



PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.

DATE DUE	DATE DUE	DATE DUE
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

**NUMERICAL SIMULATION OF THE AIR AND FUEL
MIXING PROCESS IN FUEL-INJECTED ENGINES
USING DISCRETE VORTEX DYNAMICS**

By

Asuquo Bassey Ebiana

A DISSERTATION

**Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of**

DOCTOR OF PHILOSOPHY

Department of Mechanical Engineering

1990

ABSTRACT

NUMERICAL SIMULATION OF THE AIR AND FUEL MIXING PROCESS IN FUEL-INJECTED ENGINES USING DISCRETE VORTEX DYNAMICS

By

Asuquo Bassey Ebiana

The numerical modeling of turbulent combustion problems continues to invite the application of novel analytical techniques. Contemporary numerical models seek to utilize the observable eddy structures of the turbulent flow to simulate the physical systems directly without recourse to conventional finite difference formulation. Vortex methods represent such a class of numerical techniques. Vortex methods are particularly appealing because they are suitable for use in high Reynolds number flows without the problems of spatial resolution, stability and numerical viscosity associated with finite difference schemes.

The discrete vortex method is used in this study to simulate the mixing process prior to chemical reaction in the fuel-injected cylinder of a piston engine. Mixing descriptions are restricted to a two-dimensional, incompressible, inviscid, formulation of the bulk diffusion process. The analysis employs the fundamental

conservation equations of mass, momentum and scalar species with appropriate simplifying assumptions. Although emphasis is limited to inviscid calculations, the the theoretical and numerical considerations relating viscous and molecular diffusion effects are fully addressed. The incorporation of viscous and molecular diffusion effects will be the focus of a future study.

In the present study, the initial vorticity distribution inside the cylinder is approximated by discrete fluid vortices each of quasi Gaussian-shaped distribution. The initialization scheme makes use of prescribed velocity data and an imaging system for the vortices found by modifying the circle theorem of Milne-Thomson. Fuel injection is simulated using a discrete jet model analogous to Ashurst's scheme. At each time step, a grid-insensitive Vortex-In-Cell (VIC) solution technique is used to approximate the velocity field inside the cylinder. Subgrid accuracy is achieved through quadratic spline interpolation using Couet's numerical filter. Each vortex is advanced every time step using Heun's second order accurate time integration scheme.

Predictions of the numerical studies show excellent agreement with theoretical results obtained using the method of images. Plausible, qualitative flow patterns consistent with theory and intuition are illustrated. The primary conclusion of this study is that optimum condition is reached for a 60° injection angle, swirl rate proportional to $0.125 \text{ m}^2/\text{s}$ and 1.26 m/s injection speed. The artificial complete mixing time for this optimum condition is estimated to be 0.159 sec . The implication that there is an optimum swirl strength for minimum complete mixing time could be of significance in engine design.

**Dedicated to the memory of my father, Bassey Esang Ebiana,
who instilled in me the value of education.**

ACKNOWLEDGEMENTS

I would first like to express my deepest gratitude to Professor Merle C. Potter, my dissertation advisor, for the unselfish aid he has given me during the research and writing of this dissertation. Without his invaluable insight, new ideas, direction and guidance, much of the intricacies of this project would have been difficult to unravel. I am very much appreciative of the generous disposition of his time. Thanks are also due Professors James V. Beck, Mahlon C. Smith and David H. Y. Yen for serving on my committee. Their comments, corrections and suggestions are well noted and appreciated.

I am deeply indebted to the Mathematics Department for its generous offer of graduate assistantships without which I would not have been able to sustain myself and family financially this far. Special appreciation is extended to Barbara Miller, Kathy Trebilcote, Dr. Elizabeth Phillips, Professor Charles Seebeck and Professor Jacob Plotkin for their sensitivity to my financial plight. I am also deeply appreciative of the department's recognition of my efforts by way of the Excellence-In-Teaching citation award accorded me. I wish to thank my department, the Mechanical Engineering Department, too for its financial support via laboratory teaching opportunities. In particular, the Fluid Dynamics Laboratory teaching experience was quite rewarding and I am grateful to Professor John F. Foss for refurbishing my undergraduate background with the fundamentals of experimental fluid dynamics and for his guidance and direction.

Special thanks are extended to Dr. Kenneth Ebert of the Office for International Students and Scholars for his kind disposition and professional understanding of foreign students' concerns. Appreciations are extended too to the staff of the Case Center for Computer Aided Engineering who acquainted me with the VAX/VMS, SUN and HP computer systems and helped me with system related problems. In particular, Mike Mcpherson, Margaret Wilke, Fred Hall, Greg Fell, Dan Mcknown and George Lupanoff deserve to be specially acknowledged. I would also like to thank Mr. Tom Volkening, the chief librarian for his help in searching for literature materials for me. Thanks too to my colleagues Dr. Bashar AbdulNour and Dr. Syed Ali for their friendship throughout the years.

I wish to thank my mother, Mrs. Ukpong Bassey Ebiana for all she has done for me. My parents-in-law Elder and Mrs. Ene Okon Iban and family have been great sources of financial and emotional support and for this I am indeed very sincerely thankful. My late sister-in-law Carol Okpok Ene Iban (may God bless her soul) deserves special acknowledgement for her relentless efforts to encourage and spur me on. My cousins Mr. Nweme Eyo Eyo, Mr. Asuquo Eyo Eyo and Mr. Abang Okon Eyo and their families have also been emotionally very supportive and I thank them for their support. My sincere thanks and deep appreciation to a very special friend Dr. Abiodun Oyewale, for his encouragement and incredible financial sacrifice in my behalf. I am especially and deeply indebted to my wife, Bassey and daughter Victoria, for their encouragement in the face of great odds, emotional support, sacrifice, patience and understanding throughout the duration of my study.

Above all I thank God Almighty for His care, protection and guidance always.

TABLE OF CONTENTS

	Page
LIST OF TABLES	viii
LIST OF FIGURES	ix
NOMENCLATURE	xiii
INTRODUCTION	1
CHAPTERS	
1. PREVIOUS INVESTIGATIONS	8
1.1 Flow Field	8
1.2 Mixing Models	11
2. DESCRIPTION OF PROBLEM	14
3. THE DISCRETE VORTEX METHOD	19
3.1 Introduction	19
3.2 The Flow Field	20
3.2.1 The interior region	23
3.2.2 Boundary layer region	28
3.2.3 Intake flow	34
3.2.4 Summary	35
3.3 Exact Solution	36

3.4	Mixing	39
3.5	Choice of Numerical Parameters and Functions	46
4.	NUMERICAL FILTERS	47
4.1	Introduction	47
4.2	Description of Filters	47
4.3	Analysis of Filter Performance	51
4.4	Summary	54
5.	NUMERICAL METHOD	56
5.1	Conformal Transformation	56
5.1.1	Grid transformation	56
5.1.2	Velocity transformation	59
5.2	Point Vortex Initialization Scheme	61
5.3	Vortex-In-Cell (VIC) Algorithm	63
5.4	Vortex Sheet Algorithm	66
5.5	Advection Scheme	68
5.6	Sorting Algorithm	71
5.7	Mixing Algorithm	72
6.	NUMERICAL EXPERIMENTS AND DISCUSSION OF RESULTS	74
6.1	Introduction.....	74
6.2	Optimization of Numerical Parameters	75
6.2.1	Mesh size	75
6.2.2	Injector nozzle diameter	77
6.2.3	Vortex core radius	78

6.2.4	Time step	81
6.3	Validation of Numerical Codes	83
6.3.1	Conformal transformation.....	83
6.3.1.1	Grid transformation	83
6.3.1.2	Velocity transformation	84
6.3.1.3	Corner singularities	85
6.3.2	Point vortex initialization.....	86
6.3.3	Velocity field and vortex trajectories	87
6.3.3.1	Velocity field	87
6.3.3.2	Vortex trajectories	88
6.4	Numerical Simulations	91
6.4.1	Swirling motion with no jet flow	95
6.4.2	Jet flow with no swirling motion	96
6.4.3	Swirling motion with jet flow	97
7.	SUMMARY AND CONCLUSION	99
8.	RECOMMENDATIONS	103
TABLES		106
FIGURES.....		113
APPENDICES		
A.	Numerical Filters	168
A.I	Derivation of Christiansen's filter as an illustrative example	168
B.	Functions	173
B.I	Smoothing function G_{ij} for convergence method [39]	173

B.II	Derivation of the streamfunction ψ inside a finite rectangular domain using method of images	174
C.	Coordinate Transformation	179
C.I	Square $(\zeta, \eta) \rightarrow$ Circle (x, y)	179
C.II	Circle $(x, y) \rightarrow$ Square (ζ, η)	181
D.	Evaluating the Strengths of the Point Vortices	187
E.	List of Computer Programs	189
	BIBLIOGRAPHY	286

LIST OF TABLES

- Table 1. One-dimensional filter models.
- Table 2. Dyer's experimental data.
- Table 3. Comparison of the absolute value of the relative error (%) between the modified exact and numerical tangential velocity values at grid points.
- Table 4. Error statistics for grid point coordinate and velocity transformations.
- Table 5. Velocity field initialization data.
- Table 6. L , θ_j optimization data (swirl strength $S_\theta = 12.5 \text{ cm}\cdot\text{m/s}$).
- Table 7. N_c optimization data (swirl strength $S_\theta = 12.5 \text{ cm}\cdot\text{m/s}$).
- Table 8. Optimization data for swirl strength S_θ .

LIST OF FIGURES

- Figure 1.** Three phases of heat release.
- Figure 2.** Sketch of intake flow.
- Figure 3.** Numerical filters for vorticity deposition on the grid and for interpolation of local effects of the velocity field from the grid to the Lagrangian points.
- Figure 4.** Vorticity distribution in a grid, with shading showing assignment of vorticity to grid point for VIC method.
- Figure 5.** Iso-vorticity contours as represented by the various filters. Large tick marks represent a half-grid length.
- Figure 6.** Fourier transform of the various filters.
- Figure 7.** Inverse Fourier transform of the various filters.
- Figure 8.** Error estimates between original filter and the inverse of its transform.
- Figure 9.** Grid transformation.
- Figure 10.** Quadratic spline vorticity distribution of three typical vortices on their nearby grid points.
- Figure 11.** Zone of influence of a vortex sheet.

- Figure 12.** Vortex sorting.
- Figure 13.** Relative error between exact and numerical streamfunction values in the square domain using a single vortex and a system of 105 vortices.
- Figure 14.** Relative error between the initial and final positions of a point vortex after one complete revolution.
- Figure 15.** Radial distribution of tangential velocity component.
- Figure 16.** Geometric distribution of the system of 115 discrete vortices ($\tau_0=100$ ms).
- Figure 17.** Relative errors in V_r and V_θ velocity components as a function of time.
- Figure 18.** Trajectories of systems of discrete vortices.
- Figure 19.** Tangential velocity profiles at discrete time steps representative of short and long time errors.
- Figure 20.** Radial velocity profiles at discrete time steps representative of short and long time errors.
- Figure 21.** Uniform grid squares superimposed on circular domain.
- Figure 22.** Complete mixing time as a function of the number of cells (Swirl strength $S_\theta=12.5$ cm·m/s, Number of vortices per cell $N_c=2$).
- Figure 23.** Complete mixing time as a function of the inlet jet velocity (Swirlstrength $S_\theta=12.5$ cm·m/s, Number of vortices per cell $N_c=2$).

- Figure 24.** Average complete mixing time as a function of the number of cells (Swirl strength $S_\theta=12.5 \text{ cm}\cdot\text{m/s}$. Number of vortices per cell $N_c=2$).
- Figure 25.** Complete mixing time as a function of the number of vortices per cell (Swirl strength $S_\theta=12.5 \text{ cm}\cdot\text{m/s}$. Number of cells $L=32$).
- Figure 26.** Complete mixing time as a function of inlet jet velocity (Swirl strength $S_\theta=12.5 \text{ cm}\cdot\text{m/s}$. Number of cells $L=32$).
- Figure 27.** Complete mixing time as a function of the swirl strength (Number of cells $L=32$. Number of vortices per cell $N_c=2$).
- Figure 28.** Complete mixing time as a function of the inlet jet velocity (Number of cells $L=32$. Number of vortices per cell $N_c=2$).
- Figure 29.** Geometric distribution of the system of 115 discrete vortices and velocity vector representation of the induced swirling motion.
- Figure 30.** Fuel particle trajectory and vorticity profile at $y=0$ as a function time. Injector at 60 degrees (counterclockwise from horizontal). $U_{in}=210.0 \text{ cm/s}$
- Figure 31.** Streamlines and velocity profiles at $y=0$ as a function of time (without swirl). Injector at 60 degrees (counterclockwise from horizontal). $U_{in}=210.0 \text{ cm/s}$.
- Figure 32.** Air and Fuel particle trajectory and vorticity profile at $y=0$ as a function of time. Injector at 60 degrees (counterclockwise from horizontal). $U_{in}=210 \text{ cm/s}$.
- Figure 33.** Streamlines and velocity profiles at $y=0$ as a function of time (swirl). Injector at 60 degrees (counterclockwise from horizontal). $U_{in}=210.0 \text{ cm/s}$.

- Figure 34. Fuel particle trajectory and vorticity profile at $y=0$ as a function time. Injector at 90 degrees (counterclockwise from horizontal). $U_{in}=210.0$ cm/s.
- Figure 35. Streamlines and velocity profiles at $y=0$ as a function of time (without swirl). Injector at 90 degrees (counterclockwise from horizontal). $U_{in}=210.0$ cm/s.
- Figure 36. Air and Fuel particle trajectory and vorticity profile at $y=0$ as a function of time. Injector at 90 degrees (counterclockwise from horizontal). $U_{in}=210$ cm/s.
- Figure 37. Streamlines and velocity profiles at $y=0$ as a function of time (swirl). Injector at 90 degrees (counterclockwise from horizontal). $U_{in}=210.0$ cm/s.
- Figure 38. Fuel particle trajectory and vorticity profile at $y=0$ as a function time. Injector at 120 degrees (counterclockwise from horizontal). $U_{in}=210.0$ cm/s.
- Figure 39. Streamlines and velocity profiles at $y=0$ as a function of time (without swirl). Injector at 120 degrees (counterclockwise from horizontal). $U_{in}=210.0$ cm/s.
- Figure 40. Air and Fuel particle trajectory and vorticity profile at $y=0$ as a function of time. Injector at 120 degrees (counterclockwise from horizontal). $U_{in}=210$ cm/s.
- Figure 41. Streamlines and velocity profiles at $y=0$ as a function of time (swirl). Injector at 120 degrees (counterclockwise from horizontal). $U_{in}=210.0$ cm/s.
- Figure 42. Vorticity distribution in a grid, with shading showing assignment of vorticity to grid point for VIC method.
- Figure 43. Method of Images.

NOMENCLATURE

English Letter Symbols

- a - Radius of circle in Milne-Thomson's Circle Theorem.
- C, C_1, C_2 - Constants.
- c, s, d - Jacobian elliptic functions with real arguments (also c_1, s_1, d_1).
- cn, sn, dn - Jacobian elliptic functions with complex arguments.
- D - Finite circular domain representing the combustion chamber.
- $\frac{D}{Dt}$ - Total derivative.
- d - Distance between mesh points.
- dA_{ij} - Area element of vorticity assigned to the grid point (i, j) .
- d_j - Injector nozzle diameter.
- e_{11}, e_{12} - Elements of a 2×2 matrix.
- F - Matrix of geometric factors.
- $f(z)$ - Complex potential function.
- $f(\Phi, t)$ - Distribution function.
- G_{ij} - Green's function or vortex core smoothing function
- h - Boundary discretization length. Also, length of vortex sheet segment.

i	- Grid index in the x (or ζ) direction.
j	- Grid index in the y (or η) direction.
K	- The complete elliptic integral of the first kind. Also, dimension of square domain.
k	- Modulus. Also, wave number.
L	- Number of square meshes or cells within the circular domain.
l	- Initial number of vortex sheet segments created at wall test points.
m	- Vortex sheet integer tags. Also number of dimensions.
M	- Number of grid points in either ζ or η directions.
N	- Number of vortex elements or fluid particles.
N_c	- Number of vortex elements required per cell.
n	- Number of indices.
$\mathbf{n}$	- Unit normal vector on S .
P	- Averaged joint probability distribution function.
p	- Pressure of fluid. Also, instantaneous joint probability distribution function and parameter of interest.
R	- Radius of combustion chamber.
Re	- Reynolds number.
r_0	- Discrete vortex core radius.
r^*	- Dimensionless coordinate r/R .
S	- Matrix of discrete point vortex strengths.
S	- Boundary of circular domain.
S_θ	- Swirl strength.

S'	- Boundary of square domain.
s	- Unit tangent vector on S .
t	- Time.
t^*	- Artificial time characterizing complete mixing.
U_{in}	- Inlet jet velocity.
U_{δ_s}	- Velocity at the outer edge of the numerical shear layer (i.e., velocity at "infinity").
u	- Velocity component in the x-direction.
v	- Velocity component in the y-direction.
$V=(u,v)$	- Velocity vector.
V_r	- Radial velocity component.
V_θ	- Tangential velocity component.
$\overline{V_\theta}$	- Average tangential velocity.
V_θ	- Matrix of tangential velocity components.
$W(z)$	- Complex potential function.
$\mathbf{x}=(x,y)$	- Position vector.
(x,y)	- Cartesian coordinates in the circular domain.
$z=x + iy$	- Space coordinate in the complex circular plane.

Symbols

α	- Dummy variable.
β	- Mixing frequency. Also average distance between vortices.

Δt	- Length of time step.
Δ_x, Δ_y	- Half-widths for the vorticity distribution in Meng/Thomson's filter.
δ_s	- Numerical shear layer thickness.
δ	- Dirac delta function.
∇	- The del operator.
∇^2	- The Laplacian operator.
Γ	- The coefficient of molecular diffusion.
Γ_t	- Turbulent diffusion coefficient.
γ_p	- Discrete point vortex circulation.
γ_s	- Vortex sheet circulation.
$\gamma_{\max}$	- Maximum allowable vortex circulation.
$\vec{\eta}(\eta_1, \eta_2)$	- Vector with Gaussianly distributed random variables.
$(\eta_1)_j$	- x-component random walk due to vortex j.
$(\eta_2)_j$	- y-component random walk due to vortex j.
O	- Of the order of.
$\kappa = \frac{\gamma_p}{2\pi}$	- Discrete point vortex strength.
k/π	- Reduced wave number.
μ	- Mean.
ν	- Kinematic viscosity of fluid.
θ_j	- Angle of injection.

- $\omega=\zeta+i\eta$ - Space coordinate in the complex square plane.
- $\Phi(\phi_1, \dots, \phi_\sigma)$ - Set of σ -dimensional scalars.
- ϕ - Velocity potential. Also, scalar function.
- φ - Filter function.
- ρ - Density of fluid.
- σ - Number of independent scalar species. Also, standard deviation (variance = σ^2).
- $\Psi(\psi_1, \dots, \psi_\sigma)$ - Set of σ -dimensional probability spaces corresponding to Φ .
- Ψ' - Dummy variable.
- ψ - Stream function. Also probability space.
- ξ - z-component vorticity.
- (ζ, η) - Non-dimensional cartesian coordinates in the square domain.

Subscripts

- 0 - Initial parameter.
- 1 - For constants and functions.
- α - Variable parameter, $\alpha = 1, 2, \dots, \sigma$.
- A - Valve lip region.
- B - Valve seat region.
- i - Index in the x(or ζ) direction.
- j - Index in the y(or η) direction. Also, jet.

p	- Discrete point vortex.
pt.	- Piston.
r	- Radial component.
s	- Vortex sheet. Also, shear layer, and source.
t	- Turbulent.
θ	- Tangential.
ξ	- Vorticity.

Superscripts

n	- Time step number.
σ	- Number of scalar species.
t	- Turbulent.
—	- Mean.
'	- Fluctuating component. Also, transformed boundary.

Abbreviations

C.A.	- Crank Angle.
C/D	- Coalescence and Dispersion.
CIC	- Cloud-In-Cell.
CPU	- Central Processor Unit.
ESCIMO	- Engulfment, Stretching, Coherent, Inter-diffusion, Moving, Observer.

FFT	-	Fast Fourier Transform.
FISHPAK	-	A portable elliptic boundary value problem solver.
HP	-	Hewlett Packard.
IMSL	-	International Mathematical and Statistical Libraries.
LDV	-	Laser-Doppler Velocimetry.
NCAR	-	National Center for Atmospheric Research (Graphics package).
PDF	-	Probability Density Function.
PLOTTT	-	Interactive Graphics and Statistics.
TDC	-	Top Dead Center.
2D	-	Two dimensions.
VAX/VMS	-	Virtual Architecture Extended/Virtual Memory System.
VIC	-	Vortex-In-Cell.

INTRODUCTION

In recent years, growing concern over automobile emissions and fuel economy has spurred great demand for further improvement of the internal combustion engine. Since efficiency and emissions are influenced by the details of the combustion process, it is logical that much of the recent engine research has focused on this process. The important details of combustion include processes such as mixing, chemical kinetics, vaporization, and so on. These processes are aided and abetted by turbulent motions which produce large concentration gradients and large surface areas of interface between fuel and air. To model these processes more accurately, it is necessary to incorporate new ideas and approaches as our understanding of fluid motion, turbulent structures, and their effects on combustion in an engine improves. The current knowledge of the above processes is limited, and so a precise prediction of these events in an engine has yet to be realized.

Turbulent motion of fluids pose problems of great mathematical and numerical complexity. The theoretical analysis of turbulent motion has traditionally been by statistical methods. Because the mechanism of turbulent mixing is intricately linked to the dynamics of turbulent motion, the statistical theory of turbulent mixing has been developed parallel to the statistical theory of the turbulent motion problem. In either case, the choice of a statistical approach leads to a closure problem which

requires arbitrary postulates of the unknown terms. In turbulent motion, the closure is a direct consequence of the non-linearity of the Navier-Stokes equations. However, in turbulent mixing it is caused by the non-linearity in the stochastic variables even though the scalar conservation equation is essentially linear. Because randomness and non-linearity combine to make the equations of turbulence nearly intractable, turbulence theory suffers from the absence of sufficiently powerful mathematical methods. In the absence of a completely formal mathematical theory, experimental measurements and computer modeling of the actual behavior in a firing engine have proven to be invaluable tools in the engine design process.

Description of experimental methods to measure and observe turbulent fluid motion in an engine cylinder abound in the literature. These methods fall into either of two broad classifications: direct measurements or Schlieren photography. Direct measurements include the use of hot-wire anemometers for time-resolved temperature distributions, continuous-wave Raman spectroscopy for fuel and air concentrations, pulsed laser Raman spectroscopy for density fluctuations, Laser-Doppler Velocimetry (LDV) for the velocity and turbulence intensity and more conventional measuring devices for the cylinder pressure. The single point measurement and directional ambiguity of the hot-wire anemometer, the large pressure and temperature corrections that must be applied to hot-wire measurements and the fact that hot wires do not survive the combustion process present significant difficulties. The use of Laser-Doppler anemometry, which in principle has none of the above disadvantages, evolved from the recognition that engine design must be firmly based on the techniques of analysis and fundamental understanding of the in-cylinder processes. Schlieren photographic methods have been used to provide a highly

informative qualitative assessment of the fluid motion inside the combustion chamber. When coupled with other diagnostic efforts, the visualization allows the detailed behavior at a point to be placed in context with other phenomena within the chamber. Thus, the value of each diagnostic is enhanced by the application of the other. For example, the understanding of the flow structure would alleviate the directional ambiguity of the hot-wire anemometer and also enable investigators to choose more strategic locations for point measurements.

A broad range of predictive models and mechanistic theories also abound in the literature. ESCIMO, "population balance", "eddy breakup", coalescence/dispersion and probability models are but a few of the prominent models proposed by researchers in the field of turbulent combustion. The most widely applicable and popular numerical implementation of the predictive models usually consists of finite difference formulations of the appropriate equations. Although such techniques work well in many cases, the need for a fine grid in the boundary layer where sharp gradients exist places a computational upper bound on the size of the Reynolds number that can be effectively modeled. Analysis implies that at least several grid points must fall within the boundary layer whose thickness is $O(Re^{-1/2})$ and one always has a finite amount of space available on the computer. This problem is especially important in cases dealing with flows in wakes or separated flows when the initial boundary layer may not be visible to a coarse finite difference grid. Another drawback of finite differences is that in areas near the boundaries sharp gradients may give rise to large truncation errors which swamp the original approximation. This problem is further complicated by the truncation errors which may cause a numerical viscosity to form which is greater than the viscosity of interest, [32]. Also, strongly

interacting transient fuel sprays are difficult to model using finite difference techniques. The difficulties stem from the fact that the spray contains a wide range of droplet sizes, length scales and time scales. Furthermore, the inclusion of detailed combustion chemistry in this approach is far beyond the capability of modern computers, and cannot be seriously considered as a technique for modeling the combustion processes in practical combustion devices.

The past two decades have brought enormous advances in computers and computational techniques on the one hand, and measurements and data processing on the other hand. The impact of such capabilities has led to a revolution both in the understanding of the vortical structures which are important in the mechanics of turbulence as well as in predictive methods for application in technology. The existence of identifiable, large scale coherent structures (eddies) in turbulent flows, as observed by Brown and Roshko [12], Winant and Browand [72], Roshko [61] and Dimotakis and Brown [27], has led to the growing belief that turbulence or vorticity fluctuations are not so random as was commonly thought. This development has prompted several researchers to propose turbulent mixing and combustion models based on the creation, evolution, interaction and decay of these eddies. Vortex methods represent a class of this new generation of numerical models.

The discrete vortex method was proposed by Chorin [16,17,18] for modeling fluid flow. In Chorin's scheme, the vorticity in the fluid and in the boundary layers is subdivided into vorticity functions with finite cores (discrete vortices), and the equations of motion are solved by following the discrete vortices throughout the fluid. The fundamental advantage of this method is that it is suitable for use in high

Reyn

Reyn

Reyn

Green

Wing

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Tha

Reynolds number flows without the problems of spatial resolution, stability and numerical viscosity associated with conventional finite difference schemes. Some reliable results have been obtained using this scheme [2,16,64]. However, for Green's function formulation [45], the dynamics of the vorticity field translates into a computational effort which scales as the square of the number of vortex elements. This is due to the fact that if N discrete vortices are present in the fluid, then $O(N^2)$ interactions must be computed per time step, since each vortex influences all the others. Furthermore, the discrete vortices move according to two components, one of which is a random displacement. This displacement leads to a partially random distribution of vorticity which becomes more random as the number of discrete vortices present in the fluid decreases. Hence to get accurate results, one must use many vortices or average over ensembles.

In this study, air and fuel mixing processes are examined within the fuel-injected cylinder of a piston engine. The mixing process is important because it controls combustion which in turn controls the pollutant formation mechanisms. A detailed understanding of the mixing phenomena allows advanced engine concepts to be explored in search of higher efficiency, lower emissions and greater fuel flexibility.

The idea of mixing is best described perhaps by the combination of bulk, eddy and molecular diffusion effects. Bulk diffusion includes diffusion other than eddy and molecular, e.g., diffusion caused by the bulk effect of fluid motion. In turbulent flows, the bulk motion of large groups of molecules (eddies) give rise to eddy diffusion. Molecular diffusion is a product of relative molecular motion across the boundaries of fluid elements. In any fluid system, molecular diffusion is always occurring and may

not always be neglected. Superimposed on this could be either bulk or eddy diffusion or both. The attainment of sub-microscopic homogeneity, where molecules are uniformly distributed over the field, is the ultimate measure of the effectiveness of mixing in any mixing process. Chemical reaction occurs only where the fluids are molecularly mixed.

Fuel-injected internal combustion engines typically employ injectors producing finely atomized sprays with complex distribution patterns and time-varying injection rates. The interaction of the injected fuel jet with the primary air motion inside the engine cylinder compounds the complexity of the fluid motion. The ensuing chemical kinetics of the combustion process makes the complexity of the fluid motion even more staggering. Given this consideration, attention is limited to the mixing-dominated phase prior to chemical reaction. Specifically, the bulk effect of the inviscid fluid motion on the scalar field is numerically simulated.

The two-dimensional, inviscid calculations employ the discrete vortex method developed by Chorin in conjunction with the Vortex-In-Cell (VIC) solution technique to approximate the fuel intake and the mean flow generated by the primary air swirl inside the combustion chamber. The VIC technique combines the best features of the vortex tracking concept of the discrete vortices with the stream function-vorticity finite-difference method. It employs a numerical filter to spread the vorticity onto a grid for use by a fast Fourier transform routine. The VIC technique is conceptually more efficient than Green's function or Biot-Savart integration techniques, in part, because the work load increases linearly with the number of vortex elements. Although the numerical simulation of the viscous and eddy diffusion effects and

and

and

and

and

and

and

and

and

and

and

and

and

and

and

and

and

and

and

and

molecular activity are neglected for simplicity, the theoretical framework relating these effects are fully addressed in this study.

The basis for the simulation is discussed and tested against theoretical results for the velocity field obtained by the method of images. The numerical predictions of the effect of bulk diffusion on the scalar field are qualitatively assessed and shown to be consistent with theory and intuition. Finally, a conclusion is drawn with regard to the optimum swirl condition for minimum mixing time.

The plan of the rest of this dissertation is as follows: Chapter 1 provides a brief historical perspective on the evolution of the discrete vortex method. The direction of earlier efforts, difficulties encountered and the work of previous investigators are outlined. A discussion of some of the mixing models consistent with the ideas and techniques of the discrete vortex method is also presented. Chapter 2 provides a clearer focus on the problem under investigation. The problem parameters, assumptions and governing equations are outlined. A more in-depth discussion of the theoretical basis of the discrete vortex method and its numerical implementation are presented in Chapters 3 and 5. Chapter 4 provides a description of the available numerical filters in the literature for use in the VIC method and a discussion of the considerations leading to the choice of an optimal filter. In Chapter 6 numerical experiments and discussion of results are presented. A summary and concluding remarks regarding the study are provided in Chapter 7 and in Chapter 8 suggestions for future studies are outlined.

CHAPTER 1

PREVIOUS INVESTIGATIONS

1.1 Flow Field

Calculations using vortices to simulate the gross features of an incompressible two-dimensional fluid motion have a long history of evolution from Rosenhead's [60] hand calculations in 1931 using a few line vortices to recent computer calculations involving hundreds of vortices. Rosenhead was the first to introduce a method of analysis in which the interface between two uniform parallel streams of an inviscid, incompressible fluid is approximated by an array of line vortices. Using similar calculations, Abernathy and Kronauer [1] simulated the von Karman vortex street and Westwater [71] studied the rolling-up of a vortex sheet of finite breadth. In this classical line vortex method, it is assumed that the vorticity is concentrated at a number of points. This corresponds to approximating the vorticity by a sum of delta-functions. Each line vortex is moved by the velocity field induced by the other vortices.

1870

1871

1872

1873

1874

1875

1876

1877

1878

1879

1880

1881

1882

1883

1884

1885

1886

1887

1888

1889

1890

1891

1892

1893

With the advent of computers, Birkhoff and Fisher [10] reexamined and refined Rosenhead's calculation by following more line vortices with more accurate integration time schemes. They showed that some of the vortices actually move about irregularly in contrast to Rosenhead's result. It was then apparent that the velocity field becomes unbounded near a line vortex, and this leads to a spurious interaction of neighboring vortices. This effect is not present in the original calculations by Rosenhead and Westwater, possibly because of the small number of vortices used or the limited accuracy of their calculations. Takami [69] and Moore [52], using a large number of vortices, indicate that the classical line vortex method is unreliable.

Most of the earlier efforts were directed toward understanding the time evolution of finite-area vorticity regions or initial break up of the shear layer in an inviscid, incompressible fluid. Due to the difficulty with the velocity divergence near the line vortex center, numerical solutions using line vortices have had some poor convergence results. In the early part of the last two decades, Chorin and Bernard [19] initiated a resurgence of interest in the use of vortex methods, for modeling real high Reynolds number flows of practical significance, when they suggested the use of discrete vortices (modified line vortices with finite cores) for smoothing the singularity associated with line vortices. The finite core radius is analogous to the introduction of a small viscosity which allows the vorticity in the line vortex to diffuse only a small distance. The discrete vortices are assumed to retain the same shape for all time. Chorin improved the convergence of flow past a circular cylinder [16] by an appropriate choice of the core or "cutoff" radius. The inviscid component of the real flow is solved via the discrete vortices while the viscous effects between vortices are

incorporated using a random walk for each vortex. Chorin [17] further improved the slow rate of convergence near solid boundaries when he introduced a vortex sheet algorithm to generate finite vortex sheets only as needed to satisfy the no-slip boundary conditions in the surrounding potential flow. The rate of convergence for the various forms of the vortex core smoothing function have been discussed by various authors including Hald and Del Prete [39], Milinazzo and Saffman [50] and Beal and Madja [8]

Other investigators have performed discrete vortex calculations on a variety of flow problems. Ashurst simulated turbulent mixing-layers [2,3], fluid motion in a four-stroke piston cylinder [4] and various combustion problems [5]. Shestakov [64] derived a hybrid method for the cavity flow problem. Meng and Thomson [49] combined the discrete vortex method with the VIC technique to study a class of non-linear hydrodynamic problems which includes the transport of aircraft trailing vortices in a wind shear, and the gravity current. A literature search shows that the majority of the calculations using discrete vortices have been performed for external flows. The method has not been extensively applied to problems involving internal flows. A comprehensive review of the general class of vortex methods which should serve as a good entry point into the literature is given by Leonard [45]. Other notable reviews appearing elsewhere include Clements and Mull [21] and Zabusky [74,75].

Con

Exp

Aero

Obs

Wind

Temp

Dir

Wind

Wind

Wind

Wind

Wind

Wind

Wind

Wind

Wind

Wind

Wind

Wind

1.2 Mixing Models

Contemporary mixing models which utilize the turbulent structure of the flow in engines is perhaps best illustrated by Spalding [66] in his ESCIMO theory. The acronym stands for Engulfment, Stretching, Coherent, Inter-diffusion, Moving Observer. In this model, Spalding describes the engulfment process of the eddies as well as the details of the internal processes within an eddy. Although the model appears to conceptually embody much of the physics of turbulent mixing, validation of the model is still at an early stage.

The Coalescence and Dispersion (C/D) model, which admits finite rate micromixing, was first suggested by Curl [26] when he proposed a rather complicated integro-differential equation for which no analytical solution is known. Curl's C/D model was originally intended to describe drop interactions in a two-liquid phase chemical reactor. It was later extrapolated to gaseous turbulent systems and utilized in the fields of chemical engineering and turbulent combustion [35,36,57]. The model suggests a simple mechanism for the exchange of scalar concentration between initially segregated fluid particles that are fed continuously into a perfectly-stirred reactor. The feed rate was set to a predetermined rate so that a constant particle population was maintained. The fluid particles are pictured as undergoing independent pair collisions, representative of the molecular diffusion time scale.

Levenspiel and Spielman [46] first used the Monte Carlo numerical technique to solve the integral term in Curl's equation. The Monte Carlo method is a stochastic fluid particle interaction model of the turbulent mixing process. It is conceptually

simple and especially suited for mixing in discrete systems. The flexibility, clarity and practicality of this formulation has resulted in its subsequent widespread use. Flagan and Appleton [36] give a lucid formulation of the Monte Carlo technique for the solution of Curl's equation. The works of Shreider [65] and Hammersley and Handscomb [40] describe various applications of the Monte Carlo method.

The probability density function (PDF) model plays an increasingly significant role in the theoretical treatment of turbulent flows and turbulent mixing of scalar fields. Because transport of the PDF is accounted for in both position and probability spaces, these equations give a more complete description of the system than their moment counterpart and the physics of the system can be more exhaustively explored. The PDF formalism is particularly attractive for reacting flows, since the effects of reaction appear in closed form irrespective of the complexity and non-linearity of the reaction scheme. These favorable attributes have led several authors to base turbulence closures on transport equations for the PDF's. Lundgren [47] devised a method to derive a hierarchy of transport equations for N-point probability density functions for fluctuating variables in turbulent incompressible flow. The Lundgren approach has been applied to turbulent flames by Pope [55] and Janicka, et. al. [42]. Fox [37] and Dopazo and O'Brien [30,31] extended this method to compressible and reacting flows.

The attraction of the PDF approach appears to diminish when solving the transport equation. Analytical solutions have been obtained in a few simple cases, but in general, numerical methods are required. Such numerical methods (e.g., the Monte Carlo method) have to cope with the integro-differential nature of the

equation

involves

solution

than 2

specification

equations and with the large dimensionality of the PDF when many species are involved. An order of magnitude analysis [56] shows that the finite-difference solution of the PDF transport equation is impractical for a number of species greater than 3. The computational expense is found to rise exponentially with the number of species.

CHAPTER 2

DESCRIPTION OF PROBLEM

The numerical simulation of air-swirl generated inside a flat disc-shaped cylinder representing the compression volume of a piston engine combustion chamber is described in this chapter. Under the effect of a high pressure differential, a fuel jet is introduced at an adjustable angle into the cylinder through a nozzle orifice located on the cylinder wall and the problem of the evolution of the velocity and vorticity fields due to bulk diffusion is addressed.

For liquid fuel injection into the cylinder of a piston engine, an aerodynamic force will be encountered due to the swirling air in the cylinder. The fuel, internally turbulent and non-uniform at the exit plane of the nozzle, will exhibit a tendency to disintegrate that ultimately will lead to the breakup of the fuel into droplets of different sizes and concentrations in the spray. The distribution of droplet sizes will depend on the relative magnitudes of the aerodynamic, viscous, inertial, and surface forces present. The atomized fuel spray will be deflected by the air swirl and will evaporate and mix with air at the fringe of the jet at a rate determined by the primary air swirl, turbulent motion, and molecular viscosity. At some time during fuel atomization and mixing, combustion would occur with the ensuing flame spreading

across the chamber.

The time between injection and attainment of chemical reaction conditions is the physical delay period. Lyn [48] observed that the combustion process can be divided into three distinguishable heat release phases, as shown in Figure 1. The first phase is associated with a period of rapid combustion and rapid cylinder pressure rise and lasts for only about 3° crank angle (C.A.). During this phase, fuel that has evaporated and mixed with air during the physical delay period burns with a non-luminous flame. The burning rate decays in the second phase which lasts about 40° C.A. and the jet burns with a luminous, highly turbulent diffusion flame. In the final phase, the remaining fuel burns at a very low rate extending through the entire expansion stroke.

The present study is concerned with the mixing-dominated physical delay period prior to chemical reaction. The addition of combustion chemistry will be the topic of a future study.

Experimental studies of the fluid flow in a firing engine cylinder prior to and during injection, reveals a complex combination of fluid motions. The nature of the motion in the engine cylinder can be broadly classified into three types : swirl, turbulence and squish. Swirl denotes a large scale rotary motion of the charge in the combustion chamber about the chamber axis, more or less. It is created during induction and amplified during compression by the combustion chamber configuration. Turbulence is the fluctuating velocity component superimposed on the mean velocity

of the fluid motion. Squish denotes the organized radial motion created by the piston movement near top dead center (TDC), that is, near the end of compression. Because it is anticipated that squish effects will significantly alter the flow patterns at the end of compression, a right cylindrical disc-shaped cylinder with no squish zones is used to model the combustion chamber near TDC. The formulation of a mixing model, which neglects bulk fluid motion typified by the air swirl inside the cylinder cannot hope to realistically predict design criteria.

The theoretical formulation will consider a system of equations composed of the conservation of mass, momentum and scalar species. The problem of evolution toward a uniform final state of the fuel undergoing inviscid bulk diffusion at constant pressure and temperature will be numerically investigated using discrete vortex dynamics. At high Reynolds number, the aerodynamics of the flow are dominated by inertial forces except at the immediate vicinity of the solid surface where viscosity plays a dominant role. In this study, the effects of molecular viscosity, turbulence and molecular mixing are neglected for simplicity. The closure of the turbulent transport term and a satisfactory model for the molecular mixing term in the scalar conservation equation are particularly challenging. A future study will attempt to incorporate these effects. A discrete jet model analogous to Ashurst's scheme [4] and a Vortex-In-Cell solution technique will be used to approximate the fuel injection and the fluid motion inside the cylinder, respectively.

Because of the three-dimensional and turbulent nature of the phenomena occurring within the engine combustion system, it is not advisable to solve the fundamental equations even with the use of the largest computers. To keep the

mathematical difficulties involved within reasonable bounds, assumptions must be introduced to allow the simplification of the equations and the evolution of a tractable mixing theory :

- (i) It will be assumed that the controlling features of the problem can be described by a two-dimensional, constant volume, mean flow formulation. The flat, disc-shaped configuration of the cylinder and the fact that combustion in an internal combustion engine occurs near TDC where the volume does not change appreciably, justify this assumption.
- (ii) For two-dimensional flows, the stretching and rotation term in the vorticity transport equation is zero. The only nonzero component of the vorticity vector will be normal to both the velocity vector and the direction of the piston motion.
- (iii) Under engine operating conditions, the chamber conditions are near or above the thermodynamic critical point of the fuel. Test results by Shahed, et. al. [63] show that a major portion of the spray appears in vapor form. Droplets appear only near the jet nozzle exit within a few orifice diameters. This finding agrees with the conclusions of several other workers [13,48,53]. For simplicity, this study represents the fuel jet as air jet. The consideration of a single phase eliminates complications caused by fuel droplets.

- (iv) The change in temperature and pressure of the vaporized mixture in the physical delay period is minimal. This assumption can be justified from energy conservation analysis. Thus the kinematic viscosity is assumed constant since the pressure is assumed constant prior to combustion.

- (v) We will also assume that, to a first approximation, the atomized fuel spray, idealized as discrete vortices, moves with the entrained flow. This is a fair assumption because parametric estimates [59] of the aerodynamic forces on typical droplets in a fuel spray suggest that the relative velocities of droplets in air rapidly decay so that droplets move with the entrained air in the jet flow.

- (vi) An incompressible fluid is assumed since there is no volume source or sink in the flow situation considered. Volume displacement effects are neglected since the volume occupied by the fuel is small compared to the volume occupied by the swirling air flow.

- (vii) Gravity is not a factor. Centrifugal convection is due primarily to the centrifugal body force which is accounted for in the total derivative.

CHAPTER 3

THE DISCRETE VORTEX METHOD

3.1 Introduction

Modern experimental investigations identifying the existence of large scale coherent structures (eddies) in turbulent flows have rekindled much interest in the numerical modeling of turbulent combustion problems. Contemporary numerical models seek to utilize the observable eddy structures of the turbulent flow to simulate the physical systems directly without recourse to conventional finite difference formulation. Vortex methods represent such a class of numerical techniques.

The central theme of Chorin's discrete vortex method is based on the realization that it is the interaction of the eddies that produces transport of mass, momentum and energy. The basic concept of the method for flow simulation is to represent the observable eddy structures in the vorticity-containing regions of the flow by discrete vortices and then track this discretization in a Lagrangian reference frame. The essentially grid-free formulation permits the simulation of the growth and decay of eddies encountered in real fluids. The method is particularly appealing

because it is specifically designed to deal with problems of increased spatial resolution, stability and numerical viscosity associated with finite difference formulation.

3.2 The Flow Field

At any point in a non-reactive mixing flow system, the state of the fluid can be characterized by the velocity V , the pressure p and a set of scalars $\Phi=(\phi_1,\phi_2,\dots,\phi_\sigma)$ representing the species mass fractions; σ is the number of independent species. As a consequence of the idealizations discussed in the previous chapter, the conservation of mass and the Navier-Stokes equations governing the flow field can be expressed in the following simple form :

$$\text{Conservation of Mass :} \quad \nabla \cdot V = 0 \quad (3.1)$$

$$\text{Navier-Stokes :} \quad \frac{DV}{Dt} = - \frac{\nabla p}{\rho} + \nu \nabla^2 V \quad (3.2)$$

where $V=(u,v)$ is the velocity vector, ν , ρ and p are the kinematic viscosity, density and pressure of the fluid, respectively. $\frac{D}{Dt} = \frac{\partial}{\partial t} + \nabla \cdot V$ is the total derivative, ∇ is the del operator and ∇^2 is the Laplacian operator. The flow field is satisfied by the solution of these equations in a finite circular domain D with boundary S (representing the combustion chamber), subject to the no-slip boundary condition

$$V=0 \quad \text{on} \quad S. \quad (3.3)$$

The pressure can be eliminated by introducing the vorticity

$$\xi = \partial_x v - \partial_y u \quad (3.4)$$

The curl of Eq. (3.2) gives the vorticity transport equation

$$\frac{D\xi}{Dt} = \nu \nabla^2 \xi \quad (3.5)$$

The initial condition $\xi(x,y,t=0)$ is assumed known in D . The coordinates x and y are cartesian coordinates and t is time. The discrete vortex formulation makes use of the simplified system of equations which now consists of Eqs. (3.1), (3.4) and (3.5) to describe the flow.

A convenient solution device is the operator splitting technique, discussed extensively by Yanenko [69], whereby the transport of vorticity (Eq. 3.5) can be implemented in two fractional steps : advection and diffusion. The component equations are

$$\text{Advection :} \quad \frac{D\xi}{Dt} = 0 \quad (3.6)$$

$$\text{Diffusion :} \quad \partial_t \xi = \nu \nabla^2 \xi \quad (3.7)$$

This technique permits the application of the most efficient numerical algorithm to compute each part independently and in succession. The combination of the solutions of the advection and diffusion equations produces an approximation to the solution for the full flow.

The Euler equation (Eq. 3.6) is equivalent to the statement that the conserved vorticity field is advected by its own velocity field. The appearance of the vorticity diffusion term in equation (3.7) has two ramifications : vorticity creation at the boundary and diffusion of concentrations of vorticity in the flow field. Chorin [16,17,18] has constructed a vorticity creation algorithm at the boundary using vortex sheets and suggested that viscous diffusion in both the interior and boundary layer regions can be accomplished in the Lagrangian framework by adding a random walk component to each vortex at each time step, Δt .

The divergence free velocity field can be constructed from the vorticity distribution through superposing the velocity field produced by each individual vortex. Often times this is done in terms of the classical Green's function or by use of the Biot-Savart integration technique. A hybrid method utilizing Vortex-In-Cell formulation which employs an efficient elliptic boundary value problem solver for the interior region of the flow domain and Chorin's [17] vortex sheet algorithm near the solid boundary can also be used to approximate the velocity field.

In Chorin's vortex sheet algorithm, the random variables are to be drawn from a Gaussian distribution with zero mean and a variance that grows with time as $2\nu\Delta t$. This concept of stochastic modeling of diffusion is essentially based on Einstein's

studies of Brownian motion [34] and the analysis by Courant, et. al. [25]. The idea is that the effects of real viscosity are correctly reproduced in a statistical sense.

3.2.1 The interior region

Since the flow is incompressible and two-dimensional, the velocity components u and v can be expressed in terms of the streamfunction ψ as follows :

$$u = \partial_y \psi \quad ; \quad v = -\partial_x \psi \quad (3.8)$$

$$\text{and from } \xi = \partial_x v - \partial_y u, \quad \nabla^2 \psi = -\xi \quad (3.9)$$

with the normal boundary condition

$$\mathbf{V} \cdot \mathbf{n} = 0 \quad (3.10)$$

where $\mathbf{n}$ is the unit normal vector on S .

An initial distribution of irrotational discrete vortices is selected so that an approximation to the vorticity field is obtained. Equation (3.9) is used to determine ψ up to an additive constant. In terms of the classical Green's function, it is given by

$$\psi = \sum_{\substack{j=1 \\ j \neq i}}^N G_{ij}(z_i - z_j) \kappa_j \quad (3.11)$$

where N is the number of vortices, $z_i = (x_i, y_i)$ is the i^{th} point of interest in the flow domain, $z_j = (x_j, y_j)$ is the position of the j^{th} discrete vortex, $G_{ij}(z_i - z_j)$ is the Green's function and

$$\kappa_j = \frac{(\gamma_p)_j}{2\pi} \quad (3.12)$$

is the strength of vortex j . The parameter $(\gamma_p)_j$ is the circulation of vortex j . If the normal boundary condition is to hold, then at the boundary, the streamfunction must satisfy

$$\psi = \text{constant} \quad (3.13)$$

Each vortex is moved by the velocity field induced by all the other vortices. Thus combining Eq. (3.9) with Eq. (3.11) we get

$$u_j = \sum_{\substack{j=1 \\ j \neq i}}^N \partial_y G_{ij}(z_i - z_j) \kappa_j \quad (3.14)$$

$$v_j = - \sum_{\substack{j=1 \\ j \neq i}}^N \partial_x G_{ij}(z_i - z_j) \kappa_j \quad (3.15)$$

where u_j and v_j are the velocity components at the vortex position in the cartesian coordinate system. To advance the vorticity to the next time step, the velocity is calculated at the current position of each discrete vortex and using an appropriate time integration scheme, the advection step is accomplished.

Typically, the Lagrangian coordinates of the vortices satisfy a non-linear system of ordinary differential equations :

$$\frac{dx_j}{dt} = u_j(x_j, y_j) \quad ; \quad \frac{dy_j}{dt} = v_j(x_j, y_j) \quad (3.16)$$

This system can be solved by giving the time evolution of the coordinates:

$$x_j^{n+1} = x_j^n + \int_{t_0}^{t_0+\Delta t} u_j^n dt \quad (3.17)$$

$$y_j^{n+1} = y_j^n + \int_{t_0}^{t_0+\Delta t} v_j^n dt \quad (3.18)$$

where t_0 is the time at the beginning of integration, Δt is the time step and n represents the step number. A higher-order time integration scheme is suggested by Hald and Del Prete [39].

Viscous diffusion of vorticity in the interior region is modeled according to Chorin's random walk scheme [16]. If $\vec{\eta}=(\eta_1, \eta_2)$ is a vector whose components are Gaussianly distributed random variables, with mean zero and variance $2\nu\Delta t$, then Eq. (3.5) is approximated by

$$x_j^{n+1} = x_j^n + \int_{t_0}^{t_0+\Delta t} u_j^n dt + (\eta_1^n)_j \quad (3.19)$$

$$y_j^{n+1} = y_j^n + \int_{t_0}^{t_0+\Delta t} v_j^n dt + (\eta_2^n)_j \quad (3.20)$$

where $(\eta_1^n)_j$ and $(\eta_2^n)_j$ are the Gaussianly distributed random variables at time step number n .

It has long been known that the convergence of the resulting flow fields to the exact solutions of Equations (3.1) and (3.6) is rather poor; the reason is connected with the singularity in the velocity field when $z_i=z_j$, i.e., at the vortex positions. An obvious modification of the method is to pick $|z_i-z_j|$ small but not zero, and ensure

that V remains smooth as $z_i \rightarrow z_j$. The techniques for doing this are implicit in the work of Hald and Del Prete [39], Millinazzo and Saffman [50] or Beal and Majda [8]. The form of the vortex core smoothing function or Green's function $G_{ij}(z_i - z_j)$ given by Hald and Del Prete is

$$G_{ij} = \left[C_1 \left(1 - \frac{|z_i - z_j|^3}{r_0^3} \right) - C_2 \left(1 - \frac{|z_i - z_j|^2}{r_0^2} \right) + \log r_0 \right] \quad ; \quad |z_i - z_j| \leq r_0 \quad (3.21)$$

where r_0 is the radius or "cut-off" of the vortex, $C_1=2/3$ and $C_2=3/2$. Outside of this radius an irrotational motion is assumed and inside of this radius "rigid body" motion is assumed. For details concerning the accuracy or consistency of $G_{ij}(z_i - z_j)$ see Hald and Del Prete [39].

The application of Green's function formulation is clearly troubled by the source singularities and by long running times since a large number of vortices are required for reasonable accuracy. Recall that $O(N^2)$ interactions must be computed per time step.

This study circumvents these problems by employing the Vortex-In-Cell (VIC) formulation for the numerical approximation of the inviscid velocity field. In the VIC formulation, the inversion of Poisson's equation, (Eq. 3.9), is performed on a grid superimposed over the vortices by using a Fast Fourier Transform (FFT) method

whose work load increases linearly with the number of vortex elements. A numerical filter, or shape function is applied to each vortex element to temporarily create a mesh record of the vorticity field. This eliminates the singularity associated with the vortex position. A centered difference technique, which assures local conservation of mass, is applied to the mesh record of the stream function resulting from Poisson inversion to obtain local effects of the velocity field. The filter, again, provides the means for interpolating the local effects of the velocity field from the grid to the vortex positions which are now updated.

It turns out that interpolation to and from the grid are the most sensitive procedures of the VIC algorithm. The choice of a good numerical filter is therefore crucial in the successful implementation of the VIC algorithm. A discussion of numerical filters is presented in chapter 4.

3.2.2 Boundary layer region

The boundary layer region acts as a transition zone from the still fluid layer at the wall to the high speed flow in the interior. Since the viscous sublayer is thin compared to the radius of the combustion chamber, the Prandtl boundary layer equations for incompressible flow over a flat plate

$$\frac{D\xi}{Dt} = \nu \partial_{yy}^2 \xi \quad (3.22)$$

$$\xi = - \partial_y u \quad (3.23)$$

provide a reasonable description of the propagation of vorticity in this region. Prandtl boundary layer equations are approximations to the Navier-Stokes equations derived from the assumptions that : (i) vorticity is mainly produced from large gradients of u in the y direction, hence $\xi = \partial_x v - \partial_y u \approx -\partial_y u$; and, (ii) along the wall, the large gradients in the y direction allow $\partial_{xx}^2 \xi \approx 0$. Here, x and y are directions parallel and normal to the solid wall, respectively.

The computation of the flow field generated by point and discrete vortices near solid boundaries has been shown by Chorin [17] to have a slow rate of convergence. To improve the convergence rate in this region, Chorin introduced the vortex sheet algorithm to simulate a solution of Prandtl's boundary layer equations. Vortex sheets are surfaces parallel to solid walls across which the fluid velocity has a continuous normal component, but a discontinuous tangential component. The jump in the tangential velocity is the strength of the vortex sheet. In addition to the improved rate of convergence, the vortex sheet algorithm, like the VIC algorithm, avoids all problems with singularities and offers substantial saving in the amount of computational effort expended.

To describe the motion of vortex sheets, one begins with equations (3.1) and (3.23). The vorticity in equation (3.23) can be integrated in the form

$$u(x,y) = U_{\delta_s}(x) + \int_y^{\delta_s} \xi(x, \alpha) d\alpha \quad (3.24)$$

where δ_s is the numerical shear layer thickness at the wall, U_{δ_s} is $u(x,y)$ at $y = \delta_s$, that is, the velocity at "infinity" (the outer edge of the numerical shear layer) seen by the layer. From Eq. (3.1) and the boundary condition $v(x,0) = 0$, we get using central difference approximation

$$v(x,y) = -\partial_x \int_0^y u(x,\alpha) d\alpha = -\frac{1}{h} \left[\int_0^y u(x+h/2,\alpha) d\alpha - \int_0^y u(x-h/2,\alpha) d\alpha \right] \quad (3.25)$$

Equations (3.24) and (3.25) allow one to determine u and v if the vortex sheet distribution $\xi = \xi(x,y)$ is known.

Consider a collection of N vortex sheet segments each with circulation γ_s .

The equivalent discrete expression for $\mathbf{V} = (u,v)$ at the center $\mathbf{r}$ of the j^{th} vortex sheet segment due to the presence of other segments can be written as [17]

$$u(x_j, y_j) = U_{\delta_s}(x_j) + \frac{1}{2} \gamma_{s_j} + \sum_{\substack{i=1 \\ i \neq j}}^N \gamma_{s_i} d_i \quad (3.26)$$

$$v(x_j, y_j) = -\frac{(I_+ - I_-)}{h} \quad (3.27)$$

where

$$d_i = 1 - \frac{|x_j - x_i|}{h} \quad ; \quad 0 \leq d_i \leq 1 \quad \text{and} \quad y_j \leq y_i \quad (3.28)$$

$$I_{\pm} = U_{\delta_s} [x_j \pm \frac{h}{2}] y_j + \sum_{\substack{i=1 \\ i \neq j}}^N \gamma_s d_i^{\pm} y_i^* \quad (3.29)$$

$$d_i^{\pm} = 1 - \frac{|x_j \pm \frac{h}{2} - x_i|}{h} \quad ; \quad 0 \leq d_i^{\pm} \leq 1 \quad (3.30)$$

$$y_i^* = \min(y_i, y_j) \quad (3.31)$$

and h is the length of the vortex sheet segment

Note that the total number of interactions between vortex sheets is small in particular in comparison with what happens when discrete point vortex interactions are taken into account; one can use sorting algorithms to minimize the number of decisions involved in carrying out the several summations (see Article 5.6).

The position of the sheet is then given by

$$x_j^{n+1} = x_j^n + \int_{t_0}^{t_0+\Delta t} u_j^n dt \quad (3.32)$$

$$y_j^{n+1} = y_j^n + \int_{t_0}^{t_0+\Delta t} v_j^n dt + (\eta_2^n)_j \quad (3.33)$$

The Gaussianly distributed random variable $(\eta_2^n)_j$ appears only in the y-component because Prandtl boundary layer equations take into account diffusion in the y direction only. A first-order time integration scheme is used to advance the solution in this case. Because the solution is computed in the (x,y) plane, without a change of variables, it should be easy to match the computed boundary layer solution with an inviscid solution outside the layer.

The coupling of the interior and boundary layer solutions is provided by $U_{\delta_s}(x)$.

This is the tangential velocity from the interior which is used as the velocity at infinity for the boundary layer. This fact is evident in equations (3.26) and (3.27). This allows the interior calculation to determine the production of vorticity within the boundary layer. It should be noted that the number of vortex sheets will increase in time and that discrete vortices appear in the interior domain only as a consequence of the displacement of vortex sheets outside the boundary layer, thereby modeling the mechanism of the generation of turbulence under actual flow conditions.

The combined flow fields of the two vortex algorithms must satisfy the no-slip condition for the velocity. By construction, the normal boundary condition $\mathbf{V} \cdot \mathbf{n} = 0$ on S , is satisfied by both algorithms. However, the no-slip boundary condition $\mathbf{V} \cdot \mathbf{s} = 0$, $\mathbf{s}$ tangent on S , will require special attention. The vortex sheet algorithm was specially designed to impose this condition by mimicking the actual vorticity generation processes present on the wall and in so doing, provide a vehicle for the introduction of vortex elements into the computational domain.

If at the wall the tangential velocity $V_\theta \neq 0$, the effect of viscosity will be to create a thin boundary layer near the wall; the total circulation in the layer per unit length of the wall is

$$\int_0^{\text{interior}} \xi dy = \int_0^{\text{interior}} \left(-\frac{\partial u}{\partial y}\right) dy = -V_\theta \quad (3.34)$$

Thus, one has to create a vortex sheet of circulation $\gamma_s = -V_\theta$ per unit length of the wall to satisfy the tangential no-slip condition.

Intuitively, the vortex sheets move about in ideal flow together with a random y-component simulating viscous diffusion. Vortex sheets newly created to accommodate the boundary conditions diffuse out from the boundary by means of the random component and then get swept by the main flow. This mechanism then has most of the features a boundary layer should have.

3.2.3 Intake flow

The initial air-fuel composition $\phi_\alpha(\mathbf{x},0)$, ($\alpha = 1, 2$) are established by an air particle injection process simulating the intake fuel flow. The intake process of internal combustion engines produces thin shear layers that dominate the interior flow patterns. Since a finite-difference calculation would numerically smear these layers [38], the dynamics of the injected air particles are analyzed using a discrete jet model analogous to Ashurst's scheme [4].

The injector nozzle valve is modeled as a single port with diameter d_j equal to the mesh size d (see Figure 2). At each time step, for the duration of the intake stroke, the intake flow is simulated using two discrete vortices, one created at the valve lip (region A) and the other created at the valve seat (region B) with the appropriate sign and strength for the circulation given as [4]

$$(\gamma_p)_A = - \frac{U_{in}^2}{2} \Delta t \quad ; \quad (\gamma_p)_B = + \frac{U_{in}^2}{2} \Delta t \quad (3.35)$$

where U_{in} is the inlet jet velocity, $(\gamma_p)_A$ is the source strength at the valve lip and $(\gamma_p)_B$ is the source strength at the valve seat. The source strength is assumed constant in this study. The valve face is capable of closing the inlet flow area and the angle of injection θ_j can be varied between 0° and 180° . The particles are injected

with i

the en

solid w

number

:

where V

V_s is the

the inte

obtained

324 S

A

is follow

(1)

(2)

with initial velocity sufficient to clear the numerical boundary layer thickness δ_s . At the end of the intake flow simulation, the valve face is closed and represented as a solid wall.

Note that at each time step, for the duration of the injection process, the number of vortices in the system increases by two and the velocity field has two parts :

(3.36)

$$\mathbf{V}(x,y) = \mathbf{V}_\xi + \mathbf{V}_s$$

where $\mathbf{V}_\xi$ is the velocity produced by all the vorticity that is already in the system and $\mathbf{V}_s$ is the velocity produced by the source represented by the two discrete vortices at the inlet flow area. The distribution of the jet vortices in the system at later times is obtained by letting the vortices move with the swirling fluid.

3.2.4 Summary

A summary of the hybrid vortex algorithm for modeling the entire flow field is as follows :

- (1) Find the velocity $\mathbf{V}$ for each vortex in the interior flow domain.
- (2) Find the value of the tangential velocity at the boundary, and use this as the free stream velocity to compute $\mathbf{V}$ for each element of

the vortex sheet.

- (3) Create vorticity as needed to satisfy the no-slip condition by adding vortex sheets on the boundary.
- (4) Move the interior discrete vortices according to Eqs. (3.19) and (3.20) and the vortex sheets according to Eqs. (3.32) and (3.33), with the stipulation that if a sheet moves into the interior, it becomes a discrete vortex and vice versa.
- (5) Steps (1) and (2) gives the velocity V at vortex locations in the flow field. The velocity at any non-grid point in the flow field can be found by interpolation.

3.3 Exact Solution

For purposes of comparison, the exact irrotational velocity field in the circular domain and the streamfunction in the square domain are calculated via an imaging system for the vortices. A modified circle theorem of Milne-Thomson [51] is employed for the velocity field in the circular domain.

The circle theorem applies to an irrotational two-dimensional, incompressible, inviscid flow and states in essence that a flow field's complex potential $f(z)$ will, upon the introduction of a circular cylinder into the field of flow, be modified to

$$W(z) = f(z) + \overline{f}(a^2/z) \quad (3.37)$$

where the overbar signifies the complex conjugate of the complex function f and a is the radius of the circular cross-section of the cylinder. Since all singularities of $f(z)$ are by hypothesis exterior to the circle, all the singularities of $\overline{f}(a^2/z)$ are interior to the circle.

Consider the application of the theorem to the flow modeled by a single point vortex j with complex potential given by

$$f(z) = -i \kappa_j \ln(z - z_j) \quad (3.38)$$

By the circle theorem, the complex potential of the flow becomes

$$W(z) = -i \kappa_j \ln(z - z_j) + i \kappa_j \ln(a^2/z - \overline{z}_j) \quad (3.39)$$

which can be rearranged to give

$$W(z) = -i \kappa_j [\ln(z - z_j) - \ln(a^2/\overline{z}_j - z) + \ln(z_j/z)] \quad (3.40)$$

On the right hand side of Eq. (3.40), the first term represents the contribution of the vortex outside of the circle, the second term is the image vortex located inside the circle, and the final term, which can be neglected for internal flows [7], is a vortex located at the origin to preserve the net circulation in the computational domain. With

this modification, the complex potential becomes

$$W(z) = -i \kappa_j [\ln(z - z_j) - \ln(a^2/\bar{z}_j - z)] \quad (3.41)$$

The velocity field produced by this complex potential can be expressed as

$$[u - i v]_j = \frac{dW(z)}{dz} = -i \kappa_j [1/(z - z_j) - 1/(z - a^2/\bar{z}_j)] \quad (3.42)$$

The singularities are smoothed using equation (3.21) as shown in Appendix B. The velocity field due to a collection of vortices can be constructed through superposition :

$$[u - i v] = \sum_{j=1}^N [u - i v]_j \quad (3.43)$$

A similar construction for the streamfunction in the square domain is presented in Appendix B. The result is

$$\psi = \frac{\kappa}{2} \ln \left[\frac{\sinh^2(\zeta_m) + \sin^2(\eta_p)}{\sinh^2(\zeta_m) + \sin^2(\eta_m)} \right] \quad (3.44)$$

where $\zeta_m = \frac{\pi}{2K}(\zeta - \zeta_0)$, $\eta_m = \frac{\pi}{2K}(\eta - \eta_0)$ and $\eta_p = \frac{\pi}{2K}(\eta + \eta_0)$. The parameter K is the dimension of the square domain as illustrated in Figure 43 in Appendix B.

3.4 Mixing

The scalar field $\phi_\alpha(\mathbf{x},t)$ ($\alpha = 1,2,\dots,\sigma$) in an incompressible flow subject to a forcing turbulent velocity $\mathbf{V}(\mathbf{x},t)$ and undergoing Fickian molecular diffusion is governed by the scalar conservation equation [54] :

$$\rho \left[\frac{\partial \phi_\alpha}{\partial t} + V_i \frac{\partial \phi_\alpha}{\partial x_i} \right] = \frac{\partial}{\partial x_i} \left(\Gamma \frac{\partial \phi_\alpha}{\partial x_i} \right) \quad (3.45)$$

where $\rho(\Phi)$ is the density, $\Phi = (\phi_1, \phi_2, \dots, \phi_\sigma)$ is the set of σ -dimensional scalars representing the species mass fractions, $V_i(\mathbf{x},t)$ is the i^{th} component of the prescribed turbulent velocity vector field and Γ is the coefficient of molecular diffusion. Summation convention over repeated indices is implied.

Since averaged quantities are primarily of interest in turbulent flows, use can be made of a formulation which models and solves the conservation equation for $P(\Psi; \mathbf{x}, t)$, the averaged joint probability density function (PDF) of the scalars. The quantity $\Psi = (\psi_1, \psi_2, \dots, \psi_\sigma)$ is the σ -dimensional probability space corresponding to Φ . Henceforth, the argument $(\Psi; \mathbf{x}, t)$ is written (Ψ) , the dependence upon $\mathbf{x}$ and t being implied.

The formal derivation of the transport equation for $P(\Psi)$ is given in [54]. The resulting equation is

$$\rho \left[\frac{\partial P(\Psi)}{\partial t} + \langle V_i \rangle \frac{\partial P(\Psi)}{\partial x_i} \right] + \frac{\partial \rho \langle v_i' p(\Psi) \rangle}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\Gamma \frac{\partial P(\Psi)}{\partial x_i} \right) - \frac{\partial^2}{\partial \Psi^2} \langle p(\Psi) \Gamma \frac{\partial \phi_\alpha}{\partial x_i} \frac{\partial \phi_\alpha}{\partial x_i} \rangle \quad (3.46)$$

where v_i' is the fluctuating component of V_i ; that is

$$V_i = \langle V_i \rangle + v_i' \quad (3.47)$$

The averaged joint PDF $P(\Psi)$ can be defined as the ensemble average of the instantaneous joint PDF $p(\Psi)$, a random function due to the dependence of the distribution of ϕ_α upon the initial conditions $\phi_\alpha(\mathbf{x}, 0)$, and the random forcing turbulence field $V(\mathbf{x}, t)$. Averaging over all possible realizations of initial conditions and V fields yields

$$P(\Psi) = \prod_{\alpha=1}^{\sigma} \langle p(\psi_\alpha; \mathbf{x}, t) \rangle = \prod_{\alpha=1}^{\sigma} \langle \delta(\psi_\alpha - \phi_\alpha(\mathbf{x}, t)) \rangle \quad (3.48)$$

where the angled brackets denote an ensemble averaged quantity and δ is a Dirac delta function. The justification for defining a probability density as the ensemble average of a collection of Dirac spikes can be found in Stratonovich [67].

It is interesting to observe the correspondences between the physically recognizable terms in equation (3.45) for the fluctuating scalar quantity ϕ_α and equation (3.46) for the averaged joint PDF $P(\Psi)$. The convective term retains its structure in the PDF-equation but the turbulent convective contribution is added. The molecular transport term appears in the PDF-equation twofold : as a transport term in position space of the same structure as in the ϕ_α -equation, and as a transport term in probability space. Molecular transport in position space represents molecular diffusion while molecular transport in probability space represents mixing by molecular action. It is shown in [54] that the effect of the mixing term is to reduce the variance of $P(\Psi)$ without affecting the mean.

Omitting the molecular diffusion terms, which are negligible at high Reynolds number, equation (3.46) reduces to

$$\rho \left[\frac{\partial P(\Psi)}{\partial t} + \langle V_i \rangle \frac{\partial P(\Psi)}{\partial x_i} \right] = - \frac{\partial \rho \langle v_i' P(\Psi) \rangle}{\partial x_i} - \frac{\partial^2}{\partial \psi^2} \langle p(\Psi) \Gamma \frac{\partial \phi_\alpha}{\partial x_i} \frac{\partial \phi_\alpha}{\partial x_i} \rangle \quad (3.49)$$

The transport equation in this form is not immediately soluble, because, whereas the term on the left is exact, the terms on the right, representing turbulent transport and molecular mixing, are unknown and need to be modeled in order to close the transport equation for $P(\Psi)$.

The closure of the turbulent transport term is the fundamental stumbling block to progress in probability formulation of turbulent scalar mixing. This is because turbulent mass transport or eddy diffusion is a complex process that is dependent on the scale and intensity of turbulence in the flow. Various models, which tend to make the mathematics easier but at best are a crude representation of reality, have been suggested in the literature [28,42,54] to describe the phenomenon. Pope's [55] simple gradient diffusion model,

$$\frac{\partial \rho \langle v_i' p(\Psi) \rangle}{\partial x_i} = - \frac{\partial}{\partial x_i} \Gamma_t \frac{\partial P(\Psi)}{\partial x_i} , \quad (3.50)$$

introducing the turbulent diffusion coefficient $\Gamma_t(\mathbf{x},t)$, seems promising and better behaved for inert scalars and for large departures from Gaussianity and homogeneity.

As with the turbulent transport term, a number of models for the molecular mixing term have been proposed in the literature [26,31,42,55] but a completely satisfactory model is yet to be demonstrated. The fluid particle coalescence and dispersion model, proposed by Curl [26], is the only model applicable to an arbitrary number of scalar quantities and can be expected to produce qualitatively correct results. The model makes no attempt to describe the detailed dynamics of the turbulent flow but, instead, relies on a probability density function of concentration, itself dependent on empirically determined mixing frequency, to describe the non-uniformities in the flow. Flagan and Appleton [36], Evangelista et. al., [35], and Pratt [57] have modeled the mixing term in the form

$$-\frac{\partial^2}{\partial \Psi^2} \langle P(\Psi) \Gamma \frac{\partial \phi_\alpha}{\partial x_i} \frac{\partial \phi_\alpha}{\partial x_i} \rangle = -\beta P(\Psi) + 2^\sigma \beta \int P(\Psi+\Psi') P(\Psi-\Psi') d\Psi' \quad (3.51)$$

where $\beta(\mathbf{x},t)$ is the appropriate mixing frequency and Ψ' is a dummy variable. The first term on the right-hand side models the disappearance of probability in the range $(\Psi, \Psi+d\Psi)$ by coalescence with other scalar ranges. The integral represents the "generation" of probability in the range $(\Psi, \Psi+d\Psi)$ by coalescence from all concentration ranges.

Given these modeling considerations, Eq. (3.49) can now be written as

$$\rho \frac{DP(\Psi)}{Dt} = \frac{\partial}{\partial x_i} \Gamma_t \frac{\partial P(\Psi)}{\partial x_i} - \beta P(\Psi) + 2^\sigma \beta \int P(\Psi+\Psi') P(\Psi-\Psi') d\Psi' \quad (3.52)$$

Equation (3.52) is to be solved in a finite domain D with boundary S . The problem is posed in the following terms : Given the initial joint PDF $P(\Psi; \mathbf{x}, 0)$, with two equally probable states (zero and unity) for $\phi_\alpha(\mathbf{x}, t)$, $\alpha = 1, 2$, and the dynamic equation (3.52), obtain $P(\Psi)$ for $t > 0$.

A numerical solution for $P(\Psi)$ can be attempted by treating the simultaneous action of the advection, diffusion and mixing processes sequentially using the technique of operator splitting mentioned in Section 3.2. The component equations are

Advection :
$$\frac{\partial P(\Psi)}{\partial t} + \langle V_i \rangle \frac{\partial P(\Psi)}{\partial x_i} = 0 \quad (3.53)$$

Eddy Diffusion :
$$\rho \frac{\partial P(\Psi)}{\partial t} = \frac{\partial}{\partial x_i} \left(\Gamma_t \frac{\partial P(\Psi)}{\partial x_i} \right) \quad (3.54)$$

Molecular Mixing:
$$\rho \frac{\partial P(\Psi)}{\partial t} = -\beta P(\Psi) + 2\sigma\beta \int P(\Psi+\Psi')P(\Psi-\Psi')d\Psi' \quad (3.55)$$

The advection of $P(\Psi)$ by the mean flow (i.e., bulk diffusion) is obtained using the solutions from the analysis of the flow field discussed in Section 3.2. It is assumed that, to a first approximation, the motion of the injected scalar particle is identical to the motion of the Lagrangian point of fluid that would occupy the position of the particle if it were not there. That is, the conserved, passive, scalar field is advected by the fluid.

Turbulent diffusion of $P(\Psi)$ can be approximated by adding turbulent velocity components to the droplet velocities. This requires that the turbulence intensity and length scales be specified. This point remains a formidable closure problem. The use of random walk in modeling turbulent diffusion was suggested by Chorin [15] as a generalization of the random walk model used in his vortex sheet method for viscous

diffusion.

Molecular mixing of $P(\Psi)$ can be approximated using the Monte Carlo method, mentioned in Chapter 1 for the simulation of the coalescence-dispersion term. An essential feature of the Monte Carlo computational technique is that at any location in the combustion chamber, the joint PDF is represented by an ensemble of fluid particles which are identified in terms of their individual thermodynamic states. The turbulent mixing process is described as a sequence of interactions between randomly chosen groups of n representative fluid particles which mix (with frequency β) completely with one another to form n new particles having an identical thermodynamic state. The simulation is allowed to continue until the ensemble statistics (mean and variance of properties) are observed to stabilize. Finally, the ensemble average and other statistics are obtained to represent the properties of interest. The Monte Carlo method is insensitive to the dimensionality of the PDF and the computational expense rises only linearly with σ , which is the best that can be achieved by any algorithm.

The combination of the solutions for the advection, diffusion and mixing equations produces an approximation to the joint PDF solution for the turbulent mixing problem. However, because of problems posed by closure approximations, the present study focuses primary attention on the numerical solution of the advection or bulk diffusion component equation. Eddy diffusion and molecular effects will be incorporated in a future study.

3.5 Choice of Numerical Parameters and Functions

Specifying threshold values for the following nine control parameters :

- (i) d , mesh size,
- (ii) d_j , injector nozzle diameter,
- (iii) r_0 , discrete vortex core radius,
- (iv) h , boundary discretization length,
- (v) Δt , length of time step,
- (vi) δ_s , numerical shear layer thickness,
- (vii) $\gamma_{\max}$, maximum allowable vortex circulation.
- (viii) l , initial number of "smaller" vortex sheet segments created at wall test points,
- (ix) β , mixing frequency,

and the proper choice of a vortex core smoothing function G_{ij} and a numerical filter function ϕ will produce a discrete vortex calculation which describes a real flow. If the evolution of the important features of the real flow are to be predicted, then an appreciation for the interplay between these control parameters must be developed.

