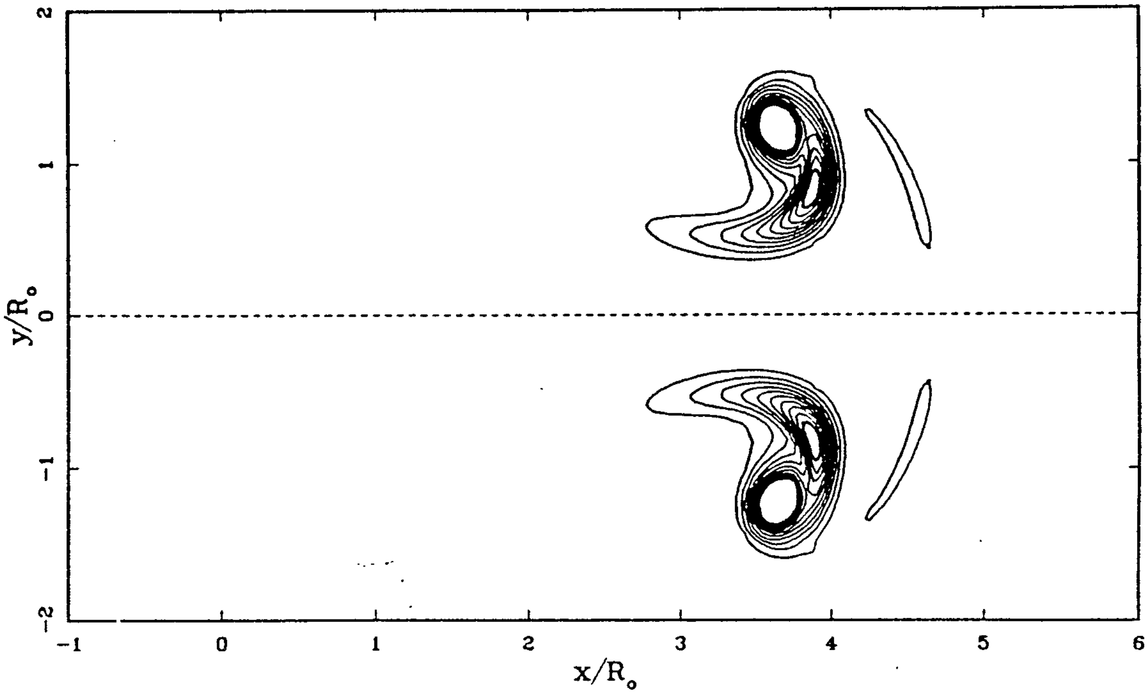


Figure 5.11f.

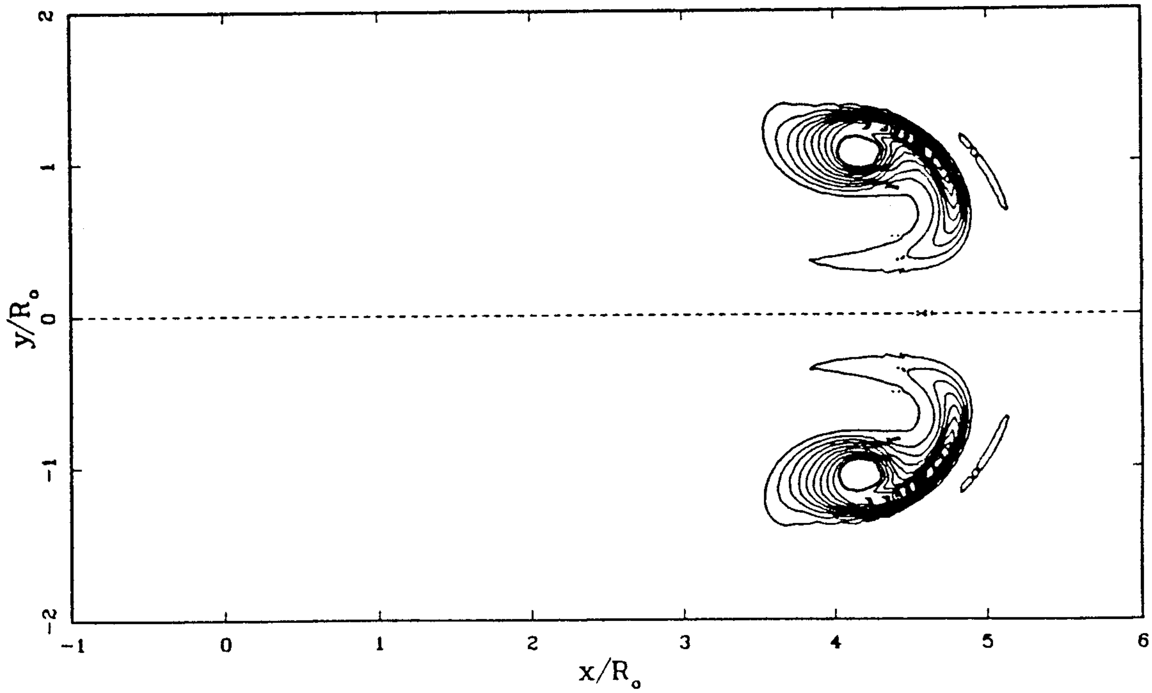


Figure 5.11g.

c-2

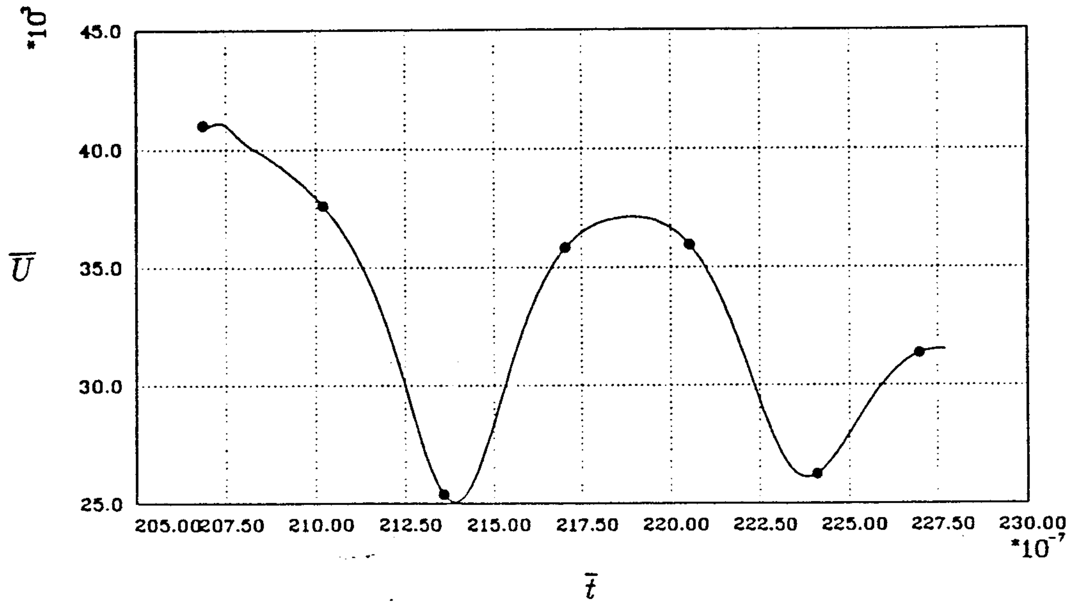


Figure 5.11h. Speed of the ring versus time. Dots correspond to vorticity plots (a)-(g).

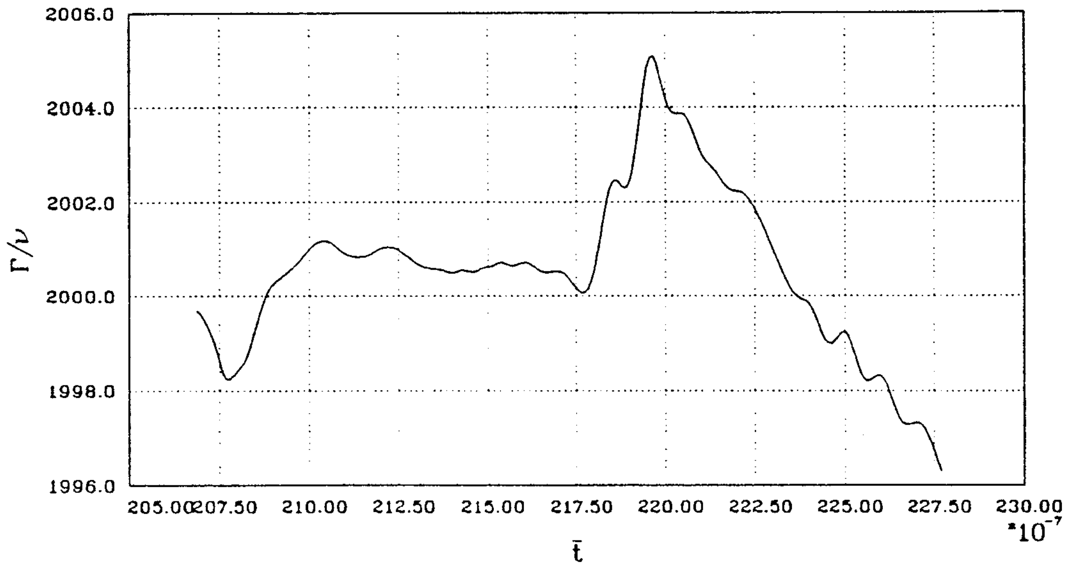


Figure 5.11i. Circulation versus time.

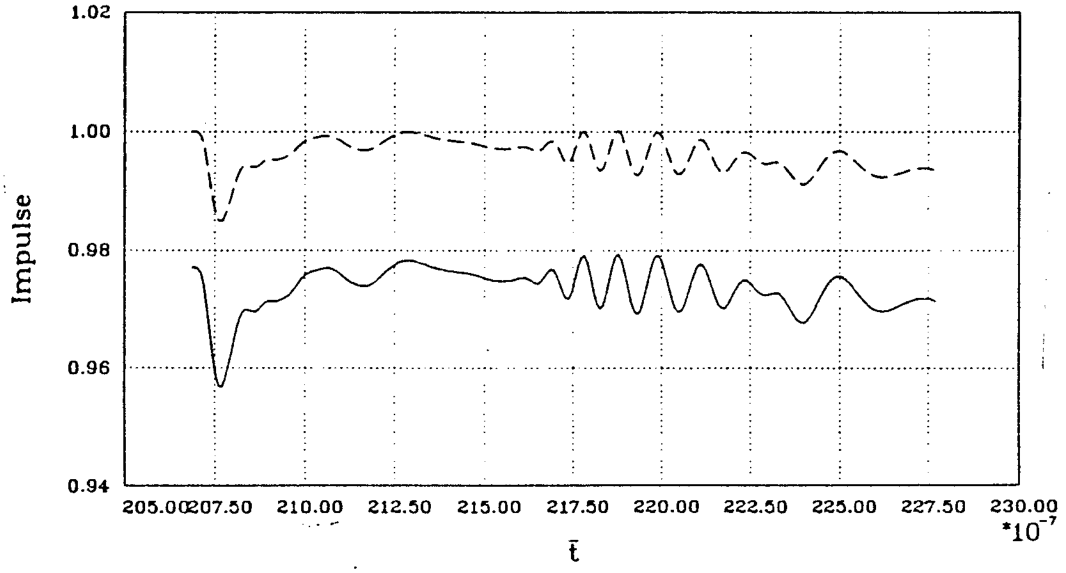


Figure 5.11j. Impulse versus time, computed by a multipole expansion (——) and computed using Gauss Quadrature (- - -).

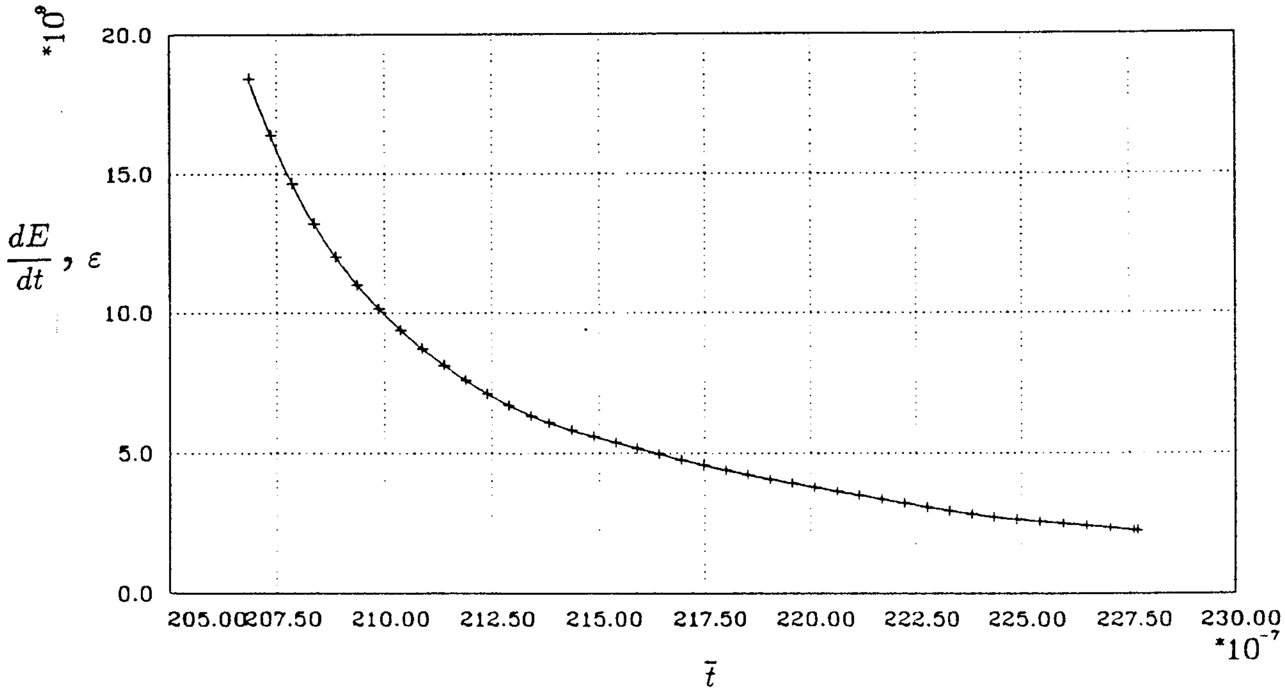


Figure 5.11k. Rate of change of energy (——) and dissipation (+) versus time.

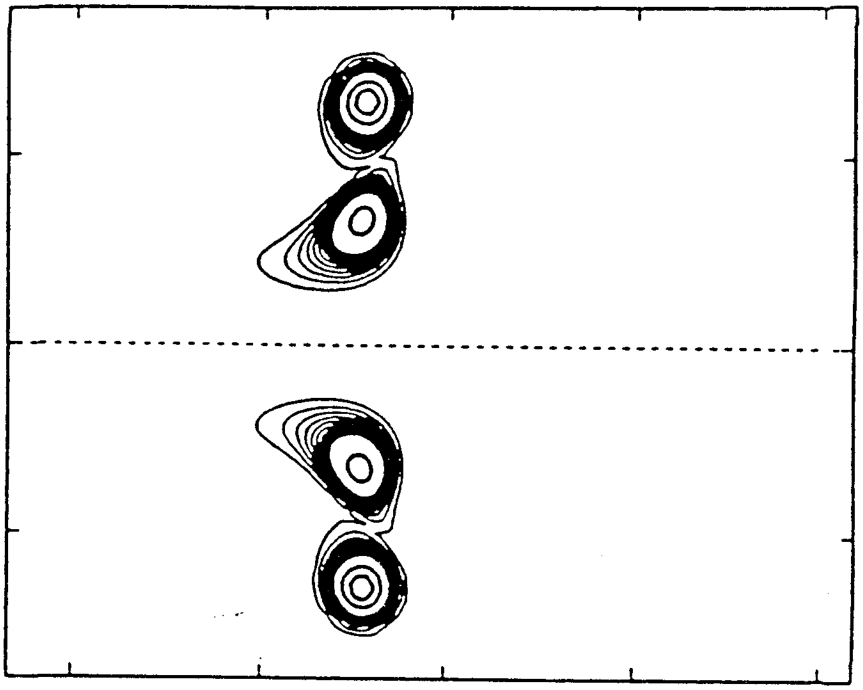


Figure 5.12a. Navier-Stokes calculation (same as fig. 11); $(a/R)_o = 0.1$, Gaussian vorticity in core.

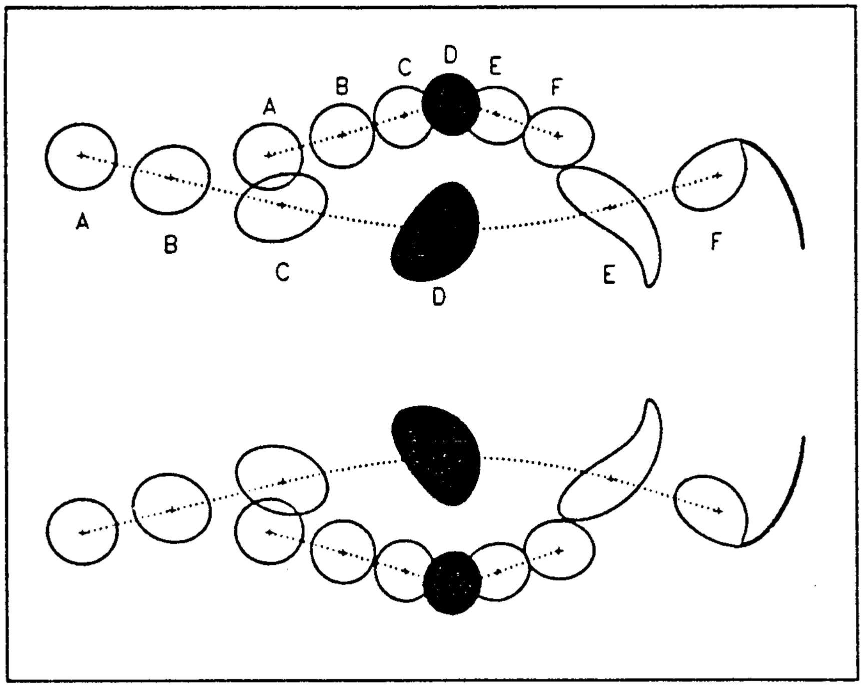


Figure 5.12a′. Euler calculations of vortex rings by Shariff (1987); $(a/R)_o = 0.18$, uniform vorticity in core.

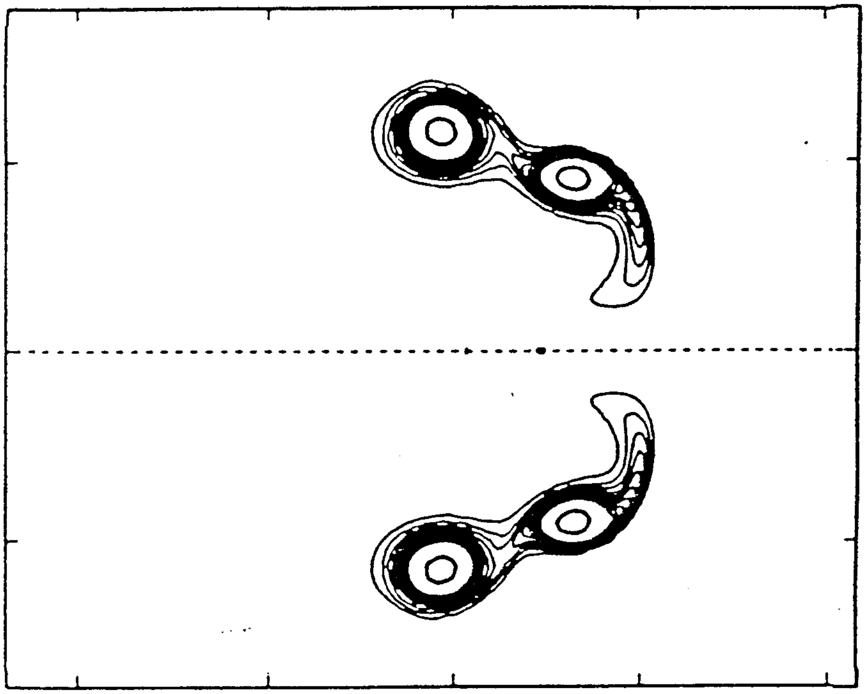


Figure 5.12b. Navier-Stokes calculation (same as fig. 11); $(a/R)_o = 0.1$, Gaussian vorticity in core.

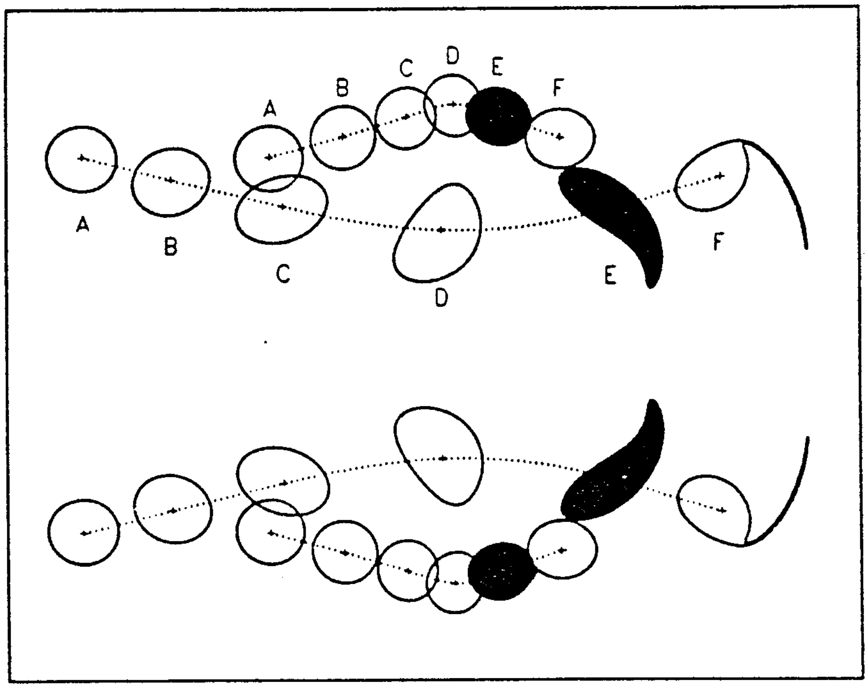


Figure 5.12b′. Euler calculations of vortex rings by Shariff (1987); $(a/R)_o = 0.18$, uniform vorticity in core.

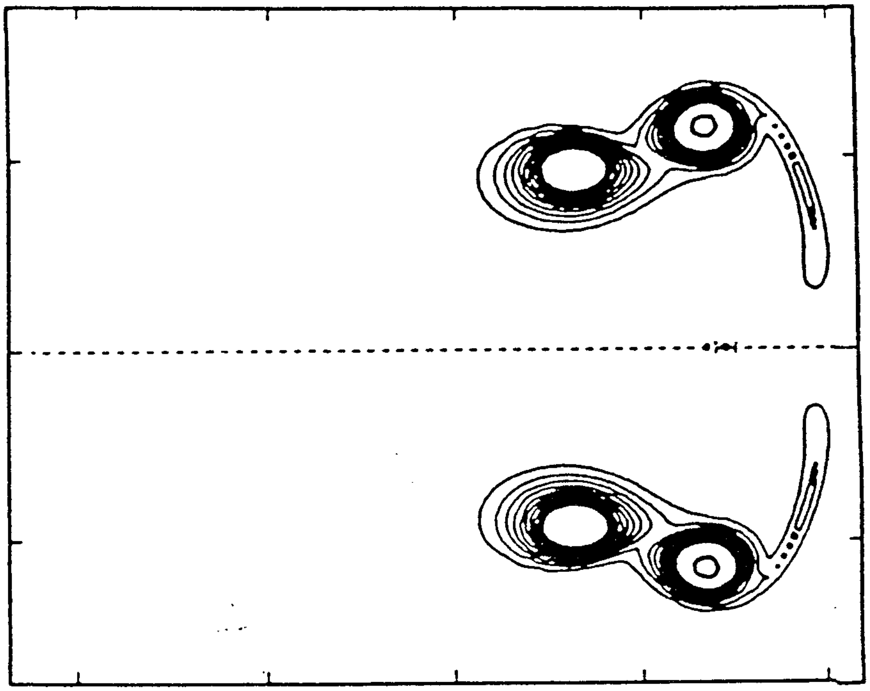


Figure 5.12c. Navier-Stokes calculation (same as fig. 11); $(a/R)_o = 0.1$, Gaussian vorticity in core.

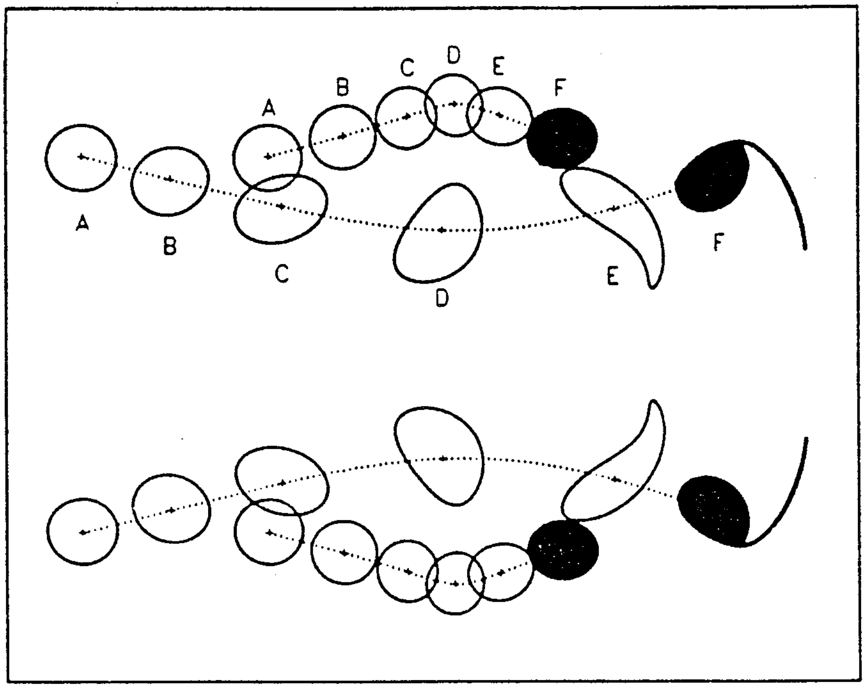


Figure 5.12c′. Euler calculations of vortex rings by Shariff (1987); $(a/R)_o = 0.18$, uniform vorticity in core.

Chapter 6

Concluding Remarks

6.1 Summary

A spectral method was developed which solves the incompressible Navier-Stokes equations in an unbounded domain. In particular, the flow of interest is that of vortex rings and their interactions. Because the the domain is infinite, and regions of large gradients vary both temporally and spatially, this a more difficult problem than those to which spectral methods are typically applied. The method is based on divergence-free basis functions. A set of functions was developed satisfying several constraints. In particular, each function is divergence-free. The solution is smooth everywhere and the functions combined represent the correct decay of vorticity at infinity. In addition, with the particular choice of functions, the linear terms in the matrix equations (resulting from numerically approximating the Navier-Stokes equations) are orthogonal (ie., diagonal) in two coordinate directions and banded in the third. This is an important property when solving a three-dimensional, time-dependent problem such that the computational time is manageable. Furthermore, it is important in order to alleviate stiffness arising in flows where there is a large range of length scales (ie., *high Re* flows).

Computations of axisymmetric rings demonstrated that the method works well, and that it is very accurate in computing rings over extended periods of time.

In contrast to most spectral methods (where typically the flow is assumed to be periodic) the flow was solved in an infinite domain. Several tests demonstrated that the far field was handled properly. The impulse, which is conserved in an infinite domain with confined vorticity, was shown to be conserved to very high accuracy. The drift at large times agreed with the analytic results of Rott (1988) to four significant digits. The arguments leading to the analytic drift are based on the behavior of the far field flow. The total integrated momentum was found to be 2/3 of the impulse (within several significant digits), as expected (Cantwell, 1986). All of these are primarily a check of the behavior of the velocity at large distances. Other diagnostics showed that the solution was accurate. The energy loss was shown to be due only to the true viscous dissipation. The impulse was typically conserved to several significant digits. Convergence studies of impulse showed that the error in impulse does indeed go to zero properly as the time step is decreased (ie., second order) and the resolution is increased (ie., faster than algebraic).

Computations were conducted over a wide range of Re. The algorithm developed is able to compute flows from thin rings at high Re to large time ($Re \to 0$). Two kinds of asymptotic studies were performed. The first is the limit of thin cores in a time frame where very little vorticity is permitted to diffuse across the axis of symmetry ($\Gamma \approx constant$), with Re_Γ as a parameter ($0 \leq Re < \infty$). A common theme was observed. A universal solution is approached after the transients associated with the starting condition have ended. For thin rings, it was shown that the propagation speed of rings with several different initial core sizes collapsed onto an envelope. That envelope corresponds to the time-dependent solution which would exist if the initial condition were given by an infinitely thin ring, and depends only on Re_Γ. As expected, as $Re_\Gamma \to 0$ the solution and envelope of propagation speed versus time is bounded. Furthermore, the vorticity distribution in the core is nearly Gaussian and simply spreads with time. A more surprising result was the apparent limit as $Re \to \infty$ (note that we can only extrapolate finite-Re_Γ results). For Re_Γ greater than around 200, curves of propagation speed versus time (normalized by circulation and ring radius) collapsed. This means that a universal time-dependent solution (ie., vorticity distribution) is approached for viscous solutions at sufficiently high Re.

The second asymptotic limit considered was $t \to \infty$ or $Re \to 0$. Flows were computed until $Re < 1$ for several different initial vorticity fields. For all cases studied, the solution returned to the drifting Stokes solution. The constant of proportionality relating $\overline{t}^{-3/2}$ and $\overline{U}$ was found to be independent of the initial condition. The large time solution, therefore, only remembers the impulse and kinematic viscosity.

In computing the propagation speed for thin rings (where $\Gamma \approx constant$), another useful observation was made. The next order term was found to the asymptotic theory of the propagation speed of viscous vortex rings (Saffman, 1970) for $Re_\Gamma = 0$. Furthermore, this correction is observed to to be an upper bound for all Re's. With this result, the range of validity of the asymptotic theory is quantified for a/R as large as 0.35.

The computations demonstrated the behavior of viscous vortex flows not predicted by inviscid models: shedding of vorticity into the wake of a propagating ring, and fusing of interacting rings. As postulated by experimentalists (Maxworthy, 1972), vorticity is shed behind a propagating ring. When considering the interaction of two rings, we find that they are able to leapfrog. Furthermore, the effect of diffusion is to smooth the vorticity peaks caused by the straining field, and for viscous flow the rings will merge into one.

6.2 Future Work

Future work will involve extending the method to three dimensions. Studies can then be made of mode selection associated with azimuthal instabilities, vortex collisions and reconnections, and noise generation due to collisions. The addition of a straining field could also be considered.

Appendix A
Translating and Expanding Coordinates

To compute rings over long time periods, it is very helpful to transform the equations to expanding and translating coordinates. The center of the coordinate system is translating with the centroid of the vorticity distribution (actually of the centroid of ωy). The rate of expansion of the coordinates is that of diffusion ($\sqrt{\nu t}$). The transformation is chosen for convenience. We have transformed to some new equations, slightly different from the Navier-Stokes equations. The two terms which are not invariant are the scalar quantity, (Φ), and the velocity in the $u \times \omega$ term. In the new coordinates, the dependent variables refer to the fixed coordinates while the independent variables refer to the translating frame. This is of no consequence since the results are easily transformed back to the original variables for plotting and interpretation. In this appendix, the new equations are derived.

