University of Windsor
Scholarship at UWindsor

Electronic Theses and Dissertations Theses, Dissertations, and Major Papers

1987

A numerical study of particle collection by single water droplets.

Kevin R. J. Ellwood
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

Ellwood, Kevin R. J., "A numerical study of particle collection by single water droplets." (1987). *Electronic Theses and Dissertations*. 690.
https://scholar.uwindsor.ca/etd/690

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

Canada

A NUMERICAL STUDY
OF
PARTICLE COLLECTION
BY
SINGLE WATER DROPLETS

By

Kevin R. J. Ellwood

A Thesis Submitted to the
Faculty of Graduate Studies and Research
Through the Department of Chemical Engineering
in Partial Fulfillment of the Requirements for the
Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada
1987

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-39608-3

© KEVIN R. J. ELLWOOD 1987

To My Precious Mother and Father

ABSTRACT

Particle collection efficiencies of water droplets were calculated by numerically integrating the generalized particle equation of motion using a variable step-size Runge-Kutta method. Considerations were given to the collection mechanisms of thermophoresis, diffusiophoresis, inertial impaction, and wake capture, as well as the effects of drop deformation. The computations were carried out for a droplet temperature of 10°C with saturated gas temperatures of 20°C, 35°C, 65°C, and 95°C. The temperature, water vapor, and velocity distributions around the collector were determined by direct numerical integration of the appropriate governing partial differential equation using a convenient orthogonal grid generation technique.

The results of this investigation show that:

- depostion of fine particles ($0.1\mu m \leq r_p \leq 5\mu m$) can be significantly enhanced by phoretic forces
- flux deposition of fine particles can be related to Reynolds number through the proportionality:

$$E_{flux} \alpha \; Nre^{-0.78}$$

- the flux deposition of fine particles on the rear of the collector is significant for low Reynolds numbers
- wake capture is relatively insignificant, although definite mechanisms were established in terms of the interactions of flux forces and the hydrodynamic forces of the fluid motion.

- drop deformation can improve the collection of larger particles by inertial impaction as a result of an increase in projected area over that of an undeformed droplet.
- drop deformation can improve the collection of smaller particles as a result of increased hydrodynamic affects on such particles.

Comparisons with the simplified calculations of Pilat and Prem [39] show that, at low Reynolds numbers, inaccuracies can result from the assumption of thin boundary layers in which temperature and vapor concentrations gradients are constant. In addition, their assumption of potential flow proved to be unrealistic for the specified conditions

ACKNOWLEDGEMENTS

At this point, I would like to express my deep sense of gratitude and indebtedness to my advisors, Dr. A.W. Gnyp, Dr. C.C. St. Pierre, and Dr. S. Viswanathan, for their guidance, patience, and encouragement throughout this investigation.

I am also greateful to Dr. R. Baron and Dr. O.P. Chandna for their mathematical assistance and Dr. J.A. McCorquodale for the useful literature that he provided.

Financial support from the Natural Sciences and Engineering Research Council (NSERC) made this study possible.

CONTENTS

Abstract v
Acknowledgements. vii
Contents viii
List of Figures x
List of Tables. xii

1. Introduction 1

2. Literature Survey 4

3. Particle Collection 7
    3.1 Efficiency Definition 7
    3.2 Inertial Impaction 10
    3.3 The Interception Mechanism 11
    3.4 Phoretic Forces 12
        3.4.1 Thermophoresis 12
        3.4.2 Diffusiophoresis 15
        3.4.3 Combined Phoretic Forces 17

4. Mathematical Models 19
    4.1 General Orthogonal Coordinates 20
    4.2 Equation of Particle Motion 25
    4.3 Fluid Flow Equations 29
    4.4 Heat and Mass Transfer Equations 32

5. Coordinate Generation 35

6. Computational Procedure 47
    6.1 Finite Difference Approximations 48
        6.1.1 Coordinate Generation Algorithm 51
        6.1.2 Navier-Stokes Algorithm 60
        6.1.3 Convective Diffusion Algorithm 68
    6.2 Particle Dynamic Equations 70

7. Results and Discussions 75
    7.1 Orthogonal Grids 76
    7.2 Collector Flow Field 82
        7.2.1 Viscous Flow Model 82
        7.2.2 Potential Flow Model 91
    7.3 Convective Diffusion Model 93
    7.4 Flux Deposition Model 99
    7.5 General Results 103
        7.5.1 Collection in the Wake 113
        7.5.2 The Effects of Reynolds Number 122
        7.5.3 The Effect of Particle Mass 126
        7.5.4 The Effect of Drop Deformation 130
    7.6 Comparison with Existing Models 133

8. Conclusions 143
8.1 Recommendations for Future Study 145

REFERENCES 149

NOMENCLATURE 154

APPENDIX A General Orthogonal Differential Operators . . 159

APPENDIX B Particle Acceleration in General Orthogonal Coordinates 161

APPENDIX C Optimal Acceleration Factor 165

APPENDIX D Orthogonal Grid Generation Algorithm 168

APPENDIX E The Navier-Stokes Algorithm 178

APPENDIX F Potential Flow Algorithm 189

APPENDIX G The Convective Diffusion Algorithm 198

APPENDIX H Flux Deposition Algorithm 207

APPENDIX I Solutions for Viscous Flow and Convective Diffusion Equations 222

VITA AUCTORIS 241

LIST OF FIGURES

Figure		Page
3.1	Collection Efficiency Definition	9
3.2	A Simplified Model of Thermophoresis	13
3.3	A Simplified Model of Diffusiophoresis	16
4.1	Orthogonal Transformation	21
4.2	General Unit Tangent Vector	21
5.1	Transformed Computational Domain	38
5.2	Application of Boundary Conditions	43
5.3	Grid Packing in a Unit Square	46
6.1	Rotated Axisymmetric Grid	59
6.2	Trajectory Integration Procedure	73
6.3	Grid Point Interpolation	73
7.1	An Orthogonal Grid for a Deformed Droplet	81
7.2	Viscous Flow Around a Sphere for a Reynolds Number of 200	86
7.3	Calculated Surface Vorticity ($\zeta_{surface}$)	89
7.4	Comparison of $\zeta_{surface}$ with data of LeClair [27]	90
7.5	Calculated Potential Flow Solution for a Deformed Droplet	92
7.6	Calculated Temperature/Vapor Profile Around a Droplet with a Reynolds Number of 200	95
7.7	Computed Surface Nusselt Numbers (Nu_{ℓ})	97
7.8	Comparison of Nu_{ℓ} with data of Woo [52]	98
7.9	Potential Flow Collection Efficiencies for a Sphere	100

7.10 Recalculation of the data of Pilat and Prem [39] 102
7.11 Flux Deposition for a Reynolds Number of 1.54 107
7.12 Flux Deposition for a Reynolds Number of 30 108
7.13 Flux Deposition for a Reynolds Number of 100 109
7.14 Flux Deposition for a Reynolds Number of 200 110
7.15 Flux Deposition for a Reynolds Number of 300 111
7.16 Flux Deposition for a Reynolds Number of 400 112
7.17 Rear Collision of a 0.3μm Particle 114
7.18 Radial Velocity Diagram for Rear Collision 115
7.19 Wake Entrainment of a 0.5μm Particle 118
7.20 Radial Velocity Diagram for Wake Entainment 119
7.21 Fraction of Deposition Occurring on the Front of a Collector 121
7.22 The Effects of Reynolds Number on Flux Depostion 123
7.23 The Reduction of Thermal/Diffusive Boundary Layer 125
7.24 Illustrated Minimum Collection for Nre of 30 127
7.25 Trajectories for 0.1μm, 5μm, and 10μm Particles 129
7.26 Collection Efficiency Calculations for Deformed Drops 132
7.27 Particle Trajectories for a Deformed Drop 134
7.28 Rigorous Recalculation of the Data of Pilat and Prem [39] 136
7.29 Low Reynolds Number Comparison of the Present Temperature Profile with the Simplified Profile assumed by Pilat and Prem [39] 138
7.30 High Reynolds Number Comparison of the Present Temperature Profile with the Simplified Profile assumed by Pilat and Prem [39] 142

LIST OF TABLES

Table		Page
7.1	Grid Dimensions and Packing Functions	77
7.2	Calculated Maximum and Minimum Relaxation Factors for the Navier-Stokes Algorithm	84
7.3	Comparisons of Angle of Flow Separation (θ_s)	88
7.4	Comparisons of Vortex Length (L)	88
7.5	Calculated Maximum and Minimum Relaxation Factors for the Convective Diffusion Algorithm	94

1. INTRODUCTION

Many industrial processes emit harmful fine particulate material. The need to control emissions of fine particles into the environment has led to the design and application of a variety of particulate matter control devices. One of the most effective and economical methods of cleaning dirty gas streams involves contact with water sprays. Examples of devices that apply this method of particle collection are venturi scrubbers and countercurrent spray towers.

A number of experimental and theoretical studies have been conducted on wet collection devices. A fundamental parameter that determines the effectiveness of a typical unit is the single droplet collection efficiency. Reliable experimental single drop collection data are difficult to obtain. Even when they are available they can be applied only in a qualitative manner [47]. Consequently, researchers turn to theoretical and numerical approaches to study mechanisms responsible for collection of particles by water droplets.

The phenomena known to affect the collection of aerosol particles by water droplets are [47,53]:

- inertial impaction
- interception
- thermophoresis
- diffusiophoresis
- wake capture
- Brownian motion
- turbulent diffusion
- molecular attraction
- electrical interactions

Large particles ($r_p \geq 5\mu m$) are captured primarily by inertial impaction and interception while small particles are collected as a result of phoretic forces, wake capture and Brownian motion [16,29].

The Air Quality Group at the University of Windsor, Department of Chemical Engineering has been involved in the development of wet scrubber technology. As part of this program, this investigation was to provide a rigorous model for the prediction of single droplet collection efficiencies. Emphasis was focused on particle sizes varying from 0.1μm to 2μm in radius because of the reported difficulty associated with the removal of these particles [31]. This size range is commonly referred to as the Greenfield gap after the earliest researcher to identify it [16]. It has been shown that no single mechanism is responsible for the capture of particles in this size range [29].

The principal mechanisms considered in this study are:

- inertial impaction
- thermophoresis
- diffusiophoresis.

For the particle size range under consideration in this investigation, Brownian diffusion can be neglected because it has been shown to be unimportant for particles with radii greater than 0.05μm [47].

Previous models used crude representations of the flow field, temperature distributions, and vapor distributions around the collecting droplet. In this study, these quantities were determined using accurate numerical methods to extend the reliability of the model.

The formation of a wake has long been known to promote capture of small particles [14]. The numerical methods used during this investigation allowed wake capture to be studied in terms of its interactions with phoretic forces. In addition, the generalized numerical model facilitated estimation of the affects of drop deformation on collection efficiency. This phenomenon occurs with free falling droplets whose radii are greater than 2000μm [41,42]. This drop size is within the optimum size range for countercurrent scrubber operations (mean drop radius of 1000μm to 2000μm) [31].

2. LITERATURE SURVEY

The importance of particulate matter removal from stack gases has made single droplet collection efficiency the focus of many experimental and theoretical studies. A large body of literature exists because particulate matter collection has relevance to both the control of industrial particulate matter emissions with spray scrubbers and the scavenging of atmospheric aerosol particles by clouds and rain droplets [39].

Langmuir and Blodgett [25] were perhaps the earliest researchers who computed collection efficiences using a model assuming potential flow (high Reynolds numbers) or Stokes flow (low Reynolds numbers) with the dominant collection mechanism being inertial impaction. To evaluate collection efficiencies at intermediate Reynolds numbers, Langmuir [24] later proposed a scheme of interpolation between the two limiting flow conditions. Unfortunately, this approach appears to be based on intuition since no rigorous analysis has ever been given. Other authors; such as Johnstone and Roberts [22], Ranz and Wong [44], Pemberton [37], Fonda and Herne [11], and Micheal and Norey [32]; made refinements to the basic calculations of Langmuir. Some experimental verification was also provided. All of these investigators restricted themselves to the single mechanism of inertial impaction under ideal flow conditions.

Wu [53] was one of the earliest to consider the effects of flows at intermediate Reynolds numbers on collection efficiencies of spherical obstacles. In his approach, boundary layer flow approximations were used to model the flow of the fluid around the collector. Wu reported that his calculations agreed reasonably well with experimental data. Improvements were made for the effects of intermediate Reynolds numbers by Beard and Grover [3]. These authors solved the steady state Navier-Stokes equations in spherical coordinates to provide collection efficiences of solid spheres, assuming that only inertial impaction was significant. The results of their calculations were correlated and an equation was proposed for the prediction of collection efficiency as a function of Reynolds number. Tardos et. al. [48] and Ellwood et. al. [9] also considered the effects of intermediate Reynolds numbers by using the flow field proposed by Hameleic et. al. [18]. Ellwood also provided a convenient scheme for interpolating between Stokes and potential flow efficiences on the basis of a rigorous nonlinear regression of the intermediate Reynolds number data.

Greenfield [15] appears to be the first to consider the contributions of several mechanisms to collection efficiencies. He assumed that Brownian diffusion, inertial impaction, and turbulence were responsible for scavenging of particles by rain drops. Grover et. al. [16] combined the effects of inertial impaction, phoretic forces, and electrical forces in their consideration of raindrops falling at their terminal velocities. Their flow field was determined in a manner similar to that of Beard and Grover [3]. The temperature difference between the droplet surface and the gas was due to the effect of ventilation on the droplet. Leong et. al. [29] developed a model similar to that of Grover. al. where consideration was given only to phoretic forces. The works of Leong et. al. and Grover et. al. were applied only to the case of evaporating raindrops where the two phoretic forces oppose each other.