Details of the numerical implementation of the foregoing theoretical analysis are presented in Chapter 5. The numerical experiments germane to the present study and discussion of results are presented in Chapter 6.

CHAPTER 4

NUMERICAL FILTERS

4.1 Introduction

Numerical filters serve a dual purpose in Vortex-In-Cell (VIC) formulation. They are used for vorticity deposition on the grid and for interpolation of local effects of the velocity field from the grid to the Lagrangian points. These interpolation procedures are subject to errors. The extent to which these errors are minimized is dependent upon the design of the filter. The choice of a carefully constructed filter for use in the VIC algorithm is therefore a critical factor in obtaining consistent numerical results. This choice is dictated by several considerations including accuracy, grid independence, a reduction of undesirable grid effects or aliasing, Fourier transformability and computational efficiency.

4.2 Description of Filters

A survey of the literature shows that there have been essentially five methods used to derive the proper numerical filter. Some have assumed a "shape" for the vortex core in real space while others have generated the shape in wave vector space to obtain some desirable features during the Fast Fourier Transform (FFT). A sixth

filter developed by Bartholomew [7] is included in this description. An important consideration is that the remainder of the fluid not sense that an interpolation has occurred, hence, the vorticity carried by each vortex should be spread to as few mesh points as possible.

The Cloud-In-Cell (CIC) method, first reported by Christiansen [20] and later used by Baker [6], assumes each point vortex to possess a uniform or "top hat" vorticity distribution within a square cell. The distribution involves the nearest two grid points in a given coordinate direction.

Meng and Thomson [49] used a slightly modified "top hat" profile of Christiansen. They assumed a vorticity distribution over a rectangle with dimensions x and y of one cell according to

$$\xi(x,y) \approx \frac{1}{\left[1 + \frac{3(x-x_j)}{\Delta x}\right]^2} \cdot \frac{1}{\left[1 + \frac{3(y-y_j)}{\Delta y}\right]^2} \quad (4.1)$$

where $\Delta x/3$ and $\Delta y/3$ are the half-widths characterizing the spreading and x_j, y_j are the coordinates of the j^{th} vortex. Like Christiansen's filter, the distribution involves the nearest two grid points in a given coordinate direction.

To improve the numerics of the CIC method, Hockney et. al. [41] proposed the use of a truncated-Gaussian core having a larger cross-section ($2d \times 2d$) for the

vorticity distribution function :

$$\begin{aligned}\xi(x_i) &= \exp \left\{ - \sum_{i=1}^m \frac{x_i^2}{2r_0^2} \right\} & |x_i| \leq d \\ &= 0 & |x_i| > d\end{aligned}\tag{4.2}$$

where $1 \leq i \leq m$, $r_0 = 0.455d$, d is the distance between mesh points, x_i is the coordinate in the i^{th} dimension relative to the vortex and m is the number of dimensions. This distribution involves the nearest three grid points in a given coordinate direction. That is, the Gaussian distribution is truncated beyond 1.5 grid lengths.

Couet [23,24] and Wang [70] developed their Gaussian distributions for the vorticity in wave vector space and then transformed them into real space. The results gave quadratic and cubic splines, respectively, and involve the nearest three and four grid points, respectively, in a given coordinate direction.

Bartholomew [7] assumed a polynomial expression to describe the fraction of the total circulation credited to the nearest three grid points in a given coordinate direction.

These distribution functions can be put into a common format so that in two dimensions the filter is the product of two one-dimensional filters. These one-dimensional filter models, the scheme used in generating them, the number of nearest

grid points involved for each scheme and their authors are listed in Table 1. It is from Bartholomew's perspective of using the definition of circulation that the compact forms of the filters, with the exception of Couet's and Wang's, were independently derived. Profiles of the six filters are illustrated in Figure 3. It is observed that the profile of Meng/Thomson's filter is in fact a slight modification of Christiansen's triangular profile. The others appear to assume a quasi-Gaussian profile which is brought smoothly to zero at the extent of each filter.

A generalization of the formulations listed in Table 1 can be made as follows :

A uniform rectangular grid system having coordinate directions x and y is first defined as shown in Figure 4. Next, an assumed vorticity distribution function

$$\xi(x,y) \approx f(x,y), \quad (4.3)$$

with the point vortex coordinates (x_p, y_p) taken to lie at the center of this distribution, is normalized, $\xi(\zeta, \eta)$, where $\zeta = x/d$ and $\eta = y/d$, and a local grid point which serves as the origin for the coordinates of the discrete vortex is chosen. The normalized filter $\phi(\zeta, \eta)$ which determines the fraction $(\gamma_p)_{i,j}$ of the total circulation γ_p that is credited to the surrounding grid points of interest is now obtained by integrating the normalized vorticity distribution over the overlapping area in the cell enclosing the grid point :

$$\phi(\zeta, \eta) = C \int \xi(\zeta, \eta) dA_{i,j} = \phi(\zeta)\phi(\eta) \quad (4.4)$$

and

$$(\gamma_p)_{i,j} = \phi(\zeta)\phi(\eta)\gamma_p \quad (4.5)$$

The quantity C is a constant, $dA_{i,j}$ is the area element of vorticity assigned to the grid point (i,j) of interest and $\phi(\zeta)$, $\phi(\eta)$ are one dimensional filters. The grid point which served as the origin is now generalized to give the compact expression for the filter. This interpolation scheme has been sometimes referred to as "area weighting" in the literature. An illustrative example of this procedure for Christiansen's filter is provided in Appendix A.

To reassemble the velocity information from the grid to the Lagrangian point of interest, each grid point in the neighborhood of the Lagrangian point is assigned a weighting value which shows the fraction that the Lagrangian point contributes to the total velocity field estimate.

Because the expressions $\phi(\zeta,\eta)$ for the filters are algebraically complicated, they are numerically evaluated. Two-point Gaussian-Legendre quadrature has been found to be sufficient for all the filters but Meng/Thomson's. In this case, a ten-point formulation is required for convergence.

4.3 Analysis of Filter Performance

Filter performance with regard to accuracy, grid independence, Fourier transformability and computational efficiency are discussed in this section.

One possible measure of the formulation's accuracy is its ability to satisfy integral constraints of the fluid motion such as the conservation of circulation. How

well the local circulation is conserved in the Eulerian reference frame can be tested analytically or numerically by varying the position of a single discrete vortex within a square cell. The circulation deposited to the nearest grid points of interest is then summed and checked to see if it reproduces the correct amount of circulation. Analytical results show that the total circulation is conserved exactly for all of the filters except Bartholomew's. Numerical evaluation of Bartholomew's filter gave an error on the order of 0.01%.

Grid independence refers to the fact that the VIC formulation can resolve length scales smaller than the grid scale used by FFT. For the formulation to be essentially grid free, a necessary requirement is for the spreading filter to exhibit a spherical symmetry that is unaffected by the existence of the grid points. This point is best understood from the following argument: consider a point in an infinite three-dimensional space. Let the point diffuse for a short time interval, and then "turn-off" the viscosity. The exact solution for this process, according to [14], shows that the amount of vorticity at any location is proportional to

$$\xi(x,y,z) \approx \exp\left(-\frac{r^2}{4\nu t}\right) \approx \exp\left(-\frac{x^2}{4\nu t}\right) \exp\left(-\frac{y^2}{4\nu t}\right) \exp\left(-\frac{z^2}{4\nu t}\right) \quad (4.6)$$

where r is the distance from the point vortex to the spatial point of interest (x,y,z) , ν is the kinematic viscosity and t is the time. This distribution is referred to as the "3D filter" in Figure 5. To obtain the corresponding two-dimensional result, one must distribute points along a line.

Figure 5 show iso-vorticity contours of the function $\phi(\zeta, \eta)$ for five of the filters as well as the 3D filter. The large tick marks on the figure represent a half grid length. Hockney's filter bears a striking similarity to Couet's filter and has therefore been omitted. It is evident that the 3D filter as well as Couet's and Wang's filters are essentially spherically symmetric, implying that they exhibit considerable subgrid resolution and therefore less numerical instability. The contours developed by Christiansen's, Meng/Thomson's and Bartholomew's filters tend to "square-off" into rounded corners which translate into a positional dependence that is not grid independent.

In addition to our concern for a spherically symmetric filter, one must be concerned with the behavior of the filter in transform space where a mesh can resolve only non-dimensional wave numbers less than 1.0. The Fourier transform of the filters are shown in Figure 6. Any harmonic with a non-dimensional wave number greater than 1.0 will be misinterpreted and thus will, in effect, add a mixture of overtones to the approximation of the pure harmonic. This mesh-induced fluctuation is known as "aliasing" and aliasing sets a limit at $(k/\pi)_{\max} = 1.0$ in transform space. Christiansen's and Meng/Thomson's filters produce the worst aliasing. Wang's filter transforms with virtually no "halos". However, even when the transform smoothly approaches zero, there are still a significant number of components outside a non-dimensional wave number of 1.0. This fact will prove troublesome when the inverse transform is performed.

Figure 7 shows the inverse transform of the filters when a non-dimensional wave number cutoff of 1.0 is used; with the exception of Wang's filter which uses a

cutoff of 1.5. The "halos" in these functions are due to the fact that the transform process can only resolve wave components through π modes. This problem would be essentially non-existent if the transform could include modes through 2π . In practice, the collection of vortices used in modeling a vortical region would result in a "smoothing" of the solution and an additional reduction in the "halos". The result of an error analysis between the original filter and the inverse of its transform for a wave number cutoff of 1.0 is represented in Figure 8. Bartholomew's, Couet's and Hockney's filters are seen to be most recoverable with error in the neighborhood of $\pm 6\%$.

A final criterion for choosing the optimal filter is based upon the central processor unit (CPU) times for evaluating the filter and for using the filter to spread vorticity to grid points. The results of these computational efforts are shown in Table 1. It is no surprise that from an execution speed point of view, Hockney's filter is least efficient since it requires 24 error function evaluations for spreading in 2D. Christiansen's filter evidently sacrifices accuracy for efficiency. On the average, the CPU times for Meng/Thomson's and Couet's filters compare quite well.

4.4 Summary

This investigation has highlighted the specific areas for consideration when deciding on the choice of a numerical filter. A good filter should be grid independent, conserve local circulation, minimize aliasing errors, be computationally accurate and efficient, and lastly, certainly not the least, be in a convenient compact form for easy implementation.

Based upon the preceeding discussion of spherical symmetry, Fourier transformability, compact representation, accuracy and computational efficiency, the filter developed by Couet represents, in this researcher's best judgement, the best choice of filter presently available in the literature. Couet's filter was therefore chosen for the VIC algorithm employed in this study to both spread the vorticity and to reconstruct the velocity field.

CHAPTER 5

NUMERICAL METHOD

5.1 Conformal Transformation

The application of conformal transformation in potential theory provides the **flexibility** for performing calculations in a simpler domain. In this study, conformal **transformation** of position and velocity is a necessary requirement imposed by the **choice** of the VIC formulation. The VIC algorithm is implemented on a uniform square **grid** and the resulting velocity field is transformed onto an equivalent non-uniform grid **system** in the circular domain in order to advance the solution in time. Computational **economy** dictates that only the mesh record of the position and velocity information be **transformed**. Information at off-grid locations is obtained via an interpolation scheme.

5.1.1 Grid transformation

The basis for the transformability of the grid systems shown in Figure 9 is **provided** [11] by the equation

$$z = \frac{\operatorname{sn}(\omega)\operatorname{dn}(\omega)}{\operatorname{cn}(\omega)} \quad (5.1)$$

where $z = x + iy$ and $\omega = \zeta + i\eta$; the coordinates (x,y) and (ζ,η) are non-dimensional grid point coordinates in the circular and square domains with boundaries S and S' respectively; the functions $\text{cn}(\omega)$, $\text{dn}(\omega)$ and $\text{sn}(\omega)$ are Jacobian elliptic functions with complex arguments.

The transformation (5.1) will transform the square $-K/2 \leq \zeta \leq K/2$, $-K/2 \leq \eta \leq K/2$ onto the unit circle $|z| = 1$. The complete elliptic integral of the first kind K , defined as

$$K = \int_0^{\frac{\pi}{2}} \frac{d\alpha}{\sqrt{1 - k^2 \sin^2 \alpha}} \quad (5.2)$$

where the modulus k is a constant, is evaluated using Gauss-Legendre quadrature. Note that the length of the square is equal to the value for K . The parameter α is a dummy variable. The square domain is divided into $M \times M$ grid squares giving the grid point coordinates (ζ,η) . M is the desired number of grid points in the ζ and η directions. The corresponding grid point coordinates (x,y) are determined by first reducing Eq.(5.1) into a system of non-linear algebraic equations of the form

$$x = f(c, c_1, d, d_1, s, s_1, k) = f(\zeta, \eta, k) \quad (5.3)$$

$$y = g(c, c_1, d, d_1, s, s_1, k) = g(\zeta, \eta, k) \quad (5.4)$$

where $c = \text{cn}(\zeta, k)$, $c_1 = \text{cn}(\eta, k')$, $d = \text{dn}(\zeta, k)$, $d_1 = \text{dn}(\eta, k')$, $s = \text{sn}(\zeta, k)$ and $s_1 = \text{sn}(\eta, k')$ are Jacobian elliptic functions with real arguments. The complementary modulus $k' = \sqrt{1-k^2}$. Equations (5.3) and (5.4) are derived in Appendix C. For a square domain, it can be shown that $k'^2 = k^2 = 0.5$. Next, the elliptic functions are numerically evaluated using a descending Landen transformation [11] and substitution in (5.3) and (5.4) gives the grid point coordinates (x, y) . The resulting corner singularities in the circular domain (see Figure 9) would prove to be a troublesome factor with regard to interpolation procedures. This point is discussed further in chapter 6.

The reverse transformation, $(x, y) \rightarrow (\zeta, \eta)$, involves an elaborate four-step procedure :

- (i) The functions $s(\zeta, k)$ and $s_1(\eta, k)$ are pretabulated.
- (ii) Equation (5.1) is reduced to another system of non-linear, algebraic equations of the form (see Appendix C)

$$f(k, s, s_1, x) = 0 \quad (5.5)$$

$$f(k, s, s_1, y) = 0 \quad (5.6)$$

- (iii) Equations (5.5) and (5.6) are then solved simultaneously for s and s_1 , for every x and y input, using a quasi Newton-Raphson numerical method.

- (iv) The coordinates (ζ, η) are recovered from the pretabulated functions $s(\zeta, k)$ and $s_1(\eta, k)$ via linear interpolation.

5.1.2 Velocity transformation

Let a fluid motion inside the square region of the ω -plane be given by the complex potential,

$$W(\omega) = \phi + i\psi \quad (5.7)$$

where ϕ is the velocity potential and ψ is the stream function. Then at corresponding points z and ω , given by Eq. (5.1), W and therefore ϕ and ψ take the same values. Now, S' is a boundary and so a streamline, and therefore $\psi = C$, a constant, at all points of S' . Since S corresponds point by point with S' , $\psi = C$ at all points of S . Therefore S is a streamline, in the motion given by (5.7) and (5.1) together, in the z -plane.

The actual form of the complex potential in terms of z would be obtained by eliminating ω between (5.1) and (5.7), but it is often preferable to look on z as a parameter and forgo the elimination. Thus, to find the fluid velocity at (x, y) in the z -plane corresponding with (ζ, η) in the ω -plane, we have

$$\frac{dW}{dz} = \frac{dW}{d\omega} \cdot \frac{d\omega}{dz} \quad (5.8)$$

The equivalent matrix form of Eq. (5.8) can be written as

$$\begin{bmatrix} u \\ v \end{bmatrix}_{(x,y)} = \begin{bmatrix} e_{11} & e_{12} \\ -e_{12} & e_{11} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}_{(\zeta,\eta)} \quad (5.9)$$

where e_{11} , e_{12} are elements of the 2×2 matrix. The velocity components in the z -plane can be transformed to give velocity components in the cylindrical coordinate system as :

$$V_\theta = v \cos \theta - u \sin \theta \quad (5.10)$$

$$V_r = -v \sin \theta - u \cos \theta \quad (5.11)$$

Conversely, the fluid velocity at (ζ, η) in the ω -plane corresponding with the fluid velocity at (x, y) in the z -plane is given by

$$\frac{dW}{d\omega} = \frac{dW}{dz} \cdot \frac{dz}{d\omega} \quad (5.12)$$

or,

$$\begin{bmatrix} u \\ v \end{bmatrix}_{(\zeta,\eta)} = \begin{bmatrix} e_{11} & -e_{12} \\ e_{12} & e_{11} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}_{(x,y)} \quad (5.13)$$

Notice that the matrix system in Eq. (5.13) is the inverse of that in Eq. (5.9).

Representation of the grid velocity field in the circular domain is done via conformal transformation as discussed above. Off-grid point velocity calculation in either domain can be accomplished using an interpolation routine relevant to each grid system.

5.2 Point Vortex Initialization Scheme

The irrotational vorticity field at the initial time can be approximated using discrete vortices in two ways. The first scheme involves analyzing the problem from the point in time when the mechanism that is responsible for the production of vorticity within the boundary layer first introduced vorticity into the interior domain to the point where the flow field matches the desired initial conditions. This method is computationally expensive. It would be computationally more advantageous to develop a method that generates the initial distribution and strengths of the vortices from a prescribed velocity field directly, or alternately, from the derivative of the velocity field. In either case an iterative method is implied.

The initialization scheme employed in this study makes use of the tangential velocity field prescribed by the experimental data of Dyer [33]; see Table 2. The data is classified according to a time τ_0 which is the time from the closing of the shrouded intake valve to the time of measurement. Dyer's data were used to represent different initial swirl conditions for this study. The strengths and optimum distribution of the discrete vortices are determined as follows :

- (i) Discrete vortices are arbitrarily but uniformly placed in concentric circles about the origin of the circular domain.
- (ii) The modified circle theorem of Milne-Thomson is applied to the system of vortices to compute geometric factors and using Dyer's velocity data, a matrix inversion routine supplied by IMSL is used to determine the strengths of the vortices.
- (iii) The optimum distribution is found when the sum of the circulation due to each vortex is equal to the global circulation $2\pi R \overline{V_\theta}$, i.e.

$$\sum_{j=1}^N (\gamma_p)_j = 2\pi R \overline{V_\theta} \quad \text{or} \quad \sum_{j=1}^N \kappa_j = R \overline{V_\theta} \quad (5.14)$$

where N is the number of discrete point vortices, $(\gamma_p)_j$ is the circulation of vortex j , κ_j is the strength of vortex j , R is the radius of the combustion chamber and $\overline{V_\theta}$ is the average tangential velocity or slip velocity obtained by numerically integrating Dyer's experimental velocity data.

To obtain the strengths of the discrete point vortices, consider an arbitrary distribution of the vortices as noted above. The velocity at any point z_k inside the circular domain is the sum of the velocities due to all vortices located at z_j given by

the modified circle theorem :

$$[u - iv]_k = \sum_{\substack{j=1 \\ j \neq k}}^N -i\kappa_j \left[\frac{1}{z_k - z_j} - \frac{1}{z_k - a^2/\bar{z}_j} \right] \quad (5.15)$$

Equation (5.15) is composed of three factors : the desired velocity $(u-iv)_k$, the discrete point vortex strength κ_j and a geometric factor relating the influence of vortex j at point z_k . This information can be represented in matrix form as

$$(V_\theta)_k = F_{kj} S_j \quad (5.16)$$

where $(V_\theta)_k$, F_{kj} , S_j are matrices for the prescribed tangential velocities of the swirl at the points z_k , geometric factors and vortex strengths respectively. Solving for the strengths by inverting the F_{kj} matrix, we get

$$S_j = F_{kj}^{-1} (V_\theta)_k \quad (5.17)$$

See the Appendix D for the development of the matrices.

5.3 Vortex-In-Cell (VIC) Algorithm

A VIC technique is essentially a formulation that combines the Lagrangian vortex tracking concept with the stream function-vorticity finite difference

formulation. A distinguishing feature of the VIC method is the fact that vorticity information is contained in the Lagrangian vortex element rather than on the Eulerian mesh. The Eulerian mesh is introduced for fast Fourier transform and convenient representation of results, but is not explicitly used to advance the solution in time as would be done in the finite difference method.

As a first step towards solving Poisson's equation (Eq. 3.8) on the square of fixed mesh size d , we must create, at the beginning of each time step, a mesh record of the vorticity field starting from the Lagrangian representation. This is done by an interpolation scheme using Couet's filter

$$\varphi(\eta) = \left\{ \begin{array}{ll} \frac{1}{2} \left[\frac{3}{2} - |\eta|^2 \right] & ; \quad \frac{1}{2} \leq |\eta| \leq \frac{3}{2} \\ \frac{3}{4} - \eta^2 & ; \quad |\eta| \leq \frac{1}{2} \\ 0 & ; \quad \text{Otherwise} \end{array} \right\} \quad (5.18)$$

where $\eta = y/d$ or x/d , to distribute the vorticity from each vortex over the nearest three grid points in a given coordinate direction.

Figure 10 illustrates an application of the above one-dimensional model of Couet's filter. Three quadratic spline distributions of vorticity with different total amplitudes are shown to have their centers located at positions a, b, and c. The three nearest grid points to each of the centers will share the vorticity distribution according to the spline function weighting on them. Grid points N-2 to N will share the vortex "a" with grid points N-1 receiving the largest weight. Similarly, points N-1 to N+2 will share the vortices "b" and "c".

After the vorticity information is represented on the uniform grid, a grid stream function can be generated by using FFT techniques to invert Poisson's equation (3.8). The stream function is centrally differenced on the grid to produce a velocity field which is conformally transformed onto an equivalent grid system in the circular domain. The velocity at the vortex position is now obtained by interpolation. The vortices are then convected to new positions and the procedure is repeated each time step until the flow evolution is completed.

The convenience of FFT is available only when boundary conditions are imposed over the square domain. In order to satisfy the no slip normal boundary condition, the value $\psi = 0$ is specified on the grid boundary for this study. The grid used is a modest 60×60 but extension to larger dimensions is feasible with the use of mainframe computers. Swartztrauber, et. al. [68] have developed an elliptic boundary value problem solver (FISHPAK) which allows for general boundary conditions and is portable, i.e., it runs on a Cray, Cyber, IBM, or Vax computer with no modifications. This software is used in this study to accomplish the Poisson inversion.

In general, if the grid is $M \times M$, this is an order $M^2 \log_2 M$ calculation. In a typical 2D calculation, the number of grid squares $M \times M$ is of the same order of the total number of vortices N and thus FFT requires on the order of $N^2 \log_2 N$ operations per time step. The interpolations take of the order N operations per time step. When N is large, $N^2 \log_2 N$ is considerably smaller than N^2 . Thus the VIC technique offers a resolution of the problem associated with excessive computation time.

5.4 Vortex Sheet Algorithm

The velocity field in the boundary layer region can be constructed using vortex sheets as described in Section 3.2.2. The u and v component velocities are given by Eqs. (3.26) and (3.27). At the initial time, the tangential velocity V_θ along the wall is fixed by the potential flow solution of Eq. (3.8). Theoretically, the tangential no slip boundary condition, $V_\theta=0$, can be satisfied by generating a continuous sheet of vorticity along the wall with circulation $\gamma_s = -V_\theta$ per unit length of the wall. However, it has been observed computationally [2,18] that $\gamma_s = -V_\theta$ will satisfy the tangential no-slip boundary condition only in the limit as $\Delta t \rightarrow 0$.

In order to satisfy the boundary condition exactly (except possibly at a finite set of values for t), Chorin [18], algorithmically, extended the velocity field across the wall ($y=0$) at the beginning of each time step by anti-symmetry $u(x,-y) = -u(x,y)$. As a result $\xi(x,-y) = \xi(x,y)$ for $y \neq 0$, and a continuous vortex sheet of circulation $\gamma_s = -2V_\theta$ per unit length of the wall appears at $y = 0$ if the tangential velocity V_θ does not vanish at the wall. This sheet is segmented into elements of finite length h , centered at test points along the wall, specifying the spatial resolution in the x -direction.

In order to have a reasonable approximation of the diffusion equation (3.7), the resolution in the y -direction can be improved by replacing each vortex segment by 1 number of "smaller" segments with circulation $(\gamma_s)_j = \gamma_s/l$ such that $|(\gamma_s)_j| \leq |\gamma_{\max}|$

where $\gamma_{\max}$ is a predetermined value. Vortex elements from each segment h diffuse normal to the wall via random walks and induce a velocity field which advects them downstream. At subsequent time steps, V_θ is calculated using the obvious specialization of Eq.(3.26). The correction term $(\gamma_s)_j/2$ in Eq. (3.26) was added by Chorin [18] to account for the effect of the image of the discrete point vortex the sheet can generate (see Figure 12).

Unlike the "elliptic" flow modeled by point vortices, where the effect of each point vortex extends throughout the field, the zone of influence of a vortex sheet is restricted to the "shadow" below it, a consequence of Eq.(3.24). Thus, Eqs. (3.26) and (3.27) imply that the sum is over all vortex sheets whose "shadow" on the wall engulfs the sheet whose center is located at (x_j, y_j) . This is illustrated in Figure 11. The darker the shadow, the smaller $u(x, y)$ becomes. Whatever fluid enters a shadow region from the left and cannot leave on the right must leave upwards.

To reduce the total number of sheets retained each time step, the vortex sheet algorithm can be implemented such that exactly half of them disappear. This is accomplished as follows : at each wall test point an even number of sheets are created and their intensity is adjusted such that if $|(\gamma_s)_j| < |\gamma_{\max}|$ no sheets are created. A rejection technique [18] is then used to ensure that any two successive η_2 's used at the wall will have differing signs. This rejection technique can be used only at the wall, or else it would destroy the independence of the successive η_1 's in the interior and thus fail to describe the diffusion process correctly.

A tagging and variance reduction technique can be used to effect a substantial reduction in the statistical error. Since diffusion takes place only in the y-direction, the random numbers $(\eta_2)_j$ used in (Eq. 3.33) need be independent of each other only when they are used with vortex sheet segments whose centers lie in a narrow strip perpendicular to the wall. As vortex sheets are created, they are assigned integer tags, m_j being the tag assigned to the j^{th} vortex sheet segment. All elements with the same η_2 are assigned the same tags. The effect of tagging is to piece together the elements created at the several boundary points into coherent vortex sheets with the elements of each sheet identified by a common tag.

5.5 Advection Scheme

Vortices can be advected in only two ways, by convection or by diffusion. Once the local velocity is known, the convection step is accomplished by using an appropriate time integration scheme. For vortices in the interior region, Heun's time integration method is used to evaluate the integrals in Eq.(3.17) and Eq.(3.18), namely

$$x_j^{n+1} = x_j^n + \frac{\Delta t}{2} \left[u(x_j^n, y_j^n) + u(x_j^*, y_j^*) \right] \quad (5.19)$$

$$y_j^{n+1} = y_j^n + \frac{\Delta t}{2} \left[v(x_j^n, y_j^n) + v(x_j^*, y_j^*) \right] \quad (5.20)$$

$$\text{where } x_j^* = x_j^n + \frac{\Delta t}{2} \left[u(x_j^n, y_j^n) \right] \text{ and } y_j^* = y_j^n + \frac{\Delta t}{2} \left[v(x_j^n, y_j^n) \right] \quad (5.21)$$

Heun's technique is second order accurate in time. Such accuracy lessens the tendency for discrete vortices to spread out in spinning motion as they come close together. A first-order scheme merely takes the tangent at a given point and constructs a straight line approximation to the trajectory, in effect spreading out the vortices. Since Heun's method requires two evaluations of the velocity field per time step, this can prove expensive when using the direct $O(N^2)$ method of calculating vortex interactions. The FFT technique is advantageous in this respect.

Near the solid boundary, an explicit Euler integration which is first-order accurate can be used to evaluate the integrals in Eq. (3.32) and (3.33) in order to accomplish the convection step, namely

$$x_j^{n+1} = x_j^n + u_j^n \Delta t \quad (5.22)$$

$$y_j^{n+1} = y_j^n + v_j^n \Delta t \quad (5.23)$$

The use of a second-order time integration scheme for the vortex sheets is not necessary for two reasons. First, the convection field within the boundary layer is essentially a one-dimensional flow in a direction parallel to the wall and thus poses no great problem for a first-order scheme. The main component driving the sheets away from the wall is the diffusion term. Second, the implementation of a second-order scheme is a very complicated procedure, since sheets may turn into point vortices during the predictor step and thus must be treated differently.

It can be assured that the vortices are not convected beyond one mesh width (for discrete point vortices) or sheet length (for vortex sheets) as long as the Courant condition

$$|v_{\max}| \frac{\Delta t}{2} \leq d \text{ or } h \quad (5.24)$$

is satisfied. The Courant condition sets an upper bound on Δt for numerical stability.

Diffusion as discussed by Chorin [18] is equivalent to a random walk which is governed by statistical laws derived from the solutions of the diffusion equation (3.7). The integration of this equation yields a Gaussian distribution for the flux of vorticity [40]. Tschebysheff's theorem [44] states that the Gaussianly distributed random variable with mean zero will rarely be greater in magnitude than three standard deviations. During a time step of duration Δt , vorticity will diffuse with a standard deviation of $\sqrt{2\nu\Delta t}$. Hence, in a time step, interior vortices will not likely diffuse beyond one mesh width if $3\sqrt{2\nu\Delta t} \leq d$. In some physical sense, statistical simulation of diffusion steps back from the concept of a material continuum, and employs a sort of kinetic theory of giant and imaginary particles which wander randomly through the region of interest carrying macroscopic elements of the diffusing field. The outcome of a calculation is a list of particle positions and the elements of the concentration field they are employed to transport.

5.6 Sorting Algorithm

Vortex sheets and discrete vortices are objects of a very similar nature; they are both determined by position and circulation. A computational element (x_j, y_j, ξ_j) can be treated as either a sheet or a discrete point vortex, depending on the circumstances. A sheet of negative circulation casts a shadow which slows the fluid under it; by the equation of continuity, this creates an upward flow to the left and a downward flow to the right, just as if the sheet were a point vortex. Conservation of circulation arguments show that a sheet segment of circulation γ_s should transform into a point vortex of circulation $\gamma_p = \gamma_s h$, where h is the length of the sheet.

These facts can be used to create a transition between the discrete vortices and the vortex sheets. Following Chorin's argument [18], vorticity is transferred from the boundary layer to the interior by allowing those vortex sheets located a distance greater than δ_s from the wall to become point vortices. Any vortex which finds itself less than δ_s from the solid boundary (inside or outside) becomes a sheet or is reflected accordingly (see Figure 12). Reflection of any sheet which crosses the wall back into the fluid imposes the antisymmetry condition. By taking δ_s large relative to the variance of the steps, it is unlikely that a discrete point vortex will move further outside the domain than δ_s in one time step; if this does occur, it is removed.

5.7 Mixing Algorithm

The physics of the evolution of scalar field undergoing turbulent and molecular mixing is embodied in the PDF transport equation (3.47). The Lagrangian view of molecular mixing can be numerically simulated using the Monte Carlo coalescence and dispersion (C/D) scheme.

The initial bimodal joint PDF, $P(\psi; \mathbf{x}, 0)$, can be characterized by the random distribution inside the combustion chamber of N equal-mass, chemically-inert fluid particles with values 0 designating the swirling air particles and 1 designating the injected air particles. The fluid particles are idealized turbulent eddies (i.e., finite-cored vortices) assumed to be larger than molecular dimensions, but sufficiently small with respect to the turbulence microscale. The thermodynamic state of each particle (i.e., pressure, mass fraction, temperature, etc.) is assumed to be uniform throughout its volume at any instant of time, but the number of particles in a given state may vary with time as a result of mixing between particles.

The Monte Carlo C/D scheme is a stochastic fluid particle interaction model. Each fluid particle is assigned a number index, $1 \leq i \leq N$, and a pseudo-random number generator is used to generate n different indices, $i_1, i_2, \dots, i_n$ ($i_j \neq i_k$) thereby identifying the particles to be involved in the first interaction. In the simplest binary C/D mixing model ($n=2$) proposed by Curl [26] pairs of fluid particles are allowed to mix completely by adopting the average values of their properties and then separate. The mixing interactions are computed sequentially; the first interaction representing a

forward step in time equal to

$$\Delta t_{(\text{mix})} = \frac{1}{\beta(t)N} \quad (5.25)$$

where $\beta(t)$ is the empirically determined, time dependent mixing frequency and N is the fluid particle population in the combustion chamber.

A new group of pairs of fluid particles is next chosen at random, from the entire ensemble of N fluid particles, to be involved in the next mixing interaction; this way the calculation proceeds forward in time inside the combustion chamber. At any time during the mixing process, the mean composition and other mean properties of the fluid may be evaluated by taking an ensemble average over N fluid particles. Furthermore, the shape of the evolving distribution function $f(\Phi, t)$ may be determined by plotting the ensemble population as a function of any one of the thermodynamic state variables or any other local property.

CHAPTER 6

NUMERICAL EXPERIMENTS AND DISCUSSION OF RESULTS

6.1 Introduction

This chapter documents the numerical experiments conducted to evaluate the **various** aspects of the inert scalar mixing study. The results of the experiments are **discussed** and compared with theoretical predictions. The experiments can be broadly **classified** into three sets :

- (i) Optimization of Numerical Parameters
- (ii) Validation of Numerical Codes
- (iii) Numerical Simulation.

The computational analyses were conducted on a VMS/VAX 11/750 computer **system**. Several Fortran programs (see Appendix F) used to analyze the different **components** of the study. A portable elliptic boundary value problem solver (**FISHPAK**) which allows for general boundary conditions, developed by Swartztrauber **et. al.** [68], IMSL routines, PLOTT and NCAR graphics packages were the basic

computational tools used in these analyses. Some of the graphs were generated using the FrameMaker utility on a Hewlett Packard workstation.

6.2 Optimization of Numerical Parameters

The relevant numerical parameters in the present study are the mesh size d , the jet nozzle diameter d_j , the discrete vortex core radius r_0 and the length of the time step Δt . The relevant numerical functions are the vortex core smoothing function or Green's function G_{ij} , and the filter function ϕ . The accuracy of the exact and numerical schemes presented in this study is dependent on these parameters, the functions G_{ij} and ϕ . The considerations leading to the choice of Hald and Del Prete's vortex core smoothing function G_{ij} and Couet's filter function ϕ have been discussed in chapters 3 and 4 respectively. In this section, error estimates guiding the choice of d , d_j , r_0 and Δt are considered.

6.2.1 Mesh size

Because the VIC formulation incorporates the streamfunction-vorticity finite-difference scheme, the question of grid resolution must be addressed. The accuracy of a finite difference scheme depends on the scale of the grid spacing (mesh size) relative to the scale of the exact solution or features of the real flow which is being approximated numerically. To obtain a meaningful solution, the mesh size must be small enough to resolve the flow adequately. Theoretically, convergence of the numerical solution with exact solution should occur as the mesh size tends to zero.

The choice of the mesh size d used in this study is based on a comparison of **the** relative errors in the grid point stream function values between exact and **numerical** solutions in the square domain. The percent relative error was computed **as a** function of the mesh size by :

$$\text{Relative error (\%)} = \left[\frac{\sum_{i=1}^M \sum_{j=1}^M (\psi_E - \psi_N)^2 \cdot d^2}{\sum_{i=1}^M \sum_{j=1}^M \psi_E^2 \cdot d^2} \right]^{1/2} \times 100 \quad (6.1)$$

where M is the number of grid points in the ζ and η directions. The numerical stream function, $\psi_N = \psi_N[\zeta(id, jd), \eta(id, jd)]$, is computed via the VIC algorithm while the exact stream function, $\psi_E = \psi_E[\zeta(id, jd), \eta(id, jd)]$, is obtained using equation (3.44). The **parameters** $\zeta(id, jd)$ and $\eta(id, jd)$ are the computed grid point positions.

Two tests were performed. The first compared the errors in the stream function field induced by a single discrete point vortex of strength $\kappa=2\pi$, placed **centrally** in the computational grid. The second examined the effect of a system of 105 discrete point vortices ($0.0 < \kappa < 1.06$) initialized in accordance with the initialization scheme discussed in section 5.2. Plots of the percent relative errors for the various **choices** of grid sizes are presented in Figure 13 . The plots confirm the theoretical **predictions** that the numerical approximation will converge to the exact solution as **the** mesh size tends to zero. This tendency is much more gradual for the second test case, especially for grid sizes within the range 60×60 to 120×120 . The choice of grid size anywhere in this range will produce comparable errors. From the practical

standpoint of computational time and storage, a modest 60×60 grid size ($d = 0.0625$ cm.) corresponding to 3721 computational grid points was decided upon. The indicated relative error for this grid size is approximately 6% for the single vortex test case and 0.3% for the system of 105 vortices. The peak-to-peak variation observed in the second test suggests some interactions between neighboring vortices. These interactions act to reduce the relative error by as much as a factor of 20 when compared with the result for the single vortex test case. One should note that in practice, computations will rarely deal with an isolated vortex. Instead, a collection of vortices will be used to model a vortical region. Thus for the chosen mesh size, convergence is assured to a reasonable degree.

6.2.2 Injector nozzle diameter

No particular emphasis was placed in regard to the details of the injector nozzle design as these are not germane to the overall objective of this study. An optimization scheme for the injector nozzle diameter was not feasible in the absence of an exact jet flow solution. It is known however that for appropriate resolution of the jet features, the mesh size should be comparable to or smaller than the injector nozzle diameter. An injector nozzle diameter d_j equivalent to the mesh size d was deemed satisfactory for this study.

6.2.3 Vortex core radius

Turbulent flows can be approximated by the vortex method with many computational vortices used per physical vortex (see Ashurst [2]) or the amount of labor can be reduced by choosing computational vortices which agree in structure and spacing with the physical vortices. In either case, an assumption is made concerning the local vorticity distribution within the computational vortex element and this leads to a discussion of the vortex core cut-off, r_0 . There is considerable difference of opinion in the literature as to the optimal smoothing parameter or cut-off. Chorin and Bernard [19] have observed that the results of the computations are quite insensitive to the exact details of the smoothing. Shestakov [64] and Chorin [18] use fairly large values proportional to the length of the segments into which the boundary was divided ($r_0 \sim h$). This selection is connected with the formation of new vortices at a grid wall in order to satisfy the tangential no-slip boundary condition. On the other hand, Millinazzo and Saffman [50] believe that the cut-off r_0 should be as small as possible and use $\beta/50$, where β is the average distance between vortices. Saffman questions the relevance of Chorin's criterion for the diffusion and motion of the vortices away from walls. Hald and Del Prete [39] have proved some interesting convergence results. In particular, they have shown that the optimum cut-off depends upon the smoothness of the flow under consideration, and that for smooth flows, as the number of vortices increases, r_0 should be of order $\beta^{2/3}$. This implies that r_0 tends to zero more slowly than β ; that is, there must be an ever increasing overlap of the circular vortices in order to maintain high accuracy in approximating a continuum of vorticity. This is natural, since as the vortices become packed closer together, the

effect of the singularity in the fundamental solution of the Poisson equation [Eqn. (3.8)] becomes more pronounced and requires ever greater smoothing.

In light of the disagreement in the literature over what constitutes an acceptable value for the vortex core cut-off r_0 , an empirical procedure has been devised in this study to optimize r_0 . Since the vortex core cut-off is needed to evaluate Green's function [Eq.(3.21)] in the modified exact solution scheme and the VIC numerical solution scheme is independent of r_0 , the procedure seeks to vary r_0 and compare at a fixed initial time, the exact and numerical grid point solutions for the tangential component of velocity induced by a single discrete point vortex of strength $\kappa=2\pi$ in the circular domain. The choice of vortex strength is arbitrary. A value for r_0 that yielded the smallest error was considered the optimal choice for r_0 for use in this study.

The numerical solution was initiated in the square domain followed by the transformation equation (5.11) which provides the corresponding solution in the circular domain. A test grid 50×50 was chosen, within the 60×60 computational square domain, so the influence of the boundary would not be felt. This was necessary so that the entire quasi Gaussian distribution of each vortex lies within the square domain. For each arbitrarily chosen r_0 , the absolute value of the percent relative error at each grid point was measured as

$$\text{Relative error (\%)} = \left| \frac{(v_\theta)_E - (v_\theta)_N}{(v_\theta)_E} \times 100 \right| \quad (6.2)$$

and the minimum and maximum values noted. The parameters $(V_\theta)_E = (V_\theta)_E[x(id,jd)]$ and $(V_\theta)_N = (V_\theta)_N[x(id,jd)]$ are the exact and numerical tangential velocities respectively at the computed grid points $x(id,jd)$. Note that the numerical solution is independent of the cut-off radius r_0

Two tests cases were considered. In the first test, the vortex was located at the grid point in the middle of the square domain for symmetry. In the second test, the vortex was located at the center of one of the four cells surrounding the centrally located grid point. In each case, an equivalent vortex location in the circular domain was used for computing the exact solution. The results of the calculations for all grid points and for all grid points sufficiently removed from the vortex singularity are shown in Table 3. Note that the criterion "sufficiently removed from vortex singularity" is assured for grid points greater than two mesh lengths from vortex position at a grid point and for grid points greater than one and a half mesh lengths from vortex position at a cell center. When all grid points are considered the maximum error was observed to be minimum at $r_0 = 0.036$ for grid point location and $r_0 = 0.048$ for cell center location. For grid points sufficiently removed from the vortex location, the trend in this case suggests that convergence would occur in the limit as r_0 tends to zero. For equivalent r_0 , the maximum errors were lower than for all grid points by less than factors of 4 and 6 for grid point and cell center locations, respectively. This was anticipated. It would seem reasonable to choose an average value of $r_0 = 0.042$. However, $r_0 = 0.05$ was considered convenient for this study. The associated errors were comparable and within acceptable range. The minimum relative error was zero for each consideration in both test cases.

li

5.

tr

ex

tr

Fi

Se

ra

ca

vor

the

line

that

6.2.4 Time step

A general guideline for choosing a value for Δt is not discussed in the literature. Instead, an upper bound for Δt is given by the Courant condition (Eq. 5.24). In this study, the value of the time step Δt is optimized by observing the trajectory of a single discrete point vortex in the circular domain. Theoretically, it is expected that the vortex, whose motion is induced by its own velocity field, would trace a circle in one complete revolution. An error results if the initial radial position r_i of the vortex differs from the final radial position r_f after a complete revolution. Several tests were performed using a vortex of strength $\kappa=2\pi$. For each chosen radial location, Δt is varied and the relative error after one complete revolution is calculated as

$$\text{Relative error (\%)} = \left| \frac{r_f - r_i}{r_i} \times 100 \right| \quad (6.3)$$

The results of these tests are shown in Figure 14. It is observed that the vortex motion in the central portion of the domain ($r < 0.6$) is relatively insensitive to the time step and that convergence would occur as Δt tends to zero, which is again in line with theory. The time step $\Delta t=0.001$ seconds was chosen sufficiently small, so that a decrease in the time step would not significantly affect the results.

To summarize, the threshold values for the control parameters and the numerical functions employed in this study are listed below :

$$\text{Grid size, } M \times M = 60 \times 60$$

$$\text{Mesh size, } d \times d = 0.0625 \text{ cm.} \times 0.0625 \text{ cm.}$$

$$\text{Injector nozzle diameter, } d_j = 0.0625 \text{ cm.}$$

$$\text{Vortex core cut-off, } r_0 = 0.05 \text{ cm.}$$

$$\text{Time step, } \Delta t = 0.001 \text{ sec.}$$

Couet's filter :

$$\varphi(\eta) = \begin{cases} \frac{1}{2} \left[\frac{3}{2} - |\eta|^2 \right] & ; \quad \frac{1}{2} \leq |\eta| \leq \frac{3}{2} \\ \frac{3}{4} - \eta^2 & ; \quad |\eta| \leq \frac{1}{2} \\ 0 & ; \quad \text{Otherwise} \end{cases} \quad (6.4)$$

Vortex core smoothing function :

$$G_{ij} = \left[C_1 \left(1 - \frac{|z_i - z_j|^3}{r_0^3} \right) - C_2 \left(1 - \frac{|z_i - z_j|^2}{r_0^2} \right) + \log r_0 \right] ; \quad |z_i - z_j| \leq r_0 \quad (6.5)$$

6.3 Validation of Numerical Codes

The numerical codes consist of the conformal transformation, point vortex initialization, velocity and vortex trajectory algorithms. Each component was tested for accuracy by comparison with exact data.

6.3.1 Conformal transformation

The validity of the conformal grid point coordinate and velocity transformation schemes discussed in chapter 5 was tested using the reverse transformation procedure for the grid point coordinates and by comparing exact with numerical solutions for the velocities in the circular domain. The resulting error statistics are listed in Table 4. The large maximum relative errors noted for the velocity components result from the exact values being zero at the test point. Thus to avoid dividing by zero a small number ($1.0\text{E-}06$) is added to the denominator.

6.3.1.1 Grid transformation

The distribution of the non-uniform grid points in the circular domain is induced by the coordinate transformation equation (5.1) starting with the uniform grid points in the square domain (see Figure 9). The reverse coordinate transformation seeks to recover the uniform grid system from the non-uniform grid system. As stated in chapter 5, this involves the use of pretabulated Jacobian elliptic functions, intensive algebraic manipulations, a quasi Newton-Raphson numerical method and linear interpolation.

A smaller error is observed (see Table 4) in the results for the η -coordinate than for the ζ -coordinate. This asymmetry in error could be attributed to at least two factors. First, the construction, by hand, of the algebraic expressions for the system of equations (5.3) through (5.6) is quite a substantial challenge (see Appendix C). The algebraic complexities offer many opportunities for mistakes both in their derivations and in their coding for Fortran programming. Secondly, the use of Newton-Raphson's method brings with it certain difficulties, e.g., the lack of convergence because of poor initial guesses. The hope is that the initial guesses are close enough to the actual solutions and that the Jacobian matrix of the system of equations (5.5) and (5.6) is continuous and invertible at the solution points.

6.3.1.2 Velocity transformation

The errors resulting from the velocity transformation scheme were calculated by comparing the exact solutions obtained using the modified circle theorem for the system of 105 vortices with numerical solutions resulting from the transformation equation (5.11). Because of algebraic complexities, the elements of the matrix system in equation (5.11) were obtained by numerically differentiating equation (5.1). The resulting error is smaller for the v-component velocity than for the u-component velocity and the values in general are more symmetric than for the coordinate transformation scheme. This is perhaps attributable to the fact that the algebraic manipulations for the velocity transformation scheme is less prone to mistakes.

In line with expectation, the result of the error calculation shows comparatively poor convergence at the corners of the domains because of the singularities inherent in the coordinate transformation scheme. Outside of these singular regions excellent convergence was obtained, the constraints of algebraic complexities and initial guesses implicit in the Newton-Raphson numerical method notwithstanding.

6.3.1.3 Corner singularities

The corner singularities precluded velocity vector and streamline representations of the grid point results in the circular domain. Clearly, interpolation schemes appropriate to the non-uniform grid system and implicit in the associated NCAR graphics routines would not recognize the singular points. Procedures to circumvent these restrictions were devised. For plots of velocity vectors and streamlines, the circular domain was superimposed with a uniform grid (mesh width $< d$) and velocity values from the non-uniform grid system were interpolated onto the superimposed uniform grid points lying within the circle. Velocity interpolation at vortex locations was performed in the square domain, using Couet's filter, by first transforming vortex coordinates in the circular domain to equivalent positions in the square domain and then transferring grid velocity values from the circular domain to corresponding grid points in the square domain. The resulting off grid velocity values were then used to advect the vortices in the circular domain. The justification for doing this is implicit in the conformal transformation concept and the invariance of the complex potential and hence the streamfunction at corresponding transformable points. The validity of these schemes was tested by observing that the qualitative

features of the velocity vector and streamline plots and the locus of a discrete vortex after a complete revolution were correct and consistent with theory.

6.3.2 Point vortex initialization

The results from the vorticity initialization program described in Section 5.2 are presented in Table 5. Table 5(a) illustrates the velocity field initialization iteration data for $\tau_0=100$ ms. The optimum distribution of vortices satisfying the requirement of equation (5.14) was obtained at trial number 13. This illustration is representative of similar iterations performed for $100 \leq \tau_0 \leq 500$ ms. The initialization results for the different times are presented in Table 5(b). The mean value μ and the standard deviation σ of the strengths of the vortices are observed to decrease and increase, respectively, with time. Naturally, the swirling motion loses its intensity in the course of time if no external sustaining mechanism for the motion is in place.

The experimental data of Dyer (see Table 2) are plotted in Figure 15 at corresponding positions in the circular domain used in this study. Curves show the decay of velocity with time. Radial position is normalized by the chamber radius, $R = 1.0$ cm. The geometric distribution of the vortices for $\tau_0=100$ ms is plotted in Figure 16. The geometric distributions at other times are similar except for the difference in the number of vortices initialized. The initialization scheme successfully reproduced the indicated velocity profiles. This is not surprising since the scheme makes use of the prescribed velocity information to initialize the positions and strengths of the vortices. It should be noted that the iteration procedure is obviously open ended and

time consuming and that the number of vortices and geometric distribution for each τ_0 are not unique. The iterations were performed with a view toward minimizing the number of vortices since the computational work load is $O(N^2)$ for the exact solution scheme.

6.3.3 Velocity field and vortex trajectories

To estimate the accuracy of the numerical approximation for Equations (3.8)-(3.10) with assumed initial condition $\zeta(x,y,0)$ in D , computed velocity components (tangential and radial) and vortex trajectories in the circular domain are compared with exact results obtained via the modified circle theorem of Milne-Thomson. The velocity field induced by the system of 105 discrete vortices and the trajectories of vortex systems containing 1, 2, 13 and 37 discrete vortices are compared.

6.3.3.1 Velocity field

The system of 105 discrete vortices used to generate the velocity field in the circular domain were initialized in accordance with the initialization scheme discussed in Section 5.2. The relative errors in both the tangential and radial component of velocity are measured at each time step as

$$\text{Relative error (\%)} = \left[\frac{\sum_{i=1}^M (v_E - v_N)^2}{\sum_{i=1}^M (v_E)^2} \right]^{1/2} \times 100 \quad (6.4)$$

where $V_E = V_E[x(id,jd),t]$ is the exact velocity and $V_N = V_N[x(id,jd),t]$ is the numerical velocity at the computed grid points $x(id,jd)$ and time t . The errors are computed until $t=1.1$ seconds. Figure 17 shows the plot of the percent relative error versus time. Shortly after the onset of calculation, the error is observed to grow almost linearly with time for about 0.20 sec. Beyond $t=0.25$ sec., the error is observed to grow at a rate which is relatively independent of time, i.e. the method is essentially stable. Comparable errors in the component velocities noted earlier in the velocity transformation scheme are again realized. The velocity errors give reliable indication of the accuracy of the numerical prescription. In general, the results indicate that the method is convergent to within 20% relative error and essentially stable. In light of the susceptibility to errors of the various components involved in the numerical evaluation of the velocity field, this result seems acceptable. Further more vortex methods are generally low to medium accuracy methods. Plots of the tangential and radial velocity profiles at selected times corresponding to short and long time errors are illustrated in Figures 19 and 20 respectively.

6.3.3.2 Vortex trajectories

Figure 18 illustrates the trajectories for the systems of discrete vortices. In each system, the initial conditions for each vortex are determined by position, strength and sign of circulation. The initial coordinates for the single vortex is (0.6,0.0) (Figure 18a); for the 2-vortex systems, the vortex coordinates are (0.6,0.0) and (0.0,0.6) (Figure 18b and c) and for the 13- and 37-vortex systems, the vortices are uniformly initialized on concentric circles with a vortex placed in the center of the domain (Figure 18d, e, f and g). The single vortex is assigned a strength $\kappa=2\pi$ and

positive circulation. For each of the systems with more than one vortex, two cases with the same initial vortex positions were considered. In the first case (Figure 18b, d and f), the vortices were assigned the same strengths ($\kappa=2\pi$) and positive circulations. In the second case (Figure 18c, e and g), the vortices assumed different circulations and the same or different strengths. Specifically, the vortices in the 2-vortex system (Figure 18c), maintained the same strengths ($\kappa=2\pi$) but were assigned opposite circulations; counter clockwise for vortex 1 and clockwise for vortex 2. In the 13- and 37-vortex systems (Figure 18e and g), the vortices assumed randomly assigned strengths and signs of circulation.

The trajectories after one complete cycle are shown for the 1- and 2-vortex systems. The vortices in the 13- and 37-vortex systems were allowed to proceed for $1/4$ th and $1/8$ th of a cycle, respectively, enough to show a clear trend for comparison. A satisfactory agreement is observed in each case between the exact and numerical solutions validating the numerical procedure. Differences that are apparent are due to numerical errors resulting from interpolation procedures implicit in the numerical scheme and finite core effects implicit in the exact solution scheme.

The motions of the constituents in the 2-vortex systems are noteworthy. The mutually induced velocities of the two vortices gives the pair qualities reminiscent of a classical particle. The vortices with the same strength and same sign of circulation (Figure 18b) are beginning to wrap around one another. After several periods, it is expected that a single vortex will develop from the interaction of the two vortices. Rossow [62] introduced the term "convective merging" to describe this process which is available within the framework of a two-dimensional Euler equation, i.e., it does

not require any viscous effects. ⁹⁰ Convective merging survives as an important mechanism in vortex pairing [72] and it plays an essential role in our understanding of the dynamics of vortex structures in a shear layer. Additionally, the phenomenon of merging illustrates very clearly the generation of large scales from smaller scales (two small vortices fuse to one larger vortex), which in two-dimensional turbulence is responsible for the inverse energy cascade [43, 58]. In contrast, as the constituents in the pair of vortices with opposite sign and same strength (Figure 18c) come close together the vortices deviate from their original paths, pass one another and continue moving in opposite directions. The exact form of the paths followed depends on the relative strength of the vortices. This type of interaction between vortices of opposite sign thus never involves any pairing behavior of the vortices.

The realization that deterministic dynamic systems with very few degrees of freedom can display chaotic behavior has had a profound influence in our understanding of transition to turbulent flow. As the number of vortices is increased (Figure 18d, e, f and g), individual trajectories become of less concern and attention is shifted to collective modes exhibited by groups of many vortices. For simple initial conditions as described above, these collective modes take the form of clusters or regions of vorticity that take on an identity of their own and become the basic building blocks of the flow. We now have chaotic dynamics with many degrees of freedom or a turbulent flow situation (Figure 18g). But the surprise is that this regime does not correspond to a completely disorganized distribution of vorticity. Rather, we find flows with considerable structural features. As mentioned in the Introduction, the stimulus for studying the two-dimensional case has come from the experimental discoveries of so-called coherent vortex structures in certain laboratory and

geophysical shear flows.

6.4 Numerical Simulations

The Fortran computer code for this study has the capability to track the jet vortices from the near wall injection point through their evolution in the interior of the cylinder. The jet algorithm was intended not as a rigorous model of the real jet behavior but more to explore by a numerical method how a jet flow might be expected to interact with a swirling cross flow. Numerical experiments were conducted that systematically allowed different parameters to influence the mixing action. The parameters involved are the angle of injection θ_j , inlet jet velocity U_{in} , swirl strength S_θ ($\sim \Sigma \kappa_j$) and time t . The angle of injection θ_j is set counter clockwise from the positive x-axis. The inlet jet velocity U_{in} was determined by calculating, for each θ_j orientation, the distance per unit time step Δt covered by the vortex pair exiting the plane of the injector nozzle into the cylinder interior and just clearing the numerical shear layer thickness δ_s . To simulate different initial swirl conditions, vortices were initialized as shown on Table 5 to match Dyer's data (see Table 2) at the different times. The ultimate objective was to identify from the available data, the optimum conditions for minimum complete mixing time t^* defined as the time required for the jet vortices to "completely fill" the cylinder. The parameter t^* is therefore artificial. The criterion for "completely fill" was established by superimposing a square grid on the circular domain as shown in Figure 21 and ensuring that at least N_c vortices reside in each of the L square meshes or cells whose centers lie on the boundary or within the circular domain.

The effects of varying the grid size (and therefore the number of cells L), the number of vortices per cell N_c , the angle of injection θ_j , the swirl strength S_θ and the inlet jet velocity U_{in} on the mixing time t^* were first investigated to enable an appropriate choice of values for these parameters. Three tests were performed each with the inlet jet velocity U_{in} as a variable parameter. The first test sought to optimize L and θ_j for fixed values of N_c and S_θ . The second sought to optimize N_c for the chosen values of L , θ_j , S_θ . In the final test, the swirl strength S_θ was varied with L , θ_j and N_c fixed.

The considerations leading to the choice of the optimum grid size (or L) and angle of injection θ_j began with an investigation of three grid sizes 6×6 , 8×8 and 10×10 illustrated in Figure 21. For $N_c=2$, $S_\theta=12.5$ cm·m/s and L , θ_j and U_{in} as variable parameters, t^* data are recorded as shown in Table 6. These results are plotted in Figure 22.. As expected, t^* , is observed to increase with L for any combination of θ_j and U_{in} . When t^* is viewed from the standpoint of its dependence on U_{in} , results for $L=32$ and $L=80$ ($\theta_j=30^\circ$) seem relatively well behaved as illustrated in Figure 23. The case of $L=32$, $\theta_j=60^\circ$ best illustrates the expected exponentially decreasing function. However, for the cases $L=52$ and $L=80$ ($\theta_j=60^\circ$ and 90°) t^* appears essentially insensitive to variations in the injection speed. These observations suggest that the optimum grid size may have been exceeded at $L=52$. Given this relative independence on U_{in} , the average of the t^* values, t^*_{avg} , in each column of Table 6 is plotted in Figure 24 as a function of the number of cells L with the angle of

injection θ_j as a variable parameter. Although the lowest injection angle $\theta_j=30^\circ$ appears slightly advantageous, the advantage is not significant, especially in comparison with $\theta_j=60^\circ$. Given the observed trends, it would seem most appropriate to choose $L=32$ and $\theta_j=60^\circ$.

Having chosen the grid size 6×6 ($L=32$) and the angle of injection $\theta_j=60^\circ$, the optimum number of vortices required per cell was investigated next. With $L=32$, $\theta_j=60^\circ$, $S_\theta=12.5$ cm·m/s and N_c , U_{in} as variable parameters t^* data are recorded as shown on Table 7. The results are plotted in Figure 25. As with the case of $t^*(L)$, $t^*(N_c)$, in general, exhibited the expected trend; that is, t^* increases with increase in the number of vortices per cell. The notable exception is the case for $U_{in}=112.3$ cm/s which appears to be insensitive to N_c . In observing the trends for $t^*(U_{in})$, more clearly illustrated in Figure 26, it becomes evident that $N_c=2$ is the best candidate since t^* decreases with increase in U_{in} throughout the range of values for U_{in} in a much smoother fashion. The $t^*(U_{in})$ trends for $N_c=1$ and $N_c=3$ are not as smooth.

With the parameters L , N_c and θ_j established, the final test examined the effect of varying the swirl strength S_θ on the complete mixing time t^* , for a given inlet jet velocity U_{in} . The data recorded for $L=32$, $N_c=2$, and $\theta_j=60^\circ$ is shown on Table 8 and plotted in Figure 27. With the exception of the result for $U_{in}=125.7$ cm/s the general trend suggests that t^* is essentially independent of the swirl strength.

Perhaps the absence of viscosity may in part explain the indicated trend. It should be noted that randomness is inherently characteristic of vortex methods and that the observed outcome is a manifestation of the interactions and changing dynamics of the inviscid vortices in the flow domain. Bulk diffusion results from the interactions of the swirl and jet vortices. If the force field of a vortex or cluster of neighboring vortices modeling the swirling flow is stronger than that of a neighboring jet vortex or cluster, mixing is inhibited. The higher swirl strength will only aid the transport of the jet vortices to the cylinder wall due to the centrifugal force acting on the jet vortices. On the other hand, mixing is enhanced when a stronger jet overcomes the centrifugal force field and penetrates the flow. These contending phenomena are dictated by the relative strengths of neighboring vortices, and it is the net effect of these vortex interactions over time that results in the "completely filled" situation. It should be mentioned too that each swirl condition for these tests was obtained from a single realization of the vortex initialization scheme (see Table 5). Randomness would be reduced if each swirl condition is obtained by averaging over several possible realizations. That is, ensemble averaging would give a more realistic result. From the point of view of the variation of t^* with U_{in} , for fixed S_θ , the choice of $S_\theta=12.5$ cm·m/s and $U_{in}=125.7$ cm/s are preferred as can be deduced from Figure 28.

In summary, the arguments from these investigations point to the choice of the following parameters for the mixing simulation: 6 x 6 grid size ($L=32$ cells), $N_c=2$ vortices per cell, $\theta_j=60^\circ$, $S_\theta=12.5$ cm·m/s and $U_{in}=125.7$ cm/s.

To help illustrate the nature of the flow in the engine cylinder, the results of

the numerical experiments are presented for (i) swirling motion with no jet flow, (ii) jet flow with no swirling motion and (iii) swirling motion with jet flow. Streamline, velocity, vortex trajectory and vorticity results for incremental time steps of 10 ms up to when the cylinder wall is hit and including the time needed for jet vortices to fill the entire domain is filled with jet vortices are illustrated. The velocity and vorticity measurements serve to quantify the simulation process.

6.4.1 Swirling motion with no jet flow

Figure 29 shows the geometric distribution for the system of 115 discrete vortices and velocity vector representation of the swirling motion induced by these vortices as a function of time. Results are presented for selected times corresponding to short and long time errors illustrated in Figure 18. Because of interpolation problems associated with the corner singularities in the non-uniform grid system in the circular domain, the velocity vector field is displayed on a uniform 60 x 60 grid superimposed on the circular flow domain. Vector information at every other grid point is illustrated. The magnitude of the vector at each grid point denotes the relative speed of the flow. The swirling fluid motion is characterized by a much slower moving, small central core region and a nearly constant tangential velocity for the bulk of the faster moving fluid. At the initial time, a high degree of swirling air-flow symmetry is achieved. In the course of time, the center-of-rotation of the swirling flow is offset from the cylinder axis resulting in the oscillatory motions most apparent near the center of the cylinder. The precession of the off-set center-of-rotation is gradually dissipated by the bulk of the fluid, the dissipation rate stabilizing somewhat beyond $t=200$ ms. It was observed that at times less than $t=50$ ms, corresponding to the

relatively short, horizontal, lower left portion of Figure 18, the vortex distribution was still relatively organized. But at times greater than $t=50$ ms, the organized point distribution was lost as is evident from Figure 29. The larger relative error associated with the latter is essentially due to the critical closeness of the vortex elements, which allows the motion to be dominated by the effect of the singularity. Hald and Del Prete [39] suggest a larger smoothing parameter in order to ensure better convergence results in the disorganized state.

6.4.2 Jet flow with no swirling motion

Figures 30 and 31 show stages in the computed development of the jet vortices issuing at $U_{in}=210.0$ cm/s from an injector nozzle oriented at $\theta_j=60^\circ$ into a quiescent environment. It is observed from Figure 30 that the dominant structure in the first few diameters of the jet after the onset of injection consists of an array of vortex pairs. A non-linear region followed where instabilities give rise to small clumps, or eddies which interact with each other and evolve into larger eddies as they move across the cylinder toward the wall. The vorticity coalescence results in the eddies losing their identity. The occurrence and abruptness of the coalescence process depend on the irregularity in lateral spacing of the structures. A jet vortex first hits the wall at $t=29$ ms and its vorticity theoretically is destroyed when it is combined with its image vortex implying no flow through the wall. Algorithmically, this is simulated by bouncing back onto the wall vortices which translate beyond the boundary of the cylinder. In Figure 31, the streamlines begin to distort as the vortices coalesce and the "completely filled" situation at $t=271$ ms translates into the flow becoming more random, approaching a "turbulent" state. The closeness of

streamlines are indicative of a high fluid velocity in that region. The gradual evolution of the velocity and vorticity fields across the cylinder at $y=0$ is observed to change drastically as the jet spread gets close to the wall. The highest velocity and vorticity values are recorded when the cylinder is filled with vortices. Note that at the onset of injection, the radial velocity is comparable to the tangential velocity, but as the vortices get closer to the wall, the radial velocity begins to diminish.

6.4.3 Swirling motion with jet flow

A global view of the mixing process is provided by Figures 32 and 33. The jet nozzle is set at $\theta_j=60^\circ$. Jet fluid at $U_{in}=210.0$ cm/s issues into a swirling fluid of strength 12.5 cm·m/s. As the jet enters the cylinder, it is deflected by the swirling flow and moves with it. The core of the jet is not significantly deflected by the air swirl as the jet passes across the cylinder toward the wall, implying that the jet vortices are stronger at those locations and are able to penetrate the centrifugal force field of the swirling flow. Soon after the jet vortices interact with the swirling flow, the organized point distribution in the immediate vicinity of contact is lost. Because of the absence of turbulent motion and molecular viscosity, the distribution of jet vortices (i.e., mixing via bulk diffusion) is determined primarily by the air swirl. The streamline patterns, illustrated in Figure 33, provide a qualitative appreciation of the mixing process. The regions of great distortions of the streamline patterns are indicative of intense vortex interactions. The radial motion of the vortex particles is observed to be a small fraction of the tangential velocity due to centrifugal forces acting on them. This accounts for the jet vortices reaching the vicinity of the cylinder

wall earlier ($t=10$ ms) than for the no swirl case ($t=20$ ms). Note that the fact that a jet vortex first hits the wall ($t=29$ ms) for the no swirl case than for the swirl case ($t=32$ ms) in no way diminishes the transport of jet fluid to the wall by the swirling flow. The algorithm was set up to identify the time when a vortex first hits the wall and not when the bulk of the fluid first hits the wall. The fact that swirling flow does indeed aid the mixing process is seen in the faster complete mixing time ($t^*=198$ ms) registered by this situation than for the no swirl case ($t^*=271$ ms).

The above results are representative of results for $\theta_j=90^\circ$ and $\theta_j=120^\circ$ presented in Figures 34 through 41. At $\theta_j=120^\circ$ the injector is opposing the swirling flow. Some symmetry is observed between the results for $\theta_j=60^\circ$ and $\theta_j=120^\circ$ at corresponding times.

CHAPTER 7

SUMMARY AND CONCLUSION

A major challenge in engine research is to produce an engine that runs efficiently and smoothly on lean mixtures. To do this an engine must have the ability to burn very lean mixtures quickly and completely. Cylinder charge motion is thought to be the most practical means of reaching this goal. To use the mixture motion effectively, it is desirable that the lean-burn engine designer be aware of how the flame behaves in turbulence and swirl. The effectiveness of turbulence in reducing combustion times has received and is still receiving attention. But there appears to be little information available on how a flame behaves under the effect of swirl.

A flexible research program has been initiated in this study to address the mixing effect of swirl on an inert scalar field. Mixing descriptions based on the concepts of discrete vortex dynamics are restricted to a two-dimensional, inviscid, Lagrangian formulation. The general structure and flexibility of the numerical schemes employed in this study make the inclusion of viscosity, turbulence and chemical kinetics of the combustion process a viable future objective.

In meeting the present goal, several topics were explored. An empirical scheme was developed to determine the vortex cut-off radius, the associated controversy in the literature notwithstanding. A foundation was laid for a technique to initialize the vortex element distribution from prescribed velocity data. Useful insights have been provided on conformal transformation techniques and real space application of numerical filters.

Although the model provided herein is an idealized representation of a real complex problem, the results and physical insights gained from this study can be extended qualitatively to actual engines. Comparison with exact solutions shows the validity of this method of approximation. Streamfunction values compare to within 6% for values induced by a single vortex and 0.3% for values induced by a system of 105 vortices. An estimate of the uncertainty in a given velocity measurement is about 20%. This seems an acceptable result considering the many variables involved in the velocity computation. The predictions of theory have been validated to the extent that convergence would occur as the mesh size d and time step Δt tend to zero. The method is relatively stable and d and Δt are constrained only by the accuracy requirement of the Courant condition. The swirling air motion and the features of a real jet are well simulated, at least qualitatively, by the motion of a system of discrete point vortices, adding confidence to the numerical description. Some general features of mixing in swirling flow have emerged from this work :

- (i) Swirl enhances the transport process.
- (ii) The strengths of the jet vortices scale with flow velocities at the nozzle valve.

- (iii) The stronger the jet kinetic energy, the greater the centripetal movement of fuel particles. This implies greater mixing. It can be inferred from this that the combustion time will be shorter and the leaner the mixture, the more obvious this trend.
- (iv) There is an optimal swirl-rate for minimum mixing time, a conclusion which could be of significance in engine design.

The advantages gained from this study are derived from the use of an Eulerian and Lagrangian formulation. The Eulerian grid in the VIC method allows the application in real space of an efficient numerical filter capable of subgrid resolution and leaving invariant the total circulation within each cell. The method generates a solution independent of the core radius of the discrete vortex elements thereby avoiding the singularity inherent in the Green's function method. The Lagrangian treatment eliminates the need to explicitly treat convective derivatives. Unlike other methods, the discrete vortex methods are not limited at high Reynolds number of practical significance by the difficulty in distinguishing real from numerical diffusion. The number of mesh points as well as the amount of computational labor required to obtain a solution would be prohibitive, in particular for the finite element and finite difference methods which can handle only a bounded range of frequencies. The VIC technique offers a resolution of the problem of excessive computation time. The fast Fourier transform allows for efficient solution of the Poisson's equation for the stream function in every time step of the simulation. Computational economy is achieved because only $O(N)$ computer memory is required per time step, where N is the number of vortices, and the number of operations per time step in the FFT increases

linearly with the number of vortex elements and mesh points. Additional benefits include an increased understanding of vortex interactions at a level of sophistication consistent with the present needs of the design engineer.

Vortex methods are limited, however, like other methods, by the fact that effects not resolved cannot be seen, i.e., if there are not enough computational elements to represent a phenomenon, that phenomenon will not be observed. The dependence of the method on an assumed non-deformable structure of the vortex makes analysis difficult since in reality vortices of finite size deform. Misrepresentation of the interaction of neighboring vortices can lead to a significant statistical error in any one realization of the solution. This obviously is the price that must be paid for the removal of numerically induced viscosity. In the VIC method, the vortex core cutoff is replaced, in effect, by the cell size which has an effect on the computations, e.g., the finite mode Fourier transform error, also known as aliasing error. It is necessary to eliminate such errors generated by the numerical procedure in order to obtain a smooth solution. Finally, proper application requires a mathematical effort which is relatively substantial.

It should be noted that mixing experiments remain one of the most important tools in combustion research since chemical reactions can only occur when the reactants have been mixed to the appropriate composition. When the airflow process in an engine cylinder is generally understood and we can model the non-combustion mixing flow effectively, we will have a much better model of the combustion flow and thus equip the engine designer with the necessary facts for predicting design trends.

CHAPTER 8

RECOMMENDATIONS

Vortex methods have certainly not reached the reliability of, say, finite element methods for elliptic problems. They are useful and important because they apply to problems where other methods are not necessarily applicable. It is hoped that the further development of vortex methods will lead to ever increasing reliability. The brunt of future work should be focused on the fluid mechanics which is an integral and essential aspect of the mixing problem.

The distribution of the computational vortex elements in the vorticity initialization technique will usually be dictated by the geometry of the system as well as accuracy considerations of the integration formula implicit in the scheme. It is recommended that a more systematic approach employing an appropriately derived integration formula for the initialization procedure be investigated instead of the use of the arbitrary, non-unique distribution employed in this study. For circular domains, it is expected that a radially symmetric distribution, with the distribution weighted more heavily toward the center, will produce smaller errors than a distribution on a rectangular grid. The imaging system which makes use of

experimental information to produce the desired vorticity and velocity fields provides a good foundation upon which to build. Averaging over all possible realizations of the initial conditions and velocity fields would lessen the characteristic random outcome of a vortex method and result in a smoother solution.

The extent to which chaotic motion is damped by finite core effects is an important issue that should be addressed more thoroughly in the future especially considering the controversy in the literature concerning the core radius. Boundary effects (neglected in this study) should also be included in the optimization scheme for the cut-off radius.

Future studies should investigate the inter-dependence of mesh size and jet nozzle diameter on the resolution of the features of the jet flow. The jet nozzle diameter was chosen to be equal to the mesh size in this study.

It is recommended that alternative transformation schemes, free of singularities be chosen. The exact form of the transformation is not important. However, it is essential that the transformation function be smooth. Some computing centers now offer programs for the symbolic manipulations of expressions and functions. Such programs could be of tremendous help in the construction of the system of equations used for the grid transformation. This should eliminate the rather lengthy and tedious algebra employed in this study.

No particular effort has been directed toward computational efficiency in this study since the emphasis of this study has been the development of a model rather than a computer program. Future studies could focus some attention on computer time usage.

It is believed that if these improvements are made, a versatile discrete vortex approximation scheme would emerge that would provide a firm and promising foundation for a more accurate two-dimensional inviscid flow simulation and ultimately a more reliable framework for the incorporation of viscous effects and combustion chemistry.

Table 1. One-dimensional filter models.

Author(s)	Scheme	Filter	Number of grid points involved in two dimensions.	CPU time (ms.) for computing in the interval $0.0 < \eta < 1.0$	CPU time (ms.) for spreading vorticity to grid points.
Bartholomew	Polynomial	$\varphi(\eta) = \begin{cases} A\eta^8 + B\eta^6 + C\eta^4 + D\eta^2 + E & ; \quad \eta \leq \frac{3}{2} \\ 0 & ; \quad \text{Otherwise} \end{cases}$	9	0.105	1.779
Christiansen and Baker	Area Weighting	$\varphi(\eta) = \begin{cases} 1 - \eta & ; \quad \eta \leq 1.0 \\ 0 & ; \quad \text{Otherwise} \end{cases}$	4	0.066	0.660
Couet	Quadratic Spline Interpolation	$\varphi(\eta) = \begin{cases} \frac{1}{2} \left[\frac{3}{2} - \eta ^2 \right] & ; \quad \frac{1}{2} \leq \eta \leq \frac{3}{2} \\ \frac{3}{4} - \eta^2 & ; \quad \eta \leq \frac{1}{2} \\ 0 & ; \quad \text{Otherwise.} \end{cases}$	9	0.081	1.519
Hockney	Truncated Gaussian	$\varphi(\eta) = \begin{cases} \frac{1}{2\text{erf}(\alpha)} \left[\text{erf}(\alpha) - \text{erf} \left(\alpha \left(\eta - \frac{1}{2} \right) \right) \right] & ; \quad \frac{1}{2} \leq \eta \leq \frac{3}{2} \\ \frac{1}{2\text{erf}(\alpha)} \left[\text{erf} \left(\alpha \left(\frac{1}{2} - \eta \right) \right) + \text{erf} \left(\alpha \left(\frac{1}{2} - \eta \right) \right) \right] & ; \quad \eta \leq \frac{1}{2} \\ 0 & ; \quad \text{Otherwise} \end{cases}$	9	0.535	6.574
Meng and Thomson	Modified Area Weighting	$\varphi(\eta) = \begin{cases} \frac{1}{2} \left[1 + \frac{\text{Tan}^{-1} \left[3 \left(\frac{1}{2} - \eta \right) \right]}{\text{Tan}^{-1} \left(\frac{3}{2} \right)} \right] & ; \quad \eta \leq 1.0 \\ 0 & ; \quad \text{Otherwise} \end{cases}$	4	0.194	1.350
Wang	Cubic Spline Interpolation	$\varphi(\eta) = \begin{cases} \frac{2}{3} - \frac{1}{2} \eta^2 (2 - \eta) & ; \quad \frac{1}{2} \leq \eta \leq \frac{3}{2} \\ \frac{1}{6} - \frac{1}{2} \left[\frac{1}{2} - \eta \right]^3 & ; \quad 1.0 \leq \eta \leq 2.0 \\ 0 & ; \quad \text{Otherwise} \end{cases}$	16	0.093	3.987

Table 2. Dyer's experimental data.