In Cartesian tensor notation, the transformation is given by:

$$\begin{aligned} \widetilde{x}_i &\equiv \frac{(\overline{x}_i - \overline{X}_i(t))}{\overline{t}^{1/2}} & \widetilde{u}_i &\equiv \overline{u}_i\, \overline{t}^{1/2} \\ & & \widetilde{\omega}_i &\equiv \overline{\omega}_i\, \overline{t} \\ \widetilde{t} &\equiv \log \overline{t} & \widetilde{p} &\equiv \overline{p}\, \overline{t}. \end{aligned} \tag{A.1}$$

where $\overline{X}_i(t)$ is the centroid of the vorticity distribution, and its time-derivative is the ring speed:

$$\begin{aligned} \overline{U}_i(t) &\equiv \frac{d\overline{X}_i(t)}{d\overline{t}} \\ \widetilde{U}_i(t) &\equiv \overline{U}_i(t)\, \overline{t}^{1/2} \end{aligned} \tag{A.2}$$

In Cartesian tensor notation, the Navier-Stokes equations are given by:

$$\underbrace{\frac{\partial \overline{u}_i}{\partial \overline{t}}}_{A} + \overline{u}_j \underbrace{\frac{\partial \overline{u}_i}{\partial \overline{x}_j}}_{B} = -\frac{\partial \overline{p}}{\partial \overline{x}_i} + \underbrace{\frac{\partial^2 \overline{u}_i}{\partial \overline{x}_j \partial \overline{x}_j}}_{C} \qquad (A.3)$$

The transformations are applied to equation (A.3), one term at a time. Starting with the time-dependent term (A), substitute for the dependent variable, and apply the product rule:

$$\frac{\partial \overline{u}_i}{\partial \overline{t}} = \frac{\partial}{\partial \overline{t}} \left(\frac{\widetilde{u}_i}{\overline{t}^{1/2}} \right) = \frac{1}{\overline{t}^{1/2}} \underbrace{\frac{\partial \widetilde{u}_i}{\partial \overline{t}}}_{D} - \frac{1}{2\,\overline{t}^{3/2}} \widetilde{u}_i \qquad (A.4)$$

Next, the chain rule is applied to term D:

$$\frac{\partial \widetilde{u}_i}{\partial \overline{t}} = \frac{\partial \widetilde{u}_i}{\partial \widetilde{t}} \underbrace{\frac{\partial \widetilde{t}}{\partial \overline{t}}}_{E} + \frac{\partial \widetilde{u}_i}{\partial \widetilde{x}_j} \underbrace{\frac{\partial \widetilde{x}_j}{\partial \overline{t}}}_{F} \qquad (A.5)$$

Term E is given by:

$$\frac{\partial \widetilde{t}}{\partial \overline{t}} = \frac{\partial \log \overline{t}}{\partial \overline{t}} = \frac{1}{\overline{t}} \qquad (A.6)$$

Term F is given by:

$$\frac{\partial \widetilde{x}_j}{\partial \overline{t}} = \frac{\partial}{\partial \overline{t}} \left[\frac{(\overline{x}_i - \overline{X}_i(t))}{\overline{t}^{1/2}} \right] = -\frac{1}{2\,\overline{t}} \widetilde{x}_j - \frac{1}{\overline{t}^{1/2}} \frac{\partial \overline{X}_i(t)}{\partial \overline{t}} \qquad (A.7)$$

Combining the results from equation (A.5) and equation (A.6), term D becomes:

$$\frac{\partial \widetilde{u}_i}{\partial \overline{t}} = \frac{1}{\overline{t}} \frac{\partial \widetilde{u}_i}{\partial \widetilde{t}} - \frac{1}{2\,\overline{t}} \left(\widetilde{x}_j \frac{\partial \widetilde{u}_i}{\partial \widetilde{x}_j} \right) - \frac{1}{\overline{t}^{1/2}} \left(\frac{\partial \overline{X}_i(t)}{\partial \overline{t}} \frac{\partial \widetilde{u}_i}{\partial \widetilde{x}_j} \right) \qquad (A.8)$$

Using the chain rule, term B becomes:

$$\frac{\partial \overline{u}_i}{\partial \overline{x}_j} = \frac{1}{\overline{t}^{1/2}} \frac{\partial \widetilde{u}_i}{\partial \widetilde{x}_j} \qquad (A.9)$$

Multiplying equation (A.9) by $\overline{u}_j$ and transforming to $\widetilde{u}_j$, the convection term is:

$$\overline{u}_j \frac{\partial \overline{u}_i}{\partial \overline{x}_j} = \frac{1}{\overline{t}^{3/2}} \widetilde{u}_j \frac{\partial \widetilde{u}_i}{\partial \widetilde{x}_j} \qquad (A.10)$$

Following equation (A.9), the viscous term transforms to:

$$\frac{\partial^2 \overline{u}_i}{\partial \overline{x}_j \partial \overline{x}_j} = \frac{1}{\overline{t}^{3/2}} \frac{\partial^2 \widetilde{u}_i}{\partial \widetilde{x}_j \partial \widetilde{x}_j} \qquad (A.11)$$

Using the above results, the transformed governing equations are:

$$\begin{aligned}\frac{1}{\tilde{t}^{3/2}}\left\{\frac{\partial \tilde{u}_i}{\partial \tilde{t}} + \tilde{u}_j\frac{\partial \tilde{u}_i}{\partial \tilde{x}_j} = -\frac{\partial \tilde{p}}{\partial \tilde{x}_i} + \frac{\partial^2 \tilde{u}_i}{\partial \tilde{x}_j \partial \tilde{x}_j}\right.\\ \left. + \frac{1}{2}\tilde{u}_i + \frac{1}{2}\left(\tilde{x}_j\frac{\partial \tilde{u}_i}{\partial \tilde{x}_i}\right) + \tilde{t}^{1/2}\left(\frac{\partial \overline{X}_i(t)}{\partial \tilde{t}}\frac{\partial \tilde{u}_i}{\partial \tilde{x}_j}\right)\right\}\end{aligned} \qquad (A.12)$$

In vector notation, this becomes:

$$\begin{aligned}\frac{\partial \tilde{\mathrm{u}}}{\partial \tilde{t}} + (\tilde{\mathrm{u}}\cdot\tilde{\nabla})\,\tilde{\mathrm{u}} = -\,\tilde{\nabla}\tilde{p} + \tilde{\nabla}^2\tilde{\mathrm{u}}\\ + \frac{1}{2}\tilde{\mathrm{u}} + \frac{1}{2}\underbrace{(\tilde{\mathrm{x}}\cdot\tilde{\nabla})}_{G}\,\tilde{\mathrm{u}} + \underbrace{(\tilde{\mathrm{U}}\cdot\tilde{\nabla})\tilde{\mathrm{u}}}_{H}\end{aligned} \qquad (A.13)$$

Next, vector identities are applied to term G and H. Term G is given by:

$$(\tilde{\mathrm{x}}\cdot\tilde{\nabla})\,\tilde{\mathrm{u}} = \tilde{\nabla}(\tilde{\mathrm{x}}\cdot\tilde{\mathrm{u}}) - \underbrace{(\tilde{\mathrm{u}}\cdot\tilde{\nabla})\,\tilde{\mathrm{x}}}_{\tilde{\mathrm{u}}} - \tilde{\mathrm{x}}\times\underbrace{(\tilde{\nabla}\times\tilde{\mathrm{u}})}_{\tilde{\omega}} - \tilde{\mathrm{u}}\times\tilde{\mathrm{u}}\times\underbrace{(\tilde{\nabla}\times\tilde{\mathrm{x}})}_{0} \qquad (A.14)$$

Applying the same vector identity to H gives:

$$(\tilde{\mathrm{U}}\cdot\tilde{\nabla})\,\tilde{\mathrm{u}} = \tilde{\nabla}(\tilde{\mathrm{U}}\cdot\tilde{\mathrm{u}}) - \underbrace{(\tilde{\mathrm{u}}\cdot\tilde{\nabla})\,\tilde{\mathrm{U}}}_{0} - \tilde{\mathrm{U}}\times\underbrace{(\tilde{\nabla}\times\tilde{\mathrm{u}})}_{\tilde{\omega}} - \tilde{\mathrm{u}}\times\tilde{\mathrm{u}}\times\underbrace{(\tilde{\nabla}\times\tilde{\mathrm{U}})}_{0} \qquad (A.15)$$

Substituting equation (A.14) and (A.15) into equation (A.13) gives:

$$\begin{aligned}\frac{\partial \tilde{\mathrm{u}}}{\partial \tilde{t}} + (\tilde{\mathrm{u}}\cdot\tilde{\nabla})\,\tilde{\mathrm{u}} = -\,\tilde{\nabla}\tilde{p} + \tilde{\nabla}^2\tilde{\mathrm{u}}\\ + \frac{1}{2}\tilde{\mathrm{u}} + \frac{1}{2}[\tilde{\nabla}(\tilde{\mathrm{x}}\cdot\tilde{\mathrm{u}}) - \tilde{\mathrm{u}} - \tilde{\mathrm{x}}\times\tilde{\omega}] + [\tilde{\nabla}(\tilde{\mathrm{U}}\cdot\tilde{\mathrm{u}}) - \tilde{\mathrm{U}}\times\tilde{\omega}]\end{aligned} \qquad (A.16)$$

Applying the identity, $(\tilde{\mathrm{u}}\cdot\tilde{\nabla})\,\tilde{\mathrm{u}} = \tilde{\nabla}(\tilde{\mathrm{u}}\cdot\tilde{\mathrm{u}})/2 - \tilde{\mathrm{u}}\times\tilde{\omega}]$, to equation (A.16) gives the equations in expanding and translating coordinates which are solved numerically:

$$\frac{\partial \tilde{\mathrm{u}}}{\partial \tilde{t}} + \tilde{\nabla}\,\tilde{\Phi} - \tilde{\nabla}^2\,\tilde{\mathrm{u}} = \tilde{\mathrm{u}}_{mod}\times\tilde{\omega} \qquad (A.17)$$

where $\tilde{\mathrm{u}}_{mod}$ and $\tilde{\Phi}$ are given by:

$$\begin{aligned}\tilde{\mathrm{u}}_{mod} &= \tilde{\mathrm{u}} - \frac{1}{2}\tilde{\mathrm{x}} - \tilde{\mathrm{U}}\\ \tilde{\Phi} &= \tilde{p} + \tilde{\mathrm{u}}\cdot\tilde{\mathrm{u}}/2 - \frac{1}{2}\tilde{\mathrm{x}}\cdot\tilde{\mathrm{u}} - \tilde{\mathrm{U}}\cdot\tilde{\mathrm{u}}.\end{aligned} \qquad (A.18)$$

Appendix B

Divergence-Free Basis Functions

Vector spherical harmonics (VSH) comprise three vector functions which are in turn functions of scalar spherical harmonics (see Appendix C for their definitions and properties). These functions span the space of any vector function on the surface of a sphere, i.e. they form a complete set. To represent an arbitrary function in a volume, each of the three families of functions, $\mathbf{X}_{\ell m}(\theta,\phi)$, $\mathbf{V}_{\ell m}(\theta,\phi)$, and $\mathbf{W}_{\ell m}(\theta,\phi)$, is multiplied by scalar radial functions, $F_{1_{\ell m}}(r,t)$, $F_{2_{\ell m}}(r,t)$, and $F_{3_{\ell m}}(r,t)$.

$$\mathbf{u} = \sum_{\ell m} F_{1_{\ell m}}(r,t)\mathbf{X}_{\ell m}(\theta,\phi) + F_{2_{\ell m}}(r,t)\mathbf{V}_{\ell m}(\theta,\phi) + F_{3_{\ell m}}(r,t)\mathbf{W}_{\ell m}(\theta,\phi) \qquad (B.1)$$

To find the divergence-free set which is complete, substitute equation (B.1) into the continuity equation,

$$\nabla\cdot\mathbf{u} = \sum_{\ell m}\left\{\nabla\cdot(F_{1_{\ell m}}\mathbf{X}_{\ell m}) + \nabla\cdot(F_{2_{\ell m}}\mathbf{V}_{\ell m}) + \nabla\cdot(F_{3_{\ell m}}\mathbf{W}_{\ell m})\right\} = 0 \qquad (B.2)$$

From the properties of VSH (Appendix C), $\nabla\cdot(F(r)\mathbf{X}_{\ell m}) = 0$ for any function F. On the other hand, both $\mathbf{V}_{\ell m}$ and $\mathbf{W}_{\ell m}$ contribute to $\nabla\cdot\mathbf{u}$:

$$0 = k_2\, Y_\ell^m \left\{-\left(\frac{dF_{2_{\ell m}}}{dr} + \frac{(\ell+2)}{r}F_{2_{\ell m}}\right) + k_1\left(\frac{dF_{3_{\ell m}}}{dr} - \frac{(\ell-1)}{r}F_{3_{\ell m}}\right)\right\}. \qquad (B.3)$$

where $k_1 \equiv (\frac{\ell}{2\ell+1})^{1/2}$, and $k_2 \equiv (\frac{\ell+1}{2\ell+1})^{1/2}$. Divide equation (B.3) by $k_2\, Y_\ell^m$ (recall that the Y_ℓ^m*'s* are orthonormal) and factor both terms,

$$-\frac{1}{r^{\ell+2}}\,\frac{d}{dr}\,(r^{\ell+1}F_{2_{\ell m}}) + k_3\,\frac{1}{r^{-\ell+1}}\,\frac{d}{dr}\,(r^{-\ell+1}F_{3_{\ell m}}) = 0 \qquad (B.4)$$

where $k_3 \equiv k_1/k_2 \equiv (\frac{\ell}{\ell+1})^{1/2}$. We wish to show that because of equation (B.4) the functions $F_{2_{\ell m}}$ and $F_{3_{\ell m}}$ can be uniquely expressed in terms of one function, $F^+_{\ell m}$, such that $F^+_{\ell m}(r)\,\mathbf{X}_{\ell m}$ is a vector potential:

$$\nabla\times(F^+_{\ell m}(r)\mathbf{X}_{\ell m}) = F_{2_{\ell m}}(F)\mathbf{V}_{\ell m}(\theta,\phi) + F_{3_{\ell m}}(F)\mathbf{W}_{\ell m}(\theta,\phi). \qquad (B.5)$$

For the functional dependencies, $F_{2_{\ell m}}(F^+_{\ell m})$ and $F_{3_{\ell m}}(F^+_{\ell m})$, see Appendix C. Factoring these,

$$F_{2_{\ell m}} = k_1\left[\frac{dF^+_{\ell m}}{dr} - \frac{\ell}{r}F^+_{\ell m}\right] = \frac{k_1}{r^{-\ell}}\,\frac{d}{dr}\,(r^{-\ell}F^+_{\ell m}(r,t)) \qquad (B.6)$$

$$F_{3_{\ell m}} = k_2\left[\frac{dF^+_{\ell m}}{dr} + \frac{(\ell+1)}{r}F^+_{\ell m}\right] = \frac{k_2}{r^{\ell+1}}\,\frac{d}{dr}\,(r^{\ell+1}F^+_{\ell m}(r,t)). \qquad (B.7)$$

Substitute equation (B.6) into (B.4), thus eliminating $F_{2_{\ell m}}$,

$$\frac{k_1}{k_3}\left[\frac{d^2F^+_{\ell m}}{dr^2} + \frac{2}{r}\frac{dF^+_{\ell m}}{dr} - \frac{\ell(\ell+1)}{r}F^+_{\ell m}\right] = \frac{1}{r^{-\ell+1}}\,\frac{d}{dr}\,(r^{-\ell+1}F_{3_{\ell m}}) \qquad (B.8)$$

Note that $k_1/k_3 = k_2$. Factor the left hand side of equation (B.8),

$$k_2\frac{1}{r^{-\ell+1}}\,\frac{d}{dr}\left[r^{-2\ell}\frac{d}{dr}(r^{\ell+1}F^+_{\ell m})\right] = \frac{1}{r^{-\ell+1}}\,\frac{d}{dr}(r^{-\ell+1}F_{3_{\ell m}}) \qquad (B.9)$$

And finally, integrate equation (B.9),

$$k_2\,r^{-\ell-1}\,\frac{d}{dr}\,(r^{\ell+1}F^+_{\ell m}) = F_{3_{\ell m}}, \qquad (B.10)$$

which is in agreement with equation (B.7). Therefore, the complete set of three-dimensional divergence-free basis functions is

$$\mathbf{u} = F^-_{\ell m}(r,t)\,\mathbf{X}_{\ell m}(\theta,\phi) + \nabla\times(F^+_{\ell m}(r,t)\mathbf{X}_{\ell m}(\theta,\phi)) \qquad (B.11)$$

with two scalar radial functions, $F^-_{\ell m}$ and $F^+_{\ell m}$. Using the continuity equation, we have reduced the number of radial functions (and time dependent coefficients) from three to two.

Note that the $-$ modes can also be interpreted as deriving from a vector potential, since

$$\nabla\times(F(r)\mathbf{V}_{\ell m}) = i\left(\frac{\ell}{2\ell+1}\right)^{1/2}\left[\frac{dF}{dr} + \frac{\ell+2}{r}F\right]\mathbf{X}_{\ell m} \qquad (B.12)$$

Limiting forms at large radius:

$$\lim_{r\to\infty}(1-\xi) \approx \frac{r_1^2}{r^2} \qquad \lim_{r\to\infty}\xi = 1 \tag{C.5}$$

Limiting forms at small radius:

$$\lim_{r\to 0}(1-\xi) = 1 \qquad \lim_{r\to 0}\xi \approx \frac{r^2}{r_1^2} \tag{C.6}$$

Vector Spherical Harmonics

Vector spherical harmonics (VSH), $\mathbf{X}_{\ell m}$, $\mathbf{V}_{\ell m}$, and $\mathbf{W}_{\ell m}$, form a complete set of vector functions on the surface of sphere. They are functions of scalar spherical harmonics which, in turn, are functions of Legendre polynomials and complex exponentials. With the aide of relatively simple expressions for their Laplacian and divergence, the divergence-free subset is extracted from the complete set (see Appendix B). Furthermore, since the VSH functions are orthonormal, the mass matrix is completely decoupled in θ and ϕ. Because the Laplacian of a vector spherical harmonic function does not modify the angular dependence (see eqn. C.13), the viscous matrix is also completely decoupled in θ and ϕ. Many of the relevant properties of these functions are given below, and a more complete list is found in the paper by Hill (1953).

Orthogonality properties of VSH ($\mathbf{C}$ and $\mathbf{D}$ are $\mathbf{X}_{\ell m}$, $\mathbf{V}_{\ell m}$, or $\mathbf{W}_{\ell m}$):

$$\int_0^{2\pi}\int_0^{\pi} \mathbf{C}_{\ell m}\cdot(\mathbf{D}_{\ell' m'})^{*}\,\sin\theta\,d\theta\,d\phi = \delta_{\mathrm{CD}}\,\delta_{\ell\ell'}\,\delta_{mm'} \tag{C.7}$$

Definitions of VSH in terms of the scalar spherical harmonic, (Y_ℓ^m):

$$\begin{aligned}\mathbf{V}_{\ell m} \equiv \hat{r}\left\{-\left(\frac{\ell+1}{2\ell+1}\right)^{1/2} Y_\ell^m\right\} &+ \hat{\theta}\left\{\frac{1}{[(\ell+1)(2\ell+1)]^{1/2}}\frac{\partial Y_\ell^m}{\partial\theta}\right\}\\ &+ \hat{\phi}\left\{\frac{imY_\ell^m}{[(\ell+1)(2\ell+1)]^{1/2}\sin\theta}\right\}\end{aligned} \tag{C.8}$$

$$\mathbf{X}_{\ell m} \equiv \hat{\theta}\left\{\frac{-mY_\ell^m}{[\ell(\ell+1)]^{1/2}\sin\theta}\right\} + \hat{\phi}\left\{\frac{-i}{[\ell(\ell+1)]^{1/2}}\frac{\partial Y_\ell^m}{\partial\theta}\right\} \tag{C.9}$$

$$\begin{aligned}\mathbf{W}_{\ell m} \equiv & \hat{r}\left\{\left(\frac{\ell}{2\ell+1}\right)^{1/2} Y_\ell^m\right\} + \hat{\theta}\left\{\frac{1}{[\ell(2\ell+1)]^{1/2}}\frac{\partial Y_\ell^m}{\partial\theta}\right\} \\ & + \hat{\phi}\left\{\frac{imY_\ell^m}{[\ell(2\ell+1)]^{1/2}\sin\theta}\right\}\end{aligned} \tag{C.10}$$

Divergence and curl of $\mathbf{X}_{\ell m}$:

$$\nabla\cdot[F(r)\mathbf{X}_{\ell m}] = 0 \tag{C.11}$$

$$\begin{aligned}\nabla\times[F(r)\mathbf{X}_{\ell m}] = & i\left(\frac{\ell}{2\ell+1}\right)^{1/2}\left[\frac{d\,F}{dr} - \frac{\ell}{r}F\right]\mathbf{V}_{\ell m} \\ & + i\left(\frac{\ell+1}{2\ell+1}\right)^{1/2}\left[\frac{d\,F}{dr} + \frac{\ell+1}{r}F\right]\mathbf{W}_{\ell m}\end{aligned} \tag{C.12}$$

Laplacian of the product of VSH with an arbitrary radial function ($F(r)$):

$$\begin{aligned}\nabla^2[F(r)\mathbf{V}_{\ell m}] &= L_{\ell+1}(F)\mathbf{V}_{\ell m} \\ \nabla^2[F(r)\mathbf{X}_{\ell m}] &= L_\ell(F)\mathbf{X}_{\ell m} \\ \nabla^2[F(r)\mathbf{W}_{\ell m}] &= L_{\ell-1}(F)\mathbf{W}_{\ell m} \\ L_\ell &\equiv \frac{\partial^2}{\partial^2 r} + \frac{2}{r}\frac{\partial}{\partial r} - \frac{\ell(\ell+1)}{r^2}\end{aligned} \tag{C.13}$$

Scalar spherical harmonics:

$$Y_\ell^m(\theta,\phi) = \Theta_\ell^m(\theta)\cdot\frac{e^{im\phi}}{(2\pi)^{1/2}} \tag{C.14}$$

$$\Theta_\ell^m(\theta) = \left[\frac{2\ell+1}{2}\cdot\frac{(\ell-m)!}{(\ell+m)!}\right]^{1/2} P_\ell^m(\mu) \tag{C.15}$$

Hill's definition (1953) for the Legendre polynomials, $P_\ell^m(\mu)$, differs slightly from that used here (i.e. by a factor of $(-1)^m$).

Associated Legendre Polynomials

Notation: $P_\ell^m(\mu) = Q_{\ell m}(\theta)$

Weight function: $w(x) = 1$

Domain: $-1 \leq \mu \leq 1, \qquad \mu = \cos\theta$

Orthogonality:

$$\int_{-1}^{1} P_\ell^m \, P_{\ell'}^m \, d\mu = \frac{2\,(\ell+m)!}{(2\ell+1)(\ell-m)!}\delta_{\ell\ell'} \qquad (C.16)$$

Rodrigues formula:

$$P_\ell^m(\mu) = \frac{(-1)^m}{2^\ell \ell!}\,(1-\mu^2)^{m/2}\,\frac{d^{\ell+m}}{d\mu^{\ell+m}}(\mu^2-1)^\ell \qquad (C.17)$$

First few polynomials:

	$m=0$	$m=1$	$m=2$
$\ell=1$	μ	$-(1-\mu^2)^{1/2}$	not defined
$\ell=2$	$\frac{1}{2}(3\mu^2-1)$	$-3\mu(1-\mu^2)^{1/2}$	$3(1-\mu^2)$
$\ell=3$	$\frac{1}{2}\mu(5\mu^2-3)$	$-\frac{3}{2}(5\mu^2-1)(1-\mu^2)^{1/2}$	$15\mu(1-\mu^2)$

$(C.18)$

Recurrence relations:

$$(m-\ell-1)P_{\ell+1}^m + (2\ell+1)\mu P_\ell^m - (m+\ell)P_{\ell-1}^m = 0 \qquad (C.19)$$

Derivative relations:

$$\frac{dQ_{\ell 0}}{d\theta} = P_\ell^1(\mu); \qquad \ell \geq 1 \qquad (C.20)$$

$$\frac{dQ_{\ell m}}{d\theta} = \frac{1}{2}\left\{P_\ell^{m+1} - (m+\ell)(\ell-m+1)P_\ell^{m-1}\right\}; \qquad m \geq 1 \qquad (C.21)$$

Shifted Jacobi Polynomials

The shifted Jacobi polynomials, G_n^ℓ, are similar to those defined in Abramowitz and Stegun (A&S), (1972, eqn. 22.2.2) with two important differences. First, in the notation of A&S, $p = 2\ell - 1$ and $q = \ell - 1/2$. Second, the polynomials are orthonormal with respect to the weight function, $w(\xi)$. The properties are summarized below.

Notation: $G_{n'}^\ell$

Weight function: $w(\xi) = (1-\xi)^{\ell-\frac{1}{2}}\ \xi^{\ell-\frac{3}{2}}$

Domain: $0 \leq \xi \leq 1$

Orthogonality:

$$\int_0^1 G_n^\ell(\xi)\, G_{n'}^\ell(\xi)\, w(\xi) d\xi = \delta_{nn'} \tag{C.22}$$

Rodrigues formula:

$$G_n^\ell(\xi) = \frac{\left(\frac{2(n+2\ell-2)!}{n!}\right)^{1/2}}{\left(\frac{2n+2\ell-3}{2}\right)!}\ (\xi-1)^{-\ell+\frac{1}{2}}\xi^{-\ell+\frac{3}{2}}\ \frac{d^n}{d\xi^n}\left((\xi-1)^{n+\ell-\frac{1}{2}}\xi^{n+\ell-\frac{3}{2}}\right) \tag{C.23}$$

First few polynomials:

$$G_0^\ell = \frac{(2(2\ell-2)!)^{1/2}}{\left(\frac{2\ell-3}{2}\right)!}$$

$$\frac{G_1^\ell}{G_0^\ell} = \frac{1}{(2\ell-1)^{1/2}}\,(4\ell\xi - 2\ell + 1) \tag{C.24}$$

$$\frac{G_2^\ell}{G_0^\ell} = \left(\frac{\ell}{2\ell-1}\right)^{1/2}\ [(8\ell+8)\xi^2 - (8\ell+4)\xi + 2\ell - 1]$$

Recurrence relation:

$$\begin{aligned}\xi G_n^\ell(\xi) &= \frac{\left[(n+1)(n+2\ell-1)\right]^{1/2}}{4(n+\ell)}\, G_{n+1}^\ell(\xi) \\ &+ \frac{(2n^2+4\ell n-2n+2\ell^2-3\ell+1)}{4(n+\ell-1)(n+\ell)}\, G_n^\ell(\xi) \\ &+ \frac{\left[n(n+2\ell-2)\right]^{1/2}}{4(n+\ell-1)}\, G_{n-1}^\ell(\xi)\end{aligned} \qquad (C.25)$$

Derivative relation:

$$\begin{aligned}\xi(\xi-1)\frac{dG_n^\ell(\xi)}{d\xi} &= \frac{n\left[(n+1)(n+2\ell-1)\right]^{1/2}}{4(n+\ell)}\, G_{n+1}^\ell(\xi) \\ &- \frac{n(n+2\ell-1)}{4(n+\ell-1)(n+\ell)}\, G_n^\ell(\xi) \\ &- \frac{[n(n+2\ell-2)]^{1/2}(n+2\ell-1)}{4(n+\ell-1)}\, G_{n-1}^\ell(\xi)\end{aligned} \qquad (C.26)$$

Appendix C

Properties of Algebraic Mapping and Special Functions

Algebraic Mapping

The radial direction, $0 \leq r < \infty$, is mapped to the domain [0,1] in the new variable ξ. The mapping, and its properties are shown below.

Definition:

$$\xi = \frac{r^2}{r^2 + r_1^2} \qquad r^2 = \frac{r_1^2\,\xi}{1-\xi} \tag{C.1}$$

Derivatives:

$$\frac{d\xi}{dr} = \frac{2}{r_1}\xi^{1/2}(1-\xi)^{3/2} = 2\,r_1^2\frac{r}{(r^2+r_1^2)^2} \tag{C.2}$$

$$\frac{d^2\xi}{dr^2} = \frac{-2}{r_1^2}(4\xi-1)(1-\xi)^2 = \frac{-2\,r_1^2\,(3r^2 - r_1^2)}{(r^2+r_1^2)^3} \tag{C.3}$$

$$r^2\,dr = \frac{r_1^3}{2}\xi^{1/2}(1-\xi)^{-5/2}\,d\xi \tag{C.4}$$

Appendix D

Coordinate Singularities

In spherical polar coordinates, there are coordinate singularities near the origin and on the polar axis, $\theta = 0$ and π. Therefore, an arbitrary expansion in these coordinates can have discontinuous derivatives at these singular points giving, for example, infinite vorticities or Laplacians. Our expansions involve vector spherical harmonics multiplying arbitrary radial functions. The vector spherical harmonics ensure that the function is smooth along the polar axis. We are left with the task of finding appropriate radial functions, and therefore knowing the allowable combinations of $\sin\phi$, $\cos\phi$, $\cos\theta$, $\sin\theta$ and r such that the basis functions are smooth near the origin and are also complete.

Spalart (1988) derived the functional dependence for the general class of the radial functions in polar and spherical polar coordinates, for scalar and vector fields. This was accomplished by writing the solution at the origin in a Taylor series expansion, applying a Laplacian operator repeatedly, and requiring that all of the derivatives remain bounded. From this, a set of constraints on the radial functional dependence is found. He then shows that this is not only a necessary condition but also a sufficient condition by counting the number of degrees of freedom imposed by this dependence and the number of degrees of freedom available in the Taylor expansion in x, y, z coordinates. By showing that these are equal, it follows that this form of the radial expansion in the neighborhood of the origin, is both sufficient and necessary for completeness. This appendix describes this procedure for scalar and vector fields in spherical polar coordinates.

For a scalar field, $g(r,\theta,\phi)$, we have an expansion of the form

$$g(r,\theta,\phi) = \sum_{\ell,m} h_{\ell m}(r)\, Y_{\ell m}(\theta,\phi) \qquad (D.1)$$

We wish to find the functional form of $h_{\ell m}(r)$ such that $g(r,\theta,\phi)$ is smooth near the origin. Applying the Laplacian operator n times gives

$$(\nabla^2)^n g = \sum_{\ell,m} (\nabla^2)^n (h_{\ell m} Y_{\ell m}) = \sum_{\ell,m} ((L_\ell)^n h_{\ell m}) Y_\ell^m \qquad (D.2)$$

where L_ℓ is given by equation (C.13). For all the derivatives of g to be bounded at the origin, $((L_\ell)^n h_{\ell m})$ must be bounded for all n. Now, write a Taylor series expansion of $h_{\ell m}$ about the origin:

$$h_{\ell m} = \sum_{p=o}^{\infty} a_p\, r^p. \qquad (D.3)$$

Substitute equation (D.3) into (D.2) giving

$$\begin{aligned}(\nabla^2)^n g &= \sum_{\ell,m} (L_\ell)^n (\sum_{p=0}^{\infty} a_p r^p)\, Y_\ell^m \\ &= \sum_{\ell,m} \sum_{p=0}^{\infty} a_p ((L_\ell)^n\, r^p)\, Y_\ell^m\end{aligned} \qquad (D.4)$$

Again, for all derivatives of g to be bounded, $(L_\ell)^n r^p$ must be bounded. From equation (C.13) it follows that

$$L_\ell(r^q) = (q-\ell)(q+\ell+1)\, r^{q-2}. \qquad (D.5)$$

From (D.5), it is clear that for $L_\ell(r^q)$ to remain bounded, q must equal ℓ before $q-2<0$. Furthermore, repeated applications of the Laplacian (eqn. D.4) gives

$$(L_\ell)^n\, r^p \sim r^{p-2n}. \qquad (D.6)$$

and $q = p-2n$. Since $q=\ell$, then $p-2n=\ell$ before $p-2n<0$, or $p=\ell+2n$. Combining this with equation (D.3) gives

$$\begin{aligned}h_{\ell m} &\sim \sum_{n=0}^{\infty} a_n r^{\ell+2n} \\ &\sim r^\ell \sum_{n=0}^{\infty} a_n (r^2)^n \\ &\sim r^\ell f_{\ell m}(r^2)\end{aligned} \qquad (D.7)$$

where $f_{\ell m}$ is a smooth function. This constrains the form of the expansion. The next step is to show that these constraints are sufficient, or that the the number of degrees of freedom in the spherical harmonic expansion is equal to the number of constraints imposed by equation (D.7).

In a Taylor series expansion in x, y, z, we have

$$g(x,y,z) = \sum_{i,j,k=0}^{\infty} a_{ijk} x^i y^j z^k \tag{D.8}$$

where $x = r\sin\theta\cos\phi$, $y = r\sin\theta\sin\phi$, and $z = r\cos\theta$. Writing equation (D.8) in spherical polar coordinates gives

$$\begin{aligned} g(x,y,x) &= \sum_{i,j,k=0}^{\infty} a_{ijk}\, r^{i+j+k}\, s_{ijk}(\theta,\phi) \\ &= \sum_{p=0}^{\infty} r^p \left[\sum_{i+j+k=p} a_{ijk}\, s_{ijk} \right] \end{aligned} \tag{D.9}$$

where the functions s_{ijk} contain the angular dependence. We now ask how many combinations of (i,j,k) there are for a given p, such that $i+j+k=p$, where $i \geq 0$, $j \geq 0$, and $k \geq 0$.

First, we eliminate k by writing

$$k = p - i - j \tag{D.10}$$

or

$$p \geq i + j, \qquad \begin{aligned} i \geq 0 \\ j \geq 0 \end{aligned} \tag{D.11}$$

By expressing equation (D.11) as

$$j \leq p - i, \qquad i \geq 0 \tag{D.12}$$

we can count $p - i + 1$ values of j that are compatible with a given i. Therefore, the total number of degrees of freedom is

$$\begin{aligned} \sum_{i=0}^{p}(p-i+1) = \sum_{i=0}((p+1)-i) &= \sum_{i=0}^{p}(p+1) - \sum_{i=0}^{p} i \\ &= (p+1)^2 - \frac{p(p+1)}{2} \\ &= \frac{(p+1)(p+2)}{2} \end{aligned} \tag{D.13}$$

Next, we count the degrees of freedom in the expansion given by (D.1). Combining this with the radial dependence given by (D.7), we have

$$g \sim \sum_{\ell,m,n} r^{\ell+2n} Y_{\ell m}(\theta,\phi) \tag{D.14}$$

where $n \geq 0$, $|m| \leq \ell$, $\ell \geq 0$, and $p = \ell + 2n$.

We begin by counting the number of values of m, giving $2\ell+1$. Furthermore, write $\ell = p - 2n$, $n \geq 0$, $\ell \geq 0$. Now, we eliminate ℓ giving $2(p-2n)+1$ values of m, where $n \geq 0$. Note that the upper bound on n is given by $n \leq \left[\frac{p}{2}\right]$ where $\left[\frac{p}{2}\right]$ indicates the integer value. Therefore, the total number of degrees of freedom is

$$\sum_{n=0}^{[\frac{p}{2}]} (2(p-2n)+1) = \left(\left[\frac{p}{2}\right]+1\right)\left(2p - 2\left[\frac{p}{2}\right]+1\right). \tag{D.15}$$

Considering the case of even and odd p separately, the equation (D.15) simplifies to $\frac{(p+1)(p+2)}{2}$, which is equal to the number of degrees of freedom (eqn. D.13).

Next, we generalize this to 3D vectors in spherical polar coordinates. The expansion is in terms of the radial functions multiplying the vector spherical harmonic functions, $\mathbf{X}_{\ell m}$, $\mathbf{V}_{\ell m}$, and $\mathbf{W}_{\ell m}$. Applying the same procedure as that above, we take the Laplacian of the expansion function and require that all of the derivatives are bounded. From equation (C.13) for the Laplacian of each of the VSH functions, we find $p = \ell + 2n$ for $\mathbf{X}_{\ell m}$, $p = \ell + 1 + 2n$ for $\mathbf{V}_{\ell m}$, and $p = \ell - 1 + 2n$ for $\mathbf{W}_{\ell m}$ and the corresponding total number of degrees of freedom is $(3p^2 + 9p + 8)/2$. In comparison with the number of degrees of freedom for the Taylor expansion in x, y, z (ie., $3(p+1)(p+2)/2$), we find that we have 1 too many degrees of freedom. This apparent discrepancy can be accounted for by the special cases of $\mathbf{X}_{0,0}$ and $\mathbf{W}_{0,0}$ which are equal to 0. If p is even it multiplies $\mathbf{X}_{0,0}$ and if it is odd it multiplies $\mathbf{W}_{0,0}$, making all of the degrees of freedom accounted for.

In summary, in order for an the expansion given by equation (2.3.1) to be C_∞ (infinitely differentiable), the radial functions multiplying each of the vector spherical harmonics, $\mathbf{X}_{\ell m}$, $\mathbf{V}_{\ell m}$, $\mathbf{W}_{\ell m}$, must (near the origin) be of the form

$$\begin{gathered} r^\ell f_{Xlm}(r^2), \\ r^{\ell+1} f_{Vlm}(r^2), \\ r^{\ell-1} f_{Wlm}(r^2), \end{gathered} \tag{D.16}$$

respectively, where the $f's$ are smooth.

Appendix E
Gauss Quadrature

Gauss quadrature is an efficient and accurate method to evaluate certain types of integrals. In particular, when an integrand is composed of the product of a polynomial and a weight function of a Sturm Liouville polynomials (Legendre, Jacobi, Chebychev, etc.) and the limits correspond to the same family as the weight function, then such an integral can be integrated exactly with a finite number of points. It happens that for the problem which we are solving, integrals of this form arise in transforms from real space to wave space and those defining global properties of the flow (i.e. impulse, momentum, etc.). Actually, this property of our integral expressions was a constraint which determined the basis functions.

Gauss quadrature frequently refers to integrals using Legendre polynomials of as interpolating polynomials. Here we are using this term in a more general sense to mean any interpolating polynomials which are solutions of a Sturm-Liouville problem. This method is defined not only by weights multiplying the function at each grid point, but also the particular choice of the grid points. By using these extra degrees of freedom, a polynomial, P_{2N-1}, of order $2N-1$ is integrated exactly using only N collocation points. This appendix develops this numerical method and presents a scheme for determining the collocation points and weights for a particular family of interpolating polynomials (Golub and Welsch, 1969).

Theory

A definite integral, I, is evaluated numerically as a discrete sum

$$I = \int_a^b f(x)\,w(x)\,dx = \sum_{n=1}^{N} f(x_n)\,A_n \qquad (E.1)$$

The particular quadrature rule is determined by the choice of the points, x_n (N degrees of freedom), and the weights, A_n (N degrees of freedom), totaling $2N$ degrees of freedom.

One means of determining the weights, A_n, is to approximate the function, $f(x)$, as a polynomial of degree N,

$$P_N(x) \approx \sum_{n=1}^{N} \frac{\alpha(x)}{(x-x_n)\,\alpha'(x_n)} f(x_n) \qquad (E.2)$$

where $\alpha(x) = (x-x_1)(x-x_2)(x-x_3)...(x-x_N)$ is a completely factored N^{th} degree polynomial. Therefore, $f(x)$ is exact at the points, x_n. Furthermore, if $f(x)$ is a polynomial of order $N-1$, then the approximation is exact everywhere (the space of polynomials of degree $0, 1, \ldots, N-1$ is of dimension N).

Substituting equation (E.2) into (E.1), and exchanging the sum and integral operators gives

$$\int_a^b f(x)\,w(x)\,dx \approx \sum_{n=1}^{N} f(x_n) \int_a^b \frac{\alpha(x)}{(x-x_n)\,\alpha'(x_n)}\,w(x)\,dx \qquad (E.3)$$

With equally spaced points, x_n, we get gives various Newton-Cotes formulas, among which is the commonly used Simpson's rule ($w(x) = 1$).

It was pointed out by Gauss, that the values of x_n represented unused parameters, giving the potential of a more accurate scheme. Indeed this is true when x_n is chosen such that $\alpha(x)$ is the n^{th}-degree polynomial which is orthogonal to all lower degree polynomials over $[a, b]$ with respect to the weighting factor $w(x)$. With this, a function, $f(x) = P_{2N-1}$ is integrated exactly with N collocation points. The most common sets of orthogonal polynomials are those resulting from solutions of a Sturm-Liouville problem. Therefore, if we are able to express our integrals in the form of equation (E.1) where the weight function, $w(x)$, and interval, $[a, b]$, correspond to a family of polynomials which are solution of a Sturm-Liouville equations, and if the remaining part of the integrand, $f(x)$, is a polynomial, then we can integrate the function exactly with a finite number of collocation points.