Pilat and Prem [39] and Mehta [31] developed models for micron and submicron particle collection by phoretic forces, inertial impaction, and Brownian diffusion. Their models are more applicable to industrial wet collectors because their temperature differences were much larger than those associated with rain scavenging. Although both research groups assumed potential flow conditions and that the temperature, vapor, and particulate distributions were linear in thin boundary layers, there appears to be an inconsistency between the two models because predicted efficiencies differ by as much as 40%. The work of Ganguli [14] is similar to that of Mehta and Pilat and Prem. He considered only micron sized particles. In this particle size range, all of the models agree reasonably well.

3. PARTICLE COLLECTION

The collection of particles by moving liquid drops depends on a number of competing factors which may either enhance or retard the capture of a particle. The underlying principle of most of the collection mechanisms is acceleration of the particle towards the collector. The mechanisms considered to be responsible for particle collection by free falling droplets in a countercurrent scubber are inertial impaction, interception, thermophoresis, and diffusiophoresis.

3.1 Efficiency Definition

When considering the definition for collection efficiency, the basis upon which a particle is captured by a moving droplet must be established. The process of particle capture is very complex since consideration should be given to interactions between drops, between particles, and between drops and particles [53]. In this study, particle-particle and droplet-droplet interactions are neglected and consideration is given only to the interaction between a single particle and a single water droplet.

The collision efficiency is defined as the ratio of the number of particles predicted to strike the collector to the number of particles flowing through the projected area of the collector. For deformed collectors, this projected area is defined as the projected area of a sphere of equivalent volume. Particle collision will occur from within an area defined by trajectories that would be traced by particles that just graze the collector as shown in Figure 3.1. If Y_o is the maximum verticle displacement from the collector centerline that will allow the particle to collide with the collector, the droplet collection efficiency is given by:

$$E = \left[\frac{Y_o}{a_o}\right]^2 \qquad 3.1.1$$

where

a_o = the volume radius of the collector

A particle, upon colliding with the collector, may have one of the following fates:

- the particle could bounce off the surface of the collector
- the particle could penetrate and pass through the collector; exiting at some other point
- the particle could penetrate the surface and be captured by the collector (coalescence).

The particular fate of a given particle is a function of many variables but the dominant ones, as given by Wu [53], are:

- nature of the surface forces
- wettability of the particle
- impact velocity of the particle
- angle of contact of the particle.

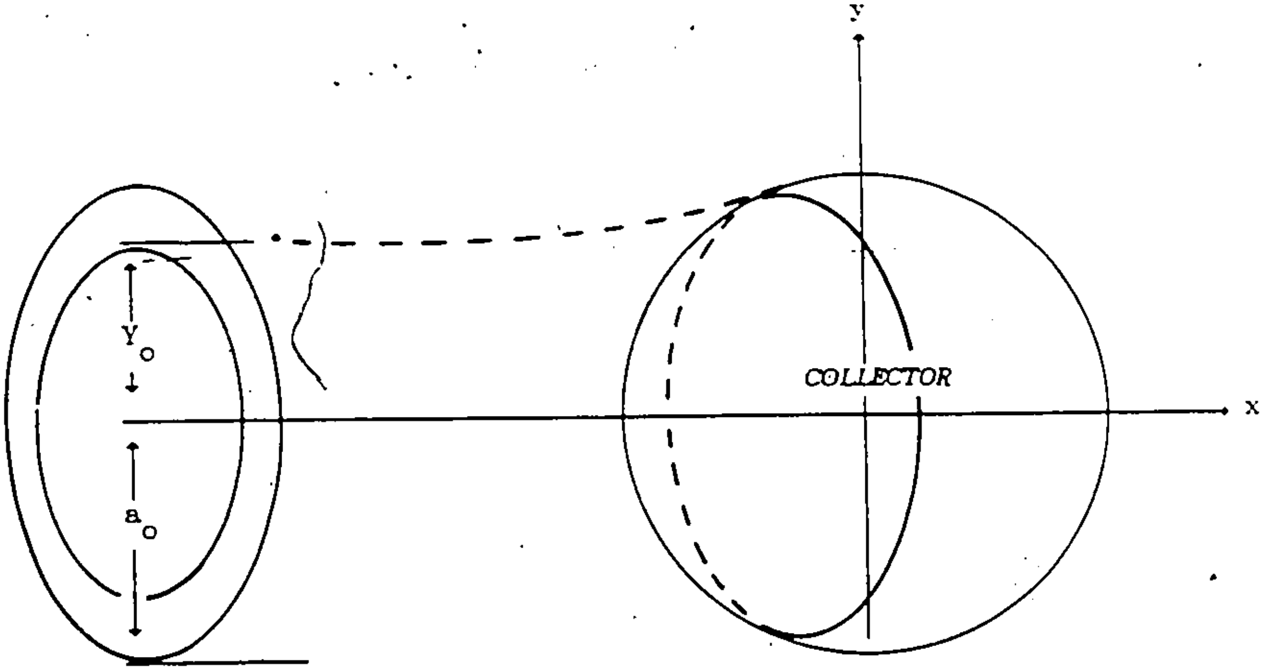

FIGURE 3.1: Collection Efficiency Definition

Taking into acount the action of all of these variables would be difficult but essential to the determination of the actual collection efficiency. Fortunately, Grover et. al. [16] point out that nearly all particles that collide with a water drop are retained by it and the collision efficiency can be taken to be identical to the collection efficiency.

3.2 Inertial Impaction

The collection of particles due to inertial impaction is the result of the particle's resistance to flow around the collector with the fluid. This resistance is due to the inertial effects of the particle mass. It can be shown that inertial impaction alone is a function of a dimensionless parameter, K, called the Stokes number [9] defined as:

$$K = \frac{2\rho_p r_p^2 U_\infty}{9\mu_f a_o C_f}$$

where

ρ_p = the particle density
r_p = the particle radius
U_∞ = the undisturbed gas velocity
μ_f = the fluid viscosity
a_o = the volume radius of the collecting body
C_f = the Cunningham correction factor.

On the basis of Newton's law of motion, a particle with a very large mass would deviate only slightly from its staight path as the fluid flows around the collector. Consequently, the particle can easily strike the collector. The converse is true if the particle has zero mass; it will follow the fluid around the collector and by-pass the obstacle to fluid flow. Using these two limiting cases, it is evident that collection should increase with increasing mass (or increasing K) when considering inertial impaction alone.

3.3 The Interception Mechanism

The interception mechanism accounts for real particle dimensions. This mechanism can be included in a model through the escape condition for the particle. This condition specifies that:

- the particle is captured if its center-line position is within one particle radius, r_p, of the collector surface
- the particle otherwise escapes capture.

Exclusion of this mechanism means that the particle is considered to be a point mass.

The efficiency of particle collection by the interception mechanism is a function of the dimensionless interception parameter:

$$Nr = \frac{r_p}{a_o}$$

It has been shown that interception is significant as a collection mechanism only when the radius of the collecting element is of the same order of magnitude as the particles being collected, or Nr has a value close to unity [1]. According to Lin and Lee [30], a particle effects the flow field of the collector at values of Nr greater than 0.1. Under such conditions, special modeling techniques must be introduced. From a practical standpoint, the value of Nr is expected to be less than 0.1 since liquid drops in most gas cleaning devices never approach the submicron size [1]. For such small values of Nr, it has been shown that interception plays a negligible role in particle collection and can be negected [1,9].

3.4 Phorectic Forces

Thermophoresis and diffusiophoresis are phoretic forces that result from the interaction of gas molecules on the surface of the particle. These forces are important to this work because they can increase the potential for fine particle collection. A detailed discussion of theoretical and experimental aspects of these forces has been provided by Alias [1].

3.4.1 Thermophoresis

Thermophoresis describes the phenomenon responsible for particles moving in the direction of the lower temperature in a fluid medium with a temperature gradient [29]. This process can be understood more readily by the use of a simple model. Refering to Figure-3.2, consider a particle suspended in a gas which is contained between two plates, one of which is heated. The gas molecules on one side of the particle will be at a higher temperature than the ones on the other side. Consequently, the molecules from the hotter side will impart more momentum to the particle as they collide with it than those approaching from the cooler side. This imbalance in momentum transfer will cause the particle to move towards the cooler plate.

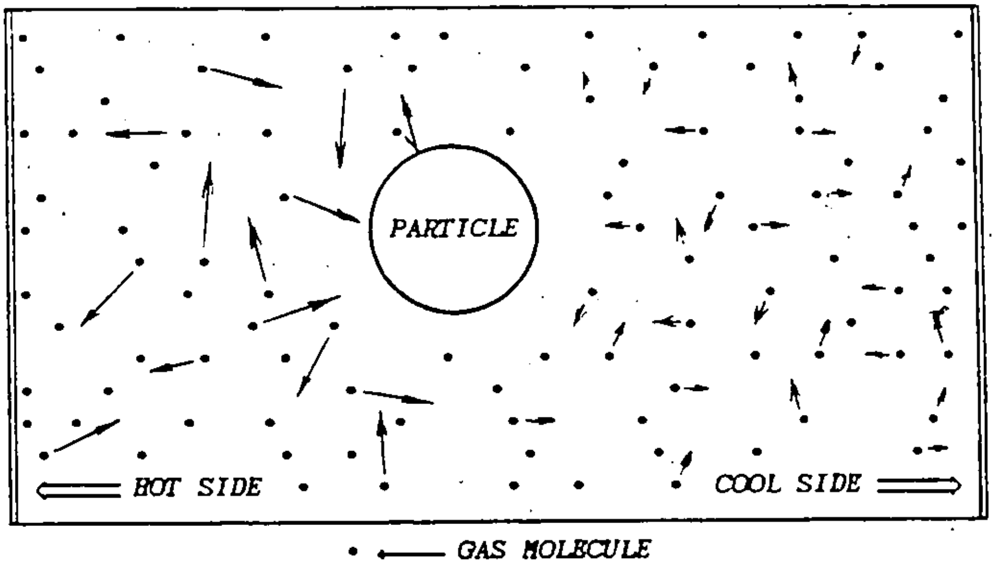

FIGURE 3.2: A Simplified Model of Thermophoresis

From the above example, it is clear that the particle motion is affected by the difference in the velocities of molecules approaching from the hot and cool sides. Thus, the particle motion is related to the temperature difference or the temperature gradient across the particle. The thermophoretic force, $\bar{F}_t$, on a particle in a uniform fluid is given by Brock [5] as:

$$\bar{F}_t = -\left[\frac{9\pi\mu_f^2 r_p}{\rho_f T_f}\left[\frac{1}{1+3C_m \cdot \mathrm{Nkn}}\right]\left[\frac{K_f/K_p + C_t \cdot \mathrm{Nkn}}{1+2K_f/K_p+2C_t \cdot \mathrm{Nkn}}\right]\bar{\nabla} T_f\right] \qquad 3.4.1$$

where

Nkn = the dimensionless Knudsun number

$= \frac{\lambda}{r_p}$

λ = mean free path of the gas

r_p = the particle radius

C_m = dimensionless constant

C_t = dimensionless constant

μ_f = viscosity of fluid

ρ_f = density of fluid

T_f = absolute fluid temperature

K_f = thermal conductivity of the fluid

K_p = thermal conductivity of the particle

In general, this equation has limitations [1] but it can be applied to the entire range of particle sizes normally encountered in gas cleaning devices as long as K_p is not too large [16,50]. For air, C_m varies between 1.0 and 1.3 while C_t ranges between 1.9 and 2.6 [29]. In this work, the values of C_m and C_t were taken to be 1.00 and 2.50 respectively. These values are used most commonly in the literature [16,31]

It is customary to write the thermophoretic force in terms of a thermophoretic velocity by equating Equation 3.4.1 to the viscous drag equation according to:

$$\bar{F}_t = \frac{6\pi\mu_f r_p \bar{V}_t}{C_f} \qquad 3.4.2$$

where

$\bar{V}_t$ = the thermoporetic velocity
C_f = the Cunningham correction factor

As a result, the thermophoretic velocity is defined as:

$$\bar{V}_t = -\left[\frac{3\mu_f C_f}{2\rho_f T_f}\left[\frac{1}{1+3C_m \cdot Nkn}\right]\left[\frac{K_f/K_p + C_t \cdot Nkn}{1+2K_f/K_p + 2C_t \cdot Nkn}\right]\bar{\nabla} T_f\right] \qquad 3.4.2$$

3.4.2 Diffusiophoresis

A diffusiophoretic force acts on a particle in an isothermal gas mixture with a concentration gradient. As in the case of thermophoresis, a simple model can be used to illustrate the process of diffusiophoresis. Refering to Figure 3.3, consider a particle suspended in a gas through which a dilute vapor is diffusing from one side. As diffusing vapor molecules strike the surface of the particle they impart momentum to it. This action will cause the particle to move in the direction of the diffusing vapor. Clearly, if a gas stream has a relatively high concentration of water vapor, particle collection will be enhanced by the diffusion of the water vapor to a cool droplet surface.