$\overline{V}_\theta$ = Mean tangential velocity, (m/s)

r = Radial distance (cm)

R = Chamber radius (cm)

τ_0 = Time from the closing of the shrouded intake valve to time of measurement.

$\overline{V}_\theta$ (m/s)					
r/R	τ_0 (ms)				
	100	150	200	300	500
0.125	12.00	8.60	7.30	5.30	3.50
0.200	24.50	14.70	12.60	10.00	4.90
0.300	33.10	21.60	16.60	11.40	5.50
0.400	37.20	25.10	17.00	10.30	5.20
0.500	39.00	23.90	16.30	9.60	4.90
0.600	37.60	22.20	15.10	9.00	4.70
0.700	33.70	20.40	14.00	8.40	4.50
0.800	30.00	18.80	13.10	7.80	4.30
0.900	28.20	17.35	12.40	7.60	4.30
0.950	27.80	17.00	12.10	7.50	4.30
Average (m/s)	27.80	17.50	12.50	7.90	4.10

Table 3. Comparison of the absolute value of the relative error (%) estimates between the modified exact and numerical tangential velocity values at grid points.

(Mesh size = $d \times d$, $d = 0.0625$ cm., Grid size = 60×60 , Test grid size = 50×50 , $K = 1.0$)

POINT VORTEX AT GRID POINT

Cutoff Value	For All Grid Points		For All Grid Points > 2.0d	
	Minimum	Maximum	Minimum	Maximum
...	...	...	...	...
0.030	0.000000E+00	3.3103230E+01	0.000000E+00	8.2269478E+00
0.031	0.000000E+00	3.3123978E+01	0.000000E+00	8.2309284E+00
0.032	0.000000E+00	3.3041328E+01	0.000000E+00	8.2384529E+00
0.033	0.000000E+00	3.2821266E+01	0.000000E+00	8.2486515E+00
0.034	0.000000E+00	3.2506222E+01	0.000000E+00	8.2618771E+00
0.035	0.000000E+00	3.2149502E+01	0.000000E+00	8.2778234E+00
0.036	0.000000E+00	3.1954117E+01	0.000000E+00	8.2969713E+00
0.037	0.000000E+00	3.2173676E+01	0.000000E+00	8.3195601E+00
0.038	0.000000E+00	3.2553860E+01	0.000000E+00	8.3456974E+00
0.039	0.000000E+00	3.3207455E+01	0.000000E+00	8.3754749E+00
0.040	0.000000E+00	3.3617626E+01	0.000000E+00	8.4087601E+00
0.041	0.000000E+00	3.4297836E+01	0.000000E+00	8.4481955E+00
0.042	0.000000E+00	3.5085903E+01	0.000000E+00	8.4927826E+00
...	...	...	...	...
...	...	...	...	...

POINT VORTEX AT CELL CENTER

Cutoff Value	For All Grid Points		For All Grid Points > 1.5d	
	Minimum	Maximum	Minimum	Maximum
...	...	...	...	...
0.042	0.000000E+00	2.8972696E+01	0.000000E+00	4.5753188E+00
0.043	0.000000E+00	2.7691328E+01	0.000000E+00	4.5920501E+00
0.044	0.000000E+00	2.6544624E+01	0.000000E+00	4.6130338E+00
0.045	0.000000E+00	2.5605698E+01	0.000000E+00	4.6367159E+00
0.046	0.000000E+00	2.4890265E+01	0.000000E+00	4.6635318E+00
0.047	0.000000E+00	2.4483385E+01	0.000000E+00	4.6949511E+00
0.048	0.000000E+00	2.4455280E+01	0.000000E+00	4.7295132E+00
0.049	0.000000E+00	2.5077896E+01	0.000000E+00	4.7691174E+00
0.050	0.000000E+00	2.6744957E+01	0.000000E+00	4.8136029E+00
0.051	0.000000E+00	2.8853853E+01	0.000000E+00	4.8622284E+00
0.052	0.000000E+00	3.1229172E+01	0.000000E+00	4.9168506E+00
0.053	0.000000E+00	3.4065449E+01	0.000000E+00	4.9761343E+00
0.054	0.000000E+00	3.8109451E+01	0.000000E+00	5.0421910E+00
...	...	...	...	...
...	...	...	...	...

Table 4. Error statistics for grid point coordinate and velocity transformations.

	Coordinate Transformation		Velocity Transformation	
	ζ	η	u	v
Avg. absolute error	2.0397014E-07	1.4282386E-07	1.5206969E-01	1.4264177E-01
Max. absolute error	1.3470206E-05	6.3128371E-06	3.4926014E+00	4.0869408E+00
Max. relative. error (%)	1.9993558E-02	1.0198522E-02	4.0998020E+07	9.2584425E+05
Root mean square error	6.8885259E-07	3.5780748E-07	3.3185267E-01	3.0976245E-01
Grid point location of max. absolute error	(2,1), (2,61), (60,1), (60,61)	(1,3), (1,59), (61,59), (61,3)	(2,1), (2,61), (60,1), (60,61)	(1,3), (1,59), (61,59), (61,3)

Note :

$$\text{Average absolute error} = \frac{\sum_{i=1}^M \sum_{j=1}^M |p_E - p_N|}{M^2} \quad (6.6)$$

$$\text{Relative error (\%)} = \left[\frac{\sum_{i=1}^M \sum_{j=1}^M (p_E - p_N)^2}{\sum_{i=1}^M \sum_{j=1}^M p_E^2} \right]^{1/2} \times 100 \quad (6.7)$$

$$\text{Root mean square error} = \left[\sum_{i=1}^M \sum_{j=1}^M (p_E - p_N)^2 / M^2 \right]^{1/2} \quad (6.8)$$

where p is the parameter of interest.

Table 5(a). Velocity field initialization iteration data ($\tau_0 = 100$ ms).

Trial No.	r/R								$(\Sigma \kappa_j)_{\text{desired}} = R\bar{V}_\theta = 27.80 \text{ cm. m/s}$					$\mu = \frac{\Sigma \kappa_j}{N}$	
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	$\bar{V}_\theta = 27.80 \text{ m/s}$					$\sigma^2 = \left(\frac{\Sigma \kappa_j ^2}{N} \right) - \mu^2$	
	Point Vortex Distribution								N	$\Sigma \kappa_j$	$ \kappa_j _{\min}$	$ \kappa_j _{\max}$	μ	$\frac{\sigma}{\mu}$	
1	6	10	10	12	24	32	24	10	128	26.07	0.09	0.72	0.20	0.71	
2	6	10	10	12	24	32	26	10	130	26.18	0.10	0.72	0.20	0.72	
3	6	10	10	12	24	32	22	10	126	25.96	0.09	0.73	0.21	0.71	
4	6	10	10	12	24	32	32	10	136	26.50	0.10	0.72	0.19	0.74	
5	4	6	6	7	28	32	28	8	119	28.72	0.11	1.03	0.24	0.77	
6	4	6	6	7	28	32	30	8	121	28.86	0.11	1.03	0.24	0.77	
7	4	6	6	7	28	32	26	8	117	28.59	0.11	1.03	0.24	0.76	
8	4	6	6	7	28	32	20	8	111	28.17	0.11	1.04	0.25	0.75	
9	4	6	6	7	28	32	14	8	105	27.75	0.10	1.05	0.26	0.73	
10	4	6	6	7	28	32	12	8	103	27.60	0.10	1.05	0.27	0.72	
11	4	6	6	7	28	32	16	8	107	27.89	0.10	1.05	0.26	0.73	
12	4	6	6	7	28	32	15	8	106	27.82	0.10	1.05	0.26	0.73	
13	4	6	6	6	28	32	15	8	105	27.80	0.10	1.06	0.26	0.73	

Table 5(b). Velocity field initialization data (100 ms < τ_0 < 500 ms).

Time (ms.)	r/R								$(\Sigma \kappa_j)_{\text{desired}}$					$\mu = \frac{\Sigma \kappa_j}{N}$	
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	$\left\{ \begin{array}{l} @ 100 \text{ ms.} = 27.80 \text{ cm.m/s} \\ @ 150 \text{ ms.} = 17.50 \text{ cm.m/s} \\ @ 200 \text{ ms.} = 12.50 \text{ cm.m/s} \\ @ 300 \text{ ms.} = 7.90 \text{ cm.m/s} \\ @ 500 \text{ ms.} = 4.10 \text{ cm.m/s} \end{array} \right.$					$\frac{\sigma}{\mu}$	
	Point Vortex Distribution								N	$\Sigma \kappa_j$	$ \kappa_j _{\min}$	$ \kappa_j _{\max}$	μ	$\frac{\sigma}{\mu}$	
100	4	6	6	6	28	32	15	8	105	27.80	0.10	1.06	0.26	0.73	
150	4	6	6	8	31	32	15	8	110	17.50	0.02	0.55	0.16	0.73	
200	4	6	6	13	31	32	15	8	115	12.50	0.05	0.43	0.11	0.74	
300	4	6	6	16	34	34	18	8	126	7.90	0.03	0.34	0.06	0.84	
500	4	6	6	6	26	30	16	8	102	4.10	0.03	0.22	0.04	0.86	

Table 6. L , θ_j optimization data (Swirl strength $S_\theta = 12.5 \text{ cm}\cdot\text{m/s}$).

$N_c = 2$	$\theta_j = 30^\circ$		
U_{in} (cm/s)	Grid Size		
	6 x 6 ($L=32$)	8 x 8 ($L=52$)	10 x 10 ($L=80$)
	t^* (ms)		
112.3	173	185	344
125.7	149	229	327
152.6	112	213	319
210.0	124	184	250

$N_c = 2$	$\theta_j = 60^\circ$		
U_{in} (cm/s)	Grid Size		
	6 x 6 ($L=32$)	8 x 8 ($L=52$)	10 x 10 ($L=80$)
	t^* (ms)		
112.3	185	207	343
125.7	159	198	307
152.6	130	213	304
210.0	99	198	343

$N_c = 2$	$\theta_j = 90^\circ$		
U_{in} (cm/s)	Grid Size		
	6 x 6 ($L=32$)	8 x 8 ($L=52$)	10 x 10 ($L=80$)
	t^* (ms)		
112.3	215	267	356
125.7	191	236	371
152.6	138	268	373
210.0	108	167	330

Table 7. N_c optimization data (Swirl strength $S_\theta = 12.5 \text{ cm}\cdot\text{m/s}$).

$L = 32$	$\theta_j = 60^\circ$		
U_{in} (cm/s)	N_c		
	1	2	3
	t^* (ms)		
112.3	179	185	185
125.7	128	159	173
152.6	113	130	167
210.0	82	99	152

Table 8. Optimization data for swirl strength S_θ .

$L = 32$ $N_c = 2$	$\theta_j = 60^\circ$				
U_{in} (cm/s)	Swirl strength S_θ (cm·m/s)				
	4.1	7.9	12.5	17.5	27.8
	t^* (ms)				
112.3	175	178	185	152	146
125.7	210	185	159	150	128
152.6	141	164	130	123	149
210.0	116	88	99	100	118

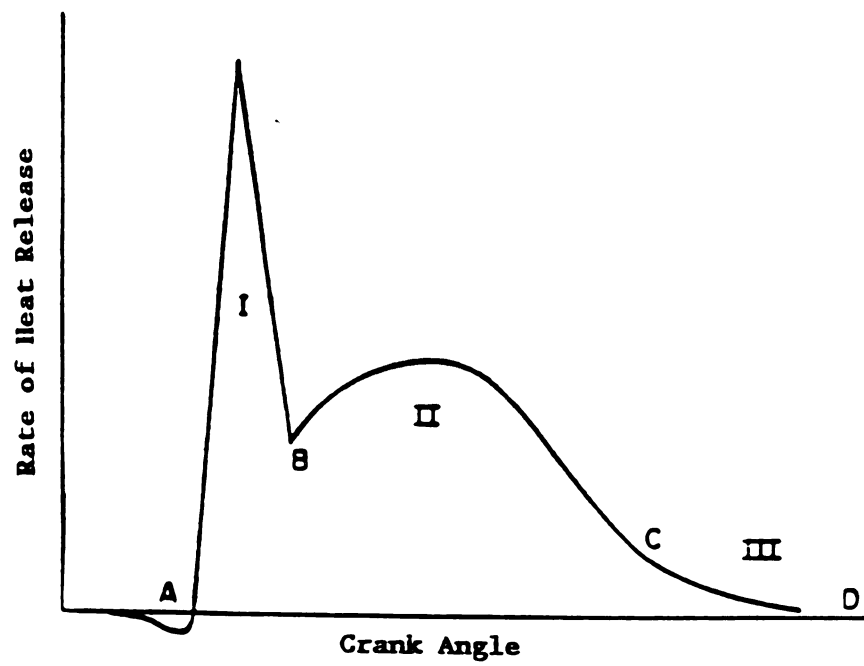


Figure 1. Three phases of heat release.

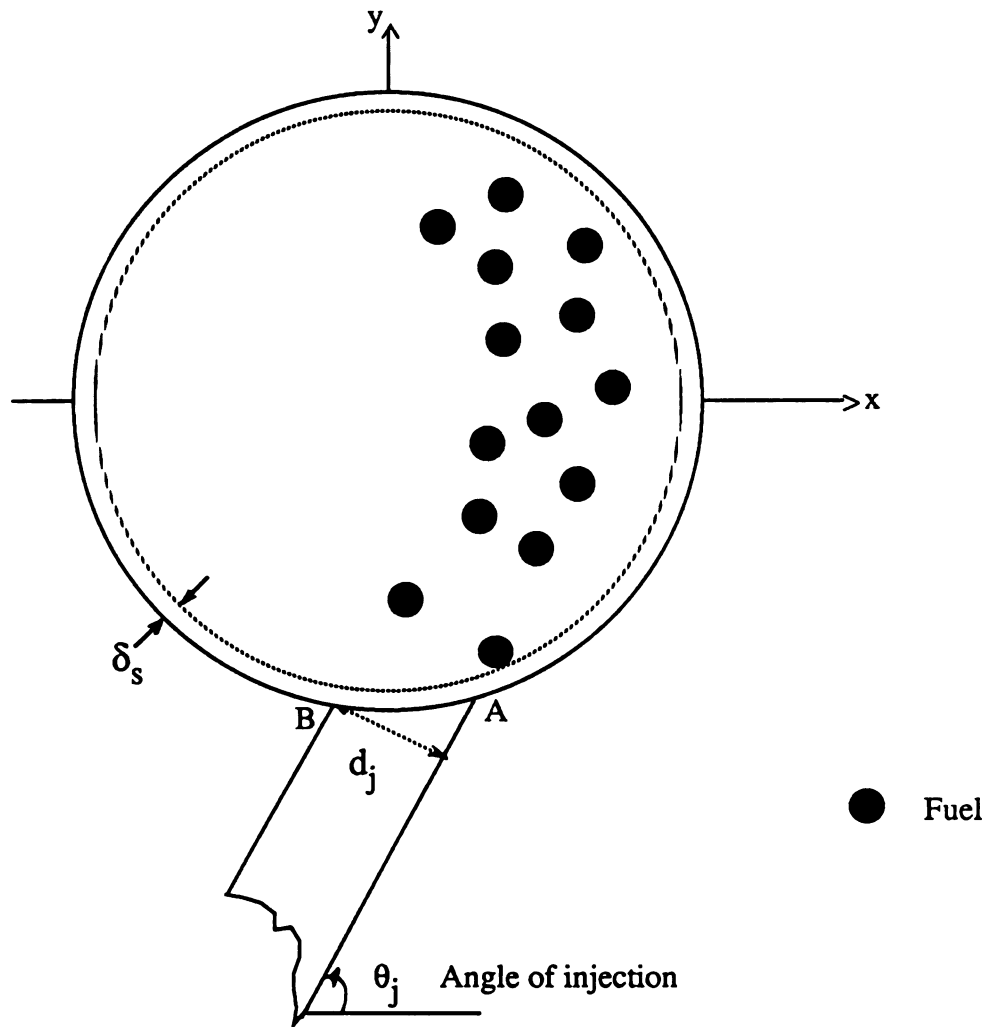


Figure 2. Sketch of intake flow

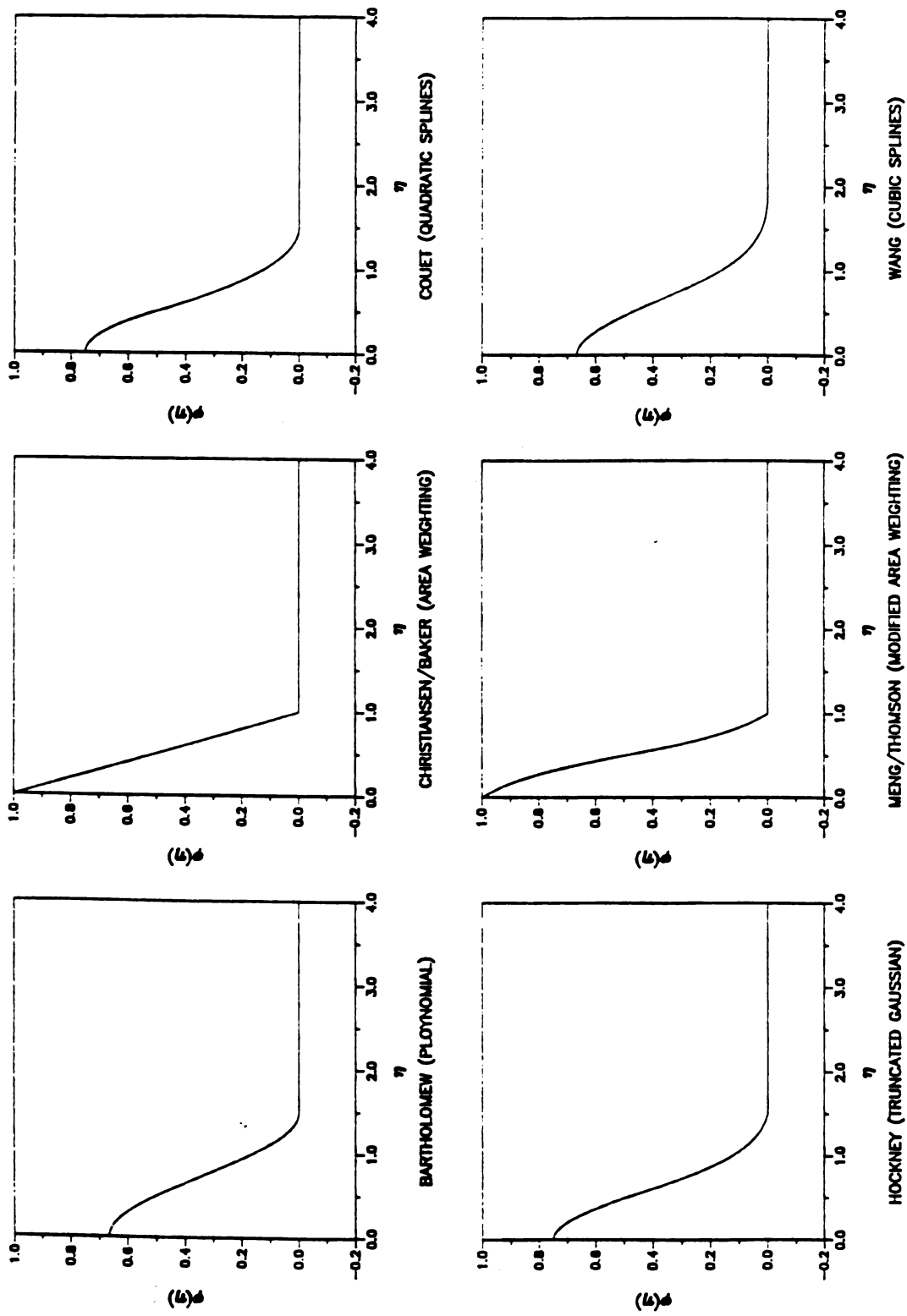


Figure 3. Numerical filters for vorticity deposition on the grid and for interpolation of local effects of the velocity field from the grid to the Lagrangian points.

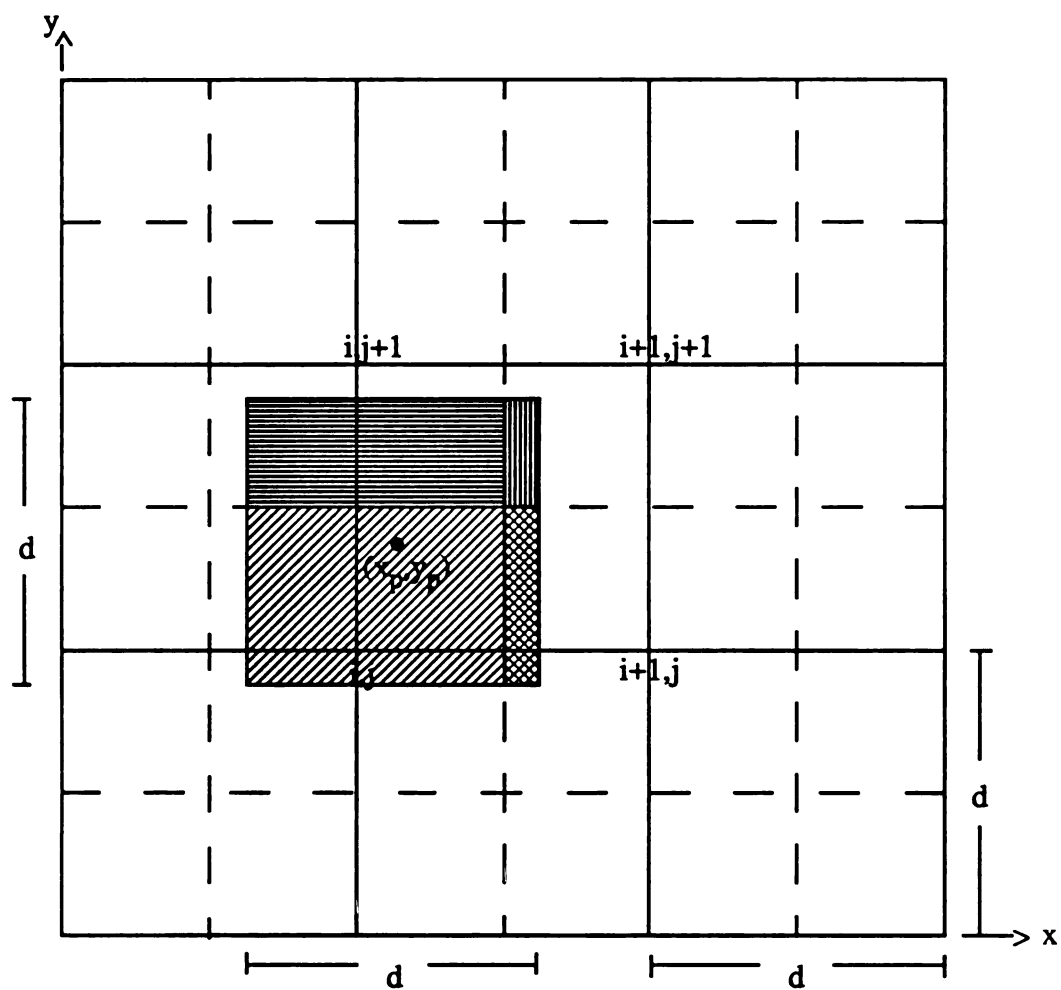


Figure 4. Vorticity distribution in a grid, with shading showing assignment of vorticity to grid point for VIC method.

-  is assigned to grid point (i,j)
-  is assigned to grid point $(i+1,j)$
-  is assigned to grid point $(i+1,j+1)$
-  is assigned to grid point $(i,j+1)$

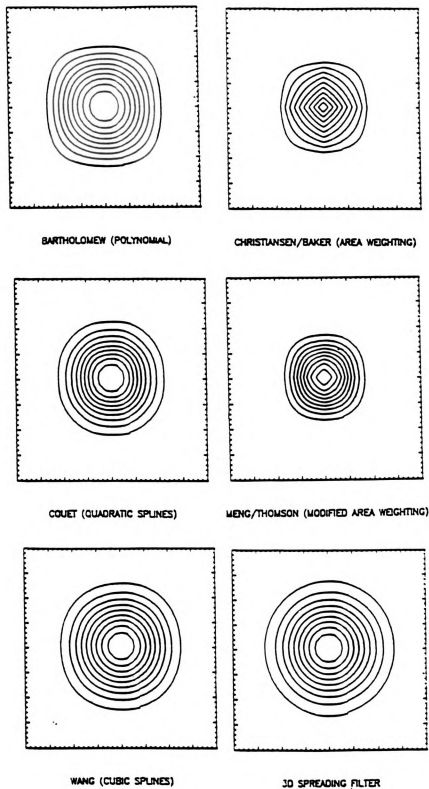


Figure 5. Iso-vorticity contours as represented by the various filters. Large tick marks represent a half-grid length.

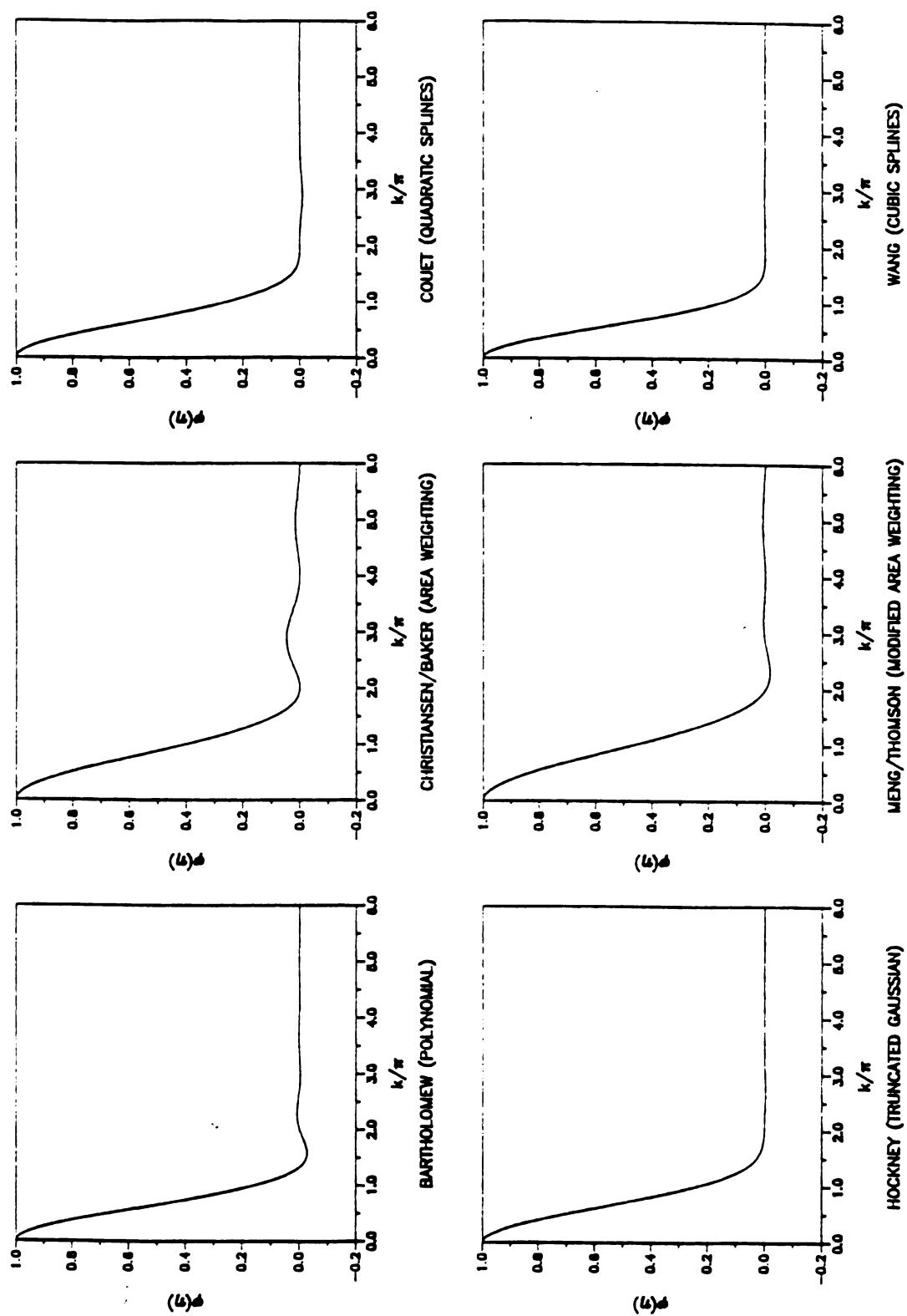


Figure 6. Fourier transform of the various filters.

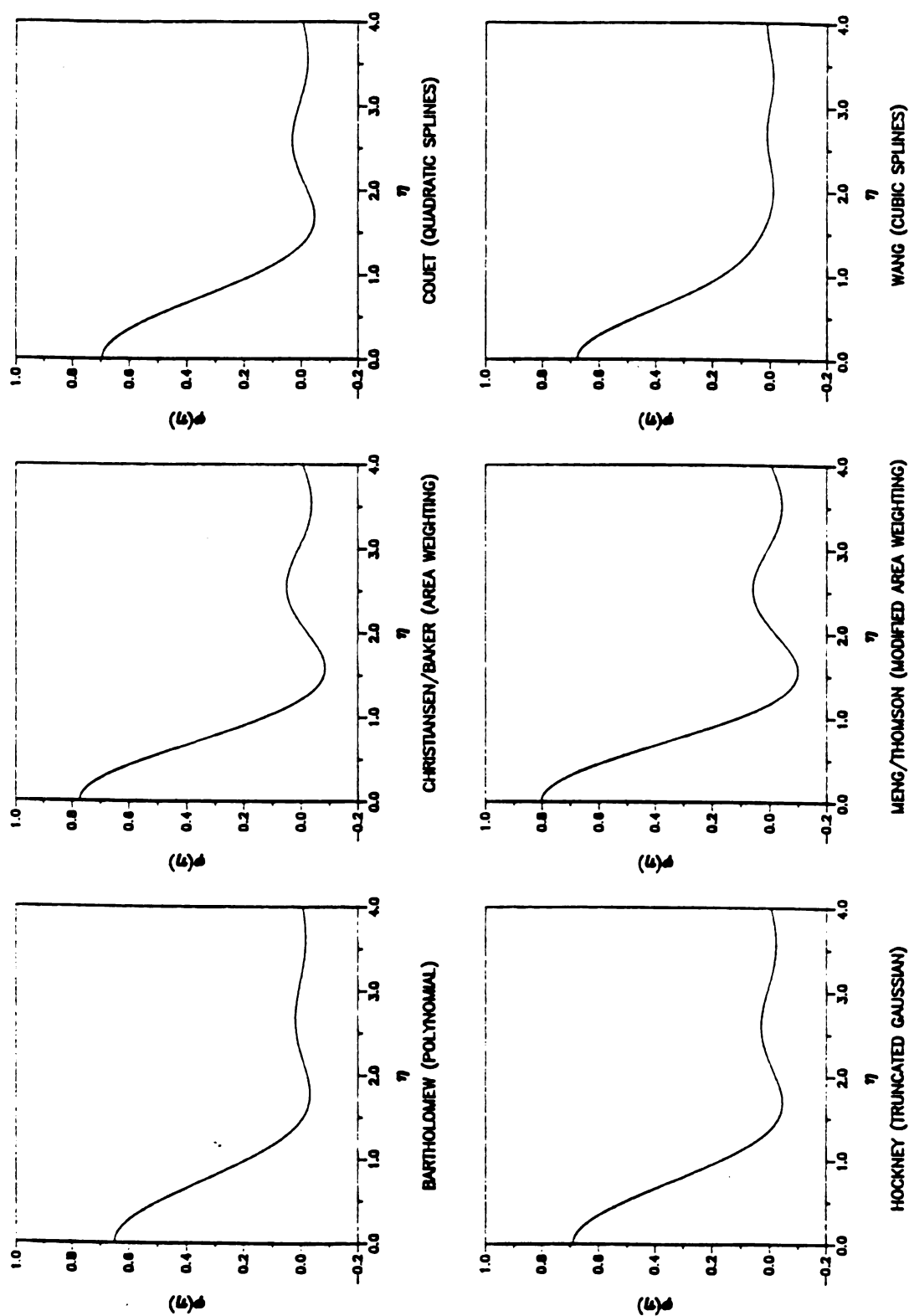


Figure 7. Inverse Fourier transform of the various filters.

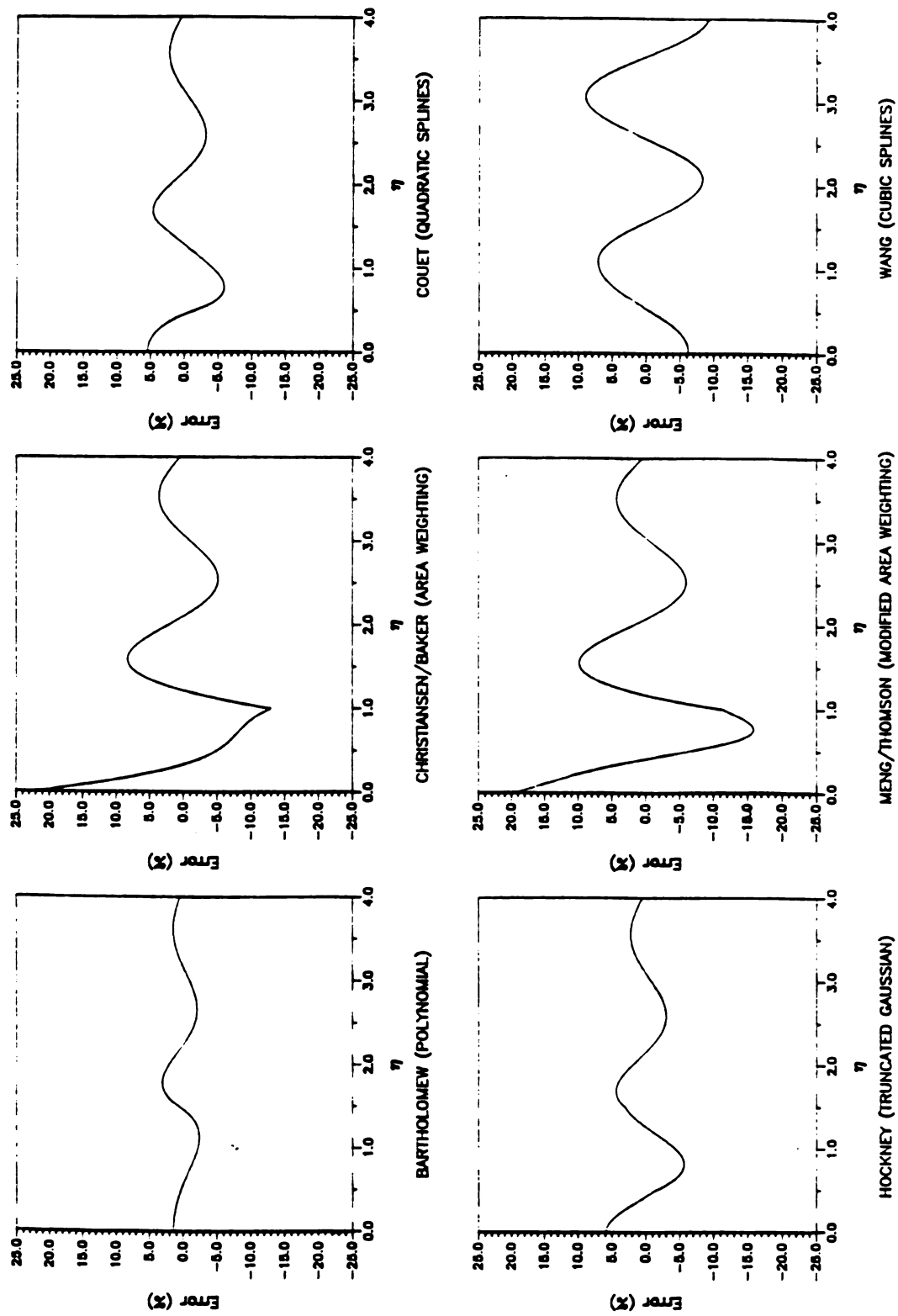
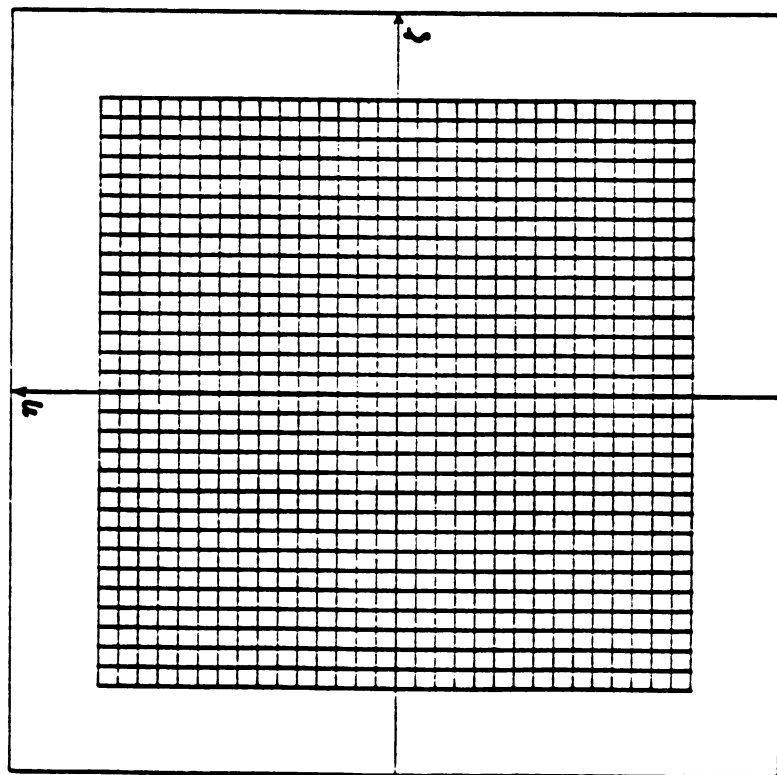
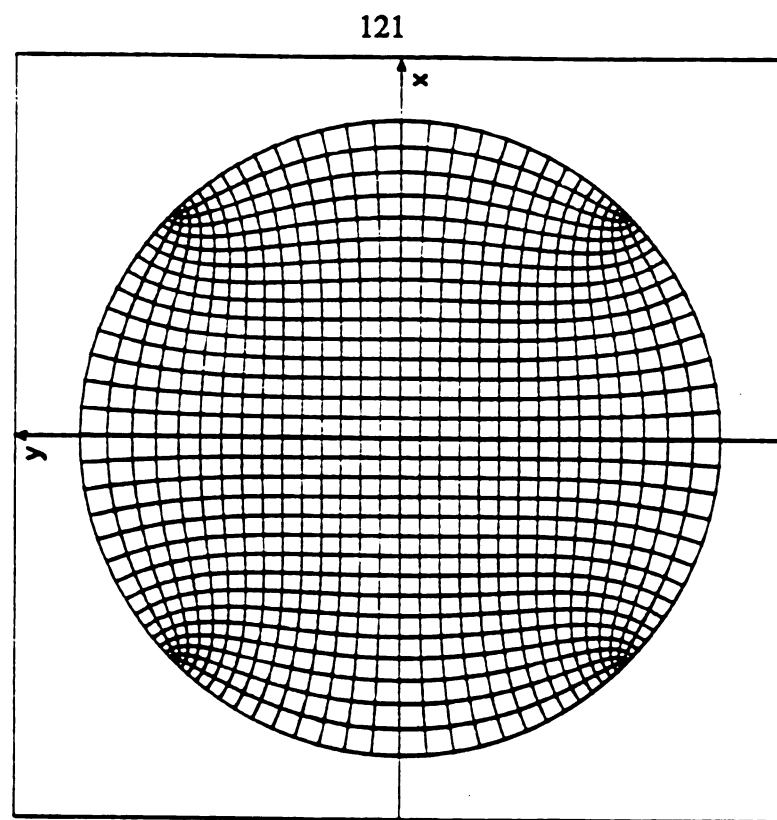


Figure 8. Error estimates between original filter and the inverse of its transform.



ω -plane

$$z = \frac{\operatorname{sn}(\omega) \operatorname{dn}(\omega)}{\operatorname{cn}(\omega)}$$



z -plane

Figure 9. Grid transformation.

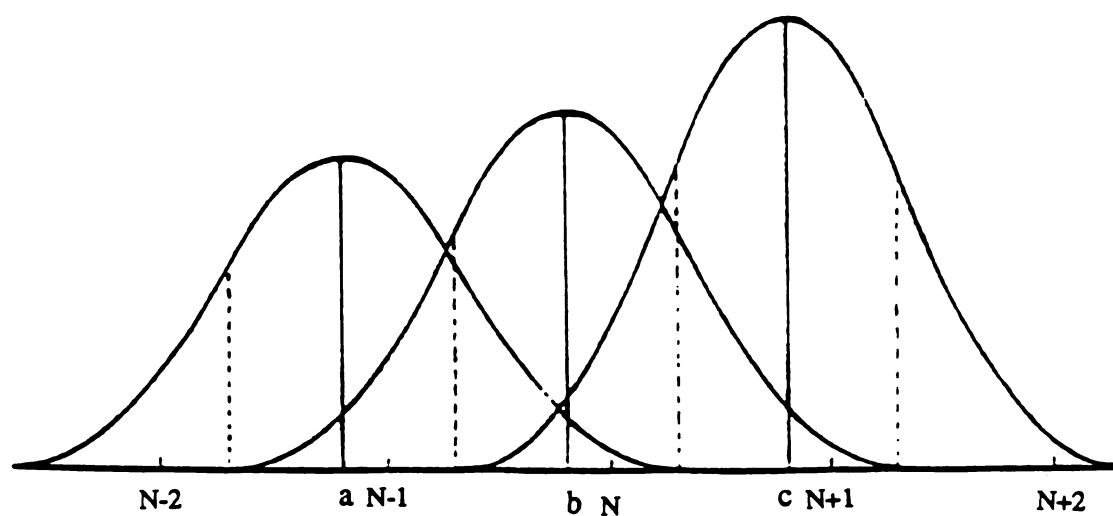


Figure 10. Quadratic spline vorticity distribution of three typical vortices on their nearby grid points.

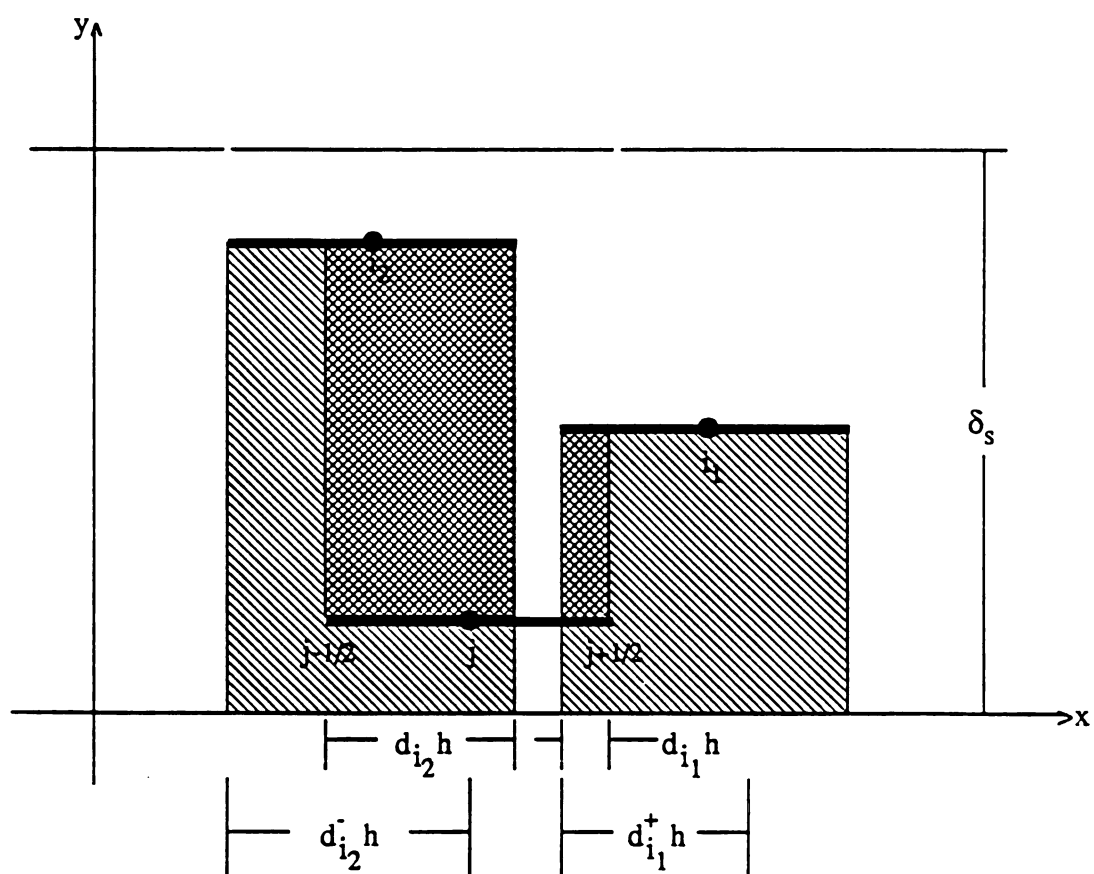


Figure 11. Zone of influence of a vortex sheet.

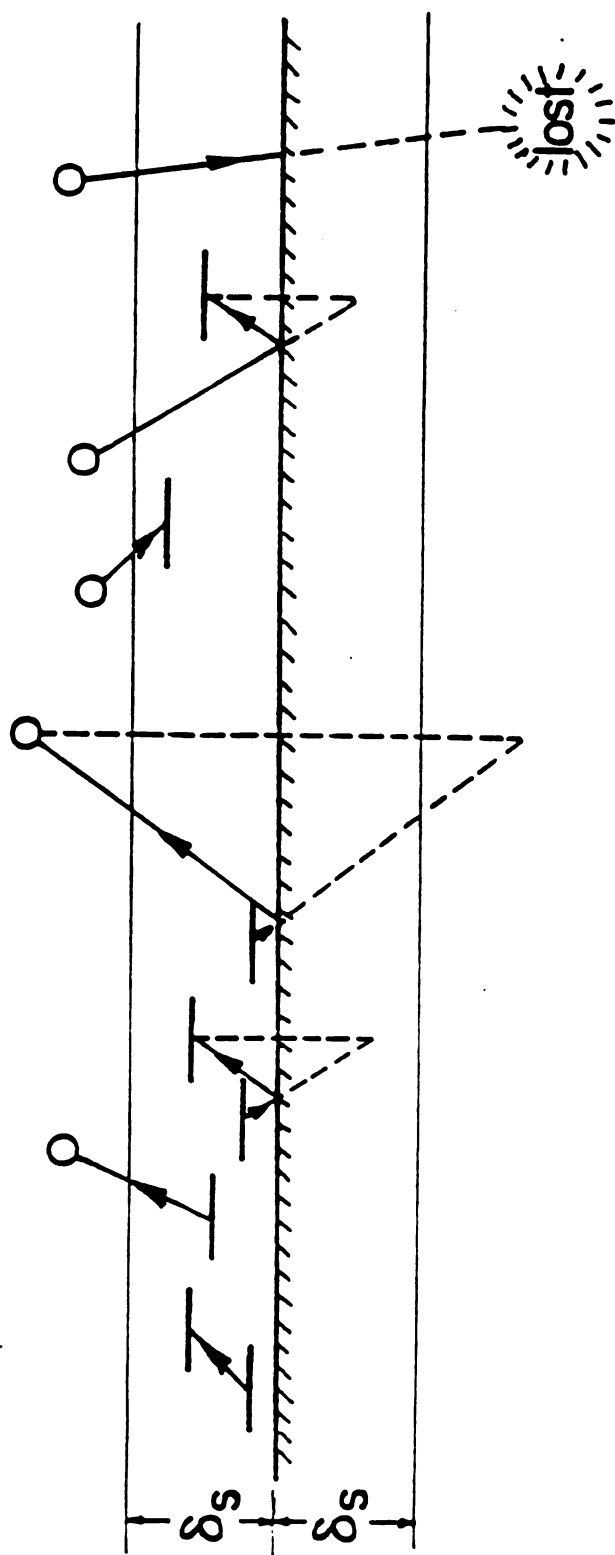


Figure 12. Vortex sorting

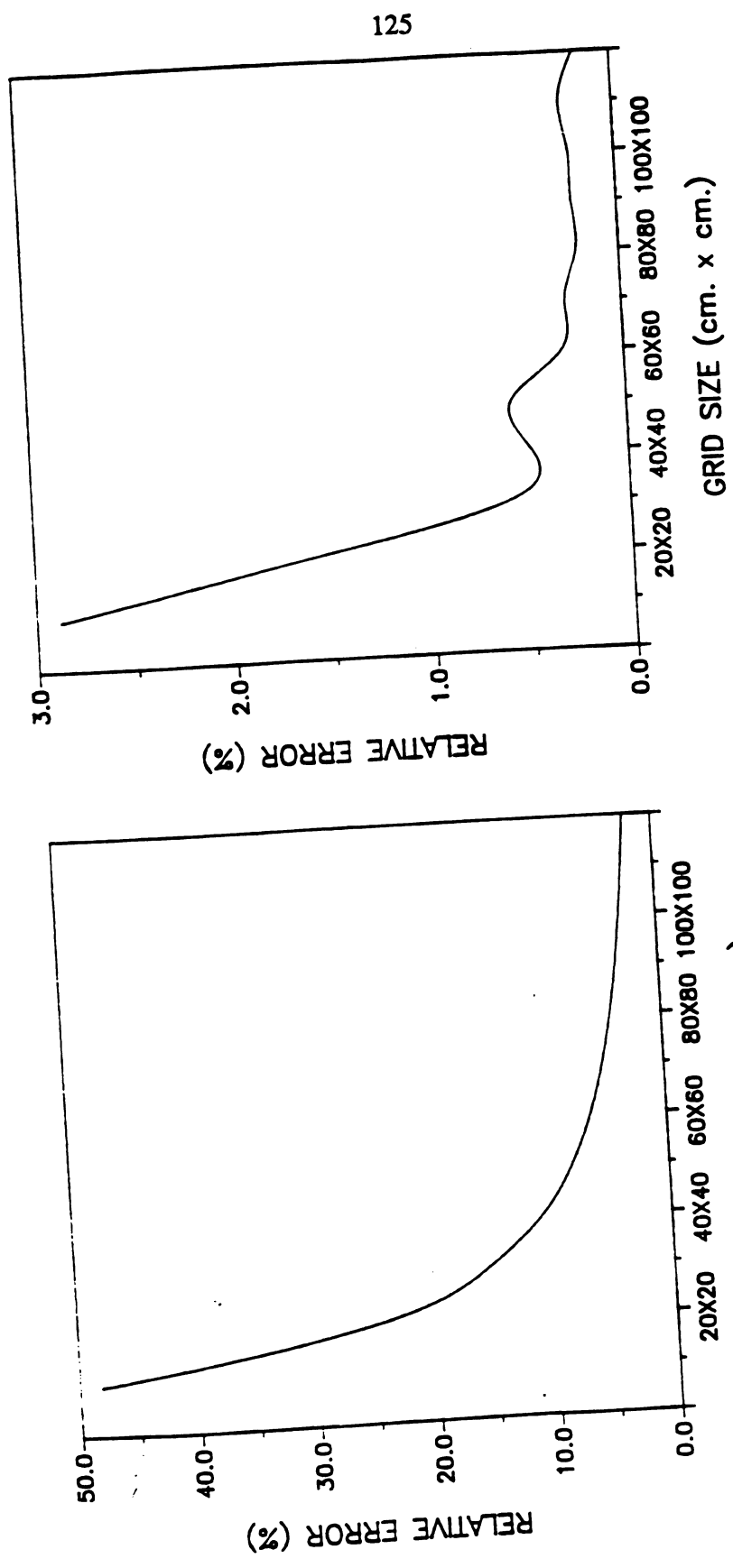


Figure 13. Relative error between exact and numerical streamfunction values in the square domain using a single vortex and a system of 105 vortices.

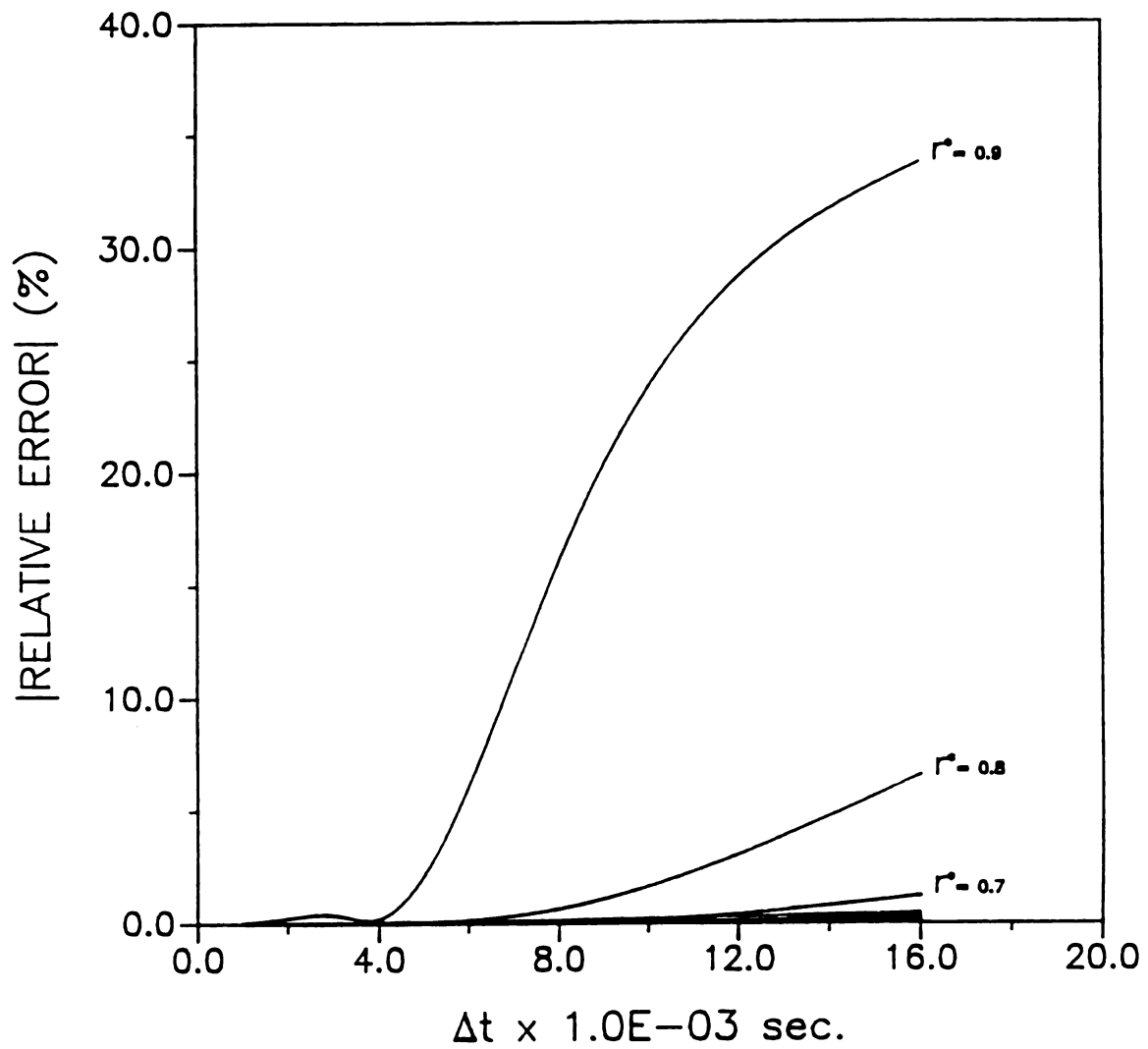


Figure 14. Relative error between the initial and final positions of a point vortex after one complete revolution.

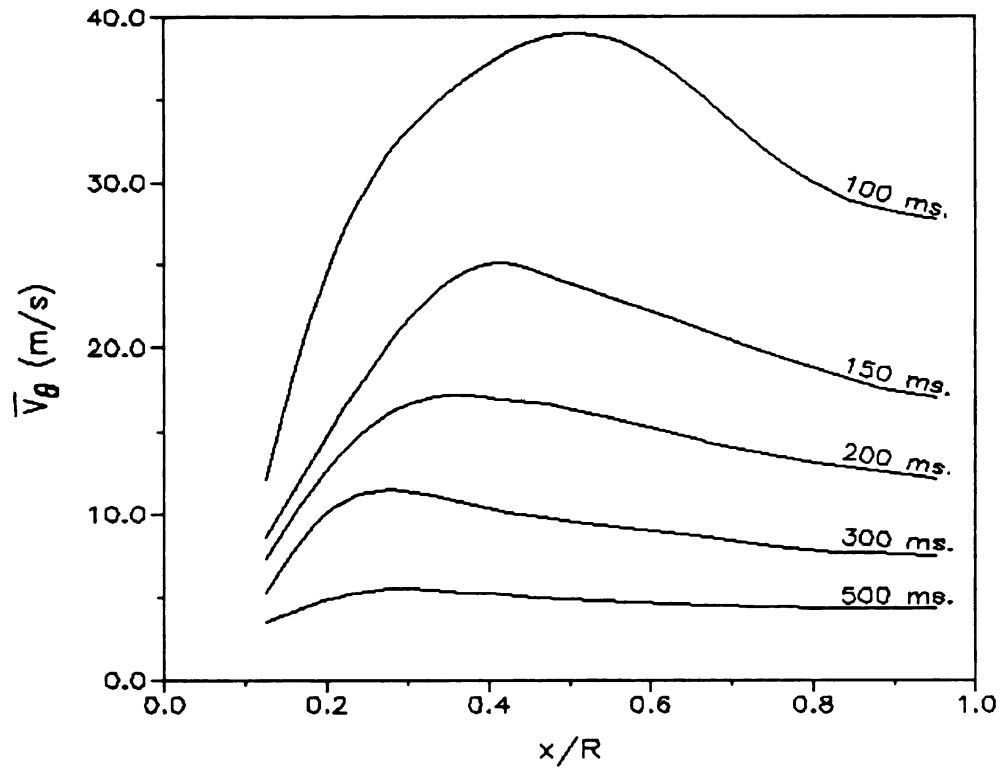


Figure 15. Radial distribution of tangential velocity component.

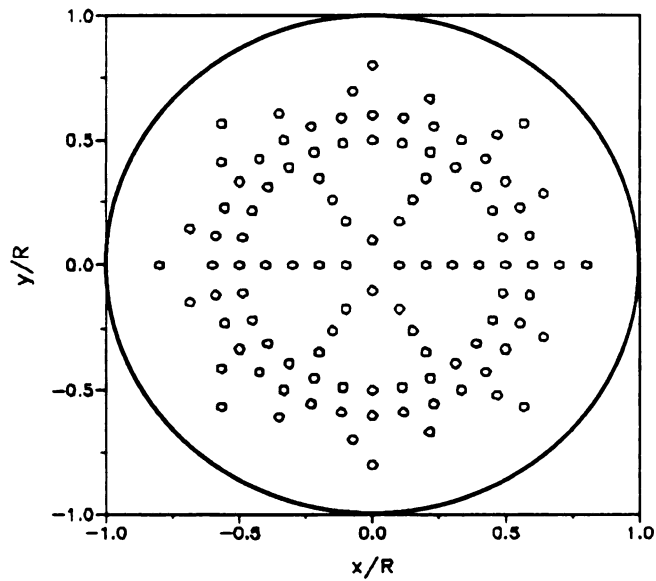


Figure 16. Geometric distribution of the system of 105 discrete vortices ($\tau_0=100$ ms.)

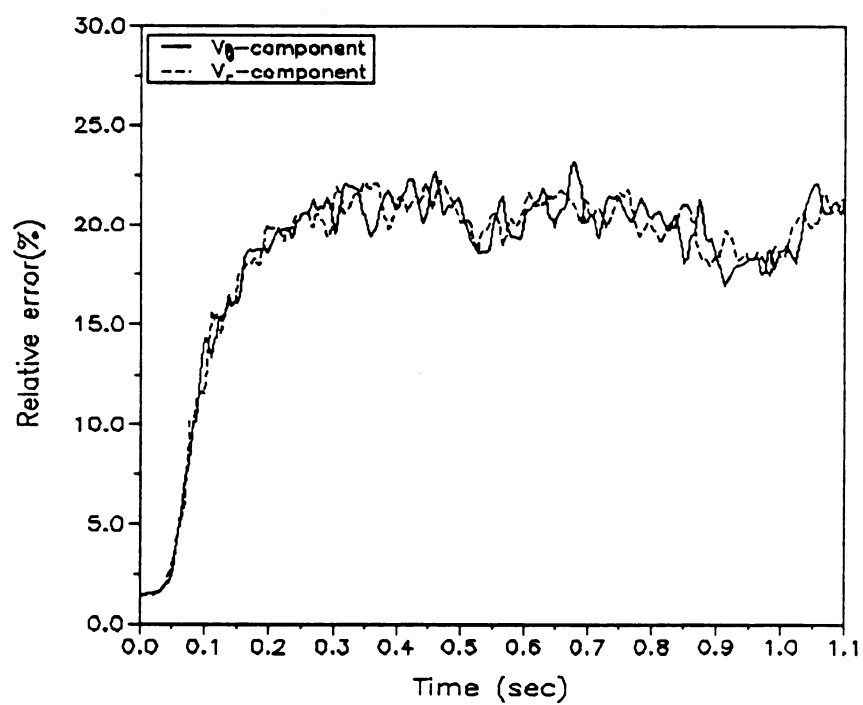


Figure 17. Relative errors in V_r and V_θ component velocities as a function of time.

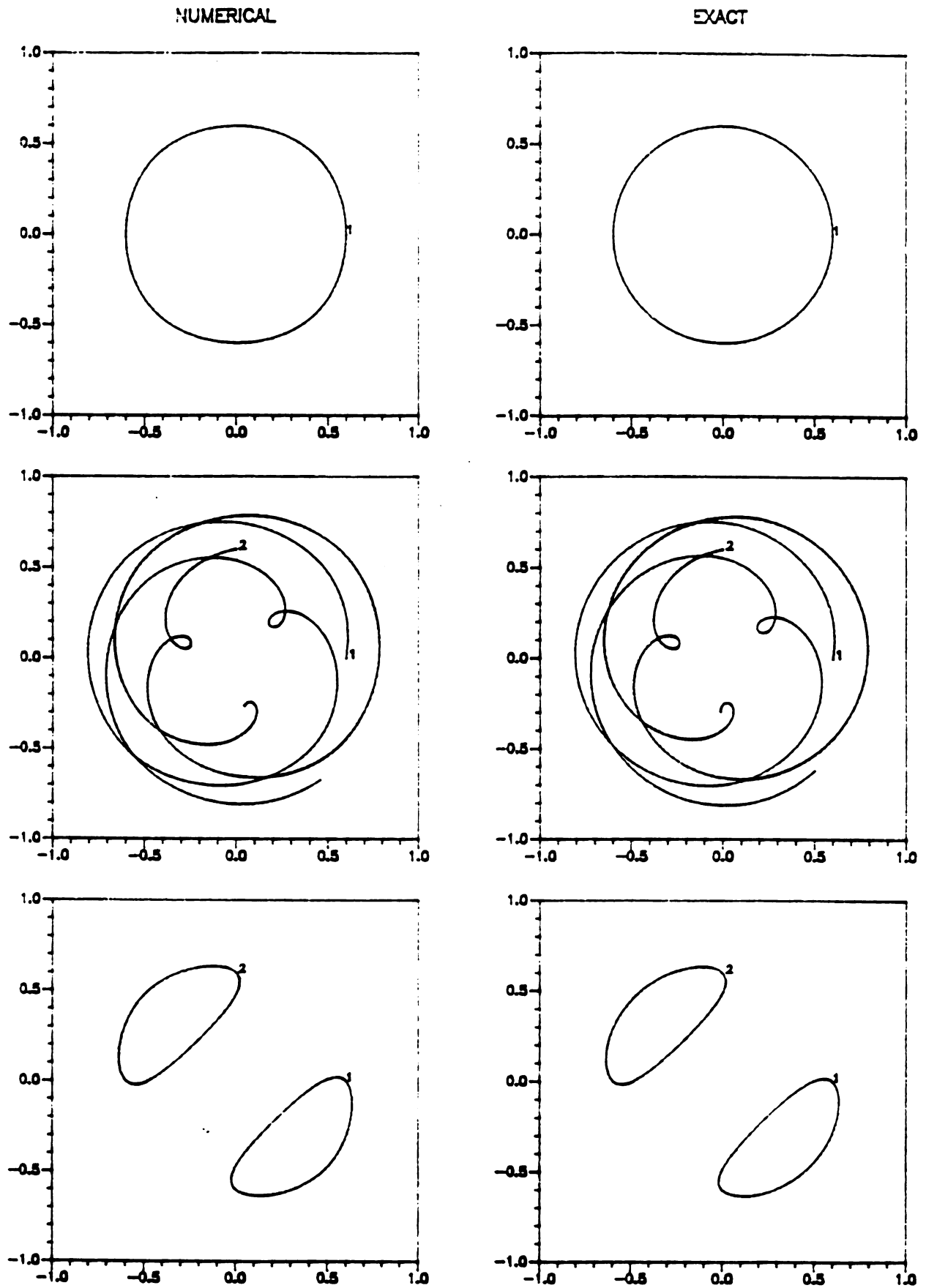


Figure 18. Trajectories of systems of discrete vortices

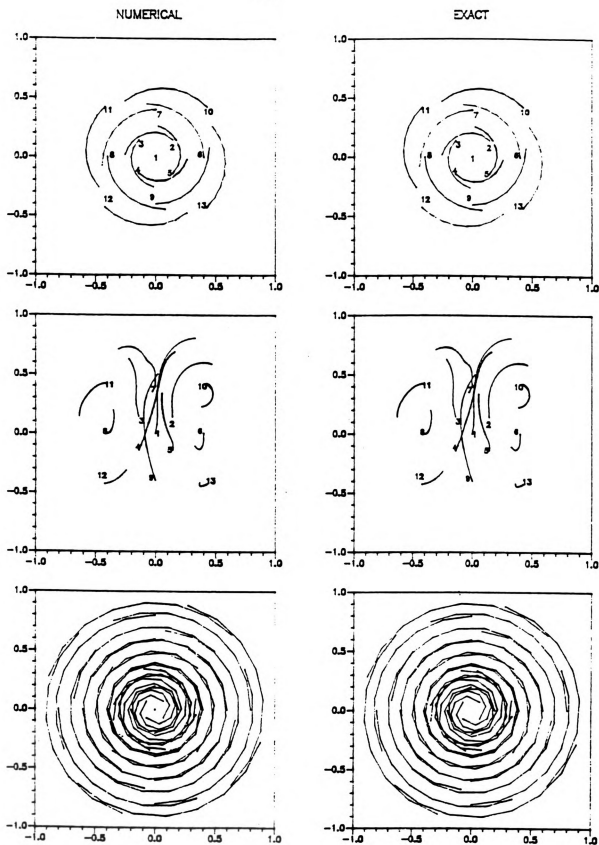


Figure 18. (con'td)

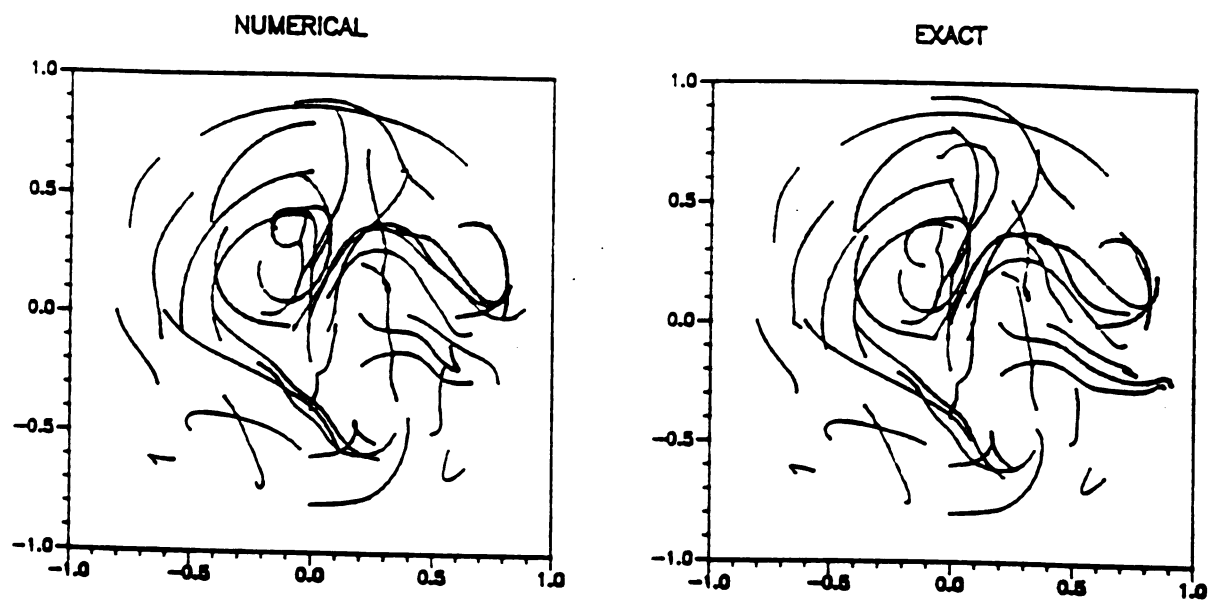


Figure 18. (con'td)

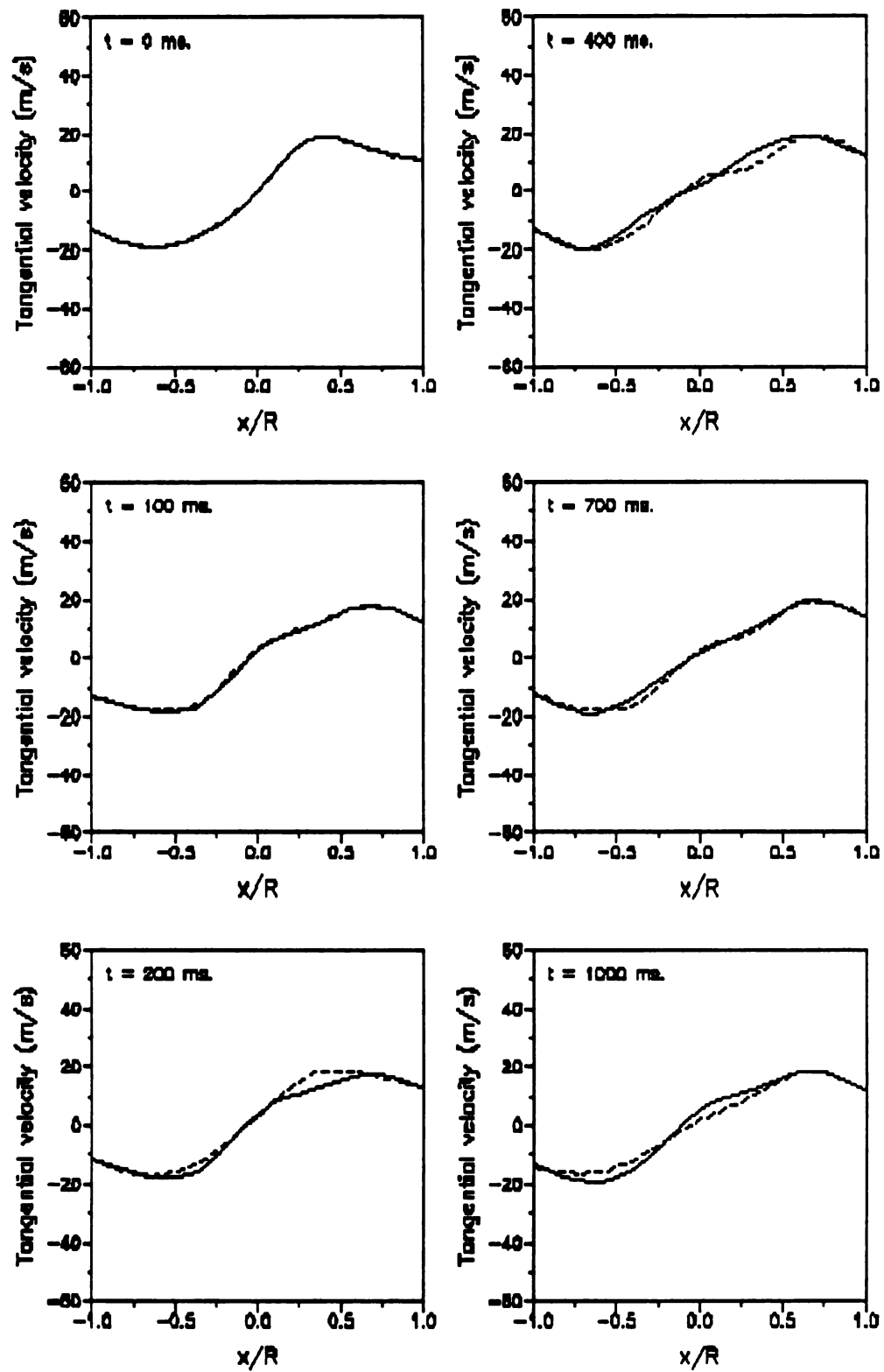


Figure 19. Tangential velocity profiles at discrete time steps representative of short and long time errors.