Finding the Collocation Points and Weights

The mechanics of numerically computing the collocation points, x_i, and the weights in the quadrature formula, A_i, are described below. For a more complete mathematical treatment, see Golub and Welsh (ref.). We start with the recurrence relation for the particular special functions of interest, cast this as an eigenvalue problem, normalize the matrix, and solve for the eigenvalues and the eigenvectors. The collocation points, x_n, are the eigenvalues (which are also the n zeros of the n^{th} degree polynomial) and the weights are found from the eigenvectors. This procedure tends to be less sensitive to roundoff errors than that of finding the roots of the n^{th} polynomial, and then the weights by a linear system.

For any solution of a Sturm-Liouville problem, a three term recurrence relation can be found, showing the relation between the polynomials of different orders.

$$p_{j+1}(x) = (a_{j+1}\, x + b_{j+1})\, p_j(x) - c_{j+1}\, p_{j-1}(x) \qquad (E.4)$$

for $j = 0, 1, ..., N-1$, $p_{-1}(x) = 0$, and $p_0(x) = 1$. The coefficients, a_{j+1}, b_{j+1}, and c_{j+1} are tabulated in several books, for example Abramowitz and Stegun (ref.). Rewriting equation (E.4),

$$\begin{aligned} x\,\mathbf{p}(x) &= \mathbf{T}\,\mathbf{p}(x) + \frac{1}{a_N} p_N(x)\mathbf{e}_N \\ \mathbf{e}_N &= [0, 0, ..., 0, 1]^T \\ \mathbf{p}(x) &= [p_0(x), p_1(x), p_2(x), ..., p_{N-1}(x)]^T \\ \mathbf{T} &= \begin{pmatrix} -\frac{b_1}{a_1} & \frac{1}{a_1} & & & 0 \\ \frac{c_2}{a_2} & -\frac{b_2}{a_2} & \frac{1}{a_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{c_{N-1}}{a_{N-1}} & -\frac{b_{N-1}}{a_{N-1}} & \frac{1}{a_{N-1}} \\ 0 & & & \frac{c_N}{a_N} & -\frac{b_N}{a_N} \end{pmatrix} \end{aligned} \qquad (E.5)$$

The eigenvalue comes about from evaluating equation (E.5) at $x = t_j$, and setting $p_N(t_j) = 0$,

$$t_j\,\mathbf{p}(t_j) = \mathbf{T}\,\mathbf{p}(t_j) \qquad (E.6)$$

The matrix, $\mathbf{T}$, is symmetric if the polynomials, $p_j(x)$, are orthonormal. If this is not the case, a diagonal similarity transform is performed,

$$\mathbf{DTD}^{-1} = \mathbf{J} = \begin{pmatrix} \alpha_1 & \beta_1 & & & 0 \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{N-2} & \alpha_{N-1} & \beta_{N-1} \\ 0 & & & \beta_{N-1} & \alpha_N \end{pmatrix} \qquad (E.7)$$

where

$$\alpha_i = -\frac{b_i}{a_i} \qquad \left(\beta_i = \frac{c_{i+1}}{a_i\, a_{i+1}}\right)^{1/2},$$

to obtain a symmetric matrix.

The eigenvalues and eigenvectors of the matrix J are computed from a standard eigenvalue solver. The eigenvalues, which are distinct and real since J is symmetric, are equal to the collocation points. Knowing the eigenvalues, the eigenvectors are easily obtained. The first component of the eigenvectors, $q_{0\,j}^2$ is used to compute the weights,

$$A_n = q_{0\,j}^2\, \mu_0 \qquad \mu_0 = \int_a^b w(x)dx. \tag{E.8}$$

We have obtained the collocation points, x_n, and the weights, A_n, defining the quadrature rule.

Appendix F

Computing Centroid and Impulse

Centroid and Propagation Speed

We begin with the definition of the centroid suggested by Lamb (1945), Saffman (1970) and others.

$$\mathbf{X} = \frac{1}{2}\int_V \frac{\mathbf{r} \times \omega \cdot \mathbf{I}}{\mathbf{I}^2}\,\mathbf{r}\,dV \qquad (F.1)$$

For axisymmetric flow, equation (F.1) reduces to

$$X = \frac{1}{I_x/\rho}\int_A \omega_\phi y^2 x dA \qquad (F.2)$$

As seen in figure F.1, A is the area enclosed by the half plane, $-\infty < x < \infty$, and $0 \leq y < \infty$. The velocity of the centroid, U, is found by taking the time-derivative of equation (F.2):

$$U = \frac{dX}{dt} = \frac{1}{I_x/\rho}\int_A \frac{\partial \omega_\phi}{\partial t} y^2 x dA \qquad (F.3)$$

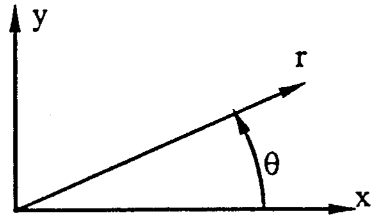


Figure F.1. Definitions of Cartesian coordinates, and spherical polar coordinates with ϕ=constant.

In order to evaluate U for a given vorticity field, we use the vorticity equation (obtained from the curl of the momentum equation).

$$\frac{\partial \omega}{\partial t} = \nabla \times (\mathbf{u} \times \omega) + \nu \nabla^2 \omega \tag{F.4}$$

The rate-of-change of vorticity in equation (F.3) is then replaced by the right hand side of equation (F.4). As mentioned by Saffman (1970), the viscous term does not contribute to the velocity of the centroid, U. This can be shown by expressing the Laplacian of ω_ϕ in cartesian coordinates and rearranging the viscous term, $xy^2\nabla^2\omega$, such that it is expressed as total differentials. Integrating by parts and considering the decay of vorticity at infinity, together with zero vorticity on the axis of symmetry, shows that this term is zero.

Assuming that the flow is axisymmetric and without swirl ($u_\phi = 0$, $\omega_r = \omega_\theta = 0$), the only component of the convection term is

$$(\nabla \times (\mathbf{u} \times \omega))_\phi = -\frac{1}{r}\frac{\partial}{\partial r}(r u_r \omega_\phi) - \frac{1}{r}\frac{\partial u_\theta \omega_\phi}{\partial \theta} \tag{F.5}$$

Substituting equation (F.5) into (F.3) (using eqn. F.4 and the fact that the viscous term does not contribute), and integrating by parts, the velocity of the centroid becomes

$$U = \frac{1}{I_x/\rho}\int_{-1}^{1}\int_{0}^{\infty}\left\{3u_r + u_\theta\frac{(3\mu^2-1)}{\mu\sin\theta}\right\}\omega_\phi r^3 dr \sin\theta \mu d\mu \tag{F.6}$$

where $\mu = \cos\theta$. Therefore, given the velocity or vorticity (since one can be derived from the other), the velocity of the centroid is computed using equation (F.6.).

Impulse

For flows which are unbounded and unforced with a quiescent far-field, the impulse is conserved. Because the governing equations conserve impulse, and it is not built into the method, this serves as a very useful diagnostic. It is computed in two ways. The first is by a multipole expansion (described below), and the second is a direct integration using Gauss quadrature (see Appendix E).

Multipole Expansion:

The ϕ component of the vector potential as $r \to \infty$ to leading order is

$$\Upsilon_\phi = \frac{I}{4\pi}\frac{\sin\theta}{r^2} \tag{F.7}$$

where I is the impulse (Cantwell, 1986). Furthermore Υ_ϕ is related to the stream function, Ψ_ϕ by

$$\Upsilon_\phi = \frac{\Psi_\phi}{r\sin\theta} \tag{F.8}$$

Therefore,

$$\Psi_\phi = \frac{I}{4\pi}\frac{\sin^2\theta}{r} \tag{F.9}$$

From this we can get the velocity component in the radial direction, u, as

$$u = \frac{1}{r^2\sin\theta}\frac{\partial\Psi_\phi}{\partial\theta} = \frac{1}{r^3}\frac{I}{2\pi}\cos\theta \tag{F.10}$$

Now, our spectral expansion for the radial component of velocity is

$$u = \sum_{n\,\ell} a_{n\ell}\ell(\ell+1)\frac{f^+_{n\ell}}{r}P^0_\ell \tag{F.11}$$

where $f^+_{n\ell} = (1-\xi)^{(\ell+1)/2}\xi^{\ell/2}G^\ell_n(\xi)$. For $\ell = 1$,$P^0_1 = \cos\theta$. Hence, we only need the $\ell = 1$ terms. The velocity becomes

$$u = 2\,\frac{(1-\xi)\xi^{1/2}}{r}\,\cos\theta\sum_n a_{n\,1}G^1_n(\xi) \tag{F.12}$$

where $G^1_n(\xi)$ is the normalized Jacobi polynomial family where $p = 1$, $q = 1/2$.

Next, we need to determine the leading term in u as $r \to \infty$. or $\xi \to 1$.

$$u_\infty \approx \frac{2}{r}\cos\theta\,\xi^{1/2}\,(1-\xi)\sum_n a_{n1}\lim_{\xi\to 1}G_n(\xi) \tag{F.13}$$

The results for $\lim_{\xi\rightarrow 1}(1-\xi)$ and $\lim_{\xi\rightarrow 1} G_n(\xi)$, described in detail below, are

$$\lim_{\xi\rightarrow 1}(1-\xi) \approx r_1^2/r^2 \qquad (F.14)$$

$$\lim_{\xi\rightarrow 1} G_n(\xi) \approx \left(\frac{2}{\pi}\right)^{1/2} (2n+1) \qquad (F.15)$$

Substituting equation (F.14) and (F.15) into equation (F.13) gives

$$u_\infty = 2\cos\theta \left(\frac{2}{\pi}\right)^{1/2} r_1^2/r^2 \sum_n a_{n\,1}(2n+1) \qquad (F.16)$$

The velocity at a large radius, u_∞, is then substituted into equation (F.10) giving

$$I = 4\pi \frac{2}{\pi}^{1/2} r_1^2 \sum_n a_{n\,1}(2n+1) \qquad (F.17)$$

Limiting form of Jacobi polynomials as $r \rightarrow \infty$:

Abramowitz and Stegun (1972) give coefficients of the expansion for Jacobi polynomials, $P_n^{(\alpha,\beta)}(x)$ (in their notation) where $1 \leq x \leq 1$, of the form:

$$P_n^{(\alpha,\beta)}(x) = a_n^{-1} \sum_{m=0}^{n} c_m\,(x-1)^m. \qquad (F.18)$$

These polynomials, $P_n^{(\alpha,\beta)}(x)$, are related to shifted Jacobi polynomials through A&S equation (22.5.2). In addition, we must scale the Jacobi polynomials in A&S such that they are orthonormal. Applying these transfomations gives an expansion in terms of $1-\xi$, where ξ is the mapped coordinate (eqn. C.1). From equation (C.5), note that the leading term for $1-\xi$ at large r is of order $1/r^2$. Therefore, the limiting behavior as $r \rightarrow \infty$ (or $\xi \rightarrow 1$) is simply the first term in the expansion. From this analysis, we find that the leading term as a function of n for orthonormal shifted Jacobi polynomials at large radius is

$$\lim_{r\rightarrow\infty} G_n^\ell \approx \frac{\sqrt{2}}{\sqrt{\pi}}(2n+1). \qquad (F.19)$$

Appendix G
MACSYMA Code

MACSYMA is a high level programming language which is able to manipulate symbols. In addition, it includes extensive libraries of algorithms for solving mathematical problems such as integral equations, differential equations, etc.. It is especially powerful in solving problems where the algebra becomes very lengthy and involved. The elements of the mass and viscous matrix were computed analytically using MACSYMA (see section 2.2 and 3.4). One reason to compute these analytically is to mimimize the roundoff error (see section 3.5).

We are interested in evaluating the integrals given by equations (3.4.1) and (3.4.3), the mass and viscous matrices, respectively. The way in which this was done was to express these integrals in terms of sums of Jacobi polynomials (using the derivative and recurrence relations of Appendix C) and then apply the orthogonality relations (Appendix C) to evaluate products of Jacobi polynomials. Therefore, the problem proceeds in two stages. The first stage is to find the recurrence relations and differential relations for the Jacobi polynomials used in this work. The next stage is to write a program (or macro) to carry out the process of substituting the recurrence and derivatives relations, and evaluating the integrals.

As discussed in Appendix C, we are using orthonormal, shifted Jacobi polynomials. These are a modified form of those shown in Abramowitz and Stegun (1972). The Jacobi polynomials are functions of two indices (n and ℓ) in order that the matrices are banded. This leads more complex recurrence, derivative, and Rodrigues formulas. To avoid mistakes, this process is programmed: the listing is entitled **Jacobi**. Three formulas are derived here. The recurrence relation, *recur*, the derivative relation, *difeq*, and the macro which generates the Rodrigues formulas of any specified order, *Rodrigues.*

The second listing, **Matrix Macros**, shows several marcos which are called in succession in order to evaluate the integrals, followed by the a listing of the program used to obtain the elements of the mass matrix. The macros are: (1) *repxi* which replaces ξG_n^ℓ with the appropriate linear combination of G_n^ℓ (eqn. $C.25$), (2) *repder* which replaces occurances of $\xi(1-\xi)\frac{dG_n^\ell}{d\xi}$ with the appropriate linear combination (eqn. $C.26$), (3) *getcoef* extracts the coefficients in front of each of the Jacobi polynomials, and (4) *genmat* uses the orthogonality relations and solves for the matrix elements (ie. the diagonal and off-diagonal nonzero elements).

In the mass matrix shown by equation (3.4.1), there are two terms. Because the there wasn't sufficient memory to solve the entire problem at one time, each of the two terms are computed during a separate session and the results from the two sessions were combined to give the final answer. The macro which combines these elements is *forma.*

The viscous matrix was computed in a similar manner. This matrix involves another derivative, and 4 terms instead of 2, so the algebra is considerably more involved. Again because of memory restrictions, the problem had to be solved in pieces and then combined to give the final result.

Jacobi

```
recurrance(recur)::=buildq([recur],
 (assume(n>=0),
 assume(el>=1),
 assume(n+el>=1),

 /* coefficients in Jacobi recurrance relation */
 p:2*el-1,
 q:el-1/2,
 a1 : factor(-(2*n*(n+p) +q*(p-1)) /((2*n+p)^2-1)),
 a2 : factor(n *(n+q-1) *(n+p-1) *(n+p-q) /((2*n+p-2)*(2*n+p-1)^2*(2*n+p))),

 /* normalized Jacobi polynomial factors */
 h[n]:N!*GAMMA(P+N)*GAMMA(-Q+P+N+1)*GAMMA(Q+N)/((P+2*N)*GAMMA(P+2*N)**2),
 h[n]:makefact(h[n]),
 h[n+1]:subst(n+1,n,h[n]),
 h[n-1]:subst(n-1,n,h[n]),
 NORM1:sqrt(factor(minfactorial(h[n+1]/h[n]))),
 NORM2:sqrt(factor(minfactorial(h[n-1]/h[n]))),

 /* recurrence relation for normalized Jacobi polynomials */
 recur: y*g[n](y) = NORM1*g[n+1](y) - a1*g[n](y) + NORM2*a2*g[n-1](y)))$

 Rodrigues(nmax,G,term2) ::=buildq([nmax,G,term2],
   (p:2*el-1, q:el-1/2, alpha:el-1/2, beta:el-3/2,
   rho: (1-x)^alpha * (1+x)^beta,
   g: 1 - x^2,
   rhoy: subst(2*y-1,x,rho),
   gy: subst(2*y-1,x,g),
   for n:0 thru nmax do
   (  an:(-1)^n * 2^n * factorial(n),
   term1: (factorial(n)*gamma(n+p))/gamma(2*n+p),
   h[n]:N!*GAMMA(P+N)*GAMMA(-Q+P+N+1)
             *GAMMA(Q+N)/((P+2*N)*GAMMA(P+2*N)^2),
   NORM:sqrt(h[n]),
   term2[n]: term1/(norm*an*rhoy*2^n),
   G[n]:factor(diff(rhoy*gy^n,y,n)*term2[n]),
   print("G[",n,"] is", G[n])
   )
   )
                                     )$

 /* Differential recurrence relation*/
 alpha : el - 1/2;
 beta : el - 3/2;

 g2 : (2*n + alpha + beta) * (1-xi^2);
 g1 : n*(alpha - beta - (2*n + alpha + beta) *xi);
 g0 : 2* (n + alpha) * (n + beta);

 eq:g2 * diff(f[n](xi),xi) = g1*f[n](xi) + g0*f[n-1](xi);
 eq: subst(2*xi-1,xi,eq);
 eq: ratsubst(diff(f[n](2*xi-1),xi)/2,diff(f[n](2*xi-1),2*xi-1),eq);

 z:2*xi-1;
 for k:n-1 thru n do
 ( term[k]:makefact( gamma(2*k+p)/ (factorial(k)*gamma(k+p)) ),
   f[k](z)  := g[k](xi)*term[k] );

 eq:eq/term[n],diff;
 eq:factor(minfactorial(expand(eq)));
 eq:eq/(-4*(n+el-1));

 /* eq2: normalized differential relation */
 c[n]:ratcoeff(rhs(eq),g[n](xi));
 c[n-1]:factor(ratcoeff(rhs(eq),g[n-1](xi)));
 eq2:lhs(eq) = c[n]*g[n](xi) + NORM2*c[n-1]*g[n-1](xi);

 /*as a check, eq2:lhs(eq) = c[n]*g[n](xi) + c[n-1]*g[n-1](xi);*/
 /*then, is ratsimp(rhs(eq)-rhs(eq2)) = 0 */

 eq2:ratsubst(rhs(recur),lhs(recur),eq2);

 lhseq2:factor(lhs(eq2));
 c2[n+1]:factor(ratcoeff(rhs(eq2),g[n+1](xi)));

 c2[n]:factor(ratcoeff(rhs(eq2),g[n](xi)));
 c2[n-1]:factor(ratcoeff(rhs(eq2),g[n-1](xi)));
 difeq:lhseq2=c2[n-1]*g[n-1](xi)  + c2[n]*g[n](xi) + c2[n+1]*g[n+1](xi);
```

Matrix Macros

```
/*Procedure to replace xi * Jacobi polynomial with appropriate linear combin
repxi ( func , wid ) ::= buildq ( [ func , wid ],
for k: n-wid thru n+wid do
func : ratsubst  ( Gt[k+1] (xi)*sqrt(k+1)*sqrt(k+2*el-1)/(k+el)/4
        +Gt[k] (xi)*(2*k^2+4*el*k-2*k+2*el^2-3*el+1)/((k+el-1)*(k+el))/4
        +Gt[k-1] (xi)*sqrt(k)*sqrt(k+2*el-2)/(k+el-1)/4,
       xi * Gt[k](xi), func )  )$

/*Procedure to replace xi * (1-xi) * dGn/dxi with appropriate linear combina
repder ( func , wid ) ::= buildq ( [ func , wid ],
for k: n-wid thru n+wid do
func : ratsubst  ( -Gt[k+1] (xi)  *k*sqrt(k+1)*sqrt(k+2*el-1)/(k+el)/4
        +Gt[k] (xi)  *k*(k+2*el-1)/((k+el-1)*(k+el))/4
        +Gt[k-1] (xi)  *sqrt(k)*sqrt(k+2*el-2)*(k+2*el-1)/(k+el-1)/4,
        xi * (1-xi) * diff( Gt[k](xi), xi), func ) )$

/* Extract coefficients */
getcoef(func,wid,coef,funcout) ::= buildq([func,wid,coef,funcout],
       (funcout:0,
       for j:-wid thru +wid do
          (tmp:divide(func,Gt[n+j](xi),Gt[n+j](xi)),
           coef[j]:factor(tmp[1]),
           func:tmp[2],
           funcout:funcout+coef[j]*Gt[n+j](xi)
          )
       )
                                )$

genmat(coef1,coef2,wid,matrix,amat0,amat1) ::=
                              buildq( [coef1,coef2,wid,matrix,amat0,amat1],
    (coef2[-wid-2]:0, coef2[-wid-3]:0,
     for j:0 thru 2*wid+1 do
        (matrix[n+j]:0,
         for k: -wid thru wid do
              (matrix[n+j]:matrix[n+j] + coef1[k]*subst(n+j,n,coef2[k-j])
              ), matrix[n+j]:factor(matrix[n+j]),
         print("matrix element ",n+j, "is", matrix[n+j]),
         amat0[n]: -(coef1[-1]*coef2[0]+coef1[0]*coef2[-1]+coef1[1]*coef2[-2]
         amat1[n]: -coef1[-1]*coef2[-2],
         amat0[n+1]: -(coef1[-1]*subst(n+1,n,coef2[-1])
                                        + coef1[0]*subst(n+1,n,coef2[-2])),
         amat0[n+2]: -coef1[-1]*subst(n+2,n,coef2[-2])
         )
   )
                                            )$

/* form the mass matrix elements by combining the first and second parts */
forma(mass1,mass2,a1mat0,a2mat0,a1mat1,a2mat1,wid,mat,mat0,mat1) ::=
    buildq([mass1,mass2,a1mat0,a2mat0,a1mat1,a2mat1,wid,mat,mat0,mat1],
      (for j:0 thru wid do
      (mat[n+j]:factor(ratsimp(-r1^3*(mass1[n+j] + 2*mass2[n+j])/6) )),
      mat0[n]:  factor(ratsimp(-r1^3*( (mass1[n]+a1mat0[n] )
                        + 2*(mass2[n]+a2mat0[n] ) )/6) ),
      mat0[n+1]:factor(ratsimp(-r1^3*( (mass1[n+1]+a1mat0[n+1] )
                        + 2*(mass2[n+1]+a2mat0[n+1] ) )/6) ),
      mat0[n+2]:factor(ratsimp(-r1^3*( (mass1[n+2]+a1mat0[n+2] )
                        + 2*(mass2[n+2]+a2mat0[n+2] ) )/6) ),
      mat1[n]:  factor(ratsimp(-r1^3*( (mass1[n]+a1mat1[n] )
                        + 2*(mass2[n]+a2mat1[n] ) )/6) )
      )
         )$
```

Listing for Mass Matrix

```
/* Building blocks of functions */
dxdr:2*sqrt(xi)*(1-xi)^(3/2)/rl;
Fnl: (1-xi)^((el+1)/2)*xi^(el/2)*Gt[n](xi);
r: rl*sqrt(xi)/sqrt(1-xi);

/* First half of mass matrix, first function */
f:ratsimp((dxdr*diff(Fnl,xi)-el*Fnl/r)*xi^(-(el-1)/2)*(1-xi)^(-(el+2)/2));

repder(f,0);
repxi(f,0);

getcoef(f,1,cf,fsimp)$

/* First half of mass matrix, second function */
fp:fsimp*xi;
repxi(fp,1);
getcoef(fp,2,cfp,fpsimp)$

/* With first and second function, find the first half of mass matrix */
genmat(cf,cfp,1,mass1,almat0,almat1)$

/* second half of mass matrix, first function */
/*f2:ratsimp((dxdr*diff(Fnl,xi)+(el+1)*Fnl/r)
                     *xi^(-(el-1)/2)*(1-xi)^(-(el+2)/2)); */

/*repder(f2,0); */
/*repxi(f2,0); */

/*getcoef(f2,1,cf2,f2simp)$ */

/* second half of mass matrix, second function */
/*f2p:f2simp*xi; */
/*repxi(f2p,1); */

/*getcoef(f2p,2,cf2p,f2psimp)$ */

/* With first and second function, find the second half of mass matrix */
/*genmat(cf2,cf2p,1,mass2,a2mat0,a2mat1)$ */

/* form the mass matrix elements */
/*forma(mass1,mass2,almat0,a2mat0,almat1,a2mat1,3,amat,amat0,amat1)$ */
```

Appendix H

Initial Condition Code

The task of solving the Navier-Stokes equations is divided into two separate codes. The first (*IC*) takes an initial condition in the form of a vorticity distribution, and finds the coefficients of the expansion approximating the vorticity field. The second code (*NS*) starts with the coefficients computed by *IC* and marches the solution forward in time, a specified number of time steps. This Appendix will describe *IC*. We will begin by deriving the matrix equation which is solved to obtain the initial condition. Then a flowchart for solving this problem is presented and finally, a listing is attached.

The vorticity field for axisymmetric flow in spherical polar coordinates contains only one component

$$\omega_\phi = \sum_{n\ell} a_{n\ell}(t)\, L_\ell(f_{n\ell})\, P_\ell^1 \qquad (H.1)$$

Multiply both sides of equation (H.1) by $\left(L_{\ell'}\left(f_{n'\ell'}\right)P_{\ell'}^1\right)^*$ and integrate over the domain, $0 \leq \mu \ < 1$ and $0 \leq r < \infty$. Apply the orthogonality relations for associated Legendre polynomials

$$\frac{2\ell(\ell+1)}{2\ell+1} \sum_n a_{n\ell} \int_0^\infty L_\ell(f_{n\ell})\, L_\ell(f_{n'\ell})\, r^2 dr = \int_0^\infty \int_{-1}^1 \omega_\phi\, L_\ell(f_{n'\ell})\, P_\ell^1\, d\mu\, r^2 dr \qquad (H.2)$$

Rearranging equation (H.2) gives a set of N coupled equations for each ℓ, with

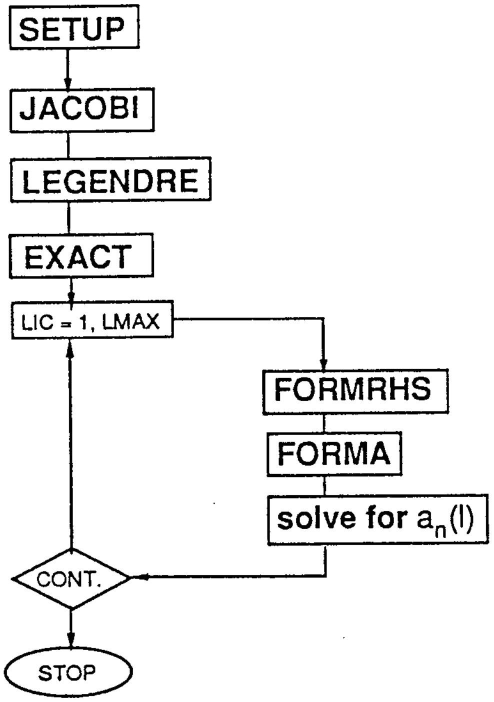


Figure H.1. Flowchart of *IC*.

unknown vectors, $a_{n\ell}$,

$$C^{\ell}_{n\,n'}\, a_{n\ell} = \frac{2\ell+1}{2\ell(\ell+1)} \int_0^{\infty} \left[\int_{-1}^{1} \omega_\phi\, P^1_\ell(\mu)\, d\mu \right] L_\ell(f_{n'\ell})\, r^2 dr \qquad (H.3)$$

where $C^{\ell}_{n\,n'} \equiv \int_0^\infty L_\ell(f_{n\ell})\, L_\ell(f_{n'\ell})\, r^2 dr$. This matrix, $C^{\ell}_{n\,n'}$, is symmetric and positive-definite, and has a semi-bandwidth of 5. It is related to the viscous matrix. The elements of $C^{\ell}_{n\,n'}$ were analytically computed using MACSYMA.

For a specified vorticity distribution, the right hand side of equation (H.3) is computed using Gauss-Legendre and Gauss-Jacobi quadrature. The matrix $C^{\ell}_{n\,n'}$ is inverted by Cholesky decomposition, giving the coefficients, $a_{n\ell}$.