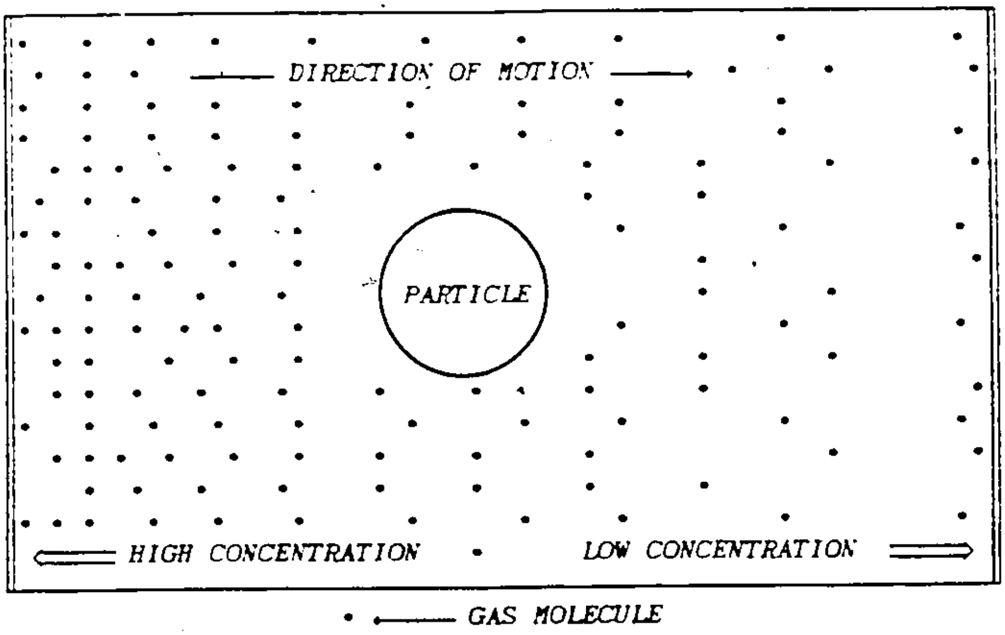

FIGURE 3.3: A Simplified Model of Diffusiophoresis

A diffusiophoretic velocity can be defined in the same manner as the thermophoretic velocity. The diffusiophoretic force can be defined as:

$$\bar{F}_d = \frac{6\pi\mu_f r_p \bar{V}_d}{C_f} \qquad 3.4.4$$

The diffusiophoretic velocity in the free molecular regime, (10 < Nkn < ∞), is given by Waldman [50] as:

$$\bar{V}_d = -\frac{M_1^{1/2}}{\gamma_1 M_1^{1/2} + \gamma_2 M_2^{1/2}} \cdot \frac{\mathcal{D}_{12}}{\gamma_2} \cdot \bar{\nabla}\gamma_1 \qquad 3.4.4$$

where

M_1 = the molecular weight of component 1
M_2 = the molecular weight of component 2
γ_1 = the mole fraction of component 1
γ_2 = the mole fraction of component 2
$\mathcal{D}_{12}$ = the diffusivity of component 1 in component 2

The theoretical limitation of this equation does not allow the entire range of particles sizes to be considered in a model. However, as pointed out by Schmitt and Waldman [16], it is accurate within 9% for particles larger than the mean free path length of the gas. Consequently, Equation 3.4.4 can by applied to particles with Nkn less than unity.

3.4.3 Combined Phoretic Forces

To calculate the trajectory of a particle, the total force on the particle must be determined. This total exerted force is simply the sum of all of the individual force terms, including the forces resulting from thermophoresis and diffusiophoresis. A problem exists because Equation 3.4.2 was derived for the case with no concentration gradients and Equation 3.4.4 was derived for the case where the gas has a uniform temperature. In other words, one phoretic force was derived assuming the other is absent.

Strictly speaking, there are coupling terms in the general expressions for the thermophoretic and diffisiophoretic forces [16]. Adding Equations 3.4.2 and 3.4.4 means that these coupling terms are neglected. As pointed out by Annis and Mason [2], if the mole fraction of the diffusing vapor is much less than the mole fraction of the gas through which it diffuses, the uncoupled phoretic force terms are additive. In this investigation, it was assumed that the mole fraction of the diffusing water vapor is small compared to the mole fraction of the gas.

4. MATHEMATICAL MODELS

Frequently, the solution of an engineering problem requires the application of mathematical relationships to simulate the behavior of the system. The complexity of the formulation clearly depends upon the problem itself, but, more importantly, it depends upon the simplifying assumptions involved. Advances in a given model are often the result of relaxing limiting assumptions, thus increasing the level of complexity of the model. The analyst must be careful to avoid the situation where the additional gain in accuracy is only marginal but the model is no longer competitive with simpler ones.

To model the capture of aerosol particles it is important to understand the influence of several important physical variables. It is clear that the particle motion will be affected by the flow of the fluid over the body of the collector; thus, an ability to model the flow of the fluid accurately is essential. When the collecting body is a water droplet, it has been shown that under certain conditions the collector shape can no longer be considered spherical [41,42]. Consequently there is a need to describe the flow of a fluid over an arbitrary body shape.

As mentioned earlier, the presence of vapor and temperature gradients around the collector would influence the path of a moving particle. As a result, some appreciation of the distribution of these variables around the collecting body becomes essential.

4.1 General Orthogonal Coordinates

In solving the various fluid mechanics equations, as well as the equations that govern particle dynamics, it would be convenient to have a coordinate system that coincides with the collector surface. For example, if the collector is spherical in shape, the surface would coincide with the surface generated in spherical polar coordinates (r,θ,ϕ) by setting the radius (r) to unity. With this simplification it is possible to write all of the hydrodynamic equations in spherical coordinates for a model of particle collection. In general, actual collectors can deviate from spherical shapes, as occurs with water droplets at high velocities or cylindrical fibres after a particulate film has accumulated [13].

The generation of boundary conforming coordinates is accomplished by devising a scheme for transforming the irregular Cartesian physical domain into a regular computational domain as shown in Figure 4.1. This transformation can be stated in general terms as:

$$x = x(\alpha_1,\alpha_2,\alpha_3)$$
$$y = y(\alpha_1,\alpha_2,\alpha_3)$$
$$z = z(\alpha_1,\alpha_2,\alpha_3)$$

or

$$\bar{r} = \bar{r}(\bar{\alpha}) \qquad 4.1.1$$

where $\bar{r} = x\cdot\bar{i}+y\cdot\bar{j}+z\cdot\bar{k}$

$\bar{\alpha} = \alpha_1\cdot\bar{e}_1+\alpha_2\cdot\bar{e}_2+\alpha_3\cdot\bar{e}_3$

$\bar{e}_i$ = a unit vector in the direction of increasing α_i

For example, the transformation equations for spherical polar

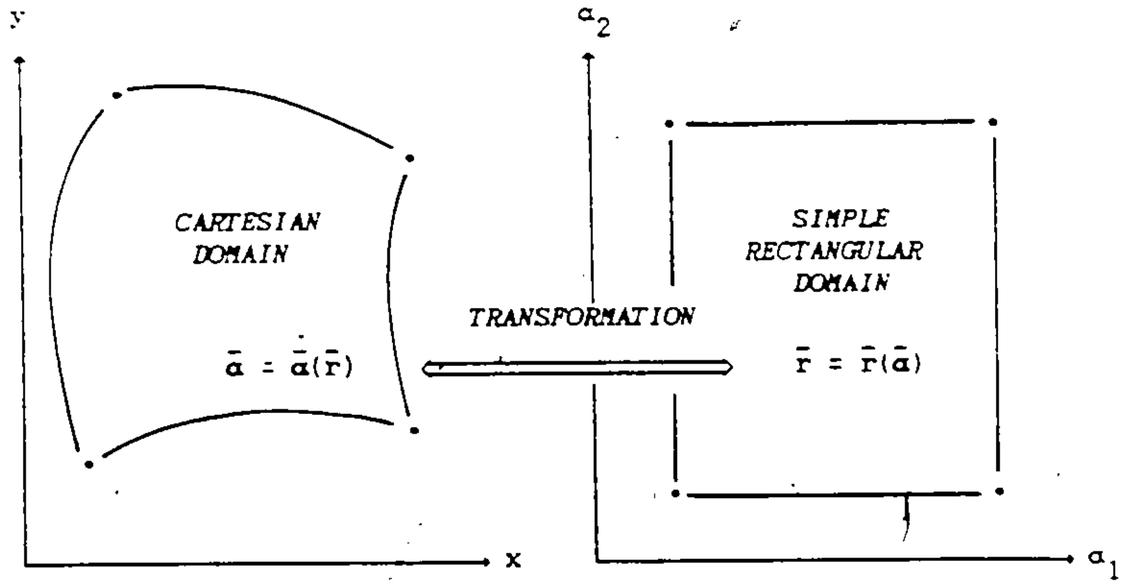
y
x
α_2
α_1
CARTESIAN DOMAIN
$\bar{\alpha} = \bar{\alpha}(\bar{r})$
TRANSFORMATION
SIMPLE RECTANGULAR DOMAIN
$\bar{r} = \bar{r}(\bar{\alpha})$

FIGURE 4.1: Orthogonal Transformation

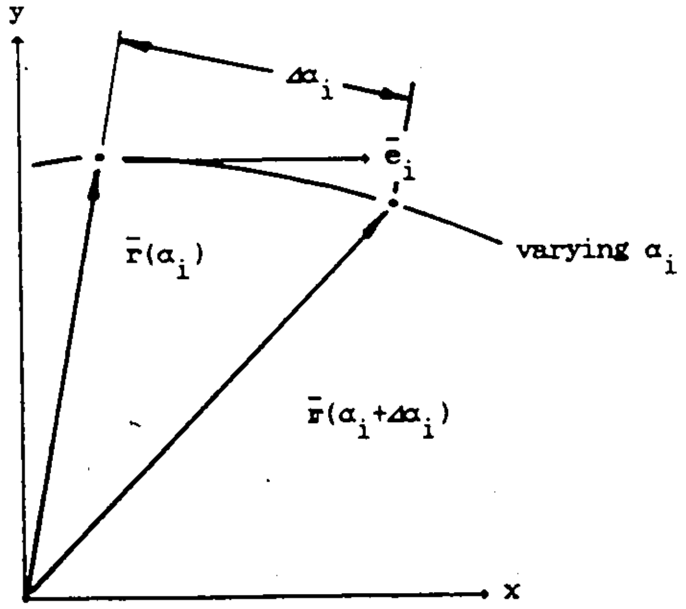

FIGURE 4.2: General Unit Tangent Vector

coordinates are given by:

$$x = \alpha_1 \cdot \sin(\alpha_2) \cdot \cos(\alpha_3)$$
$$y = \alpha_1 \cdot \sin(\alpha_2) \cdot \sin(\alpha_3)$$
$$z = \alpha_1 \cdot \cos(\alpha_2)$$

where $\bar{\alpha} = r \cdot \bar{e}_r + \theta \cdot \bar{e}_\theta + \phi \cdot \bar{e}_\phi$

The relationship between $\bar{r}$ and $\bar{\alpha}$ is assumed

- to be single-valued
- to have continuous derivatives so that the correspondence between the coordinates is unique.

With respect to Figure 4.2, consider a coordinate line along which only α_i varies. A tangent to the curve $\bar{r}(\alpha_i)$ is given by:

$$\lim_{\delta\alpha_i \to 0} \frac{\bar{r}(\alpha_i + \delta\alpha_i) - \bar{r}(\alpha_i)}{\delta\alpha_i} = \frac{\partial \bar{r}}{\partial \alpha_i}$$

Also, from Figure 4.2, a unit vector in the direction of increasing α_i is:

$$\bar{e}_i = \frac{\partial \bar{r}}{\partial \alpha_i} \cdot \left| \frac{\partial \bar{r}}{\partial \alpha_i} \right|^{-1} \qquad 4.1.3$$

At this point it is necessary to impose the condition where the α_i coordinate lines are mutually perpendicular in the physical domain.

Consequently, at a position $\bar{r}$, the tangents to these orthogonal coordinate lines are themselves mutually perpendicular; therefore:

$$\bar{e}_i \cdot \bar{e}_j = 0 \quad i \neq j$$

or

$$\frac{\partial \bar{r}}{\partial \alpha_i} \cdot \frac{\partial \bar{r}}{\partial \alpha_j} = 0 \quad i \neq j \qquad 4.1.4$$

In general, it can be shown by a simple chain rule that a differential increment in $\bar{r}$ is given by

$$d\bar{r} = \sum_{i=1}^{3} \frac{\partial \bar{r}}{\partial \alpha_i} \cdot d\alpha_i \qquad 4.1.5$$

An increment of arc length, given a differential increment in $\bar{r}$, is given by:

$$(dS)^2 = d\bar{r} \cdot d\bar{r} = \sum_{i=1}^{3} \sum_{j=1}^{3} \frac{\partial \bar{r}}{\partial \alpha_i} \cdot \frac{\partial \bar{r}}{\partial \alpha_j} d\alpha_i d\alpha_j \qquad 4.1.6$$

Equation 4.1.6 allows the introduction of some standard terminology. The arc length increment depends on the dot product of the tangent vectors to the coordinate curves. These dot products form a symetric covariant metric tensor as follows [49]:

$$\begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{bmatrix}$$

where $g_{ij} = \frac{\partial \bar{r}}{\partial \alpha_i} \cdot \frac{\partial \bar{r}}{\partial \alpha_j}$

Earlier it was assumed that the $\bar{\alpha}$ coordinate lines were mutually orthogonal in the physical plane; therefore, by Equation 4.1.4, all of the off-diagonal terms in the transformation metric tensor are zero. Often, quantities called coordinate scale factors (h_i) are introduced for convenience when the coordinate system is orthogonal. These scale factors are given by:

$$h_i = \left| \frac{\partial \bar{r}}{\partial \alpha_i} \right| = \sqrt{g_{ii}} \qquad 4.1.7$$

At this point, all the information necessary to develop a general model has been given. Differential operators; such as

- the gradient $(\bar{\nabla}\phi)$
- the divergence $(\bar{\nabla}\cdot\bar{\phi})$
- the curl $(\bar{\nabla}\times\bar{\phi})$

have equivalent forms to their Cartesian counterparts in this new coordinate system. Appendix A contains the necessary differential operators for further model development. A more extensive and detailed development of generalized coordinate transformations is provided by Thompson [49].