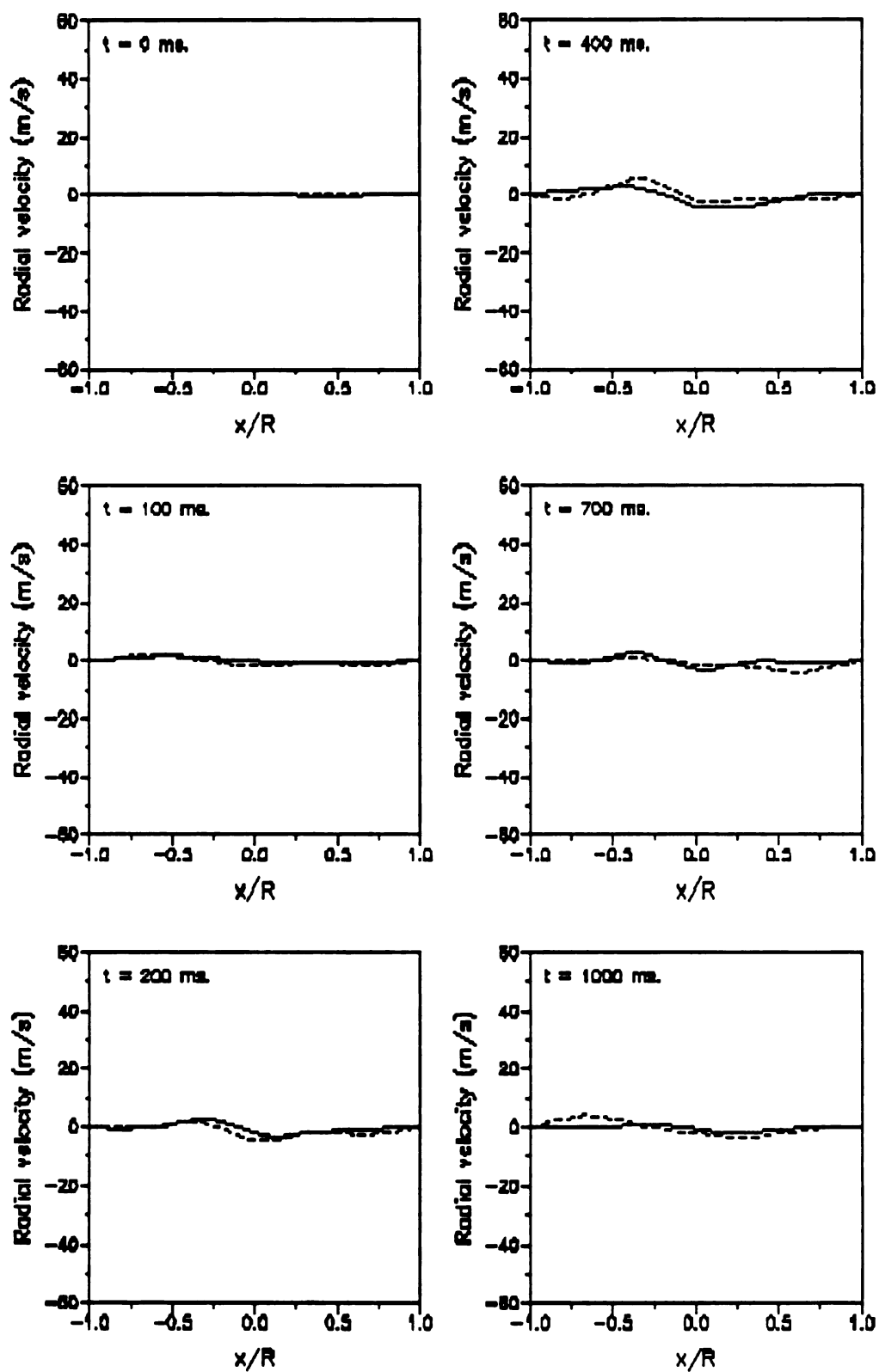
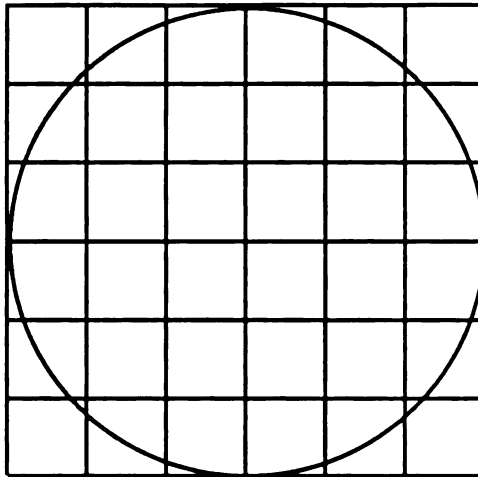
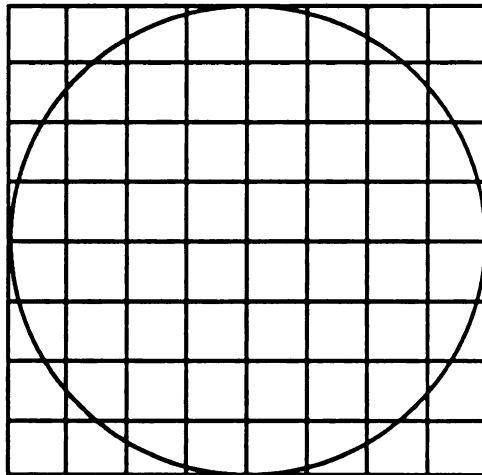


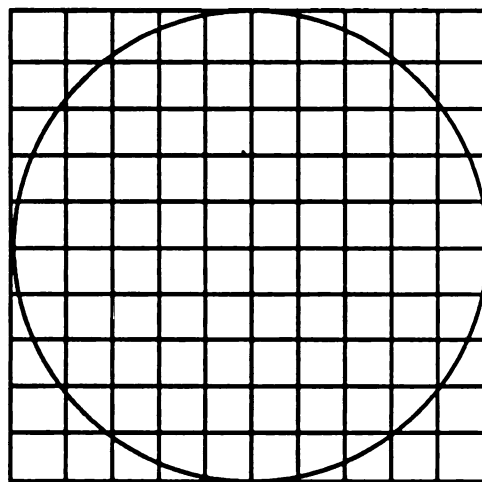
Figure 20. Radial velocity profiles at discrete time steps representative of short and long time errors.



6 x 6
(L = 32)



8 x 8
(L = 52)



10 x 10
(L = 80)

Figure 21. Uniform grid squares superimposed on circular domain.

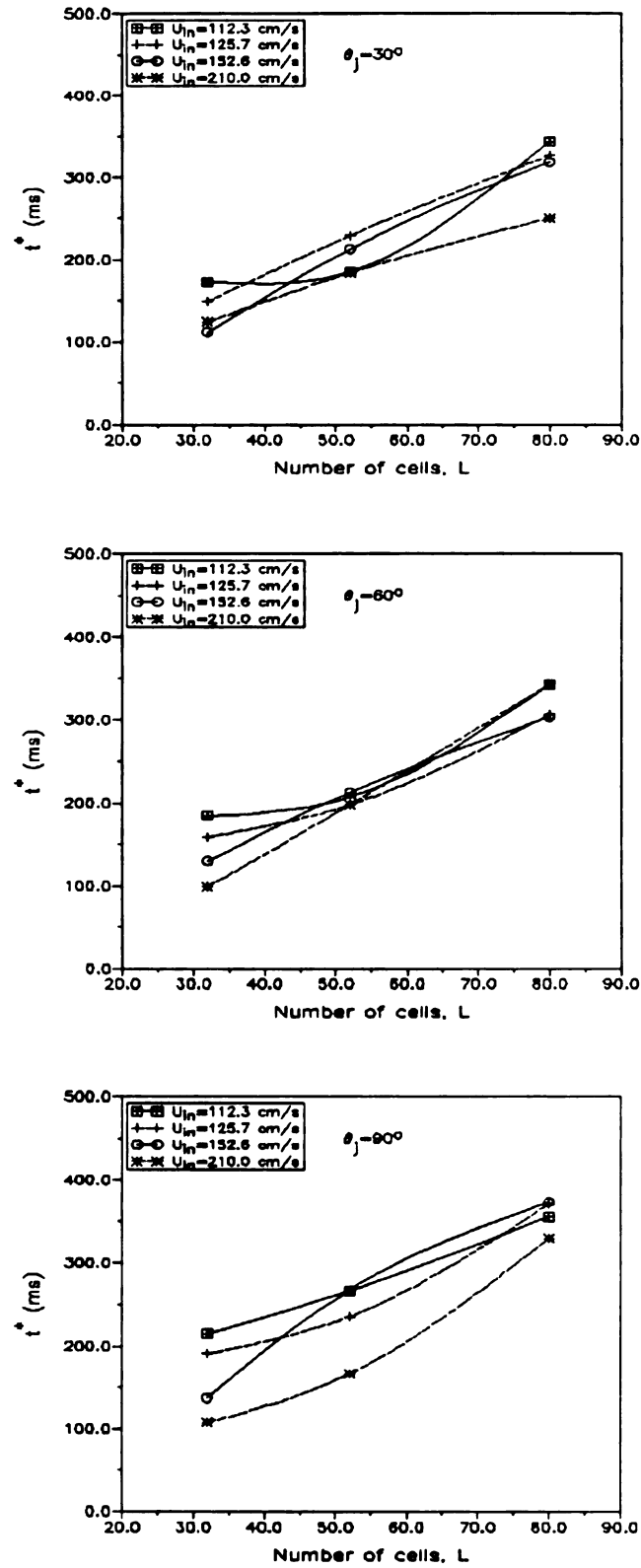


Figure 22. Complete mixing time as a function of the number of cells.
(Swirl strength $S_0 = 12.5$ cm·m/s. Number of vortices per cell $N_c = 2$)

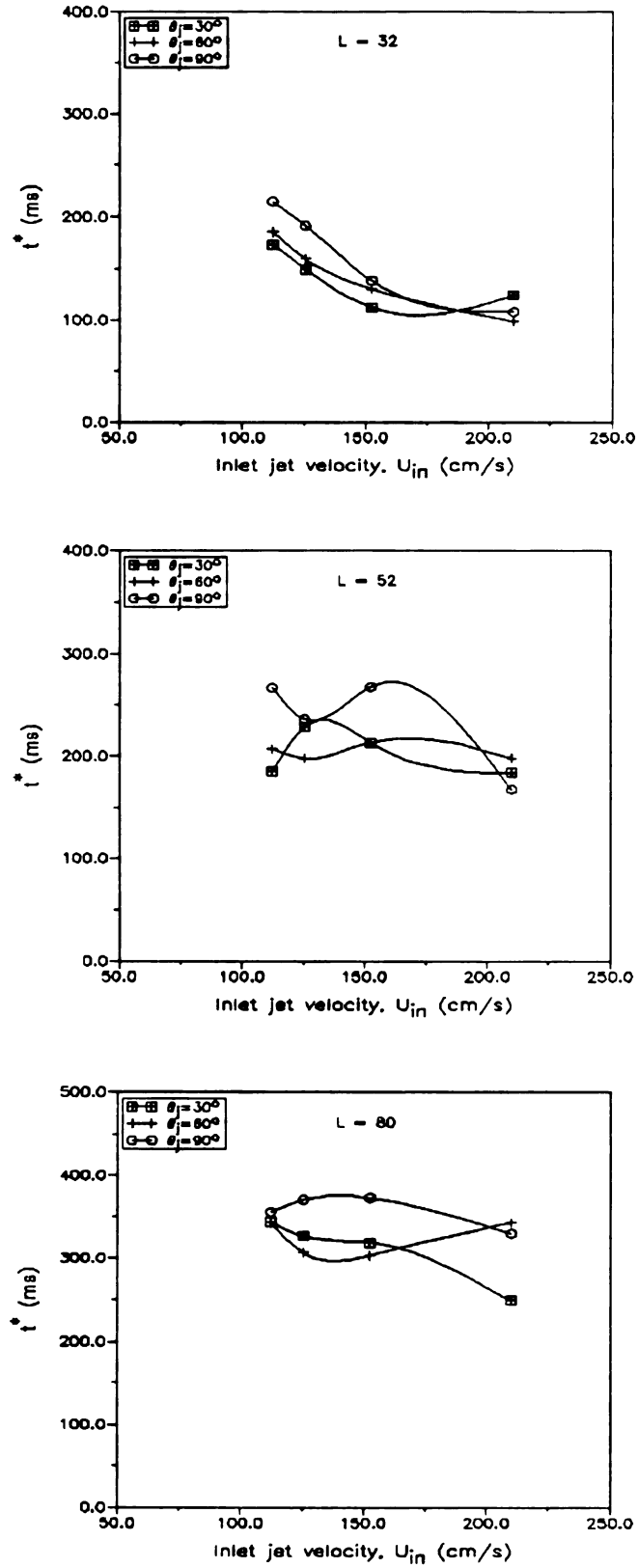


Figure 23. Complete mixing time as a function of the inlet jet velocity.
(Swirl strength $S_\theta = 12.5$ cm·m/s. Number of vortices per cell $N_c = 2$)

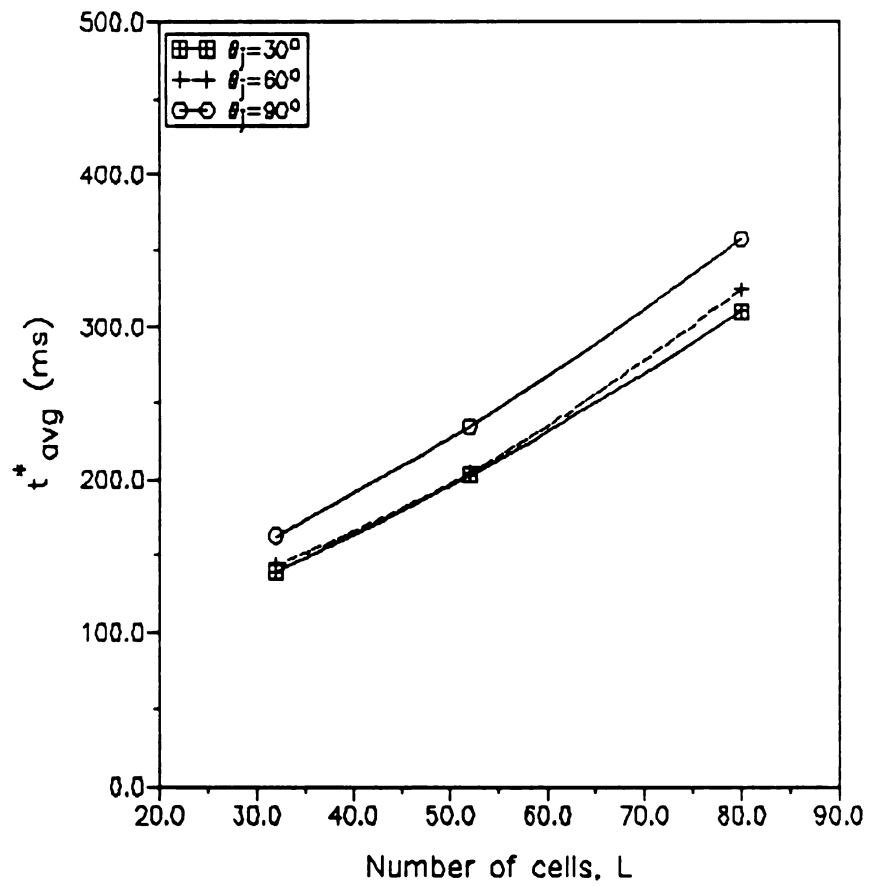


Figure 24. Average complete mixing time as a function of the number of cells. (Swirl strength $S_\theta = 12.5$ cm·m/s. Number of vortices per cell $N_c = 2$)

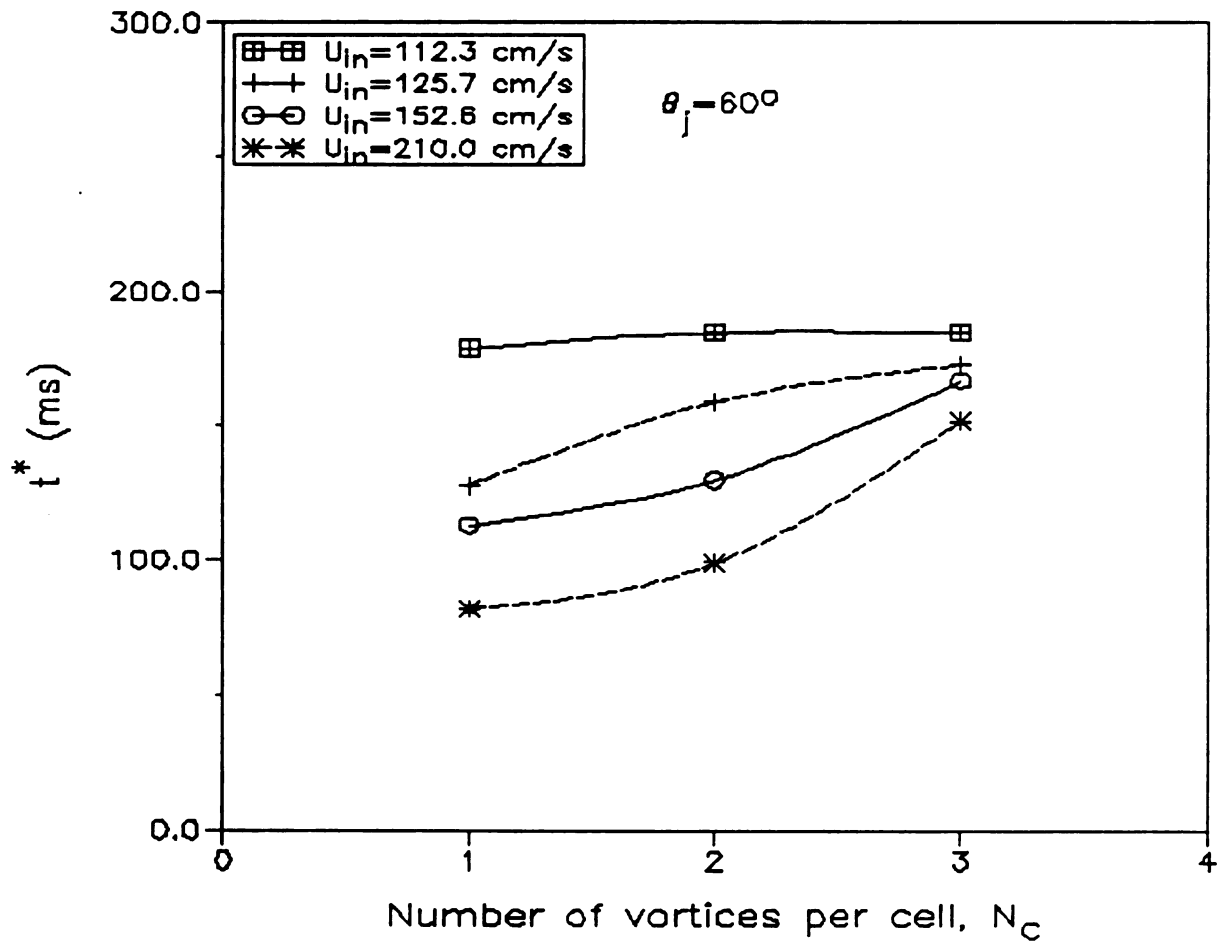


Figure 25. Complete mixing time as a function of the number of vortices per cell (Swirl strength $S_\theta = 12.5$ cm·m/s. Number of cells $L = 32$).

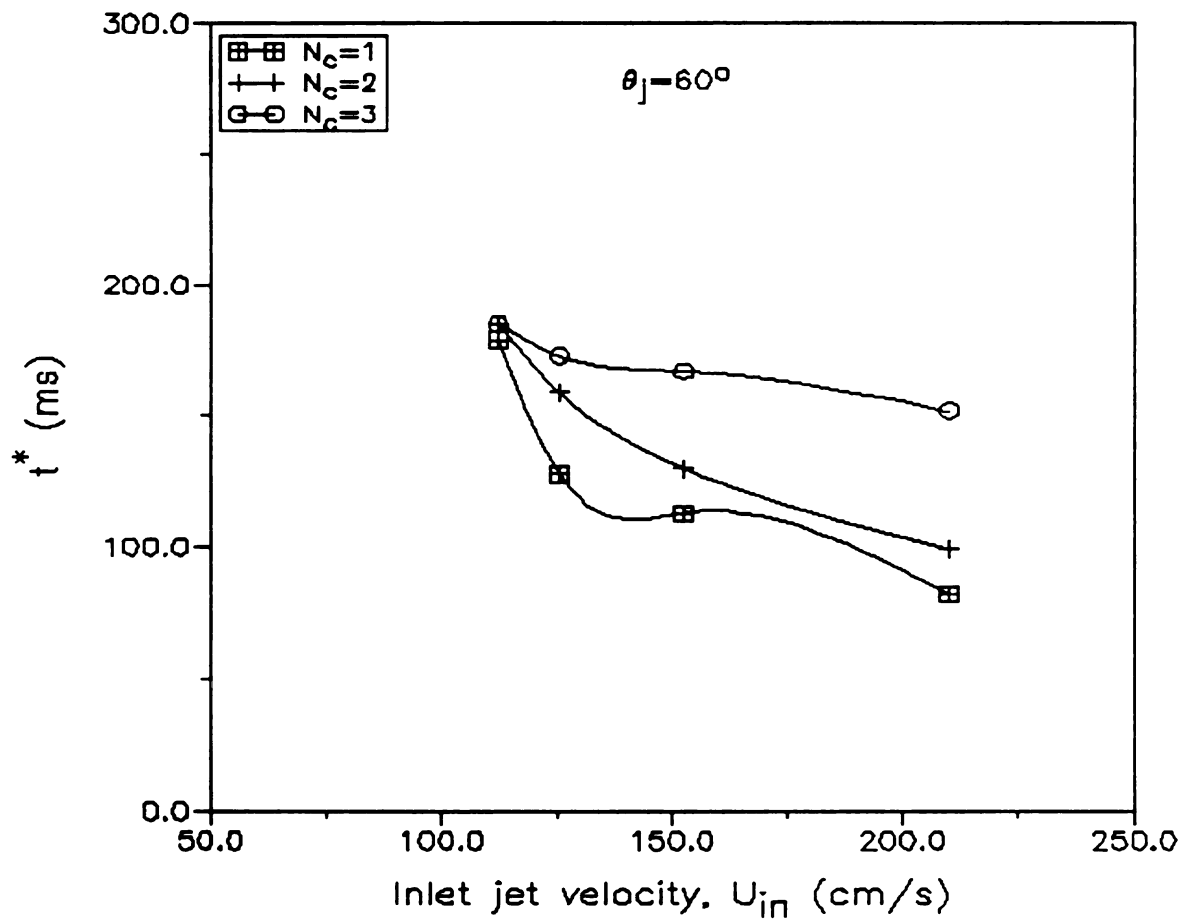


Figure 26. Complete mixing time as a function of the inlet jet velocity (Swirl strength $S_\theta=12.5$ cm·m/s. Number of cells $L=32$).

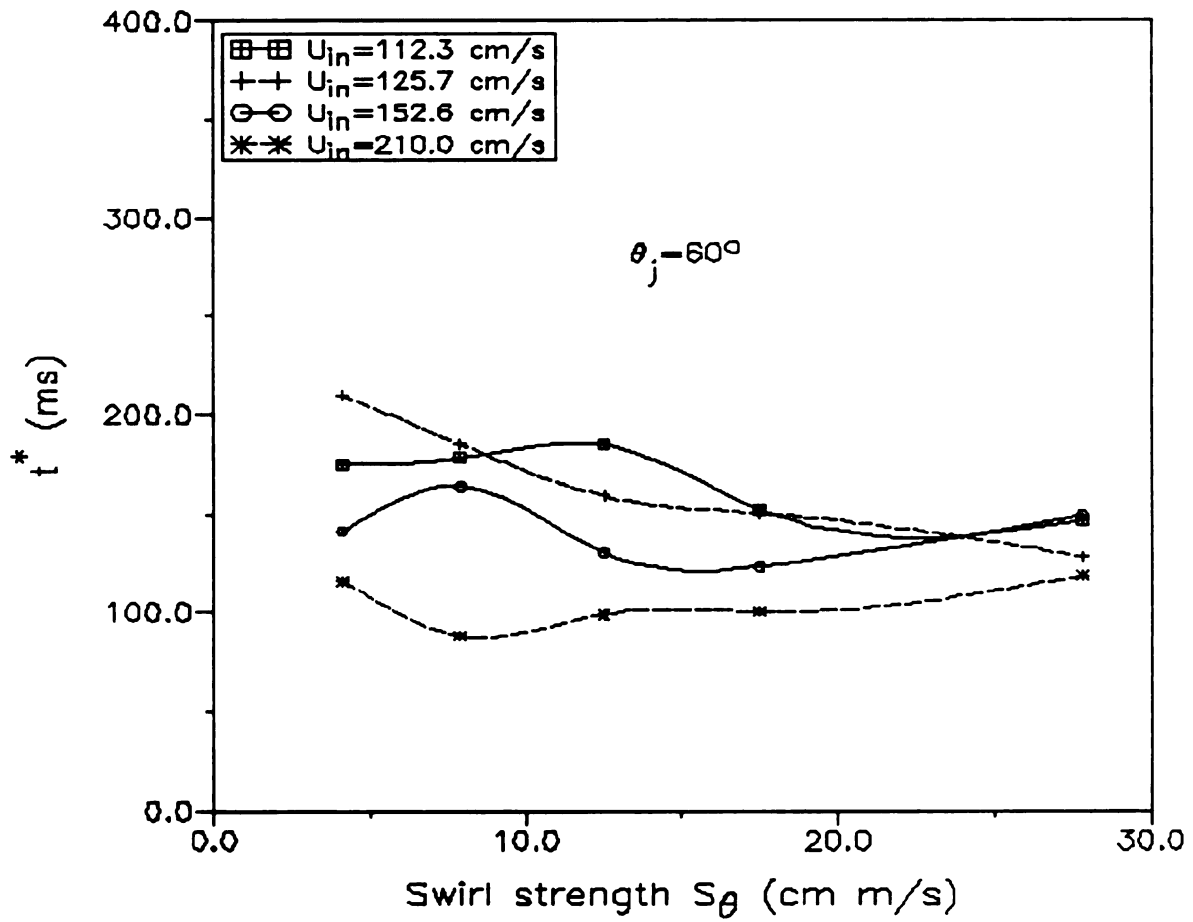


Figure 27. Complete mixing time as a function of the swirl strength.
(Number of cells $L=32$. Number of vortices per cell $N_c=2$)

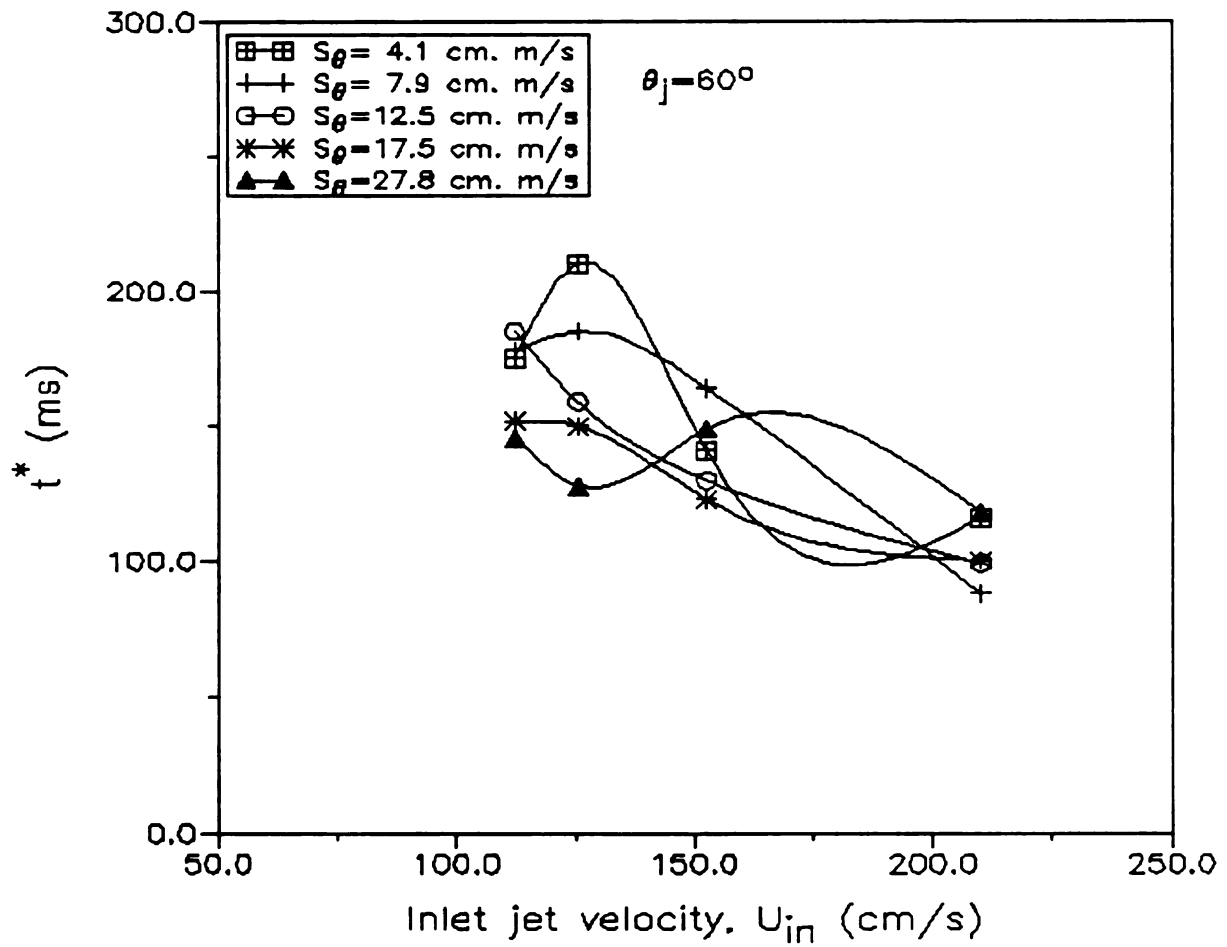


Figure 28. Complete mixing time as a function of the inlet jet velocity.
(Number of cells $L=32$. Number of vortices per cell $N_c=2$)

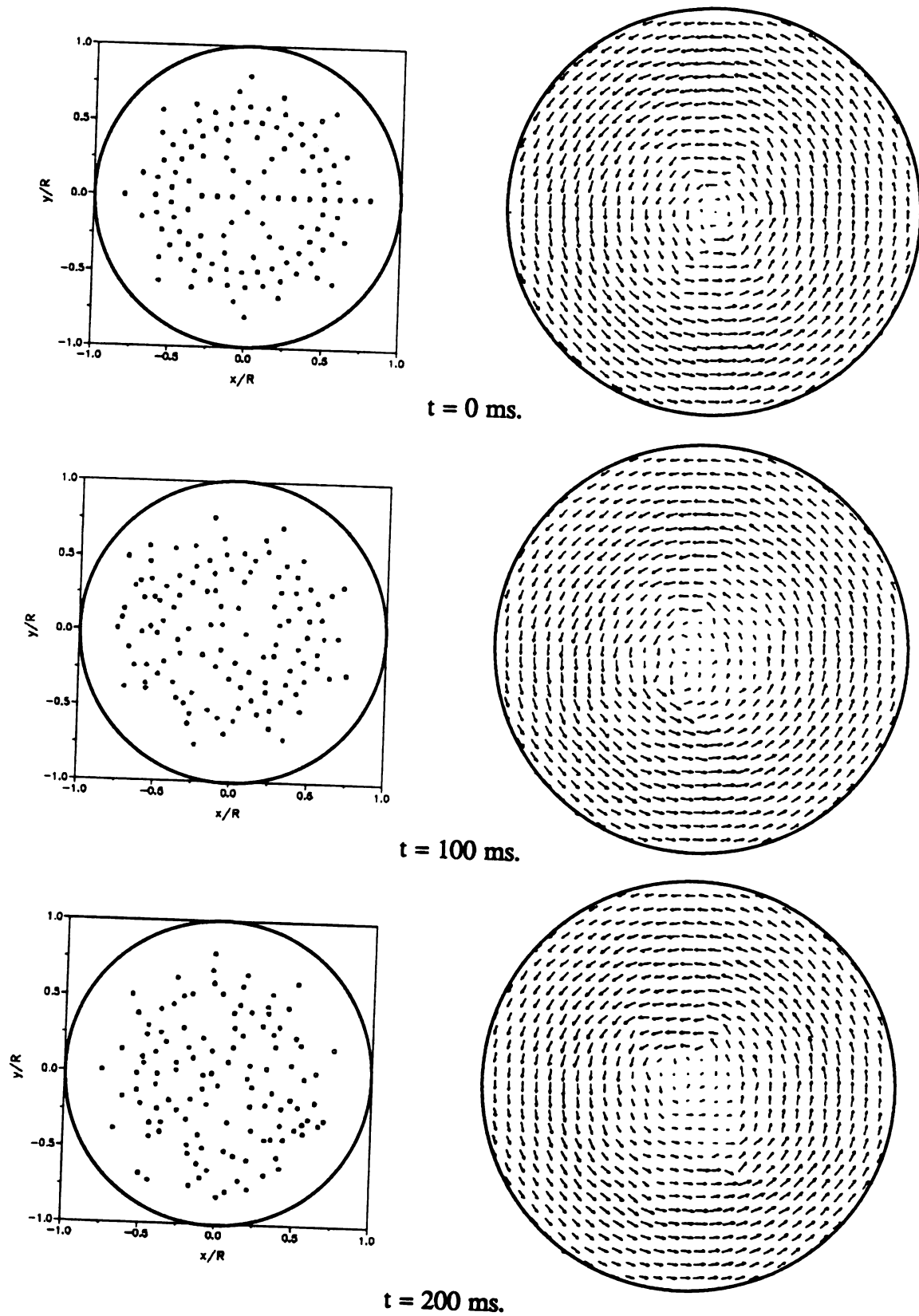


Figure 29. Geometric distribution of the system of 115 discrete vortices and velocity vector representation of the induced swirling motion.

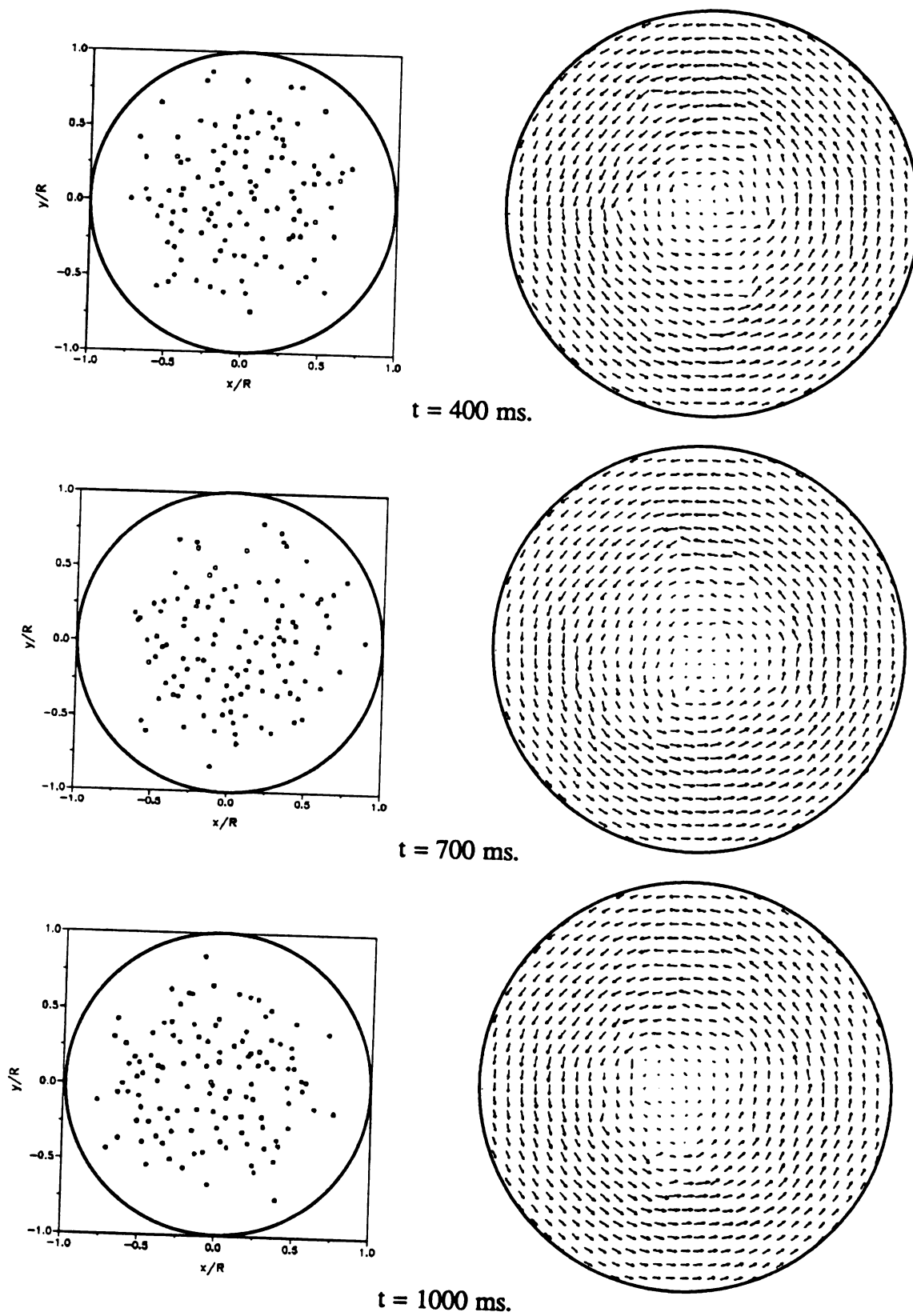


Figure 29 (contd.).

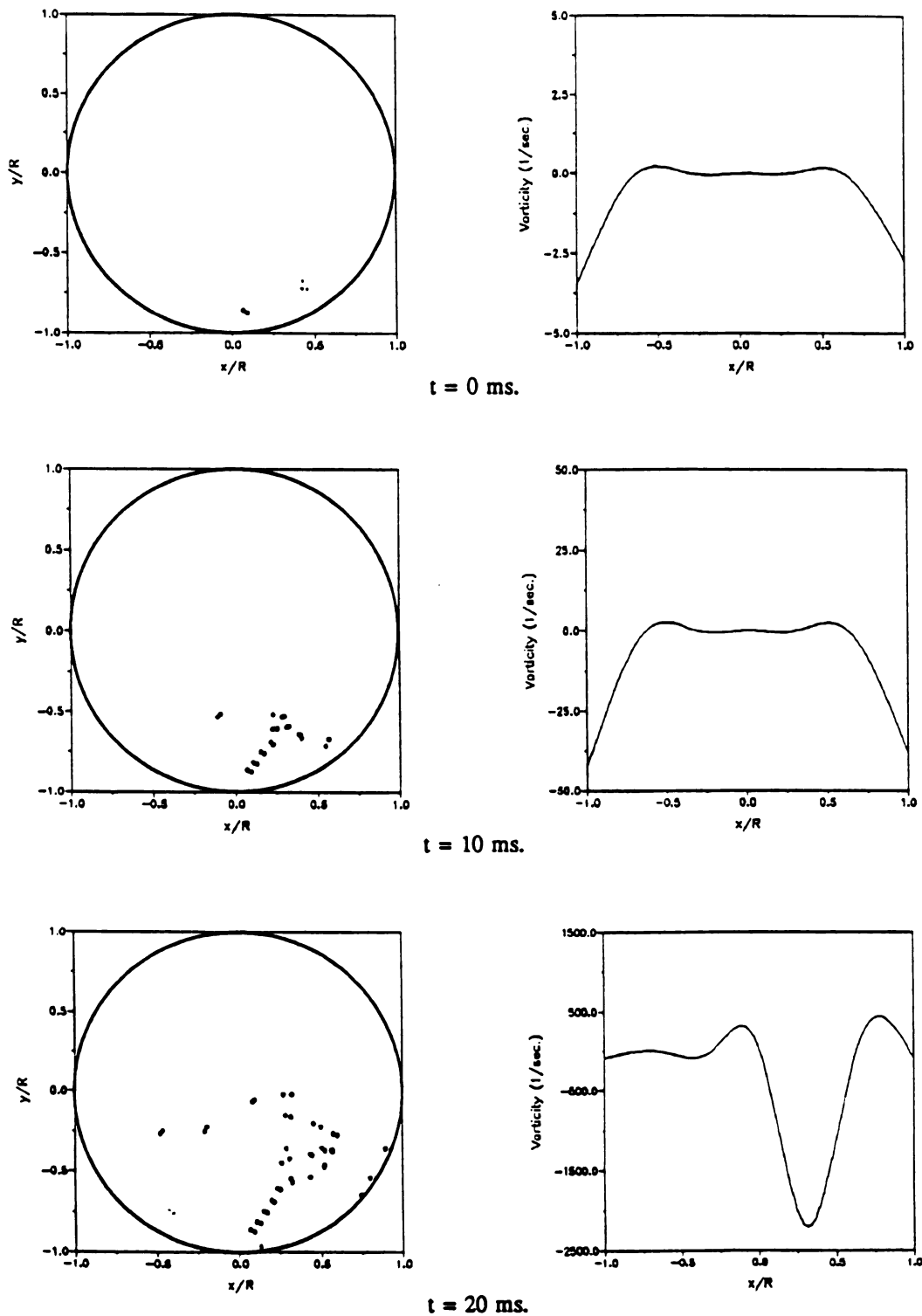
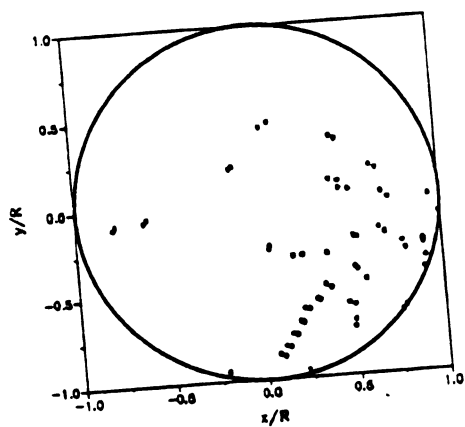
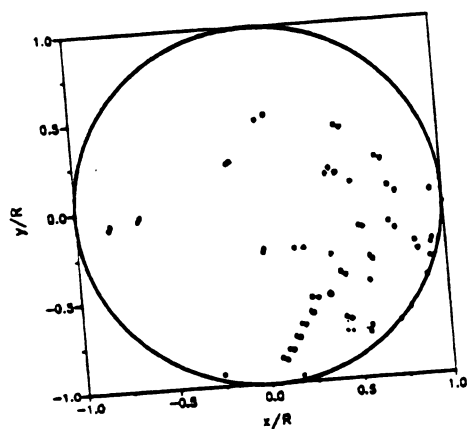
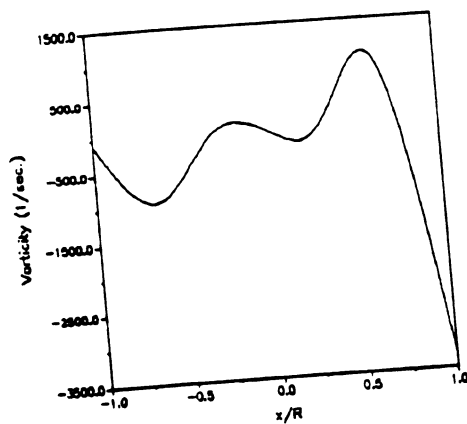


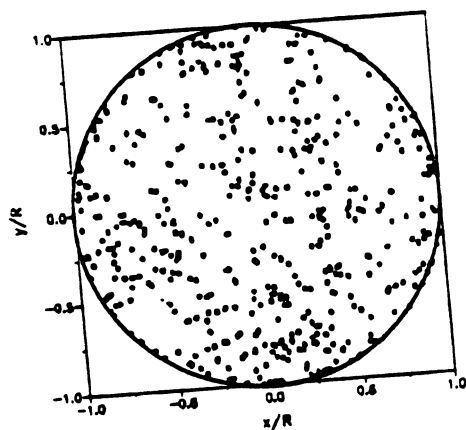
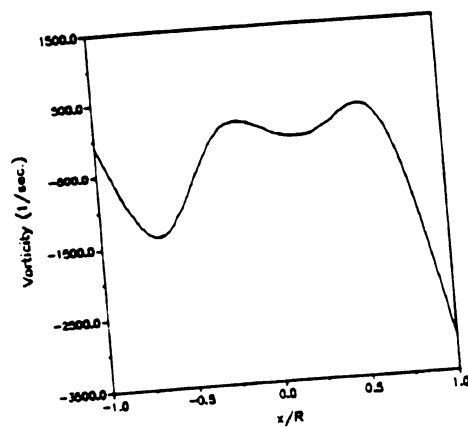
Figure 30. Fuel particle trajectory and vorticity profile at $y=0$ as a function of time. Injector at 60 degrees (counterclockwise from horizontal). $U_{in} = 210.0$ m/s.



$t = 29$ ms.



$t = 30$ ms.



$t = 271$ ms.

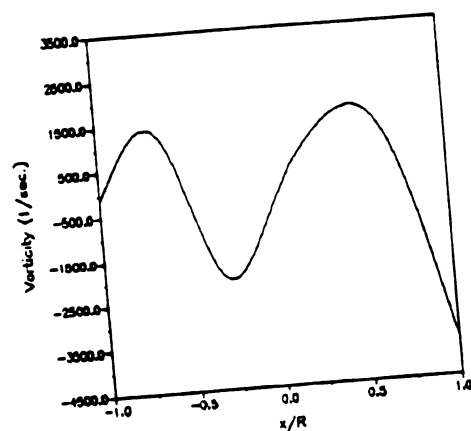
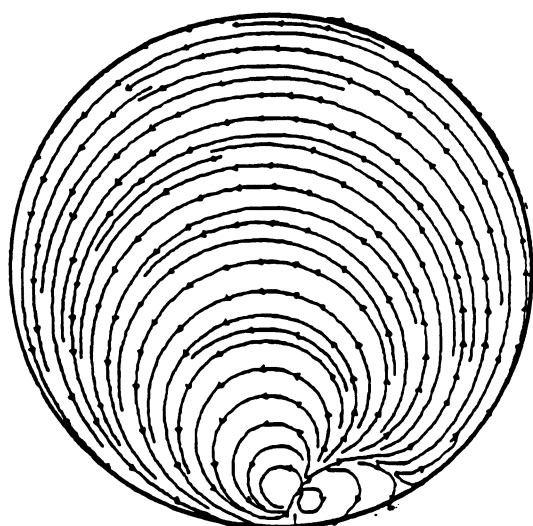
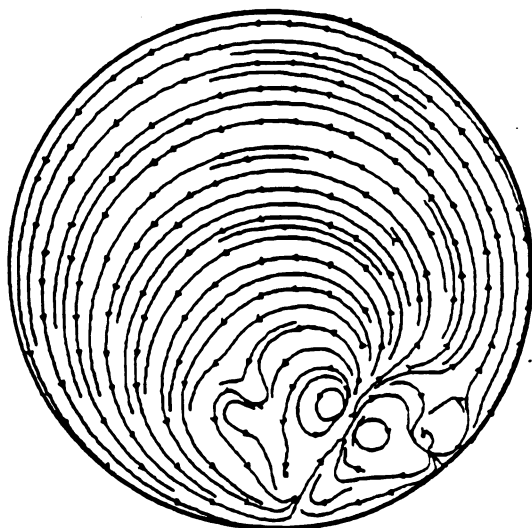
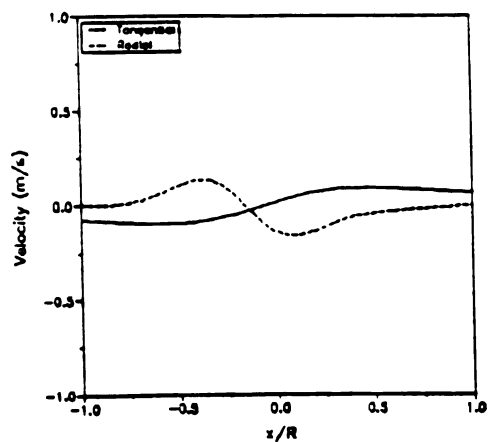


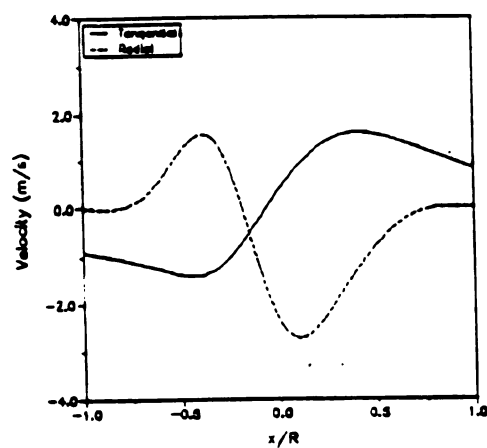
Figure 30 (con'td).



$t = 0$ ms.



$t = 10$ ms.



$t = 20$ ms.

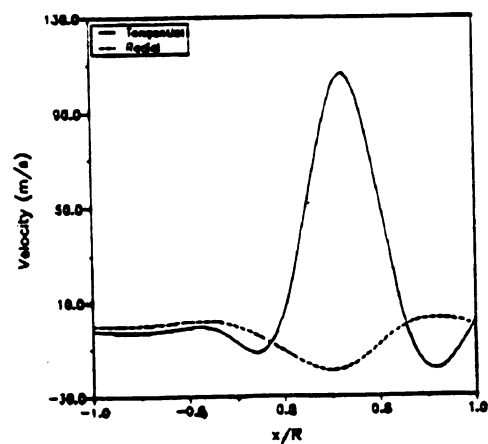
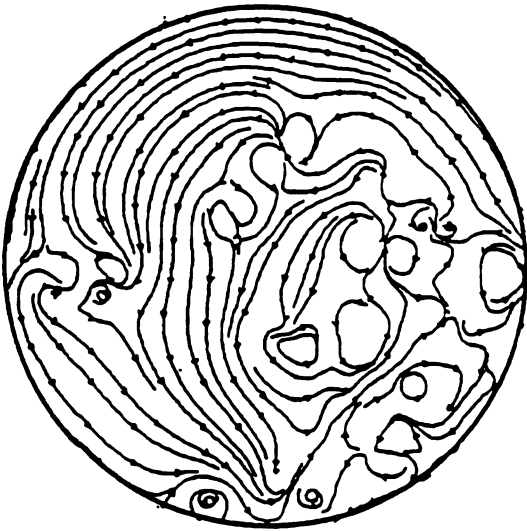
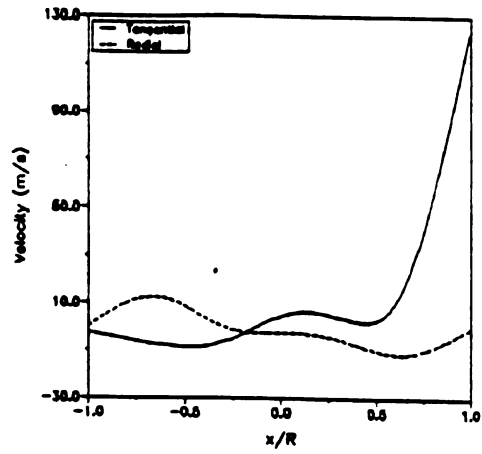


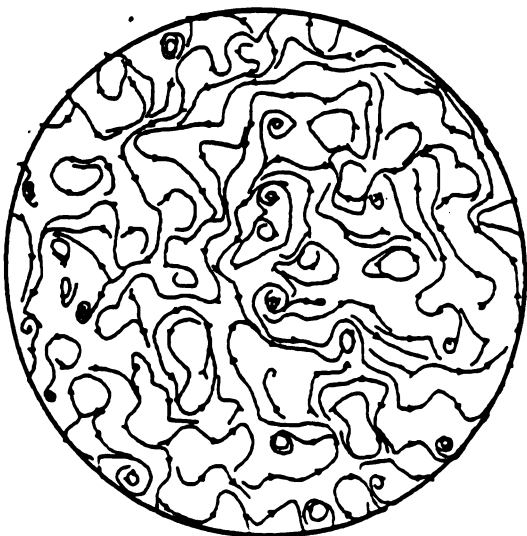
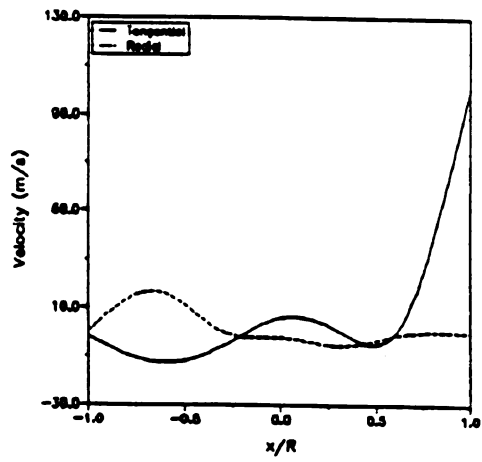
Figure 31. Streamlines and velocity profiles at $y=0$ as a function of time (without swirl). Injector at 60 degrees (counterclockwise from horizontal). $U_{in} = 210.0$ m/s.



$t = 29$ ms.



$t = 30$ ms.



$t = 271$ ms.

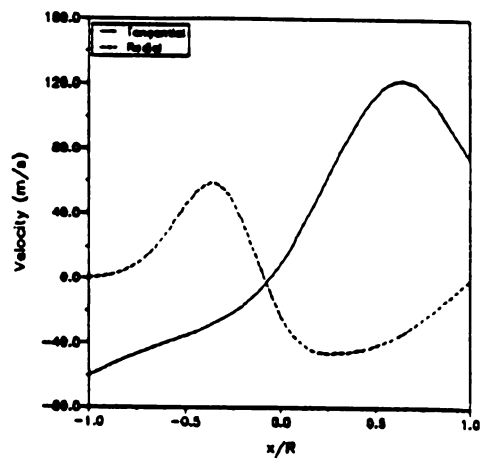


Figure 31 (con'td).

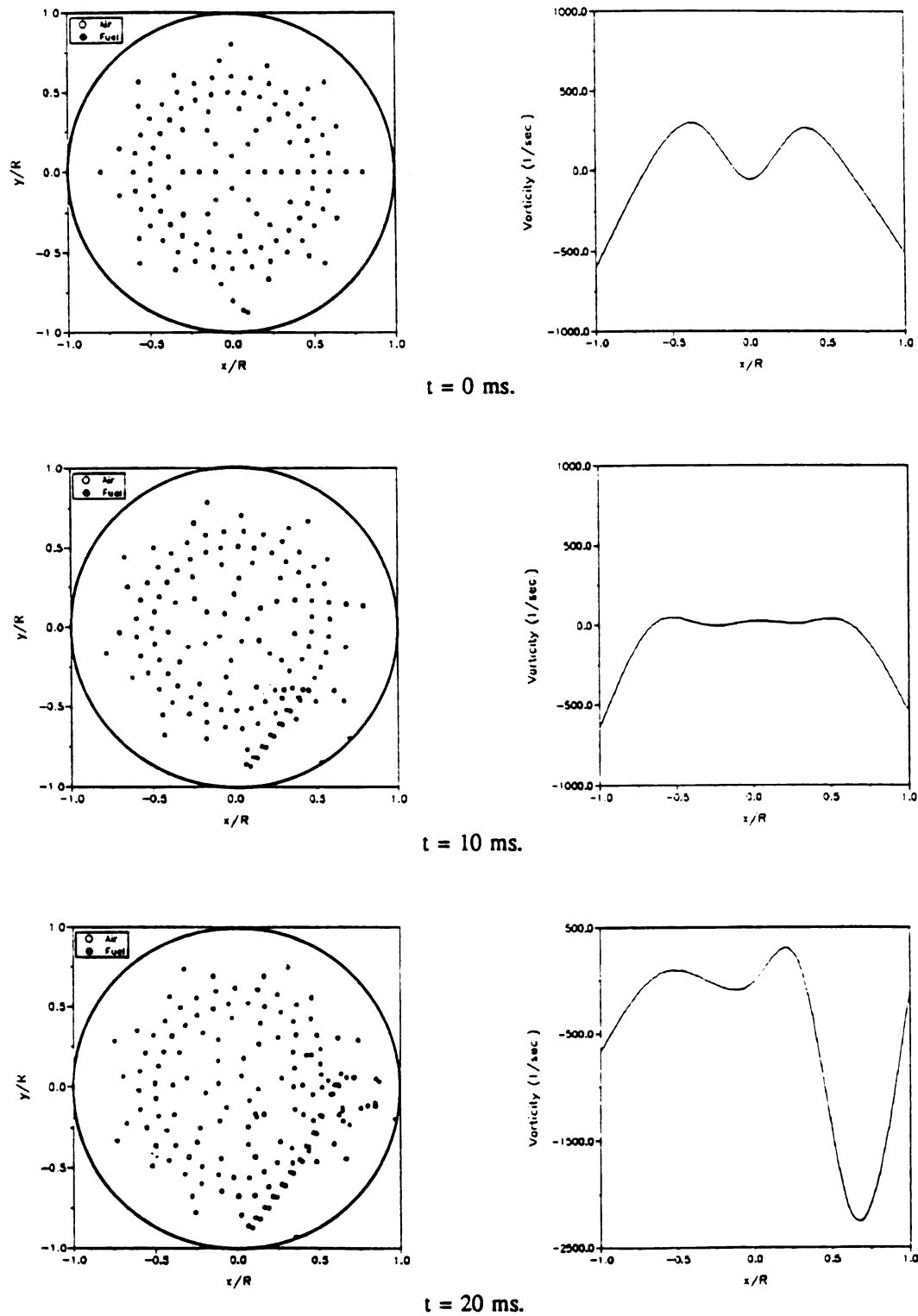
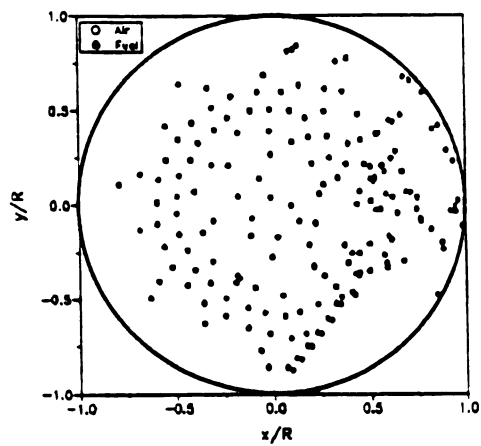
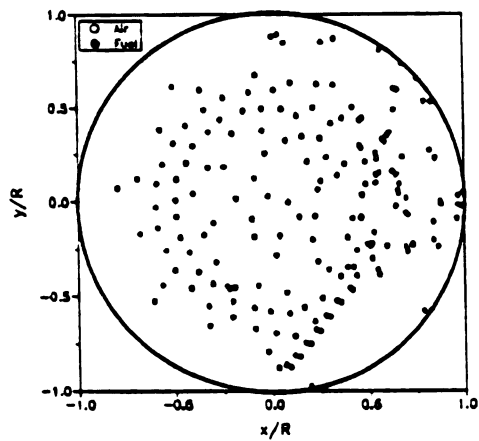
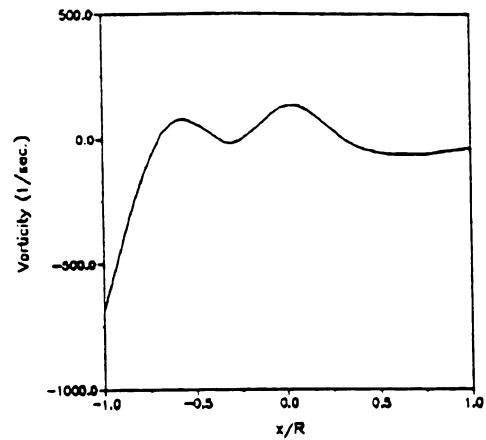


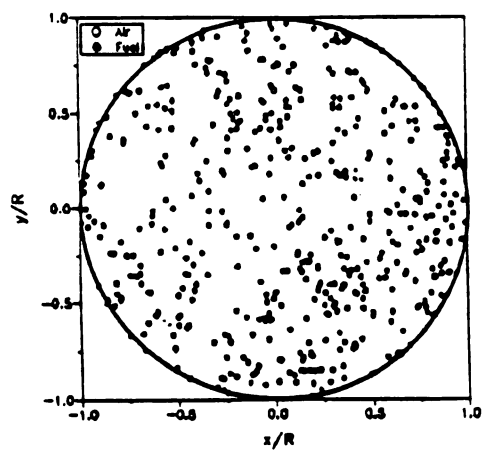
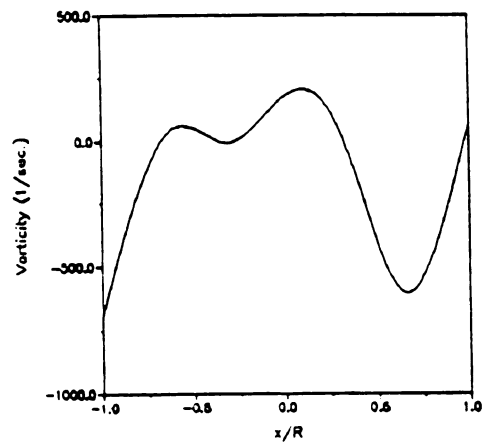
Figure 32. Air and Fuel particle trajectory and vorticity profile at $y=0$ as a function of time. Injector at 60 degrees (counterclockwise from horizontal). $U_{in} = 210.0$ m/s.



$t = 30$ ms.



$t = 32$ ms.



$t = 198$ ms.

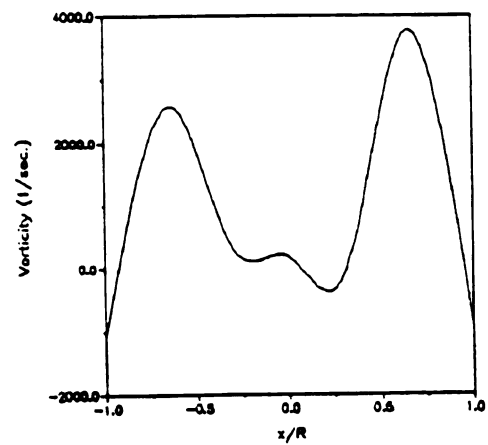
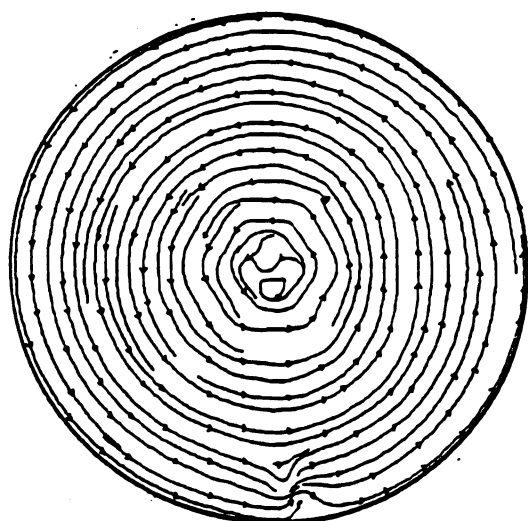
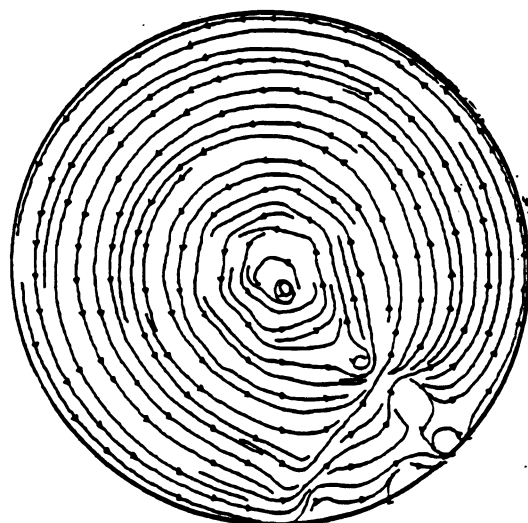
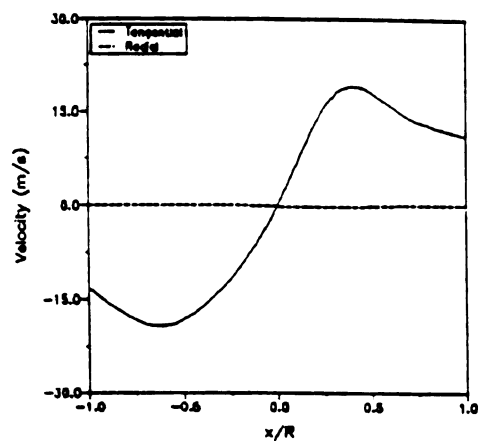


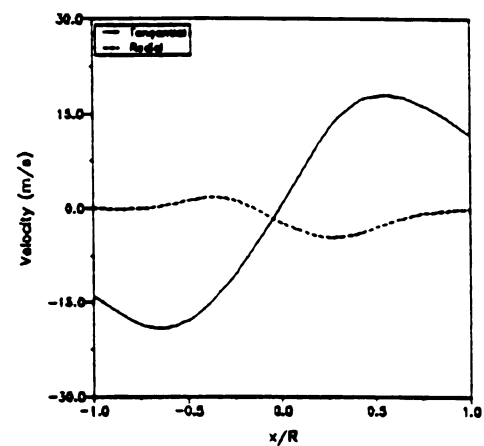
Figure 32 (con'td).



$t = 0$ ms.



$t = 10$ ms.



$t = 20$ ms.

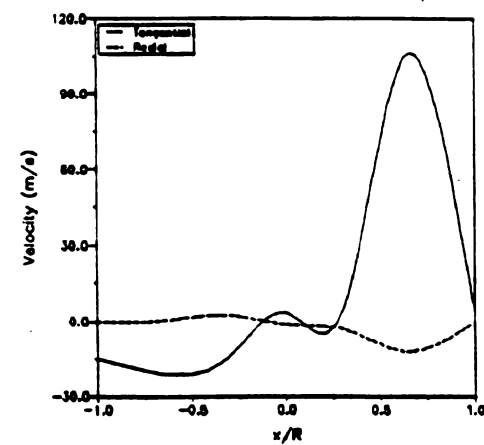
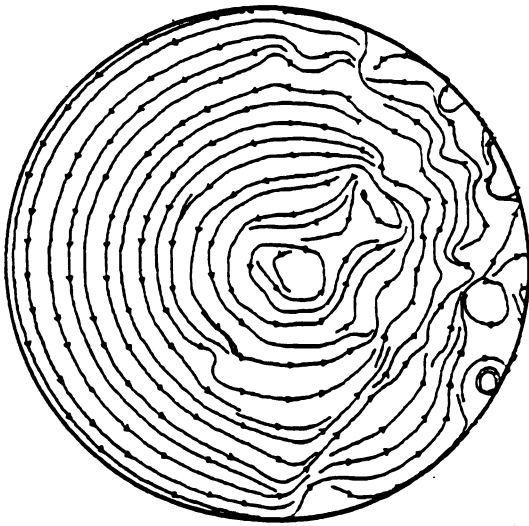
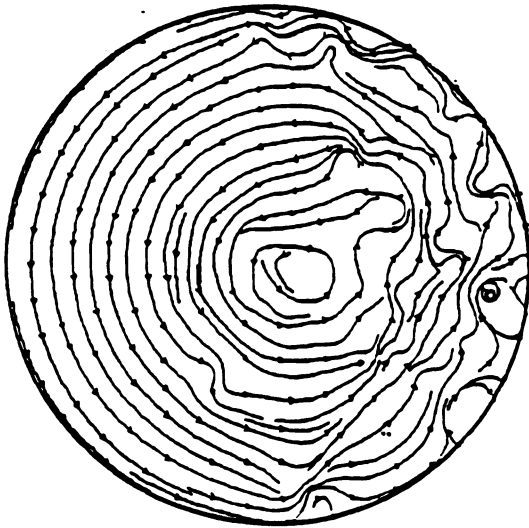
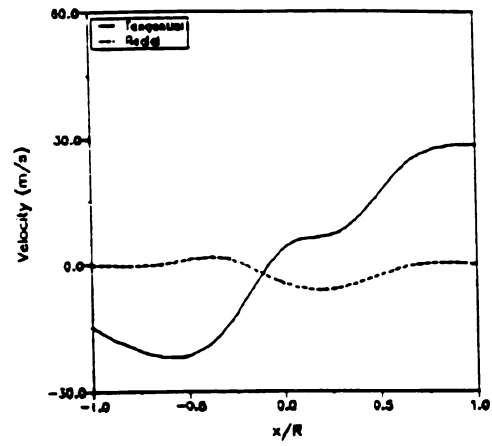


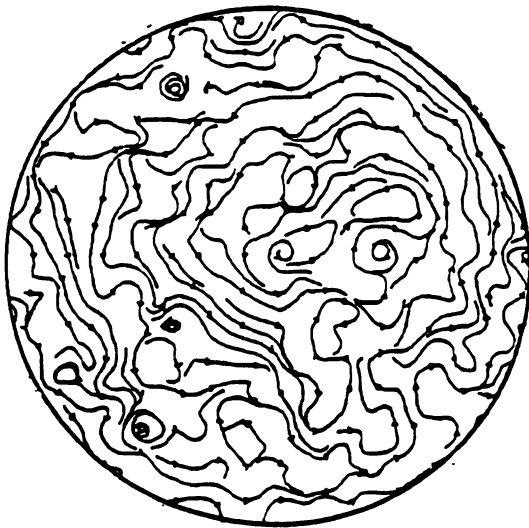
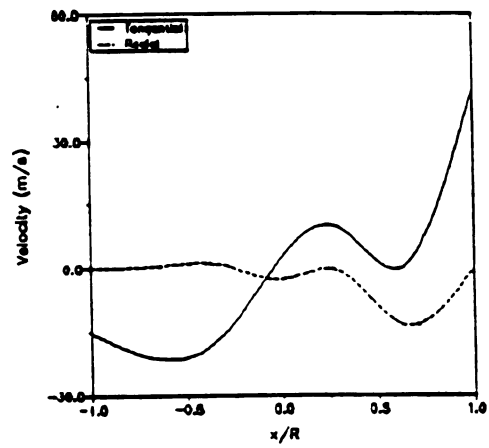
Figure 33. Streamlines and velocity profiles at $y=0$ as a function of time (with swirl). Injector at 60 degrees (counterclockwise from horizontal). $U_{in} = 210.0$ m/s.



$t = 30$ ms.



$t = 32$ ms.



$t = 198$ ms.

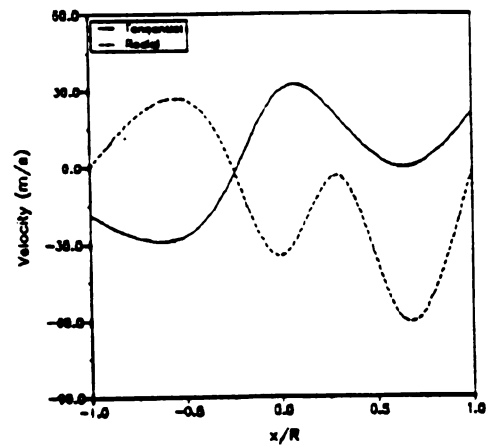


Figure 33 (con'td).

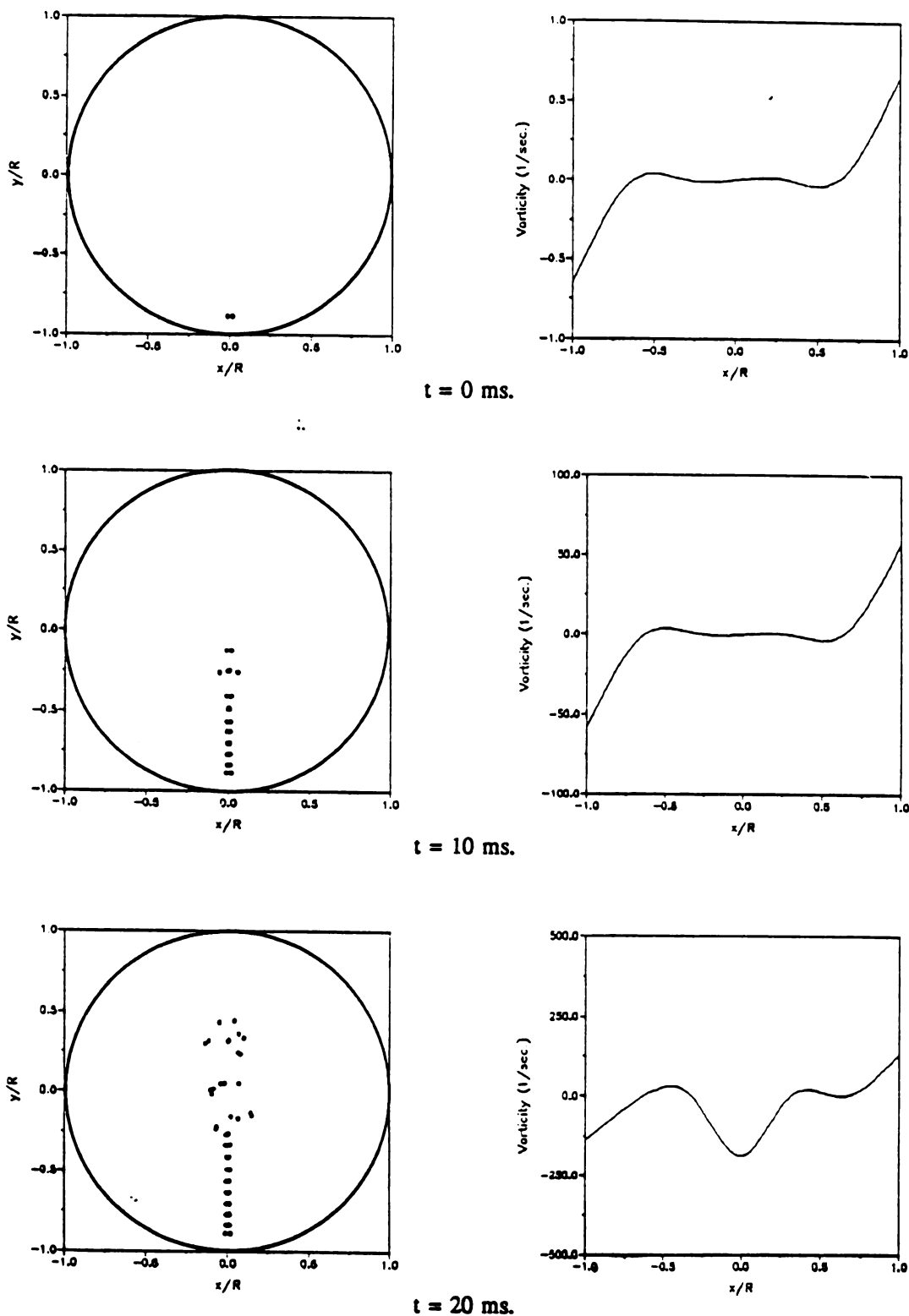
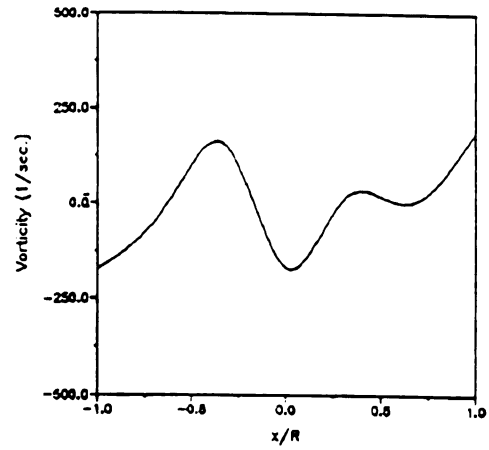
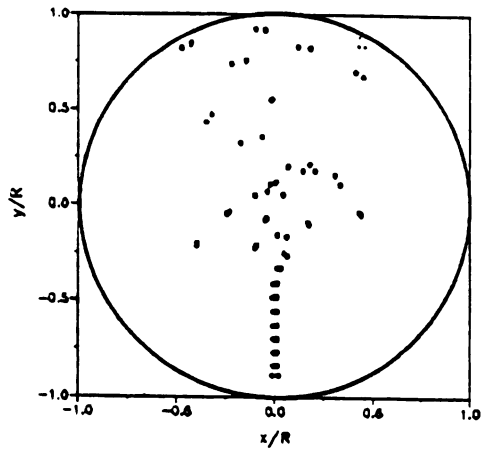
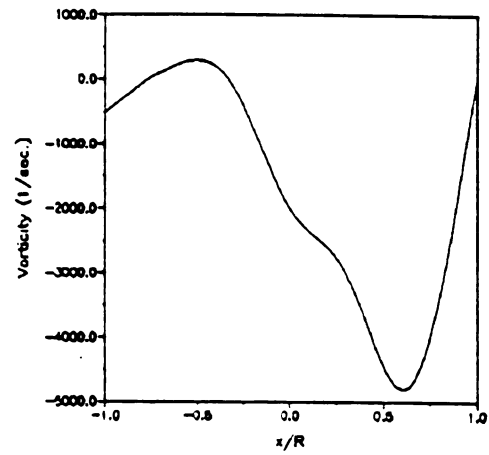
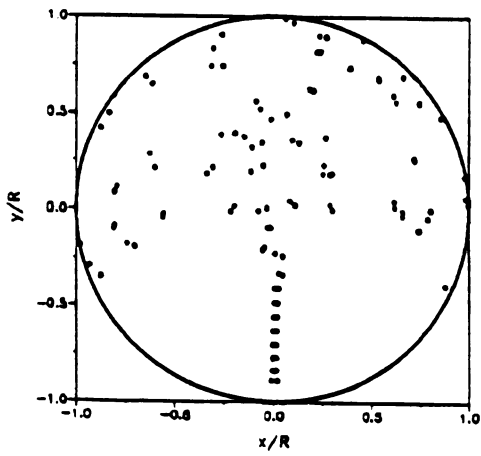


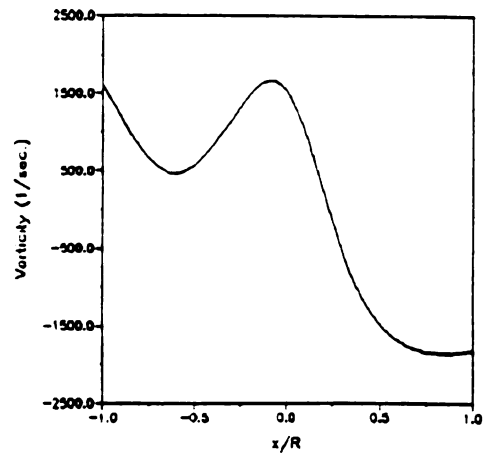
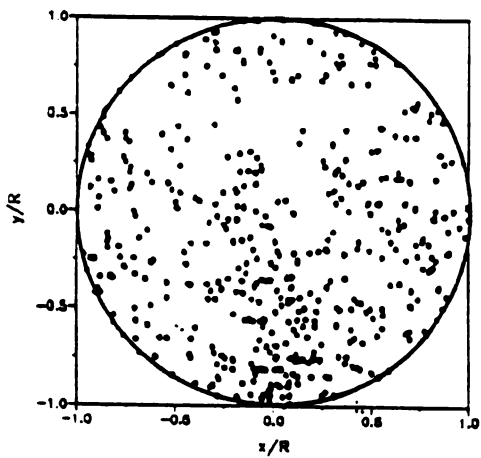
Figure 34. Fuel particle trajectory and vorticity profile at $y=0$ as a function of time. Injector at 90 degrees (counterclockwise from horizontal). $U_{in} = 210.0$ m/s.