A flowchart of *IC* is shown in figure H.1. The solution procedure is quite straightforward. **SETUP** reads the inputs, N, N_c, L, L_c and the parameters defining the initial vorticity field, and solves for the collocation points and weights for Gauss-Legendre and Gauss-Jacobi quadrature. **JACOBI** evaluates the radial basis functions at the collocation points, $i = 1, \ldots, N_c$ for each $n \equiv NI$ and $\ell \equiv LI$. **LEGENDRE** evaluates the polar basis functions at the collocation points, $j = 1, \ldots, L_c$, for each $\ell \equiv LI$. **EXACT** contains the initial conditions of Stokes and thin rings, where the specified initial condition are evaluated at the collocation points. For a given ℓ, **FORMRHS** computes the right hand side of equation (H.3) and **FORMA** computes the elements of the matrix $C^{\ell}_{nn'}$. The coupled set of ordinary differential equations are then solved using the IMSL banded Cholesky solvers **LUDAPB** (decomposes $C^{\ell}_{nn'}$) and **LUELPB** (solves the linear system) to get the coefficients $a_{n\ell}$ for that ℓ. This procedure is repeated for each ℓ.

```
*COMDECK VORT4
      PARAMETER( NMDIM=92, NCDIM=92, IMDIM=92, LCDIM=92,
     1           PI = 3.14159265358979323846)
      PARAMETER(LDWRITE=3,LREAD=5,IWRITE=6)
      DOUBLE PRECISION XIDD,RCONSTD
      COMMON/XIDOUBL/ XIDD(NCDIM)
      COMMON/BCOLLOC/ XI(NCDIM),WIR(NCDIM),CTH(LCDIM),WTTH(LCDIM),
     1                R(NCDIM)
      COMMON/BJACOBI/ GVD(NMDIM,IMDIM,NCDIM)
      COMMON/BLEGEND/ P1(IMDIM,LCDIM)
      COMMON/BMATRIX/ SIC(NMDIM,6),SICINV(NMDIM,6),RHS(NMDIM),
     1                SOL(NMDIM)
      COMMON/BEXACT/ VOR(NCDIM,LCDIM)
      COMMON/BCOEFF/ TIME,ALP(NMDIM,IMDIM)
      COMMON/BGNRL/ NMAX,NCOL,IMAX,LCOL,NMAXA(NMDIM),IC
      COMMON/BPARM/ RADIUS,RCONST,VISC,GAM,CtR,RCONSTD,
     1              IRUN,IFRAME
*DECK ICMAIN
      PROGRAM MAIN
C
C ***------------- MAIN ROUTINE -------------*****
*CALL VORT4

      CALL SETUP
      DO 25 LI = 1,IMAX
        NMAXA(LI) = NMAX - LI +1
 25   CONTINUE

      CALL LEGENDRE

      CALL JACOBI

      CALL EXACT

C begin l index here:
      DO 100 LI=1,IMAX
        NMAX = NMAXA(LI)

        CALL FORMRHS(LI)

        CALL FORMA(LI)

c decompose matrix, SIC
        N = NMAXA(LI)
        NC = 5
        IA = NMDIM
        IU = NMDIM
        CALL LUDAPB (SIC,N,NC,IA,SICINV,IU,D1,D2,IER)

        IF (IER.NE.0) THEN
        WRITE(LWRITE,1000)LI
 1000   FORMAT(X,'WARNING: the error parameter is nonzero for LI =',I2)
        END IF

c find SOL, given RHS and decomposed matrix, SICINV
        CALL LUELPB(SICINV,RHS,N,NC,IA,SOL)

C Copy resultant vector into ALP(NI,LI):
        DO 95 NI=1,NMAX
          ALP(NI,LI)=SOL(NI)
 95     CONTINUE

 100  CONTINUE

      CALL OUTPUT

      STOP
      END

C ------------------------------------------------------------

      SUBROUTINE SETUP
C
*CALL VORT4
      DOUBLE PRECISION XI1D,RADD
C Input:
C      > the order of the matrix (Nmax).
C      > the number of collocation points (Ncol).
C      > the greatest value of the quantum number in theta (Imax).
C      > the number of collocation points in theta (Lcol).
C      > the radius of the vortex ring (Radius).
C      > initial condition

      READ(LREAD,1040)
      READ(LREAD,*) NMAX, NCOL, IMAX, LCOL
      READ(LREAD,1040)
      READ(LREAD,*) RADIUS, VISC, GAM, CtR
      READ(LREAD,1040)
      READ(LREAD,*) IC, IRUN, IFRAME
 1040 FORMAT( 1X )
C
C Call quad to get absissas and weights in the radial direction.
      CALL QUAD(NCOL,WIR,XI,XIDD)

C Call QUADAZ to get The collocation points in the theta direction.
      CALL QUADAZ(LCOL,WTTH,CTH)
C
C Calculate RCONST.
       IMIN = INT(REAL(NCOL)*3.2/5.)
       XI1 = XI(IMIN)
       RCONST = RADIUS*((1.-XI1)/XI1)**(1./2.)
       XI1D = XIDD(IMIN)
       RADD = RADIUS
       RCONSTD = RADD*((1.-XI1D)/XI1D)**(1./2.)
C
C Find the radial coordinate at each collocation point.
      DO 31 I = 1,NCOL
        XI1 = XI(I)
        R(I) = RCONST*(XI1/(1.-XI1))**(1./2.)
 31   CONTINUE

      RETURN
      END

C ------------------------------------------------------------

      SUBROUTINE JACOBI
C this subtroutine is defined for p=i, q=1/2

*CALL VORT4
C G(NI,I) or G(NMDIM,NCDIM)
      DOUBLE PRECISION N,EL,G(90,90),DUM,GNORM,XID,r1
C      DOUBLE PRECISION DGAMMA
      REAL GAMMA,ARG1,ARG2,SNORM

      R1 = RCONST
      DO 50 LI = 1,IMAX

        EL = LI

C compute the Jacobi Polynomials for all n and xi at a given EL.
C this is only an intermediate result, it is not needed outside of
C this subroutine.
      DO 10 I = 1,NCOL
C       n = 0, NI = 1
        ARG1 = 2*EL-1
        ARG2 = (2*EL-1)/2.0
        SNORM = SQRT(2.)*SQRT(GAMMA(ARG1))/GAMMA(ARG2)
        GNORM = SNORM
        G(1,I) = 1.0d+0
C       n = 1, NI = 2
        XID = XIDD(I)
        G(2,I) = (4*EL*XID-2*EL+1)/DSQRT(2*EL-1)
 10   CONTINUE
C general recurrance formula  - at that paricular el.
C Note: if necessary, some memory can be saved by storing only
C ---- 5 Gn's at a given time... Gn+2, Gn+1, Gn, Gn-1, Gn-2.
      DO 20 NI = 3,NMAXA(LI)+2
        DO 30 I=1,NCOL

      N = NI-2
      XID = XIDD(I)
      G(NI,I) = (((4*N**2+(8*EL-4)*N+4*EL**2-4*EL)*XID-2*N**2+(2-
     1  4*EL)*N-2*EL**2+3*EL-1)*G(NI-1,I)+(-N*DSQRT(N**2+2*EL*N-2*N)-
     2  EL*DSQRT(N**2+2*EL*N-2*N))*G(NI-2,I))/((N+EL-1)*DSQRT(N**2
     3  +2*EL*N+2*EL-1))

 30     CONTINUE
 20   CONTINUE
```

```
C Knowing the Jacobi polynomials, compute (what amounts to) the
C vorticity basis function.
      DO 60 I = 1,NCOL
        XID = XIDD(I)
        dum = XID*((EL*(4*EL+8)+3)*XID+(-4*EL-8)*EL-3)/R1**2
        GVD(1,LI,I) = dum * GNORM
        dum = XID*(XID*(EL*(EL*(16*EL+64)+60)*XID+EL*
     1   ((-24*EL-92)*EL-82)+3)+EL*(EL*(8*EL+28)+22)-3)
     2   /(DSQRT(2*EL-1)*R1**2)
        GVD(2,LI,I) = dum * GNORM
 60   CONTINUE

      DO 70 NI = 3,NMAXA(LI)
      N = NI-1

        DO 80 I = 1,NCOL
      DUM =
     1((N*(N*(N*(4*N+16*EL-4)+EL*(24*EL-12)-13)+EL*(EL*(16*EL-12)-26)+7)
     1   +EL*(EL*(EL*(4*EL-4)-13)+7)+6)*DSQRT(N**4+(4*EL+2)*N**3+(4*EL**
     2   2+10*EL-1)*N**2+(12*EL**2+2*EL-2)*N+8*EL**2-4*EL)*G(NI+2,I)+(N
     3   *(N*((-8*N-32*EL+12)*N+(32-48*EL)*EL+20)+EL*((28-32*EL)*EL+46)-
     4   18)+EL*(EL*((8-8*EL)*EL+26)-14)-12)*DSQRT(N**2+2*EL*N+2*EL-1)*
     5   G(NI+1,I)+(N*(N*(N*(N*((-8*N-48*EL+24)*N+(88-128*EL)*EL-2)+EL*(
     6   (128-192*EL)*EL+56)-36)+EL*(EL*((96-168*EL)*EL+186)-68)+4)+EL*(
     7   EL*(EL*((40-80*EL)*EL+196)-42)-64)+18)+EL*(EL*(EL*(EL*((8-16*EL
     8   )*EL+68)-10)-64)+2)+12)*G(NI,I)+(N*(N*(N*(8*N+32*EL-20)+EL*(4
     9   8*EL-56)-8)+EL*(EL*(32*EL-52)-18)+26)+EL*(EL*(EL*(8*EL-16)-10)+
     :   20)+6)*DSQRT(N**2+(2*EL-2)*N)*G(NI-1,I)+(N*(N*(N*(4*N+16*EL-12
     ;   )+EL*(24*EL-36)-1)+EL*(EL*(16*EL-36)-2)+15)+EL*(EL*(EL*(4*EL-12
     <   )-1)+15))*DSQRT(N**4+(4*EL-6)*N**3+(4*EL**2-14*EL+11)*N**2+(-4*
     =   EL**2+10*EL-6)*N)*G(NI-2,I))/((N*(N*(N*(16*N+64*EL-32)+EL*(96*
     >   EL-96)-16)+EL*(EL*(64*EL-96)-32)+32)+EL*(EL*(EL*(16*EL-32)-16)+
     ?   32))*R1**2)

C Copy the double precision variable, DUM, to the single precision array
        GVD(NI,LI,I) = DUM * GNORM

 80     CONTINUE
 70   CONTINUE

 50   CONTINUE

      RETURN
      END

C --------------------------------------------------------------------
C
      SUBROUTINE LEGENDRE

*CALL VORT4

C Calulate Legendre polynomial, P1(LI,J).
      DO 63 J=1,LCOL
        X1=CTH(J)
        SQX2=SQRT(1.-X1*X1)
        P1(1,J)=-SQX2
        P1(2,J)=-3.*X1*SQX2
 63   CONTINUE

      DO 64 LI=3,LMAX
        L=LI-1
        FAC1=(2.*L+1.)/(L+1.)
        FAC2=L/(L+1.)
        FAC3=(2.*L+1.)/L
        DO 65 J=1,LCOL
          X1=CTH(J)
          P1(LI,J)=FAC3*X1*P1(LI-1,J)-P1(LI-2,J)/FAC2
 65     CONTINUE
 64   CONTINUE

      RETURN
      END
C
C --------------------------------------------------------------------
C
      SUBROUTINE EXACT

*CALL VORT4

C Calculate velocity at collocation points.
```

```
      READ(LREAD,1040)
      READ(LREAD,*) TIME
 1040 FORMAT( 1X )
      DX = 0.0
      RNUT = VISC * TIME
      IF (IC.EQ.1) GOTO 10
      IF (IC.EQ.5) GOTO 15
      IF (IC.EQ.2) GOTO 20
      IF (IC.EQ.3) GOTO 30
      IF (IC.EQ.4) GOTO 40

C Gaussian ring with an image. Input: R, a/R, Gamma/nu
C -------------------------------------------------------
 10    CONTINUE
      FIXED = 1.2564392281
      CAPR = RADIUS
      DO 110 J=1,LCOL
        STH = SIN( ACOS(CTH(J)) )
        DO 100 I=1,NCOL
          R1 = R(I)
          ARG =  EXP(  -(FIXED/CtR**2)* ( 1.+ (R1*R1/(CAPR*CAPR))
     1    - 2.*R1*STH/CAPR )  )
          ARG2 =  EXP( -(FIXED/CtR**2)* ( 1.+ (R1*R1/(CAPR*CAPR))
     1    + 2.*R1*STH/CAPR )  )
          VOR(I,J)= FIXED *GAM *(ARG-ARG2)
     1    /(PI* VISC *CAPR**2 *CtR**2)
 100    CONTINUE
 110  CONTINUE
      GOTO 1000

C Two Gaussian rings with an image. Input: R, a/R, Gamma/nu
C -------------------------------------------------------
 15    CONTINUE
      FIXED = 1.2564392281
      CAPR = RADIUS
      XX = 1.5
      DO 115 J=1,LCOL
        DO 105 I=1,NCOL
          X1 = R(I)*CTH(J) - XX
          Xb = R(I)*CTH(J) + XX
          Y = R(I)*SIN(ACOS(CTH(J)))
          RN1 = SQRT(X1**2 + Y**2)
          THN1 = ATAN2(Y,X1)
          STHN1 = SIN(THN1)
          RNb = SQRT(Xb**2+ Y**2)
          THNb = ATAN2(Y,Xb)
          STHNb = SIN(THNb)
          ARG1 =  EXP(  -(FIXED/CtR**2)* ( 1.+ (RN1*RN1/(CAPR*CAPR))
     1    - 2.*RN1*STHN1/CAPR )  )
          ARG2 =  EXP(  -(FIXED/CtR**2)* ( 1.+ (RN1*RN1/(CAPR*CAPR))
     1    + 2.*RN1*STHN1/CAPR )  )
          ARG1b =  EXP(  -(FIXED/CtR**2)* ( 1.+ (RNb*RNb/(CAPR*CAPR))
     1    - 2.*RNb*STHNb/CAPR )  )
          ARG2b =  EXP(  -(FIXED/CtR**2)* ( 1.+ (RNb*RNb/(CAPR*CAPR))
     1    + 2.*RNb*STHNb/CAPR )  )
          VOR(I,J)= FIXED *GAM *(ARG1+ARG1b-ARG2-ARG2b)
     1    /(PI* VISC *CAPR**2 *CtR**2)
 105    CONTINUE
 115  CONTINUE
      GOTO 1000

C Gaussian ring without an image. Input: time
C ------------------------------------------
 20    CONTINUE
C Vorticity of a vortex ring of r(core)/R which is determined by "TIME"

      DO 130 J=1,LCOL
        STH = SIN( ACOS(CTH(J)) )
        DO 140 I=1,NCOL
          R1 = R(I)
          ARG = RADIUS*RADIUS + R1*R1 + DX*DX
     1             - 2.*R1*(RADIUS*STH + DX*CTH(J))
          IF (ARG.LT.0.) ARG = 0.
          RHO = SQRT(ARG)
          EXPFUN = EXP(-RHO*RHO/4.)
          VOR(I,J) = (GAM/VISC) * EXPFUN / (4.*PI)
 140    CONTINUE
 130  CONTINUE
      GOTO 1000
```

```
C Stokes ring. Input: time
C ------------------------
 30    CONTINUE
C Angular dependence:
      DO 150 J=1,LCOL
       STH = SIN(ACOS(CTH(J)))
C Radial dependence:
       DO 160 I=1,NCOL
         VOR(I,J) = STH*R(I)*EXP( -R(I)**2/4. )
     1      /(TIME*16.*PI**(3./2.))
 160   CONTINUE
 150  CONTINUE
      GOTO 1000

C Two Stokes rings. Input: time
C -----------------------------
 40    CONTINUE
C A test case for the low Re no. vortex ring
       XX = 3.0
C Angular dependence:
        DO 170 J=1,LCOL
          DO 180 I=1,NCOL

c Two Stokes vortex rings:
C Radial dependence:
          X1 = R(I)*CTH(J) - XX
          X2 = R(I)*CTH(J) + XX
          Y = R(I)*SIN(ACOS(CTH(J)))
          RNEW1 = SQRT(X1**2 + Y**2)
          THNEW1 = ATAN2(Y,X1)
          RNEW2 = SQRT(X2**2+ Y**2)
          THNEW2 = ATAN2(Y,X2)
         VOR(I,J) = ( .5*SIN(THNEW1) *RNEW1 *EXP(-RNEW1**2/4.)
     1      + .5*SIN(THNEW2) *RNEW2 *EXP(-RNEW2**2/4.)
     2      )/(TIME*16.*PI**(3./2.))
 180   CONTINUE
 170  CONTINUE

 1000  CONTINUE

      RETURN
      END

C ----------------------------------------------------------
C
      SUBROUTINE FORMA(LI)

*CALL VORT4
      INTEGER NI,LI
C DIC(NMDIM,6)
      DOUBLE PRECISION N,EL,DIC(90,6)
      DOUBLE PRECISION DSQRT

      EL = LI

C Form DIC (initial condition Matrix).

      DIC(1,1) = 0.
      DIC(1,2) = 0.
      DIC(1,3) = 0.
      DIC(1,4) = 0.
      DIC(1,5) = 0.

      DIC(2,1) = 0.
      DIC(2,2) = 0.
      DIC(2,3) = 0.
      DIC(2,4) = 0.

      DIC(3,1) = 0.
      DIC(3,2) = 0.
      DIC(3,3) = 0.

      DIC(4,1) = 0.
      DIC(4,2) = 0.

      DIC(5,1) = 0.

      DIC(1,6) =
```

```
1(EL*(EL*(EL*(EL*(EL*(EL*(128*EL+832)+1952)+1936)+536)-356)-258)-
1    45)/(EL*(EL*(512*EL+1536)+1024))

 DIC(2,5) =
1(EL*(EL*(EL*((-32*EL-208)*EL-496)-536)-258)-45)*DSQRT(8*EL**3-12*
1    EL**2+6*EL-1)/(EL*(EL*(512*EL+1536)+1024))

 DIC(3,4) =
1DSQRT((2*EL-1)/EL)*(EL*(EL*(EL*(EL*((-64*EL-576)*EL-2032)-3552)-
1    3196)-1380)-225)/((EL*(EL*(256*EL+1536)+2816)+1536))

 DIC(4,3) =
1(EL*(EL*(EL*(16*EL+112)+288)+324)+135)*DSQRT((48*EL**4+48*EL**3-
1    12*EL-3)/EL)/((EL*(EL*(256*EL+1536)+2816)+1536))

 DIC(5,2) =
1(EL*(EL*(EL*(16*EL+160)+552)+792)+405)*DSQRT((96*EL**4+96*EL**3-
1    24*EL-6)/(EL**2+EL))/((EL*(EL*(512*EL+4608)+13312)+12288))

 DIC(6,1) =
1((-4*EL-24)*EL-35)*DSQRT((3840*EL**7+21120*EL**6+43200*EL**5+
1    37920*EL**4+8400*EL**3-7560*EL**2-4860*EL-810)/(EL**2+EL))
2    /((EL*(EL*(512*EL+4608)+13312)+12288))


 DIC(2,6) =
1(EL*(EL*(EL*(EL*(EL*(EL*(EL*(128*EL+1472)+6624)+14768)+16744)+8468
1    )+1002)+9)+135)/(EL*(EL*(EL*(512*EL+3072)+5632)+3072))

 DIC(3,5) =
1(EL*(EL*(EL*(EL*(EL*((-64*EL-576)*EL-1936)-3056)-2476)-1372)-795)-
1    225)/((EL*(EL*(256*EL**(3.0d+0/2.0d+0)+1536*DSQRT(EL))+2816
2    *DSQRT(EL))+1536*DSQRT(EL)))

 DIC(4,4) =
1DSQRT((6*EL+3)/EL)*(EL*(EL*(EL*(EL*(EL*((-64*EL-896)*EL-5008)
1    -14208)-21420)-15960)-4059)+540)/((EL*(EL*(EL*(256*EL+2560)
2    +8960)+12800)+6144))

 DIC(5,3) =
1(EL*(EL*(EL*(EL*(EL*(128*EL+1440)+6288)+13424)+14208)+6282)+405)*
1    DSQRT((12*EL+6)/(EL**2+EL))/((EL*(EL*(512*EL+4608)+13312)
2    +12288))

 DIC(6,2) =
1(EL*(EL*(EL*(EL*(16*EL+224)+1120)+2372)+1735)-175)*DSQRT((480*
1    EL**4+2400*EL**3+4320*EL**2+3240*EL+810)/(EL**2+EL))/((EL*
2    (EL*(EL*(512*EL+7168)+36352)+78848)+61440))

 DIC(7,1) =
1(EL*((-24*EL-252)*EL-858)-945)*DSQRT((160*EL**5+800*EL**4+1440*EL
1    **3+1080*EL**2+270*EL)/(EL**2+3*EL+2))/((EL*(EL*(256*EL+3072)+
2    12032)+15360))


 DIC(3,6) =
1(EL*(EL*(EL*(EL*(EL*(EL*(EL*(128*EL+2240)+16352)+64752)+151464)+21
1    3988)+178730)+80661)+14715)/((EL*(EL*(EL*(512*EL+5120)+17920)+2
2    5600)+12288))

 DIC(4,5) =
1DSQRT(6*EL+3)*(EL*(EL*(EL*(EL*(EL*((-64*EL-832)*EL-4368)-12272)-
1    20972)-23468)-16499)-5385)/((EL*(EL*(EL*(512*EL+5120)+17920)+
2    25600)+12288))

 DIC(5,4) =
1DSQRT((12*EL+6)/(EL+1))*(EL*(EL*(EL*(EL*(EL*((-64*EL-1280)*EL-
1    10544)-46192)-115804)-165336)-123605)-36975)/((EL*(EL*(EL*
2    (256*EL+3584)+18176)+39424)+30720))

 DIC(6,3) =
1(EL*(EL*(EL*(EL*(16*EL+232)+1300)+3538)+4714)+2485)*DSQRT((480
1    *EL**4+2400*EL**3+4320*EL**2+3240*EL+810)/(EL+1))/((EL*(EL*
2    (EL*(256*EL+3584)+18176)+39424)+30720))

 DIC(7,2) =
1(EL*(EL*(EL*(EL*(EL*(96*EL+2064)+17568)+75384)+169950)+186081)+727
1    65)*DSQRT((40*EL**2+80*EL+30)/(EL**2+3*EL+2))/((EL*(EL*(EL*(512
2    *EL+9216)+60928)+175104)+184320))
```

```
      DIC(8,1) =
     1(EL*((-24*EL-324)*EL-1434)-2079)*DSQRT((2240*EL**6+23520*EL**5+
     1   98560*EL**4+208880*EL**3+233100*EL**2+127750*EL+26250)/(EL+2))
     2   /((EL*(EL*(512*EL+9216)+60928)+175104)+184320))

      DO 70 NI = 4,NMAXA(LI)

      N = NI-1

      DIC(NI,6) =
     1(N*(N*(N*(N*(N*(N*(N*(N*(N*(96*N+960*EL-480)+EL*(4384*EL-3632)+672
     1   )+EL*(EL*(12032*EL-12032)+2624)+192)+EL*(EL*(EL*(22048*EL-22432
     2   )+32)+136)-4026)+EL*(EL*(EL*(EL*(28352*EL-25120)-18528)-3760)-1
     3   0900)+9390)+EL*(EL*(EL*(EL*(EL*(26080*EL-16432)-48640)-16104)+1
     4   926)+38413)-2097)+EL*(EL*(EL*(EL*(EL*(EL*(17024*EL-5120)-62944)
     5   -29696)+42232)+68288)-16618)-10272)+EL*(EL*(EL*(EL*(EL*(EL*(EL*
     6   (7552*EL+256)-46464)-28432)+65608)+69752)-32292)-21489)+5085)+E
     7   L*(EL*(EL*(EL*(EL*(EL*(EL*(2048*EL+640)-18816)-13920)+42976
     8   )+41656)-27816)-21458)+7626)+1440)+EL*(EL*(EL*(EL*(EL*(EL*(
     9   EL*(EL*(256*EL+128)-3264)-2784)+10800)+12024)-7684)-11258)-864)
     :   +2106)+540)/((N*(N*(N*(N*(N*(1024*N+6144*EL-3072)+EL*(15360*EL-
     ;   15360)-5120)+EL*(EL*(20480*EL-30720)-20480)+15360)+EL*(EL*(EL*(
     <   15360*EL-30720)-30720)+46080)+4096)+EL*(EL*(EL*(EL*(6144*EL-153
     =   60)-20480)+46080)+8192)-12288)+EL*(EL*(EL*(EL*(EL*(1024*EL-3072
     >   )-5120)+15360)+4096)-12288)))

      DIC(NI+1,5) =
     1(N*(N*(N*(N*(N*(N*((-16*N-128*EL)*N+(-480*EL-128)*EL-40)+EL*((-108
     1   8*EL-768)*EL-240))+EL*(EL*((-1680*EL-2176)*EL-472)+736)+431)+EL
     2   *(EL*(EL*((-1856*EL-3584)*EL-288)+2944)+1724))+EL*(EL*(EL*(EL*(
     3   (-1408*EL-3456)*EL+560)+5696)+2800)-1016)-105)+EL*(EL*(EL*(EL*(
     4   EL*((-640*EL-1792)*EL+1056)+5504)+2152)-2032)-210))+EL*(EL*(EL*
     5   (EL*(EL*(EL*((-128*EL-384)*EL+480)+2016)+360)-2088)-838)+402)+1
     6   80)*DSQRT(N**2+2*EL*N+2*EL-1)/((N*(N*(N*(N*(1024*N+5120*EL)+
     7   10240*EL**2-5120)+EL*(10240*EL**2-15360))+EL**2*(5120*EL**2-
     8   15360)+4096)+EL*(EL**2*(1024*EL**2-5120)+4096)))

      DIC(NI+2,4) =
     1(N*(N*(N*(N*(N*(N*((-32*N-256*EL-128)*N+(-928*EL-1040)*EL-64)+EL*(
     1   (-1984*EL-3648)*EL-816)+256)+EL*(EL*((-2720*EL-7136)*EL-3024)+1
     2   976)+1022)+EL*(EL*(EL*((-2432*EL-8384)*EL-5248)+5696)+6200)+146
     3   8)+EL*(EL*(EL*(EL*((-1376*EL-5904)*EL-4800)+7880)+12626)+4047)-
     4   971)+EL*(EL*(EL*(EL*(EL*((-448*EL-2304)*EL-2256)+5296)+10844)+3
     5   656)-3427)-1641)+EL*(EL*(EL*(EL*(EL*(EL*((-64*EL-384)*EL-432)+1
     6   392)+3396)+1104)-2477)-2085)-450)*DSQRT(N**4+(4*EL+2)*N**3+
     6   (4*EL
     7   **2+10*EL-1)*N**2+(12*EL**2+2*EL-2)*N+8*EL**2-4*EL)/((N*(N*(N*(
     8   N*(N*(512*N+3072*EL+1536)+EL*(7680*EL+7680)-2560)+EL*(EL*(10240
     9   *EL+15360)-10240)-7680)+EL*(EL*(EL*(7680*EL+15360)-15360)-23040
     :   )+2048)+EL*(EL*(EL*(EL*(3072*EL+7680)-10240)-23040)+4096)+6144)
     ;   +EL*(EL*(EL*(EL*(EL*(512*EL+1536)-2560)-7680)+2048)+6144)))

   70 CONTINUE

      DO 80 NI = 4,NMAXA(LI)

      N = NI-1

      DIC(NI+3,3) =
     1(N*(N*(N*(N*(N*(48*N+288*EL+288)+EL*(800*EL+1664)+744)+EL*(EL*(128
     1   0*EL+4096)+3872)+1056)+EL*(EL*(EL*(1200*EL+5184)+7304)+3440)+51
     2   )+EL*(EL*(EL*(EL*(608*EL+3296)+5968)+3376)-1146)-1242)+EL*(EL*(
     3   EL*(EL*(EL*(128*EL+832)+1792)+992)-1368)-1836)-540)
     3   *DSQRT(N**6+(
     4   6*EL+6)*N**5+(12*EL**2+36*EL+10)*N**4+(8*EL**3+72*EL**2+64*EL)*
     5   N**3+(48*EL**3+132*EL**2+24*EL-11)*N**2+(88*EL**3+72*EL**2-22*E
     6   L-6)*N+48*EL**3-12*EL)/((N*(N*(N*(N*(2048*N+10240*EL+10240)+EL*
     7   (20480*EL+40960)+10240)+EL*(EL*(20480*EL+61440)+30720)-10240)+E
     8   L*(EL*(EL*(10240*EL+40960)+30720)-20480)-12288)+EL*(EL*(EL*(EL*
     9   (2048*EL+10240)+10240)-10240)-12288)))

      DIC(NI+4,2) =
     1DSQRT(N+1)*DSQRT(N+2)*DSQRT(N+3)*DSQRT(N+4)*DSQRT(N+2*EL-1)
     1   *DSQRT(N+2*EL)*DSQRT(N+2*EL+1)*DSQRT(N+2*EL+2)
     1   *(N*(N*(N*(N*(N*(32*N+192*EL+288
     2   )+EL*(480*EL+1456)+832)+EL*(EL*(640*EL+2944)+3424)+672)+EL*(EL*
     3   (EL*(480*EL+2976)+5280)+2216)-702)+EL*(EL*(EL*(EL*(192*EL+1504)
     4   +3616)+2416)-1236)-1242)+EL*(EL*(EL*(EL*(EL*(32*EL+304)+928)+87
     5   2)-534)-1197)-405)/((N*(N*(N*(N*(N*(2048*N+12288*EL+18432)+EL*(
     6   30720*EL+92160)+51200)+EL*(EL*(40960*EL+184320)+204800)+30720)+
```

```
     7   EL*(EL*(EL*(30720*EL+184320)+307200)+92160)-53248)+EL*(EL*(EL*(
     8   EL*(12288*EL+92160)+204800)+92160)-106496)-49152)+EL*(EL*(EL*(E
     9   L*(EL*(2048*EL+18432)+51200)+30720)-53248)-49152)))

      DIC(NI+5,1) =
     1DSQRT(N+1)*DSQRT(N+2)*DSQRT(N+3)*DSQRT(N+4)*DSQRT(N+5)
     1   *DSQRT(N+2*EL-1)*DSQRT(N+2*EL)*DSQRT(N+2*EL+1)
     1   *DSQRT(N+2*EL+2)*DSQRT(N+2*EL+3)*(N*(N*
     2   ((-16*N-64*EL-128)*N+(-96*EL-384)*EL-344)+EL*((-64*EL-384)*EL-6
     3   88)-352)+EL*(EL*((-16*EL-128)*EL-344)-352)-105)/((N*(N*(N*(N*(2
     4   048*N+10240*EL+20480)+EL*(20480*EL+81920)+71680)+EL*(EL*(20480*
     5   EL+122880)+215040)+102400)+EL*(EL*(EL*(10240*EL+81920)+215040)+
     6   204800)+49152)+EL*(EL*(EL*(EL*(2048*EL+20480)+71680)+102400)+49
     7   152)))

   80 CONTINUE

C convert double precision matrix to single precision.
      DO 20 NI = 1,NMAX
        SIC(NI,1) = DIC(NI,1)
        SIC(NI,2) = DIC(NI,2)
        SIC(NI,3) = DIC(NI,3)
        SIC(NI,4) = DIC(NI,4)
        SIC(NI,5) = DIC(NI,5)
        SIC(NI,6) = DIC(NI,6)
        IF (LI.EQ.1) THEN
        END IF
   20 CONTINUE

      RETURN
      END

C -----------------------------------------------------------------
C
      SUBROUTINE FORMRHS(LI)
C
*CALL VORT4

      L = LI
      RL = REAL(L)
      CL1 = (2.*RL + 1.) / (2.*RL* (RL + 1.))
      R1 = RCONST

C Form RHS vector.
      DO 150 NIP=1,NMAX
        SUM = 0.
        DO 160 I=1,NCOL
C         Test function:
          TESTF = R1**4 * GVD(NIP,LI,I) * (1-XI(I))**((RL-3)/2)
     1                   * XI(I)**(RL/2) /2
C         an inner loop begins here for the double integration
          SUM1 = 0.
C         Weight function of quadrature method:
          WFth = 1.
          DO 165 J = 1,LCOL
            SUM1 = SUM1 + VOR(I,J)*P1(LI,J)*WITH(J)/WFth
  165     CONTINUE
C         weight function of quadrature method
          SUM = SUM + SUM1*TESTF*WIR(I)
  160   CONTINUE
        RHS(NIP) = SUM * CL1
  150 CONTINUE

      RETURN
      END

C -----------------------------------------------------------------
C
      SUBROUTINE OUTPUT
C
*CALL VORT4
C
      WRITE (LDWRITE,1002)
      WRITE (LDWRITE,*) TIME,0.
      WRITE (LDWRITE,1000)
      WRITE (LDWRITE,*) RCONSTD,RADIUS,TIME
      WRITE (LDWRITE,1007)
      WRITE (LDWRITE,*) VISC, GAM, CtR
      WRITE (LDWRITE,1001)
      WRITE (LDWRITE,*) IRUN, IFRAME
      WRITE (IDWRITE,1005)
      WRITE (IDWRITE,*) NMAXA(1),NCOL,IMAX,LCOL

      WRITE (IDWRITE,1009)
      DO 5 LI=1,IMAX
        WRITE (IDWRITE,*) LI,NMAXA(LI)
5     CONTINUE

      WRITE (IDWRITE,1010)
      DO 10 I=1,NCOL
        WRITE (IDWRITE,*) I,XIDD(I),WIR(I)
10    CONTINUE

      WRITE (IDWRITE,1020)
      DO 20 J=1,LCOL
        WRITE (IDWRITE,*) J,CTH(J),WITH(J)
20    CONTINUE

      WRITE (IDWRITE,1030)
      DO 30 LI=1,IMAX
        DO 40 NI=1,NMAXA(LI)
          WRITE (IDWRITE,*) LI,NI,ALP(NI,LI)
40      CONTINUE
30    CONTINUE

1002 FORMAT (5X,'TIME',13X,'ZCENT')
1000 FORMAT (5X,'RCONST',10X,'RADIUS',10X,'TIMEo')
1001 FORMAT (5X,'IRUN',10X,'IFRAME')
1005 FORMAT (9X,'NMAX',8X,'NCOL',8X,'IMAX',8X,'LCOL')
1009 FORMAT (5X,'LINDEX',10X,'NMAX(L)')
1007 FORMAT ( 5X,'VISC',12X,'GAM',12X,'Core to Ring radius')
1010 FORMAT ( 10X,'I',7X,'XI(I)',10X,'WIR(I)')
1020 FORMAT ( 10X,'J',7X,'CTH(J)',10X,'WITH(J)')
1030 FORMAT ( 9X,'LI',10X,'NI',7X,'ALP(NI,LI)')
C
      RETURN
      END
C
C ------------------------------------------------------------------

      SUBROUTINE QUAD(N,WTS,XIS,D)

C This routine calculates the weights and absisscas for orthogonal polynomia
c quadriture routines.  It can be used for any orthogonal polynomials by
c replacing the coefficients AK1,AK2,AK3,AK4 with the appropriate ones from
c the 3 term recurrance relation (p. 782 Abracmowitz and Stegun)
C 10/11/86: modified for p=1, q=1/2.  check case worked to all signif. figur
C
      LOGICAL WANTX
      REAL XIS(90),WTS(90)
      DOUBLE PRECISION D(90),B(90),X(90,90),WT(90),A1K,A2K,
     1 A3K,A4K,F,E,WTI,XT,XN,TEMP,PC,QC,PI
C
      PI = 3.14159265358979323846
C
      PC = 1.0
      QC = 0.5
      D(1) = QC/(PC+1.)
      I=1
      K=I
      F=1.0
      TEMP=(2.*K+PC-2.)*(2.*K+PC-1.)*(2.*K+PC)
      A3K=TEMP*(2.*K+PC+1.)*(2.*K+PC-1.)
      A4K=K*(K+QC-1.)*(K+PC-1.)*(K+PC-QC)*(2.*K+PC+1.)
      E=A4K/A3K
      B(I+1)=DSQRT(F*E)
C
      DO 100 I=2,N
        K=I-1
C input the following coefficients for a given polynomial
c Ref. Abramowitz and Stegen  p. 782
C ______replace coefficients here__________________
        TEMP=(2.*K+PC-2.)*(2.*K+PC-1.)*(2.*K+PC)
        A1K=TEMP*(2.*K+PC+1.)*(2.*K+PC-1.)
        A2K=-(2.*K*(K+PC)+QC*(PC-1.))*TEMP
        A3K=TEMP*(2.*K+PC+1.)*(2.*K+PC-1.)
        A4K=K*(K+QC-1.)*(K+PC-1.)*(K+PC-QC)*(2.*K+PC+1.)
C ---------end replace coeff---------------------------
C
C Combine coeffieclents t be in the recurrance form:
```

```
c P(k+1)= ( f(k+1) *x +d(k+1) )*P(k) - e(k+1)*P(k-1)
c
c D(I) : diagonal of symmetric tridiagonal matrix
        D(I)=-A2K/A3K
        IF(I.EQ.N) GOTO 100
C
C         F=A1K/A3K
        F=1.0
C
        K=I
C ______replace coefficients here____________________
        TEMP=(2.*K+PC-2.)*(2.*K+PC-1.)*(2.*K+PC)
        A3K=TEMP*(2.*K+PC+1.)*(2.*K+PC-1.)
        A4K=K*(K+QC-1.)*(K+PC-1.)*(K+PC-QC)*(2.*K+PC+1.)
C ---------end replace coeff---------------------------
c
        E=A4K/A3K
        B(I+1)=DSQRT(F*E)
C
 100    CONTINUE
C
c Find eigenvalues and eigenvectors
      WANTX= .TRUE.
      NDIM=90
      CALL TRIEIGD(NDIM,N,D,B,WANTX,X)
C
C -------------note that this is also particular to the polynomial------
c CALCULATE MUo = integral from a to b of the weight function
c          where a and b are the appropriate limits for the polynomial
      WTI=PI/2.
C ---------------------------------------------------------------------
C
C Normalize eigenvectors
      DO 20 J=1,N
        XN=0.0
        DO 30  I=1,N
          XT=X(I,J)
          XN=XT*XT+XN
 30     CONTINUE
        X(1,J)=X(1,J)/XN
 20   CONTINUE
C
C Calculate weights
      DO 40 J=1,N
        XT=X(1,J)
        WT(J)=XT*XT*WTI
 40   CONTINUE
C
C  Arrange Points In Asscending Order.
      DO 200 M=N,2,-1
        DO 210 I=2,M
          IF(D(I-1).LT.D(I)) GO TO 210
          TEMP=D(I-1)
          D(I-1)=D(I)
          D(I)=TEMP
          TEMP=WT(I-1)
          WT(I-1)=WT(I)
          WT(I)=TEMP
 210     CONTINUE
 200   CONTINUE
C
      DO 220 J=1,N
        WTS(J)=WT(J)
        XIS(J)=D(J)
 220   CONTINUE
C
      RETURN
      END
C ---------------------------------------------------------------
C ------------------------------------------------------------
      SUBROUTINE QUADAZ(N,WTS,DS)
C This routine calculates the weights and absisscas for orthogonal polynomia
c quadriture routines.  It can be used for any orthogonal polynomials by
c replacing the coefficients AK1,AK2,AK3,AK4 with the appropriate ones from
c the 3 term recurrance relation (p. 782 Abracmowitz and Stegun)
      LOGICAL WANTX
      REAL WTS(139),DS(139)
      REAL D(139),B(139),X(139,139),WT(139),
     1  A1K,A2K,A3K,A4K,F,E,WTI,XT,XN,AMU
C
```

```
      AMU = 0.
      DO 100 I=1,N
        K=I-1
C input the following coefficients for a given polynomial
c Ref. Abromowitz and Stegen  p. 782
C ______replace coefficients here__________________
        A1K=K-AMU+1.
        A2K=0.
        A3K=2.*K+1.
        A4K=K+AMU
C ---------end replace coeff------------------------
C
C Combine coeffieciemts t be in the recurrance form:
c P(k+1)= ( f(k+1) *x +d(k+1) )*P(k) - e(k+1)*P(k-1)
c
c D(I) : diagonal of symmetric tridiagonal matrix
        D(I)=-A2K/A3K
        IF(I.EQ.N) GOTO 100
C
        F=A1K/A3K
C
        K=I
C ______replace coefficients here__________________
        A3K=2.*K+1
        A4K=K+AMU
C ---------end replace coeff------------------------
c
        E=A4K/A3K
        B(I+1)=SQRT(F*E)
C
 100  CONTINUE
C
c Find eigenvalues and eigenvectors
      WANTX= .TRUE.
      NDIM=139
      CALL TRIEIG(NDIM,N,D,B,WANTX,X)
C
C --------------note that this is also particular to the polynomial------
c CALCULATE MUo = integrai from a to b of the weight function
c          where a and b are the appropriate limits for the polynomial
      WTI=2.
C ------------------------------------------------------------------------
C
C Normalize eigenvectors
      DO 20 J=1,N
        XN=0.0
        DO 30  I=1,N
          XT=X(I,J)
          XN=XT*XT+XN
 30     CONTINUE
        X(1,J)=X(1,J)/XN
 20   CONTINUE
C
C Calculate weights
      DO 40 J=1,N
        XT=X(1,J)
        WT(J)=XT*XT*WTI
 40   CONTINUE
C
C  ARRANGE POINTS IN DESCENDING ORDER.
      DO 200 M=N,2,-1
        DO 210 I=2,M
          IF(D(I-1).GT.D(I)) GO TO 210
          TEMP=D(I-1)
          D(I-1)=D(I)
          D(I)=TEMP
          TEMP=WT(I-1)
          WT(I-1)=WT(I)
          WT(I)=TEMP
210     CONTINUE
200   CONTINUE
C
      DO 220 J=1,N
        WTS(J)=WT(J)
        DS(J)=D(J)
 220  CONTINUE
C
      RETURN
      END
C
```

```
C ---------------------------------------------------------------------
      SUBROUTINE TRIEIG(NDIM,N,D,E,WANTX,X)
      INTEGER NDIM,N
      LOGICAL WANTX
C      DOUBLE PRECISION D(NDIM),E(NDIM),X(NDIM,NDIM)
      REAL D(NDIM),E(NDIM),X(NDIM,NDIM)
C
C Computes Eigenvalues And Eigenvectors Of Real Tridiagonal Symmetric
C Matrix
C
C  NDIM = declared row dimension of A (AND X).
C  N = order of A
C  D = N-VECTOR.  output, eigenvalue
C  E = N-VECTOR.  lower diagonal of symmentric matrix E(2)...E(N)
C  WANTX = .TRUE. if eigenvectors desired, .FALSE. if not.
C  X = N-BY-N MATRIX
C    IF (WANTX) THEN OUTPUT X(*,J) IS EIGENVECTOR ASSOCIATED
C    WITH EIGENVALUE D(J).
C
C      DOUBLE PRECISION ALPHA,BETA,GAMMA,KAPPA,AIJ,T,C,S,F
      REAL ALPHA,BETA,GAMMA,KAPPA,AIJ,T,C,S,F
C      DOUBLE PRECISION DABS,DSQRT
C
C Initialize X As An Idenity Matrix
      DO 101 I=1,N
        DO 102 J=1,N
          X(I,J)=0.0
 102    CONTINUE
        X(I,I)=1.0
 101  CONTINUE
C     TRIDIAGONAL QR ALGORITHM
C     IMPLICIT SHIFT FROM LOWER 2-BY-2
C
   20 DO 27 MB = 2, N
         M = N+2-MB
         MM1 = M-1
         ITER = 0.
         L = 1
   21    E(L) = 0.
C
C        FIND L SUCH THAT E(L) IS NEGLIGIBLE
C
         L = M
C   22    S = DABS(D(L-1)) + DABS(D(L))
   22    S = ABS(D(L-1)) + ABS(D(L))
C         T = S + DABS(E(L))
         T = S + ABS(E(L))
         IF ( T .EQ. S ) GO TO 23
         L = L-1
         IF (L.GE.2) GO TO 22
C
C        IF E(M) IS NEGLIGIBLE, THEN D(M) IS AN EIGENVALUE, SO ...
C
   23    IF (L.EQ.M) GO TO 27
         IF (ITER.GE.30) GO TO 27
         ITER = ITER + 1
C
C        FORM IMPLICIT SHIFT
C
         T = (D(M-1) - D(M))/(E(M) + E(M))
C         S = DSQRT(1. + T*T)
         S = SQRT(1. + T*T)
         IF (T.LT.0.) S = -S
         S = D(M) - E(M)/(T+S)
         E(L) = D(L) - S
         F = E(L+1)
C
C        CHASE NONZERO  F  DOWN MATRIX
C
         DO 26 J = L, MM1
C            T = DABS(E(J)) + DABS(F)
            T = ABS(E(J)) + ABS(F)
C            ALPHA = T*DSQRT((E(J)/T)**2 + (F/T)**2)
            ALPHA = T*SQRT((E(J)/T)**2 + (F/T)**2)
            C = E(J) / ALPHA
            S = F / ALPHA
            BETA = S*(D(J+1) - D(J)) +2.*C*E(J+1)
            E(J) = ALPHA
            E(J+1) = E(J+1) - C*BETA
            T = S*BETA
            D(J) = D(J) + T
            D(J+1) = D(J+1) - T
            IF (J.EQ.MM1) GO TO 24
            F = S*E(J+2)
            E(J+2) = -C*E(J+2)
C
   24       IF (.NOT.WANTX) GO TO 26
            DO 25 I = 1, N
               T = X(I,J)
               X(I,J) = C*T + S*X(I,J+1)
               X(I,J+1) = S*T - C*X(I,J+1)
   25       CONTINUE
C
   26    CONTINUE
         GO TO 21
C
   27 CONTINUE
      RETURN
      END
C ------------------------------------------------------------
      SUBROUTINE TRIEIGD(NDIM,N,D,E,WANTX,X)
      INTEGER NDIM,N
      LOGICAL WANTX
      DOUBLE PRECISION D(NDIM),E(NDIM),X(NDIM,NDIM)
C      REAL D(NDIM),E(NDIM),X(NDIM,NDIM)
C
C Computes Eigenvalues And Eigenvectors Of Real Tridiagonal Symmetric
C Matrix
C
C  NDIM = declared row dimension of A (AND X).
C  N = order of A
C  D = N-VECTOR.  output, eigenvalue
C  E = N-VECTOR.  lower diagonal of symmentric matrix E(2)...E(N)
C  WANTX = .TRUE. if eigenvectors desired, .FALSE. if not.
C  X = N-BY-N MATRIX
C    IF (WANTX) THEN OUTPUT X(*,J) IS EIGENVECTOR ASSOCIATED
C    WITH EIGENVALUE D(J).
C
      DOUBLE PRECISION ALPHA,BETA,GAMMA,KAPPA,AIJ,T,C,S,F
C      REAL ALPHA,BETA,GAMMA,KAPPA,AIJ,T,C,S,F
      DOUBLE PRECISION DABS,DSQRT
C
C Initialize X As An Idenity Matrix
      DO 101 I=1,N
        DO 102 J=1,N
          X(I,J)=0.0
 102    CONTINUE
        X(I,I)=1.0
 101  CONTINUE
C     TRIDIAGONAL QR ALGORITHM
C     IMPLICIT SHIFT FROM LOWER 2-BY-2
C
   20 DO 27 MB = 2, N
         M = N+2-MB
         MM1 = M-1
         ITER = 0.
         L = 1
   21    E(L) = 0.
C
C        FIND L SUCH THAT E(L) IS NEGLIGIBLE
C
         L = M
   22    S = DABS(D(L-1)) + DABS(D(L))
C   22    S = ABS(D(L-1)) + ABS(D(L))
C         T = S + DABS(E(L))
         T = S + ABS(E(L))
         IF ( T .EQ. S ) GO TO 23
         L = L-1
         IF (L.GE.2) GO TO 22
C
C        IF E(M) IS NEGLIGIBLE, THEN D(M) IS AN EIGENVALUE, SO ...
C
   23    IF (L.EQ.M) GO TO 27
         IF (ITER.GE.30) GO TO 27
         ITER = ITER + 1
C
C        FORM IMPLICIT SHIFT
C
         T = (D(M-1) - D(M))/(E(M) + E(M))
         S = DSQRT(1. + T*T)
C         S = SQRT(1. + T*T)
         IF (T.LT.0.) S = -S
         S = D(M) - E(M)/(T+S)
         E(L) = D(L) - S
         F = E(L+1)
C
C        CHASE NONZERO  F  DOWN MATRIX
C
         DO 26 J = L, MM1
            T = DABS(E(J)) + DABS(F)
C            T = ABS(E(J)) + ABS(F)
            ALPHA = T*DSQRT((E(J)/T)**2 + (F/T)**2)
C            ALPHA = T*SQRT((E(J)/T)**2 + (F/T)**2)
            C = E(J) / ALPHA
            S = F / ALPHA
            BETA = S*(D(J+1) - D(J)) +2.*C*E(J+1)
            E(J) = ALPHA
            E(J+1) = E(J+1) - C*BETA
            T = S*BETA
            D(J) = D(J) + T
            D(J+1) = D(J+1) - T
            IF (J.EQ.MM1) GO TO 24
            F = S*E(J+2)
            E(J+2) = -C*E(J+2)
C
   24       IF (.NOT.WANTX) GO TO 26
            DO 25 I = 1, N
               T = X(I,J)
               X(I,J) = C*T + S*X(I,J+1)
               X(I,J+1) = S*T - C*X(I,J+1)
   25       CONTINUE
C
   26    CONTINUE
         GO TO 21
C
   27 CONTINUE
      RETURN
      END
```

Appendix I
Navier-Stokes Code

In this appendix, the Navier-Stokes code is described and the listing is included. We begin by deriving the working equations for the axisymmetric Navier-Stokes calculation. Next, a brief flowchart outlines the structure of the code by showing the first level of subroutines. A detailed flowchart follows of the algorithm used in transforming to and from coefficient space (in $O(N^3)$ operations). Next, we include a FLOP (floating point operations) trace showing the breakdown of the speed and percent of total time spent in each subroutine. Finally, the code is listed.