4.2 Equation of Particle Motion

The equation of particle motion can be readily derived from Newton's law of motion when several simplifications are made. Fundamental assumptions specify that:

- the particle size is so small in comparison to the collector size that it will not influence the collector flow field
- particle–particle interactions are negligible
- heat and mass transfer between the particle and fluid are negligible
- the particle is spherical in shape.

Generally, many more assumptions must be made to completely define a physical model, but they are not needed in the formulation of the equation of particle motion.

A simple force balance on a particle leads to the relationship:

$$m_p \bar{A} = \bar{F}_{drag} + \bar{F}_e \qquad 4.2.1$$

where $\bar{A}$ = particle acceleration vector

$\bar{F}_{drag}$ = drag force vector

$\bar{F}_e$ = external force vector

m_p = particle mass

The drag force on an object can be expressed as:

$$\bar{F}_{drag} = \frac{1}{2}\rho_f \cdot \frac{Cd}{C_f}(\bar{U} - \bar{V}_p)\,|\bar{U} - \bar{V}_p|A_p \qquad 4.2.2$$

where ρ_f = fluid density
Cd = drag coefficient
A_p = projected particle area
$\bar{U}$ = fluid velocity vector
$\bar{V}_p$ = particle velocity vector
C_f = Knudsen-Weber correction factor

The Knudsen-Weber correction factor, C_f, must be introduced into the familiar drag force expression (Equation 4.2.2) in order to correct for the reduction of the drag force from the theoretical value when the particle size is of the order of the mean free path length of the gas. This deviation from theory results when such small particles experience velocity slip at the surface because the fluid is no longer a continuous medium with respect to the particle [53]. The correction is a function of the dimensionless Knudsen number (N_{kn}) and its final form is [1,53]:

$$C_f = 1 + N_{kn}(1.257 + 0.40\exp[-1.1N_{kn}^{-1}]) \qquad 4.2.3$$

where N_{kn} = Knudsen number
$= \frac{\lambda}{r_p}$
λ = mean free path length of the gas
r_p = particle radius

Equations 4.2.1 and 4.2.2 can be combined and rearranged into a dimensionless form by making the following substitutions:

$$\bar{A} = \frac{U_\infty^2}{r_c}\,\bar{a}$$

$$\bar{F}_e = \frac{U_\infty^2\, m_p}{r_c}\,\bar{f}_e$$

$$\bar{U} = U_\infty\,\bar{u}$$

$$\bar{V}_p = U_\infty\,\bar{v}_p$$

$$T = \frac{r_c}{U_\infty}\,t$$

where U_∞ = the undisturbed gas velocity
r_c = volume radius of collector
T = time
t = dimensionless time
$\bar{a}$ = dimensionless acceleration vector
$\bar{f}_e$ = dimensionless external force vector
$\bar{u}$ = dimensionless fluid velocity vector
$\bar{v}_p$ = dimensionless particle velocity vector

Combining equation 4.2.1 and 4.2.2 and making the appropriate substitutions yeilds:

$$\bar{a} = \frac{1}{2K}\left(\bar{u} - \bar{v}_p \right) + \bar{f}_e \qquad 4.2.4$$

where K = Stokes number
$= \dfrac{C_f \rho_p U_\infty r_p^2}{9\,\mu_f r_c}$
ρ_p = particle density
μ_f = fluid viscosity

The deviation of the particle from Stokesian behavior is taken into account by the factor γ which is given by:

$$\gamma = \frac{Cd\ Nre_p}{24}$$

where Nre_p = particle Reynolds number.

The value of γ has been expressed in the form:

$$\gamma = 1 + \frac{Nre_p}{109} + 0.15\ Nre_p^{0.6} \qquad 4.2.5$$

by using the expression for the drag coefficient given by Dickinson and Marshall [9].

Equations 4.2.3 and 4.2.4 can be solved simultaneously given any external force, $\bar{f}_e$, to produce particle trajectories. It should be noted that the particle acceleration vector has a complicated form in a generalized coordinate system. The particle acceleration vector, as derived in Appendix B, is given by:

$$a_1 = h_1 \frac{d^2\, \alpha_1}{dt^2} + \frac{\partial\, h_1}{\partial \alpha_1} \left[\frac{d\, \alpha_1}{dt}\right]^2 + 2\frac{\partial\, h_1}{\partial \alpha_2} \left[\frac{d\, \alpha_1}{dt} \frac{d\, \alpha_2}{dt}\right] - \frac{h_2}{h_1} \frac{\partial\, h_2}{\partial \alpha_1} \left[\frac{d\, \alpha_2}{dt}\right]^2 \qquad 4.2.6$$

$$a_2 = h_2 \frac{d^2\, \alpha_2}{dt^2} + \frac{\partial\, h_2}{\partial \alpha_2} \left[\frac{d\, \alpha_2}{dt}\right]^2 + 2\frac{\partial\, h_2}{\partial \alpha_1} \left[\frac{d\, \alpha_1}{dt} \frac{d\, \alpha_2}{dt}\right] - \frac{h_1}{h_2} \frac{\partial\, h_1}{\partial \alpha_2} \left[\frac{d\, \alpha_1}{dt}\right]^2 \qquad 4.2.7$$

where $\bar{a} = a_1\,\bar{e}_1 + a_2\,\bar{e}_2$

4.3 Fluid Flow Equations

Fundamental to the evaluation of particle trajectories is the establishment of the velocity field around the collector. The fluid, like the particle, must obey several conservation laws at every point in the fluid. These laws involve

- the conservation of mass
- the conservation of linear momentum.

In order to accurately predict the flow of a Newtonian fluid at intermediate Reynolds numbers the Navier-Stokes equations must be solved. These equations, which relate to the conservation of linear momentum, take the following form:

$$\rho_f \frac{D\,\bar{U}}{DT} = -\bar{\nabla}p + \frac{\mu_f}{3}\bar{\nabla}(\bar{\nabla}\cdot\bar{U}) + \mu_f\nabla^2\bar{U} \qquad 4.3.1$$

This vector equation is subject to the constraint

$$\frac{\partial\,\rho_f}{\partial T} + \bar{\nabla}\cdot(\rho_f\bar{U}) = 0 \qquad 4.3.2$$

which results from the conservation of mass.

In the derivation of Equation 4.3.1 it has been assumed that the viscosity, μ_f, is constant. In addition, it would be convenient to assume that the density is constant since this condition would lead to considerable simplifications. For moderate temperature differences between the collector and the gas stream, the dependence of viscosity and density on temperature can be neglected because the local values of these quantities should approximate the temperature averaged values. For example, if the temperature of the fluid and the collector are 65°C and 10°C respectively, the maximum deviation of viscosity and density from the averaged quantities is approximately 7%. The compressibility of the gas due to inertial effects can be neglected because the velocities of the collectors considered are much less than the speed of sound [53].

Using these simplifications and applying the dimensionless transformation discussed in Section 4.2 it follows that:

$$\frac{\partial \bar{u}}{\partial t} = -\bar{\nabla}(p^{*}+\frac{1}{2}u^{2}) + \bar{u}\times(\bar{\nabla}\times\bar{u}) + \frac{2}{Nre}\nabla^{2}\bar{u} \qquad 4.3.3$$

$$\bar{\nabla}\cdot\bar{u} = 0 \qquad 4.3.4$$

where

$$p^{*} = \frac{p}{\rho_f U_\infty^{2}}$$

$$Nre = \text{fluid Reynolds number} = \frac{2\rho_f U_\infty a_o}{\mu_f}$$

and

$$\frac{D\bar{u}}{Dt} = \frac{\partial \bar{u}}{\partial t} + (\bar{u}\cdot\bar{\nabla})\bar{u} = \frac{\partial \bar{u}}{\partial t} + \frac{1}{2}\bar{\nabla}u^{2} - \bar{u}\times(\bar{\nabla}\times\bar{u}) \qquad 4.3.5$$

The form of the substantial derivative as given in Equation 4.3.5 must be used for Equation 4.2.3 to be applicable to general orthogonal coordinates [4].

The form of the momentum equation used to solve for the fluid flow involves the vorticity/stream function formulation. This equation is obtained by taking the curl of Equation 4.3.3 to remove the pressure term and introducing the stream function. Appendix A contains all of the necessary differential operators in general orthogonal coordinates to permit formulation of Equation 4.3.4 as:

$$\frac{\partial}{\partial \alpha_1}(u_1 h_2 h_3) + \frac{\partial}{\partial \alpha_2}(u_2 h_1 h_3) = 0$$

assuming that the velocity component in the α_3 direction is zero (this represents the case of either planar or axisymetric flow). Equation 4.2.4 is satisfied everywhere if the stream function, ψ, is introduced as:

$$u_1 = \frac{1}{h_1 h_3} \cdot \frac{\partial \psi}{\partial \alpha_2} \quad ; \quad u_2 = -\frac{1}{h_2 h_3} \cdot \frac{\partial \psi}{\partial \alpha_1} \qquad 4.3.5$$

Removing the pressure term from Equation 4.3.3 and using Equation 4.2.5, the final form of the vorticity transport equation becomes:

$$\frac{\partial}{\partial t}(D^2 \psi) = \frac{h_3}{h_1 h_2} \left[\frac{\partial \psi}{\partial \alpha_1} \frac{\partial f}{\partial \alpha_2} - \frac{\partial \psi}{\partial \alpha_2} \frac{\partial f}{\partial \alpha_1} \right] + \frac{2}{Nre} D^4 \psi \qquad 4.3.6$$

with $f = \frac{D^2\psi}{h_3^2}$

$\zeta = -\frac{1}{h_3} D^2\psi$

where ζ = nonzero component of vorticity

$$D^2 = \frac{h_3}{h_1h_2}\left[\frac{\partial}{\partial\alpha_1}\left[\frac{h_2}{h_1h_3}\frac{\partial}{\partial\alpha_1}\right] + \frac{\partial}{\partial\alpha_2}\left[\frac{h_1}{h_2h_3}\frac{\partial}{\partial\alpha_2}\right]\right]$$

Equation 4.3.6 is a convenient scalar relationship that can be used to solve for incompressible, planar or axisymetric flow of a Newtonian fluid over an arbitrarily shaped body.

4.4 Heat and Mass Transfer Equations

It has been already established that a particle trajectory can be influenced by the presence of a vapor or temperature gradient. A good estimate of these quantities is, therefore, required when determining whether or not an aerosol particle is captured. Again, conservation laws can be applied to produce the governing equations.

Applying the principle of conservation of total energy (kinetic and internal) to an ideal fluid and removing the kinetic energy term with the dot product of velocity $(\bar{u})$ with Equation 4.3.1, the following expression is obtained:

$$\rho_f\frac{De}{Dt} = -p\cdot\bar{\nabla}\cdot\bar{U} + \phi + \bar{\nabla}\cdot(k_f\bar{\nabla}T) \qquad 4.4.1$$

where e = internal energy
ϕ = viscous dissipation (usually neglected)

k_f = thermal conductivity
T = fluid temperature

The assumption of constant density along with constant thermal conductivity allows Equation 4.4.1 to be written as [4]:

$$\rho_f C_p \frac{D\,T}{Dt} = k_f \nabla^2 T \qquad 4.4.2$$

where C_p = the fluid heat capacity

Finally, after applying the dimensionless transformation of Section 4.2 and introducing the stream function, the steady state energy equation takes the form:

$$\frac{\partial\,\psi}{\partial\alpha_2}\frac{\partial\,\eta}{\partial\alpha_1} - \frac{\partial\,\psi}{\partial\alpha_1}\frac{\partial\,\eta}{\partial\alpha_2} = \qquad 4.4.3$$

$$\frac{2}{Npe_h}\left[\frac{\partial}{\partial\alpha_1}\left(\frac{h_2h_3}{h_1}\frac{\partial\,\eta}{\partial\alpha_1}\right] + \frac{\partial}{\partial\alpha_2}\left(\frac{h_1h_3}{h_2}\frac{\partial\,\eta}{\partial\alpha_2}\right]\right]$$

where $\eta = \dfrac{T - T_\infty}{T_c - T_\infty}$

T_∞ = undisturbed fluid temperature
T_c = collector temperature
Npe_h = Peclet number for heat transfer
= $Nre \cdot Npr$
Npr = Prandtl number
= $\dfrac{C_p \mu_f}{k_f}$

An equation for the convective diffusion of a component of a gas mixture can be derived by applying a simple mass balance and using

Ficks law of diffusion. Assuming that the diffusing component is dilute and that it has a constant diffusivity the following equation is obtained [4]:

$$\frac{D\,C_1}{Dt} = \mathcal{D}_{12}\,\nabla^2 C_1 \qquad 4.4.4$$

where C_1 = the molar concentration of component 1
$\mathcal{D}_{12}$ = the molar diffusivity of component 1

Equations 4.4.2 and 4.4.4 are very similar and have identical dimensionless forms with the following definitions:

$\eta = \dfrac{C_1 - C_{1\infty}}{C_c - C_{1\infty}}$
$C_{1\infty}$ = concentration of component 1 in the undisturbed fluid
C_{1c} = concentration of component 1 on the collector
Npe_m = Peclet number for mass transfer
= $Nre \cdot Nsc$
Nsc = Schmidt number
= $\dfrac{\mu_f}{\rho_f \mathcal{D}_{12}}$

Having varying concentrations of a component in a gas mixture would affect the fluid flow through the density and viscosity. However, as pointed out by Woo [51], the assumption of constant physical properties has been shown to introduce very little error for water drops evaporating in air.