$t = 30$ ms.



$t = 53$ ms.



$t = 256$ ms.

Figure 34 (con'td).

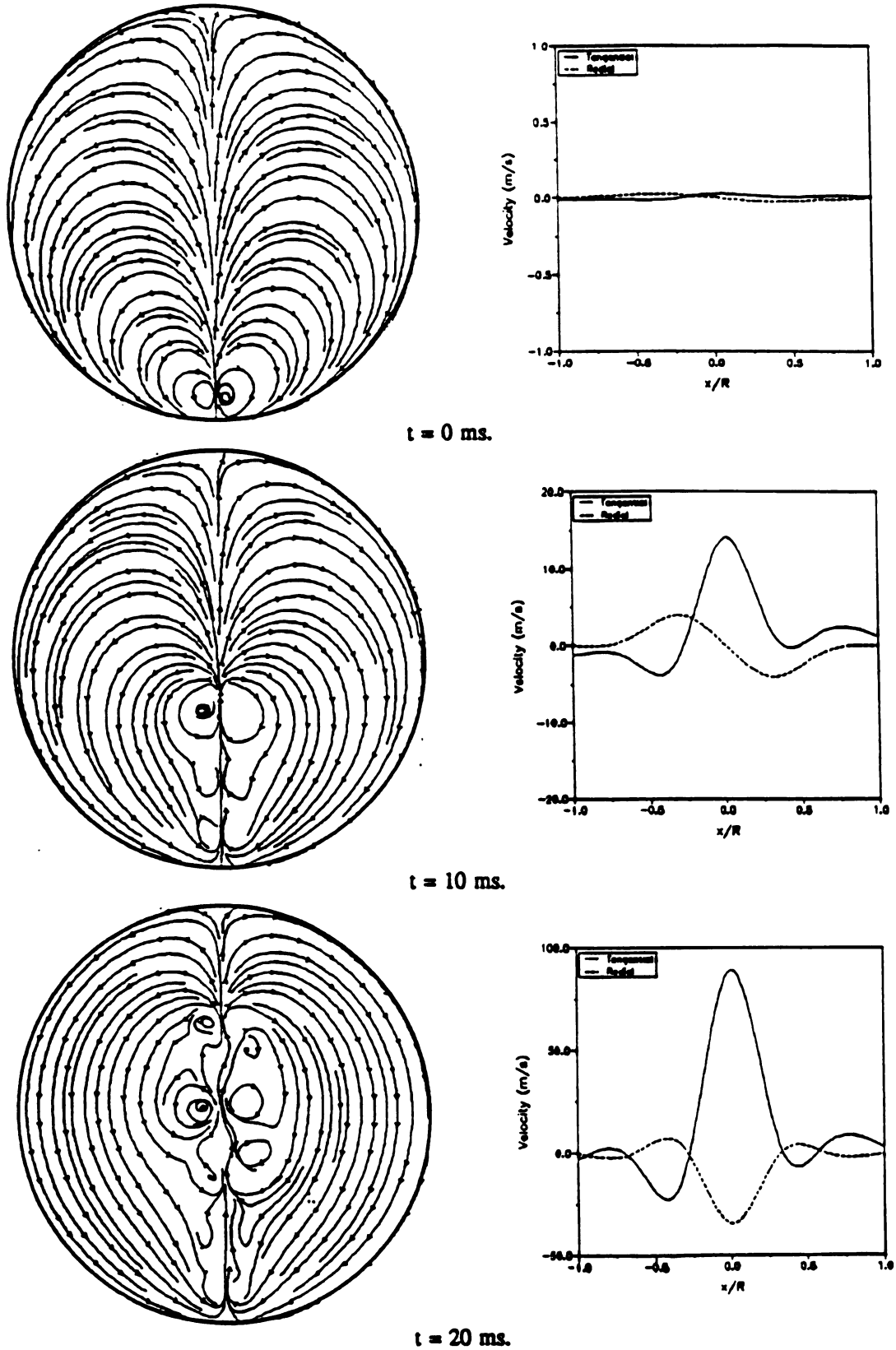
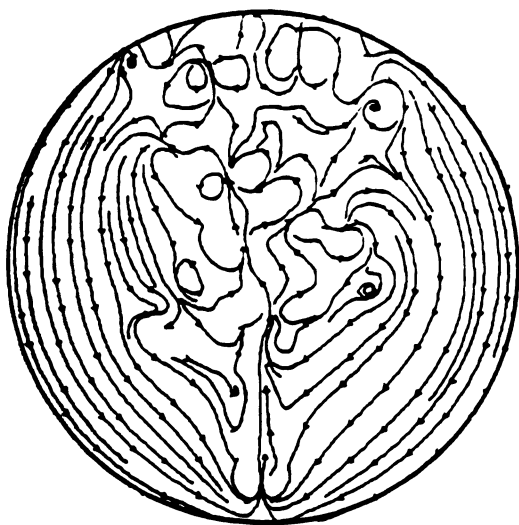
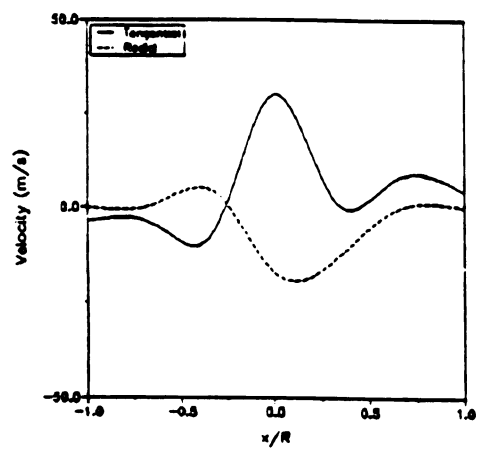


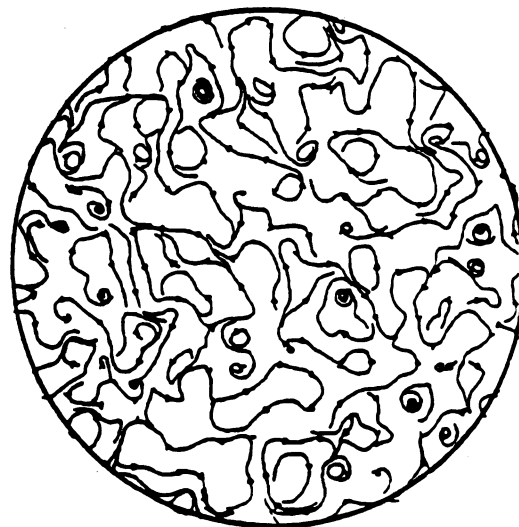
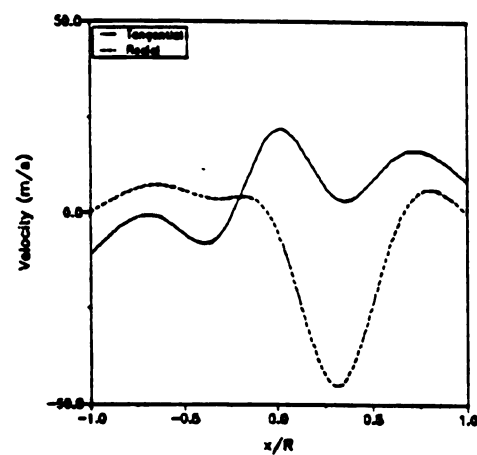
Figure 35. Streamlines and velocity profiles at $y=0$ as a function of time (without swirl). Injector at 90 degrees (counterclockwise from horizontal). $U_{in} = 210.0$ m/s.



$t = 30$ ms.



$t = 53$ ms.



$t = 256$ ms.

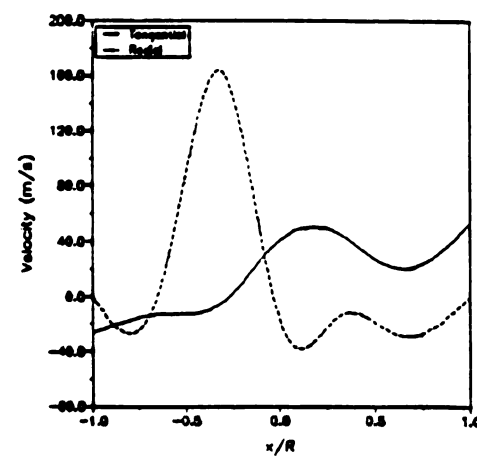


Figure 35 (con'td).

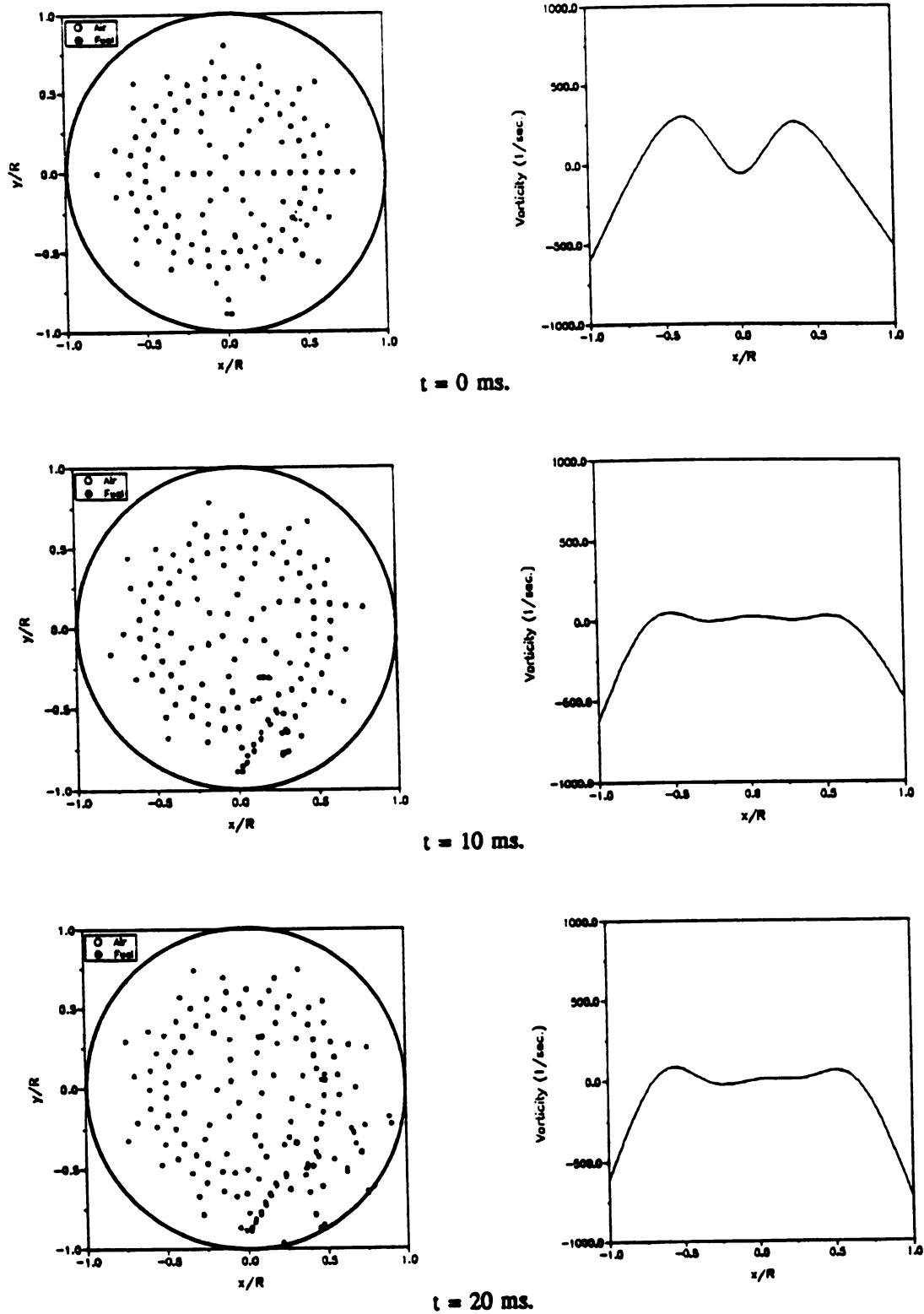
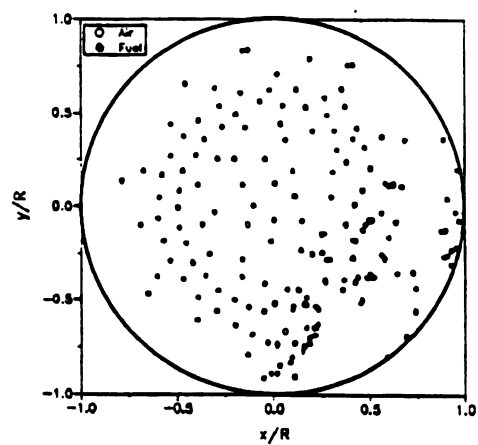
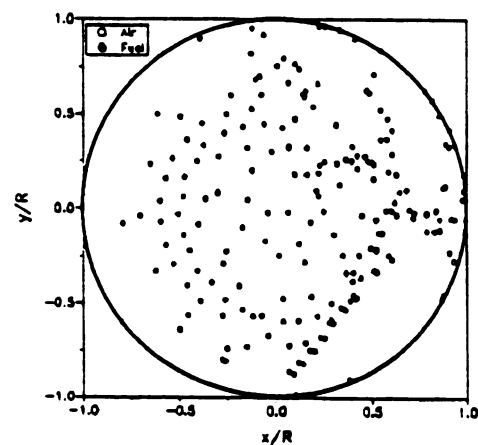
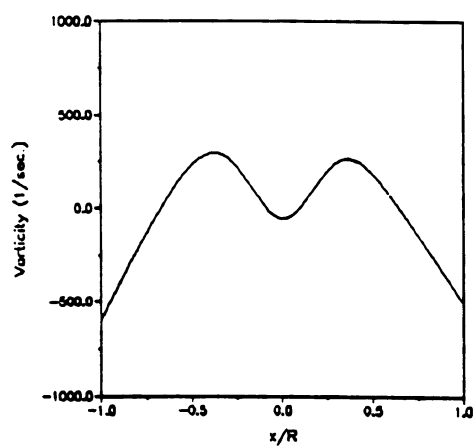


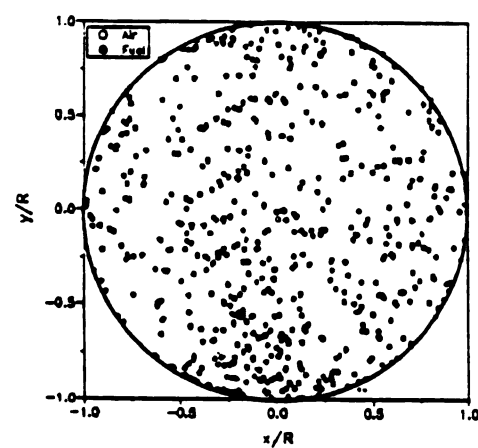
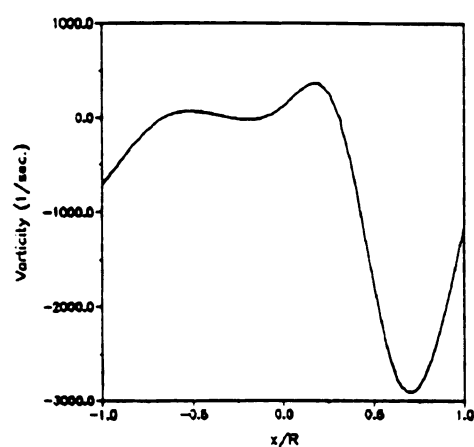
Figure 36. Air and Fuel particle trajectory and vorticity profile at $y=0$ as a function of time. Injector at 90 degrees (counterclockwise from horizontal). $U_{in} = 210.0$ m/s.



$t = 30$ ms.



$t = 40$ ms.



$t = 257$ ms.

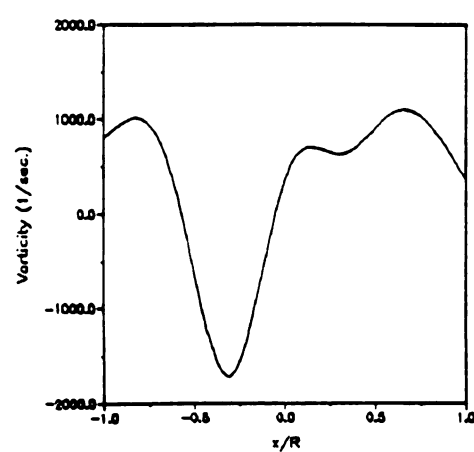
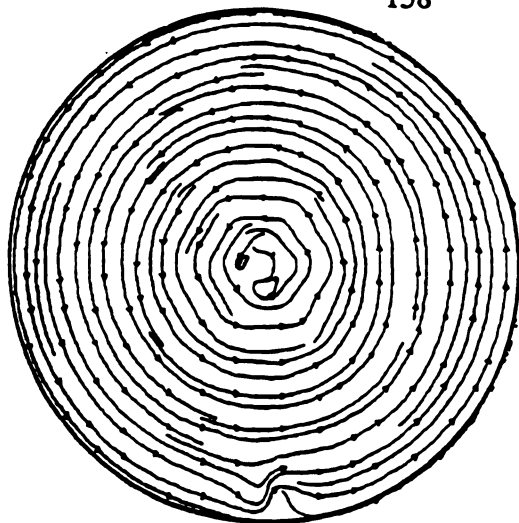
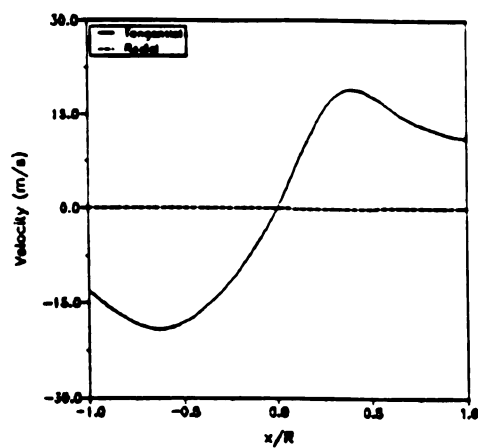


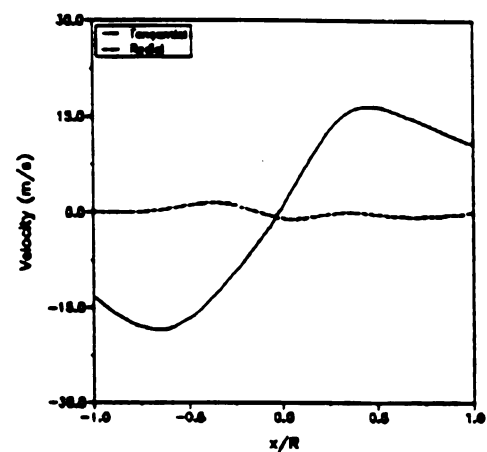
Figure 36 (con'td).



$t = 0$ ms.



$t = 10$ ms.



$t = 20$ ms.

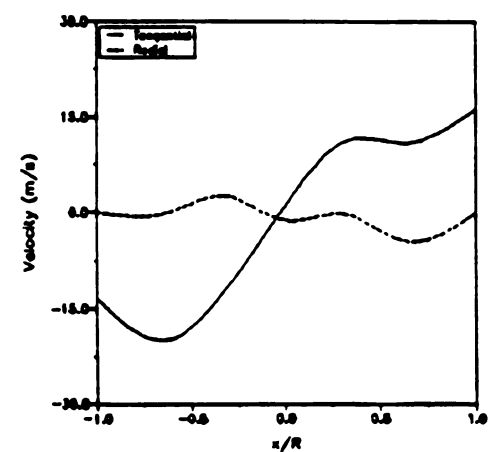
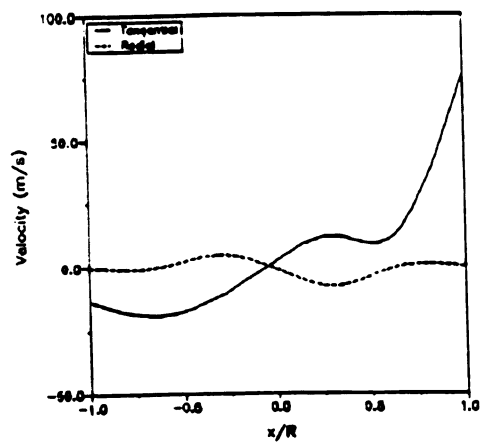


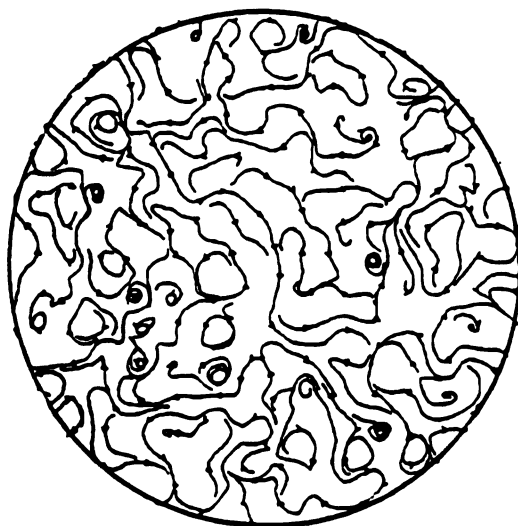
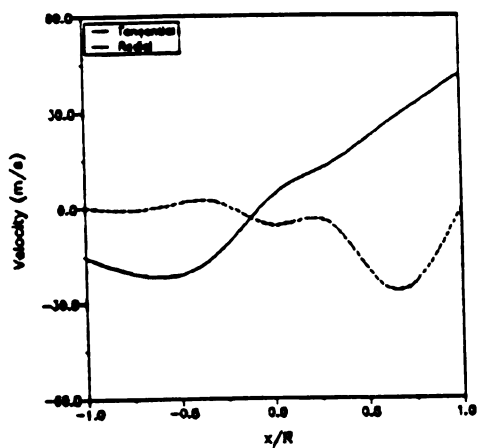
Figure 37. Streamlines and velocity profiles at $y=0$ as a function of time (with swirl). Injector at 90 degrees (counterclockwise from horizontal). $U_{in} = 210.0$ m/s.



$t = 30$ ms.



$t = 40$ ms.



$t = 257$ ms.

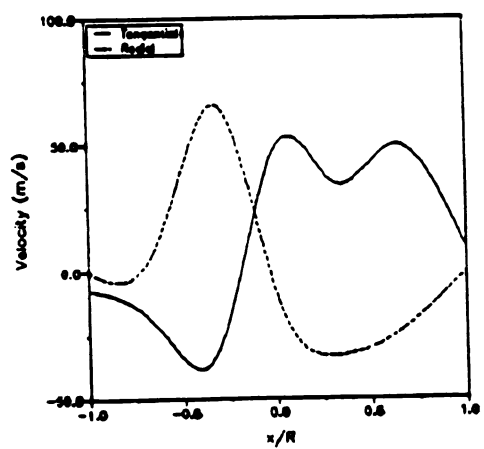


Figure 37 (con'td).

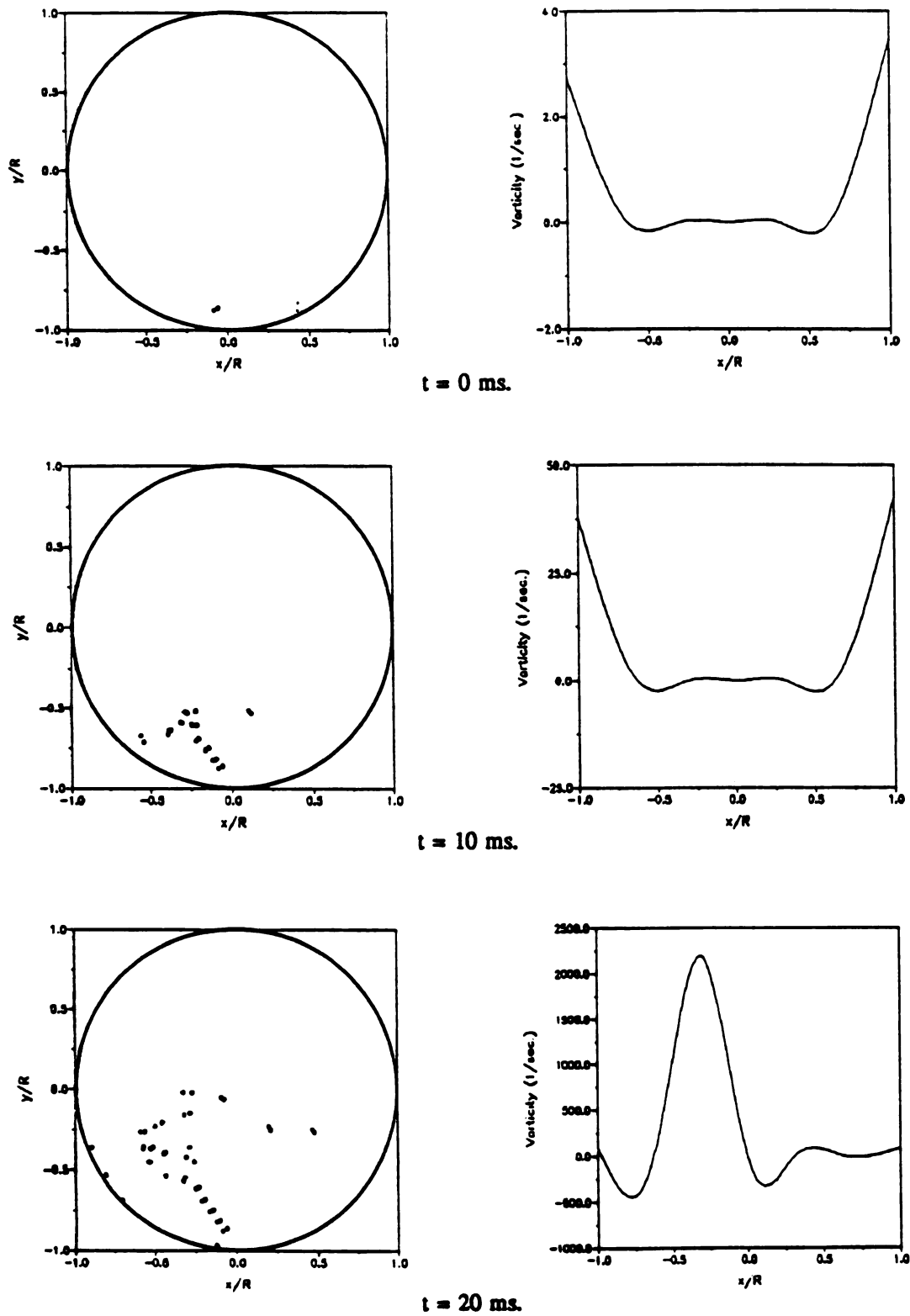
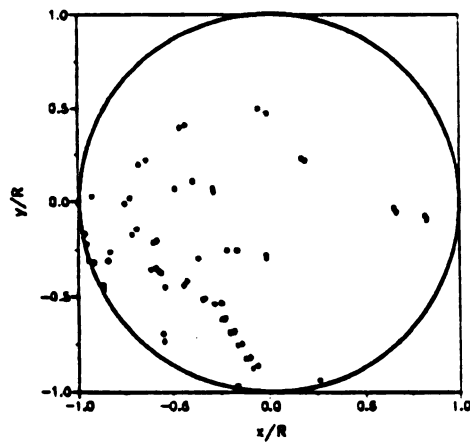
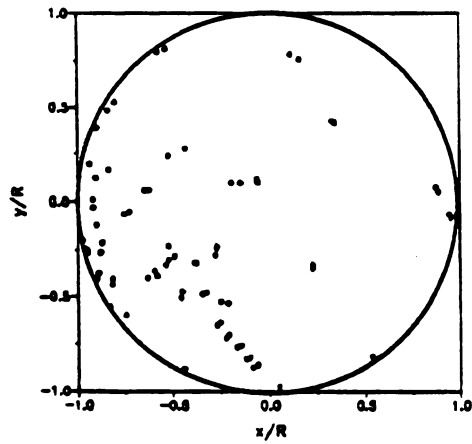
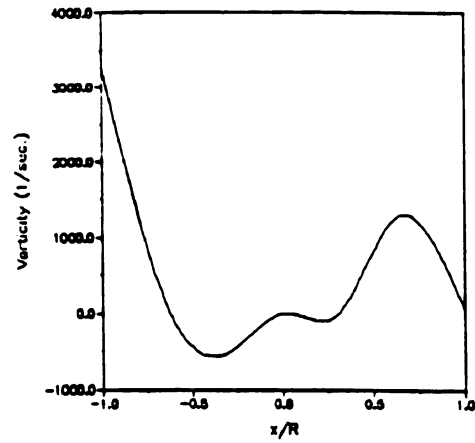


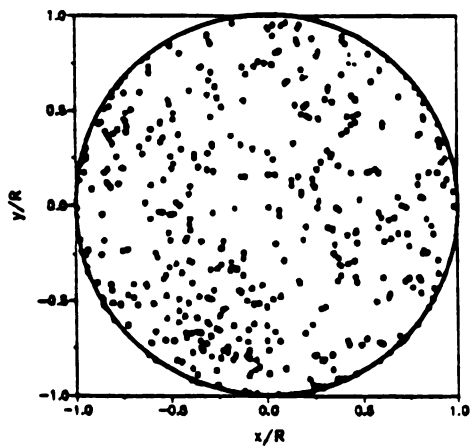
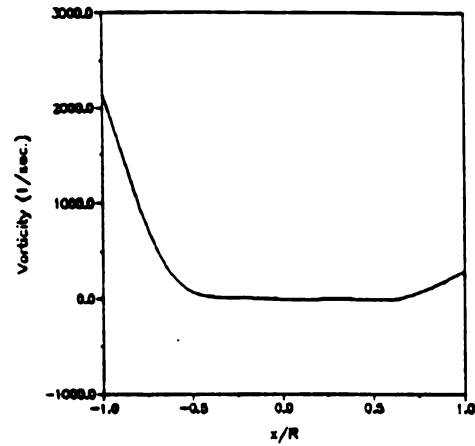
Figure 38. Fuel particle trajectory and vorticity profile at $y=0$ as a function of time. Injector at 120 degrees (counterclockwise from horizontal). $U_{in} = 210.0$ m/s.



$t = 30$ ms.



$t = 36$ ms.



$t = 259$ ms.

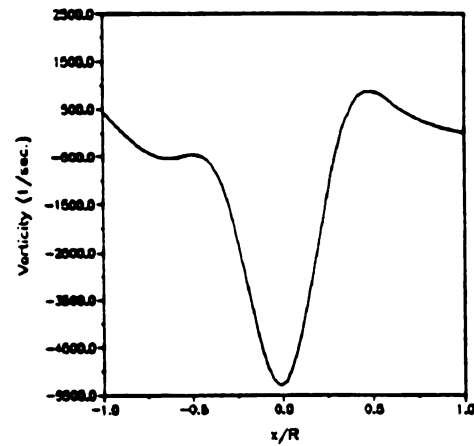
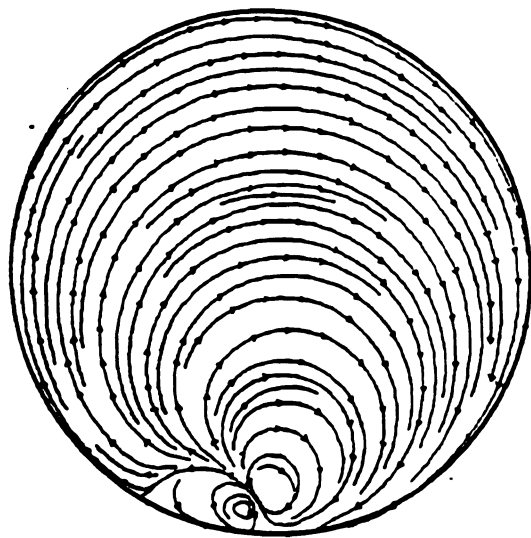
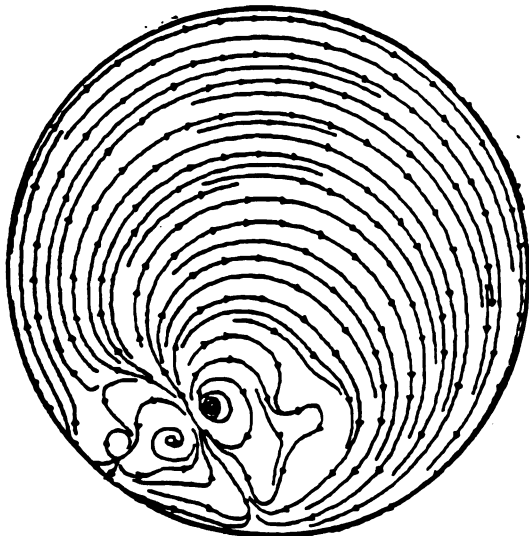
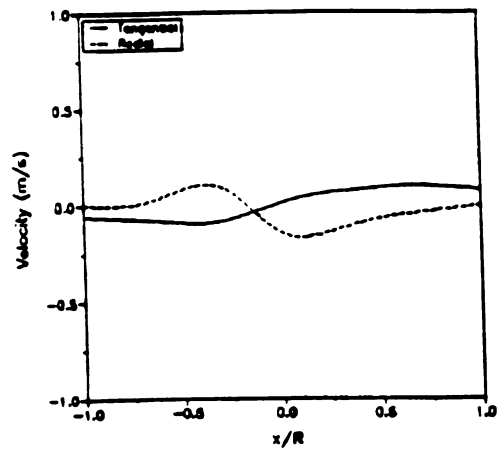


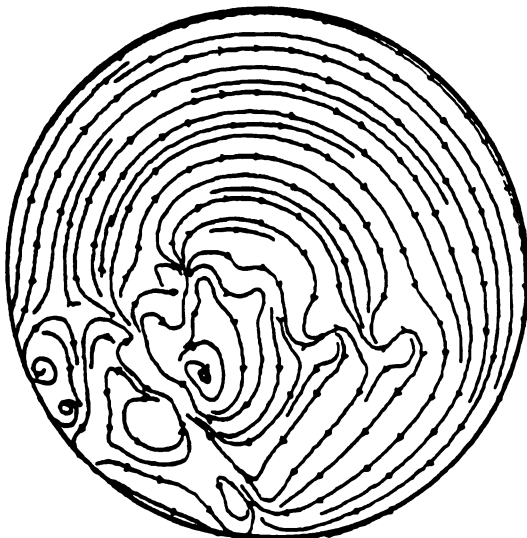
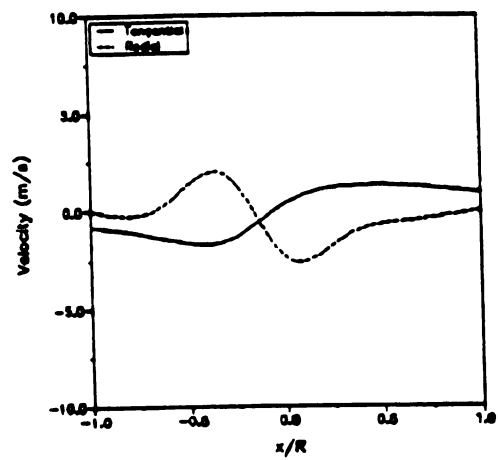
Figure 38 (con'td).



$t = 0$ ms.



$t = 10$ ms.



$t = 20$ ms.

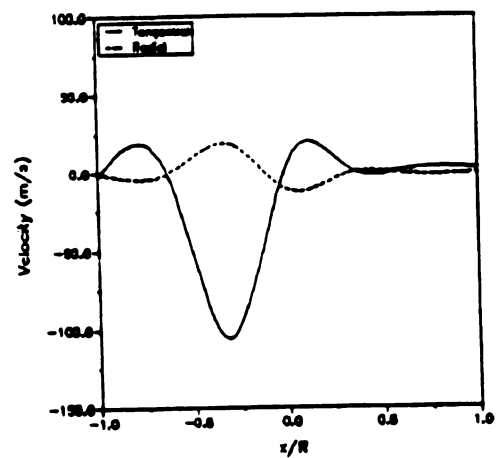
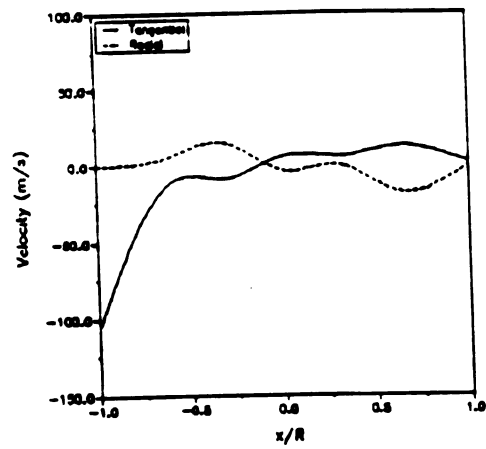


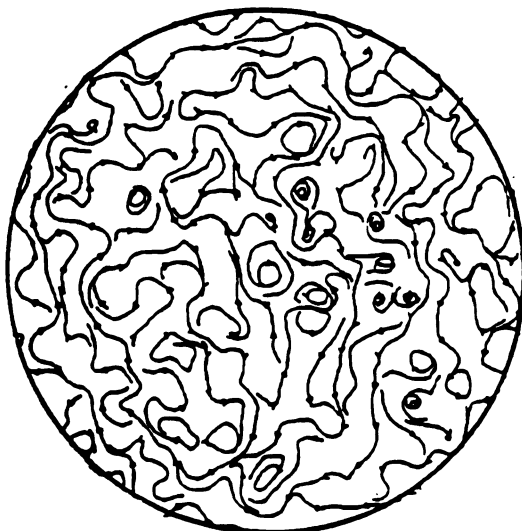
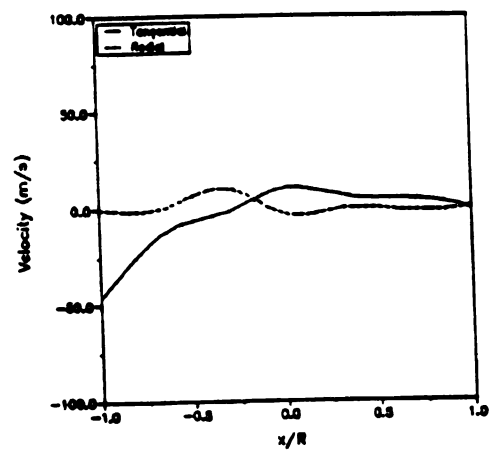
Figure 39. Streamlines and velocity profiles at $y=0$ as a function of time (without swirl). Injector at 120 degrees (counterclockwise from horizontal). $U_{in} = 210.0$ m/s.



$t = 30$ ms.



$t = 36$ ms.



$t = 259$ ms.

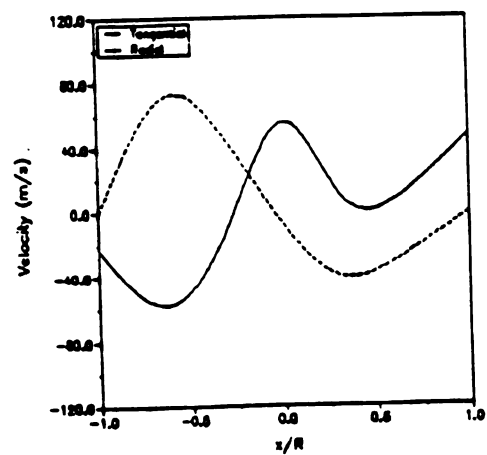


Figure 39 (con'td).

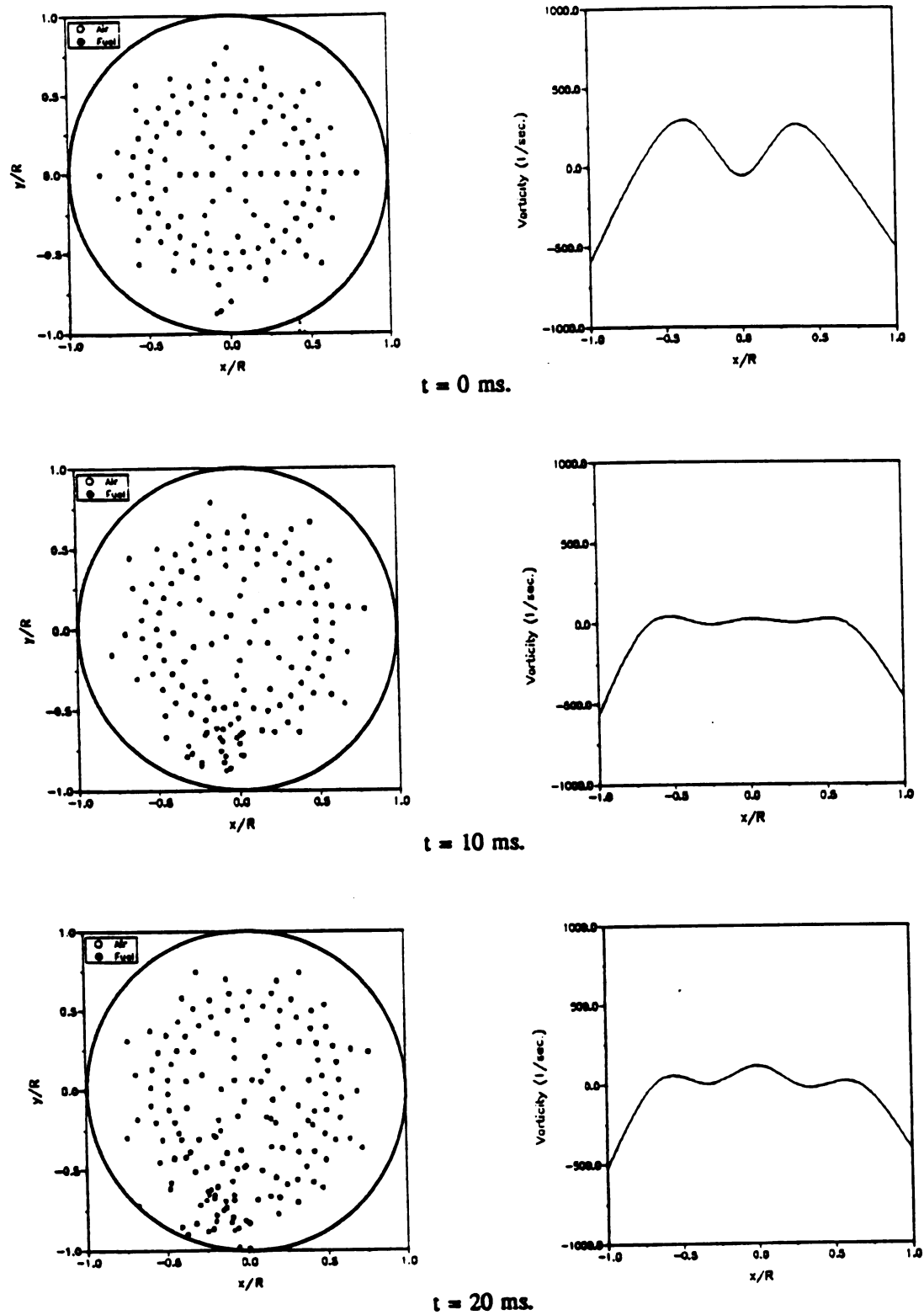
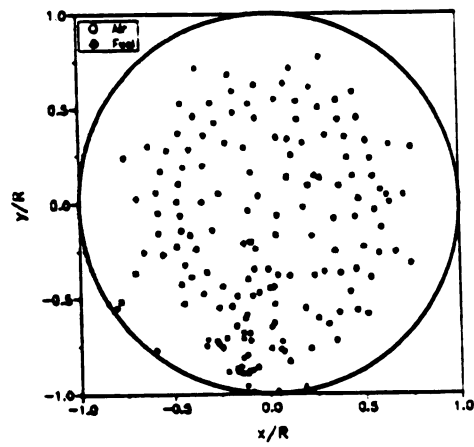
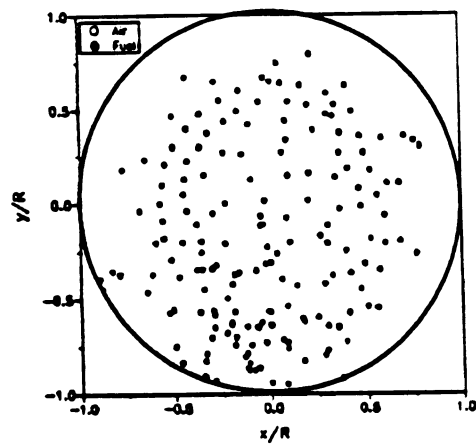
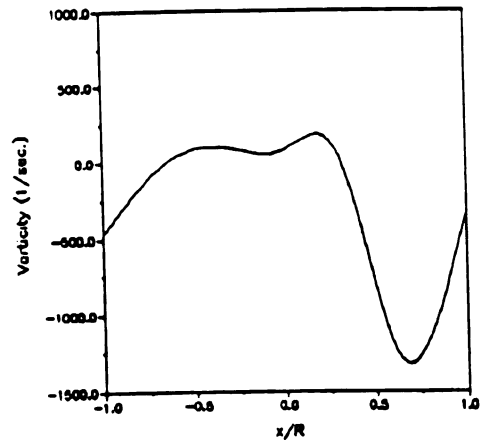


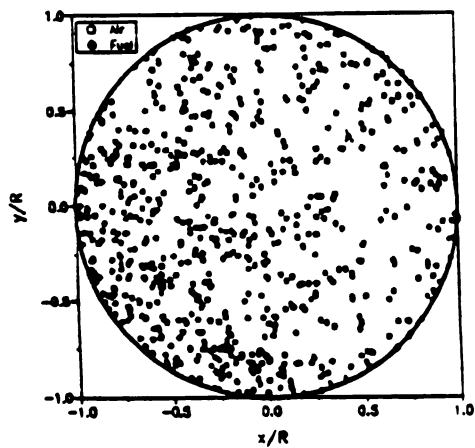
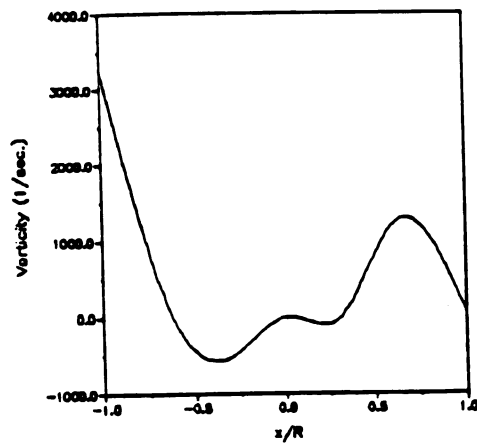
Figure 40. Air and Fuel particle trajectory and vorticity profile at $y=0$ as a function of time. Injector at 120 degrees (counterclockwise from horizontal). $U_{in} = 210.0$ m/s.



$t = 25$ ms.



$t = 30$ ms.



$t = 332$ ms.

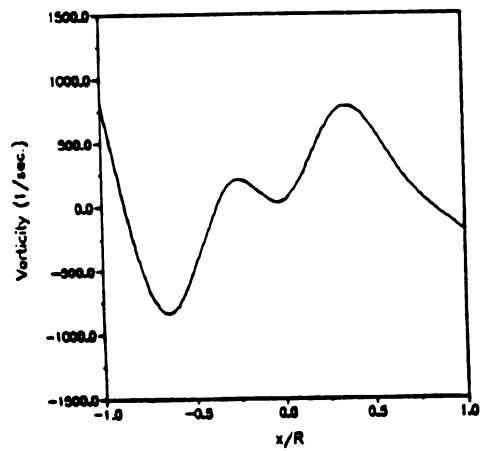
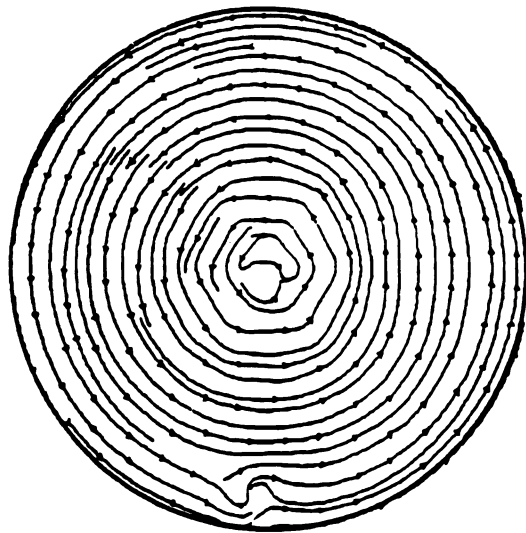
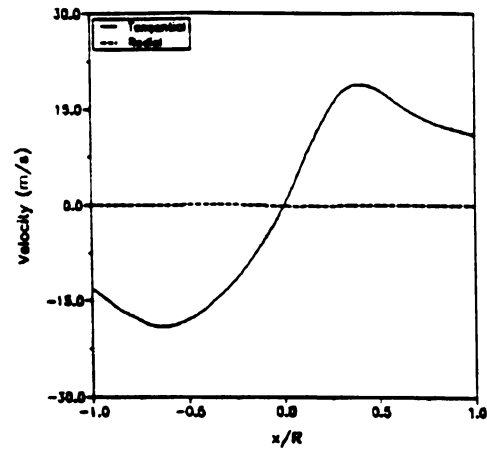


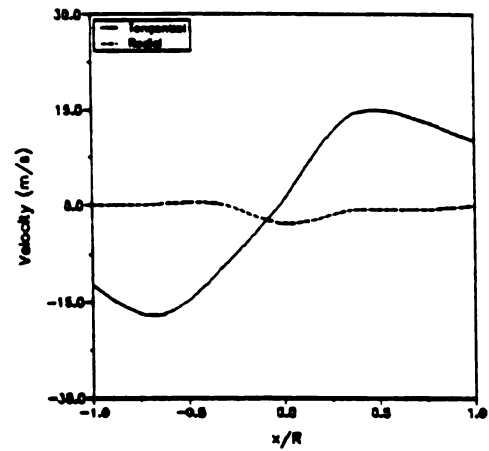
Figure 40 (con'td).



$t = 0$ ms.



$t = 10$ ms.



$t = 20$ ms.

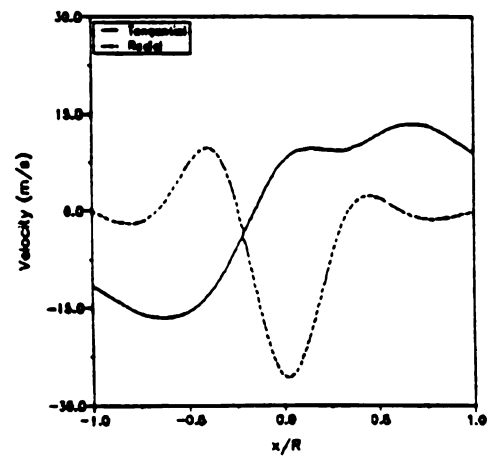
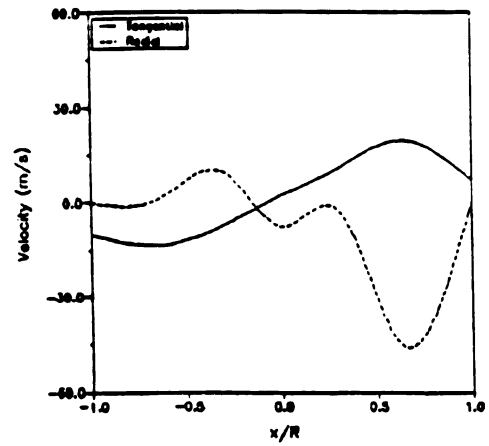


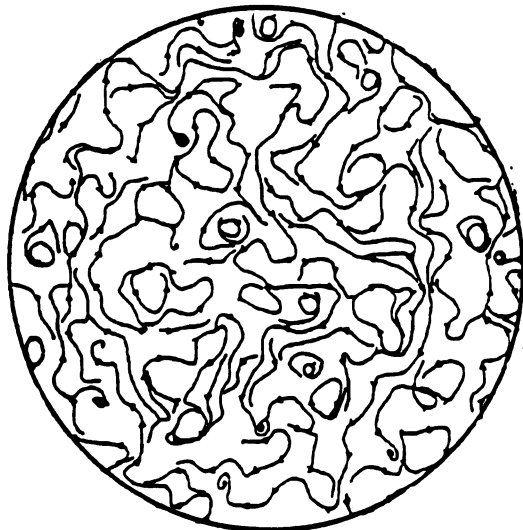
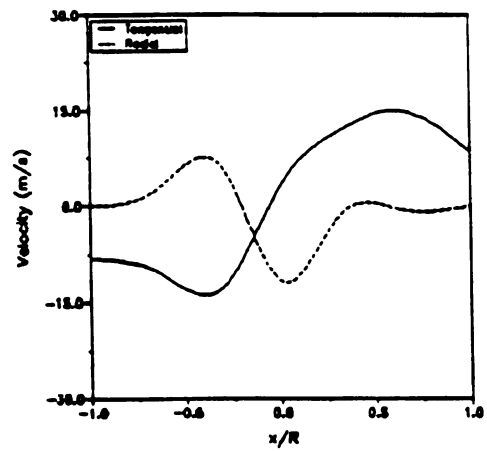
Figure 41. Streamlines and velocity profiles at $y=0$ as a function of time (with swirl). Injector at 120 degrees (counterclockwise from horizontal). $U_{in} = 210.0$ m/s.



$t = 25$ ms.



$t = 30$ ms.



$t = 332$ ms.

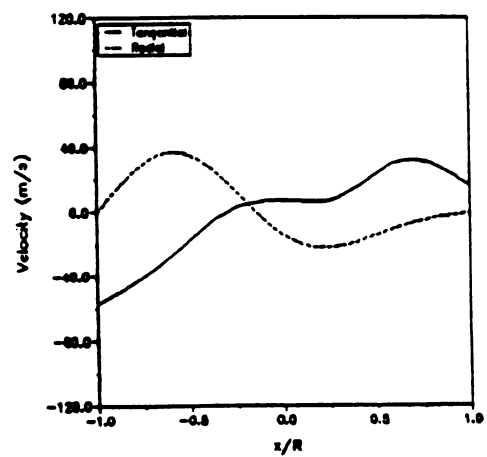


Figure 41 (con'td).

APPENDICES

APPENDIX A

APPENDIX A

NUMERICAL FILTERS

AI. Derivation of Christiansen's filter as an illustrative example

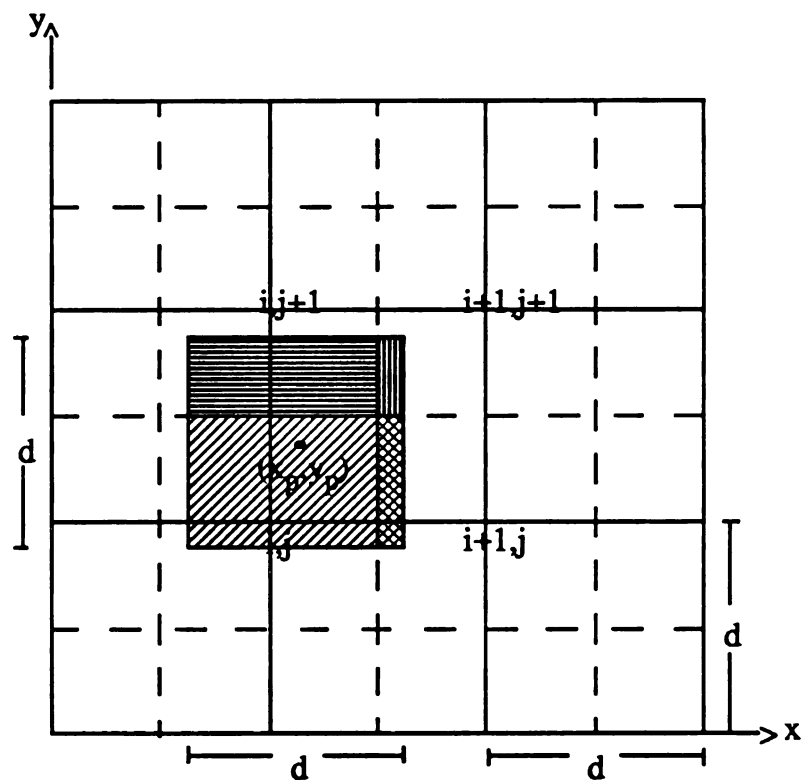


Figure 42. Vorticity distribution in a grid, with shading showing assignment of vorticity to grid point for VIC method.

-  is assigned to grid point (i,j)
-  is assigned to grid point $(i+1,j)$
-  is assigned to grid point $(i+1,j+1)$
-  is assigned to grid point $(i,j+1)$

Notes :

- (1) Each point vortex is assumed to possess uniform vorticity within a square cell with dimensions ($d \times d$).
- (2) Unit circulation ($\gamma_p=1.0$) is assumed for each vortex.
- (3) The point vortex coordinates (x_p, y_p) lie at the center of the square constant-vorticity core and must fall between four grid points.
- (4) The origin for (x_p, y_p) is the lower left hand grid point location.
- (5) The filter which credits the circulation to the surrounding four grid points is obtained through a bilinear interpolation scheme (area weighting).
- (6) A_{ij} , $(\gamma_p)_{ij}$, are the area and vortex circulation assigned to the grid point (i,j) respectively; $\xi(x,y)$ is the vorticity and (x_g, y_g) are the grid point coordinates.

$$\gamma_p = \int \xi(x,y) dA = \xi A = 1.0 \quad (A.1)$$

$$(\gamma_p)_{ij} = \int \xi(x,y) dA_{ij} = \xi A_{ij} \quad (A.2)$$

Dividing equation (A.2) by (A.1) :

$$\frac{(\gamma_p)_{ij}}{\gamma_p} = \frac{A_{ij}}{A} \quad (A.3)$$

$$(\gamma_p)_{ij} = \left(\frac{A_{ij}}{A} \right) \gamma_p \quad (A.4)$$

$$\begin{aligned}
A_{ij} &= [(x_g+d/2)-(x_p-d/2)] \cdot [(y_g+d/2)-(y_p-d/2)] \\
&= [x_g+d/2-x_p+d/2] \cdot [y_g+d/2-y_p+d/2] \\
&= [x_g-x_p+d] \cdot [y_g-y_p+d]
\end{aligned} \tag{A.5}$$

$$\begin{aligned}
\frac{A_{ij}}{A} &= \frac{[x_g-x_p+d] \cdot [y_g-y_p+d]}{d \cdot d} \\
&= [(x_g-x_p)/d+1] \cdot [(y_g-y_p)/d+1] \\
&= [1-(x_p-x_g)/d] \cdot [1-(y_p-y_g)/d] \\
&= [1-\eta(x)] \cdot [1-\eta(y)]
\end{aligned} \tag{A.6}$$

$$\begin{aligned}
A_{i+1j} &= [(x_p+d/2)-(x_g+d/2)] \cdot [(y_g+d/2)-(y_p-d/2)] \\
&= [x_p+d/2-x_g-d/2] \cdot [y_g+d/2-y_p+d/2] \\
&= [x_p-x_g] \cdot [y_g-y_p+d]
\end{aligned} \tag{A.7}$$

$$\begin{aligned}
\frac{A_{i+1j}}{A} &= \frac{[x_p-x_g] \cdot [y_g-y_p+d]}{d \cdot d} \\
&= [(x_p-x_g)/d] \cdot [(y_g-y_p)/d+1] \\
&= [(x_p-x_g)/d] \cdot [1-(y_p-y_g)/d] \\
&= [\eta(x)] \cdot [1-\eta(y)]
\end{aligned} \tag{A.8}$$

$$\begin{aligned}
A_{i+1,j+1} &= [(x_p+d/2)-(x_g+d/2)] \cdot [(y_p+d/2)-(y_g+d/2)] \\
&= [x_p+d/2-x_g-d/2] \cdot [y_p+d/2-y_g-d/2] \\
&= [x_p-x_g] \cdot [y_p-y_g]
\end{aligned} \tag{A.9}$$

$$\begin{aligned}
\frac{A_{i+1,j+1}}{A} &= \frac{[x_p-x_g] \cdot [y_p-y_g]}{d \cdot d} \\
&= [(x_p-x_g)/d] \cdot [(y_p-y_g)/d] \\
&= \eta(x) \cdot \eta(y)
\end{aligned} \tag{A.10}$$

$$\begin{aligned}
A_{i,j+1} &= [(x_g+d/2)-(x_p-d/2)] \cdot [(y_p+d/2)-(y_g+d/2)] \\
&= [x_g+d/2-x_p+d/2] \cdot [y_p+d/2-y_g-d/2] \\
&= [x_g-x_p+d] \cdot [y_p-y_g]
\end{aligned} \tag{A.11}$$

$$\begin{aligned}
\frac{A_{i,j+1}}{A} &= \frac{[x_g-x_p+d] \cdot [y_p-y_g]}{d \cdot d} \\
&= [(x_g-x_p)/d+1] \cdot [(y_p-y_g)/d] \\
&= [1-(x_p-x_g)/d] \cdot [(y_p-y_g)/d] \\
&= [1-\eta(x)] \cdot [\eta(y)]
\end{aligned} \tag{A.12}$$

We can generalize equations (A.6), (A.8), (A.10) and (A.12) by letting (x_g, y_g) be the coordinates of either grid point (i, j) or $(i+1, j)$. The results are :

$$\phi_{ij}[\eta(x)] = \begin{cases} 1 - |\eta(x)| & ; \quad |\eta(x)| \leq 1.0 \\ 0 & ; \quad \text{otherwise.} \end{cases} \quad (\text{A.13})$$

$$\phi_{ij}[\eta(y)] = \begin{cases} 1 - |\eta(y)| & ; \quad |\eta(y)| \leq 1.0 \\ 0 & ; \quad \text{otherwise.} \end{cases} \quad (\text{A.15})$$

$$\phi_{ij} = \phi_{ij}[\eta(x), \eta(y)] = \phi_{ij}[\eta(x)] \cdot \phi_{ij}[\eta(y)] \quad (\text{A.17})$$

$$\text{and from equation (A.4) : } (\gamma_p)_{ij} = \phi_{ij} \gamma_p \quad (\text{A.18})$$

APPENDIX B

APPENDIX B

FUNCTIONS

BI. Smoothing function G_{ij} for convergence of vortex method [39]

$$W(z) = -i\kappa_j G_{ij} \quad (B.1)$$

$$= -i\kappa_j \left[C_1 \left(1 - \frac{|z-z_j|^3}{r_0^3} \right) - C_2 \left(1 - \frac{|z-z_j|^2}{r_0^2} \right) + \log r_0 \right] \quad ; |z-z_j| \leq r_0$$

$$= -i\kappa_j \left[\frac{|z-z_j|^2}{r_0^2} \left(C_2 - \frac{C_1 |z-z_j|}{r_0} \right) + (C_1 - C_2 + \log r_0) \right] \quad (B.2)$$

$$[u-iv]_j = \frac{dW(z)}{dz} \quad (B.3)$$

$$= -i\kappa_j \left[\frac{|z-z_j|^2}{r_0^2} \left(-\frac{C_1 \overline{(z-z_j)}}{r_0 |z_j - z_j|} \right) + \left(C_2 - \frac{C_1 |z-z_j|}{r_0} \right) \left(\frac{2\overline{(z-z_j)}}{r_0^2} \right) \right]$$

$$= -i\kappa_j \left[\frac{2C_2 \overline{(z-z_j)}}{r_0^2} - \frac{3C_1 \overline{(z-z_j)} |z-z_j|}{r_0^3} \right]$$

$$= -i\kappa_j \left[\frac{2C_2}{r_0^2} - \frac{3C_1 |z-z_j|}{r_0^3} \right] \overline{(z-z_j)} \quad ; |z-z_j| \leq r_0 \quad (B.4)$$

where $C_1=2/3$ and $C_2=3/2$.

ВІІ. Derivation of the stream function, ψ , inside a finite rectangular domain using method of images

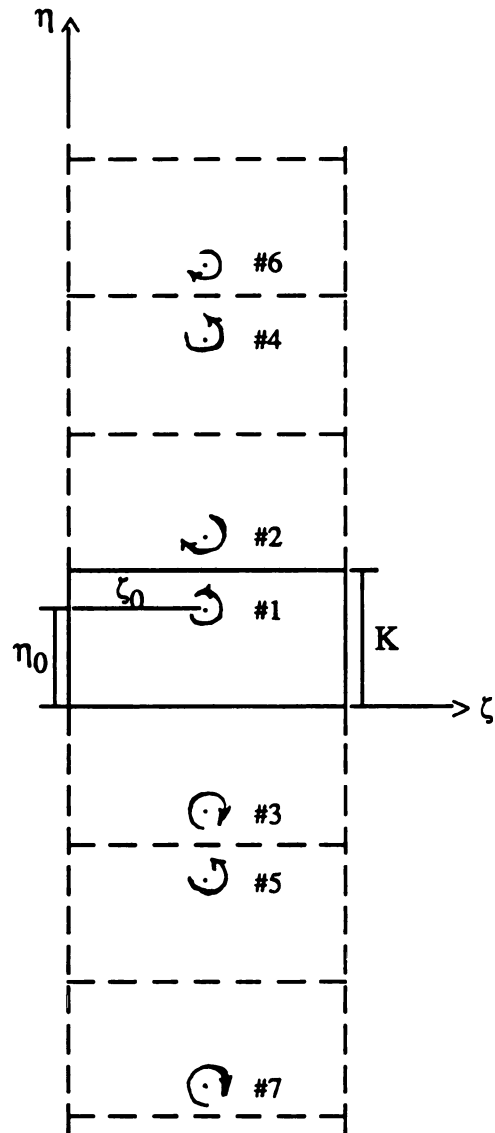


Figure 43. Method of Images

Notes : (i) Sign Convention :

Counter clockwise is positive, clockwise is negative.

$$(ii) \quad \omega = \zeta + i\eta \quad ; \quad \bar{\omega} = \zeta - i\eta \quad (B.5)$$

$$\omega_0 = \zeta_0 + i\eta_0 \quad ; \quad \bar{\omega}_0 = \zeta_0 - i\eta_0 \quad (B.6)$$

Consider the η -direction,

$$W(\omega) = -i\kappa \left\{ \begin{array}{l} \text{\#1} \quad \text{\#2} \quad \text{\#3} \\ \ln[\omega - (\zeta_0 + i\eta_0)] - \ln[\omega - (\zeta_0 + i(2K - \eta_0))] - \ln[\omega - (\zeta_0 - i\eta_0)] + \\ \text{\#4} \quad \text{\#5} \\ \ln[\omega - (\zeta_0 + i(2K + \eta_0))] + \ln[\omega - (\zeta_0 - i(2K - \eta_0))] - \\ \text{\#6} \quad \text{\#7} \\ \ln[\omega - (\zeta_0 + i(4K - \eta_0))] - \ln[\omega - (\zeta_0 - i(2K + \eta_0))] + \dots \end{array} \right\}$$

$$= -i\kappa \left\{ \begin{array}{l} \ln(\omega - \omega_0) - \ln(\omega - \bar{\omega}_0 - i2K) - \ln(\omega - \bar{\omega}_0) + \ln(\omega - \omega_0 - i2K) + \\ \ln(\omega - \omega_0 + i2K) - \ln(\omega - \omega_0 - i4K) - \ln(\omega - \bar{\omega}_0 - i2K) + \dots \end{array} \right\}$$

$$= -i\kappa \left\{ \begin{array}{l} [\ln(\omega - \omega_0) + \ln(\omega - \omega_0 - i2K) + \ln(\omega - \omega_0 + i2K) + \dots] - \\ [\ln(\omega - \bar{\omega}_0) - \ln(\omega - \bar{\omega}_0 - i2K) - \ln(\omega - \bar{\omega}_0 + i2K) + \dots] \end{array} \right\}$$

$$= -i\kappa \left\{ \sum_{n=-\infty}^{\infty} \ln[\omega - \omega_0 + i2nK] - \sum_{n=-\infty}^{\infty} \ln[\omega - \bar{\omega}_0 + i2nK] \right\}$$

$$= f(\omega) + \overline{f(\omega)} \quad (B.7)$$

where
$$f(\omega) = -i\kappa \sum_{n=-\infty}^{\infty} \ln [\omega - \omega_0 + i2nK] \quad (B.8)$$

$$\overline{f(\omega)} = i\kappa \sum_{n=-\infty}^{\infty} \ln [\omega - \bar{\omega}_0 + i2nK] \quad (B.9)$$

Now, consider $f(\omega)$:

$$\begin{aligned} f(\omega) &= -i\kappa \ln \{ (\omega - \omega_0)[(\omega - \omega_0) + i2K][(\omega - \omega_0) - i2K][(\omega - \omega_0) + i4K] + \dots \} \\ &= -i\kappa \ln \{ (\omega - \omega_0) \prod_{n=1}^{\infty} [(\omega - \omega_0)^2 + (2nK)^2] \} \\ &= -i\kappa \ln \left[\frac{\pi(\omega - \omega_0)}{2K} \cdot \frac{2K}{\pi} \prod_{n=1}^{\infty} \left[1 + \left\{ \frac{(\omega - \omega_0)}{2nK} \right\}^2 \right] (2nK)^2 \right] \\ &= -i\kappa \left\{ \ln \left[\frac{\pi(\omega - \omega_0)}{2K} \prod_{n=1}^{\infty} \left[1 + \left\{ \frac{(\omega - \omega_0)}{2nK} \right\}^2 \right] \right] + \ln \left[\frac{2K}{\pi} \prod_{n=1}^{\infty} (2nK)^2 \right] \right\} \\ &= -i\kappa \left\{ \ln \left[\text{Sinh} \left\{ \frac{\pi(\omega - \omega_0)}{2K} \right\} \right] + \ln \left[\frac{2K}{\pi} \prod_{n=1}^{\infty} (2nK)^2 \right] \right\} \\ &= -i\kappa \ln \left[\text{Sinh} \left\{ \frac{\pi(\omega - \omega_0)}{2K} \right\} \right] + C_1 \end{aligned} \quad (B.10)$$

where
$$\text{Sinh} \left\{ \frac{\pi(\omega - \omega_0)}{2K} \right\} = \left[\frac{\pi(\omega - \omega_0)}{2K} \prod_{n=1}^{\infty} \left[1 + \left\{ \frac{(\omega - \omega_0)}{2nK} \right\}^2 \right] \right] \quad (B.11)$$

$$C_1 = \ln \left[\frac{2K}{\pi} \prod_{n=1}^{\infty} (2nK)^2 \right] \quad (B.12)$$

Similarly, it can be shown that :

$$\overline{f(\omega)} = i\kappa \ln \left[\text{Sinh} \left\{ \frac{\pi(\omega - \bar{\omega}_0)}{2K} \right\} \right] - C_1 \quad (\text{B.13})$$

Substituting (B.10) and (B.13) in (B.7) :

$$\begin{aligned} W(\omega) &= -i\kappa \left\{ \ln \left[\text{Sinh} \left\{ \frac{\pi(\omega - \omega_0)}{2K} \right\} \right] - \ln \left[\text{Sinh} \left\{ \frac{\pi(\omega - \bar{\omega}_0)}{2K} \right\} \right] \right\} \\ &= -i\kappa \ln \left\{ \frac{\text{Sinh} \left\{ \frac{\pi(\omega - \omega_0)}{2K} \right\}}{\text{Sinh} \left\{ \frac{\pi(\omega - \bar{\omega}_0)}{2K} \right\}} \right\} \\ &= -i\kappa \ln \left\{ \frac{\text{Sinh}[\zeta_m + i\eta_m]}{\text{Sinh}[\zeta_m + i\eta_p]} \right\} \\ &= -i\kappa \ln \left\{ \frac{\text{Sinh}(\zeta_m)\text{Cos}(\eta_m) + i\text{Cosh}(\zeta_m)\text{Sin}(\eta_m)}{\text{Sinh}(\zeta_m)\text{Cos}(\eta_p) + i\text{Cosh}(\zeta_m)\text{Sin}(\eta_p)} \right\} \end{aligned} \quad (\text{B.14})$$

where $\zeta_m = \frac{\pi}{2K}(\zeta - \zeta_0)$, $\eta_m = \frac{\pi}{2K}(\eta - \eta_0)$ and $\eta_p = \frac{\pi}{2K}(\eta + \eta_0)$.