To get the working equations, we start with the weighted residual equation, (2.2.1). As described in section 2.2, the term involving the gradient of the pressure drops because of the divergence-free expansion and boundary conditions. The weighted residual equation therefore reduces to (2.2.3). Next, we substitute the axisymmetric velocity expansion, (3.2.1), into (2.2.3) and take the time-dependent coefficients outside of the integrals. The result is a set of $N \times N$ coupled ordinary differential equations for each ℓ:

$$A^{+}_{n'n}(\ell)\frac{da^{+}_{n\ell m}}{dt} - \nu B^{+}_{n'n}(\ell)\ a^{+}_{n\ell m} =< \mathbf{u} \times \omega, \nabla \times (f^{+}_{n'\ell}\mathbf{X}^{*}_{\ell m}) > \qquad (I.1)$$

Differencing equation (I.1) in time then gives

$$A^{\dagger}_{n'n}(\ell)\,\Delta a^{j}_{n\ell} = B^{\dagger}_{n'n}(\ell) a^{j}_{n\ell} - (3Q^{j}_{n'\ell} - Q^{j-1}_{n'\ell})\,\Delta t \qquad (I.2)$$

The elements of the matrices $A^{+}_{n'n}(\ell)$ and $B^{+}_{n'n}(\ell)$ were computed analytically using MACSYMA, output as FORTRAN statements, and pasted directly into the code.

At the start of a calculation, these non-zero matrix elements are computed once (in double precision) and stored (in single precision). From this, the differenced form of the matrices, $A^{\dagger}_{n'n}(\ell)$ and $B^{\dagger}_{n'n}(\ell)$, are computed and stored in place of $A^{+}_{n'n}(\ell)$ and $B^{+}_{n'n}(\ell)$.

Figure (I.1) illustrates the structure of the Navier-Stokes solver, where the words with bold capital letters denote subroutines. The chart on the left is the general flowchart while the one on the right describes **TIMEINT** in more detail. **INITIAL** reads the coefficients defining the initial vector field obtained from *IC*. Next, **JACOBI** and **LEGENDRE**, solve for the basis functions at the collocation points and store them in arrays. For each index ℓ, the mass matrix (**FORMA**) and viscous matrix (**FORMB**) are evaluated and combined appropriately in **PREINT** for the difference equation. **TIMEINT** then advances the coupled set of ordinary differential equations in time. The flowchart to the right describes the steps taken to advance the solution in **TIMEINT**.

The program marches forward in time for a specified number of time steps, *ncloop*, with the nonlinear term treated as a forcing on the right hand side of the equation and computed pseudospectrally. Several diagnostics such as ring speed, impulse, momentum, etc. are also computed in the **FORCING**, and every so often, data is written to a file in the form of numbers and plots. The advancement of the centroid is computed from the ring speed, the coefficients are advanced in time, and the process repeats.

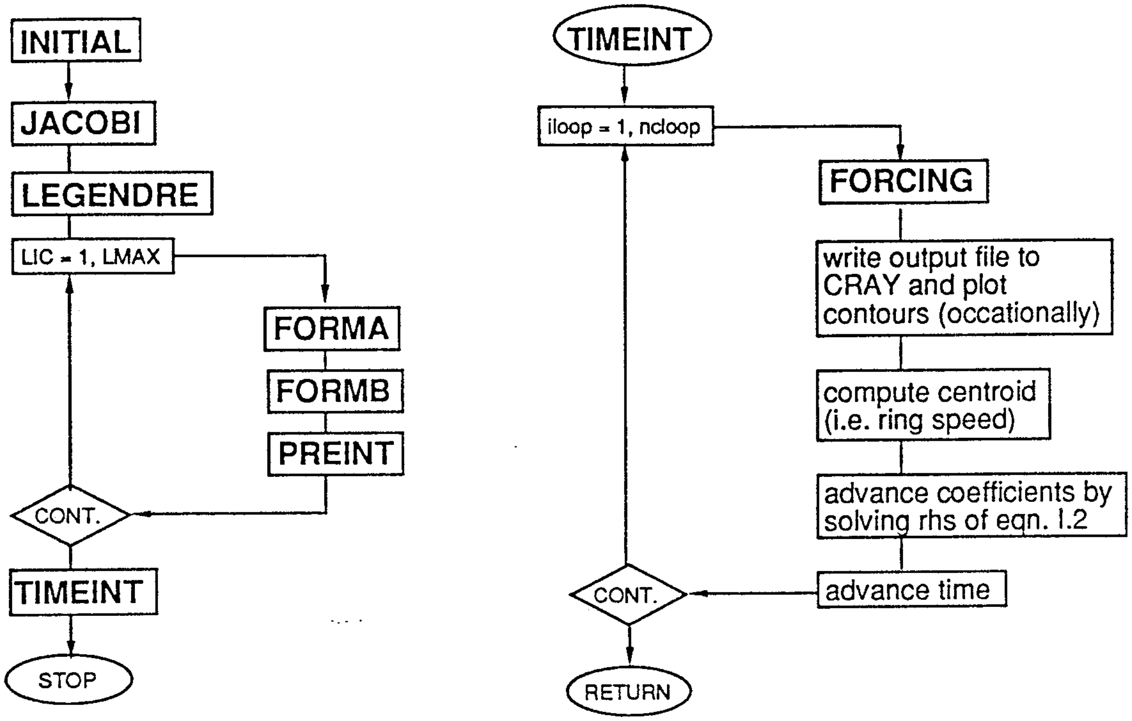

INITIAL
JACOBI
LEGENDRE
LIC = 1, LMAX
FORMA
FORMB
PREINT
CONT.
TIMEINT
STOP
TIMEINT
iloop = 1, ncloop
FORCING
write output file to CRAY and plot contours (occationally)
compute centroid (i.e. ring speed)
advance coefficients by solving rhs of eqn. I.2
advance time
CONT.
RETURN

Figure I.1. Flowchart of the Navier-Stokes solver, NS.

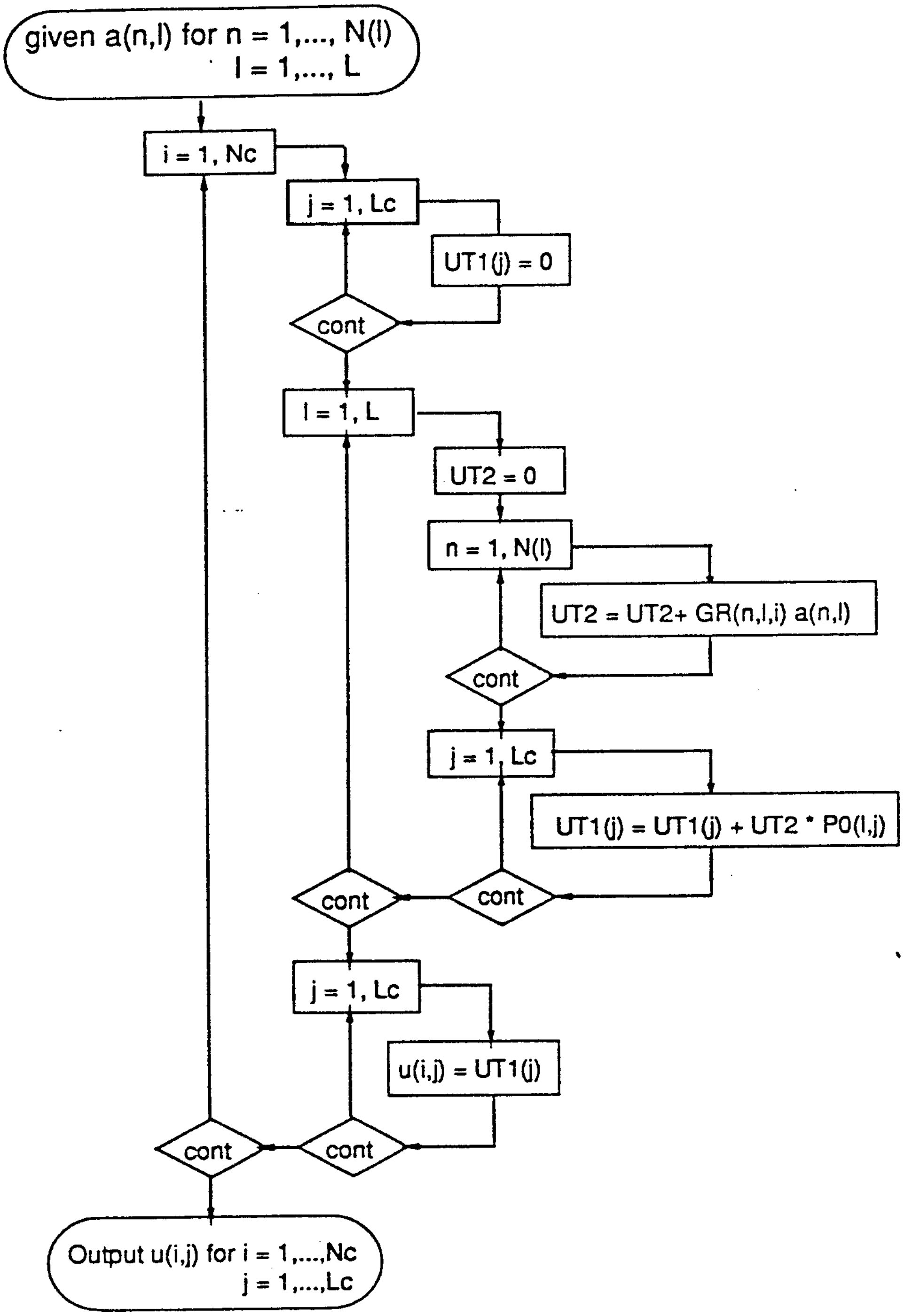
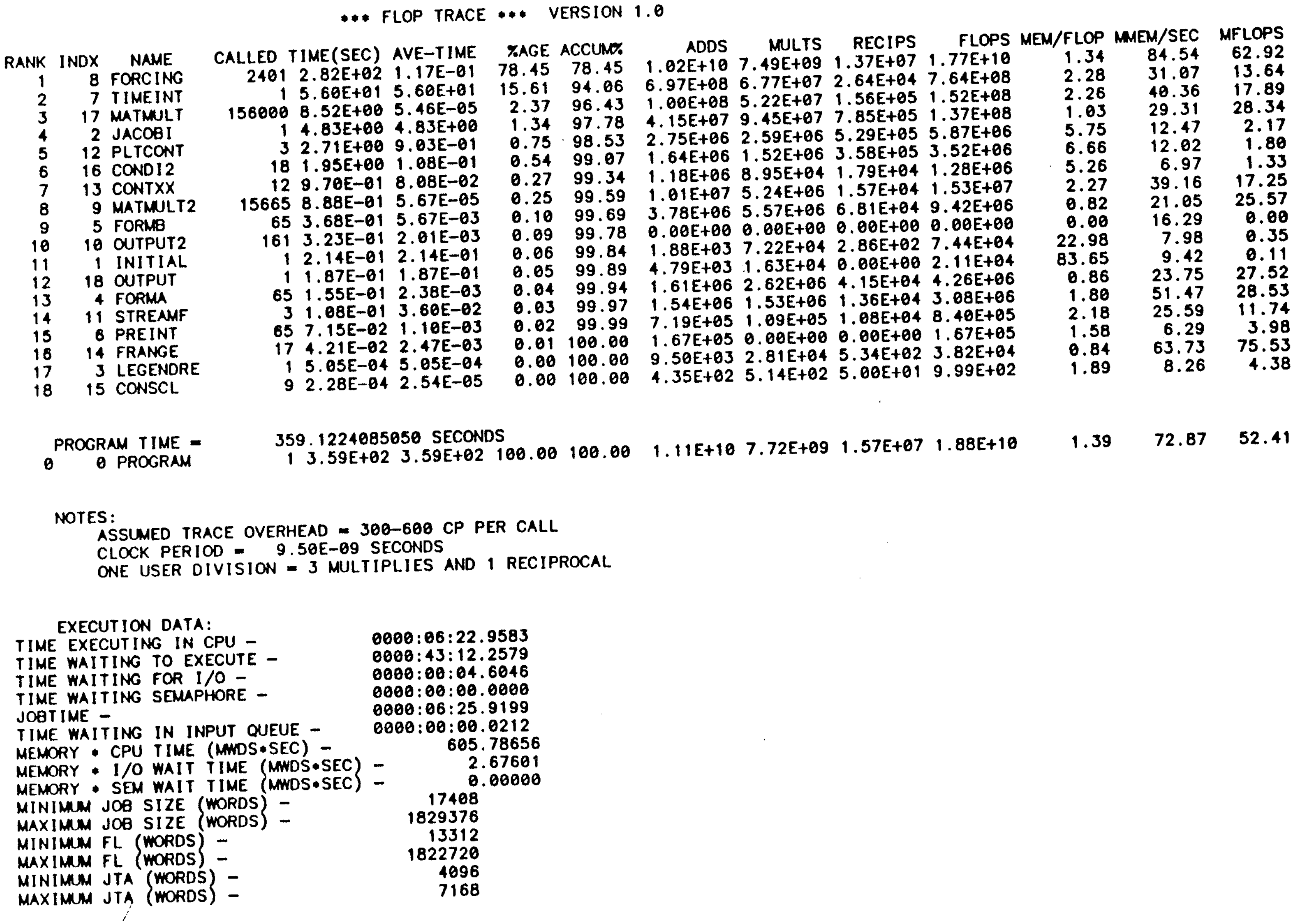


Figure I.2. Flowchart for transforming to and from coefficient space in $O(N^3)$ operations.



```
                              *** FLOP TRACE ***   VERSION 1.0

RANK INDX    NAME       CALLED TIME(SEC) AVE-TIME   %AGE  ACCUM%     ADDS     MULTS    RECIPS     FLOPS MEM/FLOP MMEM/SEC  MFLOPS
   1    8 FORCING         2401 2.82E+02 1.17E-01   78.45   78.45  1.02E+10 7.49E+09 1.37E+07 1.77E+10     1.34    84.54   62.92
   2    7 TIMEINT            1 5.60E+01 5.60E+01   15.61   94.06  6.97E+08 6.77E+07 2.64E+04 7.64E+08     2.28    31.07   13.64
   3   17 MATMULT       156000 8.52E+00 5.46E-05    2.37   96.43  1.00E+08 5.22E+07 1.56E+05 1.52E+08     2.26    40.36   17.89
   4    2 JACOBI             1 4.83E+00 4.83E+00    1.34   97.78  4.15E+07 9.45E+07 7.85E+05 1.37E+08     1.03    29.31   28.34
   5   12 PLTCONT            3 2.71E+00 9.03E-01    0.75   98.53  2.75E+06 2.59E+06 5.29E+05 5.87E+06     5.75    12.47    2.17
   6   16 CONDI2            18 1.95E+00 1.08E-01    0.54   99.07  1.64E+06 1.52E+06 3.58E+05 3.52E+06     6.66    12.02    1.80
   7   13 CONTXX            12 9.70E-01 8.08E-02    0.27   99.34  1.18E+06 8.95E+04 1.79E+04 1.28E+06     5.26     6.97    1.33
   8    9 MATMULT2       15665 8.88E-01 5.67E-05    0.25   99.59  1.01E+07 5.24E+06 1.57E+04 1.53E+07     2.27    39.16   17.25
   9    5 FORMB             65 3.68E-01 5.67E-03    0.10   99.69  3.78E+06 5.57E+06 6.81E+04 9.42E+06     0.82    21.05   25.57
  10   10 OUTPUT2          161 3.23E-01 2.01E-03    0.09   99.78  0.00E+00 0.00E+00 0.00E+00 0.00E+00     0.00    16.29    0.00
  11    1 INITIAL            1 2.14E-01 2.14E-01    0.06   99.84  1.88E+03 7.22E+04 2.86E+02 7.44E+04    22.98     7.98    0.35
  12   18 OUTPUT             1 1.87E-01 1.87E-01    0.05   99.89  4.79E+03 1.63E+04 0.00E+00 2.11E+04    83.65     9.42    0.11
  13    4 FORMA             65 1.55E-01 2.38E-03    0.04   99.94  1.61E+06 2.62E+06 4.15E+04 4.26E+06     0.86    23.75   27.52
  14   11 STREAMF            3 1.08E-01 3.60E-02    0.03   99.97  1.54E+06 1.53E+06 1.36E+04 3.08E+06     1.80    51.47   28.53
  15    6 PREINT            65 7.15E-02 1.10E-03    0.02   99.99  7.19E+05 1.09E+05 1.08E+04 8.40E+05     2.18    25.59   11.74
  16   14 FRANGE            17 4.21E-02 2.47E-03    0.01  100.00  1.67E+05 0.00E+00 0.00E+00 1.67E+05     1.58     6.29    3.98
  17    3 LEGENDRE           1 5.05E-04 5.05E-04    0.00  100.00  9.50E+03 2.81E+04 5.34E+02 3.82E+04     0.84    63.73   75.53
  18   15 CONSCL             9 2.28E-04 2.54E-05    0.00  100.00  4.35E+02 5.14E+02 5.00E+01 9.99E+02     1.89     8.26    4.38


   PROGRAM TIME =        359.1224085050 SECONDS
  0     0 PROGRAM            1 3.59E+02 3.59E+02  100.00  100.00  1.11E+10 7.72E+09 1.57E+07 1.88E+10     1.39    72.87   52.41


   NOTES:
       ASSUMED TRACE OVERHEAD = 300-600 CP PER CALL
       CLOCK PERIOD =     9.50E-09 SECONDS
       ONE USER DIVISION = 3 MULTIPLIES AND 1 RECIPROCAL


    EXECUTION DATA:
TIME EXECUTING IN CPU -               0000:06:22.9583
TIME WAITING TO EXECUTE -             0000:43:12.2579
TIME WAITING FOR I/O -                0000:00:04.6046
TIME WAITING SEMAPHORE -              0000:00:00.0000
JOBTIME -                             0000:06:25.9199
TIME WAITING IN INPUT QUEUE -         0000:00:00.0212
MEMORY * CPU TIME (MWDS*SEC) -               605.78656
MEMORY * I/O WAIT TIME (MWDS*SEC) -            2.67601
MEMORY * SEM WAIT TIME (MWDS*SEC) -            0.00000
MINIMUM JOB SIZE (WORDS) -              17408
MAXIMUM JOB SIZE (WORDS) -            1829376
MINIMUM FL (WORDS) -                    13312
MAXIMUM FL (WORDS) -                  1822720
MINIMUM JTA (WORDS) -                    4096
MAXIMUM JTA (WORDS) -                    7168
```