5. COORDINATE GENERATION

The solution of the coupled set of equations discussed in Chapter 4 is a formidable task. The Navier-Stokes equations, being nonlinear, have so far proven to be insoluble for the problem of planar or axisymmetric flow around spheroids and no closed solution exists for the entire range of all parameters [14,34]. Since the remaining transport equations, as well as the particle dynamic equations, are coupled to the Navier-Stokes equations, analytical solutions are scarce. Consequently, numerical algorithms must be implemented to solve these equations.

Presently, there are a number of different techniques for solving both linear and nonlinear partial differential equations. The most popular are finite difference and finite element approaches. The application of finite elements has been vary prevalent because the method is not restricted by coordinate geometry. As a result it is a viable candidate for determining the flow characteristics around deformed water droplets. Finite difference techniques are relatively simple to apply but are generally somewhat dependent on coordinate geometry. It is usual to choose a fixed coordinate system in which to work. Recent developments in the application of numerical grid generation have enabled analysts to produce competitive finite difference codes for problems in arbitrary domains.

The concept of grid generation for solving problems is quite simple. A set of boundary conforming coordinates is established and used much like polar or oblate speroidal coordinates are used to solve a problem. The flexibility lies in the fact that the coordinates (or grids) are generated numerically allowing the same numerical code to solve problems for many arbitrary regions. Thompson et. al. [49] have compiled extensive literature on the subject, with general-izations for nonorthogonal grids.

In this work, the method of orthogonal grid generation has been chosen for use in the model. This choice is a direct result of several distinct advantages realized since:

- extensive literature now exists on the topic of grid generation with its application to fluid dynamic problems
- the basic schemes are relatively simple to apply because finite diference techniques are well established
- there is a great deal known about the convergence and stability of finite difference algorithms
- there has been considerable work done in the area of optimal behavior of iterative schemes which is important to problems which are nonlinear [8]
- the condition of orthogonality greatly reduces the complexity of equations and derivative boundary conditions [33,40].

In Chapter 4, a brief discussion was given on general orthogonal coordinates and their implementation in the governing equation through the scale factors. In this chapter the emphasis will be focussed on the development of a scheme that can be used to obtain the required orthogonal transformation. Discussions will be limited to the generation of two dimensional orthogonal grids since the problem of interest can be viewed in two dimensions.

Restricted to a planar grid, the general transformation metric tensor is given by:

$$\begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}$$

where $g_{ij} = \frac{\partial \bar{r}}{\partial \alpha_i} \cdot \frac{\partial \bar{r}}{\partial \alpha_j}$

Orthogonality requires

$$g_{12} = g_{21} = 0$$

Therefore:

$$\frac{\partial x}{\partial \alpha_1} \cdot \frac{\partial x}{\partial \alpha_2} + \frac{\partial y}{\partial \alpha_1} \cdot \frac{\partial y}{\partial \alpha_2} = 0 \qquad 5.1$$

Equation 5.1 is a general statement of orthogonality. There can exist many different schemes to force Equation 5.1 to be true. Authors, such as Haussling and Coleman [19]; Pope [40]; Hung and Brown [20]; and Ryskin and Leal [45], have proposed algorithms for generating boundary conforming orthogonal coordinate systems. The method used in this work is based on the algorithm first proposed by Mobely and Stewart [33].

The fundamental basis of the algorithm is the existence of a conformal mapping between the physical region of interest and a rectangular domain as shown in Figure 5.1. For the mapping to be conformal the transformation must satisfy the Cauchy-Riemann equations:

$$\frac{\partial x}{\partial \alpha_1'} = \frac{\partial y}{\partial \alpha_2'} \qquad \text{(a)}$$

$$\frac{\partial x}{\partial \alpha_2'} = -\frac{\partial y}{\partial \alpha_1'} \qquad \text{(b)} \qquad 5.2$$

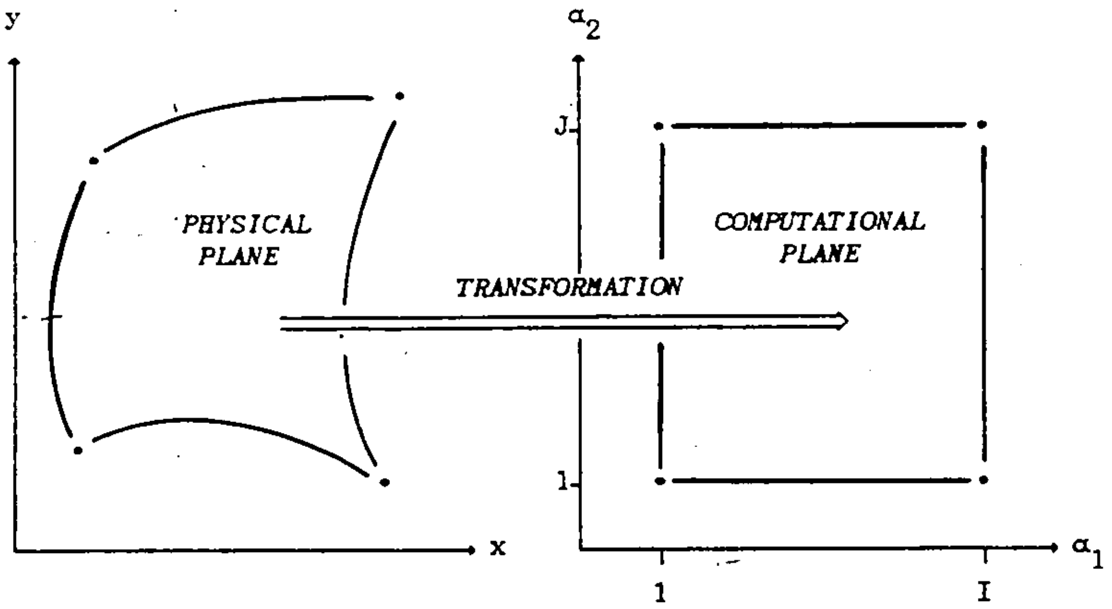

FIGURE 5.1: Transformed Computational Domain

Since Equations 5.2 satisfy Equation 5.1, the transformation is orthogonal.

Globally, when a conformal mapping is made, straight lines in the physical plane are mapped into curves. As a result distances may be distorted. In infinitesimal regions, angles are preserved; parallel lines are still parallel; and squares will be mapped into squares.

This fact is easily seen when Equations 5.2 are substituted into Equations 4.1.7. The two scale factors (h_1 and h_2) are equal which means that equal increments in the α_1 and α_2 directions correspond to two equal displacements in the physical plane [23].

In applications where there will be large deviations in the flow field, it would be desirable to 'pack' the grid lines in the regions where these deviations will occur to achieve better resolution. A direct result of the scale factors being equal is the loss of control over the grid spacing. An algorithm could be devised on the basis of Equations 5.2 but this would be severly limiting.

To make the transformation easier to handle numerically, it is useful to assign a size to the computational rectangle. Assuming that there are (I) points in the α_1 direction and (J) points in the α_2 direction, a convenient size would be:

$$1 \le \alpha_1 \le I$$
$$1 \le \alpha_2 \le J$$

so that

$$\Delta\alpha_1 = \Delta\alpha_2 = 1$$

As pointed out by Thompson [49], if the computational rectangle is completely specified (ie. all four sides are fixed) the problem is overdetermined. The Riemann Mapping Theorem states that a conformal mapping is uniquely determined by specifying any three real parameters [49]. This overspecification can be overcome by multiplying either the α_1 or the α_2 axis by an adjustable parameter (s) to allow the computational plane to be fixed while the conformal plane is open ended. As a result Equations 5.2 become:

$$s \cdot \frac{\partial x}{\partial \alpha_1^\circ} = \frac{\partial y}{\partial \alpha_2^\circ} \qquad \text{a)}$$

$$\frac{\partial x}{\partial \alpha_2^\circ} = -s \cdot \frac{\partial y}{\partial \alpha_1^\circ} \qquad \text{b)} \qquad 5.3$$

where

$$\alpha_1^\circ = s \cdot \alpha_1'$$
$$\alpha_2^\circ = \alpha_1'$$

It should be noted that Equations 5.3 no longer represent a conformal mapping but it is still an orthogonal mapping since it satisfies Equation 5.1.

By differentiating Equations 5.3 with respect to α_1° and α_2° it is easily shown that x and y must satisfy:

$$s^2 \cdot \frac{\partial^2 x}{\partial \alpha_1^{\circ 2}} + \frac{\partial^2 x}{\partial \alpha_2^{\circ 2}} = 0 \qquad \text{(a)}$$

$$s^2 \cdot \frac{\partial^2 y}{\partial \alpha_1^{\circ 2}} + \frac{\partial^2 y}{\partial \alpha_2^{\circ 2}} = 0 \qquad \text{(b)} \qquad 5.4$$

in the computational plane.

At this point, the scheme is still limited by a lack of control over the grid spacing. The method used by Mobley and Stewart [33] involved the introduction of packing functions for α_1° and α_2° according to:

$$\alpha_1^o = f(\alpha_1) \qquad\qquad 5.5$$
$$\alpha_2^o = g(\alpha_2)$$

where f and g are monotonically increasing functions of the new computational variables α_1 and α_2. The transformation in Equations 5.5 can be viewed as a simple coordinate stretch of the intermediate varibles α_1^o and α_2^o and monotonicity ensures that the mapping is still one to one.

The new forms of Equations 5.3 are:

$$s\cdot\frac{\partial x}{\partial \alpha_1} = \frac{f'}{g'}\cdot\frac{\partial y}{\partial \alpha_2} \qquad \text{(a)}$$
$$5.6$$
$$\frac{\partial x}{\partial \alpha_2} = -s\cdot\frac{g'}{f'}\cdot\frac{\partial y}{\partial \alpha_1} \qquad \text{(b)}$$

It can be shown that x and y must satisfy:

$$\mathcal{L}(x) = 0 \qquad \text{a)}$$
$$5.7$$
$$\mathcal{L}(y) = 0 \qquad \text{b)}$$

where

$$\mathcal{L} = s^2\cdot\frac{\partial^2}{\partial \alpha_1^2} - s^2\cdot\frac{f''}{f'}\cdot\frac{\partial}{\partial \alpha_1} + \left[\frac{f'}{g'}\right]^2\cdot\frac{\partial^2}{\partial \alpha_2^2} - \frac{g''}{g'}\left[\frac{f'}{g'}\right]^2\cdot\frac{\partial}{\partial \alpha_2}$$

Clearly, Equations 5.6 satisfy Equation 5.1 to produce an orthogonal mapping with some control gained over the grid spacing through the stetching functions f and g.

The boundary conditions for Equations 5.7 are obtained from Equations 5.6. These latter equations were developed to achieve an orthogonal mapping of an arbitrary domain into a rectangle. By implication, these equations must hold true on the boundaries [33]. According to Equations 5.6, it can be shown that, if y is specified on an α_1 boundary, $\frac{\partial y}{\partial \alpha_1}$ is known which means that $\frac{\partial x}{\partial \alpha_2}$ is known. Therefore, a Neuman (derivative) boundary condition must be specified for x. The converse is true if x is specified. This condition illustrates the fact that Equations 5.7 are not explicitly coupled in the interior but that they are coupled by the boundary conditions. Figure 5.2 provides an example of a typical problem statement in both the physical and computational planes. Only the corner points of the transformation can by fully defined since both x and y cannot be expressed on a given computational boundary.

One final condition must be used to define the constant 's'. This constant can be determined from either relationship of Equations 5.6 by integrating along a line of constant α_1 or α_2. An example of this approach is illustrated by:

$$s \cdot [x(i,\alpha_2) - x(1,\alpha_2)] = \int_{\alpha_1=1}^{\alpha_1=I} \frac{f'(\alpha_1)}{g'(\alpha_2)} \cdot \frac{\partial y}{\partial \alpha_2} \, d\alpha_1 \qquad 5.8$$

In a numerical procedure, it is possible to choose any α_2 grid line to perform the integration.