Similarly,

$$\overline{W(\bar{\omega})} = i\kappa \ln \left\{ \frac{\text{Sinh}(\zeta_m)\text{Cos}(\eta_m) - i\text{Cosh}(\zeta_m)\text{Sin}(\eta_m)}{\text{Sinh}(\zeta_m)\text{Cos}(\eta_p) - i\text{Cosh}(\zeta_m)\text{Sin}(\eta_p)} \right\} \quad (\text{B.15})$$

From $W(\omega) = \phi + i\psi$

and $\overline{W(\omega)} = \phi - i\psi$

$$\psi = \frac{W(\omega) - \overline{W(\omega)}}{2i} \quad (B.16)$$

Substituting (B.14) and (B.15) in (B.16) :

$$\psi = \frac{-i\kappa \ln \left\{ \frac{\left[\frac{\sinh(\zeta_m) \cos(\eta_m) + i \cosh(\zeta_m) \sin(\eta_m)}{\sinh(\zeta_m) \cos(\eta_p) + i \cosh(\zeta_m) \sin(\eta_p)} \right]}{\left[\frac{\sinh(\zeta_m) \cos(\eta_m) - i \cosh(\zeta_m) \sin(\eta_m)}{\sinh(\zeta_m) \cos(\eta_p) - i \cosh(\zeta_m) \sin(\eta_p)} \right]} \right\}}{2i}$$

$$\psi = -\frac{\kappa}{2} \ln \left[\frac{\sinh^2(\zeta_m) \cos^2(\eta_m) + \cosh^2(\zeta_m) \sin^2(\eta_m)}{\sinh^2(\zeta_m) \cos^2(\eta_p) + \cosh^2(\zeta_m) \sin^2(\eta_p)} \right]$$

$$\psi = -\frac{\kappa}{2} \ln \left[\frac{\sinh^2(\zeta_m) + \sin^2(\eta_m)}{\sinh^2(\zeta_m) + \sin^2(\eta_p)} \right]$$

$$\psi = \frac{\kappa}{2} \ln \left[\frac{\sinh^2(\zeta_m) + \sin^2(\eta_p)}{\sinh^2(\zeta_m) + \sin^2(\eta_m)} \right] \quad (B.17)$$

APPENDIX C

APPENDIX C

COORDINATE TRANSFORMATION

CI. Square $(\zeta, \eta) \rightarrow$ Circle (x, y)

We begin with the transformation (5.1) (See Bowman [11]):

$$z = \frac{\text{sn}(\omega)\text{dn}(\omega)}{\text{cn}(\omega)} \quad (\text{C.1})$$

where

$$z = x + iy \quad (\text{C.2})$$

$$\omega = \zeta + i\eta \quad (\text{C.3})$$

$$\text{sn}(\omega) = [(\text{sd}_1 + i\text{cds}_1\text{c}_1)/(1 - d^2\text{s}_1^2)] \quad (\text{C.4})$$

$$\text{dn}(\omega) = [(\text{dc}_1\text{d}_1 - ik^2\text{scs}_1)/(1 - d^2\text{s}_1^2)] \quad (\text{C.5})$$

$$\text{cn}(\omega) = [(\text{cc}_1 - i\text{sd}_1\text{s}_1\text{d}_1)/(1 - d^2\text{s}_1^2)] \quad (\text{C.6})$$

$$c = \text{cn}(\zeta, k) \quad (\text{C.7})$$

$$c_1 = \text{cn}(\eta, k') \quad (\text{C.8})$$

$$d = \text{dn}(\zeta, k) \quad (\text{C.9})$$

$$d_1 = \text{dn}(\eta, k') \quad (\text{C.10})$$

$$s = \text{sn}(\zeta, k) \quad (\text{C.11})$$

$$s_1 = \text{sn}(\eta, k') \quad (\text{C.12})$$

$$k' = \sqrt{1 - k^2} \quad (\text{C.13})$$

Substituting equations (C.2), (C.4), (C.5) and (C.6) in (C.1) we get

$$\begin{aligned}
 x + iy &= \frac{[(sd_1 + icds_1c_1)/(1-d^2s_1^2)] [(dc_1d_1 - ik^2scs_1)/(1-d^2s_1^2)]}{[(cc_1 - isds_1d_1)/(1-d^2s_1^2)]} \quad (C.14) \\
 &= \frac{(sd_1 + icds_1c_1)(dc_1d_1 - ik^2scs_1)(cc_1 + isds_1d_1)}{(cc_1 - isds_1d_1)(1-d^2s_1^2)(cc_1 + isds_1d_1)} \\
 &= \frac{[sd_1^2dc_1 - ik^2s^2d_1cs_1 + icd^2s_1c_1^2d_1 + k^2c^2ds_1^2c_1s](cc_1 + isds_1d_1)}{(c^2c_1^2 + d^2d_1^2s^2s_1^2)(1-d^2s_1^2)} \\
 &= \frac{[c_1ds(d_1^2 + k^2c^2s_1^2) + icd_1s_1(c_1^2d^2 - k^2s^2)](cc_1 + isds_1d_1)}{(c^2c_1^2 + d^2d_1^2s^2s_1^2)(1-d^2s_1^2)} \\
 &= \frac{(cc_1^2ds + ic_1d^2d_1s^2s_1)(d_1^2 + k^2c^2s_1^2) + (ic^2c_1d_1s_1 - cd_1^2dss_1^2)(c_1^2d^2 - k^2s^2)}{(c^2c_1^2 + d^2d_1^2s^2s_1^2)(1-d^2s_1^2)} \\
 &= \frac{cds[c_1^2(d_1^2 + k^2c^2s_1^2) - d_1^2s_1^2(c_1^2d^2 - k^2s^2)] + ic_1d_1s_1[d^2s^2(d_1^2 + k^2c^2s_1^2) + c^2(c_1^2d^2 - k^2s^2)]}{(c^2c_1^2 + d^2d_1^2s^2s_1^2)(1-d^2s_1^2)} \\
 &= \frac{cds[c_1^2d_1^2(1-d^2s_1^2) + k^2s_1^2(c^2c_1^2 + d_1^2s^2)] + ic_1d_1s_1[d^2(d_1^2s^2 + c^2c_1^2) + k^2c^2s^2(d^2s_1 - 1)]}{(c^2c_1^2 + d^2d_1^2s^2s_1^2)(1-d^2s_1^2)} \\
 &= \left[\frac{cds[c_1^2d_1^2(1-d^2s_1^2) + k^2s_1^2(c^2c_1^2 + d_1^2s^2)]}{(c^2c_1^2 + d^2d_1^2s^2s_1^2)(1-d^2s_1^2)} \right] + \\
 &\quad i \left[\frac{c_1d_1s_1[d^2(d_1^2s^2 + c^2c_1^2) + k^2c^2s^2(d^2s_1 - 1)]}{(c^2c_1^2 + d^2d_1^2s^2s_1^2)(1-d^2s_1^2)} \right]
 \end{aligned}$$

Thus,

$$x = \left[\frac{cds[c_1^2 d_1^2 (1-d^2 s_1^2) + k^2 s_1^2 (c^2 c_1^2 + d_1^2 s^2)]}{(c^2 c_1^2 + d^2 d_1^2 s^2 s_1^2)(1-d^2 s_1^2)} \right] = f(\zeta, \eta, k) \quad (C.15)$$

$$y = \left[\frac{c_1 d_1 s_1 [d^2 (d_1^2 s^2 + c^2 c_1^2) + k^2 c^2 s^2 (d^2 s_1 - 1)]}{(c^2 c_1^2 + d^2 d_1^2 s^2 s_1^2)(1-d^2 s_1^2)} \right] = g(\zeta, \eta, k) \quad (C.16)$$

II. Circle (x,y) → Square (ζ,η)

X-Coordinate

Squaring both sides of equation (C.15) we get

$$x^2 = \left[\frac{cds[c_1^2 d_1^2 (1-d^2 s_1^2) + k^2 s_1^2 (c^2 c_1^2 + d_1^2 s^2)]}{(c^2 c_1^2 + d^2 d_1^2 s^2 s_1^2)(1-d^2 s_1^2)} \right]^2 \quad (C.17)$$

Now, c^2 , d^2 can be expressed in terms of s^2 ; and c_1^2 , d_1^2 in terms of s_1^2 (see [11]) as

$$c^2 = 1 - s^2 \quad (C.18)$$

$$d^2 = 1 - k^2 s^2 \quad (C.19)$$

$$c_1^2 = 1 - s_1^2 \quad (C.20)$$

$$d_1^2 = 1 - k^2 s_1^2 \quad (C.21)$$

Substituting equations (C.18)→(C.21) in equation (C.22) gives

$$x^2 = \frac{(1-s^2)(1-k^2s^2)s^2[(1-s_1^2)(1-k^2s_1^2)\{1-(1-k^2s^2)s_1^2\}+k^2s_1^2\{(1-s^2)(1-s_1^2)+(1-k^2s_1^2)s^2\}]^2}{[(1-s^2)(1-s_1^2)+(1-k^2s^2)(1-k^2s_1^2)s^2s_1^2]^2[1-(1-k^2s^2)s_1^2]^2} \quad (C.22)$$

$$= \frac{P_{nx2}s^2+P_{nx4}s^4+P_{nx6}s^6+P_{nx8}s^8+P_{nx10}s^{10}}{P_0+P_2s^2+P_4s^4+P_6s^6+P_8s^8+P_{10}s^{10}+P_{12}s^{12}} \quad (C.23)$$

where P_{nx2} , P_{nx4} , P_{nx6} , P_{nx8} , P_{nx10} , P_0 , P_2 , P_4 , P_6 , P_8 , P_{10} , P_{12} are polynomial functions of s_1 :

$$P_{nx2} = [1-2(3-2k^2)s_1^2+(15-18k^2+4k^4)s_1^4-4(5-3k^2)(1-k^2)s_1^6+(15-13k^2)(1-k^2)s_1^8-6(1-k^2)^2s_1^{10}+(1-k^2)^2s_1^{12}] \quad (C.24)$$

$$P_{nx4} = -[(1+k^2)-2\{3+2(1-k^2)k^2\}s_1^2+(15+7k^2-22k^4+4k^6)s_1^4-4(5+7k^2-5k^4)(1-k^2)s_1^6+(15+22k^2-29k^4)(1-k^2)s_1^8-2(3+8k^2)^2(1-k^2)^2s_1^{10}+(1+3k^2)^2(1-k^2)^2s_1^{12}] \quad (C.25)$$

$$P_{nx6} = [1-2(4-k^2)s_1^2+(5+k^2)(5-4k^2)s_1^4-4(5-k^2)(2+k^2)(1-k^2)s_1^6+(7+5k^2)(5-4k^2)(1-k^2)s_1^8-2(8+7k^2)(1-k^2)s_1^{10}+3(1+k^2)^2(1-k^2)^2s_1^{12}]k^2 \quad (C.26)$$

$$P_{nx8} = [-2s_1^2+(11-7k^2)s_1^4-4(6-k^2)(1-k^2)s_1^6+2(13-7k^2-2k^4)(1-k^2)s_1^8-2(7+2k^2)(1-k^2)^2s_1^{10}+(3+k^2)(1-k^2)^2s_1^{12}]k^4 \quad (C.27)$$

$$P_{nx10} = [s_1^4-4(1-k^2)s_1^6+2(3-2k^2)(1-k^2)s_1^8-4(1-k^2)s_1^{10}+(1-k^2)^2s_1^{12}]k^6 \quad (C.28)$$

$$P_0 = [1-4s_1^2+6s_1^4-4s_1^6+s_1^8] \quad (C.29)$$

$$P_2 = -2[1-(5+k^2)s_1^2+2(5+k^2)s_1^4-10s_1^6+(5-2k^2)s_1^8-(1-k^2)^2s_1^{10}] \quad (C.30)$$

$$P_4 = 1-6(1+k^2)s_1^2+(15+22k^2-k^4)s_1^4-4(5+7k^2-2k^4)s_1^6+3\{5+4(1-k^2)k^2\}s_1^8-2(3+2k^2)(1-k^2)s_1^{10}+(1-k^2)^2s_1^{12} \quad (C.31)$$

$$P_6 = -4[-s_1^2+(5+k^2)s_1^4-(10+k^2-k^4)s_1^6+2(10-3k^2-2k^4)s_1^8-5(1-k^2)^2s_1^{10}+(1-k^2)^2s_1^{12}]k^2 \quad (C.32)$$

$$P_8 = [6s_1^4-4(6-k^2)s_1^6+(36-20k^2-k^4)s_1^8-4(6-k^2)(1-k^2)s_1^{10}+6(1-k^2)^2s_1^{12}]k^4 \quad (C.33)$$

$$P_{10} = -2[-2s_1^6+3(2-k^2)s_1^8-(6-k^2)(1-k^2)s_1^{10}+2(1-k^2)^2s_1^{12}]k^6 \quad (C.34)$$

$$P_{12} = [s_1^8-2(1-k^2)s_1^{10}+(1-k^2)s_1^{12}]k^8 \quad (C.35)$$

Equation (C.23) can be written as

$$P_{x0}+P_{x2}s^2+P_{x4}s^4+P_{x6}s^6+P_{x8}s^8+P_{x10}s^{10}+P_{x12}s^{12} = 0 \quad (C.36)$$

$$\text{where } P_{x0} = -P_0x^2 \quad (C.37)$$

$$P_{x2} = P_{nx2}-P_2x^2 \quad (C.38)$$

$$P_{x4} = P_{nx4}-P_4x^2 \quad (C.39)$$

$$P_{x6} = P_{nx6} - P_6 x^2 \quad (C.40)$$

$$P_{x8} = P_{nx8} x^2 - P_8 x^2 \quad (C.41)$$

$$P_{x10} = P_{nx10} - P_{10} x^2 \quad (C.42)$$

$$P_{x12} = -P_{12} x^2 \quad (C.43)$$

Y-Coordinate

Squaring both sides of equation (C.16) we get

$$y^2 = \left[\frac{c_1 d_1 s_1 [d_1^2 s^2 + c^2 c_1^2] + k^2 c^2 s^2 (d^2 s_1 - 1)}{(c^2 c_1^2 + d^2 d_1^2 s^2 s_1^2)(1 - d^2 s_1^2)} \right]^2 \quad (C.44)$$

Substituting equations (C.18)→(C.21) in equation (C.44) gives

$$y^2 = \frac{(1-s_1^2)(1-k^2 s_1^2) s_1^2 [(1-k^2 s^2) \{ (1-k^2 s_1^2) s^2 + (1-s^2)(1-s_1^2) \} + k^2 (1-s^2) s^2 \{ (1-k^2 s^2) s_1^2 - 1 \}]^2}{[(1-s^2)(1-s_1^2) + (1-k^2 s^2)(1-k^2 s_1^2) s^2 s_1^2]^2 [1 - (1-k^2 s^2) s_1^2]^2} \quad (C.45)$$

$$= \frac{P_{ny0} + P_{ny2} s^2 + P_{ny4} s^4 + P_{ny6} s^6 + P_{ny8} s^8 + P_{ny10} s^{10} + P_{ny12} s^{12}}{P_0 + P_2 s^2 + P_4 s^4 + P_6 s^6 + P_8 s^8 + P_{10} s^{10} + P_{12} s^{12}} \quad (C.46)$$

where $P_{ny0}, P_{ny2}, P_{ny4}, P_{ny6}, P_{ny8}, P_{ny10}, P_{ny12}, P_0, P_2, P_4, P_6, P_8, P_{10}, P_{12}$ are polynomial functions of s_1 viz :

$$P_{ny0} = s_1^2 - (4-k^2)s_1^4 + 3(2-k^2)s_1^6 - (4-3k^2)s_1^8 + (1-k^2)s_1^{10} \quad (C.47)$$

$$P_{ny2} = -2[2s_1^2 - (9-2k^2)s_1^4 + (15-7k^2)s_1^6 - (11-8k^2)s_1^8 + 3(1-k^2)s_1^{10}]k^2 \quad (C.48)$$

$$P_{ny4} = [2(1+2k^2)s_1^2 - 2(4+11k^2-2k^4)s_1^4 + (12+43k^2-20k^4)s_1^6 - (8+36k^2-29k^4)s_1^8 + (2+13k^2)(1-k^2)s_1^{10}]k^2 \quad (C.49)$$

$$P_{ny6} = -4[s_1^2 - (5+k^2)s_1^4 + (3-k^2)(3+2k^2)s_1^6 - (7+3k^2-5k^4)s_1^8 + (1-k^2)(2+3k^2)s_1^{10}]k^4 \quad (C.50)$$

$$P_{ny8} = [s_1^2 - (4+7k^2)s_1^4 + (6+23k^2-4k^4)s_1^6 - (4+25k^2-10k^4-4k^6)s_1^8 + (1+10k^2+4k^4)(1-k^2)s_1^{10}]k^4 \quad (C.51)$$

$$P_{ny10} = -2[-s_1^4 + (3+k^2)s_1^6 - \{3+2k^2(1-k^2)\}s_1^8 + (1+2k^2)(1-k^2)s_1^{10}]k^6 \quad (C.52)$$

$$P_{ny12} = [s_1^6 - (2-k^2)s_1^8 + (1-k^2)s_1^{10}]k^8 \quad (C.53)$$

Equation (C.46) can be written as

$$P_{y0} + P_{y2}s^2 + P_{y4}s^4 + P_{y6}s^6 + P_{y8}s^8 + P_{y10}s^{10} + P_{y12}s^{12} = 0 \quad (C.54)$$

$$\text{where } P_{y0} = P_{ny0} - P_0 y^2 \quad (C.55)$$

$$P_{y2} = P_{ny2} - P_2 y^2 \quad (C.56)$$

$$P_{y4} = P_{ny4} - P_4 y^2 \quad (C.57)$$

$$P_{y6} = P_{ny6} - P_6 y^2 \quad (C.58)$$

$$P_{y8} = P_{ny8} - P_8 y^2 \quad (C.59)$$

$$P_{y10} = P_{ny10} - P_{10} y^2 \quad (C.60)$$

$$P_{y12} = P_{ny12} - P_{12} y^2 \quad (C.61)$$

Equations (C.36) and (C.54) are, respectively, of the form

$$f(k, s, s_1, x) = 0 \quad (C.62)$$

$$f(k, s, s_1, y) = 0 \quad (C.63)$$

Equations (C.62) and (C.63) are solved simultaneously for s and s_1 , for every x and y input, using a quasi Newton-Raphson numerical method.

APPENDIX D

APPENDIX D

EVALUATING THE STRENGTHS OF THE POINT VORTICES

We separate

$$[u - i v]_k = \sum_{\substack{j=1 \\ j \neq k}}^N -i\kappa_j \left[\frac{1}{z_k - z_j} - \frac{1}{z_k - a^2/\bar{z}_j} \right] \quad (D.1)$$

into velocity components to get :

$$u = \kappa_j \left[\frac{(A02)(AY)}{DENOM2} - \frac{YMY0}{DENOM1} \right] = f(\text{vortex strength, geometric factors}) \quad (D.2)$$

$$v = \kappa_j \left[\frac{XMX0}{DENOM1} - \frac{(A02)(AX)}{DENOM2} \right] = g(\text{vortex strength, geometric factors}) \quad (D.3)$$

where

κ_j = Vortex strength

$A02 = x_j^2 + y_j^2$

a = Radius of circle

$AY = y_k(A02) - a^2 y_j$

$AX = x_k(A02) - a^2 x_j$

$YMY0 = y_k - y_j$

$XMX0 = x_k - x_j$

$DENOM1 = (XMX0)^2 + (YMY0)^2$ $DENOM2 = (AX)^2 + (AY)^2$

j, k = indices for vortex location and point of interest respectively

Now,

$$(V_\theta)_k = v \cos \theta_k - u \sin \theta_k$$

$$= \kappa_j \left\{ \left[\frac{XMX0}{DENOM1} - \frac{(A02)(AX)}{DENOM2} \right] \cos \theta_k - \left[\frac{(A02)(AY)}{DENOM2} - \frac{YMY0}{DENOM1} \right] \sin \theta_k \right\}$$

$$(V_{\theta})_k = \kappa_j f_{kj} \quad (D.4)$$

$$\text{where } f_{kj} = \left[\frac{XMX0}{DENOM1} - \frac{(A02)(AX)}{DENOM2} \right] \cos\theta_k - \left[\frac{(A02)(AY)}{DENOM2} - \frac{YMY0}{DENOM1} \right] \sin\theta_k \quad (D.5)$$

From Equation (D.4) we express the tangential velocity at k points of interest due to j vortices as

$$(V_{\theta})_1 = \kappa_1 f_{11} + \kappa_2 f_{12} + \kappa_3 f_{13} + \dots + \kappa_j f_{1j}$$

$$(V_{\theta})_2 = \kappa_1 f_{21} + \kappa_2 f_{22} + \kappa_3 f_{23} + \dots + \kappa_j f_{2j}$$

$$(V_{\theta})_3 = \kappa_1 f_{31} + \kappa_2 f_{32} + \kappa_3 f_{33} + \dots + \kappa_j f_{3j}$$

$$\begin{array}{cccccc} \cdot & \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \cdot & \dots & \cdot \end{array}$$

$$(V_{\theta})_k = \kappa_1 f_{k1} + \kappa_2 f_{k2} + \kappa_3 f_{k3} + \dots + \kappa_j f_{kj}$$

Or, in matrix form we get

$$\begin{bmatrix} (V_{\theta})_1 \\ (V_{\theta})_2 \\ (V_{\theta})_3 \\ \cdot \\ \cdot \\ (V_{\theta})_k \end{bmatrix} = \begin{bmatrix} f_{11} & f_{12} & f_{13} & \dots & f_{1j} \\ f_{21} & f_{22} & f_{23} & \dots & f_{2j} \\ f_{31} & f_{32} & f_{33} & \dots & f_{3j} \\ \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot \\ f_{k1} & f_{k2} & f_{k3} & \dots & f_{kj} \end{bmatrix} \begin{bmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \\ \cdot \\ \cdot \\ \kappa_j \end{bmatrix} \quad \text{or } (V_{\theta})_k = F_{kj} S_j \quad (D.6)$$

Solving for κ_j there results

$$\begin{bmatrix} f_{11} & f_{21} & f_{31} & \dots & f_{k1} \\ f_{12} & f_{22} & f_{32} & \dots & f_{k2} \\ f_{13} & f_{23} & f_{33} & \dots & f_{k3} \\ \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot \\ f_{1j} & f_{2j} & f_{3j} & \dots & f_{kj} \end{bmatrix} \begin{bmatrix} (V_{\theta})_1 \\ (V_{\theta})_2 \\ (V_{\theta})_3 \\ \cdot \\ \cdot \\ (V_{\theta})_k \end{bmatrix} = \begin{bmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \\ \cdot \\ \cdot \\ \kappa_j \end{bmatrix} \quad \text{or } F_{kj}^{-1} (V_{\theta})_k = S_j \quad (D.7)$$

APPENDIX E

APPENDIX E

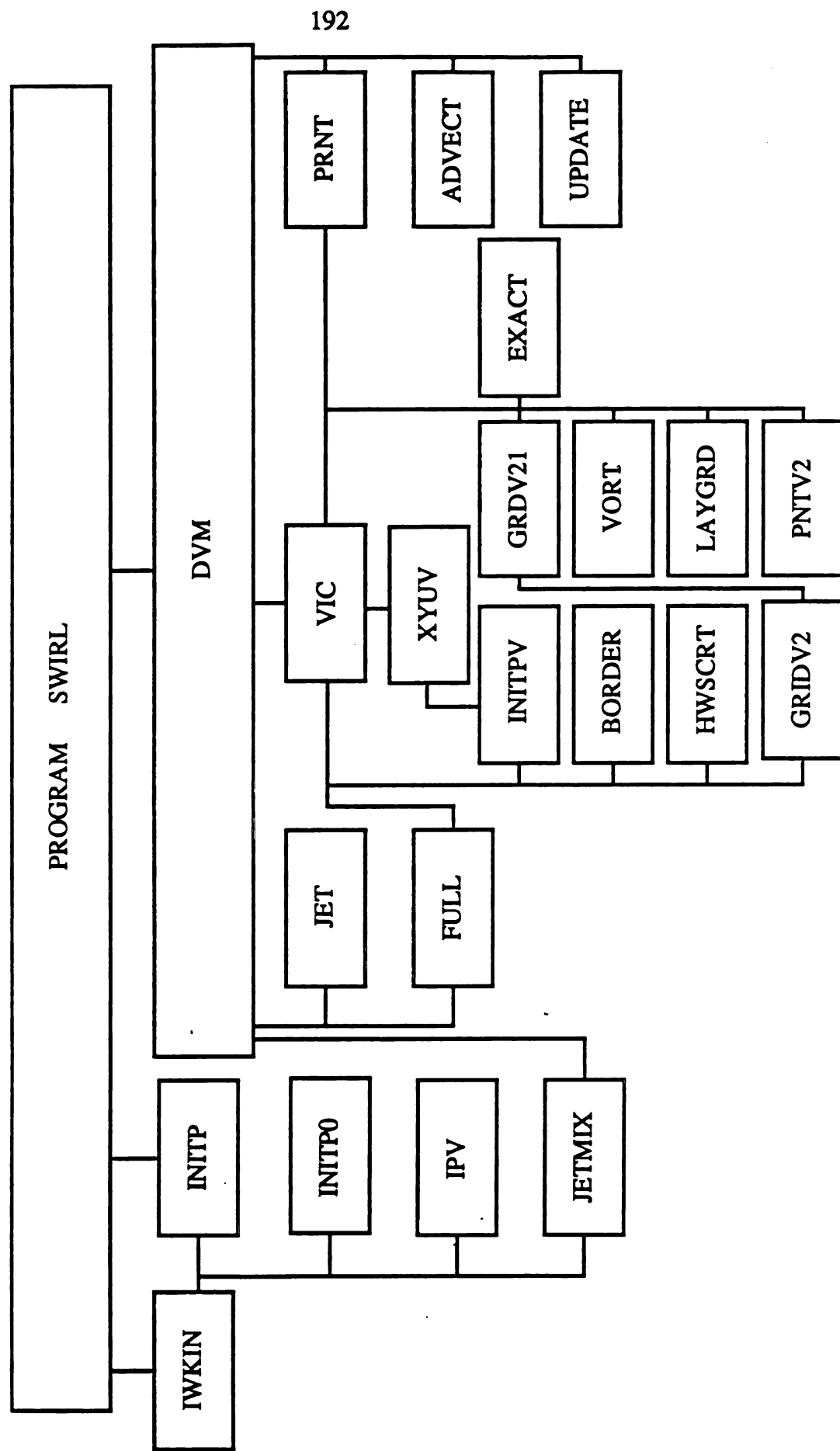
LIST OF COMPUTER PROGRAMS

	Page
Flow Chart	192
Program Nomenclature for some Arrays and Control Parameters	193
Program Swirl	197
DVM Library	199
- Subroutine INITP	199
- Subroutine INITP0	201
- Subroutine IPV	203
- Subroutine GFCTOR	204
- Subroutine JETMIX	206
- Subroutine DVM	208
- Subroutine JET	209
- Subroutine FULL	210
- Subroutine VIC	212
- Subroutine INITPV	213

- Subroutine BORDER	215
- Subroutine GRIDV2	215
- Subroutine GRDV21	217
- Subroutine EXACT	217
- Subroutine VORT	219
- Subroutine LAYGRD	221
- Subroutine PNTV2	224
- Subroutine PRNT	226
- Subroutine ADVECT	228
- Subroutine UPDATE	231
- Function PHI	231
DMAP Library	233
- Subroutine SSN1	233
- Subroutine XYUV	234
- Subroutine NAS2	234
- Subroutine NASNWR	236
- Subroutine NASJME	240
- Subroutine NASFNE	241
- Function NASEPS	242
- Subroutine NASABX	243
- Subroutine DECOM	243
- Subroutine SUBST	244

- Subroutine MOVEDD	245
- Subroutine NORMSD	245
- Subroutine EXCHSD	246
- Subroutine VIPASD	246
- Subroutine F	247
- Subroutine INTPLU	249
- Subroutine CNDNSN	250
- Function CN	251
- Function DN	251
- Function SN	251
- Function FUNCTN	252
- Function GAUSS	252
POISSON Library	254
- Subroutine COSGEN	254
- Subroutine GENBUN	255
- Subroutine HWSCRT	256
- Subroutine MERGE	262
- Subroutine POISD2	263
- Subroutine POISN2	269
- Subroutine POISP2	279
- Subroutine TRIX	282
- Subroutine TRI3	283

FLOWCHART FOR COMPUTER PROGRAMS



[illegible]

```

C***-----**
C*****
C*** Arrays :**
C*** GPNT - Vortex strength**
C*** RC - Radial locations of concentric circles**
C*** RPNT - Radial-coordinates of point vortices**
C*** S - Geometric shape factors**
C*** SINV - Inverted geometric shape factors**
C*** SRD - Dyer's velocity test points**
C*** SVD - Dyer's velocity data at test points**
C*** TPNT - Tangential-coordinates of point vortices**
C*** XPNT - x-coordinates of point vortices**
C*** YPNT - y-coordinates of point vortices**
C*****
C*** Parameters :**
C*** GPVMN - Minimum vortex strength**
C*** GPVMX - Maximum vortex strength**
C*** MXPV - Maximum allowable size for arrays NPVC(), RC()**
C*** NPV - Number of point vortices**
C*** NCC1 - Number of concentric circles inside of circular do**
C*** NCC2 - Number of reference points for Dyer's data**
C*** NPVC - Number of point vortices per concentric circle**
C***-----**
C*** JETMIX :**
C***-----**
C*****
C*** Parameters :**
C*** DSPM - Dsplcmt from injector lip to inner edge of shear layer**
C*** JETON = Turn injector on**
C*** JETOFF = Turn injector off**
C*** NXDIR - Number of grid cells in the x-direction**
C*** NYDIR - Number of grid cells in the y-direction**
C*** VIN - Inlet jet velocity**
C*** XA - x-location of injector lip**
C*** XAB - Horizontal distance between injector lip and seat**
C*** XB - x-location of injector seat**
C*** XPA - x-distance between vortex particle and lip**
C*** XPB - x-distance between vortex particle and seat**
C*** YAB - Vertical distance between injector lip and seat**
C*** YB - y-location of injector seat**
C*** YPA - y-distance between vortex particle and lip**
C*** YPB - y-distance between vortex particle and seat**
C*****
C***-----**
C*** JET :**
C***-----**
C*****
C*** Arrays :**
C*** FTAG - Tags for jet vortices**
C*****
C*** Parameters :**
C*** GPA - Strength of vortex issuing from valve lip**
C*** GPB - Strength of vortex issuing from valve seat**

```

```

C*** XLA      - x-location of vortex particle issuing from valve lip  **
C*** XLB      - x-location of vortex particle issuing from valve seat **
C*** YLA      - y-location of vortex particle issuing from valve lip  **
C*** YLB      - y-location of vortex particle issuing from valve seat **
C***                                     **
C***-----**
C*** XYUV :                                     **
C***-----**
C***                                     **
C*** Parameters :                                     **
C*** G1,G2    - Initial guesses for the iteration process NAS routine **
C*** UPNT     - x-coordinate of vortex in square domain                **
C*** VPNT     - y-coordinate of vortex in square domain                **
C***                                     **
C***-----**
C*** HWSCRT :                                     **
C***-----**
C***                                     **
C*** Arrays :                                     **
C*** BDA,BDB,BDC,BDD - Boundary data                                  **
C*** F         - Streamfunction                                       **
C*** W         - Work space                                           **
C***                                     **
C*** Parameters :                                     **
C*** IDIMF     - Number of grid points in the x-direction            **
C*** IDIMW     = 2*NCOUNT*NCOUNT                                     **
C*** IDIMWP    = IDIMW+2*NCOUNT                                       **
C*** JDIMF     - Number of grid points in the y-direction            **
C***                                     **
C***-----**
C*** GRDV21 :                                     **
C***-----**
C***                                     **
C*** Arrays :                                     **
C*** UG1       - u-component velocity in the circular domain          **
C*** UG1S      - u-comp. vel. in the circular domain (single precision) **
C*** UG2       - u-component velocity in the square domain            **
C*** VG1       - v-component velocity in the circular domain          **
C*** VG1S      - v-comp. vel. in the circular domain (single precision) **
C*** VG2       - v-component velocity in the square domain            **
C***                                     **
C*** Parameters :                                     **
C*** E11,E12- Matrices for velocity transformation                   **
C***                                     **
C***-----**
C*** EXACT :                                     **
C***-----**
C***                                     **
C*** Arrays :                                     **
C*** UG1E      - u-component velocity in the circular domain          **
C*** VG1E      - v-component velocity in the circular domain          **
C*** Parameters :                                     **
C*** NAP       - Non-analytical grid points                          **
C***                                     **

```

```

C***-----**
C*** VORT : **
C***-----**
C*** **
C*** Arrays : **
C*** Q - Vorticity **
C*** **
C***-----**
C*** LAYGRD : **
C***-----**
C*** **
C*** Arrays : **
C*** IBOX - Domain tags **
C*** NG - Singularity tags **
C*** UGN - u-component velocity on uniform grid in circular domain**
C*** VGN - v-component velocity on uniform grid in circular domain**
C*** XY - uniform (x,y) coordinates in circular domain **
C*** **
C***-----**
C*** PNTV2 : **
C***-----**
C*** **
C*** Arrays : **
C*** U - u-component velocity at vortex location **
C*** V - v-component velocity at vortex location **
C*** **
C*****
C*** FUNCTIONS **
C*****
C***-----**
C*** PHI : **
C***-----**
C*** **
C*** PHI - Interpolation function (Filter) **
C*** DHU,DHV- Weight values **
C*** **
C***-----**
C*** CN,DN,SN : **
C***-----**
C*** **
C*** CN, DN, SN - Jacobian elliptic functions **
C*** **
C***-----**
C*** GAUSS : **
C***-----**
C*** **
C*** Gauss-Legendre quadrature for integration purposes **
C*** **
C*****
C*** **

```

```

C***
C*****
      PROGRAM SWIRL
C*****
C***
C***   THIS PROGRAM EMPLOYS DISCRETE VORTICES TO SIMULATE THE MIXING **
C***   PROCESS PRIOR TO CHEMICAL REACTION IN THE FUEL-INJECTED CYLINDER **
C***   OF A PISTON ENGINE. MIXING DESCRIPTIONS ARE RESTRICTED TO A TWO **
C***   DIMENSIONAL, INCOMPRESSIBLE, INVISCID FORMULATION OF THE BULK **
C***   DIFFUSION PROCESS. THE NUMERICAL SIMULATION IS BASED ON A **
C***   VORTEX-IN-CELL FORMULATION. POISSON INVERSION IS ACCOMPLISHED **
C***   USING THE FAST FOURIER TRANSFORM PACKAGE "FISHPAK" FROM THE **
C***   NATIONAL CENTER FOR ATMOSPHERIC RESEARCH (NCAR), BOULDER, **
C***   COLORADO. PLOTIT AND NCAR GRAPHICS PACKAGES ARE USED. **
C***
C*****
C*****
      DOUBLE PRECISION G1,G2,P,SN,SS1,RK,RMOD2,XX,YY
      DOUBLE PRECISION UG1,UG1E,UVDATA,VG1,VG1E,XYDATA
      PARAMETER (JERT=1,MCOUNT=181,NPRNT=1,SWRL=1.0)
      PARAMETER (NCASE=1,NCOUNT=61,G1=0.3719D0,G2=0.3719D0)
      PARAMETER (IDIMAX=1300,MAXPV=12000,NDATA=NCOUNT*NCOUNT)
C***
      COMMON/BNDARY/MBDCND,NBDCND
      COMMON/GRID1/DU,DUINVT,DV,DVINVT,HDUINV,HDVINV
      COMMON/GRID2/ULNGTH,UMN,UMX,VLNGTH,VMN,VMX
      COMMON/NSIZE/IMAX,IMID,IMIN,ISIZE,JMAX,JMID,JMIN,JSIZE
      COMMON/CONSNT/PI,SGMA,SGMA0,TWOPI
      COMMON/GRID3/DX,DXINVT,DY,DYINVT,XMN,XX,YMN,XX
      COMMON/JETPAR/DJ,PMSIGN,TJ,VIN,XA,XB,XPA,XPB,YAB,YB,YPB
      COMMON/MAP/RK,RMOD2,XX,YY
      COMMON/MIXPAR/JETOFF,JETON,NXDIR,NYDIR
      COMMON/PTPAR1/GPVMN,GPVMX,LCODE,NO,NPV
      COMMON/PTPAR2/CNST1,H2,RW,SGMA2I,TOL
      COMMON/TIMER/DTIME,NFULL,NSTART,NSTOP,NTIME,TIME
      COMMON/WORKSP/RWKSP(2727472)
C***
      VIRTUAL   UVDATA(2,NDATA),XYDATA(2,NDATA)
      VIRTUAL   P(MCOUNT),SN(MCOUNT),SS1(2,NDATA)
      VIRTUAL   XPNT(MAXPV),YPNT(MAXPV)
      VIRTUAL   GPNT(MAXPV),RPNT(MAXPV),TPNT(MAXPV)
      VIRTUAL   UG1E(NCOUNT,NCOUNT),VG1E(NCOUNT,NCOUNT)
      VIRTUAL   UPNT(MAXPV),VPNT(MAXPV)
      DIMENSION F(NCOUNT,NCOUNT),W(IDIMAX)
      VIRTUAL   BDA(NCOUNT),BDB(NCOUNT),BDC(NCOUNT),BDD(NCOUNT)
      VIRTUAL   UG2(NCOUNT,NCOUNT),VG2(NCOUNT,NCOUNT)
      VIRTUAL   Q(NCOUNT,NCOUNT)
      VIRTUAL   UN(MAXPV),VN(MAXPV),XPN(MAXPV),YPN(MAXPV)
      VIRTUAL   UG1(NCOUNT,NCOUNT),VG1(NCOUNT,NCOUNT)
      VIRTUAL   UG1S(NCOUNT,NCOUNT),VG1S(NCOUNT,NCOUNT)
      VIRTUAL   UPV1(MAXPV),VPV1(MAXPV)
      VIRTUAL   IBOX(NCOUNT,NCOUNT),NG(NCOUNT,NCOUNT),XY(2,NDATA)
      VIRTUAL   FTAG(2,MAXPV),UGN(NCOUNT,NCOUNT),VGN(NCOUNT,NCOUNT)
C***

```



```

      OPEN (UNIT=1, FILE='SWIRLS.CLNS', TYPE='NEW')
      OPEN (UNIT=2, FILE='SWIRLS.CLNS', TYPE='NEW')
      OPEN (UNIT=3, FILE='SWIRLS.CLNS', TYPE='NEW')
      OPEN (UNIT=4, FILE='SWIRLS.CLNS', TYPE='NEW')
      OPEN (UNIT=5, STATUS='OLD', FILE='SYS$OUTPUT')
      OPEN (UNIT=6, FILE='SWIRLS.CLNS', TYPE='NEW')
      OPEN (UNIT=7, FILE='SWIRLS.CLNS', TYPE='NEW')
      OPEN (UNIT=8, FILE='SWIRLS.CLNS', TYPE='NEW')
C***
C***-----
C***  Initialize parameters :
C***-----
C***
      CALL IWKIN(2727472)
      CALL INITP (GPNT, JERT, MAXPV, MCOUNT, NCOUNT, P, RPNT, SN, SS1,
      *SWRL, TPNT, UVDATA, XPNT, XYDATA, YPNT)
C***
C***-----
C***  Perform discrete vortex vortex calculations :
C***-----
C***
      DO 1 NTIME = NSTART, NSTOP
C***
      CALL DVM (BDA, BDB, BDC, BDD, F, FTAG, G1, G2, GPNT, IBOX, JERT,
      *MAXPV, MCOUNT, NCASE, NCOUNT, NG, NPRNT, P, Q, RPNT, SN, SS1, SWRL,
      *TPNT, UGN, UG1, UG1E, UG1S, UG2, UN, UPNT, UPV1, UVDATA, VGN, VG1,
      *VG1E, VG1S, VG2, VN, VPNT, VPV1, VR, W, XPN, XPNT, XY, XYDATA, YPN, YPNT)
C***
      IF (LCODE.EQ.0.OR.NTIME.EQ.NSTOP) GO TO 2
C***
      1 CONTINUE
C***
C***-----
C***  Close the debugging file
C***-----
C***
      2 CLOSE (UNIT=1)
      CLOSE (UNIT=2)
      CLOSE (UNIT=3)
      CLOSE (UNIT=4)
      CLOSE (UNIT=6)
      CLOSE (UNIT=7)
      CLOSE (UNIT=8)
C***
      STOP 'NORMAL TERMINATION OF PROGRAM SWIRLS'
      END
C***

```

```

C***
C*****
C***      DVM LIBRARY CONTAINS THE FOLLOWING SUBROUTINES/FUNCTIONS :      ***
C***                                          **
C*** SUBROUTINES :                                          **
C*** 1. INITP      - Initializes parameters                  **
C***   -INITP0     - Initializes grid and elliptic parameters **
C***   -IPV        - Initializes coords. & strengths of the vortices **
C***   -GFCTOR     - Computes the geometric shape factors    **
C***   -LSGRR      - Inverts the geometric shape factors [IMSL] **
C***   -JETMIX     - Initializes jet and mixing parameters    **
C*** 2. DVM        - Makes calls vortex routines :         **
C***   -JET        - Simulates fluid injection               **
C***   -FULL       - Checks if cylinder fluid is "completely mixed" **
C***   -VIC        - Calls Vortex-In-Cell formulation routines **
C***   -XYUV       - Maps coordinates from circle to square [Dmaplib] **
C***   -INITPV     - Initializes vorticity on square grid     **
C***   -BORDER     - Borders active points according to bndary conds. **
C***   -HWSCRT     - Does the Fast Fourier Transform [Poissonlib] **
C***   -GRIDV2     - Computes velocities on a square grid     **
C***   -GRDV21     - Maps grid velocities : square -> circle  **
C***   -EXACT      - Computes exact velocities for non-analytical pts. **
C***   -VORT       - Computes vorticity at grid points in uv-plane **
C***   -LAYGRD     - Computes velocities on imposed uniform grid **
C***   -PNTV2      - Interpolates velocities at off grid locations **
C***   -PRNT       - Prints data                               **
C***   -ADVECT     - Advects vortices via Heun's time integrtn.scheme **
C***   -UPDATE     - Updates parameters/array size           **
C*****                                          *****
C*** FUNCTIONS :                                          **
C***                                          **
C*** 1. PHI        - Evaluates the interpolation functions      **
C***                                          **
C*** SUBROUTINE(S)/FUNCTIONS CALLED BUT NOT IN THIS LIBRARY : **
C*** SUBROUTINES :                                          **
C*** 1. CNDNSN     - Evaluates Jacobian elliptic functions CN,DN and SN **
C*** 2. SSN1       - Evaluates Jacobian elliptic function SN   **
C*** 3. XYUV       - (See above)                               **
C*** 4. HWSCRT     - (See above)                               **
C*** 5. PRNT       - (See above)                               **
C*****                                          **
C*** FUCTIONS :                                          **
C*** 2. FUNCTN     - Evaluates integrand of elliptic integral K(or K') **
C***                  [1/sqrt(1-k**2sin**2(phi))]/[1/sqrt(1-k'**2sin**2(phi))] **
C*** 3. GAUSS      - Performs Gaussian integration of elliptic integrals **
C***                                          **
C*****
C***      SUBROUTINE INITP (GPNT, JERT, MAXPV, MCOUNT, NCOUNT, P, RPNT, SN, SS1,
C***      *SWRL, TPNT, UVDATA, XPNT, XYDATA, YPNT)
C*****
C***
C***      THIS SUBROUTINE INITIALIZES PARAMETERS
C***
C*****

```

```

DOUBLE PRECISION P,RK,RMOD2,SN,SS1,UVDATA,XX,XYDATA,YY
COMMON/BNDARY/MBDCND,NBDCND
COMMON/GRID1/DU,DUINVT,DV,DVINVT,HDUINV,HDVINV
COMMON/GRID2/ULNGTH,UMN,UMX,VLNGTH,VMN,VMX
COMMON/NSIZE/IMAX,IMID,IMIN,ISIZE,JMAX,JMID,JMIN,JSIZE
COMMON/CONSNT/PI,SGMA,SGMA0,TWOPI
COMMON/GRID3/DX,DXINVT,DY,DYINVT,XMN,XX,XYDATA,YY
COMMON/JETPAR/DJ,PMSIGN,TJ,VIN,XA,XB,XPA,XPB,YAB,YB,YPB
COMMON/MAP/RK,RMOD2,XX,YY
COMMON/MIXPAR/JETOFF,JETON,NXDIR,NYDIR
COMMON/PTPAR1/GPVMN,GPVMX,LCODE,NO,NPV
COMMON/PTPAR2/CNST1,H2,RW,SGMA2I,TOL
COMMON/TIMER/DTIME,NFULL,NSTART,NSTOP,NTIME,TIME
EXTERNAL GAUSS,FUNCTN

```

C***

C*****

C***-----

C*** Grid, elliptic and interpolation parameters/arrays :

C***-----

C***

```

      CALL INITP0(DU,DV,DX,DY,MCOUNT,NCOUNT,P,PI,RW,SN,SS1,TWOPI,
      *UVDATA,XYDATA)

```

C***

C***-----

C*** Time step parameters :

C***-----

C***

```

      NFULL = 0
      NSTART = 1
      NSTOP = 1001
      DTIME = 0.001

```

C***

C***-----

C*** Vortex parameters :

C***-----

C***

```

      LCODE = 1
      NPV = 0
      GPVMN = 1.0E+06
      GPVMX = -GPVMN
      CNST1 = 2.0/3.0
      H2 = 3.0
      SGMA0 = 1.0E-06
      SGMA = 0.05
      SGMA2I = 1.0/(SGMA*SGMA)
      TOL = 10.*AMACH(4)

```

C***

C***-----

C*** Square domain parameters :

C***-----

C***

```

      MBDCND = 1
      NBDCND = 1
      DUINVT = 1.0/DU

```

```

      DXINVT = 1.0/DX
      DVINVT = 1.0/DV
      DYINVT = 1.0/DY
      HDUINV = 0.5*DUINVT
      HDVINVT = 0.5*DVINVT
      UMN = -0.5*SNGL(RK)
      UMX = 0.5*SNGL(RK)
      ULNGTH = SNGL(RK)
      VLNGTH = ULNGTH
      VMN = UMN
      VMX = UMX
      XMN = -RW
      XMX = RW
      YMN = XMN
      YMX = XMX

C***
C***-----
C*** Initialize coordinates and strengths of vortices :
C***-----
C***
      IF (SWRL.EQ.1.0) CALL IPV(GPNT,RPNT,TPNT,XPNT,YPNT)
C***
C***-----
C*** Initialize jet and mixing parameters :
C***-----
C***
      CALL JETMIX(JERT,NCOUNT)
C***
      RETURN
      END
C***
C***
C*****
      SUBROUTINE INITP0 (DU,DV,DX,DY,MCOUNT,NCOUNT,P,PI,RW,SN,SS1,TWOPI,
      *UVDATA,XYDATA)
C*****
C*** **
C*** THIS SUBROUTINE INITIALIZES GRID AND ELLIPTIC PARAMETERS : **
C*** (i) S(U) : U in P() and S in SN() **
C*** (ii) Grid point coordinates: (U,V) in UVDATA(); (X,Y) in XYDATA() **
C*** **
C*****
C***
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON/MAP/RK,RMOD2,XX,YY
      COMMON/NSIZE/IMAX,IMID,IMIN,ISIZE,JMAX,JMID,JMIN,JSIZE
      EXTERNAL GAUSS,FUNCTN
      VIRTUAL P(*),SN(*),SS1(2,*),UVDATA(2,*),XYDATA(2,*)
C***
C*****
C***-----
C*** Grid parameters :
C***-----
C***

```

```

      IMIN   = 1
      IMAX   = NCOUNT
      IMID   = (IMAX+IMIN)/2
      ISIZE  = IMAX-1
      JMID   = IMID
      JMIN   = IMIN
      JMAX   = IMAX
      JSIZE  = ISIZE

C***
C***-----
C*** Elliptic parameters :
C***-----
C***
      RMOD2  = 0.5D0
      PI     = DACOS(-1.0D0)
      TWOPI  = PI+PI
      RK     = GAUSS( 0.0D0,0.5D0*PI,15,FUNCTN,RMOD2 )
      DU     = RK/DFLOAT(ISIZE)
      DV     = DU

C***
C***-----
C*** Pretabulate S elliptic function :
C***-----
C***
      DUU    = RK/DFLOAT(MCOUNT-1)
      DO 1 I = 1,MCOUNT
      P(I)   = DUU*DFLOAT(I-1) - 0.5D0*RK
      CALL SSN1( P(I), RMOD2, SN(I) )
1 CONTINUE

C***
C***-----
C*** Initialize grid point coordinates (U,V) and (X,Y) :
C***-----
C***
      DO 2 J = JMIN,JMAX
      V      = DV*DFLOAT(J-1) - 0.5D0*RK
      CALL CNDNSN( V, RMOD2, C1, D1, S1 )
      C12    = C1*C1
      D12    = D1*D1
      S12    = S1*S1
      DO 2 I = IMIN,IMAX
      K      = I+(J-JMIN)*JMAX
      U      = DU*DFLOAT(I-1) - 0.5D0*RK
      CALL CNDNSN( U, RMOD2, C, D, S )
      C2     = C*C
      D2     = D*D
      S2     = S*S
      DENOM  = (C2*C12+D2*D12*S2*S12)*(1.0D0-D2*S12)
      XNUM   = C*D*S*(C12*D12*(1.0D0-D2*S12)+RMOD2*S12*(C2*C12+D12*S2))
      YNUM   = C1*D1*S1*(D2*(D12*S2+C2*C12)+RMOD2*C2*S2*(D2*S12-1.0D0))
      UVDATA(1,K) = U
      UVDATA(2,K) = V
      XYDATA(1,K) = XNUM/DENOM
      XYDATA(2,K) = YNUM/DENOM

```

```

C***
      IF (I.EQ.IMAX.AND.J.EQ.JMAX) THEN
        RW      = DSQRT(XYDATA(1,K)**2+XYDATA(2,K)**2)
        DX      = 2.0D0*RW/DFLOAT(ISIZE)
        DY      = DX
      END IF
C***
C***-----
C***  Store non-analytical points :
C***-----
C***
      IF (DABS(U).EQ.DABS(V)) THEN
        SS1(1,K) = S1
        SS1(2,K) = S
      END IF
C***
      2 CONTINUE
C***
      RETURN
      END
C***
C***
C*****
      SUBROUTINE IPV(GPNT,RPNT,TPNT,XPNT,YPNT)
C*****
C***
C***  THIS SUBROUTINE INITIALIZES THE COORDINATES AND STRENGTHS
C***  OF THE POINT VORTICES USING DYER'S VELOCITY DATA [33] AND
C***  MATRIX INVERSION ROUTINE FROM IMSL ROUTINES.
C***
C*****
C***
      PARAMETER(MXPV=200,NCC1=8,NCC2=10)
C***
C***  Number of vortices per concentric circle :
C***
C      DATA NPVC/4,6,6,6,28,32,15,8/           !@100 ms.
C      DATA NPVC/4,6,6,8,31,32,15,8/           !@150 ms.
      DATA NPVC/4,6,6,13,31,32,15,8/           !@200 ms.
C      DATA NPVC/4,6,6,16,34,34,18,8/           !@300 ms.
C      DATA NPVC/4,6,6,6,26,30,16,8/           !@500 ms.
C***
C***  Radial location of concentric circles :
C***
      DATA RC/0.1,0.2,0.30,0.40,0.50,0.60,0.70,0.80/
C***
C***  Velocity test points :
C***
      DATA SRD/0.00,0.10,0.20,0.30,0.40,0.50,0.60,0.70,0.80,0.90/
C***
C***  Dyer's velocity data at test points :
C***
C      DATA SVD/0.00,7.80,24.50,33.10,37.20,39.00,37.60,33.70,!@100 ms.
C      *30.00,28.20/

```

```

C      DATA SVD/0.00,6.20,14.70,21.60,25.10,23.90,22.20,20.40,!@150 ms.
C      *18.80,17.40/
C      DATA SVD/0.00,5.30,12.60,16.60,17.00,16.30,15.10,14.00,!@200 ms.
C      *13.10,12.40/
C      DATA SVD/0.00,3.20,10.00,11.40,10.30,9.60,9.00,8.40,!@300 ms.
C      *7.80,7.60/
C      DATA SVD/0.00,2.80,4.90,5.50,5.20,4.90,4.70,4.50,!@500 ms.
C      *4.30,4.30/
C***
C      COMMON/CONSNT/PI, SGMA, SGMA0, TWOPI
C      COMMON/PTPAR1/GPVMN, GPVMX, LCODE, NO, NPV
C      COMMON/PTPAR2/CNST1, H2, RW, SGMA2I, TOL
C      VIRTUAL S (NCC2, MXPV), SINV (MXPV, NCC2)
C      VIRTUAL NPVC (NCC1), RC (NCC1), SVD (NCC2), SRD (NCC2)
C***
C*****
C***-----
C*** Determine the total number of point vortices
C***-----
C***
C      DO 20 N = 1, NCC1
C      NPV = NPV + NPVC (N)
C 20 CONTINUE
C***
C***-----
C*** Compute the geometric factors, perform matrix inversion
C*** and then compute the strengths of the point vortices
C***-----
C***
C      CALL GFCTOR (GPNT, NCC2, NPV, NCC1, NCC2, NPV, NPVC, NCC2,
C      *RC, S, SINV, SRD, SVD, RPNT, TPNT, XPNT, YPNT)
C***
C      RETURN
C      END
C***
C***
C*****
C      SUBROUTINE GFCTOR (GPNT, LS, LSINV, NCC1, NCC2, NCS, NPVC, NRS, RC, S, SINV,
C      *SRD, SVD, RPNT, TPNT, XPNT, YPNT)
C*****
C*** **
C*** THIS SUBROUTINE COMPUTES THE GEOMETRIC FACTORS, PERFORMS MATRIX **
C*** INVERSION AND THEN COMPUTES THE STRENGTHS OF THE POINT VORTICES **
C*** **
C*****
C***
C      COMMON/CONSNT/PI, SGMA, SGMA0, TWOPI
C      COMMON/PTPAR1/GPVMN, GPVMX, LCODE, NO, NPV
C      COMMON/PTPAR2/CNST1, H2, RW, SGMA2I, TOL
C      VIRTUAL GPNT (*), RPNT (*), TPNT (*), XPNT (*), YPNT (*)
C      VIRTUAL NPVC (*), RC (*), S (LS, *), SINV (LSINV, *), SRD (*), SVD (*)
C      EXTERNAL LSGRR
C***
C*****

```

```

C***-----
C*** Compute the geometric shape factor relative to the reference point :
C***-----
C***
      DO 30 I = 1,NCC2
      J      = 0
      T      = 0.0
      X      = SRD(I)
      Y      = 0.0
C***
      DO 20 N = 1,NCC1
      DELT0   = TWOPI/NPVC(N)
      DO 10 K = 1,NPVC(N)
      J      = J + 1
      T0     = DELT0*FLOAT(K)
      X0     = RC(N)*COS(T0)
      Y0     = RC(N)*SIN(T0)
      IF(I.EQ.1) THEN
      RPNT(J) = SQRT(X0*X0+Y0*Y0)
      TPNT(J) = T0
      XPNT(J) = X0
      YPNT(J) = Y0
      END IF
C***
C***-----
C*** Initialize parameters for the shape factors :
C***-----
C***
      XMX0    = X-X0
      YMY0    = Y-Y0
      A02     = X0*X0+Y0*Y0
      AX      = A02*X-(RW*RW)*X0
      AY      = A02*Y-(RW*RW)*Y0
      AX2     = AX*AX
      AY2     = AY*AY
      XMX02   = XMX0*XMX0
      YMY02   = YMY0*YMY0
      DENOM1  = XMX02+YMY02
      DENOM2  = AX2+AY2
      IF(DENOM2.EQ.0.0) DENOM2 = SGMA0
      SQRTD   = SQRT(DENOM1)
      H1      = (3.0*CNST1*SQRTD)/SGMA
C***
      IF(SQRTD.LE.SGMA) THEN
      SU      = (A02*AY/DENOM2)-SGMA2I*(H2-H1)*YMY0
      SV      = -(A02*AX/DENOM2)+SGMA2I*(H2-H1)*XMX0
      ELSE
      SU      = (A02*AY/DENOM2)-(YMY0/DENOM1)
      SV      = -(A02*AX/DENOM2)+(XMX0/DENOM1)
      END IF
C***
      S(I,J)  = SV*COS(T)-SU*SIN(T)
C***
      10 CONTINUE

```



```

20 CONTINUE
30 CONTINUE
C***
C***-----
C***  Invert the S matrix :
C***-----
C***
      CALL LSGRR(NRS,NCS,S,LS,TOL,IRANK,SINV,LSINV)
C***
C***-----
C***  (i)   Compute the strengths of the point vortices.
C***  (ii)  Check for the minimum and maximum strengths.
C***-----
C***
      DO 50 J = 1,NPV
      GPNT(J) = 0.0
      DO 40 I = 1,NCC2
      GPNT(J) = GPNT(J) + SVD(I)*SINV(J,I)
40 CONTINUE
      GPVMN = AMIN1(GPNT(J),GPVMN)
      GPVMX = AMAX1(GPNT(J),GPVMX)
50 CONTINUE
C***
      RETURN
      END
C***
C*****
      SUBROUTINE JETMIX(JERT,NCOUNT)
C*****
C***
C*** THIS SUBROUTINE INITIALIZES JET AND MIXING SIMULATION PARAMETERS **
C***
C***          JDIR = 0/1 - Same/opposite direction of flow          **
C***          SIV = 0.0/1.0 - Different/same initial velocity.      **
C***
C*****
C***
      PARAMETER(JDIR=1,SIV=1.0)
      COMMON/GRID1/DU,DUINVT,DV,DVINVT,HDUINV,HDVINV
      COMMON/CONSNT/PI,SGMA,SGMA0,TWOPI
      COMMON/JETPAR/DJ,PMSIGN,TJ,VIN,XA,XB,XPA,XPB,YAB,YB,YPB
      COMMON/MIXPAR/JETOFF,JETON,NXDIR,NYDIR
      COMMON/PTPAR2/CNST1,H2,RW,SGMA2I,TOL
      COMMON/TIMER/DTIME,NFULL,NSTART,NSTOP,NTIME,TIME
C***
C*****
C***-----
C***  Initialize jet parameters
C***-----
C***
      IF(JERT.EQ.1) THEN
      JETON = NSTART
      JETOFF = NSTOP
      END IF

```

```

C***
C***-----
C*** Specify injector orifice diameter :
C***-----
C***
      DJ      =  DU
C***
C***-----
C*** Specify injector angle. CCW from horizontal :
C***-----
C***
      TJ      =  PI/3.0
C***
      IF(JDIR.EQ.1) THEN
      XB      = -0.5*DJ*SIN(TJ)
      ELSE
      XB      =  0.5*DJ*SIN(TJ)
      END IF
C***
      IF(XB.GE.0.0) PMSIGN = -1.0
      IF(XB.LE.0.0) PMSIGN =  1.0
      YB      =  0.5*DJ*COS(TJ)
      YAB     =  DJ*COS(TJ)
      XAB     =  DJ*SIN(TJ)
      XA      =  PMSIGN*XAB+XB
C***
C***-----
C*** Compute the initial minimum allowable
C*** displacement of jet particles from injector orifice
C***-----
C***
      IF(TJ.EQ.(PI/2.0)) DSPM = 3.633021*DJ
      IF(TJ.EQ.(PI/2.4)) DSPM = 4.067413*DJ
      IF(TJ.EQ.(PI/3.0)) DSPM = 4.938183*DJ
      IF(TJ.EQ.(PI/4.0)) DSPM = 6.794676*DJ
      IF(TJ.EQ.(PI/6.0)) DSPM = 16.39000*DJ
      XPB     = DSPM*COS(TJ)
      YPB     = DSPM*SIN(TJ)
      XPA     = XPB
      YPA     = YPB
C***
C***-----
C*** Initialize jet velocity :
C***-----
C***
      IF(SIV.EQ.1.0) THEN
      VIN     = 6.794676*DJ/DTIME !If velocity at 90 deg. is desired
      ELSE
      VIN     = DSPM/DTIME
      END IF
C***
C***-----
C*** Initialize mixing parameters :
C***-----

```

```

C***
      NXDIR = 8
      NYDIR = 8
C***
      RETURN
      END
C***
C***
C*****
      SUBROUTINE DVM(BDA,BDB,BDC,BDD,F,FTAG,G1,G2,GPNT,IBOX,JERT,
      *MAXPV,MCOUNT,NCASE,NCOUNT,NG,NPRNT,P,Q,RPNT,SN,SS1,SWRL,
      *TPNT,UGN,UG1,UG1E,UG1S,UG2,UN,UPNT,UPV1,UVDATA,VGN,VG1,VG1E,VG1S,
      *VG2,VN,VPNT,VPV1,VR,W,XPN,XPNT,XY,XYDATA,YPN,YPNT)
C*****
C***
C***      THIS SUBROUTINE MAKES CALLS TO DISCRETE VORTEX ALGORITHMS      **
C***
C*****
C***
      DOUBLE PRECISION G1,G2,P,SN,SS1,UG1,UG1E,UVDATA,VG1,VG1E,XYDATA
      COMMON/MIXPAR/JETOFF,JETON,NXDIR,NYDIR
      COMMON/PTPAR1/GPVMN,GPVMX,LCODE,NO,NPV
      COMMON/TIMER/DTIME,NFULL,NSTART,NSTOP,NTIME,TIME
C***
C***-----
C***      Turn injector on/off
C***-----
C***
      IF(JERT.EQ.1) THEN
      IF(NTIME.GE.JETON.AND.NTIME.LE.JETOFF) THEN
      CALL JET(FTAG,GPNT,MAXPV,RPNT,TPNT,XPNT,YPNT)
      IF(LCODE.EQ.0) GO TO 3
      END IF
C***
C***-----
C***      Check if cylinder fluid is "completely mixed"
C***-----
C***
      CALL FULL(IBOX,NCOUNT,RPNT,XPNT,YPNT)
C***
C***-----
C***      Compute advection velocities :
C***-----
C***
      CALL VIC(BDA,BDB,BDC,BDD,F,G1,G2,GPNT,IBOX,1,JERT,MCOUNT,NCASE,
      *NCOUNT,NG,P,Q,SN,SS1,UGN,UG1,UG1E,UG1S,UG2,UPNT,UPV1,UVDATA,VGN,
      *VG1,VG1E,VG1S,VG2,VPNT,VPV1,W,XPNT,XY,XYDATA,YPNT)
C***
C***-----
C***      Check if time to print data :
C***-----
C***
      IF(NPRNT.EQ.1) THEN
      CALL PRNT(F,FTAG,NCASE,NCOUNT,Q,RPNT,UG1S,UG2,UGN,UVDATA,VG1S,

```

```

      *VG2,VGN,XPNT,XY,XYDATA,YPNT)
      END IF
C***
C***-----
C***  Initialize vortex elements for the next time step :
C***-----
C***
      CALL ADVECT(BDA,BDB,BDC,BDD,F,FTAG,G1,G2,GPNT,IBOX,JERT,MCOUNT,
      *NCASE,NCOUNT,NG,P,Q,RPNT,SN,SS1,TPNT,UG1,UG1E,UG1S,UG2,UGN,UN,
      *UPV1,UPNT,UVDATA,VG1,VG1E,VG1S,VG2,VGN,VN,VPV1,VPNT,XPNT,YPNT,
      *YPNT,W,XY,XYDATA)
C***
C***-----
C***  Udata parameters and monitor array size :
C***-----
C***
      CALL UPDATE(MAXPV)
C***
      RETURN
      END
C***
C*****
      SUBROUTINE JET(FTAG,GPNT,MAXPV,RPNT,TPNT,XPNT,YPNT)
C*****
C***
C***          THIS SUBROUTINE SIMULATES THE FLUID INJECTION PROCESS      **
C***  Ref.:
C***  "Piston-Cylinder Fluid Motion Via Vortex Dynamics", W.T. Ashurst **
C***  Fluid Mechanics of Combustion Systems, ed. by T. Morel et. al. **
C***
C*****
C***
      COMMON/CONSNT/PI,SGMA,SGMA0,TWOPI
      COMMON/JETPAR/DJ,PMSIGN,TJ,VIN,XA,XB,XPA,XPB,YAB,YB,YPB
      COMMON/PTPAR1/GPVMN,GPVMX,LCODE,NO,NPV
      COMMON/PTPAR2/CNST1,H2,RW,SGMA2I,TOL
      COMMON/TIMER/DTIME,NFULL,NSTART,NSTOP,NTIME,TIME
      VIRTUAL GPNT(*),FTAG(2,*),RPNT(*),TPNT(*),XPNT(*),YPNT(*)
C***
C*****
C***  SEAT CALCULATIONS (B)
C*****
C***-----
C***  Compute location of particle issuing from valve seat (region b) :
C***-----
C***
      XLB      = XB+PMSIGN*XPB
      YLB      = (YB+YPB)-RW
      GPB      = PMSIGN*0.5*(VIN*VIN*DTIME)
      GPB      = GPB/TWOPI
C***
C***-----
C***  Update parameters and store values in point vortex arrays :
C***-----

```

```

C***
      NPV          = NPV + 1
      IF (NPV.GT.MAXPV) GO TO 4
      XPNT (NPV) = XLB
      YPNT (NPV) = YLB
      GPNT (NPV) = GPB
      RPNT (NPV) = SQRT (XLB*XLB+YLB*YLB)
      TPNT (NPV) = ATAN2 (YLB,XLB)
      NJP          = NPV

C***
C*****
C***  LIP CALCULATIONS (A) **
C*****
C***-----
C***  Compute location of particle issuing from valve lip (region a) :
C***-----
C***
      XLA          = XA+PMSIGN*XPA
      YLA          = YLB-YAB

C***
C***-----
C***  Update parameters and store values in point vortex arrays :
C***-----
C***
      NPV          = NPV + 1
      IF (NPV.GT.MAXPV) GO TO 4
      XPNT (NPV) = XLA
      YPNT (NPV) = YLA
      GPNT (NPV) = -GPB
      RPNT (NPV) = SQRT (XLA*XLA+YLA*YLA)
      TPNT (NPV) = ATAN2 (YLA,XLA)

C***
C***-----
C***  Tag the jet vortices as fuel particles
C***-----
C***
      2 DO 3 N = NJP,NPV
        FTAG(1,N) = 1.0
      3 CONTINUE

C***
      4 IF (NPV.GT.MAXPV) LCODE = 0
      RETURN
      END

C***
C***
C*****
      SUBROUTINE FULL (FTAG, IBOX, NCOUNT, RPNT, XPNT, YPNT)
C*****
C*** **
C***  THIS SUBROUTINE DETERMINES IF DOMAIN IS FILLED WITH VORTICES **
C*** **
C*****
C***
      PARAMETER (NSKM=1)

```

```

COMMON/MIXPAR/JETOFF, JETON, NXDIR, NYDIR
COMMON/PTPAR1/GPVMN, GPVMX, LCODE, NO, NPV
COMMON/PTPAR2/CNST1, H2, RW, SGMA2I, TOL
COMMON/TIMER/DTIME, NFULL, NSTART, NSTOP, NTIME, TIME
VIRTUAL FTAG(2,*), IBOX(NCOUNT,*), RPNT(*), XPNT(*), YPNT(*)

C***
C*****
C***-----
C***  Initialize parameters :
C***-----
C***
      NCELLS      = 0
      R0          = 2.0*RW/FLOAT(NCIRS)
      DO 7 J = 1, NRAYS
      DO 7 I = 1, NCIRS
      IBOX(I,J) = 0
      7 CONTINUE
C***
C***-----
C***  Tag grid cells within circle :
C***-----
C***
      DO 1 J = 1, NRAYS
      RY      = -RW+R0*FLOAT(J-1)
      RYP     = RY+R0
      HRYP    = RY+(0.5*R0)
      DO 1 I = 1, NCIRS
      RX      = -RW+R0*FLOAT(I-1)
      RXP     = RX+R0
      HRXP    = RX+(0.5*R0)
      RCC     = SQRT(HRXP*HRXP+HRYP*HRYP)
C***
      IF(RCC.LE.RW) THEN
      IBOX(I,J) = 1
      NCELLS   = NCELLS + 1
      END IF
C***
      1 CONTINUE
C***
C***-----
C***  Identify cell :
C***-----
C***
      DO 3 J = 1, NRAYS
      RY      = -RW+R0*FLOAT(J-1)
      RYP     = RY+R0
      DO 3 I = 1, NCIRS
      NYES    = 0
      RX      = -RW+R0*FLOAT(I-1)
      RXP     = RX+R0
C***
C***-----
C***  Tag cell with at least NO vortex/vortices :
C***-----

```

```

C***
      NO          = 2
      IF (IBOX(I,J).EQ.1) THEN
      DO 2 N = 1,NPV
      IF (FTAG(1,N).EQ.1.0) THEN
      IF ((XPNT(N).GE.RX.AND.XPNT(N).LE.RXP).AND.
      * (YPNT(N).GE.RY.AND.YPNT(N).LE.RYP)) THEN
      NYES          = NYES + 1
      IF (NYES.EQ.NO) THEN
      IBOX(I,J) = 2
      GO TO 3
      END IF
      END IF
      END IF
      2 CONTINUE
      END IF
      3 CONTINUE
C***
C***-----
C***  Check if each cell contains at least NO vortex/vortices :
C***-----
C***
      NYES          = 0
      DO 4 J = 1,NRAYS
      DO 4 I = 1,NCIRS
      IF (IBOX(I,J).EQ.1) GO TO 70
      IF (IBOX(I,J).EQ.2) NYES = NYES + 1
      4 CONTINUE
C***
      5 IF (NYES.EQ.NCELLS) THEN
      NFULL          = 1
      END IF
C***
      70 RETURN
      END
C***
C***
C*****
      SUBROUTINE VIC(BDA,BDB,BDC,BDD,F,G1,G2,GPNT,IBOX,IFLG,JERT,
      *MCOUNT,NCASE,NCOUNT,NG,P,Q,SN,SS1,UGN,UG1,UG1E,UG1S,UG2,UPNT,U,
      *UVDATA,VGN,VG1,VG1E,VG1S,VG2,VPNT,V,W,XPNT,XY,XYDATA,YPNT)
C*****
C***
C*** THIS SUBROUTINE COMPUTES GRID VELOCITIES IN SQUARE DOMAIN USING **
C*** VORTEX-IN-CELL (VIC) FORMULATION. OFF-GRID VELOCITIES CALCULATED **
C*** USING QUADRATIC SPLINE INTERPOLATION TECHNIQUE (COUET'S FILTER). **
C***
C*****
C***
      PARAMETER (NAP=1)
      DOUBLE PRECISION G1,G2,P,SN,SS1,UG1,UG1E,UVDATA,VG1,VG1E,XYDATA
      COMMON/TIMER/DTIME,NFULL,NSTART,NSTOP,NTIME,TIME
C***
C*****

```

```

C***
      IF (NPV.EQ.0) RETURN
C***
      IF (NCASE.EQ.2.AND.NTIME.GT.NSTART) GO TO 1
      CALL XYUV (G1,G2,1,MCOUNT,0,P,SN,SS1,UPNT,UVDATA,VPNT,XPNT,
      *XYDATA,YPNT)
      IF (NAP.EQ.0) GO TO 2
1     CALL INITPV (F,GPNT,NCOUNT,UPNT,VPNT)
      CALL BORDER (F,NCOUNT)
      CALL HWSCRT (BDA,BDB,BDC,BDD,0.0,F,NCOUNT,PERTRB,IERROR,W)
      CALL GRIDV2 (F,NCOUNT,UG2,VG2)
C***
2     IF (NCASE.EQ.1) THEN
      CALL GRDV21 (GPNT,NAP,NCOUNT,UG1,UG1E,UG1S,UG2,VG1,VG1E,VG1S,VG2,
      *XPNT,XYDATA,YPNT)
C***
      IF (IFLG.EQ.1) THEN
      IF (NAP.EQ.0) GO TO 3
      CALL VORT (NCOUNT,UG2,VG2,Q)
3     CALL LAYGRD (IBOX,NCOUNT,NG,UG1S,UGN,XY,VG1S,VGN,XYDATA)
      END IF
C***
      CALL PNTV2 (1,NCOUNT,0,UG1S,UPNT,U,VG1S,VPNT,V)
      ELSE
      CALL PNTV2 (1,NCOUNT,0,UG2,UPNT,U,VG2,VPNT,V)
      END IF
C***
      RETURN
      END
C***
C***
C*****
      SUBROUTINE INITPV (F,GPNT,NCOUNT,UPNT,VPNT)
C*****
C***
C***          THIS SUBROUTINE SPREADS VORTICITY TO GRID POINTS
C***
C*****
C***
      COMMON/CONSNT/PI, SGMA, SGMA0, TWOPI
      COMMON/GRID1/DU,DUINVT,DV,DVINVT,HUINV,HVINV
      COMMON/GRID2/ULNGTH,UMN,UMX,VLNGTH,VMN,VMX
      COMMON/NSIZE/IMAX,IMID,IMIN,ISIZE,JMAX,JMID,JMIN,JSIZE
      COMMON/PTPAR1/GPVMN,GPVMX,LCODE,NO,NPV
      DIMENSION F (NCOUNT,*)
      VIRTUAL GPNT (*),UPNT (*),VPNT (*)
      EXTERNAL PHI
C***
C*****
C***-----
C*** Initialize the charge array :
C***-----
C***
      DO 1 J = JMIN,JMAX

```



```

      DO 1 I = IMIN,IMAX
      F(I,J) = 0.0
1 CONTINUE
C***
C***-----
C*** Locate the point vortices :
C***-----
C***
      FACTR = -TWOPI*DUINVT*DVINVT
      DO 60 N = 1,NPV
      GPOINT = FACTR*GPNT(N)
      HUP = (UPNT(N)-UMN)*DUINVT+1.
      HVP = (VPNT(N)-VMN)*DVINVT+1.
      ICOL = IFIX(HUP)
      JROW = IFIX(HVP)
      DCOL = HUP-FLOAT(ICOL)
      DROW = HVP-FLOAT(JROW)
C***
      IPV = IFIX(HUP+.5)-2
      JPV = IFIX(HVP+.5)-2
C***
      DO 50 II = 1,3
      IDELTA = IPV + II
      HUC = FLOAT(IDELTA)
      DHU = ABS(HUP-HUC)
C***
C***-----
C*** Check if outside the computational domain in the u-direction :
C***-----
C***
      IF(IDELTA.LT.IMIN) IDELTA = IMIN
      IF(IDELTA.GT.IMAX) IDELTA = IMAX
C***
      DO 40 JJ = 1,3
      JDELTA = JPV + JJ
      HVR = FLOAT(JDELTA)
      DHV = ABS(HVP-HVR)
C***
C***-----
C*** Check if outside the computational domain v-direction :
C***-----
C***
      IF(JDELTA.LT.JMIN) JDELTA = JMIN
      IF(JDELTA.GT.JMAX) JDELTA = JMAX
C***
      F(IDELTA,JDELTA) = F(IDELTA,JDELTA) + GPOINT*PHI(DHU,DHV)
40 CONTINUE
50 CONTINUE
60 CONTINUE
C***
      RETURN
      END
C***
C***

```

```

C*****
      SUBROUTINE BORDER(F,NCOUNT)
C*****
C***
C***          THIS SUBROUTINE SPECIFIES THE BOUNDARY CONDITION          **
C***
C*****
C***
      COMMON/NSIZE /IMAX,IMID,IMIN,ISIZE,JMAX,JMID,JMIN,JSIZE
      DIMENSION F(NCOUNT,*)
C***
C*****
C***-----
C***  Initialize the border of the F-array :
C***-----
C***
      DO 1 I = IMIN,IMAX,ISIZE
      DO 1 J = JMIN,JMAX
      F(I,J) = 0.0
1  CONTINUE
      DO 2 J = JMIN,JMAX,JSIZE
      DO 2 I = IMIN+1,ISIZE
      F(I,J) = 0.0
2  CONTINUE
C***
      RETURN
      END
C***
C***
C*****
      SUBROUTINE GRIDV2(F,NCOUNT,UG2,VG2)
C*****
C***
C***          THIS SUBROUTINE COMPUTES VELOCITIES AT GRID POINTS          **
C***          IN THE UV-PLANE USING A CENTERED DIFFERENCE TECHNIQUE          **
C***          WHICH ASSURES LOCAL CONSERVATION OF MASS                      **
C***
C*****
C***
      COMMON/GRID1/DU,DUINVT,DV,DVINVT,HDUINV,HDVINV
      COMMON/NSIZE/IMAX,IMID,IMIN,ISIZE,JMAX,JMID,JMIN,JSIZE
      DIMENSION F(NCOUNT,*)
      VIRTUAL   UG2(NCOUNT,*),VG2(NCOUNT,*)
C***
C*****
C***-----
C***  Initialize arrays
C***-----
C***
      DO 50 J      = JMIN,JMAX
      DO 50 I      = IMIN,IMAX
      UG2(I,J)     = 0.0
      VG2(I,J)     = 0.0
50  CONTINUE