Figure I.3. FLOP (floating point operations) trace for leapfrogging solution shown in figure (5.11) (actually, 1/3 of the total time).

```
*COMDECK ARRAYS
      PARAMETER( NCD=92, LCD=92 )
      PARAMETER( NMD=92, IMD=92 )
      COMMON/PARM2/ NMAXA(IMD),ALP(NMD,IMD)
C   double precision stuff:
      DOUBLE PRECISION XIDD,RCONSTD
      COMMON/DOUBL/ XIDD(NCD),RCONSTD
C   quantities at collocation points:
      COMMON/COLL/ XI(NCD),CTH(LCD),R(NCD),
     1           UR(NCD,LCD),UTH(NCD,LCD),
     2           OMEG(NCD,LCD),SFUNC(NCD,LCD),SA(LCD),
     3           XY(NCD,LCD,2),F(NCD,LCD)
C   general quantities:
      COMMON/GNRL/ WTR(NCD),WTTH(LCD),STH(LCD)
C   radial basis functions
      COMMON/JAC/ GR(NMD,IMD,NCD),GTH(NMD,IMD,NCD),GV(NMD,IMD,NCD),
     1             RJAC(NCD),FJAC(NCD),F2JAC(NCD)
C   Polar basis functions:
      COMMON/LEGENDRE/ P0(IMD,LCD),P1(IMD,LCD)
C   Impulse calculation:
      COMMON/IMPULSE/ CIMP(NMD)
C   Mass and viscous matrices:
      COMMON/MATRIX/ AMAT(NMD,6), BMAT(NMD,6), BVEC(NMD),
     1               AMATS(NMD,6,IMD), BMATS(NMD,6,IMD),
     2               AMATE(NMD,4,IMD)
C   Energy and dissipation calculations
      COMMON/ENERGY/ DISV(NMD),ENV(NMD)
C Arrays which are local to particular subroutines
      DOUBLE PRECISION G,ROFUNC,FUNC
      COMMON/SJAC/ G(NMD,NCD),ROFUNC(NCD),FUNC(NCD)
      COMMON/SPREINT/ AINV(NMD,6),BTEMP(NMD,6)
      COMMON/STIME/ RHS(NMD),CONVM1(NMD,IMD),DELSOLN(NMD)
C     1      ,DALPOLD(NMD,IMD)
      COMMON/SFORC/ VELRA(LCD),VELTA(LCD),VORTA(LCD),COEF(IMD),
     1     P0FUN(IMD,LCD),RD2(NCD),SUM1A(NCD),SUM2A(NCD),
     2     P1DUM(NMD),P2DUM(NMD),CONV(NMD,IMD),P1FUN(IMD,LCD)
C     3      ,Q(NCD,LCD,4)

*COMDECK PARMS
      PARAMETER( PI=3.14159265358979323846)
      PARAMETER(ICREAD=2,MREAD=5,LWRITE=6,LVAX=7,LGLOBAL=8,LCRAY=9,
     >          LPLOT=10)
      COMMON/PARM/ RADppo,RCONST,TIME,VISC,GAM,NMAX,LMAX,
     >          RVELP,TINC,TSTOP,NCOL,LCOL,CtR,TIMEo,CIRC,RADpp,
     >          ZCoR,RIMPMP,RIMPGQ,DIMP,RVELPP,RMOM,DCENT,ICFLAG
      COMMON/PRINT/ LIC,iRUN,
     >          nrLOOP,iLOOP,
     >          NCRAY,jCRAY,
     >          NGLOBAL,jGLOBAL,
     >          NPLOT,iPLOT,jPLOT,j2PLOT

*DECK NSMAIN
cccccccccccccccc  START OF MAIN PROGRAM  cccccccccccccccc
C
C This is a program to find the evolution of the coefficients in the
C Spectral Stokes equatiion.

c jNAME - indicates a flag, 0 or 1
c nNAME - do NAME every n loops
C nrNAME - indicates the total Number of Records of NAME
c iNAME - indicates the current # of NAME

C NMD - array dimension (> or = NMAX+3)
C NCD - array dimension (> or = NCOL)
C IMD - array dimension (> or = LMAX)
C LCD - array dimsnsion (> or = LCOL)
C COLLOC:  XI(NCD) - radial collocation points
C          WTR(NCD) - weights for Gauss-Jacobi quadrature
C          CTH(LCD) - polar collocation points
C          WTTH(LCD) - weights for Gauss-Legendre quadrature
C          R(NCD) - radial coordinates cooresponding to XI(I)
C JACOBI:  G(NMD,NCD) - Jacobi polynomials
C LEGENDRE:P0(LNDIM,LCD) - Associated Legendre poly. (m=0)
C          P1(LNDIM,LCD) - Associated Legendre poly. (m=1)
C MATRIX:  AMATRIX(NMD,NMD) - A matrix
C          BMATRIX(NMD,NMD) - B matrix
C          RJAC(NCD) - r2 dr/d(psi)
C COEFF:   ALP(NMD,IMD) - coefficients in the expansion
C NMAX - number of terms with index n
C NCOL - number of collocation points in the radial direction
C LMAX - number of terms with index l
C LCOL - number of collocation points in the polar direction
C RADppo - initial radius of vortex ring in computational coordinates
C RCONST - constant which is optimized for clustering around core
C PI - constant pi
C OUTPUT:
C
C        name  unit #   description
C       ------ ----   ---------------
C       ICREAD  2 : initial condition
C       MREAD   5 : time marching parameters
C       LWRITE  6 : debugging stuff
C       LVAX    7 : final solution; sent to VAX
C       LGLOBAL 8 : global quantities at each time step; sent to Cray
C       LCRAY   9 : complete solution at specified times; sent to Cray
C
C ---------------------------------------------------------------------

*CALL PARMS
*CALL ARRAYS

C inputs and initialization:
      CALL INITIAL
      CALL JACOBI
      CALL LEGENDRE

C form matrices:
      DO 10 LIC=1,LMAX
        CALL FORMA
        CALL FORMB
        CALL PREINT
 10   CONTINUE

C integrate:
      CALL TIMEINT
      call donepl

C output:
      CALL OUTPUT

      STOP
      END


C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE INITIAL

*CALL PARMS
*CALL ARRAYS

C initialize flag for PLTCONT — initializing file 15
      jGLOBAL = 0
      jPLOT = 0
      j2PLOT = 0

C Read starting condition
C ------------------------

      READ  (MREAD,1040)
      READ (MREAD,*) ICFLAG
C time step, number of time steps:
      READ  (MREAD,1040)
      READ (MREAD,*)TINC,nrLOOP
C Every __ time steps, plot soln and write global info
      READ  (MREAD,1040)
      READ  (MREAD,1040)
      READ (MREAD,*)NPLOT,NGLOBAL,NCRAY

C Read data from IC.FOR
      READ  (ICREAD,1040)
      READ  (ICREAD,*) TIME,ZCoR
      READ  (ICREAD,1040)
      READ  (ICREAD,*) RCONSTD,RADppo,TIMEo
      READ  (ICREAD,1040)
      READ  (ICREAD,*) VISC, GAM, CtR
      READ  (ICREAD,1040)
      READ  (ICREAD,*) iRUN, iPLOT
      READ  (ICREAD,1040)
      READ  (ICREAD,*) NMAX,NCOL,LMAX,LCOL
      READ (ICREAD,1040)
      DO 5 LI=1,IMAX
        READ (ICREAD,*) II,NMAXA(LI)
5     CONTINUE

      READ (ICREAD,1040)
      DO 10 I=1,NCOL
        READ (ICREAD,*) II,XIDD(I),WIR(I)
10    CONTINUE

      READ (ICREAD,1040)
      DO 20 J=1,LCOL
        READ (ICREAD,*) JJ,CTH(J),WTTH(J)
20    CONTINUE

      READ (ICREAD,1040)
      DO 30 LI=1,IMAX
        DO 40 NI=1,NMAXA(LI)
          READ (ICREAD,*) LL,NN,ALP(NI,LI)
40      CONTINUE
30    CONTINUE

1040 FORMAT (1X)

      RCONST = RCONSID


C compute a few quatities, used throughout the program
C ————————————————————————————————————————————

C Find the corresponding radial values for the collocation points
      DO 50 I=1,NCOL
        XI(I) = XIDD(I)
        XI1 = XI(I)
        R(I) = RCONST*(XI1/(1.-XI1))**(1./2.)
 50   CONTINUE

c compute the coefficients in the multipole expansion for the Impulse

      CFACT = SQRT(2./PI)
      CIMP(1) = 4. * PI * RCONST*RCONST * CFACT
      DO 110 NI = 2,NMAX
        RN = REAL(NI-1)
        DUM = (2.*RN+1.)*CFACT
        CIMP(NI)=DUM*4.*PI*RCONST*RCONST
 110  CONTINUE

      RETURN
      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE JACOBI
C this subtroutine is defined for p = 2*el - 1, q = el - 1/2

*CALL PARMS
*CALL ARRAYS

C G(NI,I) or G(NMD,NCD)
      DOUBLE PRECISION N,EL,DUM,GNORM,XID,R1,VTH1
      REAL ARG1,ARG2,SNORM,GAMMA

      R1 = RCONST


C Evaluate r^2 dr/d(xi):
C ——————————————————


      DO 5 I=1,NCOL
C       r^2 * d(xi)/dr
        RJAC(I) = 0.5*R1*R1*R1 *XI(I)**(1./2.)/ (1.-XI(I))**(5./2.)
c       r^2 * d(xi)/dr / w(x)@l=1
        FJAC(I) = 0.5*R1*R1*R1*XI(I) / (1.-XI(I))**3
c       ( r^2 * d(xi)/dr / w(x)@l=1 ) * WIr-quadrature
        F2JAC(I) = 0.5 * RCONST**3 * WIR(I) *XI(I) / (1.-XI(I))**3
 5    CONTINUE


C Evaluate GR, GTH, GV:
```

```
C ————————————————

      DO 50 LI = 1,IMAX

        EL = LI

C compute the Jacobi Polynomials for all n and xi at a given EL.
C this is only an intermediate result, it is not needed outside of
C this subroutine.
       DO 10 I = 1,NCOL
C       n = 0, NI = 1
        ARG1 = 2*EL-1
        ARG2 = (2*EL-1)/2.0
        SNORM = SQRT(2.)*SQRT(GAMMA(ARG1))/GAMMA(ARG2)
        GNORM = SNORM
        G(1,I) = 1.0d+0
C       n = 1, NI = 2
        XID = XIDD(I)
        G(2,I) = (4*EL*XID-2*EL+1)/DSQRT(2*EL-1)
 10    CONTINUE
C general recurrance formula  - at that particular el.
C Note: if necessary, some memory can be saved by storing only
C ——  5 Gn's at a given time... Gn+2, Gn+1, Gn, Gn-1, Gn-2.

        DO 20 NI = 3,NMAXA(LI)+2
         DO 30 I=1,NCOL

        N = NI-2
        XID = XIDD(I)
        G(NI,I) = (((4*N**2+(8*EL-4)*N+4*EL**2-4*EL)*XID-2*N**2+(2-
     1   4*EL)*N-2*EL**2+3*EL-1)*G(NI-1,I)+(-N*DSQRT(N**2+2*EL*N-2*N)-
     2   EL*DSQRT(N**2+2*EL*N-2*N))*G(NI-2,I))/((N+EL-1)*DSQRT(N**2
     3   +2*EL*N+2*EL-1))

 30      CONTINUE
 20     CONTINUE

        DO 56 I = 1,NCOL
          XID = XIDD(I)
          ROFUNC(I) = R1*(1-XID)**((-1.0d+0)/2.0d+0-(-EL-3)/2.0d+0)*
     1    XID**(1.0d+0/2.0d+0-(2-EL)/2.0d+0)
          FUNC(I) = (1-XID)**((-EL-3)/2.0d+0)*XID**((2-EL)/2.0d+0)
 56     CONTINUE

 C compute the r component of the vorticity basis function
        DO 71 NI = 1,NMAXA(LI)
          N = NI-1
          DO 81 I = 1,NCOL
            GR(NI,LI,I) = GNORM * G(NI,I)*EL*(EL+1)*ROFUNC(I)/R1**2
 81       CONTINUE
 71     CONTINUE

 C compute the theta component of the vorticity basis function
        DO 62 I = 1,NCOL
          XID = XIDD(I)
          VTH1 = XID**(EL/2.0d+0+(-1.0d+0)/2.0d+0)*(XID*((2*EL+1)*
     1     (1-XID)**(EL/2.0d+0)*XID+(-3*EL-2)*(1-XID)**(EL/2.0d+0))+
     2     (EL+1)*(1-XID)**(EL/2.0d+0))/R1
          GTH(1,LI,I) = GNORM * VTH1
 62     CONTINUE

        DO 72 NI = 2,NMAXA(LI)
        N = NI-1
          DO 82 I = 1,NCOL
           VTH1 =
     1    (((-2*N-4*EL+1)*N+(1-2*EL)*EL+1)*DSQRT(N**2+2*EL*N+2*EL-1)*
     1    G(NI+1,I)+(N*(4*N+8*EL-4)+EL*(4*EL-3)-1)*G(NI,I)+
     2     (N*(2*N+4*EL-3)+EL*(2*EL-3))*DSQRT(N**2+(2*EL-2)*N)*G(NI-1,I)
     3    /((N*(4*N+8*EL-4)+EL*(4*EL-4))*R1**2)
           GTH(NI,LI,I) = GNORM * VTH1*ROFUNC(I)
 82       CONTINUE
 72     CONTINUE

 C Knowing the Jacobi polynomials, compute (what amounts to) the
 C vorticity basis function.
        DO 60 I = 1,NCOL
          XID = XIDD(I)
          dum = XID*((EL*(4*EL+8)+3)*XID+(-4*EL-8)*EL-3)/R1**2
          GV(1,LI,I) = GNORM * dum/FUNC(I)
          dum = XID*(XID*(EL*(EL*(16*EL+64)+60)*XID+EL*
     1    ((-24*EL-92)*EL-82)+3)+EL*(EL*(8*EL+28)+22)-3)
     2    /(DSQRT(2*EL-1)*R1**2)
        GV(2,LI,I) = GNORM * dum/FUNC(I)
 60   CONTINUE

      DO 70 NI = 3,NMAXA(LI)
      N = NI-1

        DO 80 I = 1,NCOOL
       DUM =
     1((N*(N*(N*(4*N+16*EL-4)+EL*(24*EL-12)-13)+EL*(EL*(16*EL-12)-26)+7)
     1    +EL*(EL*(EL*(4*EL-4)-13)+7)+6)*DSQRT(N**4+(4*EL+2)*N**3+(4*EL**
     2    2+10*EL-1)*N**2+(12*EL**2+2*EL-2)*N+8*EL**2-4*EL)*G(NI+2,I)+(N
     3    *(N*((-8*N-32*EL+12)*N+(32-48*EL)*EL+20)+EL*((28-32*EL)*EL+46)-
     4    18)+EL*(EL*((8-8*EL)*EL+26)-14)-12)*DSQRT(N**2+2*EL*N+2*EL-1)*
     5    G(NI+1,I)+(N*(N*(N*(N*((-8*N-48*EL+24)*N+(88-128*EL)*EL-2)+EL*(
     6    (128-192*EL)*EL+56)-36)+EL*(EL*((96-168*EL)*EL+186)-68)+4)+EL*(
     7    EL*(EL*((40-80*EL)*EL+196)-42)-64)+18)+EL*(EL*(EL*(EL*((8-16*EL
     8    )*EL+68)-10)-64)+2)+12)*G(NI,I)+(N*(N*(N*(8*N+32*EL-20)+EL*(4
     9    8*EL-56)-8)+EL*(EL*(32*EL-52)-18)+26)+EL*(EL*(EL*(8*EL-16)-10)+
     :    20)+6)*DSQRT(N**2+(2*EL-2)*N)*G(NI-1,I)+(N*(N*(N*(4*N+16*EL-12
     ;    )+EL*(24*EL-36)-1)+EL*(EL*(16*EL-36)-2)+15)+EL*(EL*(EL*(4*EL-12
     <    )-1)+15))*DSQRT(N**4+(4*EL-6)*N**3+(4*EL**2-14*EL+11)*N**2+(-4*
     =    EL**2+10*EL-6)*N)*G(NI-2,I))/((N*(N*(N*(16*N+64*EL-32)+EL*(96*
     >    EL-96)-16)+EL*(EL*(64*EL-96)-32)+32)+EL*(EL*(EL*(16*EL-32)-16)+
     ?    32))*R1**2)

C Copy the double precision variable, DUM, to the single precision array
        GV(NI,LI,I) = GNORM * DUM/FUNC(I)

 80     CONTINUE

 70   CONTINUE

 50   CONTINUE

      RETURN
      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE LEGENDRE

*CALL PARMS
*CALL ARRAYS

C Calulate Legendre polynomials, P0(LI,J) and P1(LI,J)
      DO 10 J=1,LCOL
        X1=CTH(J)
        P0(1,J)=X1
        P0(2,J)=(3.*X1*X1-1.)/2.
        SQX2=SQRT(1.-X1*X1)
        P1(1,J)=-SQX2
        P1(2,J)=-3.*X1*SQX2
 10   CONTINUE
      DO 20 LI=3,IMAX
        RL=REAL(LI-1)
        FAC1=(2.*RL+1.)/(RL+1.)
        FAC2=RL/(RL+1.)
        FAC3=(2.*RL+1.)/RL
        DO 30 J=1,LCOL
          X1=CTH(J)
          P0(LI,J)=FAC1*X1*P0(LI-1,J)-FAC2*P0(LI-2,J)
          P1(LI,J)=FAC3*X1*P1(LI-1,J)-P1(LI-2,J)/FAC2
 30     CONTINUE
 20   CONTINUE

      RETURN
      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE FORMA

*CALL PARMS
*CALL ARRAYS

      INTEGER NI,LIC
      DOUBLE PRECISION DSQRT,N,EL,R1
```

```
      R1 = RCONSTD
      EL = LIC

C Form AMAT (initial condition Matrix).
      DO 20 NI = 1,NMAXA(LIC)
        AMAT(NI,1) = 0.
        AMAT(NI,2) = 0.
 20   CONTINUE

      AMAT(1,3) = 0.
      AMAT(1,4) = 0.
      AMAT(1,5) = 0.

      AMAT(2,3) = 0.
      AMAT(2,4) = 0.

      AMAT(3,3) = 0.

      AMAT(1,6) =
     1(EL*(EL*((-16*EL-32)*EL-8)+8)+3)*R1/(EL*(64*EL+64))

      AMAT(2,5) =
     1DSQRT(2*EL-1)*(EL*((-8*EL-20)*EL-14)-3)*R1/(EL*(64*EL+64))

      AMAT(3,4) =
     1DSQRT((2*EL-1)/EL)*(EL*(EL*(8*EL+20)+14)+3)*R1/(EL
     1  *(64*EL+192)+128)

      AMAT(4,3) =
     1(2*EL+3)*DSQRT((48*EL**4+48*EL**3-12*EL-3)/EL)*R1/(EL
     1   *(64*EL+192)+128)

      AMAT(2,6) =
     1(EL*(EL*(EL*((-16*EL-80)*EL-128)-84)-31)-6)*R1/(EL*(EL
     1   *(64*EL+192)+128))

      AMAT(3,5) =
     1(EL*(EL*((-16*EL-64)*EL-72)-16)+3)*R1/(EL*(64*EL**(3.0d+0
     1   /2.0d+0)+192*DSQRT(EL))+128*DSQRT(EL))

      AMAT(4,4) =
     1DSQRT((6*EL+3)/EL)*(EL*(EL*(EL*(8*EL+44)+82)+57)+9)*R1/
     1   (EL*(EL*(64*EL+384)+704)+384)

      AMAT(5,3) =
     1(EL*(4*EL+16)+15)*DSQRT((12*EL**2+6*EL)/(EL+1))*R1/(EL
     1   *(32*EL+160)+192)


      DO 10 NI = 3,NMAXA(LIC)

      N = NI - 1

      AMAT(NI,6) =
     1(N*(N*(N*(N*((-8*N-48*EL+24)*N+(92-128*EL)*EL-14)+(144-192*EL)*EL*
     1   *2-12)+EL*(EL*((124-168*EL)*EL+102)-19)+19)+EL*(EL*(EL*((64-80*
     2   EL)*EL+144)-28)-7)-9)+EL*(EL*(EL*(EL*((16-16*EL)*EL+56)-32)-37)
     3   +7)+6)*R1/(N*(N*(N*(64*N+256*EL-128)+EL*(384*EL-384)-64)+EL*(EL
     4   *(256*EL-384)-128)+128)+EL*(EL*(EL*(64*EL-128)-64)+128))

      AMAT(NI+1,5) =
     1(N*(N*((-4*N-16*EL)*N+(-36*EL-24)*EL+1)+EL*((-40*EL-48)*EL+2))+EL*
     1    (EL*((-16*EL-24)*EL+12)+22)+6)*DSQRT(N**2+2*EL*N+2*EL-1)
     2   *R1/(N*(N*(128*N+384*EL)+384*EL**2-128)+EL*(128*EL**2-128))


      AMAT(NI+2,4) =
     1(N*(N*(N*(8*N+32*EL+16)+EL*(48*EL+44)-2)+EL*(EL*(32*EL+40)-8)-10)+
     1   EL*(EL*(EL*(8*EL+12)-6)-11)-3)
     1   *DSQRT(N**4+(4*EL+2)*N**3+(4*EL**2
     2   +10*EL-1)*N**2+(12*EL**2+2*EL-2)*N+8*EL**2-4*EL)*R1/(N*(N*(N*(1
     3   28*N+512*EL+256)+EL*(768*EL+768)-128)+EL*(EL*(512*EL+768)-256)-
     4   256)+EL*(EL*(EL*(128*EL+256)-128)-256))


      AMAT(NI+3,3) =
     1(N*(4*N+8*EL+8)+EL*(4*EL+8)+3)
     1   *DSQRT(N**6+(6*EL+6)*N**5+(12*EL**2+3
     1   6*EL+10)*N**4+(8*EL**3+72*EL**2+64*EL)*N**3+(48*EL**3+132*EL**2
     2   +24*EL-11)*N**2+(88*EL**3+72*EL**2-22*EL-6)*N+48*EL**3-12*EL)*R
     3    1/(N*(N*(128*N+384*EL+384)+EL*(384*EL+768)+256)+EL*(EL*(128*EL+
     4    384)+256))

 10   CONTINUE

      DO 25 NI = 1, NMAXA(LIC)
 25     AMATE(NI,1,LIC) = AMAT(NI,6)

      DO 30 NI = 1, NMAXA(LIC) - 1
 30     AMATE(NI,2,LIC) = AMAT(NI+1,5)

      DO 40 NI = 1, NMAXA(LIC) - 2
 40     AMATE(NI,3,LIC) = AMAT(NI+2,4)

      DO 50 NI = 1, NMAXA(LIC) - 3
 50     AMATE(NI,4,LIC) = AMAT(NI+3,3)


      RETURN
      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE FORMB

*CALL PARMS
*CALL ARRAYS

      INTEGER NI,LIC
      DOUBLE PRECISION DSQRT,N,EL,R1

      EL = LIC
      R1 = RCONSID

      bmat(1,1) =
     1(EL*(EL*(EL*(EL*(EL*(EL*(128*EL+832)+1952)+1936)+536)-356)-258)-
     1    45)/(EL*(EL*(512*EL+1536)+1024)*R1)

      bmat(1,2) =
     1(EL*(EL*(EL*((-32*EL-208)*EL-496)-536)-258)-45)*DSQRT(8*EL**3
     1    -12*EL**2+6*EL-1)/(EL*(EL*(512*EL+1536)+1024)*R1)

      bmat(1,3) =
     1DSQRT((2*EL-1)/EL)*(EL*(EL*(EL*(EL*((-64*EL-576)*EL-2032)-3552)-
     1    3196)-1380)-225)/((EL*(EL*(256*EL+1536)+2816)+1536)*R1)

      bmat(1,4) =
     1(EL*(EL*(EL*(16*EL+112)+288)+324)+135)*DSQRT((48*EL**4+48*EL**3-
     1    12*EL-3)/EL)/((EL*(EL*(256*EL+1536)+2816)+1536)*R1)

      bmat(1,5) =
     1(EL*(EL*(EL*(16*EL+160)+552)+792)+405)*DSQRT((96*EL**4+96*EL**3-
     1    24*EL-6)/(EL**2+EL))/((EL*(EL*(512*EL+4608)+13312)+12288)*R1)

      bmat(1,6) =
     1((-4*EL-24)*EL-35)*DSQRT((3840*EL**7+21120*EL**6+43200*EL**5+
     1    37920*EL**4+8400*EL**3-7560*EL**2-4860*EL-810)/(EL**2+EL))
     2    /((EL*(EL*(512*EL+4608)+13312)+12288)*R1)

      bmat(2,1) =
     1(EL*(EL*(EL*(EL*(EL*(EL*(EL*(128*EL+1472)+6624)+14768)+16744)+
     1    8468)+1002)+9)+135)/(EL*(EL*(EL*(512*EL+3072)+5632)+3072)*R1)

      bmat(2,2) =
     1(EL*(EL*(EL*(EL*(EL*((-64*EL-576)*EL-1936)-3056)-2476)-1372)-
     1    795)-225)/((EL*(EL*(256*EL**(3.0d+0/2.0d+0)+1536*DSQRT(EL))+
     2    2816*DSQRT(EL))+1536*DSQRT(EL))*R1)

      bmat(2,3) =
     1DSQRT((6*EL+3)/EL)*(EL*(EL*(EL*(EL*(EL*((-64*EL-896)*EL-5008)-
     1    14208)-21420)-15960)-4059)+540)/((EL*(EL*(EL*(256*EL+2560)
     2    +8960)+12800)+6144)*R1)

      bmat(2,4) =
     1(EL*(EL*(EL*(EL*(EL*(128*EL+1440)+6288)+13424)+14208)+6282)
     1    +405)*DSQRT((12*EL+6)/(EL**2+EL))/((EL*(EL*(512*EL+4608)
     2    +13312)+12288)*R1)

      bmat(2,5) =
     1(EL*(EL*(EL*(EL*(16*EL+224)+1120)+2372)+1735)-175)*DSQRT(
     1    (480*EL**4+2400*EL**3+4320*EL**2+3240*EL+810)/(EL**2+EL))
     2    /((EL*(EL*(EL*(512*EL+7168)+36352)+78848)+61440)*R1)

      bmat(2,6) =
     1(EL*((-24*EL-252)*EL-858)-945)*DSQRT((160*EL**5+800*EL**4+
     1    1440*EL**3+1080*EL**2+270*EL)/(EL**2+3*EL+2))/((EL*(EL
     2    *(256*EL+3072)+12032)+15360)*R1)

      bmat(3,1) =
     1(EL*(EL*(EL*(EL*(EL*(EL*(EL*(128*EL+2240)+16352)+64752)+151464)
     1    +213988)+178730)+80661)+14715)/((EL*(EL*(EL*(512*EL+5120)
     2    +17920)+25600)+12288)*R1)

      bmat(3,2) =
     1DSQRT(6*EL+3)*(EL*(EL*(EL*(EL*(EL*((-64*EL-832)*EL-4368)-12272)
     1    -20972)-23468)-16499)-5385)/((EL*(EL*(EL*(512*EL+5120)+17920)
     2    +25600)+12288)*R1)

      bmat(3,3) =
     1DSQRT((12*EL+6)/(EL+1))*(EL*(EL*(EL*(EL*(EL*((-64*EL-1280)*EL
     1    -10544)-46192)-115804)-165336)-123605)-36975)/((EL*(EL*(EL*
     2    (256*EL+3584)+18176)+39424)+30720)*R1)

      bmat(3,4) =
     1(EL*(EL*(EL*(EL*(16*EL+232)+1300)+3538)+4714)+2485)*DSQRT((480
     1    *EL**4+2400*EL**3+4320*EL**2+3240*EL+810)/(EL+1))/((EL*(EL*
     2    (EL*(256*EL+3584)+18176)+39424)+30720)*R1)

      bmat(3,5) =
     1(EL*(EL*(EL*(EL*(EL*(96*EL+2064)+17568)+75384)+169950)+186081)+
     1    72765)*DSQRT((40*EL**2+80*EL+30)/(EL**2+3*EL+2))/((EL*(EL
     2    *(EL*(512*EL+9216)+60928)+175104)+184320)*R1)

      bmat(3,6) =
     1(EL*((-24*EL-324)*EL-1434)-2079)*DSQRT((2240*EL**6+23520*EL**5+
     1    98560*EL**4+208880*EL**3+233100*EL**2+127750*EL+26250)/(EL+2))
     2    /((EL*(EL*(EL*(512*EL+9216)+60928)+175104)+184320)*R1)


      DO 10 NI = 4,NMAXA(LIC)

      N = NI - 1

      bmat(NI,1) =
     1(N*(N*(N*(N*(N*(N*(N*(N*(N*(96*N+960*EL-480)+EL*(4384*EL-3632)+672
     1    )+EL*(EL*(12032*EL-12032)+2624)+192)+EL*(EL*(EL*(22048*EL-22432
     2    )+32)+136)-4026)+EL*(EL*(EL*(EL*(28352*EL-25120)-18528)-3760)-1
     3    0900)+9390)+EL*(EL*(EL*(EL*(EL*(26080*EL-16432)-48640)-16104)+1
     4    926)+38413)-2097)+EL*(EL*(EL*(EL*(EL*(EL*(17024*EL-5120)-62944)
     5    -29696)+42232)+68288)-16618)-10272)+EL*(EL*(EL*(EL*(EL*(EL*(EL*
     6    (7552*EL+256)-46464)-28432)+65608)+69752)-32292)-21489)+5085)+E
     7    L*(EL*(EL*(EL*(EL*(EL*(EL*(EL*(2048*EL+640)-18816)-13920)+42976
     8    )+41656)-27816)-21458)+7626)+1440)+EL*(EL*(EL*(EL*(EL*(EL*(EL*(
     9    EL*(EL*(256*EL+128)-3264)-2784)+10800)+12024)-7684)-11258)-864)
     :    +2106)+540)/((N*(N*(N*(N*(N*(1024*N+6144*EL-3072)+EL*(15360*EL-
     ;    15360)-5120)+EL*(EL*(20480*EL-30720)-20480)+15360)+EL*(EL*(EL*(
     <    15360*EL-30720)-30720)+46080)+4096)+EL*(EL*(EL*(EL*(6144*EL-153
     =    60)-20480)+46080)+8192)-12288)+EL*(EL*(EL*(EL*(EL*(1024*EL-3072
     >    )-5120)+15360)+4096)-12288))*R1)

      bmat(NI,2) =
     1(N*(N*(N*(N*(N*(N*((-16*N-128*EL)*N+(-480*EL-128)*EL-40)+EL*((-108
     1    8*EL-768)*EL-240))+EL*(EL*((-1680*EL-2176)*EL-472)+736)+431)+EL
     2    *(EL*(EL*((-1856*EL-3584)*EL-288)+2944)+1724))+EL*(EL*(EL*(EL*(
     3    (-1408*EL-3456)*EL+560)+5696)+2800)-1016)-105)+EL*(EL*(EL*(EL*(
     4    EL*((-640*EL-1792)*EL+1056)+5504)+2152)-2032)-210))+EL*(EL*(EL*
     5    (EL*(EL*(EL*((-128*EL-384)*EL+480)+2016)+360)-2088)-838)+402)+1
     6    80)*DSQRT(N**2+2*EL*N+2*EL-1)/((N*(N*(N*(N*(1024*N+5120*EL)+
     7    10240*EL**2-5120)+EL*(10240*EL**2-15360))+EL**2*(5120*EL**2-
     8    15360)+4096)+EL*(EL**2*(1024*EL**2-5120)+4096))*R1)

      bmat(NI,3) =
     1(N*(N*(N*(N*(N*(N*((-32*N-256*EL-128)*N+(-928*EL-1040)*EL-64)+EL*(
     1    (-1984*EL-3648)*EL-816)+256)+EL*(EL*((-2720*EL-7136)*EL-3024)+1
     2    976)+1022)+EL*(EL*(EL*((-2432*EL-8384)*EL-5248)+5696)+6200)+146
     3    8)+EL*(EL*(EL*(EL*((-1376*EL-5904)*EL-4800)+7880)+12626)+4047)-
     4    971)+EL*(EL*(EL*(EL*(EL*((-448*EL-2304)*EL-2256)+5296)+10844)+3
     5    656)-3427)-1641)+EL*(EL*(EL*(EL*(EL*(EL*((-64*EL-384)*EL-432)+1
     6    392)+3396)+1104)-2477)-2085)-450)
     6    *DSQRT(N**4+(4*EL+2)*N**3+(4*EL
     7   **2+10*EL-1) *N**2+ (12*EL**2+2*EL-2) *N+8*EL**2-4*EL) / ( (N* (N* (N* (
     8   N* (N* (512*N+3072*EL+1536) +EL* (7680*EL+7680) -2560) +EL* (EL* (10240
     9   *EL+15360) -10240) -7680) +EL* (EL* (EL* (7680*EL+15360) -15360) -23040
     :   ) +2048) +EL* (EL* (EL* (EL* (3072*EL+7680) -10240) -23040) +4096) +6144)
     ;   +EL* (EL* (EL* (EL* (EL* (512*EL+1536) -2560) -7680) +2048) +6144) ) *R1)

 10   CONTINUE

      DO 20 NI = 4,NMAXA(LIC)

      N = NI-1

      bmat(NI,4) =
     1 (N* (N* (N* (N* (N* (48*N+288*EL+288) +EL* (800*EL+1664) +744) +EL* (EL* (128
     1   0*EL+4096) +3872) +1056) +EL* (EL* (EL* (1200*EL+5184) +7304) +3440) +51
     2   ) +EL* (EL* (EL* (EL* (608*EL+3296) +5968) +3376) -1146) -1242) +EL* (EL* (
     3   EL* (EL* (EL* (128*EL+832) +1792) +992) -1368) -1836) -540)
     3   *DSQRT (N**6+ (
     4   6*EL+6) *N**5+ (12*EL**2+36*EL+10) *N**4+ (8*EL**3+72*EL**2+64*EL) *
     5   N**3+ (48*EL**3+132*EL**2+24*EL-11) *N**2+ (88*EL**3+72*EL**2-22*E
     6   L-6) *N+48*EL**3-12*EL) / ( (N* (N* (N* (N* (2048*N+10240*EL+10240) +EL*
     7    (20480*EL+40960) +10240) +EL* (EL* (20480*EL+61440) +30720) -10240) +E
     8   L* (EL* (EL* (10240*EL+40960) +30720) -20480) -12288) +EL* (EL* (EL* (EL*
     9    (2048*EL+10240) +10240) -10240) -12288) ) *R1)

      bmat(NI,5) =
     1 (N* (N* (N* (N* (N* (32*N+192*EL+288) +EL* (480*EL+1456) +832) +EL* (EL* (640
     1   *EL+2944) +3424) +672) +EL* (EL* (EL* (480*EL+2976) +5280) +2216) -702) +
     2   EL* (EL* (EL* (EL* (192*EL+1504) +3616) +2416) -1236) -1242) +EL* (EL* (EL
     3   * (EL* (EL* (32*EL+304) +928) +872) -534) -1197) -405)
     3   *DSQRT (N**8+ (8*EL+
     4   12) *N**7+ (24*EL**2+92*EL+54) *N**6+ (32*EL**3+264*EL**2+396*EL+10
     5   8) *N**5+ (16*EL**4+336*EL**3+1076*EL**2+776*EL+69) *N**4+ (160*EL*
     6   *4+1280*EL**3+2000*EL**2+612*EL-72) *N**3+ (560*EL**4+2160*EL**3+
     7   1636*EL**2-52*EL-124) *N**2+ (800*EL**4+1568*EL**3+376*EL**2-296*
     8   EL-48) *N+384*EL**4+384*EL**3-96*EL**2-96*EL) / ( (N* (N* (N* (N* (N* (2
     9   048*N+12288*EL+18432) +EL* (30720*EL+92160) +51200) +EL* (EL* (40960*
     :   EL+184320) +204800) +30720) +EL* (EL* (EL* (30720*EL+184320) +307200) +
     ;   92160) -53248) +EL* (EL* (EL* (EL* (12288*EL+92160) +204800) +92160) -10
     <   6496) -49152) +EL* (EL* (EL* (EL* (EL* (2048*EL+18432) +51200) +30720) -5
     =   3248) -49152) ) *R1)

      bmat(NI,6) =
     1 (N* (N* ( (-16*N-64*EL-128) *N+ (-96*EL-384) *EL-344) +EL* ( (-64*EL-384) *E
     1   L-688) -352) +EL* (EL* ( (-16*EL-128) *EL-344) -352) -105)
     1   *DSQRT (N**10+ (
     2   10*EL+20) *N**9+ (40*EL**2+190*EL+165) *N**8+ (80*EL**3+720*EL**2+1
     3   480*EL+720) *N**7+ (80*EL**4+1360*EL**3+5260*EL**2+6080*EL+1743) *
     4   N**6+ (32*EL**5+1280*EL**4+9240*EL**3+20080*EL**2+13978*EL+2100)
     5   *N**5+ (480*EL**5+8000*EL**4+32200*EL**3+42760*EL**2+17030*EL+33
     6   5) *N**4+ (2720*EL**5+24800*EL**4+61320*EL**3+49480*EL**2+7500*EL
     7   -2120) *N**3+ (7200*EL**5+39920*EL**4+62440*EL**3+26340*EL**2-458
     8   0*EL-2244) *N**2+ (8768*EL**5+31520*EL**4+30160*EL**3+1720*EL**2-
     9   5688*EL-720) *N+3840*EL**5+9600*EL**4+4800*EL**3-2400*EL**2-1440
     :   *EL) / ( (N* (N* (N* (N* (2048*N+10240*EL+20480) +EL* (20480*EL+81920) +7
     ;   1680) +EL* (EL* (20480*EL+122880) +215040) +102400) +EL* (EL* (EL* (1024
     <   0*EL+81920) +215040) +204800) +49152) +EL* (EL* (EL* (EL* (2048*EL+2048
     =   0) +71680) +102400) +49152) ) *R1)

 20   CONTINUE

      RETURN
      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE PREINT

*CALL PARMS
*CALL ARRAYS

C apply time differencing: Crank-Nicholson for viscous term
C ---------------------------------------------------------

      DO 5 NI = 1,5
        DO 6 K = 1,6-NI
 6      BTEMP(NI,K) = 0.0
 5    CONTINUE

      DO 17 NI = 1, NMAXA(LIC)
 17     BTEMP(NI,6) = BMAT(NI,1)

      DO 12 NI = 2, NMAXA(LIC)
 12     BTEMP(NI,5) = BMAT(NI-1,2)

      DO 13 NI = 3, NMAXA(LIC)
 13     BTEMP(NI,4) = BMAT(NI-2,3)

      DO 14 NI = 4, NMAXA(LIC)
 14     BTEMP(NI,3) = BMAT(NI-3,4)

      DO 15 NI = 5, NMAXA(LIC)
 15     BTEMP(NI,2) = BMAT(NI-4,5)

      DO 16 NI = 6, NMAXA(LIC)
 16     BTEMP(NI,1) = BMAT(NI-5,6)


      DO 20 K = 1,6
       DO 10 NI = 1,NMAXA(LIC)
        AMAT(NI,K) = -(2.*AMAT(NI,K) - BTEMP(NI,K)*VISC*TINC)
        BMATS(NI,K,LIC) = -2.*BMAT(NI,K)*VISC*TINC
 10    CONTINUE
 20   CONTINUE

C decompose matrix AMAT
C ---------------------

      N = NMAXA(LIC)
      NC = 5
      IA = NMD
      IU = NMD
      CALL LUDAPB (AMAT,N,NC,IA,AINV,IU,D1,D2,IER)

      IF (IER.NE.0) THEN
        WRITE(LWRITE,280) IER,LIC
      END IF
 280  FORMAT(' WARNING: the error parameter is ',I3,' for LIC =',I3)

      DO 31 K = 1,6
       DO 41 NI = 1,NMAXA(LIC)
        AMATS(NI,K,LIC) = AINV(NI,K)
 41    CONTINUE
 31   CONTINUE

      RETURN

      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS


      SUBROUTINE TIMEINT

*CALL PARMS
*CALL ARRAYS

C Write header lines to files:
C ----------------------------

C       WRITE(LGLOBAL,2010)
C 2010 FORMAT(X, 'head:   FrameSaved  time    centroid    ringV
C       1  impulseMP   impulseGQ    dIdt')

C Begin integration ...
C ---------------------

      NSCOUNT = 1
      DO 10 iLOOP = 1,nrLOOP

C   form forcing function (convection term) for all values of L.
c   ------------------------------------------------------------

        CALL FORCING

c   Output solution to cray at every xx time steps
c   ----------------------------------------------
        IF (ILOOP.EQ.1) THEN
          ntCRAY = (nrLOOP -1)/NCRAY +1
        WRITE(LCRAY) ntCRAY
        WRITE(LCRAY) RCONSTD,RADppo,TIMEo
        WRITE(LCRAY) VISC, GAM, CtR
        WRITE(LCRAY) iRUN, iPLOT
        WRITE(LCRAY) NMAX,NCOL,IMAX,LCOL
        WRITE(LCRAY) (NMAXA(LI), LI=1,IMAX)
        WRITE(LCRAY) (R(I), I=1,NCOL)
        WRITE(LCRAY) (XIDD(I),WTR(I), I=1,NCOL)
        WRITE(LCRAY) (CTH(J),WTTH(J), J=1,LCOL)
      END IF

      IDUM1 = ((iLOOP-1)/NCRAY)*100000
      IDUM2 = ((iLOOP-1)*100000)/NCRAY
      IF (IDUM1-IDUM2.EQ.0) THEN
        CALL OUTPUT2
         jCRAY = NSCOUNT
        NSCOUNT = NSCOUNT+1
      ELSE
         jCRAY = 0
      END IF

c   Plot solution every xx time steps
c   ------------------------------------
      IDUM1 = ((iLOOP-1)/NPLOT)*100000
      IDUM2 = ((iLOOP-1)*100000)/NPLOT
      IF (IDUM1-IDUM2.EQ.0) THEN
        CALL STREAMF
        CALL PLTCONT(NCD,LCD,OMEG,CTH,R,XY,SFUNC,F)
        iPLOT = iPLOT + 1
        IF (jPLOT.EQ.0) THEN
          nrPLOT = (nrLOOP-1)/NPLOT+1
          WRITE(LPLOT) iRUN,GAM,TIME,CtR
          WRITE(LPLOT) TINC,NMAX,IMAX
          WRITE(LPLOT) nrPLOT
        END IF
        ENERGY = 0.
        DISSIP = 0.
        WRITE(LPLOT)TIME,CIRC,RADpp,RVELpp
        WRITE(LPLOT)ENERGY,DISSIP,RIMPMP,RIMPGQ,DIMP
         jPLOT = 1
      END IF

      IF (iLOOP.EQ.nrLOOP)GOTO 5   ! Quit integration after nrLOOP steps


C   Compute centroid:
C   ----------------

      ZCoR = ZCoR + RVELP*(EXP(TINC + ALOG(TIME)) -TIME)/