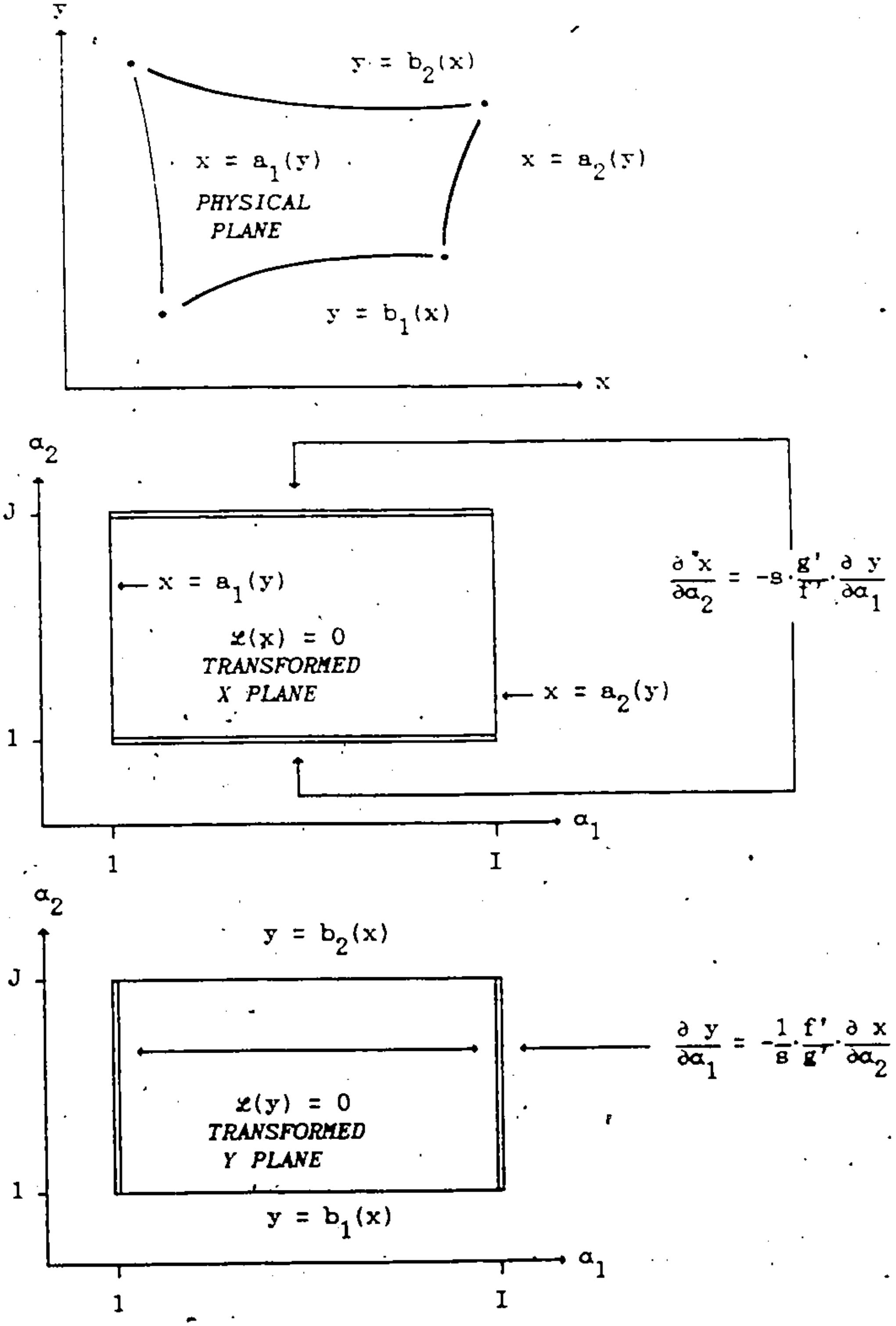

FIGURE 5.2: Application of Boundary Conditions

As mentioned earlier, the packing functions 'f' and 'g' can be used to obtain any desired density of grid lines in a region. Very fine grids are required to model variables that change quickly. Mobley and Stewart [33] have proposed four general packing functions to accomplish various grid line density distributions. These functions are:

$$z_1(\alpha_i) = \alpha_i \tag{5.9}$$

$$z_2(\alpha_i) = \frac{N+1}{2}+\frac{N-1}{2\cdot\sinh^{-1}(a)}\cdot\sinh^{-1}\left[\frac{2\cdot a}{N-1}\left[\alpha_i-\frac{N+1}{2}\right]\right] \tag{5.10}$$

$$z_3(\alpha_i) = \frac{N+1}{2}+\frac{N-1}{2\cdot\sin^{-1}(a)}\cdot\sin^{-1}\left[\frac{2\cdot a}{N-1}\left[\alpha_i-\frac{N+1}{2}\right]\right] \tag{5,11}$$

$$z_4(\alpha_i) = \exp\left[\frac{(\alpha_i^a-1)}{(N^a-1)}\cdot\ln(N)\right] \tag{5.12}$$

where

N = the maximum α_i dimension

Equations 5.9 through 5.12 perform the following tasks when applied to the α_i coordinate lines:

- z_1 represents no packing.
- z_2 packs the grid lines close to α_i =1 and α_i = N leaving the centre sparce. The magnitude of 'a' determines the degree of packing.
- z_3 packs the grid lines towards the centre leaving the edges sparce. The magnitude of 'a' determines the degree of packing.

- z_4 packs the grids lines either towards $\alpha_i = 1$ or $\alpha_i = N$ depending on whether 'a' is positve or negative. Again, the magnitude of 'a' determines the degree of packing.

The affect of applying these packing functions is demonstrated in Figures 5.3a and 5.3b for the trivial case of transforming a unit square into a rectangle.

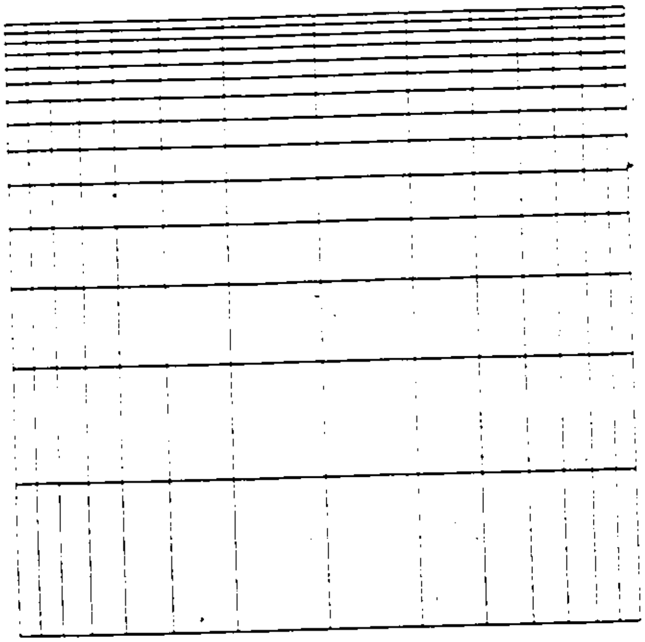

FIGURE 5.3b: Grid Packing in a Unit Square
$(f(\alpha_1) = z_2(\alpha_1)$; $a = 5.00)$
$(g(\alpha_2) = z_4(\alpha_2)$; $a = -1.0)$

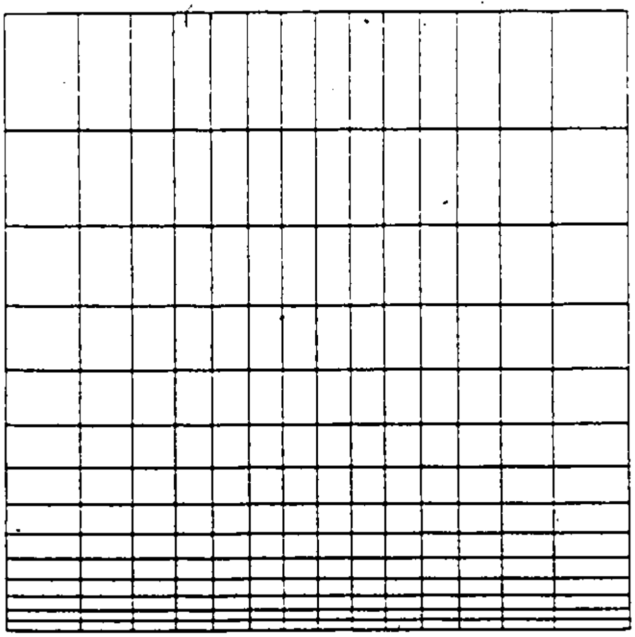

FIGURE 5.3a: Grid Packing in a Unit Square
$(f(\alpha_1) = z_3(\alpha_1)$; $a = 0.95)$
$(g(\alpha_2) = z_4(\alpha_2)$; $a = 1.00$

6. COMPUTATIONAL PROCEDURE

The Navier-Stokes, convective diffusion, and coordinate generation equations in Chapters 4 and 5 are resolved to a set of nonlinear algebraic equations by the application of the method of finite differences. These algebraic equations are solved using a new iterative procedure proposed by Ehrlich [8]. This scheme is ideal for a general model because the dependence on parameter estimation is reduced when compared to standard relaxation schemes. Once the various flow variables are determined, the particle dynamic equations of Chapter 4 are solved by the fifth-order, variable step-size, Runge-Kutta-Fehlberg algorithm to determine particle trajectories.

The governing equations are coupled to the particle dynamic equations but the entire equation set need not be solved simultaneously. Generally, the Navier-Stokes equation, Equation 4.3.1, is coupled to the energy equation, Equation 4.4.1, and the diffusion equation, Equation 4.4.4, through the density and viscosity. The assumption of constant physical properties uncouples these equations, thus allowing the flow field to be visualized before determining the temperature and vapor distributions. The equations are connected such that the following sequence of calculations can be used:

- Equations 5.1.7 a and b are solved to generate the correspondence between the computational and physical planes.

- the incompressible Navier-Stokes equation, Equation 4.3.6, is solved to produce steady state velocity profiles (Note: Forcing the vorticity to be zero at high Reynolds numbers resulted in a potential flow solution).
- the convective diffusion equation (Equation 4.4.3) is solved to provide temperature and vapor distributions.
- finally the particle dynamic equation (Equation 4.2.4) is solved.

6.1 Finite Difference Approximations

The partial differential equations developed earlier are applied to every interior grid point $\alpha_1\bar{e}_1+\alpha_2\bar{e}_2$. All of the spatial derivatives are evaluated by discrete numerical approximations. The mth derivative of $f(x_i)$ is approximated in the form:

$$\frac{d^m f(x_i)}{dx^m} \simeq \sum_{j=k}^{j=1} \beta_j f(x_{i+j}) \qquad 6.1.1$$

where the β_j's are determined by means of Taylor series expansions for the neighboring points, $f(x_{i+j})$, about the point $f(x_i)$. In this work, the interior derivatives are approximated by second-order central-difference expressions which are given as:

$$\frac{\partial \phi}{\partial \alpha_1} \simeq \Delta_1^0\phi = \frac{\phi_{i+1,j}-\phi_{i-1,j}}{2\Delta\alpha_1} = \frac{1}{2}(\phi_{i+1,j}-\phi_{i-1,j}) \qquad 6.1.2$$

$$\frac{\partial \phi}{\partial \alpha_2} \simeq \Delta_2^0\phi = \frac{\phi_{i,j+1}-\phi_{i,j-1}}{2\Delta\alpha_2} = \frac{1}{2}(\phi_{i,j+1}-\phi_{i,j-1}) \qquad 6.1.3$$

$$\frac{\partial^2 \phi}{\partial \alpha_1^2} \simeq \Delta_{11}^{\circ}\phi = \frac{\phi_{i+1,j}-2\phi_{i,j}+\phi_{i-1,j}}{\Delta\alpha_1^2}$$

$$= (\phi_{i+1,j}-2\phi_{i,j}+\phi_{i-1,j}) \qquad 6.1.4$$

$$\frac{\partial^2 \phi}{\partial \alpha_2^2} \simeq \Delta_{22}^{\circ}\phi = \frac{\phi_{i,j+1}-2\phi_{i,j}+\phi_{i,j-1}}{\Delta\alpha_2^2}$$

$$= (\phi_{i,j+1}-2\phi_{i,j}+\phi_{i,j-1}) \qquad 6.1.5$$

It should be noted that $\Delta\alpha_1=\Delta\alpha_2=1$ because of the boundary conditions employed in the construction of the grid.