```

```

C***
C***-----
C*** Interior grid points :
C***-----
C***
      DO 1 J      = JMIN+1,JSIZE
      DO 1 I      = IMIN+1,ISIZE
      UG2(I,J)    = UG2(I,J)+(F(I,J+1)-F(I,J-1))*HDVINV
      VG2(I,J)    = VG2(I,J)+(F(I-1,J)-F(I+1,J))*HDIINV
1 CONTINUE
C***
C***-----
C*** Lower boundary grid points :
C***-----
C***
      DO 3 I      = IMIN,IMAX
      UG2(I,JMIN) = UG2(I,JMIN)+2.0*F(I,JMIN+1)*HDVINV
      IF(I.EQ.IMIN) VG2(I,JMIN)=VG2(I,JMIN)-2.0*F(I+1,JMIN)*HDIINV
      IF(I.EQ.IMAX) VG2(I,JMIN)=VG2(I,JMIN)+2.0*F(I-1,JMIN)*HDIINV
      IF(I.GT.IMIN.AND.I.LT.IMAX) VG2(I,JMIN) = VG2(I,JMIN) +
      *
      (F(I-1,JMIN)-F(I+1,JMIN))*HDIINV
C***
3 CONTINUE
C***
C***-----
C*** Upper boundary grid points :
C***-----
C***
      DO 4 I      = IMIN,IMAX
      UG2(I,JMAX) = UG2(I,JMAX)-2.0*F(I,JMAX-1)*HDVINV
      IF(I.EQ.IMIN) VG2(I,JMAX)=VG2(I,JMAX)-2.0*F(I+1,JMAX)*HDIINV
      IF(I.EQ.IMAX) VG2(I,JMAX)=VG2(I,JMAX)+2.0*F(I-1,JMAX)*HDIINV
      IF(I.GT.IMIN.AND.I.LT.IMAX) VG2(I,JMAX) = VG2(I,JMAX) +
      *
      (F(I-1,JMAX)-F(I+1,JMAX))*HDIINV
4 CONTINUE
C***
C***-----
C*** Left boundary grid points :
C***-----
C***
      DO 5 J      = JMIN+1,JSIZE
      UG2(IMIN,J) = UG2(IMIN,J)+(F(IMIN,J+1)-F(IMIN,J-1))*HDVINV
      VG2(IMIN,J) = VG2(IMIN,J)-2.0*F(IMIN+1,J)*HDIINV
5 CONTINUE
C***
C***-----
C*** Right boundary grid points :
C***-----
C***
      DO 6 J      = JMIN+1,JSIZE
      UG2(IMAX,J) = UG2(IMAX,J)+(F(IMAX,J+1)-F(IMAX,J-1))*HDVINV
      VG2(IMAX,J) = VG2(IMAX,J)+2.0*F(IMAX-1,J)*HDIINV
6 CONTINUE
C***

```

```

      RETURN
      END

C***
C***
C*****
      SUBROUTINE GRDV21 (GPNT, NAP, NCOUNT, UG1, UG1E, UG1S, UG2, VG1, VG1E,
        *VG1S, VG2, XPNT, XYDATA, YPNT)
C*****
C***
C***      THIS SUBROUTINE COMPUTES GRID VELOCITIES IN THE CIRCLE DOMAIN
C***
C*****
C***
      DOUBLE PRECISION E11, E12, UG1, UG1E, VG1, VG1E, XYDATA
      COMMON / NSIZE, IMAX, IMIN, ISIZE, JMAX, JMID, JMIN, JSIZE
      VIRTUAL UG2 (NCOUNT, *), VG2 (NCOUNT, *)
      VIRTUAL UG1 (NCOUNT, *), UG1E (NCOUNT, *), UG1S (NCOUNT, *)
      VIRTUAL VG1 (NCOUNT, *), VG1E (NCOUNT, *), VG1S (NCOUNT, *)

C***
C*****
C***-----
C***      Compute exact solution
C***-----
C***
      CALL EXACT (GPNT, NCOUNT, NAP, UG1E, UG1S, VG1E, VG1S, XPNT, XYDATA, YPNT)
      IF (NAP.EQ.0) RETURN

C***
      OPEN (UNIT=3, FILE='MATRCES.CLNS', TYPE='OLD')
      DO 2 J = JMIN, JMAX
      DO 2 I = IMIN, IMAX
      READ (3, 1) E11, E12
1  FORMAT (' ', T2, 2 (2X, 1PD15.7))
      UG1 (I, J) = E11*DBLE (UG2 (I, J)) + E12*DBLE (VG2 (I, J))
      VG1 (I, J) = -E12*DBLE (UG2 (I, J)) + E11*DBLE (VG2 (I, J))
      IF (NAP.EQ.1) THEN
      IF (I.EQ.IMIN.OR.I.EQ.IMAX) THEN
      IF (J.EQ.JMIN.OR.J.EQ.JMAX) THEN
      UG1 (I, J) = UG1E (I, J)
      VG1 (I, J) = VG1E (I, J)
      END IF
      END IF
      END IF
      UG1S (I, J) = SNGL (UG1 (I, J))
      VG1S (I, J) = SNGL (VG1 (I, J))
2  CONTINUE
      CLOSE (UNIT=3)

C***
      RETURN
      END

C***
C***
C*****
      SUBROUTINE EXACT (GPNT, NCOUNT, NAP, UG1E, UG1S, VG1E, VG1S, XPNT, XYDATA,
        *YPNT)

```

```

C*****
C***
C***          THIS SUBROUTINE COMPUTES EXACT VELOCITIES          **
C***          USING A MODIFIED MILNE-THOMSON CIRCLE THEOREM      **
C***
C*****
C***
      DOUBLE PRECISION UG1E,VG1E,XYDATA
      COMMON/CONSNT/PI,SGMA,SGMA0,TWOPI
      COMMON/NSIZE/IMAX,IMID,IMIN,ISIZE,JMAX,JMID,JMIN,JSIZE
      COMMON/PTPAR1/GPVMN,GPVMX,LCODE,NO,NPV
      COMMON/PTPAR2/CNST1,H2,RW,SGMA2I,TOL
      VIRTUAL GPNT(*),XYDATA(2,*)
      VIRTUAL UG1S(NCOUNT,*),VG1S(NCOUNT,*)
      VIRTUAL UG1E(NCOUNT,*),VG1E(NCOUNT,*),XPNT(*),YPNT(*)
C***
C*****
C***-----
C*** Initialize the arrays :
C***-----
C***
      DO 1 J = JMIN,JMAX
      DO 1 I = IMIN,IMAX
      UG1E(I,J) = 0.0D0
      VG1E(I,J) = 0.0D0
      UG1S(I,J) = 0.0
      VG1S(I,J) = 0.0
1 CONTINUE
C***
      IF (NAP.EQ.0) THEN
      JNC      = 1
      INC      = 1
      ELSE
      JNC      = JSIZE
      INC      = ISIZE
      END IF
C***
      DO 3 N = 1, NPV
      A02      = XPNT(N)*XPNT(N)+YPNT(N)*YPNT(N)
C***
      DO 2 J = JMIN,JMAX,JNC
      DO 2 I = IMIN,IMAX,INC
      K      = I + (J-JMIN)*JMAX
C***
C***-----
C*** Initialize parameters for u,v velocities :
C***-----
C***
      AX      = A02*SNGL(XYDATA(1,K))-(RW*RW)*XPNT(N)
      AY      = A02*SNGL(XYDATA(2,K))-(RW*RW)*YPNT(N)
      AX2     = AX*AX
      AY2     = AY*AY
      XMX0    = SNGL(XYDATA(1,K))-XPNT(N)
      YMY0    = SNGL(XYDATA(2,K))-YPNT(N)

```

```

      XMX02      = XMX0*XMX0
      YMY02      = YMY0*YMY0
      DENOM1     = XMX02+YMY02
      DENOM2     = AX2+AY2
      IF (DENOM2.EQ.0.0) DENOM2 = SGMA0
      SQRTD      = SQRT (DENOM1)
      H1         = (3.0*CNST1*SQRTD)/SGMA

C***
C***-----
C***  Remove singularity according to Hald & Del Prete's formulation :
C***-----
C***
      IF (SQRTD.LE.SGMA) THEN
      UTMP       = GPNT (N) * ( (A02*AY/DENOM2) - SGMA2I * (H2-H1) * YMY0)
      VTMP       = GPNT (N) * ( -(A02*AX/DENOM2) + SGMA2I * (H2-H1) * XMX0)
      ELSE
      UTMP       = GPNT (N) * ( (A02*AY/DENOM2) - (YMY0/DENOM1) )
      VTMP       = GPNT (N) * ( -(A02*AX/DENOM2) + (XMX0/DENOM1) )
      END IF

C***
C***-----
C***  Compute the velocity components :
C***-----
C***
      UG1E (I,J) = UG1E (I,J) + DBLE (UTMP)
      VG1E (I,J) = VG1E (I,J) + DBLE (VTMP)

C***
      IF (NAP.EQ.0) THEN
      UG1S (I,J) = SNGL (UG1E (I,J) )
      VG1S (I,J) = SNGL (VG1E (I,J) )
      END IF

C***
      2 CONTINUE
      3 CONTINUE

C***
      RETURN
      END

C***
C***
C*****
      SUBROUTINE VORT (NCOUNT,UG2,VG2,Q)
C*****
C***
C***      THIS SUBROUTINE COMPUTES VORTICITY AT GRID POINTS IN
C***      THE UV-PLANE USING A CENTERED DIFFERENCE TECHNIQUE
C***      WHICH ASSURES LOCAL CONSERVATION OF VORTICITY
C***
C*****
C***
      COMMON/GRID1/DU,DUINVT,DV,DVINVT,HDUINV,HDVINV
      COMMON/NSIZE/IMAX,IMID,IMIN,ISIZE,JMAX,JMID,JMIN,JSIZE
      VIRTUAL  Q (NCOUNT,*),UG2 (NCOUNT,*),VG2 (NCOUNT,*)

C***
C*****

```

```

C***-----
C***  Initialize array
C***-----
C***
      DO 50 J = JMIN,JMAX
      DO 50 I = IMIN,IMAX
      Q(I,J) = 0.0
50 CONTINUE
C***
C***-----
C***  Interior grid points :
C***-----
C***
      DO 1 J = JMIN+1,JSIZE
      DO 1 I = IMIN+1,ISIZE
      DUDY = (UG2(I,J+1)-UG2(I,J-1))*HDVINV
      DVDX = (VG2(I+1,J)-VG2(I-1,J))*HDIINV
      Q(I,J) = DVDX-DUDY
1 CONTINUE
C***
C***-----
C***  Lower boundary grid points :
C***-----
C***
      DO 3 I = IMIN,IMAX
      DUDY = 2.0*UG2(I,JMIN+1)*HDVINV
      IF(I.EQ.IMIN) DVDX = 2.0*VG2(I+1,JMIN)*HDIINV
      IF(I.EQ.IMAX) DVDX = -2.0*VG2(I-1,JMIN)*HDIINV
      IF(I.GT.IMIN.AND.I.LT.IMAX) DVDX = (VG2(I+1,JMIN)-VG2(I-1,JMIN))*
      *HDIINV
      Q(I,JMIN) = DVDX-DUDY
C***
3 CONTINUE
C***
C***-----
C***  Upper boundary grid points :
C***-----
C***
      DO 4 I = IMIN,IMAX
      DUDY = -2.0*UG2(I,JMAX-1)*HDVINV
      IF(I.EQ.IMIN) DVDX = 2.0*VG2(I+1,JMAX)*HDIINV
      IF(I.EQ.IMAX) DVDX = -2.0*VG2(I-1,JMAX)*HDIINV
      IF(I.GT.IMIN.AND.I.LT.IMAX) DVDX = (VG2(I+1,JMAX)-VG2(I-1,JMAX))*
      *HDIINV
      Q(I,JMAX) = DVDX-DUDY
4 CONTINUE
C***
C***-----
C***  Left boundary grid points :
C***-----
C***
      DO 5 J = JMIN+1,JSIZE
      DUDY = (UG2(IMIN,J+1)-UG2(IMIN,J-1))*HDVINV
      DVDX = 2.0*VG2(IMIN+1,J)*HDIINV

```

```

      Q(IMIN,J) = DVDX-DUDY
5  CONTINUE
C***
C***-----
C***  Right boundary grid points :
C***-----
C***
      DO 6 J = JMIN+1,JSIZE
      DUDY = (UG2(IMAX,J+1)-UG2(IMAX,J-1))*HDVINV
      DVDX = -2.0*VG2(IMAX-1,J)*HDIINV
      Q(IMAX,J) = DVDX-DUDY
6  CONTINUE
C***
      RETURN
      END
C***
C***
C*****
      SUBROUTINE LAYGRD(IBOX,NCOUNT,NG,UG1S,UGN,XY,VG1S,VGN,XYDATA)
C*****
C***
C***          THIS SUBROUTINE COMPUTES VELOCITIES ON UNIFORM
C***          GRID POINTS SUPERIMPOSED ON THE CIRCULAR DOMAIN
C***
C*****
C***
      DOUBLE PRECISION XYDATA
      COMMON/GRID3/DX,DXINVT,DY,DYINVT,XMN,XX,YY,YYM,YYM
      COMMON/NSIZE/IMAX,IMID,IMIN,ISIZE,JMAX,JMID,JMIN,JSIZE
      COMMON/TIMER/DTIME,NFULL,NSTART,NSTOP,NTIME,TIME
      EXTERNAL PHI
      VIRTUAL UG1S(NCOUNT,*),UGN(NCOUNT,*),XY(2,*)
      VIRTUAL VG1S(NCOUNT,*),VGN(NCOUNT,*),XYDATA(2,*)
      VIRTUAL IBOX(NCOUNT,*),NG(NCOUNT,*)
C***
C*****
C***-----
C***  Initialize the arrays/parameters
C***-----
C***
      DO 1 J = JMIN,JMAX
      DO 1 I = IMIN,IMAX
      UGN(I,J) = 0.0
      VGN(I,J) = 0.0
      NG(I,J) = 0
1  CONTINUE
C***
C***-----
C***  Assign radial location of singularity (RS)
C***  and velocity reduction factor (RF)
C***-----
C***
      RS = 0.82
      RF = 0.4

```



```

C***
C***-----
C***  Tag grid points outside circle
C***-----
C***
      IF (NTIME.EQ.NSTART) THEN
      DO 2 J = JMIN,JMAX
      Y    = DY*FLOAT(J-JMIN)+YMN
      DO 2 I = IMIN,IMAX
      X    = DX*FLOAT(I-IMIN)+XMN
      K    = I + (J-JMIN)*JMAX
      IBOX(I,J) = 0
      XY(1,K) = X
      XY(2,K) = Y
      R      = SQRT(X*X+Y*Y)
      IF (R.GT.XMX) IBOX(I,J) = 1
2 CONTINUE
      END IF
C***
C***-----
C***  Interpolate velocities to grid points
C***-----
C***
      DO 70 NS = 1,2
C***
      DO 60 J = JMIN,JMAX
      DO 60 I = IMIN,IMAX
      K    = I + (J-JMIN)*JMAX
C***
C***-----
C***  Locate vortex position :
C***-----
C***
      HXP    = (SNGL(XYDATA(1,K))-XMN)*DXINVT+1.
      HYP    = (SNGL(XYDATA(2,K))-YMN)*DYINVT+1.
      ICOL   = IFIX(HXP)
      JROW   = IFIX(HYP)
      DCOL   = HXP-FLOAT(ICOL)
      DROW   = HYP-FLOAT(JROW)
C***
      IPV    = IFIX(HXP+.5)-2
      JPV    = IFIX(HYP+.5)-2
C***
C***-----
C***  Determine argument, dx, for the weighting function :
C***-----
C***
      DO 50 II = 1,3
      IDELTA = IPV + II
      IF (NS.EQ.2) GO TO 3
      HXC    = FLOAT(IDELTA)
      DHX    = ABS(HXP-HXC)
C***
C***-----

```

```

C*** Check if outside the computational domain
C***-----
C***
      3 IF(IDELTA.LT.IMIN) IDELTA = IMIN
      IF(IDELTA.GT.IMAX) IDELTA = IMAX
C***
C***-----
C*** Determine argument, dy, for the weighting function :
C***-----
C***
      DO 40 JJ = 1,3
      JDELTA      = JPV + JJ
      IF(NS.EQ.2) GO TO 4
      HYR         = FLOAT(JDELTA)
      DHY         = ABS(HYP-HYR)
C***
C***-----
C*** Check if outside the computational domain
C***-----
C***
      4 IF(JDELTA.LT.JMIN) JDELTA = JMIN
      IF(JDELTA.GT.JMAX) JDELTA = JMAX
C***
C***-----
C*** Check if grid point is within circle
C***-----
C***
      IF(IBOX(IDELTA,JDELTA).EQ.1) GO TO 40
C***
      IF(NS.EQ.2) GO TO 5
C***
C***-----
C*** Interpolate velocities to grid points
C***-----
C***
      UGN(IDELTA,JDELTA)=UGN(IDELTA,JDELTA)+UG1S(I,J)*PHI(DHX,DHY)
      VGN(IDELTA,JDELTA)=VGN(IDELTA,JDELTA)+VG1S(I,J)*PHI(DHX,DHY)
C***
      5 IF(NS.EQ.2) THEN
C***
C***-----
C*** Modify velocities within singularity zone
C***-----
C***
C***-----
C*** Check (x,y) coordinates of grid points
C***-----
C***
      X          = (IDELTA-1)*DX + XMN
      Y          = (JDELTA-1)*DY + YMN
C***
C***-----
C*** Check radial location of grid points
C***-----

```

```

C***
      R          = SQRT(X*X+Y*Y)
C***
      IF(R.GE.RS) THEN
C***
C***-----
C***  Check angular location of grid points
C***-----
C***
      T          = ATAN2D(Y,X)
      IF(Y.LT.0.0) T = T + 360.0
C***
      IF((T.GE.35.0.AND.T.LE.55.0) .OR.
        *(T.GE.125.0.AND.T.LE.145.0) .OR.
        *(T.GE.215.0.AND.T.LE.235.0) .OR.
        *(T.GE.305.0.AND.T.LE.325.0)) THEN
C***
C***-----
C***  Check if duplicating. If not tag grid point
C***-----
C***
      IF(NG(IDELTA,JDELTA).EQ.1) GO TO 40
      NG(IDELTA,JDELTA) = 1
C***
C***-----
C***  Modify grid velocity :
C***-----
C***
      UGN(IDELTA,JDELTA) = RF*UGN(IDELTA,JDELTA)
      VGN(IDELTA,JDELTA) = RF*VGN(IDELTA,JDELTA)
C***
      END IF
      END IF
      END IF
C***
      40 CONTINUE
      50 CONTINUE
      60 CONTINUE
      70 CONTINUE
C***
      RETURN
      END
C***
C***
C*****
      SUBROUTINE PNTV2(IFLG,NCOUNT,NTP,UGC,UPT,U,VGC,VPT,V)
C*****
C***
C***  THIS SUBROUTINE CALCULATES VELOCITIES AT OFF GRID LOCATIONS
C***
C*****
C***
      COMMON/GRID1/DU,DUINVT,DV,DVINVT,HDUINV,HDVINV
      COMMON/GRID2/ULNGTH,UMN,UMX,VLNGTH,VMN,VMX

```

```

COMMON/NSIZE/IMAX, IMID, IMIN, ISIZE, JMAX, JMID, JMIN, JSIZE
COMMON/PTPAR1/GPVMN, GPVMX, LCODE, NO, NPV
VIRTUAL UPT(*), VPT(*)
VIRTUAL UGC(NCOUNT, *), U(*), VGC(NCOUNT, *), V(*)
EXTERNAL PHI

```

```

C***

```

```

C*****

```

```

C***

```

```

      IF (IFLG.EQ.1) NP = NPV
      IF (IFLG.EQ.2) NP = NTP
      DO 90 N = 1, NP
        U(N) = 0.
        V(N) = 0.
        UTEMP = 0.
        VTEMP = 0.
        HUP = (UPT(N)-UMN)*DUINVT+1.
        HVP = (VPT(N)-VMN)*DVINVT+1.

```

```

C***

```

```

      ICOL = IFIX(HUP)
      JROW = IFIX(HVP)
      DCOL = HUP-FLOAT(ICOL)
      DROW = HVP-FLOAT(JROW)

```

```

C***

```

```

      IPV = IFIX(HUP+.5)-2
      JPV = IFIX(HVP+.5)-2

```

```

C***

```

```

      DO 80 II = 1, 3
        IDELTA = IPV + II
        HUC = FLOAT(IDELTA)
        DHU = ABS(HUP-HUC)

```

```

C***

```

```

C***-----

```

```

C*** Check if outside the computational domain :

```

```

C***-----

```

```

C***

```

```

      IF (IDELTA.LT.IMIN) IDELTA = IMIN
      IF (IDELTA.GT.IMAX) IDELTA = IMAX

```

```

C***

```

```

      DO 70 JJ = 1, 3
        JDELTA = JPV + JJ
        HVR = FLOAT(JDELTA)
        DHV = ABS(HVP-HVR)

```

```

C***

```

```

C***-----

```

```

C*** Check if outside the computational domain :

```

```

C***-----

```

```

C***

```

```

      IF (JDELTA.LT.JMIN) JDELTA = JMIN
      IF (JDELTA.GT.JMAX) JDELTA = JMAX

```

```

C***

```

```

      UTEMP = UTEMP + UGC(IDELTA, JDELTA)*PHI(DHU, DHV)
      VTEMP = VTEMP + VGC(IDELTA, JDELTA)*PHI(DHU, DHV)

```

```

70 CONTINUE

```

```

80 CONTINUE

```

```

C***
C***-----
C***  Store the velocity
C***-----
C***
      U(N)      = UTEMP
      V(N)      = VTEMP
      90 CONTINUE
C***
      RETURN
      END
C***
C***
C*****
      SUBROUTINE PRNT(F,FTAG,NCASE,NCOUNT,Q,RPNT,UG1S,UG2,UGN,
      *UVDATA,VG1S,VG2,VGN,XPNT,XY,XYDATA,YPNT)
C*****
C***
C***      THIS SUBROUTINE PRINTS RESULTS OF VORTEX CALCULATION
C***
C***
C*****
C***
      DOUBLE PRECISION RK,RMOD2,UVDATA,XX,YY,XYDATA
      PARAMETER(NGRD=0,NPOS=0,TANG=1.0,XDIR=1.0)
      COMMON/CONSNT/PI,SGMA,SGMA0,TWOPI
      COMMON/JETPAR/DJ,PMSIGN,STOPER,TJ,VIN,XA,XB,XPA,XPB,YAB,YB,YPB
      COMMON/MAP/RK,RMOD2,XX,YY
      COMMON/NSIZE/IMAX,IMID,IMIN,ISIZE,JMAX,JMID,JMIN,JSIZE
      COMMON/PTPAR1/GPVMN,GPVMX,LCODE,NO,NPV
      COMMON/PTPAR2/CNST1,H2,RW,SGMA2I,TOL
      COMMON/TIMER/DTIME,NFULL,NSTART,NSTOP,NTIME,TIME
      DIMENSION F(NCOUNT,*)
      VIRTUAL  UVDATA(2,*,XY(2,*,XYDATA(2,*)
      VIRTUAL  FTAG(2,*,XPNT(*),YPNT(*),RPNT(*))
      VIRTUAL  Q(NCOUNT,*,UG1S(NCOUNT,*,VG1S(NCOUNT,*)
      VIRTUAL  UGN(NCOUNT,*,UG2(NCOUNT,*,VGN(NCOUNT,*,VG2(NCOUNT,*)
C***
C*****
C***
      IF(NPV.EQ.0) GO TO 300
C***
C***-----
C***  NON-GRID POINT DATA :
C***-----
C***
      IF(NPOS.EQ.1) THEN
      IF(NCASE.EQ.1) THEN
      WRITE(1,108) NTIME
      WRITE(2,108) NTIME
      DO 1  N = 1,NPV
D      IF(FTAG(1,N).EQ.1.0) THEN
D      WRITE(1,110) XPNT(N),YPNT(N),NTIME*DTIME,RPNT(N)
D      ELSE
D      WRITE(2,110) XPNT(N),YPNT(N),NTIME*DTIME,RPNT(N)

```

```

D      END IF
D      1 CONTINUE
D      END IF
D      END IF
      108 FORMAT(' ',T2,'NTIME =',T10,I5,/)
D 110 FORMAT(' ',T2,1PE15.7,T19,1PE15.7,T37,1PE15.7,T54,1PE15.7)
C***
C***-----
C***  GRID POINT VELOCITY DATA :
C***-----
C***
      IF (NGRD.EQ.1.0) THEN
C***
      DO 2 NLOOP = 1,2
      IF (NLOOP.EQ.1) THEN
      J1          = JMID
      J2          = JMID
      J3          = 1
      I3          = 10
      ELSE
      J1          = JMAX
      J2          = JMIN
      J3          = -2
      I3          = 2
      END IF
C***
      DO 131 J = J1,J2,J3
      DO 131 I = IMIN,IMAX,I3
      K          = I + (J-JMIN)*JMAX
      IF (NLOOP.EQ.1) THEN
      IF (XDIR.EQ.0.0) K = IMID + (I-JMIN)*JMAX
      IF (TANG.EQ.1.0) THEN
      IF (SNGL(XYDATA(1,K)).EQ.0.0.AND.SNGL(XYDATA(2,K)).EQ.0.0) THEN
      T          = ATAN2(SNGL(XYDATA(2,K)),SGMA0)
      ELSE
      T          = ATAN2(SNGL(XYDATA(2,K)),SNGL(XYDATA(1,K)))
      END IF
      VTC        = ABS(VG1S(I,J))*COS(T)-ABS(UG1S(I,J))*SIN(T)
      VRC        = -ABS(VG1S(I,J))*SIN(T)-ABS(UG1S(I,J))*COS(T)
C***
      ELSE
C***
      VTC        = UG1S(I,J)
      VRC        = VG1S(I,J)
      END IF
C***
      IF (I.EQ.IMIN.AND.J.EQ.J1) THEN
      IF (NCASE.EQ.1) WRITE(3,3)  NTIME
      IF (NCASE.EQ.2) WRITE(7,3)  NTIME
      END IF
C***
      IF (XDIR.EQ.1.0) THEN
      IF (NCASE.EQ.1) THEN
      WRITE(3,4) SNGL(XYDATA(1,K)),SNGL(XYDATA(2,K)),F(I,J),

```

```

*VTC,VRC,Q(I,J)
  ELSE
    WRITE(7,4) SNGL(UVDATA(1,K)/RK),SNGL(UVDATA(2,K)/RK),
    *F(I,J),VG2(I,J),UG2(I,J),Q(I,J)
  END IF
C***
  ELSE
C***
    IF(NCASE.EQ.1) THEN
      WRITE(3,4) SNGL(XYDATA(1,K)),SNGL(XYDATA(2,K)),F(I,J),
      *VTC,VRC,Q(I,J)
    ELSE
      WRITE(7,4) SNGL(UVDATA(1,K)/RK),SNGL(UVDATA(2,K)/RK),
      *F(I,J),UG2(I,J),VG2(I,J),Q(I,J)
    END IF
  END IF
C***
  ELSE
C***
    IF(I.EQ.IMIN.AND.J.EQ.J1) THEN
      IF(NCASE.EQ.1) THEN
        WRITE(4,3) NTIME
        WRITE(6,3) NTIME
      ELSE
        WRITE(8,3) NTIME
      END IF
    END IF
C***
    IF(NCASE.EQ.1) THEN
C    WRITE(4,4) SNGL(XYDATA(1,K)),SNGL(XYDATA(2,K)),F(I,J),
C    *UG1S(I,J),VG1S(I,J),Q(I,J)
      WRITE(6,4) XY(1,K),XY(2,K),F(I,J),UGN(I,J),VGN(I,J),
      *Q(I,J)
    ELSE
      WRITE(8,4) SNGL(UVDATA(1,K)),SNGL(UVDATA(2,K)),
      *F(I,J),UG2(I,J),VG2(I,J),Q(I,J)
    END IF
  END IF
C***
  131 CONTINUE
  2 CONTINUE
  3 FORMAT(' ',/,T2,'NTIME =', I5,/)
  4 FORMAT(' ',T2,1PE15.7,T20,1PE15.7,T38,1PE15.7,T56,1PE15.7,T74,
    *1PE15.7,T92,1PE15.7,T110,1PE15.7)
C***
  END IF
C***
  300 RETURN
  END
C***
C***
C*****
  SUBROUTINE ADVECT(BDA,BDB,BDC,BDD,F,FTAG,G1,G2,GPNT,IBOX,JERT,
    *MCOUNT,NCASE,NCOUNT,NG,P,Q,RPNT,SN,SS1,TPNT,UG1,UG1E,UG1S,UG2,

```

```

      *UGN, UN, UO, UPNT, UVDATA, VG1, VG1E, VG1S, VG2, VGN, VN, VO, VPNT, XPN, XPO,
      *YPN, YPO, W, XY, XYDATA)
C*****
C***
C***          THIS SUBROUTINE ADVECTS THE VORTICES
C***          USING HEUN'S METHOD WHICH IS SECOND-ORDER ACCURATE IN TIME,
C***          AS OPPOSED TO EULER'S METHOD WHICH IS FIRST-ORDER ACCURATE
C***          (Ref. "Turbulent Combustion in Open and Closed Vessels", by
C***James Sethian, Journal of Computational Physics 54, 425-456 (1984))
C***
C*****
C***          DOUBLE PRECISION G1,G2,P,SN,SS1,UG1,UG1E,UVDATA,VG1,VG1E,XYDATA
          COMMON/CONSNT/PI, SGMA, SGMA0, TWOPI
          COMMON/JETPAR/DJ, PMSIGN, TJ, VIN, XA, XB, XPA, XPB, YAB, YB, YPB
          COMMON/GRID2/ULNGTH, UMN, UMX, VLNGTH, VMN, VMX
          COMMON/TIMER/DTIME, NFULL, NSTART, NSTOP, NTIME, TIME
          COMMON/PTPAR1/GPVMN, GPVMX, LCODE, NO, NPV
          COMMON/PTPAR2/CNST1, H2, RW, SGMA2I, TOL
          VIRTUAL RPNT(*), TPNT(*), UPNT(*), VPNT(*), FTAG(2,*)
          VIRTUAL UN(*), UO(*), VN(*), VO(*), XPN(*), XPO(*), YPN(*), YPO(*)
C***
C*****
C***-----
C***  Advect the point vortices :
C***-----
C***
          IF (NPV.EQ.0) GO TO 3
          DO 1 N = 1, NPV
          IF (NCASE.EQ.1) THEN
          XPN(N) = XPO(N) + UO(N)*DTIME
          YPN(N) = YPO(N) + VO(N)*DTIME
C***
C***-----
C***  Check if outside the computational domain :
C***-----
C***
          IF (XPN(N).EQ.0.0.AND.YPN(N).EQ.0.0) THEN
          TPNT(N) = ATAN2 (YPN(N), SGMA0)
          ELSE
          TPNT(N) = ATAN2 (YPN(N), XPN(N))
          END IF
          RPNT(N) = SQRT (XPN(N)*XPN(N)+YPN(N)*YPN(N))
C***
          IF (RPNT(N).GT.RW) THEN
          RPNT(N) = RW
          XPN(N) = RPNT(N)*COS (TPNT(N))
          YPN(N) = RPNT(N)*SIN (TPNT(N))
          END IF
C***
          ELSE
C***
          XPO(N) = UPNT(N)
          YPO(N) = VPNT(N)

```



```

      UPNT(N) = UPNT(N) + UO(N)*DTIME
      VPNT(N) = VPNT(N) + VO(N)*DTIME
C***
C***-----
C*** Check if outside the computational domain :
C***-----
C***
      IF (UPNT(N).LT.UMN) UPNT(N) = UMN
      IF (UPNT(N).GT.UMX) UPNT(N) = UMX
      IF (VPNT(N).LT.VMN) VPNT(N) = VMN
      IF (VPNT(N).GT.VMX) VPNT(N) = VMX
      END IF
C***
      1 CONTINUE
C***
C***-----
C*** Compute velocities at new vortex positions :
C***-----
C***
      CALL VIC(BDA,BDB,BDC,BDD,F,G1,G2,GPNT,IBOX,2,JERT,MCOUNT,
      *NCASE,NCOUNT,NG,P,Q,SN,SS1,UGN,UG1,UG1E,UG1S,UG2,UPNT,UN,UVDATA,
      *VGN,VG1,VG1E,VG1S,VG2,VPNT,VN,W,XPN,XY,XYDATA,YPN)
C***
C***-----
C*** Advect point vortices with average velocities :
C***-----
C***
      DO 2 N = 1,NPV
      UO(N) = 0.5*(UO(N)+UN(N))
      VO(N) = 0.5*(VO(N)+VN(N))
      XPO(N) = XPO(N)+UO(N)*DTIME
      YPO(N) = YPO(N)+VO(N)*DTIME
C***
C***-----
C*** Check if outside the computational domain :
C***-----
C***
      IF (NCASE.EQ.1) THEN
      IF (XPO(N).EQ.0.0.AND.YPO(N).EQ.0.0) THEN
      TPNT(N) = ATAN2(YPO(N),SGMA0)
      ELSE
      TPNT(N) = ATAN2(YPO(N),XPO(N))
      END IF
      RPNT(N) = SQRT(XPO(N)*XPO(N)+YPO(N)*YPO(N))
C***
      IF (RPNT(N).GT.RW) THEN
      RPNT(N) = RW
      XPO(N) = RPNT(N)*COS(TPNT(N))
      YPO(N) = RPNT(N)*SIN(TPNT(N))
      END IF
C***
C***-----
C*** Compute the tangential & radial velocities of the point vortices :
C***-----

```

```

C***
      VT(N)      =  VO(N)*COS(TPNT(N))-UO(N)*SIN(TPNT(N))
      VR(N)      = -VO(N)*SIN(TPNT(N))-UO(N)*COS(TPNT(N))
C***
      ELSE
C***
      IF(XPO(N).LT.UMN) XPO(N) = UMN
      IF(XPO(N).GT.UMX) XPO(N) = UMX
      IF(YPO(N).LT.VMN) YPO(N) = VMN
      IF(YPO(N).GT.VMX) YPO(N) = VMX
      UPNT(N)      = XPO(N)
      VPNT(N)      = YPO(N)
      END IF
2 CONTINUE
C***
3 RETURN
END
C***
C***
C*****
      SUBROUTINE UPDATE(MAXPV)
C*****
C***
C***          THIS SUBROUTINE UPDATES PARAMETERS/ARRAY SIZE          **
C***
C***
C*****
C***
      COMMON/PTPAR1/GPVMN,GPVMX,LCODE,NO,NPV
C***
C*****
C***-----
C*** Monitor array size :
C***-----
C***
      IF(NPV.GT.MAXPV) THEN
      LCODE = 0
      WRITE(5,1) NPV,MAXPV
1  FORMAT(' ',T12,'TOO MANY VORTICES CREATED :',/,T12,
*'NPV = ',I5,' (MAXPV = ',I5,')',/,T12,
*'PLEASE ADJUST ARRAYS AND RESTART CALCULATION')
      END IF
C***
      RETURN
      END
C***
C***
C*****
      FUNCTION PHI(DHU, DHV)
C*****
C***
C***          THIS FUNCTION ROUTINE EVALUATES THE INTERPOLATION FUNCTIONS          **
C***
C***
C*****
C***

```

```

DATA A, B, C, D, E / 1.763668E-03, -3.395062E-02, 2.573854E-01,
& -7.229663E-01, 6.651786E-01/
DATA ALPHA / 1.55408084 /
DATA CTAN / 0.50875376 /
DATA CONST/ 0.51438377 /
C***
DHU2 = DHU*DHU
DHV2 = DHV*DHV
C***
C***-----
C*** Filter developed by Couet ( Quadratic Spline Interpolation )
C***-----
C***
IF( DHU2.GE.2.25.OR.DHV2.GE.2.25 ) GO TO 7
IF( DHU2.LT.0.25 ) PHIU = 0.75 - DHU2
IF( DHU2.GE.0.25 ) PHIU = 0.5*((1.5-ABS(DHU))**2)
IF( DHV2.LT.0.25 ) PHIV = 0.75 - DHV2
IF( DHV2.GE.0.25 ) PHIV = 0.5*((1.5-ABS(DHV))**2)
PHI = PHIU*PHIV
RETURN
C***
C***-----
C*** Zero value for the filter
C***-----
C***
7 PHI = 0.0
RETURN
END
C***

```

```

C***
C*****
C***** DMAP LIBRARY CONTAINS THE FOLLOWING SUBROUTINES/FUNCTIONS : **
C***** **
C*** SUBROUTINES : **
C*** 1. SSN1 - Evaluates Jacobian elliptic functions SN **
C*** 2. XYUV - Performs coord. transformation from (x,y) to (u,v) **
C*** 3. NAS2 - Prepares input parameters for NASNWR **
C*** NASNWR - Solves a system of nonlinear algebraic equations **
C*** based on the Newton-Raphson method **
C*** F - Provides the system of nonlinear algebraic eqns. **
C*** NASJME - Evaluates the function Jacobi matrix **
C*** NASFNE - Evaluates the function residuals and their norm **
C*** NASABX - Solves A*X=B by LU decomp. & backwards substitution **
C*** DECOM - Does the decomposition **
C*** SUBST - Does the substitution **
C*** MOVEDD - Moves elements of a vector to another vector **
C*** NORMSD - Computes the Euclidean norm of a vector **
C*** EXCHSD - Swaps elements of two vectors **
C*** VIPASD - Calc. a vector inner product & adds to given value **
C*** 4. INTPLU - Performs linear interpolation for (u,v) coordinates **
C*** 5. CNDNSN - Evaluates Jacobian elliptic functions CN,DN and SN **
C***** **
C*** FUNCTIONS : **
C*** **
C*** 1. CN,DN,SN Jacobian elliptic functions **
C*** 2. FUNCTN - Evaluates integrand of elliptic integral K (or K') **
C*** [1/sqrt(1-k**2sin**2(phi))]/[1/sqrt(1-k'**2sin**2(phi))] **
C*** 3. GAUSS - Performs Gaussian integration of elliptic integrals **
C*** **
C***** **
SUBROUTINE SSN1( V0, CM0, SN1 )
C***** **
C*** **
C*** THIS SUBROUTINE EVALUATES JACOBIAN ELLIPTIC FUNCTIONS **
C*** USING A DESCENDING LANDEN TRANSFORMATION WITH THE REDUCED **
C*** FUNCTIONS APPROXIMATED BY CIRCULAR (TRIG) FUNCTIONS **
C*** (See page 573, "Handbook of Mathematical Functions" by Abramowitz) **
C***** **
C*** **
IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
C***
C***** **
C***-----
C*** Perform three applications of reduction to reduce the parameter :
C***-----
C***
CM1 = 1.0D0 - CM0
RMU1 = ( ( 1.0D0-DSQRT(CM1) ) / ( 1.0D0+DSQRT(CM1) ) )**2
V1 = V0 / ( 1.0D0+DSQRT( RMU1 ) )
C***
CM2 = 1.0D0 - RMU1
RMU2 = ( ( 1.0D0-DSQRT(CM2) ) / ( 1.0D0+DSQRT(CM2) ) )**2
V2 = V1 / ( 1.0D0+DSQRT( RMU2 ) )

```

```

C***
CM3 = 1.0D0 - RMU2
RMU3 = ( ( 1.0D0-DSQRT(CM3) ) / ( 1.0D0+DSQRT(CM3) ) )**2
V3 = V2 / ( 1.0D0+DSQRT( RMU3 ) )
C***
SN3 = (1.0D0+DSQRT(RMU3))*SN(V3,RMU3)
SN3 = SN3/( 1.0D0 + DSQRT(RMU3)*SN(V3,RMU3)**2 )
SN2 = (1.0D0+DSQRT(RMU2))*SN3
SN2 = SN2/( 1.0D0 + DSQRT(RMU2)*SN3**2 )
SN1 = (1.0D0+DSQRT(RMU1))*SN2
SN1 = SN1/( 1.0D0 + DSQRT(RMU1)*SN2**2 )
C***
RETURN
END
C***
C***
C*****
SUBROUTINE XYUV(G1,G2,IFLG,MCOUNT,NL,P,SN,SS1,UPT,UVDATA,
*VPT,XPT,XYDATA,YPT)
C*****
C***
C*** THIS SUBROUTINE PERFORMS COORDINATE TRANSFORMATION : **
C*** CIRCLE (x,y) -> SQUARE (u,v) **
C*** **
C*****
C***
DOUBLE PRECISION G1,G2,P,SN,SS1,UVDATA,XYDATA
DOUBLE PRECISION RK,RMOD2,X,XX,YY
COMMON/MAP/RK,RMOD2,XX,YY
COMMON/PTPAR1/GPVMN,GPVMX,LCODE,NO,NPV
DIMENSION X(2)
VIRTUAL UPT(*),VPT(*),XPT(*),YPT(*)
C***
C*****
C***
IF(IFLG.EQ.1) NP = NPV
IF(IFLG.EQ.2) NP = NL
DO 1 I = 1,NP
XX = DBLE(XPT(I))
YY = DBLE(YPT(I))
CALL NAS2(G1,G2,SS1,UVDATA,X,XYDATA)
CALL INTPLU(MCOUNT,P,UPT(I),SN,X(2))
CALL INTPLU(MCOUNT,P,VPT(I),SN,X(1))
1 CONTINUE
C***
RETURN
END
C***
C***
C*****
SUBROUTINE NAS2(G1,G2,SS1,UVDATA,X,XYDATA)
C*****
C***
C*** THIS SUBROUTINE PREPARES INPUT PARAMETERS FOR SUBROUTINE NASNR **

```

```

C***
***
C*****
C***
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      EXTERNAL      F
      INTEGER      N, MAXN, MAXIT, ISTAT
      PARAMETER     (MAXN=42)
      LOGICAL      AGIVEN
      DIMENSION     X(2), R(2), A(MAXN,MAXN), WK(MAXN,MAXN+3)
      VIRTUAL      SS1(2,*), UVDATA(2,*), XYDATA(2,*)
      COMMON/ MAP / RK, RMOD2, XX, YY
      COMMON/ NSIZE / IMAX, IMID, IMIN, ISIZE, JMAX, JMID, JMIN, JSIZE

C***
C*****
C***-----
C*** Determine the sign of S and S1 :
C***-----
C***
      AGIVEN = .FALSE.
      IF (XX.GE.0.0D0) THEN
      IF (YY.GE.0.0D0) THEN
      X(1) = G1
      X(2) = G2
      ELSE
      X(1) = -G1
      X(2) = G2
      END IF
      ELSE
      IF (YY.GE.0.0D0) THEN
      X(1) = G1
      X(2) = -G2
      ELSE
      X(1) = -G1
      X(2) = -G2
      END IF
      END IF

C***
      TOL = 1.0D-6
      MAXIT = 400
      N = 2

C***
C***-----
C*** Call the subroutine that would solve
C*** F(x)=0 using a quasi Newton method :
C***-----
C***
      CALL NASNWR(F,X,N,R,A,MAXN,TOL,WK,MAXIT,AGIVEN,ISTAT)

C***
C***-----
C*** Limiting cases :
C***-----
C***
      IF (XX.EQ.0.0D0) X(2) = 0.0D0

```

```

      IF (YY.EQ.0.0D0) X(1) = 0.0D0
      DO 22 J = JMIN, JMAX
      DO 22 I = IMIN, IMAX
      K      = I + (J-JMIN)*JMAX
      IF (XX.EQ.XYDATA(1,K) .AND. YY.EQ.XYDATA(2,K)) THEN
      IF (DABS(UVDATA(1,K)) .EQ. DABS(UVDATA(2,K))) THEN
      X(1)    = SS1(1,K)
      X(2)    = SS1(2,K)
      END IF
      END IF
22 CONTINUE
C***
      RETURN
      END
C***
C***
C*****
      SUBROUTINE NASNWR(F,X,N,R,A,MAXN,TOL,WK,MAXIT,AGIVEN,IRSTAT)
C*****
C***
C***      THIS SUBROUTINE SOLVES A SYSTEM OF NONLINEAR ALGEBRAIC
C***      EQUATIONS BASED ON THE NEWTON-RAPHSON METHOD
C***
C*****
C***
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      EXTERNAL      F
      INTEGER      N,      MAXN,      MAXIT,      ISTAT
      LOGICAL      AGIVEN
      DIMENSION    X(1),      R(1),      A(MAXN,1),      WK(MAXN,1)
      INTRINSIC    DABS,      DMIN1,      DMAX1,      DSQRT
      EXTERNAL    NASABX, NASJME, NASFNE
      EXTERNAL    MOVEDD, VIPASD
      INTEGER      MAXJME, MAXJMS
      INTEGER      IT,      IT1,      JMS,      J10
      INTEGER      NPLUS1, NPLUS2, NPLUS3
      INTEGER      J77,      J88,      J99,      IDUMMY
      LOGICAL      EVALJM, XFOUND
      PARAMETER    (MAXJME=10)
      PARAMETER    (MAXJMS=2)
C***
C*****
C***
      NPLUS1 = N+1
      NPLUS2 = NPLUS1+1
      NPLUS3 = NPLUS2+1
      EVALJM = .NOT.AGIVEN
C***
      IT      = 0
      IT1     = 0
      JMS     = 0
      DFRAC   = .5D00
C***
C***-----

```

```

C*** Investigate the initial guess :
C***-----
C***
      XNORM2 = 0.D00
      CALL VIPASD(X,1,X,1,N,XNORM2)
      IF (XNORM2.EQ.0.D00) THEN
        ISTAT = -101
        RETURN
      ENDIF
      CALL NASFNE(F,X,N,R,RNORM)
      CALL MOVEDD(N,R,WK(1,NPLUS1))

C***
C***-----
C*** Reevaluate the function Jacobi matrix if necessary :
C***-----
C***
      1 CONTINUE
      IF (EVALJM) THEN
        CALL NASJME(F,X,N,R,A,MAXN)
      ENDIF

C***
C***-----
C*** Iterate with an estimate of the function Jacobi matrix :
C***-----
C***
      2 CONTINUE
      DO 399 J99=1,MAXJME
        XFOUND = RNORM.LT.TOL
        IF (XFOUND) THEN

C***
C***-----
C*** The residual norm is within given tolerance. All done :
C***-----
C***
          ISTAT = 0
          RETURN
        ELSE

C***
C***-----
C*** Next iteration is necessary :
C***-----
C***
          TNORM = RNORM

C***
C***-----
C*** Compute the nominal iteration step :
C***-----
C***
          DO 188 J88=1,N
            WK(J88,NPLUS3) = -R(J88)
            WK(J88,NPLUS2) = WK(J88,NPLUS1)
            CALL MOVEDD(N,A(1,J88),WK(1,J88))
          188 CONTINUE
          CALL NASABX(WK,MAXN,N,WK(1,NPLUS3),WK(1,NPLUS1),R,IDUMMY)

```



```

IF (IDUMMY.EQ.0) THEN
  JMS = JMS+1
  IF (JMS.GT.MAXJMS) THEN
    CALL NASFNE(F,X,N,R,RNORM)
    CALL NASJME(F,X,N,R,A,MAXN)
    ISTAT = -10
    RETURN
  ELSE
    CALL MOVEDD(N,WK(1,NPLUS2),WK(1,NPLUS1))
  ENDIF
ELSE
  JMS = 0
ENDIF
DNORM2 = 0.D00
CALL VIPASD(WK(1,NPLUS1),1,WK(1,NPLUS1),1,N,DNORM2)
IF (DNORM2.EQ.0D00) THEN
  ISTAT = -102
  RETURN
ENDIF
CORR = DMIN1(DFRAC/DSQRT(DNORM2/XNORM2),1.D00)
X8 = CORR
J10 = 0

```

C***

C***-----

C*** Evaluate the function for new arguments :

C***-----

C***

```

200      CONTINUE
        DO 288 J88=1,N
          WK(J88,NPLUS1) = CORR*WK(J88,NPLUS1)
          WK(J88,1) = X(J88)+WK(J88,NPLUS1)
288      CONTINUE
        CALL NASFNE(F,WK,N,R,RNORM)
        IF(RNORM.GE.TNORM) GOTO 4
        IT1 = IT1+1
        IF(IT1.EQ.1) GOTO 5
        IT1 = 0
        DFRAC = DMIN1(DFRAC+.2D00,2.D00)
        GOTO 5
4        CORR = .2D00
        X8 = X8*.2D00
        IT1 = 0
        DFRAC = DMAX1(DFRAC-.05D00,.1D00)
        J10 = J10+1
        IF(J10.LE.2) GOTO 200
        X8 = X8*5.D00
5        CONTINUE

```

C***

C***-----

C*** Extrapolate the function Jacobi matrix :

C***-----

C***

```

TEMP1 = 1.D00/(DNORM2*X8**2)
DNORM2 = 1.D00-X8

```

```

      DO 388 J88=1,N
        X(J88) = WK(J88,1)
        TEMP2 = (R(J88)+WK(J88,NPLUS3)*DNORM2)*TEMP1
        DO 377 J77=1,N
          A(J88,J77) = A(J88,J77)+TEMP2*WK(J77,NPLUS1)
377      CONTINUE
388      CONTINUE
      ENDIF
399 CONTINUE
      XFOUND = RNORM.LT.TOL
      IF (XFOUND) THEN
C***
C***-----
C*** The residual norm is within given tolerance, fortunately :
C***-----
C***
          IRSTAT = 0
          RETURN
      ELSE
C***
C***-----
C*** Next iteration is necessary :
C***-----
C***
          IT = IT+MAXJME
          IF (IT.GE.MAXIT) THEN
C***
C***-----
C*** Iteration limit reached :
C***-----
C***
          IRSTAT = -1
          RETURN
          ELSEIF (RNORM.LE..5D00*TNORM) THEN
C***
C***-----
C*** The function Jacobi matrix need not be updated :
C***-----
C***
          GO TO 2
      ELSE
C***
C***-----
C*** The function Jacobi matrix must be updated :
C***-----
C***
          EVALJM = .TRUE.
          GO TO 1
      ENDIF
    ENDIF
C***
      RETURN
    END
C***

```

```

C***
C*****
      SUBROUTINE  NASJME (F,X,N,R,A,MAXN)
C*****
C***
C***      THIS SUBROUTINE EVALUATES THE FUNCTION JACOBI MATRIX      **
C***
C*****
C***
      EXTERNAL          F
      INTEGER           N,          MAXN
      DOUBLE PRECISION  X,          R,          A
      DIMENSION         X(1),      R(1),      A(MAXN,1)
      INTRINSIC         DABS
      EXTERNAL          NASEPS
      DOUBLE PRECISION  DABS,      NASEPS
      DOUBLE PRECISION  DIFFR,      XINCR,      XPERT,      XSAVE
      INTEGER           ZEROS
      INTEGER           J88,      J99
      LOGICAL           FIRST,      MODIFY
      SAVE              DIFFR,      FIRST
      DATA             FIRST/.TRUE./
C***
C*****
C***-----
C*** Determine the nominal difference-to-argument ratio :
C***-----
C***
      IF (FIRST) THEN
          DIFFR = DSQRT(NASEPS(1))
          FIRST = .FALSE.
      ENDIF
C***
C***-----
C*** Compute the Jacobi matrix (by columns) :
C***-----
C***
      DO 199 J99=1,N
          MODIFY = .FALSE.
      100 CONTINUE
C***
C***-----
C*** Compute perturbed argument :
C***-----
C***
      XSAVE = X(J99)
      IF (XSAVE.EQ..0D00.OR.MODIFY) THEN
          XINCR = DIFFR
      ELSE
          XINCR = XSAVE*DIFFR
      ENDIF
      XPERT = XSAVE+XINCR
      X(J99) = XPERT
C***

```

```

C***-----
C***  Evaluate the function residuals :
C***-----
C***
      CALL F(X,N,A(1,J99))
C***
C***-----
C***  Restore the argument :
C***-----
C***
      X(J99) = XSAVE
C***
C***-----
C***  Compute an estimated partial derivative :
C***-----
C***
      ZEROS = 0
      DO 188 J88=1,N
        A(J88,J99) = (A(J88,J99)-R(J88))/XINCR
        IF (A(J88,J99).EQ..0D00) THEN
          ZEROS = ZEROS+1
        ENDIF
      188 CONTINUE
C***
C***-----
C***  Test for possibly misleading results. Decide whether to
C***  modify the finite difference and recompute the column :
C***-----
C***
      IF (ZEROS.EQ.N.AND..NOT.MODIFY) THEN
        MODIFY = .TRUE.
        GO TO 100
      ENDIF
      199 CONTINUE
C***
      RETURN
      END
C***
C***
C*****
      SUBROUTINE NASFNE(F,X,N,R,NORM)
C*****
C***
C***  THIS SUBROUTINE EVALUATES THE FUNCTION RESIDUALS AND THEIR NORM
C***
C*****
      EXTERNAL          F
      INTEGER           N
      DOUBLE PRECISION  X,      R,      NORM
      DIMENSION         X(1),   R(1)
      INTRINSIC         DSQRT
      DOUBLE PRECISION  DSQRT
C***

```

```

C*****
C***-----
C***  Evaluate residuals first :
C***-----
C***
        CALL F(X,N,R)
C***
C***-----
C***  Evaluate the norm of R :
C***-----
C***
        CALL NORMSD(R,1,N,NORM)
C***
        RETURN
        END
C***
C***
C*****
        DOUBLE PRECISION FUNCTION NASEPS(IDUMMY)
C*****
C***
C***  THIS FUNCTION COMPUTES THE HOST ARITHMETIC UNIT ACCURACY
C***
C***
C*****
C***
        INTEGER          IDUMMY
        DOUBLE PRECISION EPS,    BUFFER
        INTEGER          NR
        INTEGER          J99
        LOGICAL          FIRST
        PARAMETER        (NR=32)
        DIMENSION        BUFFER(NR)
        SAVE             EPS,    FIRST
        DATA             EPS/1.D00/,FIRST/.TRUE./
C***
C*****
C***
        IF (FIRST) THEN
10      CONTINUE
          EPS = EPS/2.D00
          BUFFER(1) = 1.D00+EPS
          DO 199 J99=1,NR-1
            BUFFER(J99+1) = BUFFER(J99)
199      CONTINUE
          DO 299 J99=1,NR
            IF (BUFFER(J99).EQ.1.D00) THEN
              FIRST = .FALSE.
              EPS = EPS*2.D00
              NASEPS = EPS
              RETURN
            ENDIF
299      CONTINUE
          GO TO 10
        ELSE

```

```

      NASEPS = EPS
    ENDIF
C***
    RETURN
    END
C***
C***
C*****
      SUBROUTINE NASABX (A,NA,N,B,X,MP, IDET)
C*****
C***
C***          THIS SUBROUTINE SOLVES A*X=B BY LU DECOMPOSITION          **
C***                      AND BACKWARDS SUBSTITUTION                      **
C***
C***
C*****
C***
      INTEGER          NA,      N,      IDET,      MP
      DOUBLE PRECISION A,      B,      X
      DIMENSION        A (NA,1), B (1),  X (1),    MP (1)
C***
C*****
C***
      CALL DECOM(A,NA,N,X,MP, IDET)
      IF (IDET.EQ.0) RETURN
      CALL SUBST(A,NA,N,B,X,MP)
      RETURN
      END
C***
C***
C*****
      SUBROUTINE DECOM (A,NA,N,VS,MP, IDET)
C*****
C***
C***          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A (NA,1), VS (1), MP (1)
      DOUBLE PRECISION NASEPS
C***
C*****
C***
      EPS = NASEPS (1) * 2**MIN (4,N/2)
      DO 100 J00=1,N
      CALL NORMSD (A (J00,1),NA,N,TEMP)
      IF (TEMP.EQ..0D0) GO TO 4
      VS (J00) = .1D1/TEMP
      MP (J00) = J00
100 CONTINUE
      IDET = 1
      DO 200 J00=1,N
      IS = J00
      X = .0D0
      I00 = J00-1
      DO 110 J10=J00,N
      TEMP = -A (J10,J00)
      IF (I00.GT.0) CALL VIPASD (A (J10,1),NA,A (1,J00),1,I00,TEMP)

```

```

      A(J10,J00) = -TEMP
      TEMP = DABS(TEMP*VS(J10))
      IF (X.GE.TEMP) GO TO 110
      X = TEMP
      IS = J10
110  CONTINUE
      IF (IS.EQ.J00) GO TO 1
      IDET = -IDET
      CALL EXCHSD(A(IS,1),NA,A(J00,1),NA,N)
      TEMP = VS(J00)
      VS(J00) = VS(IS)
      VS(IS) = TEMP
      JS = MP(J00)
      MP(J00) = MP(IS)
      MP(IS) = JS
1    CONTINUE
      IF (X.LE.EPS) GO TO 4
      X = -.1D1/A(J00,J00)
      K00 = J00+1
      IF (K00.GT.N) GO TO 200
      DO 120 J20=K00,N
      TEMP = -A(J00,J20)
      IF (I00.GT.0) CALL VIPASD(A(J00,1),NA,A(1,J20),1,I00,TEMP)
      A(J00,J20) = X*TEMP
120  CONTINUE
200  CONTINUE
      RETURN
      4  IDET = 0
      RETURN
      END

C***
C***
C*****
      SUBROUTINE SUBST(A,NA,N,B,VS,MP)
C*****
C***
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION A(NA,1),B(1),VS(1),MP(1)

C***
C*****
C***
      DO 100 J00=1,N
      VS(J00) = B(MP(J00))
100  CONTINUE
      VS(1) = -VS(1)/A(1,1)
      IF (N.EQ.1) GO TO 1
      DO 200 J00=2,N
      TEMP = VS(J00)
      CALL VIPASD(A(J00,1),NA,VS(1),1,J00-1,TEMP)
      VS(J00) = -TEMP/A(J00,J00)
200  CONTINUE
      1  M00 = N
      DO 300 J00=1,N
      I00 = J00-1

```

```

      TEMP = VS(M00)
      IF (I00.GT.0) CALL VIPASD(A(M00,M00+1),NA,VS(M00+1),1,I00,TEMP)
      VS(M00) = -TEMP
      M00 = M00-1
300 CONTINUE
      RETURN
      END

C***
C***
C*****
      SUBROUTINE MOVEDD(N,X,Y)
C*****
C***
C***      THIS SUBROUTINE MOVES ELEMENTS OF A VECTOR TO ANOTHER VECTOR **
C***
C*****
C***
      INTEGER          N
      DOUBLE PRECISION X,      Y
      DIMENSION        X(1),   Y(1)
      INTEGER          J99

C***
C*****
C***
      DO 199 J99=1,N
          Y(J99) = X(J99)
199 CONTINUE
C***
      RETURN
      END

C***
C***
C*****
      SUBROUTINE NORMSD(X, INCX, N, NORM)
C*****
C***
C***      THIS SUBROUTINE COMPUTES THE EUCLIDEAN NORM OF A VECTOR **
C***
C*****
C***
      INTEGER          INCX,    N
      DOUBLE PRECISION X,      NORM
      DIMENSION        X(1)
      INTEGER          J99

C***
C*****
C***
      NORM = .0D00
      DO 199 J99=1, INCX*N, INCX
          NORM = NORM+X(J99)**2
199 CONTINUE
      NORM = DSQRT(NORM)
C***
      RETURN

```



```

      END
C***
C***
C*****
      SUBROUTINE EXCHSD(X, INCX, Y, INCY, N)
C*****
C***
C***          THIS SUBROUTINE SWAPS ELEMENTS OF TWO VECTORS          **
C***
C*****
C***
      INTEGER          INCX,      INCY,      N
      DOUBLE PRECISION X,          Y
      DIMENSION        X(1),      Y(1)
      INTEGER          I99,      J99
      DOUBLE PRECISION TEMP
C***
C***
C*****
C***
      I99 = 1
      DO 199 J99=1, INCX*N, INCX
          TEMP = X(J99)
          X(J99) = Y(I99)
          Y(I99) = TEMP
          I99 = I99+INCY
      199 CONTINUE
C***
      RETURN
      END
C***
C***
C*****
      SUBROUTINE VIPASD(X, INCX, Y, INCY, N, VIP)
C*****
C***
C***          THIS SUBROUTINE COMPUTES A VECTOR INNER PRODUCT          **
C***          AND ADDS IT TO A GIVEN VALUE                              **
C***
C*****
C***
      INTEGER          INCX,      INCY,      N
      DOUBLE PRECISION X,          Y,          VIP
      DIMENSION        X(1),      Y(1)
      INTEGER          I99,      J99
C***
C*****
C***
      I99 = 1
      DO 199 J99=1, INCX*N, INCX
          VIP= VIP+X(J99)*Y(I99)
          I99 = I99+INCY
      199 CONTINUE
C***
      RETURN

```

```

      END

C***
C***
C*****
      SUBROUTINE F(X,N,R)
C*****
C***
C***      THIS SUBROUTINE PROVIDES THE SYSTEM OF NONLINEAR      **
C***      ALGEBRAIC EQUATIONS FOR USE IN SUBROUTINE NASNWR      **
C***
C*****
C***
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER          N
      DIMENSION        X(1), R(1)
      COMMON/MAP/RK,RMOD2,XX,YY

C***
C*****
C***-----
C***  Initialize parameters :
C***-----
C***
C***-----
C***  K parameter :
C***-----
C***
      RK2  = RMOD2
      RK4  = RK2*RK2
      RK6  = RK4*RK2
      RK8  = RK6*RK2

C***
C***-----
C***  K prime parameter :
C***-----
C***
      RKP2 = RMOD2
      RKP4 = RKP2*RKP2

C***
C***-----
C***  S parameter :
C***-----
C***
      S2   = X(2)*X(2)
      S4   = S2*S2
      S6   = S4*S2
      S8   = S6*S2
      S10  = S8*S2
      S12  = S10*S2

C***
C***-----
C***  S1 parameter :
C***-----
C***
      S02  = X(1)*X(1)

```

```

S14 = S02*S02
S16 = S14*S02
S18 = S16*S02
S110 = S18*S02
S112 = S110*S02

C***
C***-----
C*** X,Y parameters :
C***-----
C***
      X2 = XX*XX
      Y2 = YY*YY
C***
C***-----
C*** Compute the S1 polynomials :
C***-----
C***
C***-----
C*** Numerator :
C***-----
C***
      PNX2 = 1.-2.*(3.-2.*RK2)*S02+(15.-18.*RK2+4.*RK4)*S14-
&          4.*(5.-3.*RK2)*RKP2*S16+(15.-13.*RK2)*RKP2*S18-
&          6.*RKP4*S110+RKP4*S112
      PNX4 = -(1.+RK2)-2.*(3.+2.*RK2*RKP2)*S02+(15.+7.*RK2-22.
&          *RK4+4.*RK6)*S14-4.*(5.+7.*RK2-5.*RK4)*RKP2*S16+
&          (15.+22.*RK2-29.*RK4)*RKP2*S18-2.*(3.+8.*RK2)*
&          RKP4*S110+(1.+3.*RK2)*RKP4*S112)
      PNX6 = (1.-2.*(4.-RK2)*S02+(25.-15.*RK2-4.*RK4)*S14-4.*
&          (10.+RK2-2.*RK4)*RKP2*S16+(35.-3.*RK2-20.*RK4)*
&          RKP2*S18-2.*(8.+7.*RK2)*RKP4*S110+3.*(1.+RK2)*
&          RKP4*S112)*RK2
      PNX8 = -(-2.*S02+(11.-7.*RK2)*S14-4.*(6.-RK2)*RKP2*S16+
&          2.*(13.-7.*RK2-2.*RK4)*RKP2*S18-2.*(7.+2.*RK2)*
&          RKP4*S110+(3.+RK2)*RKP4*S112)*RK4
      PNX10= (S14-4.*RKP2*S16+2.*(3.-2.*RK2)*RKP2*S18-4.*RKP4
&          *S110+RKP4*S112)*RK6
C***
      PNY0 = S02-(4.-RK2)*S14+3.*(2.-RK2)*S16-(4.-3.*RK2)*S18+RKP2*S110
      PNY2 = -2.*(2.*S02-(9.-2.*RK2)*S14+(15.-7.*RK2)*S16-
&          (11.-8.*RK2)*S18+3.*RKP2*S110)*RK2
      PNY4 = (2.*(1.+2.*RK2)*S02-2.*(4.+11.*RK2-2.*RK4)*S14+
&          (12.+43.*RK2-20.*RK4)*S16-(8.+36.*RK2-29.*RK4)*S18
&          +(2.+13.*RK2)*RKP2*S110)*RK2
      PNY6 = -4.*(S02-(5.+RK2)*S14+(3.-RK2)*(3.+2.*RK2)*S16-(7.+
&          3.*RK2-5.*RK4)*S18+(2.+3.*RK2)*RKP2*S110)*RK4
      PNY8 = (S02-(4.+7.*RK2)*S14+(6.+23.*RK2-4.*RK4)*S16-(4.+25.*RK2
&          -10.*RK4-4.*RK6)*S18+(1.+10.*RK2+4.*RK4)*RKP2*S110)*RK4
      PNY10= -2.*(-S14+(3.+RK2)*S16-(3.+2.*RK2*RKP2)*S18+
&          (1.+2.*RK2)*RKP2*S110)*RK6
      PNY12= (S16-(2.-RK2)*S18+RKP2*S110)*RK8
C***
C***-----
C*** Denominator :

```

```

C***-----
C***
    PD0  = 1.-4.*S02+6.*S14-4.*S16+S18
    PD2  = -2.*(1.-(5.+RK2)*S02+2.*(5.+RK2)*S14-10.*S16+
&        (5.-2.*RK2)*S18-RKP2*S110)
    PD4  = 1.-6.*(1.+RK2)*S02+(15.+22.*RK2-RK4)*S14-
&        4.*(5.+7.*RK2-2.*RK4)*S16+3.*(5.+4.*RK2*RKP2)*
&        S18-2.*(3.+2.*RK2)*RKP2*S110+RKP4*S112
    PD6  = -4.*(-S02+(5.+RK2)*S14-(10.+RK2-RK4)*S16+(10.-3.*
&        RK2-2.*RK4)*S18-5.*RKP2*S110+RKP4*S112)*RK2
    PD8  = (6.*S14-4.*(6.-RK2)*S16+(36.-20.*RK2-RK4)*S18-
&        4.*(6.-RK2)*RKP2*S110+6.*RKP4*S112)*RK4
    PD10 = -2.*(-2.*S16+3.*(2.-RK2)*S18-(6.-7.*RK2+RK4)*S110
&        +2.*RKP4*S112)*RK6
    PD12 = (S18-2.*RKP2*S110+RKP4*S112)*RK8
C***
C***-----
C*** Compute the coefficients of the S polynomial :
C***-----
C***
    PX0  = -PD0*X2
    PX2  =  PN2-PD2*X2
    PX4  =  PN4-PD4*X2
    PX6  =  PN6-PD6*X2
    PX8  =  PN8-PD8*X2
    PX10 =  PN10-PD10*X2
    PX12 = -PD12*X2
C***
    PY0  =  PNY0-PD0*Y2
    PY2  =  PNY2-PD2*Y2
    PY4  =  PNY4-PD4*Y2
    PY6  =  PNY6-PD6*Y2
    PY8  =  PNY8-PD8*Y2
    PY10 =  PNY10-PD10*Y2
    PY12 =  PNY12-PD12*Y2
C***
C***-----
C*** Compute the S polynomials :
C***-----
C***
    R(1) = PY0+PY2*S2+PY4*S4+PY6*S6+PY8*S8+PY10*S10+PY12*S12
    R(2) = PX0+PX2*S2+PX4*S4+PX6*S6+PX8*S8+PX10*S10+PX12*S12
C***
    RETURN
    END
C***
C***
C*****
    SUBROUTINE INTPLU(MCOUNT,P,P0,SN,S0)
C*****
C***
C***      THIS SUBROUTINE PERFORMS LINEAR INTERPOLATION FOR U OR V **
C***
C*****

```

```

C***
      DOUBLE PRECISION P,PD,S0,SN,SNQ
      VIRTUAL P(*),SN(*)
C***
C*****
C***-----
C***  Check the bounds on S0 : S/S1(1) < S0 < S/S1(NPOINT) :
C***-----
C***
      IF(S0.LT.SN(1))      S0 = SN(1)
      IF(S0.GT.SN(MCOUNT)) S0 = SN(MCOUNT)
C***
C***-----
C***  Determine the relative location of S0
C***  and interpolate for the value of U or V
C***-----
C***
      DO 1 I = 1,MCOUNT
      IF(S0.GT.SN(I)) GO TO 1
      SNQ      = (S0-SN(I))/(SN(I-1)-SN(I))
      PD       = P(I-1)-P(I)
      P0       = SNGL(SNQ*PD + P(I))
      GO TO 2
1 CONTINUE
2 RETURN
      END
C***
C***
C*****
      SUBROUTINE CNDNSN( V0, CM0, CN1, DN1, SN1)
C*****
C***
C***      THIS SUBROUTINE EVALUATES JACOBIAN ELLIPTIC FUNCTIONS      **
C***      USING A DESCENDING LANDEN TRANSFORMATION WITH THE REDUCED  **
C***      FUNCTIONS APPROXIMATED BY CIRCULAR (TRIG) FUNCTIONS      **
C***      (See page 573, "Handbook of Mathematical Functions" by Abramowitz)**
C***
C*****
C***
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
C***
C*****
C***-----
C***  Perform three applications of reduction to reduce the parameter
C***-----
C***
      CM1 = 1.0D0 - CM0
      RMU1 = ( ( 1.0D0-DSQRT(CM1) ) / ( 1.0D0+DSQRT(CM1) ) )**2
      V1 = V0 / ( 1.0D0+DSQRT( RMU1 ) )
C***
      CM2 = 1.0D0 - RMU1
      RMU2 = ( ( 1.0D0-DSQRT(CM2) ) / ( 1.0D0+DSQRT(CM2) ) )**2
      V2 = V1 / ( 1.0D0+DSQRT( RMU2 ) )
C***

```

```

      CM3 = 1.0D0 - RMU2
      RMU3 = ( ( 1.0D0-DSQRT(CM3) ) / ( 1.0D0+DSQRT(CM3) ) )**2
      V3 = V2 / ( 1.0D0+DSQRT( RMU3 ) )
C***
      DN3 = DN(V3,RMU3)**2 + DSQRT(RMU3) - 1.0D0
      DN3 = DN3/( 1.0D0 + DSQRT(RMU3) - DN(V3,RMU3)**2 )
      SN3 = (1.0D0+DSQRT(RMU3))*SN(V3,RMU3)
      SN3 = SN3/( 1.0D0 + DSQRT(RMU3)*SN(V3,RMU3)**2 )
      CN3 = CN(V3,RMU3)*DN(V3,RMU3)
      CN3 = CN3/( 1.0D0 + DSQRT(RMU3)*SN(V3,RMU3)**2 )
C***
      DN2 = DN3**2 + DSQRT(RMU2) - 1.0D0
      DN2 = DN2/( 1.0D0 + DSQRT(RMU2) - DN3**2 )
      SN2 = (1.0D0+DSQRT(RMU2))*SN3
      SN2 = SN2/( 1.0D0 + DSQRT(RMU2)*SN3**2 )
      CN2 = CN3*DN3
      CN2 = CN2/( 1.0D0 + DSQRT(RMU2)*SN3**2 )
C***
      DN1 = DN2**2 + DSQRT(RMU1) - 1.0D0
      DN1 = DN1/( 1.0D0 + DSQRT(RMU1) - DN2**2 )
      SN1 = (1.0D0+DSQRT(RMU1))*SN2
      SN1 = SN1/( 1.0D0 + DSQRT(RMU1)*SN2**2 )
      CN1 = CN2*DN2
      CN1 = CN1/( 1.0D0 + DSQRT(RMU1)*SN2**2 )
C***
      RETURN
      END
C***
C***
C*****
      FUNCTION CN( X, RM )
C*****
C***
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
      CN = DCOS(X) + 0.25D0*RM*(X-DSIN(X)*DCOS(X))*DSIN(X)
      RETURN
      END
C***
C***
C*****
      FUNCTION DN( X, RM )
C*****
C***
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
      DN = 1.0D0 - 0.5D0*RM*DSIN(X)**2
      RETURN
      END
C***
C***
C*****
      FUNCTION SN( X, RM )
C*****
C***
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )

```

```

      SN = DSIN(X) - 0.25D0*RM*(X-DSIN(X)*DCOS(X))*DCOS(X)
      RETURN
      END

C***
C***
C*****
      FUNCTION FUNCTN( DX, CM0 )
C*****
C***
C***      THIS FUNCTION EVALUATES THE VALUE OF THE INTEGRAND      **
C***      [1/sqrt(1-k**2sin**2(phi))] OR [1/sqrt(1-k'**2sin**2(phi))] **
C***      IN THE COMPLETE ELLIPTIC INTEGRAL (K) OR ITS COMPLEMENTARY (K') **
C***      RESPECTIVELY **
C*** **
C*****
C***
      IMPLICIT DOUBLE PRECISION (A-H, O-Z )
C***
C*****
C***
      FUNCTN = ( 1.0D0 - CM0*DSIN(DX)**2 )
      FUNCTN = DSQRT( FUNCTN )
      FUNCTN = 1.0D0/FUNCTN
      RETURN
      END

C***
C***
C*****
      FUNCTION GAUSS( A, B, M, FUNCTN, CM0 )
C*****
C***
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
      DIMENSION NPOINT(7), KEY(8), Z(24), WEIGHT(24)
      DATA NPOINT / 2, 3, 4, 5, 6, 10, 15 /
      DATA KEY / 1, 2, 4, 6, 9, 12, 17, 25 /
      DATA Z / 0.577350269, 0.0, 0.774596669, 0.339981044, 0.861136312,
&      0.0, 0.538469310, 0.906179846, 0.238619186, 0.661209387,
&      0.932469514, 0.148874339, 0.433395394, 0.679409568,
&      0.865063367, 0.973906529, 0.0, 0.201194094, 0.394151347,
&      0.570972173, 0.724417731, 0.848206583, 0.937273392,
&      0.987992518 /
      DATA WEIGHT / 1.0, 0.888888889, 0.555555556, 0.652145155,
&      0.347854845, 0.568888889, 0.478628671, 0.236926885,
&      0.467913935, 0.360761573, 0.171324493, 0.295524225,
&      0.269266719, 0.219086363, 0.149451349, 0.066671344,
&      0.202578242, 0.198431485, 0.186161000, 0.166269206,
&      0.139570678, 0.107159221, 0.070366047, 0.030753242 /

C***
C*****
C***-----
C*** Find the subscripts of the first Z and weight values :
C***-----
C***
      DO 1 I=1,7

```

```

      IF ( M.EQ.NPOINT(I) )      GO TO 2
1  CONTINUE
C***
C***  INVALID M USED
C***
      GAUSS = 0.0D0
      RETURN
C***
C***-----
C***  Setup initial parameters :
C***-----
C***
      2 JFIRST = KEY(I)
      JLAST = KEY(I+1) - 1
      C = 0.5D0*(B-A)
      D = 0.5D0*(B+A)
C***
C***-----
C***  Accumulate the sum in the M-POINT formula :
C***-----
C***
      SUM = 0.0D0
      DO 3 J=JFIRST, JLAST
      IF ( Z(J).EQ.0.0D0 )      SUM = SUM + WEIGHT(J)*FUNCTN(D,CM0)
      IF ( Z(J).NE.0.0D0 )      SUM = SUM + WEIGHT(J)*(FUNCTN(Z(J)*C+D,CM0)
      *                          + FUNCTN(-Z(J)*C+D,CM0))
      3 CONTINUE
C***
C***-----
C***  Make the interval correction and the return :
C***-----
C***
      GAUSS = SUM*C
      RETURN
      END
C***