     1                                     (SQRT(TIMEo)*RADppo)

C   Advance the unknown coefficients in time for each L
C   ---------------------------------------------------

      DO 20 LIC = 1,IMAX

C     Multiply B matrix times the solution vector
c     -------------------------------------------
        CALL MATMULT


C     Adams Bashforth differencing applied to the convective term.
c     ------------------------------------------------------------

        IF (iLOOP.EQ.1) THEN
          DO 30 NIP = 1,NMAXA(LIC)
 30       RHS(NIP) = -2.*CONV(NIP,LIC)*TINC + BVEC(NIP)

        ELSE
          DO 40 NIP = 1,NMAXA(LIC)
          RHS(NIP) = -(3.*CONV(NIP,LIC)-CONVM1(NIP,LIC))*TINC
     1              + BVEC(NIP)
 40       CONTINUE

        END IF


C     invert the implicit term with forcing function (RHS(NIP)) as rhs
c     -----------------------------------------------------------------
        N = NMAXA(LIC)
        NC = 5
        IA = NMD

        DO 31 K = 1,6
          DO 41 NI = 1,NMAXA(LIC)
 41       AINV(NI,K) = AMATS(NI,K,LIC)
 31     CONTINUE

        CALL LUELPB(AINV,RHS,N,NC,IA,DELSOLN)


C     the unknown, time dependent coefficients' are:
c     ----------------------------------------------

       DO 70 NIP = 1,NMAXA(LIC)
          ALP(NIP,LIC) = DELSOLN(NIP) + ALP(NIP,LIC)
 70     CONTINUE

 20    CONTINUE


C     save the convective term from the previous time step.
c     -----------------------------------------------------

       DO 90 LIC = 1,IMAX
         DO 100 NIP = 1,NMAXA(LIC)
 100     CONVM1(NIP,LIC) = CONV(NIP,LIC)
 90    CONTINUE

c     Advance "time"
c     --------------
        TIME = EXP(ALOG(TIME) + TINC)

 10   CONTINUE

C end integration.
c ---------------

 5    CONTINUE

      RETURN
      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE MATMULT

*CALL PARMS
*CALL ARRAYS


C initialize the solution vector to zero
C -------------------------------------

      DO 15 NI = 1,NMAXA(LIC)
 15     BVEC(NI) = 0.


C multiply all elements above/on the diagonal
C -------------------------------------------

      DO 10 K = 1,6
        DO 20 NI = 1,NMAXA(LIC) - 5
 20       BVEC(NI) =  BVEC(NI) + BMATS(NI,K,LIC) * ALP(NI-1+K,LIC)
 10   CONTINUE


C multiply all elements below the diagonal
C ----------------------------------------

      DO 30 K = 1,5
        DO 40 NI = 2,NMAXA(LIC) - 4
 40       BVEC(NI+K-1)=BVEC(NI+K-1)+BMATS(NI-1,K+1,LIC)*ALP(NI-1,LIC
 30   CONTINUE


C multiply the lower right hand corner elements (5x5 matrix)
C ----------------------------------------------------------
```

```
      DO 50 NI = NMAXA(LIC), NMAXA(LIC) - 4, -1
        IF (NI.LT.1) GOTO 100

C   above/on the diagonal
C   ----------------------
        DO 60 K = 1,NMAXA(LIC)-NI+1
 60       BVEC(NI) =  BVEC(NI) + BMATS(NI,K,LIC) * ALP(NI-1+K,LIC)
C   below diagonal
C   ---------------
        DO 70 K = 1,NMAXA(LIC)-NI
 70       BVEC(NI+K) = BVEC(NI+K) + BMATS(NI,K+1,LIC) *ALP(NI,LIC)

 50   CONTINUE

 100  CONTINUE


      RETURN
      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE MATMULT2(LI,NMD,IMD,NSB,ALP,NMAXA,RMAT,SOLNV)

*CALL PARMS

      DIMENSION RMAT(NMD,NSB+1,IMD), ALP(NMD,IMD), NMAXA(IMD)
      DIMENSION SOLNV(NMD)

C initialize the solution vector to zero
C ---------------------------------------

      DO 15 NI = 1,NMAXA(LI)
 15     SOLNV(NI) = 0.


C multiply all elements above/on the diagonal
C --------------------------------------------

      DO 10 K = 1,NSB+1
        DO 20 NI = 1,NMAXA(LI) - NSB
 20       SOLNV(NI) =  SOLNV(NI) + RMAT(NI,K,LI) * ALP(NI-1+K,LI)
 10   CONTINUE

C multiply all elements below the diagonal
C -----------------------------------------

      DO 30 K = 1,NSB
        DO 40 NI = 2,NMAXA(LI) - (NSB-1)
 40       SOLNV(NI+K-1)=SOLNV(NI+K-1)+RMAT(NI-1,K+1,LI)*ALP(NI-1,LI)
 30   CONTINUE


C multiply the lower right hand corner elements (NSBxNSB matrix)
C ---------------------------------------------------------------
      DO 50 NI = NMAXA(LI), NMAXA(LI) - (NSB-1), -1
        IF (NI.LT.1) GOTO 100

C   above/on the diagonal
C   ----------------------
        DO 60 K = 1,NMAXA(LI)-NI+1
 60       SOLNV(NI) =  SOLNV(NI) + RMAT(NI,K,LI) * ALP(NI-1+K,LI)

C   below diagonal
C   ---------------
        DO 70 K = 1,NMAXA(LI)-NI
 70       SOLNV(NI+K) = SOLNV(NI+K) + RMAT(NI,K+1,LI) *ALP(NI,LI)

 50   CONTINUE

 100  CONTINUE


      RETURN
      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE FORCING

*CALL PARMS
*CALL ARRAYS

C     Go FROM wave space TO real space
c     ---------------------------------

      DO 10 I=1,NCOL
        DO 20 J=1,LCOL
          VELRA(J) = 0.
          VELTA(J) = 0.
          VORTA(J) = 0.
 20     CONTINUE

        DO 30 LI=1,IMAX
          VELR = 0.
          VELT = 0.
          VORT = 0.

          DO 40 NI=1,NMAXA(LI)
            VELR = VELR + GR(NI,LI,I)*ALP(NI,LI)
            VELT = VELT + GTH(NI,LI,I)*ALP(NI,LI)
            VORT = VORT + GV(NI,LI,I)*ALP(NI,LI)
 40       CONTINUE

          DO 50 J = 1,LCOL
            VELRA(J) = VELRA(J) + VELR*P0(LI,J)
            VELTA(J) = VELTA(J) + VELT*P1(LI,J)
            VORTA(J) = VORTA(J) + VORT*P1(LI,J)
 50       CONTINUE
 30     CONTINUE

        DO 60 J = 1,LCOL
          UR(I,J) = VELRA(J)
          UTH(I,J) = VELTA(J)
          OMEG(I,J) = VORTA(J)
 60     CONTINUE
 10   CONTINUE

c Compute Saffman's velocity
      DO 120 J=1,LCOL
         STH(J) = SQRT(1.-CTH(J)**2)
 120  CONTINUE


      IF (ILOOP.EQ.1) THEN

c     For the first time step only, compute the RMS error of vorticity
c     -----------------------------------------------------------------

c the initial condition is a Stokes solution
c -------------------------------------------
      IF (ICFLAG.EQ.1) THEN
      RMSD = 0.
      rnorm = 0.
      DO 132 I = 1,NCOL
        RMSD1 = 0.
        rnorm1 = 0.
        WFth = 1.
        DO 142 J = 1,LCOL
          EXOMEG = STH(J)*R(I)*EXP(-R(I)**2/4.)/(TIME*16.*PI**(3./2.))
          RMSD1 = RMSD1 + (EXOMEG-OMEG(I,J))**2*WTTH(J)/(WFth*STH(J))
          rnorm1 = rnorm1 + EXOMEG**2*WTTH(J)/(WFth*STH(J))
 142    CONTINUE
        RMSD = RMSD + RMSD1*F2JAC(I)/R(I)
        rnorm = rnorm + rnorm1*F2JAC(I)/R(I)
 132  CONTINUE
      RMSD = SQRT(RMSD)
      WRITE(LWRITE,1100)NMAXA(1),RMSD
      WRITE(LWRITE,1101)rnorm
      END IF

C the initial condition is a thin ring
c -------------------------------------
      IF (ICFLAG.EQ.2) THEN
      FIXED = 1.2564392281
      CAPR = RADppo
      RMSD = 0.
      rnorm = 0.
      DO 131 I = 1,NCOL
        RMSD1 = 0.
        rnorml = 0.
        WFth = 1.
        DO 141 J = 1,LCOL
          ARG =  EXP(  -(FIXED/CtR**2)* ( 1.+ (R(I)*R(I)/(CAPR*CAPR))
     1    - 2.*R(I)*STH(J)/CAPR )  )
          ARG2 =  EXP( -(FIXED/CtR**2)* ( 1.+ (R(I)*R(I)/(CAPR*CAPR))
     1    + 2.*R(I)*STH(J)/CAPR )  )
          EXOMEG = FIXED *GAM *(ARG-ARG2)/(PI* VISC *CAPR**2 *CtR**2)
          RMSD1 = RMSD1 + (EXOMEG-OMEG(I,J))**2*WTTH(J)/(WFth*STH(J))
          rnorml = rnorml + EXOMEG**2*WTTH(J)/(WFth*STH(J))
 141    CONTINUE
        RMSD = RMSD + RMSD1*F2JAC(I)/R(I)
        rnorm = rnorm + rnorml*F2JAC(I)/R(I)
 131  CONTINUE
      RMSD = SQRT(RMSD)
      WRITE(LWRITE,1100)NMAXA(1),RMSD
      WRITE(LWRITE,1101)rnorm
      END IF

 1100 FORMAT(X,'NMAXA(1) = ',I2,5x,'RMSerror =',f18.9)
 1101 format(x,'rnorm = ',f19.8)
      END IF

c     Compute speed of Saffman centroid and other stuff
c     ------------------------------------------------

      RIMP = 0.
      CP1 = 0.
      CP2 = 0.
      TP1 = 0.
      TP2 = 0.
      CIRC = 0.
      RADpp = 0.
      RMOM = 0.
      DCENT = 0.
      DO 130 I = 1,NCOL
c       an inner loop begins here for the double integration
        RIMP1 = 0.
        CENT1 = 0.
        CENT2= 0.
        TIMP1 = 0.
        TIMP2 = 0.
        CIRC1 = 0.
        RADpp1 = 0.
        RMOM1 = 0.
        DCENT1 = 0.
        WFth = 1.
        DO 140 J = 1,LCOL
          UR1 = UR(I,J)
          UTH1 = UTH(I,J)
          RIMP1 = RIMP1 + OMEG(I,J)*STH(J)*WTTH(J)/WFth
          CENT1 = CENT1 + UR1*OMEG(I,J)*STH(J)*CTH(J)*WTTH(J)/WFth
          CENT2 = CENT2 + UTH1*OMEG(I,J)*(3.*CTH(J)*CTH(J)-1.)
     1              *WTTH(J)/WFth
          TIMP1 = TIMP1 + UR1*OMEG(I,J)*STH(J)*WTTH(J)/WFth
          TIMP2 = TIMP2 + UTH1*OMEG(I,J)*CTH(J)*WTTH(J)/WFth
          CIRC1 = CIRC1 + OMEG(I,J)*WTTH(J)/(WFth*STH(J))
          RADpp1 = RADpp1 + OMEG(I,J)*WTTH(J)/WFth
          RMOM1 = RMOM1 + (UR1*CTH(J)-UTH1*STH(J))*WTTH(J)/WFth
          DCENT1 = DCENT1 + OMEG(I,J)*STH(J)*CTH(J)*WTTH(J)/WFth
 140    CONTINUE

c       weight function of quadrature method
        WFrad = SQRT((1.- XI(I))/XI(I))
C Note that F2JAC(I) = RJAC(I)*WTR(I)/WFrad
        RIMP = RIMP + RIMP1*R(I)*F2JAC(I)
        CP1 = CP1 + CENT1*R(I)*F2JAC(I)
        CP2 = CP2 + CENT2*R(I)*F2JAC(I)
        TP1 = TP1 + TIMP1*RJAC(I)*WTR(I)/WFrad
        TP2 = TP2 + TIMP2*RJAC(I)*WTR(I)/WFrad
        CIRC = CIRC + CIRC1*F2JAC(I)/R(I)
c12/10/87        RADpp = RADpp + RADpp1*R(I)*F2JAC(I)
        RADpp = RADpp + RADpp1*F2JAC(I)
        RMOM = RMOM + RMOM1*F2JAC(I)
        DCENT = DCENT + DCENT1*R(I)**2*F2JAC(I)
 130  CONTINUE
C compute impulse by multipole expansion
      SUM = 0.
      DO 21 NI = 1,NMAX
        SUM = SUM + ALP(NI,1)*CIMP(NI)
 21   CONTINUE

        IF (TIME.EQ.0.) THEN
          TIME= 1./(RIMP *PI)
          TIMEo = TIME
        END IF
C BE CAREFUL HERE, THIS STATEMENT WAS CHANGED QUICKLY WITHOUT
C MUCH THOUGHT FROM ILOOP.EQ.1 TO INCLUDE IF TIME.EQ.0.
C        IF (ILOOP.EQ.1.AND.TIME.EQ.0.) TIMEo = TIME

        RIMPMP = SUM*TIME
        RIMPGQ = RIMP*PI*TIME
        RVELPP = time*pi*(3.*CP1 + CP2)
        DIMP = 2.* PI*(TP1 + TP2)
        IF (ICFLAG.EQ.2) THEN
         SAFVEL = gam*(ALOG(4.*2.24182/CtR) - 0.558)/(4.*pi*RADppo)
        ELSE
         SAFVEL = 0.0
        END IF

C COMPUTE THE CFL NUMBER AT THE ORIGIN
        IF (ILOOP.EQ.1) THEN
        CFL = TINC*(UTH(1,LCOL/2)+RVELPP)/(R(2)-R(1))
C 1300 FORMAT(' THE CFL NUMBER IS',F16.8)
        WRITE(6,*) 'I, CFLrad,  CFLtheta'
        DO 1310 I=1,NCOL-1
          CFLrad = TINC*(UTH(I,LCOL/2)+RVELPP)/(R(I+1)-R(I))
          CFLth = TINC*(UTH(I,LCOL/2)+RVELPP)/(R(I)*pi/lcol)
          WRITE(6,*) I, CFLrad, CFLth
 1310 CONTINUE
        END IF

C 4/16 SET THE RING VELOCITY TO ZERO WHEN SOLVING THE STOKES EQNS
C(4/27)       RVELPP = 0.
        RVELP = RVELPP/SQRT(TIME)
        RMOM = 3.*pi*time*rmom
        DCENT = DCENT*PI*TIME
        IF (RADpp/CIRC.LT.0) THEN
          WRITE(6,1999)RADpp/CIRC
 1999   FORMAT(X,'RADpp/CIRC = ',F16.8)
        ELSE
          RADpp = RADpp/CIRC
        END IF

        WRITE(LWRITE,1010) TIME
        WRITE(LWRITE,1015) CIRC
        WRITE(6,1003)DCENT
 1003   FORMAT(X,'the centroid computed by GQ is ',F16.8)
        WRITE(6,1002)RMOM
 1002   FORMAT(X,'H(t)/ (2/3)I/rho is ',F16.8)
        WRITE(6,1001)RADpp
 1001   FORMAT(X,'RADpp IS ',F16.8)

C write impulse and related quantities
        WRITE(LWRITE,1020) RIMPMP
        WRITE(LWRITE,1030) RIMPGQ
        WRITE(LWRITE,1040) RIMPMP-RIMPGQ
        WRITE(LWRITE,1050) DIMP

C write propagation speed and related quantities
        WRITE(LWRITE,1060) RVELPP
        WRITE(LWRITE,1070) SAFVEL
c        WRITE(LWRITE,1040) SAFVEL - (RVELPP*4.*PI*RADppo/GAM)
        WRITE(LWRITE,1040) SAFVEL - RVELPP

 1010 FORMAT(X,'time (sec) =',F16.8)
 1015 FORMAT(X,'computed circulation =',F16.8,/)
 1020 FORMAT(X,'impulse (L^4/T) (multipole expansion) =',F16.8)
 1030 FORMAT(X,'impulse (L^4/T) (Gauss Quadrature)=',F16.8)
 1040 FORMAT(X,'diff = ',F16.8,/)
 1050 FORMAT(X,'d(impulse)/dt (Gauss Quadrature =',E16.8,/)
 1060 FORMAT(X,'propagation speed (non-dim) =',F16.8)
 1070 FORMAT(X,'Saffman propagation velocity (non-dim) =',F16.8)

C compute RMS difference and maximum error for the Stokes solution at
c each time step.


c     Every xx steps, compute the energy and dissipation and write global
c     -----------------------------------------------------------------
```

```
      IDUM1 = ((iLOOP-1)/NGLOBAL)*100000
      IDUM2 = ((iLOOP-1)*100000)/NGLOBAL
      IF (IDUM1-IDUM2.EQ.0) THEN
        IF (jGLOBAL.EQ.0) THEN
          nrGLOBAL = (nrLOOP-1)/NGLOBAL+1
          WRITE(LGLOBAL) iRUN,GAM,TIME,CtR
          WRITE(LGLOBAL) TINC,NMAX,IMAX
          WRITE(LGLOBAL) nrglobal
        END IF

C    compute energy and dissipation

C      Dissipation by matrix multiplication
        DISSIP = 0.
        DO 200 LI = 1,IMAX
          RL = REAL(LI)
          CTEMP = - (4.*PI*RL*(RL+1.))/(2.*RL+1.)
          CALL MATMULT2(LI,NMD,IMD,5,ALP,NMAXA,BMATS,DISV)
          DO 210 NI = 1,NMAXA(LI)
            DISSIP = DISSIP + CTEMP*DISV(NI)*ALP(NI,LI)
 210      CONTINUE
 200    CONTINUE

        DISSIP = -DISSIP/(2.*VISC*TINC*SQRT(TIME))

C      Energy by Gauss Quadrature
        ENERG = 0.
        DO 220 I = 1,NCOL
          ENERG1 = 0.
          DO 230 J = 1,NCOL
            ENERG1 = ENERG1 + (UR(I,J)*UR(I,J)+UTH(I,J)*UTH(I,J))
     >                                    *WTH(J)/WFth
 230      CONTINUE
          ENERG = ENERG + ENERG1*F2JAC(I)
 220    CONTINUE
        ENERGY = ENERG*PI*SQRT(TIME)

        WRITE(LGLOBAL)TIME,CIRC,RADpp,RVELpp
        WRITE(LGLOBAL)ENERGY,DISSIP,RIMPMP,RIMPGQ,DIMP
      jGLOBAL = 1
      END IF


C     Go FROM real space TO wave space
c     ________________________________

C Compute this part of the code in the first pass only.
      IF (iLOOP.EQ.1) THEN

      DO 74 LI=1,IMAX
        RL = REAL(LI)
        COEF(LI) = - (2.*RL+1.) / (2.*RL*(RL+1.))
        DO 75 J = 1,LCOL
          P0FUN(LI,J) = P0(LI,J)*WTH(J)
          P1FUN(LI,J) = P1(LI,J)*WTH(J)
 75     CONTINUE
 74   CONTINUE

      DO 76 I=1,NCOL
        RD2(I) = 0.5*R(I)
 76   CONTINUE

      END IF


C     convective term in the Navier Stokes Equations.
      DO 70 LI=1,IMAX

        DO 71 I=1,NCOL
          SUM1A(I) = 0.
          SUM2A(I) = 0.
 71     CONTINUE

C       integrate wrt theta, for all LI

        DO 165 J = 1,LCOL

          RVTH = RVELPP*STH(J)
          RVR = RVELPP*CTH(J)

          DO 160 I=1,NCOL

C           WHEN COMPUTING THE STOKES SOLUTION SET RVELPP,UTH,UR = 0.
C           (ALSO SEE ABOVE)
C            UTH(I,J) = 0.
C            UR(I,J) = 0.
            SUM1A(I) = SUM1A(I)+ (UTH(I,J)+RVTH)* OMEG(I,J) *P0FUN(LI,J)
            SUM2A(I) = SUM2A(I)+ ( UR(I,J)- RVR
     1      - RD2(I)
     2      )*OMEG(I,J)*P1FUN(LI,J)

C         end of i loop
 160      CONTINUE

C       end of J loop
 165    CONTINUE

C end of transformation wrt theta...

C Integrate wrt r, for all NIP and LI

        DO 151 NIP=1,NMAXA(LI)
          P1dum(NIP) = 0.0
          P2dum(NIP) = 0.0
 151    CONTINUE

        DO 161 I=1,NCOL


          TEMP1 = SUM1A(I)*F2JAC(I)
          TEMP2 = SUM2A(I)*F2JAC(I)

          DO 150 NIP=1,NMAXA(LI)

            P1dum(NIP) = P1dum(NIP) + TEMP1 * GR(NIP,LI,I)
            P2dum(NIP) = P2dum(NIP) + TEMP2 * GTH(NIP,LI,I)

C         end of NIP loop
 150      CONTINUE

C       end of i loop
 161    CONTINUE


        DO 152 NIP=1,NMAXA(LI)
          CONV(NIP,LI) = COEF(LI)* (P1dum(NIP) - P2dum(NIP))
 152    CONTINUE

C     end of LI loop
 70   CONTINUE


      RETURN
      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE OUTPUT2

*CALL PARMS
*CALL ARRAYS

      WRITE(LCRAY) TIME,ZCoR
c      WRITE(LCRAY) ((ALP(NI,LI), NI=1,NMAXA(LI)), LI=1,IMAX)
      WRITE(LCRAY) ((omeg(I,J), I=1,NCOL), J=1,LCOL)

      RETURN
      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE OUTPUT

*CALL PARMS
*CALL ARRAYS

      WRITE (LVAX,1002)
      WRITE (LVAX,*) TIME,ZCoR
      WRITE (LVAX,1000)
      WRITE (LVAX,*) RCONSTD,RADppo,TIMEo
      WRITE (LVAX,1007)
      WRITE (LVAX,*) VISC, GAM, CtR
      WRITE (LVAX,1001)
      WRITE (LVAX,*) iRUN, iPLOT
      WRITE (LVAX,1005)
      WRITE (LVAX,*) NMAX,NCOL,IMAX,LCOL

      WRITE (LVAX,1009)
      DO 6 LI=1,IMAX
        WRITE (LVAX,*) LI,NMAXA(LI)
6     CONTINUE

      WRITE (LVAX,1010)
      DO 10 I=1,NCOL
        WRITE (LVAX,*) I,XIDD(I),WIR(I)
10    CONTINUE

      WRITE (LVAX,1020)
      DO 20 J=1,LCOL
        WRITE (LVAX,*) J,CTH(J),WTTH(J)
20    CONTINUE

      WRITE (LVAX,1030)
      DO 30 LI=1,IMAX
        DO 40 NI=1,NMAXA(LI)
c           WRITE (LWRITE,*) LI,NI,ALP(NI,LI)
          WRITE (LVAX,*) LI,NI,ALP(NI,LI)
40      CONTINUE
30    CONTINUE

1002 FORMAT(5X,'TIME',12X,'ZCoR')
1000 FORMAT (12X,'RCONST',10X,'RADppo')
1001 FORMAT (4X,'iRUN',4X,'iPLOT')
1005 FORMAT (9X,'NMAX',8X,'NCOL',8X,'IMAX',8X,'LCOL')
1007 FORMAT ( 5X,'VISC',12X,'GAM',5X,'CtR')
1009 FORMAT (5X,'LINDEX',10X,'NMAXA(L)')
1010 FORMAT ( 10X,'I',7X,'XI(I)',10X,'WIR(I)')
1020 FORMAT ( 10X,'J',7X,'CTH(J)',10X,'WTTH(J)')
1030 FORMAT ( 9X,'LI',10X,'NI',7X,'ALP(NI,LI)')

      RETURN
      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE STREAMF

*CALL PARMS
*CALL ARRAYS

c compute the stream function in a reference frame translating with the
c velocity of the Saffman centroid.
      DO 10 I=1,NCOL
        DO 20 J=1,LCOL
20        SA(J) = 0.
        DO 30 LI=1,IMAX
        . RL = REAL(LI)
          STEMP = 0.
          DO 40 NI=1,NMAXA(LI)
40          STEMP = STEMP + GR(NI,LI,I)*ALP(NI,LI)
          STEMP = STEMP*R(I)/ (RL* (RL+1.))
          DO 50 J = 1,LCOL
50          SA(J) = SA(J) + STEMP*P1(LI,J)
30      CONTINUE
C compute the Saffman ring speed
c         Uo = gam*(ALOG(4.*2.24182/CtR) - 0.558)/(4.*pi*RADppo)
        Uo = RVELPP
        DO 60 J = 1,LCOL
          SFUNC(I,J)=-(SA(J)*STH(J)*R(I) +.5*Uo*R(I)**2*STH(J)**2)
60      CONTINUE
10    CONTINUE

      RETURN
      END

C SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

      SUBROUTINE PLTCONT(NCD,LCD,OMEG,CTH,R,XY,SFUNC,F)
```

```
C DIMENSIONS/OPTIONS:
c izero:          0 : do not plot zero contour level
c                 1 : plot zero contour level
C     NCOPT             =1 (input contour levels), =2 (automatic)
C     NCOL              IN F(I,J) I = 1,NCOL
C     LCOL                        J = 1,LCOL
C     NCONT             number of contour levels
C     NXDIM             number of points in the contour definitions
C     NADDIM            number of contours
C     IADIM             try 3000 (says Peter Buning)
C ARRAYS:
C     F(NCD,LCD)        FUNCTION TO PLOT CONTOURS
C     XY(NCD,LCD)       coordinates of function
C     ACONT(NCdim)      (for NCONT=1) contour levels
C     XCONT(NXDIM)      x values of contour lines
C     YCONT(NXDIM)      y values of contour lines
C     NAD(NADDIM)       NAD(1): # contours, pointers to each contour lin
C     NLEV(NADDIM)      values of contour levels
C     IA(IADIM)         scratch array
*CALL PARMS

      PARAMETER(NCdim=25 ,NXDIM=2000 ,NADDIM=100 ,IADIM=3000)
      DIMENSION XY(NCD,LCD,2),F(NCD,LCD),SFUNC(NCD,LCD),OMEG(NCD,LCD)
      DIMENSION ACONT1(NCdim),ACONT2(NCdim),ACONT3(NCdim),ACONT4(NCdim)
      DIMENSION XCONT(NXDIM,2)
      DIMENSION NAD(NADDIM), NLEV(NADDIM), IA(IADIM)
      DIMENSION blankc(NXDIM)
      LOGICAL LABELS

      DIMENSION CTH(LCD),R(NCD)
      REAL LEFT

      LABELS = .FALSE.

C intilize disspla
      IF (j2PLOT.LE..5)THEN
        CALL DIP(15)
        j2PLOT = 1.
      END IF
      CALL PAGE(8.5,11.)
      CALL HWSCAL('SCREEN')

C set character type and size
      CALL SCMPLX
      CALL MX1ALF('L/CSTD',')')
      CALL MX2ALF('STANDARD','(')
      CALL MX3ALF ('L/CGREEK','%')
      CALL MX4ALF ('GREEK','!')
      CALL MX5ALF ('INSTRU','\')
      CALL MX6ALF ('MATH','*')
      CALL HEIGHT(0.12)
c     label axis with integer numbers
      CALL INTAXS
c     do not plot outside of subplot area
      CALL GRACE(0)
c     connect points with a polynomial fit
      CALL POLY3
c     don't check for points which are out of range
      CALL NOCHEK

C two
C lower plot, page 1 - expanding coordinates
c Plot of w't'^2 in self similar coordinates
      CALL PHYSOR(1.25,7.75)
      CALL AREA2D(6.25,2.5)

      CALL MESSAG('RUN (#) $',100,5.2,2.75)
      CALL INTNO(iRUN,'ABUT','ABUT')

      CALL MESSAG('FRAME (#) $',100,5.2,2.55)
      CALL INTNO(iPLOT,'ABUT','ABUT')
      CALL ENDGR(0)

      DO 10 I=1,NCOL
        DO 20 J=1,LCOL
          STH = SQRT(1. - CTH(J)*CTH(J))
          XY(I,J,1) = R(I)*CTH(J)
          XY(I,J,2) = R(I)*STH
          F(I,J) = OMEG(I,J)*TIME
20      CONTINUE
```

```
10    CONTINUE

      NCOPT = 1
      NCONT2 = 20
      izero = 0

      CALL PHYSOR(1.5,2.25)
      LEFT = -5.
      RIGHT = 5.
      BOT = 0.
      TOP = 5.
      SCALE = 6./(RIGHT-LEFT)
      XDIST = RIGHT-LEFT

      CALL AREA2D(SCALE*(RIGHT-LEFT),SCALE*(TOP-BOT))

      CALL XNAME('X/(()%N)t()) \EH.5)1/2\EXHX)$',100)
      CALL YNAME('Y/(()%N)t()) \EH.5)1/2\EXHX)$',100)

      CALL XREVTK
      CALL YREVTK
      CALL GRAF(LEFT,1.,RIGHT,BOT,1.,TOP)

      CALL CONTXX(NCD,LCD,1,NCOL,1,LCOL,XY,F,NCOPT,NCDIM,NCONT2,
     1            ACONT2,NXDIM,XCONT,XCONT(1,2),NADDIM,NAD,NLEV,
     2            IADIM,IA,FMIN2,FMAX2,izero)
      CALL CONDI2(1,2,LABELS,NXDIM,NADDIM,NCDIM,NCONT2,XCONT,NAD,NLEV,
     1            ACONT2,BLANKC)
      CALL RESET('XREVTK')
      CALL RESET('YREVTK')
      CALL XNONUM
      CALL YNONUM
      CALL XGRAXS(LEFT,1.,RIGHT,SCALE*(RIGHT-LEFT),' ',1,0.,
     1            XDIST*.5*SCALE)
      CALL YGRAXS(BOT,1.,TOP,SCALE*(TOP-BOT),' ',1,XDIST*SCALE,0.)

      CALL THKFRM(.02)
      CALL FRAME
      CALL ENDGR(0)

      CALL RESET('XNONUM')
      CALL RESET('YNONUM')

C one
C Second plot - physical coordinates

      XYSC = SQRT(TIME/TIMEo)
      DO 15 I=1,NCOL
        DO 25 J=1,LCOL
          STH = SQRT(1. - CTH(J)*CTH(J))
          XY(I,J,1) = (R(I)*CTH(J)*XYSC)/RADppo + ZCoR
          XY(I,J,2) = R(I)*STH*XYSC/RADppo
          F(I,J) = OMEG(I,J)*TIMEo**2/TIME
25      CONTINUE
15    CONTINUE

      izero = 0
      NCONT1 = 20
      IF (jPLOT.EQ.0) THEN
        NCOPT = 1
      ELSE
        NCOPT = 2
        CALL FRANGE(NCD,LCD,1,NCOL,1,LCOL,F,FMIN1,FMAX1)
      END IF

C "p" is used here to mean "physical" space.
      CALL PHYSOR(1.5,6.75)

C Scale the plot so that the ring is inside the box.
      IF (jPLOT.EQ.0) PXDIST = 8.
      IF (ZCoR.GT.PXDIST-1) PXDIST = PXDIST*2.
      IF (RADPP/RADPPo+.5.GT.PXDIST/4.) PXDIST = PXDIST*2.

      PLEFT = -1.
      PRIGHT = PLEFT+PXDIST
      PBOT = -PXDIST/4.
      PTOP = PXDIST/4.
      PSCALE = 6./PXDIST

      CALL AREA2D(PSCALE*(PRIGHT-PLEFT),PSCALE*(PTOP-PBOT))
```

```
      CALL XREVTK
      CALL YREVTK
      CALL XNAME('X/(R)\LH.6)O\LXHXS',100)
      CALL YNAME('Y/(R)\LH.6)O\LXHXS',100)

      CALL GRAF(PLEFT,1.,PRIGHT,PBOT,1.,PTOP)
      NCOPT = 1
      CALL CONTXX(NCD,LCD,1,NCOL,1,LCOL,XY,F,NCOPT,NCDIM,NCONT1,
     1            ACONT1,NXDIM,XCONT,XCONT(1,2),NADDIM,NAD,NLEV,
     2            IADIM,IA,FMIN1,FMAX1,izero)
      CALL CONDI2(1,2,LABELS,NXDIM,NADDIM,NCDIM,NCONT1,XCONT,NAD,NLEV,
     1            ACONT1,BLANKC)

      CALL DASH
      CALL RLVEC(PLEFT,0.,PRIGHT,0.,0)
      CALL RESET('DASH')

      DO 50 I = 1, NAD(NAD(1)+1)-1
        XCONT(I,2) = -XCONT(I,2)
50    CONTINUE

      CALL CONDI2(1,2,LABELS,NXDIM,NADDIM,NCDIM,NCONT1,XCONT,NAD,NLEV,
     1            ACONT1,BLANKC)

      CALL XNONUM
      CALL YNONUM
      CALL RESET('XREVTK')
      CALL RESET('YREVTK')
      CALL XGRAXS(PLEFT,1.,PRIGHT,PSCALE*(PRIGHT-PLEFT),' ',1,0.,
     1            PXDIST*.5*PSCALE)
      CALL YGRAXS(PBOT,1.,PTOP,PSCALE*(PTOP-PBOT),' ',
     1                                         1,PXDIST*PSCALE,0.)
      CALL THKFRM(.02)
      CALL FRAME
      CALL ENDGR(0)

      CALL ENDPL(-1)
      CALL RESET('XNONUM')
      CALL RESET('YNONUM')


C four
c plot stream function in coordinate system translating ring speed.
C First plot, second page - expanding coordinates
      CALL PHYSOR(1.25,7.75)
      CALL AREA2D(6.25,2.5)
      CALL LINEAR

      CALL MESSAG('RUN (#) $',100,5.2,2.75)
      CALL INTNO(iRUN,'ABUT','ABUT')

      CALL MESSAG('FRAME (#) $',100,5.2,2.55)
      CALL INTNO(iPLOT,'ABUT','ABUT')
      CALL ENDGR(0)

      DO 44 I=1,NCOL                            ! plot the conours of Psi
        DO 43 J=1,LCOL
          STH = SQRT(1. - CTH(J)*CTH(J))
          XY(I,J,1) = R(I)*CTH(J)
          XY(I,J,2) = R(I)*STH
          F(I,J) = SFUNC(I,J)*TIME/TIMEo
43      CONTINUE
44    CONTINUE
      LCOL2 = LCOL

C Find appropriate intervals
c   the # of positive contours (inside the ring): NPCONT
      NPCONT = 7
      CALL FRANGE(NCD,LCD,1,NCOL,1,LCOL,F,FMIN4,FMAX4)
      ftemp = sqrt(FMAX4)
      CALL CONSCL(0.,ftemp,NCDIM,NPCONT,ACONT4,1)

      DCONT = (ACONT4(2)-ACONT4(1))**2
      NCONT4 = 3*NPCONT                         ! total number of contours

      DO 60 I = 1,2*NPCONT                      ! negative contour levels
        ACONT4(I) = -DCONT*(I-2*NPCONT)**2
60    CONTINUE
      DO 61 I = 1,NPCONT                        ! positive contour levels
        ACONT4(I+2*NPCONT) = DCONT*I**2
```

```
61    CONTINUE
      izero = 1                                ! plot zero contour level
      NCOPT = 2                                ! contour levels are input

      CALL PHYSOR(1.5,2.25)
      LEFT = -5.
      RIGHT = 5.
      BOT = 0.
      TOP = 5.
      SCALE = 6./(RIGHT-LEFT)
      XDIST = RIGHT-LEFT

      CALL AREA2D(SCALE*(RIGHT-LEFT),SCALE*(TOP-BOT))
      CALL XREVTK
      CALL YREVTK
      CALL XNAME('X/(()%N)t()) \EH.5)1/2\EXHX)$',100)
      CALL YNAME('Y/(()%N)t()) \EH.5)1/2\EXHX)$',100)
      CALL GRAF(LEFT,1.,RIGHT,BOT,1.,TOP)

      CALL CONTXX(NCD,LCD,1,NCOL,1,LCOL2,XY,F,NCOPT,NCDIM,NCONT4,
     1            ACONT4,NXDIM,XCONT,XCONT(1,2),NADDIM,NAD,NLEV,
     2            IADIM,IA,FMIN4,FMAX4,izero)
      CALL CONDI2(1,2,LABELS,NXDIM,NADDIM,NCDIM,NCONT4,XCONT,NAD,NLEV,
     1            ACONT4,BLANKC)

      CALL RESET('XREVTK')
      CALL RESET('YREVTK')

      CALL XNONUM
      CALL YNONUM

      CALL XGRAXS(LEFT,1.,RIGHT,SCALE*(RIGHT-LEFT),' ',1,0.,
     1            XDIST*.5*SCALE)
      CALL YGRAXS(BOT,1.,TOP,SCALE*(TOP-BOT),' ',1,XDIST*SCALE,0.)
      CALL THKFRM(.02)
      CALL FRAME
      CALL ENDGR(0)

      CALL RESET('XNONUM')
      CALL RESET('YNONUM')
c
cC three
cC second plot, second page
c
      XYSC = SQRT(TIME/TIMEo)
      DO 16 I=i,NCOL
        XY(I,1,1) = (R(I)*XYSC)/RADppo + ZCoR
        XY(I,1,2) = 0.
        F(I,1) = 0.
        DO 26 J=1,LCOL
          STH = SQRT(1. - CTH(J)*CTH(J))
          XY(I,J+1,1) = (R(I)*CTH(J)*XYSC)/RADppo + ZCoR
          XY(I,J+1,2) = R(I)*STH*XYSC/RADppo
          F(I,J+1) = SFUNC(I,J)*SQRT(TIME/TIMEo)
 26     CONTINUE
        XY(I,LCOL+2,1) = (-R(I)*XYSC)/RADppo + ZCoR
        XY(I,LCOL+2,2) = 0.0
        F(I,LCOL+2) = 0.
 16   CONTINUE

      CALL PHYSOR(1.5,6.75)

      CALL AREA2D(PSCALE*(PRIGHT-PLEFT),PSCALE*(PTOP-PBOT))
      CALL XREVTK
      CALL YREVTK
      CALL XNAME('X/(R)\LH.6)O\LXHXS',100)
      CALL YNAME('Y/(R)\LH.6)O\LXHXS',100)
      CALL GRAF(PLEFT,1.,PRIGHT,PBOT,1.,PTOP)
      NCOPT = 2
      lcol2 = lcol+2

C FOR NOW, LET ()3=()4
      NCONT3 = NCONT4
      DO 62 I = 1,NCONT3
        ACONT3(I) = ACONT4(I)
 62   CONTINUE


      CALL CONTXX(NCD,LCD,1,NCOL,1,LCOL2,XY,F,NCOPT,NCDIM,NCONT3,
     1            ACONT3,NXDIM,XCONT,XCONT(1,2),NADDIM,NAD,NLEV,
```

```
     2            IADIM,IA,FMIN3,FMAX3,izero)
      CALL CONDI2(1,2,LABELS,NXDIM,NADDIM,NCDIM,NCONT3,XCONT,NAD,NLEV,
     1            ACONT3,BLANKC)

      CALL DASH
      CALL RLVEC(PLEFT,0.,PRIGHT,0.,0)
      CALL RESET('DASH')

Ccc scales plot to fill plotting region
      DO 75 I = 1, NAD(NAD(1)+1)-1
        XCONT(I,2) = -XCONT(I,2)
 75   CONTINUE

      CALL CONDI2(1,2,LABELS,NXDIM,NADDIM,NCDIM,NCONT3,XCONT,NAD,NLEV,
     1            ACONT3,BLANKC)

      CALL XNONUM
      CALL YNONUM
      CALL RESET('XREVTK')
      CALL RESET('YREVTK')

      CALL XGRAXS(PLEFT,1.,PRIGHT,PSCALE*(PRIGHT-PLEFT),' ',1,0.,
     1            PXDIST*.5*PSCALE)
      CALL YGRAXS(PBOT,1.,PTOP,PSCALE*(PTOP-PBOT),' ',
     1                                         1,PXDIST*PSCALE,0.)
      CALL THKFRM(.02)
      CALL FRAME
      CALL ENDGR(0)
      CALL ENDPL(-2)
      CALL RESET('XNONUM')