When approximating derivative boundary conditions, the second-order central-difference operator cannot always be applied. Consequently, it should be replaced with the appropriate (forward or backward) second-order one-sided difference approximation. These difference operators are given as:

FORWARD

$$\frac{\partial \phi}{\partial \alpha_1} \doteq \Delta_1^{++}\phi = \frac{-3\phi_{i,j}+4\phi_{i+1,j}-\phi_{i+2,j}}{2\Delta\alpha_1}$$

$$= \frac{1}{2}(-3\phi_{i,j}+4\phi_{i+1,j}-\phi_{i+2,j}) \qquad 6.1.6$$

$$\frac{\partial \phi}{\partial \alpha_2} \simeq \Delta_2^{++}\phi = \frac{-3\phi_{i,j}+4\phi_{i,j+1}-\phi_{i,j+2}}{2\Delta\alpha_2}$$

$$= \frac{1}{2}(-3\phi_{i,j}+4\phi_{i,j+1}-\phi_{i,j+2}) \qquad 6.1.7$$

BACKWARD

$$\frac{\partial \phi}{\partial \alpha_1} \simeq \Delta_1^{--}\phi = \frac{3\phi_{i,j}-4\phi_{i-1,j}+\phi_{i-2,j}}{2\Delta\alpha_1}$$

$$= \frac{1}{2}(3\phi_{i,j}-4\phi_{i-1,j}+\phi_{i-2,j}) \qquad 6.1.8$$

$$\frac{\partial \phi}{\partial \alpha_2} \simeq \Delta_2^{--}\phi = \frac{3\phi_{i,j}-4\phi_{i,j-1}+\phi_{i,j-2}}{2\Delta\alpha_2}$$

$$= \frac{1}{2}(3\phi_{i,j}-4\phi_{i,j-1}+\phi_{i,j-2}) \qquad 6.1.9$$

Finally, a scheme has been used to solve Equation 4.3.6 which will involve first-order one-sided differences. These operators are given as:

FORWARD

$$\frac{\partial \phi}{\partial \alpha_1} \simeq \Delta_1^{+}\phi = \frac{\phi_{i+1,j}-\phi_{i,j}}{\Delta\alpha_1} = \phi_{i+1,j}-\phi_{i,j} \qquad 6.1.10$$

$$\frac{\partial \phi}{\partial \alpha_2} \simeq \Delta_2^{+}\phi = \frac{\phi_{i,j+1}-\phi_{i,j}}{\Delta\alpha_2} = \phi_{i,j+1}-\phi_{i,j} \qquad 6.1.11$$

BACKWARD

$$\frac{\partial \phi}{\partial \alpha_1} \simeq \Delta_1^{-}\phi = \frac{\phi_{i,j}-\phi_{i-1,j}}{\Delta\alpha_1} = \phi_{i,j}-\phi_{i-1,j} \qquad 6.1.12$$

$$\frac{\partial \phi}{\partial \alpha_2} \simeq \Delta_2^{-}\phi = \frac{\phi_{i,j}-\phi_{i,j-1}}{\Delta\alpha_2} = \phi_{i,j}-\phi_{i,j-1} \qquad 6.1.13$$

6.1.1 Coordinate Generation Algorithm

The coordinate generation scheme proposed in Chapter 5 has been used to determine the grid scale factors, h_i, presented in Chapter 4. Once these quantities are known at each grid point, $\alpha_1\bar{e}_1+\alpha_2\bar{e}_2$, the rest of the modelling equations can be applied.

A question that arises is whether the analytical evaluation of these scale factors is more accurate than the numerical evaluation. Thompson et. al. [49] addressed this problem and proved that the numerically evaluated scale factors produced a more accurate numerical truncation than the analytical scale factors, assuming the same differencing was used for both the function derivative and the scale factor evalution. Consequently, the numerical evaluation of grid scale factors is preferable even when analytical expressions are known. This approach makes good sense since the analytical scale factors are continuous transformations and are intended for continuous grids, not discrete ones.

Additional advantages are gained by forcing the coordinates to be orthogonal. In Chapter 4, it was emphasized that the transformation metric tensor is greatly simplified by forcing the transformation to be orthogonal, thereby reducing the complexity of the transformed modelling equations. This simplification is not only a matter of convenience but it has some numerical significance as shown by Mobley et. al. [33].

Consider the evaluation of the normal derivative to a line of constant α_1 assuming that the coordinate transformation is not orthogonal. Then:

$$\frac{\partial f}{\partial n} = \hat{n}\cdot\nabla f = \frac{1}{\sqrt{g\cdot g_{22}}}\left[g_{22}\cdot\frac{\partial f}{\partial\alpha_1} - g_{21}\cdot\frac{\partial f}{\partial\alpha_2}\right] \qquad 6.1.14$$

where

$$\sqrt{g} = \frac{\partial x}{\partial\alpha_1}\cdot\frac{\partial y}{\partial\alpha_2} - \frac{\partial x}{\partial\alpha_2}\cdot\frac{\partial y}{\partial\alpha_1} .$$

If the transformation is orthogonal this expression reduces to:

$$\frac{\partial f}{\partial n} = \hat{n}\cdot\nabla f = \frac{\sqrt{g_{22}}}{\sqrt{g}}\cdot\frac{\partial f}{\partial\alpha_1} = \frac{1}{h_1}\cdot\frac{\partial f}{\partial\alpha_1} \qquad 6.1.15$$

The use of Equation 6.1.14 could produce numerical errors since it couples the derivatives in both the α_1 and the α_2 directions. If, as a result, differences are taken between two large numbers, significance would be lost because of the format in which real numbers are handled in digital computers [33].

The first step in solving Equations 5.7 numerically is to apply the central-difference operators to produce an algebraic approximation. The difference approximation to Equations 5.7 is:

$$\left[\frac{\delta}{s}\right]^2\left[1+\frac{G}{2}\right]\cdot\bar{r}_{i,j-1}+\left[1+\frac{F}{2}\right]\cdot\bar{r}_{i-1,j}-2\left[1+\left[\frac{\delta}{s}\right]^2\right]\cdot\bar{r}_{i,j}$$
$$+\left[1-\frac{F}{2}\right]\cdot\bar{r}_{i+1,j}+\left[1-\frac{G}{2}\right] \qquad 6.1.16$$

where

$$\bar{r} = x\bar{i} + y\bar{j}$$
$$F = \frac{f''}{f'}$$
$$G = \frac{g''}{g'}.$$
$$\delta = \frac{f'}{g'}.$$

The method used to solve Equation 6.1.16 is the one-step block successive over-relaxation algorithm with each grid point having its own acceleration parameter $\omega_{i,j}$ [35]. For this procedure, the (k+1)-th approximation to the vector $\bar{x}_j$ is obtained from th k-th approximation according to:

$$\underline{T}\cdot(\bar{x}_j^{k+1}-\bar{x}_j^{k}) = (\overline{\omega r})_j \qquad 6.1.17$$

where

$\underline{T}$ = tridiagonal matrix

$= B(1+\frac{F}{2},-2(1+[\delta/s]^2),1-\frac{F}{2})$

$\bar{x}_j$ = the vector of $x_{i,j}$ that corresponds to the grid line $\alpha_2 = j$

$\omega_{i,j}$ = grid point acceleration factor

$r_{i,j}$ = residual vector formed by bringing all terms in a difference equation to the right hand side.

The acceleration factor $\omega_{i,j}$ is determined by the method proposed by Erlich [8] as shown in Appendix C. Although having an acceleration factor for each grid point requires some extra work, it ultimately saves time since it provides a near optimal iteration procedure. In addition, the $\omega_{i,j}$'s usually don't need to be updated every iteration.

After the interior points, $x_{i,j}\bar{i}+y_{i,j}\bar{j}$, have been updated, the boundary condition, either relationship from Equations 5.6, is applied to the α_1 boundaries; that is α_1 equal 1 and I. This boundary condition is approximated numerically by using second-order central differencing in the α_1 direction and second-order one-sided differencing in the α_2 direction as follows:

For $\alpha_1 = 1$

$$\frac{1}{g'}\Delta_2^{++}x_{i,j} = -\frac{s}{f'}\Delta_1^{0}y_{i,j} \qquad 6.1.18$$

For $\alpha_1 = I$

$$\frac{1}{g'}\Delta_2^{--}x_{i,j} = -\frac{s}{f'}\Delta_1^{0}y_{i,j} \qquad 6.1.19$$

Similarly, for the α_2 boundary the following difference approximations are made:

For $\alpha_2 = 1$

$$\frac{1}{g'}\Delta_2^{0}x_{i,j} = -\frac{s}{f'}\Delta_1^{++}y_{i,j} \qquad 6.1.20$$

For $\alpha_2 = J$

$$\frac{1}{g'}\Delta_2^{0}x_{i,j} = -\frac{s}{f'}\Delta_1^{--}y_{i,j} \qquad 6.1.21$$

Through trial and error, it was determined that the boundary conditions produced instabilities in the numerical solution of Equation 6.1.16. Consequently, the boundary values were dampened using the formula:

$$x_{i,j}^{k+1} = \beta \cdot \hat{x}_{i,j}^{k+1} + (1-\beta) \cdot x_{i,j}^{k} \qquad 6.1.22$$

where

β = a dampening factor taken to be 0.5
$x_{i,j}^{k+1}$ = the new updated x boundary value
$\hat{x}_{i,j}^{k+1}$ = the unaccelerated Gauss-Siedel iterate

A similar formula was used for the $y_{i,j}$ boundary values.

Finally, the parameter 's' was determined from Equation 5.8 using a second-order trapezoidal rule to perform the required integration. This equation takes the form:

$$s \cdot [x(I,j)-x(1,j)] \cdot g' = \frac{f'(1)\Delta_2^{o} y_{1,j} + f'(I)\Delta_2^{o} y_{I,j}}{2} + \sum_{i=2}^{i=I-1} f'(i)\Delta_2^{o} y_{i,j} \qquad 6.1.23$$

Equation 6.1.23 can be applied to any α_2 grid line to update 's' in an iterative procedure. In this work, Equation 6.1.23 was applied to all α_2 grid lines and the parameter 's' is updated with an overall grid averaged value. In addition, Equation 6.1.22 was applied to the 's' parameter with β equal to 0.5 in order to reduce numerical oscillations.

The problem of interest in this investigation required the generation of a grid about a droplet which is deformed but has, essentially, an approximate spherical shape. It would be convenient if the problem could be defined in polar coordinates since the droplet shape will deviate from a circular cross-section. To do this, the Cauchy-Riemann equations must be expressed in polar coordinates according to:

$$\frac{\partial r}{\partial \alpha_1} = r \cdot \frac{\partial \theta}{\partial \alpha_2}$$

$$\frac{\partial r}{\partial \alpha_2} = -r \cdot \frac{\partial \theta}{\partial \alpha_1}$$

or

$$\frac{\partial \theta}{\partial \alpha_1} = \frac{\partial}{\partial \alpha_2}(\ln[1/r]) \qquad \text{(a)}$$

$$6.1.24$$

$$\frac{\partial \theta}{\partial \alpha_2} = -\cdot\frac{\partial}{\partial \alpha_1}(\ln[1/r]) \qquad \text{(b)}$$

Equations 6.1.24 are of the same form as Equations 5.2. For this reason, all of the work done for problems stated in Cartesian coordinates can be applied to problems stated in polar coordinates.

The grid scale factors, h_i, must be evaluated to complete this portion of the model. When the problem is planar and restricted to the x-y Cartesian plane the evaluation is [illegible] simple. The transformation equations for such a situat[illegible] be written as:

$$\begin{aligned} x &= x(\alpha_1,\alpha_2) \\ y &= y(\alpha_1,\alpha_2) \qquad 6.1.25 \\ z &= \alpha_3 \end{aligned}$$

Clearly, h_3 is given by:

$$h_3 = \left|\frac{\partial \bar{r}}{\partial \alpha_3}\right| = 1 \qquad 6.1.26$$

The grid scale factors, h_1 and h_2, are numerically determined using second-order central differencing in the interior. The following formulae can be use for their evaluation:

$$h_k = \left|\frac{\partial \bar{r}}{\partial \alpha_k}\right| = \left[\left[\frac{\partial x}{\partial \alpha_k}\right]^2 + \left[\frac{\partial y}{\partial \alpha_k}\right]^2\right]^{\frac{1}{2}} \qquad k=1,2$$

$$= [(\Delta_k^o x_{i,j})^2 + (\Delta_k^o y_{i,j})^2]^{\frac{1}{2}} \qquad 6.1.27$$

On the α_1 boundary, second-order one-sided differencing was used for h_1 as follows:

For $\alpha_1 = 1$

$$h_1 = [(\Delta_1^{++} x_{i,j})^2 + (\Delta_1^{++} y_{i,j})^2]^{\frac{1}{2}} \qquad 6.1.28$$

For $\alpha_1 = I$

$$h_1 = [(\Delta_1^{--} x_{i,j})^2 + (\Delta_1^{--} y_{i,j})^2]^{\frac{1}{2}} \qquad 6.1.29$$

Similarly, second-order one-sided differencing was used on the α_2 boundaries for determining h_2 as follows:

For $\alpha_2 = 1$

$$h_2 = [(\Delta_2^{++} x_{i,j})^2 + (\Delta_2^{++} y_{i,j})^2]^{\frac{1}{2}} \qquad 6.1.30$$

For $\alpha_2 = J$

$$h_2 = [(\Delta_2^{--} x_{i,j})^2 + (\Delta_2^{--} y_{i,j})^2]^{\frac{1}{2}} \qquad 6.1.31$$

For the case where the problem is axisymmetric, a grid can be generated in three dimensional space by rotating a two dimensional boundary conforming grid about the x-axis as shown in Figure 6.1. It must be recognized that this rotation is defined such that a right-handed coordinate system is generated; that is, $e_1 \cdot e_2 = e_3$. Clearly, from the geometry of the situation, the axisymmetric grid has the following coordinate transformation:

$$\begin{aligned} x &= x(\alpha_1,\alpha_2) \\ y &= y(\alpha_1,\alpha_2)\cdot\cos(\phi) \qquad 6.1.32 \\ z &= y(\alpha_1,\alpha_2)\cdot\sin(\phi) \end{aligned}$$

From Equations 6.1.32, the axisymmetric grid scale factors are:

$$\begin{aligned} h_1 &= \left[\left[\frac{\partial x}{\partial \alpha_1}\right]^2 + \left[\frac{\partial y}{\partial \alpha_1}\right]^2\right]^{\frac{1}{2}} \\ h_2 &= \left[\left[\frac{\partial x}{\partial \alpha_2}\right]^2 + \left[\frac{\partial y}{\partial \alpha_2}\right]^2\right]^{\frac{1}{2}} \qquad 6.1.33 \\ h_3 &= y \end{aligned}$$

Equations 6.1.33 show that h_1 and h_2 can be determined using Equations 6.1.27 to 6.1.31 while h_3 is nothing more than the y coordinate generated using Equation 6.1.16.