```



```

C***
C*****
C*****      POISSON LIBRARY CONTAINS THE FOLLOWING SUBROUTINES :      **
C*****                                                    **
C*** 1. COSGEN      - Computes required cosine values in ascending order **
C*** 2. GENBUN      - Reorders unknowns, reverses columns and returns    **
C*** 3. HWSCRT      - Does the Fast Fourier Transform [In Poisson/Lib]    **
C*** 4. MERGE       - Merges two ascending strings of nos. in array TCOS **
C*** 5. POISD2      - Solves Pooisson's eqn. for Dirichlet bndry conds.   **
C*** 6. POISN2      - Solves Pooisson's eqn. for Neumann bndry conds.    **
C*** 7. POISP2      - Solves Pooisson's eqn. for Periodic bndry conds.   **
C*** 8. TRIX        - Solves a system of linear equations where the      **
C***                  coefficient matrix is a rational fctn. given by    **
C***                  Tridiagonal (... , A(I), B(I), C(I), ...)          **
C*** 9. TRI3        - Solves three linear systems whose common coeffs.   **
C***                  matrix is a rational fctn.in the matrix given by  **
C***                  Tridiagonal (... , A(I), B(I), C(I), ...)          **
C***                                                    **
C*****
C*****      SUBROUTINE COSGEN (N,IJUMP,FNUM,FDEN,A)
C*****
C***
C***      THIS SUBROUTINE COMPUTES REQUIRED COSINE VALUES IN ASCENDING    **
C***      ORDER.  WHEN IJUMP .GT. 1 THE ROUTINE COMPUTES VALUES          **
C***
C***      2*COS(J*PI/L) , J=1,2,...,L AND J .NE. 0 (MOD N/IJUMP+1)      **
C***      WHERE L = IJUMP*(N/IJUMP+1).                                    **
C***      WHEN IJUMP = 1 IT COMPUTES                                     **
C***      2*COS((J-FNUM)*PI/(N+FDEN)) , J=1, 2, ... ,N                 **
C***      WHERE
C***      FNUM = 0.5, FDEN = 0.0, FOR REGULAR REDUCTION VALUES          **
C***      FNUM = 0.0, FDEN = 1.0, FOR B-R AND C-R WHEN ISTAG = 1        **
C***      FNUM = 0.0, FDEN = 0.5, FOR B-R AND C-R WHEN ISTAG = 2        **
C***      FNUM = 0.5, FDEN = 0.5, FOR B-R AND C-R WHEN ISTAG = 2        **
C***      IN POISN2 ONLY.
C***
C*****
C***
C***      DIMENSION      A(1)
C***
C*****
C***
C***      PI = 3.14159265358979
C***      IF (N .EQ. 0) GO TO 105
C***      IF (IJUMP .EQ. 1) GO TO 103
C***      K3 = N/IJUMP+1
C***      K4 = K3-1
C***      PIBYN = PI/FLOAT(N+IJUMP)
C***      DO 102 K=1,IJUMP
C***          K1 = (K-1)*K3
C***          K5 = (K-1)*K4
C***          DO 101 I=1,K4
C***              X = K1+I
C***              K2 = K5+I

```

```

          A(K2) = -2.*COS(X*PIBYN)
101    CONTINUE
102    CONTINUE
      GO TO 105
103    CONTINUE
      NP1 = N+1
      Y = PI/(FLOAT(N)+FDEN)
      DO 104 I=1,N
        X = FLOAT(NP1-I)-FNUM
        A(I) = 2.*COS(X*Y)
104    CONTINUE
105    CONTINUE
      RETURN
      END
C***
C***
C*****
      SUBROUTINE GENBUN (NPEROD,N,MPEROD,M,A,B,C,IDIMY,Y,IERROR,W)
C*****
C***
      DIMENSION      Y(IDIMY,1)
      DIMENSION      W(1)      ,B(1)      ,A(1)      ,C(1)
C***
C*****
C***
      IERROR = 0
      IF (M .LE. 2) IERROR = 1
      IF (N .LE. 2) IERROR = 2
      IF (IDIMY .LT. M) IERROR = 3
      IF (NPEROD.LT.0 .OR. NPEROD.GT.4) IERROR = 4
      IF (MPEROD.LT.0 .OR. MPEROD.GT.1) IERROR = 5
      IF (MPEROD .EQ. 1) GO TO 102
      DO 101 I=2,M
        IF (A(I) .NE. C(1)) GO TO 103
        IF (C(I) .NE. C(1)) GO TO 103
        IF (B(I) .NE. B(1)) GO TO 103
101    CONTINUE
      GO TO 104
102    IF (A(1).NE.0. .OR. C(M).NE.0.) IERROR = 7
      GO TO 104
103    IERROR = 6
104    IF (IERROR .NE. 0) RETURN
      MP1 = M+1
      IWBA = MP1
      IWBB = IWBA+M
      IWBC = IWBB+M
      IWB2 = IWBC+M
      IWB3 = IWB2+M
      IWW1 = IWB3+M
      IWW2 = IWW1+M
      IWW3 = IWW2+M
      IWD = IWW3+M
      IWTCOS = IWD+M
      IWP = IWTCOS+4*N

```

```

DO 106 I=1,M
  K = IWBA+I-1
  W(K) = -A(I)
  K = IWBC+I-1
  W(K) = -C(I)
  K = IWBB+I-1
  W(K) = 2.-B(I)
  DO 105 J=1,N
    Y(I,J) = -Y(I,J)
105  CONTINUE
106  CONTINUE
  MP = MPEROD+1
  NP = NPEROD+1
  GO TO (114,107),MP
107  GO TO (108,109,110,111,123),NP
108  CALL POISP2 (M,N,W(IWBA),W(IWBB),W(IWBC),Y,IDIMY,W,W(IWB2),
1      W(IWB3),W(IWW1),W(IWW2),W(IWW3),W(IWD),W(IWTCOS),
2      W(IWP))
  GO TO 112
109  CALL POISD2 (M,N,1,W(IWBA),W(IWBB),W(IWBC),Y,IDIMY,W,W(IWW1),
1      W(IWD),W(IWTCOS),W(IWP))
  GO TO 112
110  CALL POISN2 (M,N,1,2,W(IWBA),W(IWBB),W(IWBC),Y,IDIMY,W,W(IWB2),
1      W(IWB3),W(IWW1),W(IWW2),W(IWW3),W(IWD),W(IWTCOS),
2      W(IWP))
  GO TO 112
111  CALL POISN2 (M,N,1,1,W(IWBA),W(IWBB),W(IWBC),Y,IDIMY,W,W(IWB2),
1      W(IWB3),W(IWW1),W(IWW2),W(IWW3),W(IWD),W(IWTCOS),
2      W(IWP))
112  IPSTOR = W(IWW1)
  IREV = 2
  IF (NPEROD .EQ. 4) GO TO 124
113  GO TO (127,133),MP
114  CONTINUE
C***
C***-----
C*** Reorder unknowns when MP=0
C***-----
C***
  MH = (M+1)/2
  MHM1 = MH-1
  MODD = 1
  IF (MH*2 .EQ. M) MODD = 2
  DO 119 J=1,N
    DO 115 I=1,MHM1
      MHPI = MH+I
      MHMI = MH-I
      W(I) = Y(MHMI,J)-Y(MHPI,J)
      W(MHPI) = Y(MHMI,J)+Y(MHPI,J)
115  CONTINUE
      W(MH) = 2.*Y(MH,J)
      GO TO (117,116),MODD
116  W(M) = 2.*Y(M,J)
117  CONTINUE

```

```

        DO 118 I=1,M
            Y(I,J) = W(I)
118     CONTINUE
119 CONTINUE
        K = IWBC+MHM1-1
        I = IWBA+MHM1
        W(K) = 0.
        W(I) = 0.
        W(K+1) = 2.*W(K+1)
        GO TO (120,121),MODD
120 CONTINUE
        K = IWBB+MHM1-1
        W(K) = W(K)-W(I-1)
        W(IWBC-1) = W(IWBC-1)+W(IWBB-1)
        GO TO 122
121 W(IWBB-1) = W(K+1)
122 CONTINUE
        GO TO 107

C***
C***-----
C*** Reverse columns when NPEROD=4
C***-----
C***
123 IREV = 1
        NBY2 = N/2
124 DO 126 J=1,NBY2
            MSKIP = N+1-J
            DO 125 I=1,M
                A1 = Y(I,J)
                Y(I,J) = Y(I,MSKIP)
                Y(I,MSKIP) = A1
125     CONTINUE
126 CONTINUE
        GO TO (110,113),IREV
127 CONTINUE
        DO 132 J=1,N
            DO 128 I=1,MHMI
                MHMI = MH-I
                MHPI = MH+I
                W(MHMI) = .5*(Y(MHPI,J)+Y(I,J))
                W(MHPI) = .5*(Y(MHPI,J)-Y(I,J))
128     CONTINUE
                W(MH) = .5*Y(MH,J)
                GO TO (130,129),MODD
129     W(M) = .5*Y(M,J)
130     CONTINUE
            DO 131 I=1,M
                Y(I,J) = W(I)
131     CONTINUE
132 CONTINUE
133 CONTINUE

C***
C***-----
C*** Return storage requirements for W array

```

```

C***-----
C***
      W(1) = IPSTOR+IWP-1
      RETURN
      END
C***
C***
C*****
      SUBROUTINE HWSCRT(BDA,BDB,BDC,BDD,ELMBDA,F,IDIMF,PERTRB,IERROR,W)
C*****
C***
C*** W:A ONE-DIMENSIONAL ARRAY THAT MUST BE PROVIDED BY THE USER FOR **
C*** WORK SPACE. W MAY REQUIRE UP TO 4*(N+1)+(13+INT(LOG2(N+1)))*(M+1)**
C*** LOCATIONS. THE ACTUAL NUMBER OF LOCATIONS USED IS COMPUTED BY **
C*** HWSCRT AND IS RETURNED IN LOCATION W(1). **
C*** **
C*****
C***
      COMMON/BNDARY/MBDCND,NBDCND
      COMMON/GRID1 /DU,DUINVT,DV,DVINVT,HDUINV,HDVINV
      COMMON/GRID2 /ULNGTH,A,B,VLNGTH,C,D
      COMMON/NSIZE /MP1,IMID,IMIN,M,NP1,JMID,JMIN,N
      DIMENSION F(IDIMF,1), W(1)
      VIRTUAL BDA(*), BDB(*), BDC(*), BDD(*)
C***
C*****
C***-----
C*** Check for invalid parameters :
C***-----
C***
      IERROR = 0
      IF (A .GE. B) IERROR = 1
      IF (MBDCND.LT.0 .OR. MBDCND.GT.4) IERROR = 2
      IF (C .GE. D) IERROR = 3
      IF (N .LE. 3) IERROR = 4
      IF (NBDCND.LT.0 .OR. NBDCND.GT.4) IERROR = 5
      IF (IDIMF .LT. M+1) IERROR = 7
      IF (M .LE. 3) IERROR = 8
      IF (IERROR .NE. 0) RETURN
      NPEROD = NBDCND
      MPEROD = 0
      IF (MBDCND .GT. 0) MPEROD = 1
      TWDELX = 2./DU
      DELXSQ = 1./DU**2
      TWDELY = 2./DV
      DELYSQ = 1./DV**2
      NP = NBDCND+1
      MP = MBDCND+1
      NSTART = 1
      NSTOP = N
      NSKIP = 1
      GO TO (104,101,102,103,104),NP
101 NSTART = 2
      GO TO 104

```

```

102 NSTART = 2
103 NSTOP = NP1
    NSKIP = 2
104 NUNK = NSTOP-NSTART+1
C***
C***-----
C***  Enter boundary data for X-boundaries :
C***-----
C***
    MSTART = 1
    MSTOP = M
    MSKIP = 1
    GO TO (117,105,106,109,110),MP
105 MSTART = 2
    GO TO 107
106 MSTART = 2
    MSTOP = MP1
    MSKIP = 2
107 DO 108 J=NSTART,NSTOP
    F(2,J) = F(2,J)-F(1,J)*DELXSQ
108 CONTINUE
    GO TO 112
109 MSTOP = MP1
    MSKIP = 2
110 DO 111 J=NSTART,NSTOP
    F(1,J) = F(1,J)+BDA(J)*TWDELX
111 CONTINUE
112 GO TO (113,115),MSKIP
113 DO 114 J=NSTART,NSTOP
    F(M,J) = F(M,J)-F(MP1,J)*DELXSQ
114 CONTINUE
    GO TO 117
115 DO 116 J=NSTART,NSTOP
    F(MP1,J) = F(MP1,J)-BDB(J)*TWDELX
116 CONTINUE
117 MUNK = MSTOP-MSTART+1
C***
C***-----
C***  Enter boundary data for Y-boundaries :
C***-----
C***
    GO TO (127,118,118,120,120),NP
118 DO 119 I=MSTART,MSTOP
    F(I,2) = F(I,2)-F(I,1)*DELYSQ
119 CONTINUE
    GO TO 122
120 DO 121 I=MSTART,MSTOP
    F(I,1) = F(I,1)+BDC(I)*TWDELY
121 CONTINUE
122 GO TO (123,125),NSKIP
123 DO 124 I=MSTART,MSTOP
    F(I,N) = F(I,N)-F(I,NP1)*DELYSQ
124 CONTINUE
    GO TO 127

```

```

125 DO 126 I=MSTART,MSTOP
      F(I,NP1) = F(I,NP1)-BDD(I)*TWDELY
126 CONTINUE
C***
C***-----
C***  Multiply right side by DV**2 :
C***-----
C***
127 DELYSQ = DV*DV
      DO 129 I=MSTART,MSTOP
        DO 128 J=NSTART,NSTOP
          F(I,J) = F(I,J)*DELYSQ
128      CONTINUE
129 CONTINUE
C***
C***-----
C***  Define the A,B,C coefficients in W-array :
C***-----
C***
      ID2 = MUNK
      ID3 = ID2+MUNK
      ID4 = ID3+MUNK
      S = DELYSQ*DELXSQ
      ST2 = 2.*S
      DO 130 I=1,MUNK
        W(I) = S
        J = ID2+I
        W(J) = -ST2+ELMBDA*DELYSQ
        J = ID3+I
        W(J) = S
130 CONTINUE
      IF (MP .EQ. 1) GO TO 131
      W(1) = 0.
      W(ID4) = 0.
131 CONTINUE
      GO TO (135,135,132,133,134),MP
132 W(ID2) = ST2
      GO TO 135
133 W(ID2) = ST2
134 W(ID3+1) = ST2
135 CONTINUE
      PERTRB = 0.
      IF (ELMBDA) 144,137,136
136 IERROR = 6
      GO TO 144
137 IF ((NBDCND.EQ.0 .OR. NBDCND.EQ.3) .AND.
1    (MBDCND.EQ.0 .OR. MBDCND.EQ.3)) GO TO 138
      GO TO 144
C***
C***-----
C***  For singular problems must adjust data
C***  to insure that a solution will exist :
C***-----
C***

```

```

138 A1 = 1.
    A2 = 1.
    IF (NBDCND .EQ. 3) A2 = 2.
    IF (MBDCND .EQ. 3) A1 = 2.
    S1 = 0.
    MSP1 = MSTART+1
    MSTM1 = MSTOP-1
    NSP1 = NSTART+1
    NSTM1 = NSTOP-1
    DO 140 J=NSP1,NSTM1
        S = 0.
        DO 139 I=MSP1,MSTM1
            S = S+F(I,J)
139     CONTINUE
        S1 = S1+S*A1+F(MSTART,J)+F(MSTOP,J)
140 CONTINUE
    S1 = A2*S1
    S = 0.
    DO 141 I=MSP1,MSTM1
        S = S+F(I,NSTART)+F(I,NSTOP)
141 CONTINUE
    S1 = S1+S*A1+F(MSTART,NSTART)+F(MSTART,NSTOP)+F(MSTOP,NSTART)+
1      F(MSTOP,NSTOP)
    S = (2.+FLOAT(NUNK-2)*A2)*(2.+FLOAT(MUNK-2)*A1)
    PERTRB = S1/S
    DO 143 J=NSTART,NSTOP
        DO 142 I=MSTART,MSTOP
            F(I,J) = F(I,J)-PERTRB
142     CONTINUE
143 CONTINUE
    PERTRB = PERTRB/DELYSQ
C***
C***-----
C*** Solve the equation :
C***-----
C***
144 CALL GENBUN (NPEROD,NUNK,MPEROD,MUNK,W(1),W(ID2+1),W(ID3+1),
1              IDIMF,F(MSTART,NSTART),IERR1,W(ID4+1))
    W(1) = W(ID4+1)+3.*FLOAT(MUNK)
C***
C***-----
C*** Fill in identical values when have periodic boundary conditions :
C***-----
C***
    IF (NBDCND .NE. 0) GO TO 146
    DO 145 I=MSTART,MSTOP
        F(I,NP1) = F(I,1)
145 CONTINUE
146 IF (MBDCND .NE. 0) GO TO 148
    DO 147 J=NSTART,NSTOP
        F(MP1,J) = F(1,J)
147 CONTINUE
    IF (NBDCND .EQ. 0) F(MP1,NP1) = F(1,NP1)
148 CONTINUE

```



```

      RETURN
      END

C***
C***
C*****
      SUBROUTINE MERGE (TCOS,I1,M1,I2,M2,I3)
C*****
C***
C*** THIS SUBROUTINE MERGES TWO ASCENDING STRINGS OF NUMBERS IN THE **
C*** ARRAY TCOS. THE FIRST STRING IS OF LENGTH M1 AND STARTS AT **
C*** TCOS(I1+1). THE SECOND STRING IS OF LENGTH M2 AND STARTS AT **
C*** TCOS(I2+1). THE MERGED STRING GOES INTO TCOS(I3+1). **
C*** **
C*****
C***
      DIMENSION TCOS(1)
C***
C*****
C***
      J1 = 1
      J2 = 1
      J = I3
      IF (M1 .EQ. 0) GO TO 107
      IF (M2 .EQ. 0) GO TO 104
101 J = J+1
      L = J1+I1
      X = TCOS(L)
      L = J2+I2
      Y = TCOS(L)
      IF (X-Y) 102,102,103
102 TCOS(J) = X
      J1 = J1+1
      IF (J1 .GT. M1) GO TO 106
      GO TO 101
103 TCOS(J) = Y
      J2 = J2+1
      IF (J2 .LE. M2) GO TO 101
      IF (J1 .GT. M1) GO TO 109
104 K = J-J1+1
      DO 105 J=J1,M1
          M = K+J
          L = J+I1
          TCOS(M) = TCOS(L)
105 CONTINUE
      GO TO 109
106 CONTINUE
      IF (J2 .GT. M2) GO TO 109
107 K = J-J2+1
      DO 108 J=J2,M2
          M = K+J
          L = J+I2
          TCOS(M) = TCOS(L)
108 CONTINUE
109 CONTINUE

```

```

      RETURN
      END
C***
C***
C*****
      SUBROUTINE POISD2 (MR,NR,ISTAG,BA,BB,BC,Q,IDIMQ,B,W,D,TCOS,P)
C*****
C***
C*** SUBROUTINE TO SOLVE POISSON'S EQUATION FOR DIRICHLET BOUNDARY **
C*** CONDITIONS. **
C*** ISTAG = 1 IF THE LAST DIAGONAL BLOCK IS THE MATRIX A. **
C*** ISTAG = 2 IF THE LAST DIAGONAL BLOCK IS THE MATRIX A+I. **
C*** **
C*****
C***
      DIMENSION      Q(IDIMQ,1) ,BA(1)      ,BB(1)      ,BC(1)      ,
1      TCOS(1)      ,B(1)      ,D(1)      ,W(1)      ,
2      P(1)
C***
C*****
C***
      M = MR
      N = NR
      JSH = 0
      FI = 1./FLOAT(ISTAG)
      IP = -M
      IPSTOR = 0
      GO TO (101,102),ISTAG
101 KR = 0
      IRREG = 1
      IF (N .GT. 1) GO TO 106
      TCOS(1) = 0.
      GO TO 103
102 KR = 1
      JSTSAV = 1
      IRREG = 2
      IF (N .GT. 1) GO TO 106
      TCOS(1) = -1.
103 DO 104 I=1,M
          B(I) = Q(I,1)
104 CONTINUE
      CALL TRIX (1,0,M,BA,BB,BC,B,TCOS,D,W)
      DO 105 I=1,M
          Q(I,1) = B(I)
105 CONTINUE
      GO TO 183
106 LR = 0
      DO 107 I=1,M
          P(I) = 0.
107 CONTINUE
      NUN = N
      JST = 1
      JSP = N
C***

```

```

C***-----
C***  IRREG=1 when no irregularities have occurred, otherwise it is 2 :
C***-----
C***
    108 L = 2*JST
        NODD = 2-2*((NUN+1)/2)+NUN
C***
C***-----
C***  NODD=1 when NUN is odd, otherwise it is 2 :
C***-----
C***
    GO TO (110,109),NODD
    109 JSP = JSP-L
        GO TO 111
    110 JSP = JSP-JST
        IF (IRREG .NE. 1) JSP = JSP-L
    111 CONTINUE
C***
C***-----
C***  Regular reduction :
C***-----
C***
    CALL COSGEN (JST,1,0.5,0.0,TCOS)
    IF (L .GT. JSP) GO TO 118
    DO 117 J=L,JSP,L
        JM1 = J-JSH
        JP1 = J+JSH
        JM2 = J-JST
        JP2 = J+JST
        JM3 = JM2-JSH
        JP3 = JP2+JSH
        IF (JST .NE. 1) GO TO 113
        DO 112 I=1,M
            B(I) = 2.*Q(I,J)
            Q(I,J) = Q(I,JM2)+Q(I,JP2)
112     CONTINUE
            GO TO 115
113     DO 114 I=1,M
            T = Q(I,J)-Q(I,JM1)-Q(I,JP1)+Q(I,JM2)+Q(I,JP2)
            B(I) = T+Q(I,J)-Q(I,JM3)-Q(I,JP3)
            Q(I,J) = T
114     CONTINUE
115     CONTINUE
        CALL TRIX (JST,0,M,BA,BB,BC,B,TCOS,D,W)
        DO 116 I=1,M
            Q(I,J) = Q(I,J)+B(I)
116     CONTINUE
117 CONTINUE
C***
C***-----
C***  Reduction for last unknown :
C***-----
C***
    118 GO TO (119,136),NODD

```

```

119 GO TO (152,120),IRREG
C***
C***-----
C*** Odd number of unknowns :
C***-----
C***
120 JSP = JSP+L
    J = JSP
    JM1 = J-JSH
    JP1 = J+JSH
    JM2 = J-JST
    JP2 = J+JST
    JM3 = JM2-JSH
    GO TO (123,121),ISTAG
121 CONTINUE
    IF (JST.NE. 1) GO TO 123
    DO 122 I=1,M
        B(I) = Q(I,J)
        Q(I,J) = 0.
122 CONTINUE
    GO TO 130
123 GO TO (124,126),NODDPR
124 DO 125 I=1,M
    IP1 = IP+I
    B(I) = .5*(Q(I,JM2)-Q(I,JM1)-Q(I,JM3))+P(IP1)+Q(I,J)
125 CONTINUE
    GO TO 128
126 DO 127 I=1,M
    B(I) = .5*(Q(I,JM2)-Q(I,JM1)-Q(I,JM3))+Q(I,JP2)-Q(I,JP1)+Q(I,J)
127 CONTINUE
128 DO 129 I=1,M
    Q(I,J) = .5*(Q(I,J)-Q(I,JM1)-Q(I,JP1))
129 CONTINUE
130 CALL TRIX (JST,0,M,BA,BB,BC,B,TCOS,D,W)
    IP = IP+M
    IPSTOR = MAX0(IPSTOR,IP+M)
    DO 131 I=1,M
        IP1 = IP+I
        P(IP1) = Q(I,J)+B(I)
        B(I) = Q(I,JP2)+P(IP1)
131 CONTINUE
    IF (LR.NE. 0) GO TO 133
    DO 132 I=1,JST
        KRPI = KR+I
        TCOS(KRPI) = TCOS(I)
132 CONTINUE
    GO TO 134
133 CONTINUE
    CALL COSGEN (LR,JSTSAV,0.,FI,TCOS(JST+1))
    CALL MERGE (TCOS,0,JST,JST,LR,KR)
134 CONTINUE
    CALL COSGEN (KR,JSTSAV,0.0,FI,TCOS)
    CALL TRIX (KR,KR,M,BA,BB,BC,B,TCOS,D,W)
    DO 135 I=1,M

```

```

        IP1 = IP+I
        Q(I,J) = Q(I,JM2)+B(I)+P(IP1)
135 CONTINUE
        LR = KR
        KR = KR+L
        GO TO 152

C***
C***-----
C*** Even number of unknowns :
C***-----
C***
136 JSP = JSP+L
    J = JSP
    JM1 = J-JSH
    JP1 = J+JSH
    JM2 = J-JST
    JP2 = J+JST
    JM3 = JM2-JSH
    GO TO (137,138),IRREG
137 CONTINUE
    JSTSAV = JST
    IDEG = JST
    KR = L
    GO TO 139
138 CALL COSGEN (KR,JSTSAV,0.0,FI,TCOS)
    CALL COSGEN (LR,JSTSAV,0.0,FI,TCOS(KR+1))
    IDEG = KR
    KR = KR+JST
139 IF (JST.NE. 1) GO TO 141
    IRREG = 2
    DO 140 I=1,M
        B(I) = Q(I,J)
        Q(I,J) = Q(I,JM2)
140 CONTINUE
    GO TO 150
141 DO 142 I=1,M
    B(I) = Q(I,J)+.5*(Q(I,JM2)-Q(I,JM1)-Q(I,JM3))
142 CONTINUE
    GO TO (143,145),IRREG
143 DO 144 I=1,M
    Q(I,J) = Q(I,JM2)+.5*(Q(I,J)-Q(I,JM1)-Q(I,JP1))
144 CONTINUE
    IRREG = 2
    GO TO 150
145 CONTINUE
    GO TO (146,148),NODDPR
146 DO 147 I=1,M
    IP1 = IP+I
    Q(I,J) = Q(I,JM2)+P(IP1)
147 CONTINUE
    IP = IP-M
    GO TO 150
148 DO 149 I=1,M
    Q(I,J) = Q(I,JM2)+Q(I,J)-Q(I,JM1)

```

```

149 CONTINUE
150 CALL TRIX (IDEG, LR, M, BA, BB, BC, B, TCOS, D, W)
    DO 151 I=1, M
        Q(I, J) = Q(I, J) + B(I)
151 CONTINUE
152 NUN = NUN/2
    NODDPR = NODD
    JSH = JST
    JST = 2*JST
    IF (NUN .GE. 2) GO TO 108
C***
C***-----
C*** Start solution :
C***-----
C***
    J = JSP
    DO 153 I=1, M
        B(I) = Q(I, J)
153 CONTINUE
    GO TO (154, 155), IRREG
154 CONTINUE
    CALL COSGEN (JST, 1, 0.5, 0.0, TCOS)
    IDEG = JST
    GO TO 156
155 KR = LR + JST
    CALL COSGEN (KR, JSTSAV, 0.0, FI, TCOS)
    CALL COSGEN (LR, JSTSAV, 0.0, FI, TCOS (KR+1))
    IDEG = KR
156 CONTINUE
    CALL TRIX (IDEG, LR, M, BA, BB, BC, B, TCOS, D, W)
    JM1 = J - JSH
    JP1 = J + JSH
    GO TO (157, 159), IRREG
157 DO 158 I=1, M
        Q(I, J) = .5*(Q(I, J) - Q(I, JM1) - Q(I, JP1)) + B(I)
158 CONTINUE
    GO TO 164
159 GO TO (160, 162), NODDPR
160 DO 161 I=1, M
    IP1 = IP + I
        Q(I, J) = P(IP1) + B(I)
161 CONTINUE
    IP = IP - M
    GO TO 164
162 DO 163 I=1, M
        Q(I, J) = Q(I, J) - Q(I, JM1) + B(I)
163 CONTINUE
164 CONTINUE
C***
C***-----
C*** Start back substitution :
C***-----
C***
    JST = JST/2

```

```

JSH = JST/2
NUN = 2*NUN
IF (NUN .GT. N) GO TO 183
DO 182 J=JST,N,L
  JM1 = J-JSH
  JP1 = J+JSH
  JM2 = J-JST
  JP2 = J+JST
  IF (J .GT. JST) GO TO 166
  DO 165 I=1,M
    B(I) = Q(I,J)+Q(I,JP2)
165  CONTINUE
    GO TO 170
166  IF (JP2 .LE. N) GO TO 168
    DO 167 I=1,M
      B(I) = Q(I,J)+Q(I,JM2)
167  CONTINUE
      IF (JST .LT. JSTSAV) IRREG = 1
      GO TO (170,171),IRREG
168  DO 169 I=1,M
      B(I) = Q(I,J)+Q(I,JM2)+Q(I,JP2)
169  CONTINUE
170  CONTINUE
    CALL COSGEN (JST,1,0.5,0.0,TCOS)
    IDEG = JST
    JDEG = 0
    GO TO 172
171  IF (J+L .GT. N) LR = LR-JST
    KR = JST+LR
    CALL COSGEN (KR,JSTSAV,0.0,FI,TCOS)
    CALL COSGEN (LR,JSTSAV,0.0,FI,TCOS(KR+1))
    IDEG = KR
    JDEG = LR
172  CONTINUE
    CALL TRIK (IDEG,JDEG,M,BA,BB,BC,B,TCOS,D,W)
    IF (JST .GT. 1) GO TO 174
    DO 173 I=1,M
      Q(I,J) = B(I)
173  CONTINUE
    GO TO 182
174  IF (JP2 .GT. N) GO TO 177
175  DO 176 I=1,M
    Q(I,J) = .5*(Q(I,J)-Q(I,JM1)-Q(I,JP1))+B(I)
176  CONTINUE
    GO TO 182
177  GO TO (175,178),IRREG
178  IF (J+JSH .GT. N) GO TO 180
    DO 179 I=1,M
      IP1 = IP+I
      Q(I,J) = B(I)+P(IP1)
179  CONTINUE
      IP = IP-M
      GO TO 182
180  DO 181 I=1,M

```

```

      Q(I,J) = B(I)+Q(I,J)-Q(I,JM1)
181  CONTINUE
182  CONTINUE
      L = L/2
      GO TO 164
183  CONTINUE
C***
C***-----
C***  Return storage requirements for P vectors :
C***-----
C***
      W(1) = IPSTOR
      RETURN
      END
C***
C***
C*****
      SUBROUTINE POISN2 (M,N,ISTAG,MIXBND,A,BB,C,Q,IDIMQ,B,B2,B3,W,W2,
1          W3,D,TCOS,P)
C*****
C***
C*** SUBROUTINE TO SOLVE POISSON'S EQUATION WITH NEUMANN BOUNDARY **
C*** CONDITIONS. **
C*** **
C*** ISTAG = 1 IF THE LAST DIAGONAL BLOCK IS A. **
C*** ISTAG = 2 IF THE LAST DIAGONAL BLOCK IS A-I. **
C*** MIXBND= 1 IF HAVE NEUMANN BOUNDARY CONDITIONS AT BOTH BOUNDARIES. **
C*** MIXBND= 2 IF HAVE NEUMANN BOUNDARY CONDITIONS AT BOTTOM AND **
C*** DIRICHLET CONDITION AT TOP. (FOR THIS CASE, MUST HAVE ISTAG = 1.) **
C*** **
C*****
C***
      DIMENSION      A(1)          ,BB(1)          ,C(1)          ,Q(IDIMQ,1) ,
1          B(1)          ,B2(1)          ,B3(1)          ,W(1)          ,
2          W2(1)          ,W3(1)          ,D(1)          ,TCOS(1)          ,
3          K(4)          ,P(1)
      EQUIVALENCE      (K(1),K1) , (K(2),K2) , (K(3),K3) , (K(4),K4)
C***
C*****
C***
      FISTAG = 3-ISTAG
      FNUM = 1./FLOAT(ISTAG)
      FDEN = 0.5*FLOAT(ISTAG-1)
      MR = M
      IP = -MR
      IPSTOR = 0
      I2R = 1
      JR = 2
      NR = N
      NLAST = N
      KR = 1
      LR = 0
      GO TO (101,103),ISTAG
101  CONTINUE

```



```

      DO 102 I=1,MR
        Q(I,N) = .5*Q(I,N)
102  CONTINUE
      GO TO (103,104),MIXBND
103  IF (N .LE. 3) GO TO 155
104  CONTINUE
      JR = 2*I2R
      NROD = 1
      IF ((NR/2)*2 .EQ. NR) NROD = 0
      GO TO (105,106),MIXBND
105  JSTART = 1
      GO TO 107
106  JSTART = JR
      NROD = 1-NROD
107  CONTINUE
      JSTOP = NLAST-JR
      IF (NROD .EQ. 0) JSTOP = JSTOP-I2R
      CALL COSGEN (I2R,1,0.5,0.0,TCOS)
      I2RBY2 = I2R/2
      IF (JSTOP .GE. JSTART) GO TO 108
      J = JR
      GO TO 116
108  CONTINUE
C***
C***-----
C*** Regular reduction :
C***-----
C***
      DO 115 J=JSTART,JSTOP,JR
        JP1 = J+I2RBY2
        JP2 = J+I2R
        JP3 = JP2+I2RBY2
        JM1 = J-I2RBY2
        JM2 = J-I2R
        JM3 = JM2-I2RBY2
        IF (J .NE. 1) GO TO 109
        JM1 = JP1
        JM2 = JP2
        JM3 = JP3
109  CONTINUE
        IF (I2R .NE. 1) GO TO 111
        IF (J .EQ. 1) JM2 = JP2
        DO 110 I=1,MR
          B(I) = 2.*Q(I,J)
          Q(I,J) = Q(I,JM2)+Q(I,JP2)
110  CONTINUE
          GO TO 113
111  CONTINUE
          DO 112 I=1,MR
            FI = Q(I,J)
            Q(I,J) = Q(I,J)-Q(I,JM1)-Q(I,JP1)+Q(I,JM2)+Q(I,JP2)
            B(I) = FI+Q(I,J)-Q(I,JM3)-Q(I,JP3)
112  CONTINUE
113  CONTINUE

```

```

      CALL TRIX (I2R, 0, MR, A, BB, C, B, TCOS, D, W)
      DO 114 I=1, MR
        Q(I, J) = Q(I, J) + B(I)
114    CONTINUE
C***
C***-----
C*** End of reduction for regular unknowns :
C***-----
C***
      115 CONTINUE
C***
C***-----
C*** Begin special reduction for last unknown :
C***-----
C***
      J = JSTOP+JR
116 NLAST = J
      JM1 = J-I2RBY2
      JM2 = J-I2R
      JM3 = JM2-I2RBY2
      IF (NROD .EQ. 0) GO TO 128
C***
C***-----
C*** Odd number of unknowns :
C***-----
C***
      IF (I2R .NE. 1) GO TO 118
      DO 117 I=1, MR
        B(I) = FISTAG*Q(I, J)
        Q(I, J) = Q(I, JM2)
117    CONTINUE
      GO TO 126
118 DO 119 I=1, MR
      B(I) = Q(I, J) + .5*(Q(I, JM2) - Q(I, JM1) - Q(I, JM3))
119 CONTINUE
      IF (NRODPR .NE. 0) GO TO 121
      DO 120 I=1, MR
        II = IP+I
        Q(I, J) = Q(I, JM2) + P(II)
120 CONTINUE
      IP = IP-MR
      GO TO 123
121 CONTINUE
      DO 122 I=1, MR
        Q(I, J) = Q(I, J) - Q(I, JM1) + Q(I, JM2)
122 CONTINUE
123 IF (LR .EQ. 0) GO TO 124
      CALL COSGEN (LR, 1, 0.5, FDEN, TCOS(KR+1))
      GO TO 126
124 CONTINUE
      DO 125 I=1, MR
        B(I) = FISTAG*B(I)
125 CONTINUE
126 CONTINUE

```

```

      CALL COSGEN (KR,1,0.5,FDEN,TCOS)
      CALL TRIK (KR,LR,MR,A,BB,C,B,TCOS,D,W)
      DO 127 I=1,MR
        Q(I,J) = Q(I,J)+B(I)
127  CONTINUE
      KR = KR+I2R
      GO TO 151
128  CONTINUE
C***
C***-----
C*** Even number of unknowns :
C***-----
C***
      JP1 = J+I2RBY2
      JP2 = J+I2R
      IF (I2R .NE. 1) GO TO 135
      DO 129 I=1,MR
        B(I) = Q(I,J)
129  CONTINUE
      CALL TRIK (1,0,MR,A,BB,C,B,TCOS,D,W)
      IP = 0
      IPSTOR = MR
      GO TO (133,130),ISTAG
130  DO 131 I=1,MR
      P(I) = B(I)
      B(I) = B(I)+Q(I,N)
131  CONTINUE
      TCOS(1) = 1.
      TCOS(2) = 0.
      CALL TRIK (1,1,MR,A,BB,C,B,TCOS,D,W)
      DO 132 I=1,MR
        Q(I,J) = Q(I,JM2)+P(I)+B(I)
132  CONTINUE
      GO TO 150
133  CONTINUE
      DO 134 I=1,MR
      P(I) = B(I)
      Q(I,J) = Q(I,JM2)+2.*Q(I,JP2)+3.*B(I)
134  CONTINUE
      GO TO 150
135  CONTINUE
      DO 136 I=1,MR
      B(I) = Q(I,J)+.5*(Q(I,JM2)-Q(I,JM1)-Q(I,JM3))
136  CONTINUE
      IF (NRODPR .NE. 0) GO TO 138
      DO 137 I=1,MR
      II = IP+I
      B(I) = B(I)+P(II)
137  CONTINUE
      GO TO 140
138  CONTINUE
      DO 139 I=1,MR
      B(I) = B(I)+Q(I,JP2)-Q(I,JP1)
139  CONTINUE

```

```

140 CONTINUE
  CALL TRIX (I2R,0,MR,A,BB,C,B,TCOS,D,W)
  IP = IP+MR
  IPSTOR = MAX0(IPSTOR,IP+MR)
  DO 141 I=1,MR
    II = IP+I
    P(II) = B(I)+.5*(Q(I,J)-Q(I,JM1)-Q(I,JP1))
    B(I) = P(II)+Q(I,JP2)
141 CONTINUE
  IF (LR .EQ. 0) GO TO 142
  CALL COSGEN (LR,1,0.5,FDEN,TCOS(I2R+1))
  CALL MERGE (TCOS,0,I2R,I2R,LR,KR)
  GO TO 144
142 DO 143 I=1,I2R
  II = KR+I
  TCOS(II) = TCOS(I)
143 CONTINUE
144 CALL COSGEN (KR,1,0.5,FDEN,TCOS)
  IF (LR .NE. 0) GO TO 145
  GO TO (146,145),ISTAG
145 CONTINUE
  CALL TRIX (KR,KR,MR,A,BB,C,B,TCOS,D,W)
  GO TO 148
146 CONTINUE
  DO 147 I=1,MR
    B(I) = FISTAG*B(I)
147 CONTINUE
148 CONTINUE
  DO 149 I=1,MR
    II = IP+I
    Q(I,J) = Q(I,JM2)+P(II)+B(I)
149 CONTINUE
150 CONTINUE
  LR = KR
  KR = KR+JR
151 CONTINUE
  GO TO (152,153),MIXBND
152 NR = (NLAST-1)/JR+1
  IF (NR .LE. 3) GO TO 155
  GO TO 154
153 NR = NLAST/JR
  IF (NR .LE. 1) GO TO 192
154 I2R = JR
  NRODPR = NROD
  GO TO 104
155 CONTINUE
C***
C***-----
C***  Begin solution :
C***-----
C***
  J = 1+JR
  JM1 = J-I2R
  JP1 = J+I2R

```

```

      JM2 = NLAST-I2R
      IF (NR .EQ. 2) GO TO 184
      IF (LR .NE. 0) GO TO 170
      IF (N .NE. 3) GO TO 161
C***
C***-----
C*** Case N=3 :
C***-----
C***
      GO TO (156,168),ISTAG
156 CONTINUE
      DO 157 I=1,MR
        B(I) = Q(I,2)
157 CONTINUE
      TCOS(1) = 0.
      CALL TRIK (1,0,MR,A,BB,C,B,TCOS,D,W)
      DO 158 I=1,MR
        Q(I,2) = B(I)
        B(I) = 4.*B(I)+Q(I,1)+2.*Q(I,3)
158 CONTINUE
      TCOS(1) = -2.
      TCOS(2) = 2.
      I1 = 2
      I2 = 0
      CALL TRIK (I1,I2,MR,A,BB,C,B,TCOS,D,W)
      DO 159 I=1,MR
        Q(I,2) = Q(I,2)+B(I)
        B(I) = Q(I,1)+2.*Q(I,2)
159 CONTINUE
      TCOS(1) = 0.
      CALL TRIK (1,0,MR,A,BB,C,B,TCOS,D,W)
      DO 160 I=1,MR
        Q(I,1) = B(I)
160 CONTINUE
      JR = 1
      I2R = 0
      GO TO 194
C***
C***-----
C*** Case N = 2**P+1 :
C***-----
C***
161 CONTINUE
      GO TO (162,170),ISTAG
162 CONTINUE
      DO 163 I=1,MR
        B(I) = Q(I,J)+.5*Q(I,1)-Q(I,JM1)+Q(I,NLAST)-Q(I,JM2)
163 CONTINUE
      CALL COSGEN (JR,1,0.5,0.0,TCOS)
      CALL TRIK (JR,0,MR,A,BB,C,B,TCOS,D,W)
      DO 164 I=1,MR
        Q(I,J) = .5*(Q(I,J)-Q(I,JM1)-Q(I,JP1))+B(I)
        B(I) = Q(I,1)+2.*Q(I,NLAST)+4.*Q(I,J)
164 CONTINUE

```

```

      JR2 = 2*JR
      CALL COSGEN (JR,1,0.0,0.0,TCOS)
      DO 165 I=1, JR
         I1 = JR+I
         I2 = JR+1-I
         TCOS(I1) = -TCOS(I2)
165  CONTINUE
      CALL TRIX (JR2,0,MR,A,BB,C,B,TCOS,D,W)
      DO 166 I=1,MR
         Q(I,J) = Q(I,J)+B(I)
         B(I) = Q(I,1)+2.*Q(I,J)
166  CONTINUE
      CALL COSGEN (JR,1,0.5,0.0,TCOS)
      CALL TRIX (JR,0,MR,A,BB,C,B,TCOS,D,W)
      DO 167 I=1,MR
         Q(I,1) = .5*Q(I,1)-Q(I,JM1)+B(I)
167  CONTINUE
      GO TO 194

C***
C***-----
C*** Case of general N with NR = 3 :
C***-----
C***
168 DO 169 I=1,MR
      B(I) = Q(I,2)
      Q(I,2) = 0.
      B2(I) = Q(I,3)
      B3(I) = Q(I,1)
169  CONTINUE
      JR = 1
      I2R = 0
      J = 2
      GO TO 177
170  CONTINUE
      DO 171 I=1,MR
         B(I) = .5*Q(I,1)-Q(I,JM1)+Q(I,J)
171  CONTINUE
      IF (NROD .NE. 0) GO TO 173
      DO 172 I=1,MR
         II = IP+I
         B(I) = B(I)+P(II)
172  CONTINUE
      GO TO 175
173 DO 174 I=1,MR
         B(I) = B(I)+Q(I,NLAST)-Q(I,JM2)
174  CONTINUE
175  CONTINUE
      DO 176 I=1,MR
         T = .5*(Q(I,J)-Q(I,JM1)-Q(I,JP1))
         Q(I,J) = T
         B2(I) = Q(I,NLAST)+T
         B3(I) = Q(I,1)+2.*T
176  CONTINUE
177  CONTINUE

```

```

      K1 = KR+2*JR-1
      K2 = KR+JR
      TCOS(K1+1) = -2.
      K4 = K1+3-ISTAG
      CALL COSGEN (K2+ISTAG-2,1,0.0,FNUM,TCOS(K4))
      K4 = K1+K2+1
      CALL COSGEN (JR-1,1,0.0,1.0,TCOS(K4))
      CALL MERGE (TCOS,K1,K2,K1+K2,JR-1,0)
      K3 = K1+K2+LR
      CALL COSGEN (JR,1,0.5,0.0,TCOS(K3+1))
      K4 = K3+JR+1
      CALL COSGEN (KR,1,0.5,FDEN,TCOS(K4))
      CALL MERGE (TCOS,K3,JR,K3+JR,KR,K1)
      IF (LR .EQ. 0) GO TO 178
      CALL COSGEN (LR,1,0.5,FDEN,TCOS(K4))
      CALL MERGE (TCOS,K3,JR,K3+JR,LR,K3-LR)
      CALL COSGEN (KR,1,0.5,FDEN,TCOS(K4))
178  K3 = KR
      K4 = KR
      CALL TRI3 (MR,A,BB,C,K,B,B2,B3,TCOS,D,W,W2,W3)
      DO 179 I=1,MR
         B(I) = B(I)+B2(I)+B3(I)
179  CONTINUE
      TCOS(1) = 2.
      CALL TRIX (1,0,MR,A,BB,C,B,TCOS,D,W)
      DO 180 I=1,MR
         Q(I,J) = Q(I,J)+B(I)
         B(I) = Q(I,1)+2.*Q(I,J)
180  CONTINUE
      CALL COSGEN (JR,1,0.5,0.0,TCOS)
      CALL TRIX (JR,0,MR,A,BB,C,B,TCOS,D,W)
      IF (JR .NE. 1) GO TO 182
      DO 181 I=1,MR
         Q(I,1) = B(I)
181  CONTINUE
      GO TO 194
182  CONTINUE
      DO 183 I=1,MR
         Q(I,1) = .5*Q(I,1)-Q(I,JM1)+B(I)
183  CONTINUE
      GO TO 194
184  CONTINUE
      IF (N .NE. 2) GO TO 188
C***
C***-----
C*** Case N = 2
C***-----
C***
      DO 185 I=1,MR
         B(I) = Q(I,1)
185  CONTINUE
      TCOS(1) = 0.
      CALL TRIX (1,0,MR,A,BB,C,B,TCOS,D,W)
      DO 186 I=1,MR

```

```

      Q(I,1) = B(I)
      B(I) = 2.*(Q(I,2)+B(I))*FISTAG
186 CONTINUE
      TCOS(1) = -FISTAG
      TCOS(2) = 2.
      CALL TRIX (2,0,MR,A,BB,C,B,TCOS,D,W)
      DO 187 I=1,MR
        Q(I,1) = Q(I,1)+B(I)
187 CONTINUE
      JR = 1
      I2R = 0
      GO TO 194
188 CONTINUE
C***
C***-----
C*** Case of general N and NR = 2 :
C***-----
C***
      DO 189 I=1,MR
        II = IP+I
        B3(I) = 0.
        B(I) = Q(I,1)+2.*P(II)
        Q(I,1) = .5*Q(I,1)-Q(I,JM1)
        B2(I) = 2.*(Q(I,1)+Q(I,NLAST))
189 CONTINUE
      K1 = KR+JR-1
      TCOS(K1+1) = -2.
      K4 = K1+3-ISTAG
      CALL COSGEN (KR+ISTAG-2,1,0.0,FNUM,TCOS(K4))
      K4 = K1+KR+1
      CALL COSGEN (JR-1,1,0.0,1.0,TCOS(K4))
      CALL MERGE (TCOS,K1,KR,K1+KR,JR-1,0)
      CALL COSGEN (KR,1,0.5,FDEN,TCOS(K1+1))
      K2 = KR
      K4 = K1+K2+1
      CALL COSGEN (LR,1,0.5,FDEN,TCOS(K4))
      K3 = LR
      K4 = 0
      CALL TRI3 (MR,A,BB,C,K,B,B2,B3,TCOS,D,W,W2,W3)
      DO 190 I=1,MR
        B(I) = B(I)+B2(I)
190 CONTINUE
      TCOS(1) = 2.
      CALL TRIX (1,0,MR,A,BB,C,B,TCOS,D,W)
      DO 191 I=1,MR
        Q(I,1) = Q(I,1)+B(I)
191 CONTINUE
      GO TO 194
192 DO 193 I=1,MR
      B(I) = Q(I,NLAST)
193 CONTINUE
      GO TO 196
194 CONTINUE
C***

```



```

C***-----
C***  Start back substitution :
C***-----
C***
      J = NLAST-JR
      DO 195 I=1,MR
        B(I) = Q(I,NLAST)+Q(I,J)
195  CONTINUE
196  JM2 = NLAST-I2R
      IF (JR .NE. 1) GO TO 198
      DO 197 I=1,MR
        Q(I,NLAST) = 0.
197  CONTINUE
      GO TO 202
198  CONTINUE
      IF (NROD .NE. 0) GO TO 200
      DO 199 I=1,MR
        II = IP+I
        Q(I,NLAST) = P(II)
199  CONTINUE
      IP = IP-MR
      GO TO 202
200  DO 201 I=1,MR
        Q(I,NLAST) = Q(I,NLAST)-Q(I,JM2)
201  CONTINUE
202  CONTINUE
      CALL COSGEN (KR,1,0.5,FDEN,TCOS)
      CALL COSGEN (LR,1,0.5,FDEN,TCOS(KR+1))
      IF (LR .NE. 0) GO TO 204
      DO 203 I=1,MR
        B(I) = FISTAG*B(I)
203  CONTINUE
204  CONTINUE
      CALL TRIX (KR,LR,MR,A,BB,C,B,TCOS,D,W)
      DO 205 I=1,MR
        Q(I,NLAST) = Q(I,NLAST)+B(I)
205  CONTINUE
      NLASTP = NLAST
206  CONTINUE
      JSTEP = JR
      JR = I2R
      I2R = I2R/2
      IF (JR .EQ. 0) GO TO 222
      GO TO (207,208),MIXBND
207  JSTART = 1+JR
      GO TO 209
208  JSTART = JR
209  CONTINUE
      KR = KR-JR
      IF (NLAST+JR .GT. N) GO TO 210
      KR = KR-JR
      NLAST = NLAST+JR
      JSTOP = NLAST-JSTEP
      GO TO 211

```

```

210 CONTINUE
    JSTOP = NLAST-JR
211 CONTINUE
    LR = KR-JR
    CALL COSGEN (JR,1,0.5,0.0,TCOS)
    DO 221 J=JSTART,JSTOP,JSTEP
        JM2 = J-JR
        JP2 = J+JR
        IF (J .NE. JR) GO TO 213
        DO 212 I=1,MR
            B(I) = Q(I,J)+Q(I,JP2)
212 CONTINUE
        GO TO 215
213 CONTINUE
        DO 214 I=1,MR
            B(I) = Q(I,J)+Q(I,JM2)+Q(I,JP2)
214 CONTINUE
215 CONTINUE
        IF (JR .NE. 1) GO TO 217
        DO 216 I=1,MR
            Q(I,J) = 0.
216 CONTINUE
        GO TO 219
217 CONTINUE
        JM1 = J-I2R
        JP1 = J+I2R
        DO 218 I=1,MR
            Q(I,J) = .5*(Q(I,J)-Q(I,JM1)-Q(I,JP1))
218 CONTINUE
219 CONTINUE
        CALL TRIX (JR,0,MR,A,BB,C,B,TCOS,D,W)
        DO 220 I=1,MR
            Q(I,J) = Q(I,J)+B(I)
220 CONTINUE
221 CONTINUE
        NROD = 1
        IF (NLAST+I2R .LE. N) NROD = 0
        IF (NLASTP .NE. NLAST) GO TO 194
        GO TO 206
222 CONTINUE
C***
C***-----
C*** Return storage requirements for P vectors :
C***-----
C***
        W(1) = IPSTOR
        RETURN
        END
C***
C***
C*****
        SUBROUTINE POISP2 (M,N,A,BB,C,Q,IDIMQ,B,B2,B3,W,W2,W3,D,TCOS,P)
C*****
C***

```

```

C***   SUBROUTINE TO SOLVE POISSON EQUATION WITH PERIODIC BOUNDARY   **
C***   CONDITIONS.                                                  **
C***                                                                **
C*****
C***
      DIMENSION      A(1)      ,BB(1)      ,C(1)      ,Q(IDIMQ,1) ,
1      B(1)      ,B2(1)      ,B3(1)      ,W(1)      ,
2      W2(1)      ,W3(1)      ,D(1)      ,TCOS(1)      ,
3      P(1)
C***
C*****
C***
      MR = M
      NR = (N+1)/2
      NRM1 = NR-1
      IF (2*NR .NE. N) GO TO 107
C***
C***-----
C***   Even number of unknowns :
C***-----
C***
      DO 102 J=1,NRM1
          NRMJ = NR-J
          NRPJ = NR+J
          DO 101 I=1,MR
              S = Q(I,NRMJ)-Q(I,NRPJ)
              T = Q(I,NRMJ)+Q(I,NRPJ)
              Q(I,NRMJ) = S
              Q(I,NRPJ) = T
101      CONTINUE
102 CONTINUE
      DO 103 I=1,MR
          Q(I,NR) = 2.*Q(I,NR)
          Q(I,N) = 2.*Q(I,N)
103 CONTINUE
      CALL POISD2 (MR,NRM1,1,A,BB,C,Q,IDIMQ,B,W,D,TCOS,P)
      IPSTOR = W(1)
      CALL POISN2 (MR,NR+1,1,1,A,BB,C,Q(1,NR),IDIMQ,B,B2,B3,W,W2,W3,D,
1      TCOS,P)
      IPSTOR = MAX0(IPSTOR,INT(W(1)))
      DO 105 J=1,NRM1
          NRMJ = NR-J
          NRPJ = NR+J
          DO 104 I=1,MR
              S = .5*(Q(I,NRPJ)+Q(I,NRMJ))
              T = .5*(Q(I,NRPJ)-Q(I,NRMJ))
              Q(I,NRMJ) = S
              Q(I,NRPJ) = T
104      CONTINUE
105 CONTINUE
      DO 106 I=1,MR
          Q(I,NR) = .5*Q(I,NR)
          Q(I,N) = .5*Q(I,N)
106 CONTINUE

```

```

      GO TO 118
107 CONTINUE
C***
C***-----
C*** Odd number of unknowns :
C***-----
C***
      DO 109 J=1,NRM1
        NRPJ = N+1-J
        DO 108 I=1,MR
          S = Q(I,J)-Q(I,NRPJ)
          T = Q(I,J)+Q(I,NRPJ)
          Q(I,J) = S
          Q(I,NRPJ) = T
108 CONTINUE
109 CONTINUE
      DO 110 I=1,MR
        Q(I,NR) = 2.*Q(I,NR)
110 CONTINUE
      LH = NRM1/2
      DO 112 J=1,LH
        NRMJ = NR-J
        DO 111 I=1,MR
          S = Q(I,J)
          Q(I,J) = Q(I,NRMJ)
          Q(I,NRMJ) = S
111 CONTINUE
112 CONTINUE
      CALL POISD2 (MR,NRM1,2,A,BB,C,Q,IDIMQ,B,W,D,TCOS,P)
      IPSTOR = W(1)
      CALL POISN2 (MR,NR,2,1,A,BB,C,Q(1,NR),IDIMQ,B,B2,B3,W,W2,W3,D,
1 TCOS,P)
      IPSTOR = MAX0(IPSTOR,INT(W(1)))
      DO 114 J=1,NRM1
        NRPJ = NR+J
        DO 113 I=1,MR
          S = .5*(Q(I,NRPJ)+Q(I,J))
          T = .5*(Q(I,NRPJ)-Q(I,J))
          Q(I,NRPJ) = T
          Q(I,J) = S
113 CONTINUE
114 CONTINUE
      DO 115 I=1,MR
        Q(I,NR) = .5*Q(I,NR)
115 CONTINUE
      DO 117 J=1,LH
        NRMJ = NR-J
        DO 116 I=1,MR
          S = Q(I,J)
          Q(I,J) = Q(I,NRMJ)
          Q(I,NRMJ) = S
116 CONTINUE
117 CONTINUE
118 CONTINUE

```

```

C***
C***-----
C***  Return storage requirements for P vectors :
C***-----
C***
      W(1) = IPSTOR
      RETURN
      END

C***
C***
C*****
      SUBROUTINE TRIX(IDEGBR, IDEGCR, M, A, B, C, Y, TCOS, D, W)
C*****
C***
C*** SUBROUTINE TO SOLVE A SYSTEM OF LINEAR EQUATIONS WHERE THE
C*** COEFFICIENT MATRIX IS A RATIONAL FUNCTION IN THE MATRIX GIVEN BY
C*** TRIDIAGONAL ( . . . , A(I), B(I), C(I), . . . ).
C***
C*****
C***
      DIMENSION      A(1)      ,B(1)      ,C(1)      ,Y(1)      ,
1      TCOS(1)      ,D(1)      ,W(1)

C***
C*****
C***
      MM1 = M-1
      FB = IDEGBR+1
      FC = IDEGCR+1
      L = FB/FC
      LINT = 1
      DO 108 K=1, IDEGBR
        X = TCOS(K)
        IF (K .NE. L) GO TO 102
        I = IDEGBR+LINT
        XX = X-TCOS(I)
        DO 101 I=1,M
          W(I) = Y(I)
          Y(I) = XX*Y(I)
101      CONTINUE
102      CONTINUE
        Z = 1./(B(1)-X)
        D(1) = C(1)*Z
        Y(1) = Y(1)*Z
        DO 103 I=2,MM1
          Z = 1./(B(I)-X-A(I)*D(I-1))
          D(I) = C(I)*Z
          Y(I) = (Y(I)-A(I)*Y(I-1))*Z
103      CONTINUE
        Z = B(M)-X-A(M)*D(MM1)
        IF (Z .NE. 0.) GO TO 104
        Y(M) = 0.
        GO TO 105
104      Y(M) = (Y(M)-A(M)*Y(MM1))/Z
105      CONTINUE

```

```

DO 106 IP=1,MM1
  I = M-IP
  Y(I) = Y(I)-D(I)*Y(I+1)
106 CONTINUE
  IF (K .NE. L) GO TO 108
  DO 107 I=1,M
    Y(I) = Y(I)+W(I)
107 CONTINUE
  LINT = LINT+1
  L = (FLOAT(LINT)*FB)/FC
108 CONTINUE
  RETURN
END

C***
C***
C*****
      SUBROUTINE TRI3 (M,A,B,C,K,Y1,Y2,Y3,TCOS,D,W1,W2,W3)
C*****
C***
C*** SUBROUTINE TO SOLVE THREE LINEAR SYSTEMS WHOSE COMMON COEFFICIENT**
C*** MATRIX IS A RATIONAL FUNCTION IN THE MATRIX GIVEN BY
C*** TRIDIAGONAL (... ,A(I),B(I),C(I),...)
C***
C*****
C***
      DIMENSION      A(1)      ,B(1)      ,C(1)      ,K(4)      ,
      1              TCOS(1)   ,Y1(1)     ,Y2(1)     ,Y3(1)     ,
      2              D(1)      ,W1(1)     ,W2(1)     ,W3(1)
C***
C*****
C***
      MM1 = M-1
      K1 = K(1)
      K2 = K(2)
      K3 = K(3)
      K4 = K(4)
      F1 = K1+1
      F2 = K2+1
      F3 = K3+1
      F4 = K4+1
      K2K3K4 = K2+K3+K4
      IF (K2K3K4 .EQ. 0) GO TO 101
      L1 = F1/F2
      L2 = F1/F3
      L3 = F1/F4
      LINT1 = 1
      LINT2 = 1
      LINT3 = 1
      KINT1 = K1
      KINT2 = KINT1+K2
      KINT3 = KINT2+K3
101 CONTINUE
      DO 115 N=1,K1
        X = TCOS(N)

```

```

      IF (K2K3K4 .EQ. 0) GO TO 107
      IF (N .NE. L1) GO TO 103
      DO 102 I=1,M
        W1(I) = Y1(I)
102    CONTINUE
103    IF (N .NE. L2) GO TO 105
      DO 104 I=1,M
        W2(I) = Y2(I)
104    CONTINUE
105    IF (N .NE. L3) GO TO 107
      DO 106 I=1,M
        W3(I) = Y3(I)
106    CONTINUE
107    CONTINUE
      Z = 1./ (B(1)-X)
      D(1) = C(1)*Z
      Y1(1) = Y1(1)*Z
      Y2(1) = Y2(1)*Z
      Y3(1) = Y3(1)*Z
      DO 108 I=2,M
        Z = 1./ (B(I)-X-A(I)*D(I-1))
        D(I) = C(I)*Z
        Y1(I) = (Y1(I)-A(I)*Y1(I-1))*Z
        Y2(I) = (Y2(I)-A(I)*Y2(I-1))*Z
        Y3(I) = (Y3(I)-A(I)*Y3(I-1))*Z
108    CONTINUE
      DO 109 IP=1,MM1
        I = M-IP
        Y1(I) = Y1(I)-D(I)*Y1(I+1)
        Y2(I) = Y2(I)-D(I)*Y2(I+1)
        Y3(I) = Y3(I)-D(I)*Y3(I+1)
109    CONTINUE
      IF (K2K3K4 .EQ. 0) GO TO 115
      IF (N .NE. L1) GO TO 111
      I = LINT1+KINT1
      XX = X-TCOS(I)
      DO 110 I=1,M
        Y1(I) = XX*Y1(I)+W1(I)
110    CONTINUE
      LINT1 = LINT1+1
      L1 = (FLOAT(LINT1)*F1)/F2
111    IF (N .NE. L2) GO TO 113
      I = LINT2+KINT2
      XX = X-TCOS(I)
      DO 112 I=1,M
        Y2(I) = XX*Y2(I)+W2(I)
112    CONTINUE
      LINT2 = LINT2+1
      L2 = (FLOAT(LINT2)*F1)/F3
113    IF (N .NE. L3) GO TO 115
      I = LINT3+KINT3
      XX = X-TCOS(I)
      DO 114 I=1,M
        Y3(I) = XX*Y3(I)+W3(I)

```

```
114    CONTINUE
      LINT3 = LINT3+1
      L3 = (FLOAT(LINT3)*F1)/F4
115 CONTINUE
      RETURN
      END
```


BIBLIOGRAPHY

BIBLIOGRAPHY

1. Abernathy, F. H. and Kronauer, R. E., "The formation of vortex streets", *Journal of Fluid Mechanics*, Vol. 13, p. 1, (1962).
2. Ashurst, W. T., "Numerical Simulation of Turbulent Mixing Layers Via Vortex Dynamics", *Turbulent Shear Flows I*, edited by F. Durst et. al., Springer-Verlag, Berlin, pp. 402-413, (1979).
3. Ashurst, W. T., "Calculation of Plane Sudden Expansion Flow via Vortex Dynamics", *2nd Shear Flow*, London, (1979).
4. Ashurst, W. T., "Piston-Cylinder Fluid Motion via Vortex Dynamics", *Fluid Mechanics of Combustion Systems*, edited by T. Morel et. al., Fluid Engineering Division, ASME, Boulder Colorado, (1981).
5. Ashurst, W. T., "Lagrangian-Eulerian Calculation of Turbulence Effects on a Premixed Flame Propagation", presented at the 1981 Spring Meeting of the Western States Section of the Combustion Institute, paper WSS/CI/81-23, (1981).
6. Baker, G. R., "The Cloud-In-Cell Technique applied to the Roll-up of Vortex Sheet", *Journal of Computational Physics*, Vol.31, pp. 76-95, (1979)
7. Bartholomew, R. W., Private communication. (1985)
8. Beal, J. T. and Majda, A., "Vortex Methods II : Higher Order Accuracy in Two and Three Dimensions", *Mathematics of Computation*, Vol. 39, pp. 29-52, (1982).
9. Birdsall, C. K. and Fuss, D., "Clouds-In-Clouds, Clouds-In-Cells Physics for Many-Body Plasma Simulation", *Journal of Computational Physics*, Vol.3, pp. 494-511, (1969).
10. Birkhoff, G. and Fisher, J., "Do Vortex Sheets Roll Up?", *Rend. Circ. Mat. Palermo Ser. 2*, 8, p. 77, (1959).

11. Bowman, F., "Introduction to Elliptic Functions with Applications", Dover Publications, Inc., New York, (1961).
12. Brown, G. L. and Roshko, A. J., "On Density Effects and Large Structures in Turbulent Mixing-Layer Growth at Moderate Reynolds Number", *Journal of Fluid Mechanics*, Vol. 64, pp. 775-816, (1974).
13. Burt, R. and Troth, K. A., "Penetration and Vaporization of Diesel Fuel Sprays", Symposium on "Diesel Engine Combustion", Institution of Mechanical Engineers, (1970).
14. Carslaw, H. S. and Jaeger, J. C., Conduction in Solids, Oxford University Press, London, England, p. 256, (1959).
15. Chorin, A. J., "Numerical Methods for use in Combustion Modeling", *Proceedings of the International Conference on Numerical Methods in Science and Engineering*, Versailles, France, pp. 229-236, (1979).
16. Chorin, A. J., "Numerical Studies of Slightly Viscous Flow", *Journal of Fluid Mechanics*, Vol. 57, pp. 785-796, (1973).
17. Chorin, A. J., "Vortex Sheet Approximation of Boundary Layers", *Journal of Computational Physics*, Vol. 27, pp. 428-442, (1978).
18. Chorin, A. J., "Vortex Models and Boundary Layer Instabilities", *SIAM Journal of Scientific and Statistical Computations*, Vol. 1, pp. 1-24, (1980).
19. Chorin, A. J. and Bernard, P. S., "Discretization of a Vortex Sheet, with an Example of Roll-Up", *Journal of Computational Physics*, Vol. 13, pp. 423-429, (1973).
20. Christiansen, J. P., "Numerical Simulation of Hydrodynamics by the Method of Point Vortices", *J. Comp. Phys.*, Vol. 13, pp. 363-379, (1973).
21. Clements, R. R. and Maull, D. J., "The Representation of Sheets of Vorticity by Discrete Vortices", *Prog. Aeronaut. Sci.*, Vol. 6, pp. 129-146, (1975).
22. Corrsin, S., "Simple Theory of an Idealized Turbulent Mixer", *A.I.Ch.E.J.* Vol. 3, p. 329., (1957)
23. Couet, B., Buneman, O. and Leonard, A., "Simulation of Three-Dimensional Incompressible Flows with Vortex-In-Cell Methods", *J. Comp. Phys.*, Vol. 39, pp. 305-328, (1981).

24. Couet, B. and Leonard, A., "Exact Extension to the Three-Dimensional Incompressible Flows with Vortex Elements", *Ann. Rev. Fluid Mech.*, Vol. 17, pp. 523-559, (?).
25. Courant, R., Friedrichs, K., and Lewy, H., "Über die Partiellen Differenzengleichungen der Mathematischen Physik", *IBM J.* p.215, (March 1967); *Math. Ann.*, Vol. 100, p.32, (1928).
26. Curl, R. L., "Dispersed Phase Mixing : I. Theory and Effects of Simple Reactors", *A.I.Ch.E.J.*, Vol. 9, pp. 175-181, (1963).
27. Dimotakis, P. E. and Brown, G. L., "The Mixing Layer at High Reynolds Number : Large Structure Dynamics and Entrainment", *Journal of Fluid Mechanics*, Vol. 78, p.535, (1976).
28. Dopazo, C. and O'Brien, E. E., *Acta Astronaut*, Vol. 1, p. 1239, (1974).
29. Dopazo, C. and O'Brien, E. E., 4th. Int. Coll. on Gasdynamics of Explosions and Reactive Systems, La Jolla, (1973).
30. Dopazo, C. and O'Brien, E. E., "Statistical Treatment of Non-isothermal Chemical Reactions in Turbulence", *Combustion Science and Technology*, Vol. 13, p. 99, (1976).
31. Dopazo, C., "Probability Density Function Approach for a Turbulent Axisymmetric Heated Jet Centerline Evolution", *Phys. Fluids*, Vol. 22, pp. 20-30, (1979).
32. Dorodnicyn, A. A., "Review of methods for solving Navier-Stokes equations", Proceedings of 3rd International Conference on Numerical Methods in Fluid Mechanics, *Lecture Notes in Physics*, Springer-Verlag, pp. 1-11. (1973).
33. Dyer, T. M. "Characterization of One- and Two-Dimensional Homogeneous Combustion Phenomena in a Constant Volume Bomb", Sandia National Labs, Report SAND78-8704.
34. Einstein, A., Investigation on the Theory of the Brownian Movement, Translation, Methuen and Co., Ltd., London, (1926); Reprint, Dover Publications, Inc., New York, (1956).
35. Evangelista, J. J. , Shinnar, R. and Katz, in Twelfth Symposium International on Combustion. (The Combustion Institute, Pittsburg, Pa.), p. 401, (1968).

36. Flagan, R. C. and Appleton, J. P., "A Stochastic Model of Turbulent Mixing with Chemical Reaction: Nitric Oxide Formation in a Plug-Flow Burner", *Combustion and Flame*, Vol. 23, p. 249, (1974).
37. Fox, R. L., "Multipoint Distribution Function Hierarchy for Compressible Turbulent Flows", *Physics of Fluids*, Vol. 18, p. 1245, (1975).
38. Gosman, A. D., Johns, R. J. R. and Watkins, A. P., "Development of Prediction Methods for In-Cylinder Processes in Reciprocating Engines", Combustion Modeling in Reciprocating Engines, edited by J. N. Mattavi and C. A. Amann, Plenum Press, New York, pp. 69-129, (1980).
39. Hald, C. and Del Prete, V. M., "Convergence of Vortex Methods for Euler's Equations", *Mathematics of Computation*, Vol. 32, pp. 791-809, (1978).
40. Hammersley, J. M. and Handscomb, D. C., Monte Carlo Methods, Metheun, London, (1964).
41. Hockney, R. W., Goel, S. P. and Eastwood, J. W., "Quiet High-Resolution Computer Models of a Plasma", *J. Comp. Phys.*, Vol. 14, pp. 148-158, (1974).
42. Janicka, J., Kolbe, W. and Kolmann, W., "The Solution of a PDF-Transport Equation For Turbulent Diffusion Flames", in *Proceedings of the Heat Transfer in Fluid Mechanics Institute* (Standford University Press, Standford), p. 296, (1978).
43. Kraichnan, R. H. and Montgomery, D., "Two-dimensional turbulence", *Rep. Prog. Phys.*, Vol. 43, pp. 547-619, (1980).
44. Lamberti, J. Probability, Benjamin, New York, (1966)
45. Leonard, A., "Vortex Methods for Flow Simulations", *Journal of Computational Physics*, Vol. 37, pp. 289-335, (1980).
46. Levenspiel, O. and Spielman, L. A., "A Monte Carlo Treatment for Reacting and Coalescing Dispersed Phase Systems", *Chem. Engng. Sci.*, Vol. 20, p. 247, (1965).
47. Lundgren, T. S., "Distribution Functions in the Statistical Theory of Turbulence", *Physics of Fluids*, Vol. 10, p. 969, (1967).

48. Lyn, W. T., "Study of Burning Rate and Nature of Combustion in Diesel Engines", 9th. Symposium of Combustion, pp. 1069-1082, (1960).
49. Meng, J. C. S. and Thomson, J. A. L., "Numerical Studies of some Non-Linear Hydrodynamic Problems by Discrete Vortex Element Methods", Journal of Fluid Mechanics, Vol. 84, part 3, pp. 433-453, (1978).
50. Milinazzo, F. and Saffman, P. G., "The Calculation of Large Reynolds Number Two-Dimensional Flow Using Discrete Vortices with Random Walk", Journal of Computational Physics, Vol. 23, pp. 380-392, (1977).
51. Milne-Thomson, L. M., "Hydrodynamical Images", Proceedings of the Cambridge Philosophical Society, Vol. 36, pp. 246-247, (1940).
52. Moore, D. W., "The Discrete Vortex Approximation of a Finite Vortex Sheet", California Institute of Technology, Report AFOSR-1804-69, (1971).
53. Newman, J. A. and Brzustowski, T. A., "Behaviour of a Liquid Jet Near the Thermodynamic Critical Region", AIAA Journal, Vol. 9, 1971, No. 8.
54. Pope, S. B., Philos. Trans. R. Soc. London, Ser. A291, 529 (1979).
55. Pope, S. B., "The Probability Approach to the Modeling of Turbulent Reacting Flow", Combustion and Flame, Vol. 27, p. 299, (1976).
56. Pope, S. B., "A Monte Carlo Method for the PDF Equations of Turbulent Reactive Flow", Combustion Science and Technology, Vol. 25, pp. 159-174, (1981).
57. Pratt, D. T., Prog. "Mixing and Chemical Reaction in Continuous Combustion", Energy Combustion Sci., Vol. 1, p. 73, (1976).
58. Rhines, P. B., "Geostrophic turbulence", Ann. Rev. Fluid Mech., Vol. 11, pp. 401-41, (1979).
59. Rife, J. and Heywood, J. B., "Photographic and Performance Studies of Diesel Combustion with a Rapid Compression Machine", SAE Transactions, Vol. 83, Section 4, pp. 2942-2961, (1974).
60. Rosenhead, L., Proc. Royal Soc. Vol. A134, p. 170, (1931).
61. Roshko, A. J., "Structures and Turbulent Shear Flows : a new look", AIAA Journal, Vol. 14, pp. 1349-1357, (1976).

62. Rossow, V. J., "Convective merging of vortex cores in lift-generated wakes", *J. Aircr.*, Vol. 14, pp. 283-90, (1977).
63. Shahed, S. M., Chiu, W. S. and Lyn, W. T., "A Transient Spray Mixing Model for Diesel Combustion", A Mathematical model of diesel combustion. Conference on Combustion Engines, Cranfield, (July, 1975).
64. Shestakov, A. I., "Numerical Solution of the Navier-Stokes Equations at High Reynolds Numbers", Ph.D. Thesis, Department of Mathematics, University of California at Berkeley, (1975).
65. Shreider, Yu A., (Ed.) The Monte Carlo Method, Pergamon, London, (1966).
66. Spalding, D. B., "A General Theory of Turbulent Combustion, the Lagrangian Aspects", AIAA 15th. Aerospace Science Meeting, L. A., California, Report 77-141.
67. Stratonovich, R. L., "Topics in the Theory of Random Noise", New York, Vol. 1, Chapters 1 & 4, (1963).
68. Swartztrauber, P., Sweet, R. and Adams, J. Source listings for FISHPAK from the National Center for Atmospheric Research, Boulder, CO., (1986).
69. Takami, H., "Numerical Experiment with Discrete Vortex Approximation, with Reference to the Rolling Up of a Vortex Sheet", Dept. of Aero. and Astro., Stanford University, Report SUDAER 202, (1964).
70. Wang, S. S., "Grid-Insensitive Computer Simulation of the Kelvin-Helmholtz Instability and Shear Flow Turbulence", Ph.D. Thesis, Stanford University Institute for Plasma Research Report No. 710, (1977).
71. Westwater, F. L., British A. R. C. Reports and Memoranda No. 1692, p.116, (1936).
72. Winant, C. D. and Browand, F. K., "Vortex Pairing : the Mechanism of Turbulent Mixing Layer Growth at Moderate Reynolds Numbers", *Journal of Fluid Mechanics*, Vol. 63, part 2, pp. 237-255, (1974).
73. Yanenko, N. N., The Method of Fractional Steps, Springer Verlag, New York, (1971).

- 74 Zabusky, N. J., "Coherent Structures in Fluid Dynamics". In the Significance of Nonlinearity in the Natural Sciences. Ed. A. Perlmutter, L. F. Scott, pp. 45-205, New York : Plenum. 423 pp. (1977).
75. Zabusky, N. J., "Recent Developments in Contour Dynamics for the Euler Equations", Ann. N. Y. Acad. Sci., Vol. 373, pp. 160-170, (1981).