      CALL RESET('YNONUM')

C
C write quantitive information on next page
      RED = RVELPP*RADpp*2.
      REI = 1./SQRT(TIME)
      U32 = RVELPP*TIME

      CALL PHYSOR(1.25,7.75)
      CALL AREA2D(6.25,2.5)
      CALL HEIGHT(.12)
      CALL MESSAG('\P)(I)NPUTS:\U)$',100,0.,2.25)

      CALL CMPLX

      CALL HEIGHT(.11)

      CALL MESSAG('RUN (#) $',100,5.2,2.75)
      CALL INTNO(iRUN,'ABUT','ABUT')

      CALL MESSAG('FRAME (#) $',100,5.2,2.5)
      CALL INTNO(iPLOT,'ABUT','ABUT')

      CALL MESSAG('TIME STEP (#) $',100,5.2,2.25)
      CALL INTNO(iLOOP,'ABUT','ABUT')

      CALL MESSAG('!G)/%N) = $',100,0.9,2.25)
      CALL REALNO(GAM,1,'ABUT','ABUT')

      CALL MESSAG('a/(R) = $',100,0.9,2.0)
      CALL REALNO(CtR,2,'ABUT','ABUT')

      CALL MESSAG('\!D)\P1)(t)\P2)\G1)\E1.15)*P)\EX)\G2) = $',
     1      100,0.9,1.75)
      CALL REALNO(TINC,6,'ABUT','ABUT')

      CALL MESSAG('\P1 )T \E1.2V1) \LH.7)o \EXHX ) = $',100,2.4,2.25)
      CALL REALNO(TIMEo,-6,'ABUT','ABUT')

      CALL INTNO(NMAX,2.4,2.0)
      CALL MESSAG(' X $',100,'ABUT','ABUT')
      CALL INTNO(LMAX,'ABUT','ABUT')
      CALL MESSAG(' MODES $',100,'ABUT','ABUT')

      CALL ENDGR(0)
      CALL PHYSOR(1.25,7.25)
      CALL AREA2D(6.25,2.5)
      CALL HEIGHT(.12)
      CALL MESSAG('\P)(C)URRENT (I)NFORMATION:\U)$',100,0.,1.52)
      CALL HEIGHT(.11)

C column 1:

      CALL MESSAG('\P1 )T \E1.3V1) \EXHX) = $',100,0.,1.0)
      CALL REALNO(TIME,-6,'ABUT','ABUT')

      CALL MESSAG('(R)e\LH.6(D\EXHX) = $',100,0.,.75)
      CALL REALNO(RED,1,'ABUT','ABUT')

      CALL MESSAG('(R)e\LH.6(I/\%R\EXHX) = $',100,0.,.5)
      CALL REALNO(REI,1,'ABUT','ABUT')

      CALL MESSAG('!G)/%N) = $',100,0.,.25)
      CALL REALNO(CIRC,1,'ABUT','ABUT')

C column 2:

       CALL MESSAG(
     1'\P1) (U)\E1.2V1)\EXHX)\P2)T\E1.3V2)\EXHX)\EH.7)3/2\EXHX) = $',
     2 100,2.0,1.0)
       CALL REALNO(U32,-3,'ABUT','ABUT')

       CALL MESSAG('X\LH.6)C\LXHX(/R)\LH.6)O\LXHX) = $',
     1             100,2.0,.75)
       CALL REALNO(ZCoR,-4,'ABUT','ABUT')

       CALL MESSAG(
     1'\P1)X\P2)\G1)\E1.0)*P)\EX)\G2)\LH.6)C(GQ)\LXHX) = $',
     2 100,2.0,.5)
       CALL REALNO(DCENT,-4,'ABUT','ABUT')

       CALL MESSAG('\P1) (R)\P2)\G1)\E1.05)*P)\EX)\G2) = $',100,2.0,.25)
       CALL REALNO(RADpp,4,'ABUT','ABUT')

C column 3:

       CALL MESSAG('(I/\%R \LH.6 (GQ \EXHX) = $',100,4.2,1.0)
       CALL REALNO(RIMPGQ,-6,'ABUT','ABUT')
       CALL MESSAG('\S1)  (I/\%R \EXHX) $',100,'ABUT','ABUT')

       CALL MESSAG('(I/\%R \LH.6 (MP \EXHX) = $',100,4.2,.75)
       CALL REALNO(RIMPMP,-6,'ABUT','ABUT')
       CALL MESSAG('\S1)  (I/\%R \EXHX) $',100,'ABUT','ABUT')

       CALL MESSAG('3/2 *:  (U)d(v) = $',100,4.2,.5)
       CALL REALNO(RMOM,4,'ABUT','ABUT')

       CALL ENDGR(0)

C WRITE OUT NUMERICAL INFORMATION
       CALL PHYSOR(1.25,2.00)
       CALL AREA2D(5.75,4.75)
       CALL RESET('HEIGHT')
       CALL SCMPLX

C one INFO

       CALL MESSAG('\P1)%W\E1.2)\V1)\S1)\EXVX)\P3)T\P4)\E1.2)\V3)
     1  \EXVX)\G4)\LH.7)o\LXHX)\B1)\EH.7)2\EXHX)  $',100,0.0,4.5)

c       CALL MESSAG('%W\S1)T\LH.7)o\LXHX)\B1)\EH.7)2\EXHX)  $'
c      1 ,100,0.0,4.5)

       CALL HEIGHT(0.1)
       CALL MESSAG('MAX = $',100,0.0,4.1)
       CALL REALNO(FMAX1,-3,'ABUT','ABUT')
       CALL MESSAG('MIN = $',100,0.0,3.9)
       CALL REALNO(FMIN1,-3,'ABUT','ABUT')

       CALL MESSAG('contour levels$',100,0.0,3.6)
       CALL SIMPLX
       CALL HEIGHT(0.08)
       DO 30 I = 1,NCONT1
         IF (ACONT1(I).EQ.0.) GOTO 30
         CALL REALNO(ACONT1(I),5,0.2,3.4-0.11*(I-1))
  30   CONTINUE
       CALL RESET('HEIGHT')
       CALL SCMPLX
```

```
C two INFO
      CALL MESSAG('\P1)%W\E1.2)\V1)\S1)\EXVX)\P3)T\P4)\E1.2)\V3)
     1  \EXVX)\G4)\EH.7)2\EXHX)  $',100,1.75,4.5)
c       CALL MESSAG('%W\S1)T\EH.7)2\EXHX)  $',100,1.75,4.5)

      CALL HEIGHT(0.1)
      CALL MESSAG('MAX = $',100,1.75,4.1)
      CALL REALNO(FMAX2,-3,'ABUT','ABUT')
      CALL MESSAG('MIN = $',100,1.75,3.9)
      CALL REALNO(FMIN2,-3,'ABUT','ABUT')

      CALL MESSAG('contour levels$',100,1.75,3.6)
      CALL SIMPLX
      CALL HEIGHT(0.08)
      DO 31 I = 1,NCONT2
        IF (ACONT2(I).EQ.0.) GOTO 31
        CALL REALNO(ACONT2(I),5,1.95,3.4-0.11*(I-1))
 31   CONTINUE
      CALL RESET('HEIGHT')
      CALL SCMPLX

C three INFO
      CALL MESSAG('\P1)!Y\E1.2)\V1)\S1)\EXVX)\P3)T\P4)\E1.2)\V3)
     1  \EXVX)\G4)\LH.7)o\LXHX)\B1)\EH.7)-1/2\EXHX)  $',100,3.5,4.5)
c       CALL MESSAG('!Y\S1)T\LH.7)o\LXHX)\B1)\EH.7)-1/2\EXHX)  $'
c      1            ,100,3.5,4.5)

      CALL HEIGHT(0.1)
      CALL MESSAG('MAX = $',100,3.5,4.1)
      CALL REALNO(FMAX3,-3,'ABUT','ABUT')
      CALL MESSAG('MIN = $',100,3.5,3.9)
      CALL REALNO(FMIN3,-3,'ABUT','ABUT')

      CALL MESSAG('contour levels$',100,3.5,3.6)
      CALL SIMPLX
      CALL HEIGHT(0.08)
      DO 32 I = 1,NCONT3
        CALL REALNO(ACONT3(I),5,3.7,3.4-0.11*(I-1))
 32   CONTINUE
      CALL RESET('HEIGHT')
      CALL SCMPLX

C four INFO
      CALL MESSAG('\P1)!Y\E1.2)\V1)\S1)\EXVX)\P2)T\P4)\E1.2)\V2)
     1  \EXVX)\G4)\EH.7)1/2\EXHX)/\p3)t\E1.2)\V3)\EXVX)\LH.7)o\LXHX) $
     2  ,100,5.25,4.5)
c       CALL MESSAG('!Y\S1)T\EH.7)1/2\EXHX)\S1)/T\LH.7)o\LXHX)  $',
c      1              100,5.25,4.5)

      CALL HEIGHT(0.1)
      CALL MESSAG('MAX = $',100,5.25,4.1)
      CALL REALNO(FMAX4,-3,'ABUT','ABUT')
      CALL MESSAG('MIN = $',100,5.25,3.9)
      CALL REALNO(FMIN4,-3,'ABUT','ABUT')

      CALL MESSAG('contour levels$',100,5.25,3.6)
      CALL SIMPLX
      CALL HEIGHT(0.08)
      DO 33 I = 1,NCONT4
        CALL REALNO(ACONT4(I),5,5.45,3.4-0.11*(I-1))
 33   CONTINUE
      CALL RESET('HEIGHT')
      CALL SCMPLX

      CALL ENDPL(-3)

      RETURN
      END

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       SUBROUTINE CONTXX(IDIM,JDIM,IS,IE,JS,JE,XY,F,NCOPT,NCDIM,NCONT,
     C ACONT,NCDIM,XCONT,YCONT,NADDIM,NAD,NLEV,IADIM,IA,
     C FMIN,FMAX,izero)
C
C
C
C    I GOT THIS AND THE FOLLOWING 3 SUBROUTINES FROM PIETER G. BUNIN
C
C                    (NASA AMES RESERACH CENTER)
```

```
C
C
C
C   Calculate contour lines for the function F in the region IS to IE, JS to
C   JE.  X,Y coordinates corresponding to the grid points are in array XY.
C
C   If NCOPT=1, figure our own contour levels, up to NCONT of them, using
C   "nice" numbers.  FRANGE finds the function range in the given region, an
C   CONSCL computes the contour levels.  Note that NCONT will be revised
C   downward to correspond to the number of contour levels actually used.
C   The contour levels calculated are returned in the array ACONT.
C
C   If NCOPT=2, calculate lines for the NCONT contour levels specified in
C   ACONT.
C
C   The contour lines are returned in arrays XCONT, and YCONT.  NAD(1)
C   gives the number of contour lines, and NAD(n) points to the start of the
C   nth line (i.e. NAD(n+1) points to one past the end of the nth line).
C   NLEV(n) returns the contour level of the nth contour line.
C
C   IA is a scratch array.  Try a dimension of 3000.
C
      DIMENSION XY(IDIM,JDIM,2),F(IDIM,JDIM)
      DIMENSION ACONT(NCdim)
      DIMENSION XCONT(NXDIM),YCONT(NXDIM)
      DIMENSION NAD(NADDIM),NLEV(NADDIM),IA(IADIM)
C
C   If NCOPT=1, figure our own contour levels, up to NCONT of them, using
C   "nice" numbers.  FRANGE finds the function range in the given region, an
C   CONSCL computes the contour levels.  Note that NCONT will be revised
C   downward to correspond to the number of contour levels actually used.
C
c       IF (NCOPT.EQ.1) THEN
          CALL FRANGE(IDIM,JDIM,IS,IE,JS,JE,F,FMIN,FMAX)
      IF (NCOPT.EQ.1) THEN
          CALL CONSCL(FMIN,FMAX,NCDIM,NCONT,ACONT,izero)
      ENDIF
C ***
      DO 2222 I=1,IADIM
2222  IA(I)=0
C ***
      IW=3
C
      NAD(1)= 1
      NLINEP= 2
C
C   One little check.  If IS=IE or JS=JE, return with no contour lines.
C
      IF (IS.EQ.IE .OR. JS.EQ.JE) GOTO 110
C
C   Loop through each contour level.
C
      DO 100 ICONT= 1,NCONT
        ZA= ACONT(ICONT)
      M=0
C ****  SCAN POINTS AND DETERMINE POINTS OF IA
      DO  600  J= JS+1,JE-1
      IMB=0
      DO  600  I= IS,IE
      IF(F(I,J).LE.ZA)  GO  TO  601
      IF(IMB.NE.1)  GO  TO  600
      M=M+1
      IF(M.GT.IADIM)  GO  TO  210
      IA(M)=1000*I+J
      IMB=0
      GO  TO  600
 601  IMB=1
 600  CONTINUE
C**** SEARCH START POINT ON BOUNDARY LINE
 101  IMA=1
      IMB=0
      IXA= IS-1
      IYA= JS
 1    IXA=IXA+1
      IF(IXA.EQ.IE)  IMA=2
      GO  TO  5
 2    IYA=IYA+1
      IF(IYA.EQ.JE)  IMA=3
      GO  TO  5
 3    IXA=IXA-1
      IF(IXA.EQ.IS)  IMA=4
      GO  TO  5
 4    IYA=IYA-1
      IF(IYA.EQ.JS)  IMA=5
 5    IF(F(IXA,IYA).GT.ZA)  GO  TO  7
      IMB=1
 6    GO  TO  (1,2,3,4,91),IMA
 7    IF(IMB.NE.1)  GO  TO  6
C**** DETERMINE START POINT
      IMB=0
      IX=IXA
      IY=IYA
      S=F(IXA,IYA)
      GO  TO  (21,11,12,13,51),IMA
 11   IF(IY.NE.JS)  GO  TO  31
      GO  TO  21
 12   IF(IX.NE.IE)  GO  TO  41
      GO  TO  31
 13   IF(IY.NE.JE)  GO  TO  51
      GO  TO  41
 10   IX=IA(N)/1000
      IY=IA(N)-1000*IX
      S=F(IX,IY)
      IA(N)=0
      GO  TO  21
C**** PROCESS TO SEARCH PLOT POINT
 20   IY=IY+1
 21   IX=IX-1
      IF(IX.LT.IS)  GO  TO  90
      I=1
      IF(F(IX,IY).LE.ZA)  GO  TO  52
      S=F(IX,IY)
      GO  TO  31
 30   IX=IX-1
 31   IY=IY-1
      IF(IY.LT.JS)  GO  TO  90
      I=2
      IF(F(IX,IY).LE.ZA)  GO  TO  60
      S=F(IX,IY)
      GO  TO  41
 40   IY=IY-1
 41   IX=IX+1
      IF(IX.GT.IE)  GO  TO  90
      I=3
      IF(F(IX,IY).LE.ZA)  GO  TO  60
      S=F(IX,IY)
      GO  TO  51
 50   IX=IX+1
 51   IY=IY+1
      I=4
      IF(IY.GT.JE)  GO  TO  90
      IF(F(IX,IY).LE.ZA)  GO  TO  60
      S=F(IX,IY)
      GO  TO  21
 52   IF(M.EQ.0)  GO  TO  60
      IK=1000*IX+IY+1000
      DO  602  J=1,M
      IF(IA(J).NE.IK)  GO  TO  602
      IA(J)=0
 602  CONTINUE
C**** CALCULATE PLOT POINT
 60   XYF=(ZA-F(IX,IY))/(S-F(IX,IY))
      GO  TO  (61,62,63,64),I
 61   WXX= XY(IX,IY,1)+XYF*(XY(IX+1,IY,1)-XY(IX,IY,1))
      WYY= XY(IX,IY,2)+XYF*(XY(IX+1,IY,2)-XY(IX,IY,2))
      GO  TO  65
 62   WXX= XY(IX,IY,1)+XYF*(XY(IX,IY+1,1)-XY(IX,IY,1))
      WYY= XY(IX,IY,2)+XYF*(XY(IX,IY+1,2)-XY(IX,IY,2))
      GO  TO  65
 63   WXX= XY(IX,IY,1)+XYF*(XY(IX-1,IY,1)-XY(IX,IY,1))
      WYY= XY(IX,IY,2)+XYF*(XY(IX-1,IY,2)-XY(IX,IY,2))
      GO  TO  65
 64   WXX= XY(IX,IY,1)+XYF*(XY(IX,IY-1,1)-XY(IX,IY,1))
      WYY= XY(IX,IY,2)+XYF*(XY(IX,IY-1,2)-XY(IX,IY,2))
C**** PLOT
 65   CONTINUE
C**** DECIDE IF PLOT POINT EQUAL INITIAL PLOT POINT
      IF(IW.NE.3)  GO  TO  66
      NP=1
      NAD(NLINEP)= NAD(NLINEP-1)
      NLEV(NLINEP-1)= ICONT
      NPT=0
      XCONT(NAD(NLINEP))= WXX
      YCONT(NAD(NLINEP))= WYY
      WX=WXX
      WY=WYY
      IW=2
      GO  TO  67
66    CONTINUE
      NAD(NLINEP)= NAD(NLINEP)+1
      IF  (NAD(NLINEP).GT.NXDIM)  GOTO 220
      NP=NP+1
      XCONT(NAD(NLINEP))= WXX
      YCONT(NAD(NLINEP))= WYY
C     IF  (NP .LT. 200) GO TO 6602
C
C     CALL DRAW2D(PT,NP,2,2,0)
C     NPT=NPT+NP
C     NP=1
C     PT(1,1)= WXX
C     PT(2,1)= WYY
6602  IF(WXX.NE.WX)   GO  TO  67
      IF(WYY.EQ.WY)   GO  TO  90
C****  DETERMINE NEXT PROCESS
 67   GO  TO  (50,20,30,40),I
 90   IW=3
      NAD(NLINEP)= NAD(NLINEP)+1
      IF  (NAD(NLINEP).GT.NXDIM)  GOTO 220
      IF  (NP.GT.1)  THEN
         NLINEP= NLINEP+1
         IF  (NLINEP.GT.NADDIM)  GOTO 230
      ENDIF
C     IF  (NP  .GT.  1)  CALL DRAW2D(PT,NP,2,2,0)
      IF(IMA.NE.5)   GO  TO  6
C****  SEARCH START POINT
      IF(M.EQ.0)   GO  TO  92
 91   DO  603  N=1,M
      IF(IA(N).NE.0)   GO  TO  10
 603  CONTINUE
 92   CONTINUE
C****  CALCULATE VALUE OF NEXT CURVE
 100  CONTINUE
C
  110 CONTINUE
      NAD(1)= NLINEP-2
      RETURN
C
C   Warning - IA array full.
C
  210 CONTINUE
      WRITE(6,211) IADIM
  211 FORMAT('  Warning - Scratch array IA full in contour routine ',
     C 'CONTXX.'/
     C       '  Picture may be incomplete.  Array was dimensioned ',
     C I5,'.')
      GOTO 101
C
C   Warning - XCONT array full.
C
  220 CONTINUE
      WRITE(6,221) NXDIM
  221 FORMAT('  Warning - Contour line array XCONT full in contour ',
     C 'routine CONTXX.'/
     C       '  Picture may be incomplete.  Array was dimensioned ',
     C I5,'.')
      GOTO 110
C
C   Warning - NAD array full.
C
  230 CONTINUE
      WRITE(6,231) NADDIM
  231 FORMAT('  Warning - Contour line pointer array NAD full in ',
     C 'contour routine CONTXX.'/
     C       '  Picture may be incomplete.  Array was dimensioned ',
     C I5,'.')
      GOTO 110
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE FRANGE(IDIM,JDIM,IS,IE,JS,JE,F,FMIN,FMAX)
C
C   Find the range (minimum and maximum) of the function F in the region IS
C   IE, JS to JE.
C
      DIMENSION F(IDIM,JDIM)
C
      FMIN= F(IS,JS)
      FMAX= FMIN
      DO 10 J= JS,JE
         DO 10 I= IS,IE
            FMIN= AMIN1(FMIN,F(I,J))
            FMAX= AMAX1(FMAX,F(I,J))
   10       CONTINUE
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE CONSCL(AMIN,AMAX,NCDIM,NCONT,ACONT,izero)
C
C   Come up with a "nice" scaling of about NCONT values between AMIN and AMA
C   NCONT is updated to the number of intervals actually needed.
C
      DIMENSION ACONT(NCdim)
      DIMENSION RNICE(4)
      DATA RNICE/.1,.2,.25,.5/
      DATA NNICE/4/
C
C   As a first approximation, get the difference, its characteristic and
C   mantissa.
C
      DIFF= (AMAX-AMIN)/(NCONT+1)
      IF  (DIFF.LE.0.) GOTO 20
      CHAR= ALOG10(DIFF)+1.
C
C   Round CHAR down and get the mantissa.
C
      IF  (CHAR.GE.0.) THEN
         ICHAR= CHAR
      ELSE
         ICHAR= CHAR-1.
      ENDIF
      RMANT= DIFF*10.**(-ICHAR)
C
C   What's the next largest "nice" mantissa?
C
      DO 3 I= 1,NNICE
         IF  (RMANT.LE.RNICE(I)) GOTO 10
3        CONTINUE
      I= NNICE
C
C   Got a guess.  Calculate a DIFF, round AMIN down.
C
   10 CONTINUE
      AINC= RNICE(I)*10.**ICHAR
      IMIN= AMIN/AINC
      IF  (AMIN.GT.IMIN*AINC)  IMIN= IMIN+1
      IMAX= AMAX/AINC
      IF  (AMAX.LT.IMAX*AINC)  IMAX= IMAX-1
      NNEED= IMAX+1-IMIN
C
C   Are we under?
C
      IF  (NNEED.GT.NCONT)  THEN
C
C   Nope.  Try the next nice number.
C
         IF  (I.LT.NNICE)  THEN
            I= I+1
         ELSE
            ICHAR= ICHAR+1
            I= 1
         ENDIF
         GOTO 10
      ENDIF
C
C   Now just set up the ACONT array and update NCONT.  (Set up all (original
C   NCONT of the ACONTs for the folks back home.)
C
      DO 1 I= 1,NCONT
         ACONT(I)= (IMIN-1+I)*AINC
1        CONTINUE
      NCONT= NNEED
      GOTO 30
c
c   All values are the same -- just set up one contour level.
c
   20 CONTINUE
      ACONT(1)= AMIN
      DO 2 I= 2,NCONT
         ACONT(I)= 0.
2        CONTINUE
      NCONT= 1
c
   30 CONTINUE

C Additions made by Sharon Stanaway, Oct. 11, 1987
      if (izero.eq.1) goto 40
c     This was done so that the zero contour is not plotted
      itest = 0.
      do 100 i = 1,ncont-1
        if (acont(i).EQ. 0.) itest = 1
        if (itest.EQ.1) then
          acont(i) = acont(i+1)
        end if
 100  continue
      ncont = ncont-1

 40   CONTINUE

      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine condi2 (IX,IY,LABELS,NXDIM,NADDIM,NCDIM,NCONT,XCONT,
     1                   NAD,NLEV,ACONT,BLANKC)
c
c   Draw the contours we just calculated - this involves some
c   decoding.  Also, we may or may not want labels on the contour
c   lines.
c
      logical labels
      DIMENSION XCONT(NXDIM,2),nad(NADDIM),nlev(NADDIM),ACONT(NCdim)
      DIMENSION blankc(NXDIM)
c
      if (.not.labels) goto 100
c
c   Use DISSPLA's contour utilities so we can get labels.
c
      nd= nad(1)
      if (nd.eq.0) goto 200
      call bcomon(NXDIM)
      call conbgn
      isnext= 1
      do 10 ic= 1,nd
         is= isnext
         isnext= nad(ic+1)
         call concrv(xcont(is,ix),xcont(is,iy),isnext-is,
     c     acont(nlev(ic)))
   10    continue
c
       call conend
c
c   Set some contour drawing parameters.
c
       call conlin(0,'SOLID','LABELS',1,1)
       call conmin(2.5)
       call conang(40.)
       call height(.08)
c
c   Now actually draw them.
c
       call contur(1,'LABELS','DRAW')
       call reset('HEIGHT')
       goto 200
c
c   No labels on the contour lines - this is easy.
c
  100 continue
      nd= nad(1)
c      write(6,1000) nd
c 1000 format(x,'number of contour lines, nd(1) = ',i2)
      if (nd.eq.0) goto 200
      isnext= 1
      do 110 ic= 2,nd+1
         is= isnext
         isnext= nad(ic)
c WARNING: This line has been changed for particular needs to the user
c         IF(ACONT(NLEV(IC-1)).EQ.0.)GOTO 110
c         IF(ACONT(NLEV(IC-1)).LT.0.)CALL DASH
         call curve(xcont(is,ix),xcont(is,iy),isnext-is,0)
         CALL RESET ('DASH')
  110    continue
c
  200 continue
      return
      end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

References

Arfken, G. 1985 *Mathematical methods for physicists.* Academic Press.

Abramowitz, M., Stegun, I.A. 1972 *Handbook of Mathematical Functions.* Dover, New York, N.Y.

Batchelor, G. K. 1967 *Intoduction to fluid dynamics.* Cambridge University Press.

Blatt, J. M. & Weisskhopf, V. F. 1952 *Theoretical Physics.* Appendix A & B, John Wiley & Sons, New York.

Brasseur, J. G. 1986 Evolution charactericstics of vortex rings over a wide range of Reynolds numbers. Dept. Mech. Eng., Clemson U., Clemson, SC.

Cantwell, B. J. 1981 Organized motion in turbulent flow. *Ann. Rev. Fluid Mech.* **13**, 457-515.

Cantwell, B. J. 1986 Viscous staring jets. *J. Fluid Mech.* **173** 159-189.

Cain, A. B., Ferziger, J. H. & Reynolds, W. C. 1984 Discrete orthogonal function expansions for non-uniformm grids using the Fast Fourier Transform. *J. Comp. Phys.* **56**, **2**, 272-286.

Didden, N. 1979 On the formation of vortex rings: rolling-up and production of circulation. *J. App. Math. and Phys. (ZAMP)*, **30**, 101-116.

Fraenkel, L. E. 1970 On steady vortex rings of small cross section in an ideal fluid. *Roy. Soc. London* A **316**, 29.

Fraenkel, L. E. 1972 Examples of steady vortex rings of small cross-section in an ideal fluid. *J. Fluid Mech.* **51**, 119-135.

Golub, G. H. & Welsch, J. H. 1969 Calculation of Gauss quadrature rules. *Math. of Comp.* **29**, 221-230.

Golub, G. H. & van Loan, C. F. 1983 *Matrix computations.* The Johns Hopkins University Press, Baltimore, MD.

Gottlieb, D. & Orszag, S.A. 1977 *Numerical analysis of spectral methods*, NSF-CMBS, monograph 26, Soc. Ind. and Appl. Math., Philadelphia, Pa.

Glezer, A. 1981 An experimental study of a turbulent vortex ring. Ph.D. thesis, Graduate Aeronautical Laboratories, California Institute of Technology.

Gray 1914 Notes on Hydrodynamics. *Phil. Mag.*, 6, xxviii, 13.

Hicks, W. M. 1885 Researches on the theory of vortex rings. *Phil. Trans. R. Soc.* **176**, 725-779.

Hill, L.H. 1953 The theory of vector spherical harmonics, *Am. J. Phys.* **22**, 211-214.

Hill, M. J. M. 1894 On a spherical vortex. *Phil. Tran. R. Soc.* A **185**, 213-245

Kambe, T., Oshima, Y. 1975 Generation and decay of viscous vortex rings. *J. Phys. Soc. Japan* **38**, 271-280.

Kelvin, Lord 1880 Vibrations of a columnar vortex. *Phil. Mag.* **10**, 155.

Knio, O. M., Ghoniem, A. F. 1988 On the formation of streamwise vorticity in turbulent shear flow. AIAA-88-0728, AIAA 26th Aerospace Sciences Meeting.

Kojima, M., Hinch, E. J., Acrivos, A. 1984 The formation and expansion of a toroidal drop moving in a viscous fluid. *Phys. Fluids* **27** 19-32.

Kovasznay, L. S., Fujita, H., Lee, R. L. 1973 Unsteady turbulent puffs. *Advances in Geophysics* **18**, proceedings IUTAM-IUGG Symposium on Turb. Diffusion, Charlotsville, VA.

Krutzsch, C. H. 1939 *Ann. Phys.* **35**, 497.

Lamb, H., 1945 *Hydrodynamics*, Dover, New York, N.Y.

Leonard, A. 1981 Divergence-free vector expansions for 3-D flow simulations. Bull. Amer. Phys. Soc. **26**, 1274.

Leonard, A., Wray, A. 1982 A new numerical method for the simulation of three-dimensional flow in a pipe. *NASA* T.M. 84267.

Leray, J. 1934 Essai sur le mouvement d'un liquide visqueux emplissant l'espace. *Acta Math.* **63**, 193.

Liess, C., Didden, N. 1976 *Z. Angew. Math. Mech.* **56**, T206.

Magarvey, R. H., MacLatchy, C. S. 1964 The formation and structure of vortex rings. *Canadian J. Phys.* **42**, 678-683.

Maxworthy, T. 1972 The structure and stability of vortex rings. *J. Fluid Mech.* **51**, 15-32.

Maxworthy, T. 1974 Turbulent vortex rings. *J. Fluid Mech.* **64**, 227-239.

Maxworthy, T. 1977 Some experimental studies of vortex rings. *J. Fluid Mech.* **81**, 465-495.

Maxworthy, T. 1979 Comments on "Preliminary study of mutual slip-through of a pair of vortices". *Phys. Fluids* **22**, 200.

Moser, R. D., Moin, P., Leonard, A. 1983 A spectral numerical method for the Navier-Stokes equations with applications to Taylor-Couette flow. *J. Comp. Phys.* **52**, 524-544.

Moser, R. D., Moin, P. 1984 Direct numerical simulation of curved channel flow. Rep. no. TF-20, Thermosciences Div., Dept. Mech. Eng., Stanford U., Stanford, CA.

Norbury, J. 1973 A family of steady vortex rings. *J. Fluid Mech.* **57**, 417-431.

Oseen, C. W. 1910 *Ark. f. Mat. Astr. og Fys.* **6**, no. 29.

Oshima, Y. 1972 Motion of vortex rings in water. *J. Phys. Soc. Japan* **32**, 1125-1131.

Oshima, Y., Kambe, T., Asaka, S. 1975 Interaction of two vortex rings moving along a common axis of symmetry. *J. Phys. Soc. Japan* **38**, 1159-1166.

Phillips, O. M. 1956 The final period of decaying non-homogeneous turbulence. *Proc. Cambridge Phil. Soc.* **52**, 135-151.

Prandtl, L. & Tietjens, O. G. 1934 *Fundamentals of hydro and aeromechanics.* Dover.

Reynolds, O. 1876 *Nature* **14**, 477.

Rott, N., Cantwell, B. J. 1987 The decay of viscous vortex pairs. *Am. Phys. Soc.* Meeting - Div. of Fl. Mech., Eugene, OR., 2038.

Saffman, P. G. 1970 The velocity of viscous vortex rings. *Studies Appl. Math.* **XLIX**, 371-380.

Saffman, P. G. 1978 The number of waves on unstable vortex rings. *J. Fluid Mech.* **84**, 625-639.

Sallet, D. W., Widmayer, R. S. 1974 An experimental investigation of laminar and turbulent vortex rings in air. *Z. Flugwiss.* **22**, Heft 6, 207-215.

Shariff, K. 1987 Private communication.

Shariff, K., Leonard, A., Zabusky, N., Ferziger, J. 1988 Acoustics and dynamics of coaxial interacting vortex rings. *Fluid Dynamics Research* **3**.

Sommerfeld, A. 1950 *Mechanics of deformable bodies.* Academic.

Spalart, P. R. 1986 Numerical simulation of boundary layers: Part 1. Weak Formulation and Numerical Method. *NASA* TM-88222.

Spalart, P. R. 1988 Coordinate singularities and spectral methods. private communication.

Stanaway, S. K., Cantwell, B. J., Spalart, P. R. 1986 A full simulation of a vortex ring. 9th Australasian Fluid Mechanics Conference, Auckland.

Stanaway, S. K., Cantwell, B. J., Spalart, P. R. 1988 AIAA-88-0318: Navier-Stokes simulations of axisymmetric vortex rings. AIAA 26th Aerospace Sciences Meeting, Reno, Nevada.

Sullivan, J. P., Widnall, S. E. & Ezekiel, S. 1973 A study of vortex rings using a laser-Doppler velocimeter. *AIAA J.* **11**, 1384-1389.

Thompson, J.J. 1883 *A treatise on the motion of vortex rings.* London, MacMillan.

Thompson, Sir W. 1867 On vortex atoms. *Phil. Mag.* **34**, 15-24.

Thompson, Sir W. 1867 *Phil. Mag.* 4, xxxiii, (papers, iv. 67), p. 511.

Ting, L. 1981 Studies on the motion and decay of a vortex filament. *Advances in Fluid Mechanics*, **148**, 67-105, Spriger-Verlag.

Ting, L., Tung, C. 1967 Motion and decay of a vortex filament in a non-uniform stream. *Physics of Fluids*, **8**, 1039-1051.

Tung, C., Ting, L. 1967 Motion and decay of a vortex ring. *Phys. Fluids* **10**, 901-910.

Van Dyke, M. 1975 *Perturbation methods in fluid mechanics.* Parabolic Press, Stanford, CA.

Widnall, S.E. 1975 The structure and dynamics of vortex filaments. *Annual Review of Fluid Mechnics*, pp. 141-165, Annual Review, Palo Alto, CA.

Widnall, S.E. 1977 The instability of the thin vortex ring of constant vorticity. *Phil. Trans. Roy. Soc.*, **287**, 273-305.

Widnall, S.E., Sullivan, J. P. 1973 On the stability of vortex rings. *Proc. R. Soc. Lond. A*, **332**, 335-353.

Wilkinson, J. H. 1968 Almost diagonal matrices with multiple or close eigenvalues. *Lin. Alg. & Its Applic. 1*, 1-12.

Whitehead, K. D. 1968 The generation and development of a viscous vortex ring. Georgia Institute of Technology, School of Aerospace Eng., Rep. 68-4.

Yamada, H., Matsui, T. 1978 Preliminary study of mutual stip-through of a pair of vortices. *Phys. Fluids* **21**, 292.

Yamada, H., Matsui, T. 1979 Reply to comments of T. Maxworthy. *Phys. Fluids* **22**, 201.

Yamada, H., Matsui, T. 1979 Mutual slip through of a pair of vortex rings. *Phys. Fluids* **22**, 1245-1249.

Yamada, H., Matsui, T. 1980 Visualization of vortex interaction using smoke-wire technique. *Flow Visualization II*, Hemisphere Publ. Corp., MaGraw-Hill, 355-359.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.
NASA TM 101041

2. Government Accession No.

3. Recipient's Catalog No.

4. Title and Subtitle
A Numerical Study of Viscous Vortex Rings Using a Spectral Method

5. Report Date
October 1988

6. Performing Organization Code

7. Author(s)
S. K. Stanaway, B. J. Cantwell (Stanford University, Stanford, CA), and P. R. Spalart

8. Performing Organization Report No.
A-88306

9. Performing Organization Name and Address
Ames Research Center
Moffett Field, CA 94035

10. Work Unit No.
505-60

11. Contract or Grant No.

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Washington, D.C. 20546-0001

13. Type of Report and Period Covered
Technical Memorandum

14. Sponsoring Agency Code

15. Supplementary Notes
Point of Contact: Philippe Spalart, Ames Research Center, MS 202A-1, Moffett Field, CA 94035
(415) 694-4734 or FTS 464-4734

16. Abstract

Viscous, axisymmetric vortex rings are investigated numerically by solving the incompressible Navier-Stokes equations using a spectral method designed for this type of flow. The results presented are axisymmetric, but the method is developed to be naturally extended to three dimensions. The spectral method relies on divergence-free basis functions. The basis functions are formed in spherical coordinates using Vector Spherical Harmonics in the angular directions, and Jacobi polynomials together with a mapping in the radial direction. Simulations are performed of a single ring over a wide range of Reynolds numbers ($Re \equiv \Gamma/\nu$), $0.001 \leq Re \leq 1000$, and of two interacting rings. At large times, regardless of the early history of the vortex ring, it is observed that the flow approaches a Stokes solution that depends only on the total hydrodynamic impulse, which is conserved for all time. At small times, from an infinitely thin ring, the propagation speeds of vortex rings of varying Re are computed and comparisons are made with the asymptotic theory by Saffman. Our results are in agreement with the theory; furthermore, the error is found to be smaller than Saffman's own estimate by a factor $\sqrt{\nu t/R^2}$ (at least for $Re = 0$). The error also decreases with increasing Re at fixed core-to-ring radius ratio, and appears to be independent of Re as $Re \to \infty$). Following a single ring, with $Re = 500$, the vorticity contours indicate shedding of vorticity into the wake and a settling of an initially circular core to a more elliptical shape, similar to Norbury's steady inviscid vortices. Finally, we consider the case of "leapfrogging" vortex rings with $Re = 1000$. The results show severe straining of the inner vortex core in the first pass and merging of the two cores during the second pass.

17. Key Words (Suggested by Author(s))
Vortex rings
Navier-Stokes equation
Spectral method

18. Distribution Statement
Unlimited - Unclassified

Subject Category: 34

19. Security Classif. (of this report)
Unclassified

20. Security Classif. (of this page)
Unclassified

21. No. of pages
167

22. Price
A08

For sale by the National Technical Information Service, Springfield, Virginia 22161