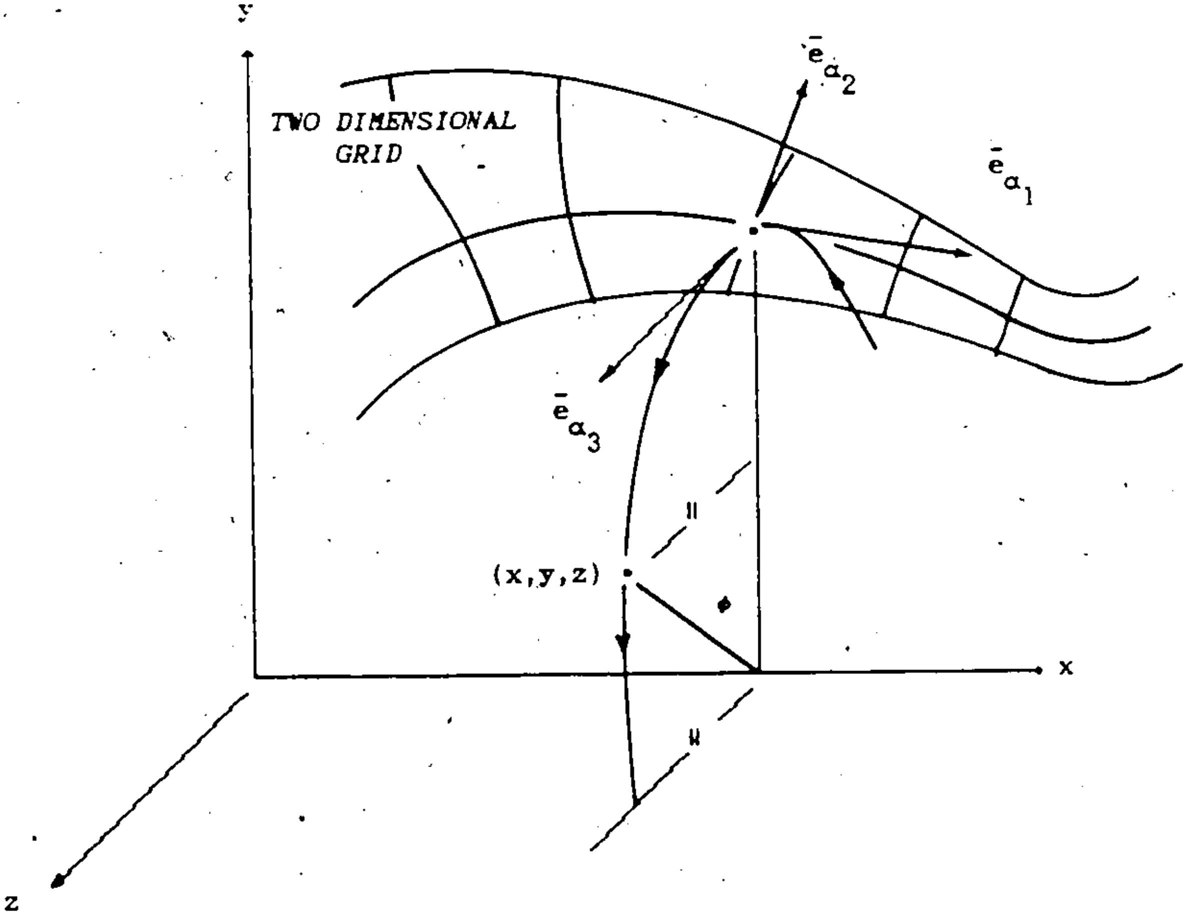

FIGURE 6.1: Rotated Axisymmetric Grid

6.1.2 Navier-Stokes Algorithm

The Navier-Stokes equation, Equation 4.3.6, can be written as two simultaneous second-order partial differential equations using the stream function, ψ, and vorticity, ζ, as follows:

$$\begin{aligned} &\frac{\partial}{\partial t}(h_3^2 f) = \left[L(\psi) + \frac{2}{N_{re}}D^2(h_3^2)\right]f && \text{(a)} \\ &D^2\psi = h_3^2 f && \text{(b)} \end{aligned} \qquad 6{,}1.34$$

where

f = modified vorticity function

$= -\dfrac{\zeta}{h_3}$

$$L(\psi) = \frac{h_3}{h_1h_2}\left[\frac{\partial\psi}{\partial\alpha_1}\frac{\partial}{\partial\alpha_2} - \frac{\partial\psi}{\partial\alpha_2}\frac{\partial}{\partial\alpha_1}\right]$$

$$D^2 = \frac{h_3}{h_1h_2}\left[\frac{\partial}{\partial\alpha_1}\left[\frac{h_2}{h_1h_3}\frac{\partial}{\partial\alpha_1}\right] + \frac{\partial}{\partial\alpha_2}\left[\frac{h_1}{h_2h_3}\frac{\partial}{\partial\alpha_2}\right]\right]$$

In this work the steady state solution of Equation 4.3.6 is required. There are two basic methods by which this solution can be achieved.

The first approach is to solve Equation 6.1.34 as an unsteady state problem or as a pseudo-unsteady state problem by introducing a fictitious time derivative into Equation 6.1.34 b [38]. The exact solution of Equation 6.1.34 b, or its modified form, need not be known at any given time because only the steady state solution is of interest. Strictly speaking, the steady state solution occurs as

$t \longrightarrow \infty$

but from a practical point the steady state solution is reached in a finite time as the time derivatives approach a small value. The disadvantage of this scheme is related to the finite time often being too long to make the method practical [38].

The second method, (which has been adopted in this work), involves the solution of Equations 6.1.34 iteratively with the time derivative removed. These equations are approximated at each grid point according to:

$$\left[L(\psi_{i,j}) + \frac{2}{Nre} D^2(h_3^2)\right] f_{i,j} = 0 \qquad (a)$$

$$D^2\psi_{i,j} = h_3^2 f_{i,j} \qquad (b) \qquad 6.1.35$$

where the operators:

- $L(\psi_{i,j})$
- D^2

are evaluated numerically. Solutions for flows over a rigid sphere have been obtained by Jenson [21] and Hameleic and Hoffman [17] by replacing:

- $L(\psi_{i,j})$
- D^2

by second-order central difference operators:

- $L^{\circ}(\psi_{i,j})$
- $D^{\circ 2}$

The case of flow over an ellipsoid was considered by Epstien [10], while LeClair et. al. [28] solved for the flow inside and around fluid spheres. Use of second-order difference operators, although accurate, creates problems of stability and convergence at higher Reynolds numbers as a result of the centred operator $L^{\circ}(\psi_{i,j})$ which causes loss of diagonal dominance in the iteration matrix [38].

The resolution of the loss of diagonal dominance is through replacement of the centred operator, $L^{\circ}(\psi_{i,j})$, with a noncentred (upwinded) operator $L^{*}(\psi_{i,j})$ [38]. This operator has the form:

$$L^{*}(\psi_{i,j}) = \frac{h_3}{h_1 h_2}\left[(\Delta_1^{\circ}\psi_{i,j})\Delta_2^{*} - (\Delta_2^{\circ}\psi_{i,j})\Delta_1^{*}\right] \qquad 6.1.36$$

where

$$\Delta_k^{*} = \frac{(1-\epsilon_{i,j}^{k})}{2}\Delta_k^{+} + \frac{(1+\epsilon_{i,j}^{k})}{2}\Delta_k^{-} = \Delta_k^{\circ} - \frac{1}{2}\epsilon_{i,j}^{k}\Delta_{kk}^{\circ} \qquad k = 1,2$$

From Equation 6.1.36, it is evident that when:

- $\epsilon_{i,j}^{k} = 1$, Δ_k^{*} is first-order backward differenced
- $\epsilon_{i,j}^{k} = 0$, Δ_k^{*} is second-order central differenced
- $\epsilon_{i,j}^{k} = -1$, Δ_k^{*} is first-order forward differenced.

After applying the required difference operators, the following difference equations can be used to solve Equations 6.1.35:

$$a_j^-\psi_{i,j-1}+a_i^-\psi_{i-1,j}-a^o\psi_{i,j}+a_i^+\psi_{i+1,j}+a_j^+\psi_{i,j+1}$$

$$= h_1h_2h_3f_{i,j} \qquad 6.1.37$$

where

$$a_j^- = \frac{h_1}{h_2h_3} - \frac{1}{2}\cdot\frac{\partial}{\partial\alpha_2}\left[\frac{h_1}{h_2h_3}\right]$$

$$a_i^- = \frac{h_2}{h_1h_3} - \frac{1}{2}\cdot\frac{\partial}{\partial\alpha_1}\left[\frac{h_2}{h_1h_3}\right]$$

$$a^o = 2\left[\frac{h_2}{h_1h_3}+\frac{h_1}{h_2h_3}\right]$$

$$a_i^+ = \frac{h_2}{h_1h_3} + \frac{1}{2}\cdot\frac{\partial}{\partial\alpha_1}\left[\frac{h_2}{h_1h_3}\right]$$

$$a_j^+ = \frac{h_1}{h_2h_3} + \frac{1}{2}\cdot\frac{\partial}{\partial\alpha_2}\left[\frac{h_1}{h_2h_3}\right]$$

$$b_j^-f_{i,j-1}+b_i^-f_{i-1,j}-b^of_{i,j}+b_i^+f_{i+1,j}+b_j^+f_{i,j+1} = 0 \qquad 6.1.38$$

where

$$b_j^- = \left[\frac{h_1}{h_2h_3} - \frac{1}{2}\cdot\frac{\partial}{\partial\alpha_2}\left[\frac{h_1}{h_2h_3}\right]\right]h^2_{3_{i,j-1}} - \frac{Nre}{4}\Delta_1^o\psi_{i,j}(1+\epsilon^2_{i,j})$$

$$b_i^- = \left[\frac{h_2}{h_1h_3} - \frac{1}{2}\cdot\frac{\partial}{\partial\alpha_1}\left[\frac{h_2}{h_1h_3}\right]\right]h^2_{3_{i-1,j}} + \frac{Nre}{4}\Delta_2^o\psi_{i,j}(1+\epsilon^1_{i,j})$$

$$b^o = 2\left[\frac{h_2h_3}{h_1}+\frac{h_1h_3}{h_2}\right]+\frac{Nre}{2}(\epsilon^1_{i,j}\Delta_2^o\psi_{i,j}-\epsilon^2_{i,j}\Delta_1^o\psi_{i,j})$$

$$b_i^{\bullet} = \left[\frac{h_2}{h_1 h_3} + \frac{1}{2}\cdot\frac{\partial}{\partial\alpha_1}\left[\frac{h_2}{h_1 h_3}\right]\right] h^2_{3_{i+1,j}} - \frac{Nre}{4}\Delta_2^{\circ}\psi_{i,j}(1-\epsilon^1_{i,j})$$

$$b_j^{\bullet} = \left[\frac{h_1}{h_2 h_3} + \frac{1}{2}\cdot\frac{\partial}{\partial\alpha_2}\left[\frac{h_1}{h_2 h_3}\right]\right] h^2_{3_{i,j+1}} + \frac{Nre}{4}\Delta_1^{\circ}\psi_{i,j}(1-\epsilon^2_{i,j})$$

and

$$\frac{\partial}{\partial\alpha_i}\left[\frac{h_j}{h_i h_3}\right] = \frac{h_j}{h_i h_3}\left[\frac{1}{h_j}\cdot\frac{\partial}{\partial\alpha_i}(h_j) - \frac{1}{h_i}\cdot\frac{\partial}{\partial\alpha_i}(h_i) - \frac{1}{h_3}\cdot\frac{\partial}{\partial\alpha_i}(h_3)\right]$$
$$= \frac{h_j}{h_i h_3}\left[\frac{\Delta_i^{\circ}h_j}{h_j} - \frac{\Delta_i^{\circ}h_i}{h_i} - \frac{\Delta_i^{\circ}h_3}{h_3}\right]$$

At this point, the development of a criterion for determining $\epsilon^k_{i,j}$ is now possible on the basis of Equation 6.1.38. To ensure diagonal dominance, it is desirable to have the magnitude of the diagonal term, b^o, in the iteration matrix as large as possible. This magnitude is achived if the following criteria are used [38]:

$$\epsilon^1_{i,j} = \text{sign}(\Delta_2^{\circ}\psi_{i,j})$$
$$\epsilon^2_{i,j} = \text{sign}(-\Delta_1^{\circ}\psi_{i,j}) \qquad 6.1.39$$

It should be noted that, on the application of Equation 6.1.39, the solution of Equation 6.1.34 becomes first-order. However, second order accuracy can be achived by setting $\epsilon^k_{i,j}$ equal to zero. The purpose of introducing $\epsilon^k_{i,j}$ is to provide an algorithm that will converge very rapidly in the initial stages to provide, essentially, a reasonable starting point for the second-order iteration.

The method used to solve algebraic Equations 6.1.37 to 6.1.39 is successive substitution. These equations are rearranged to give the Gauss-Siedel iterate:

$$\hat{\phi}^{k+1}_{i,j} = F(\phi_{i,j-1},\phi_{i-1,j},\phi_{i+1,j},\phi_{i,j+1}) \qquad 6.1.40$$

The Gauss-Siedel iterate is accelerated using the formula:

$$\phi^{k+1}_{i,j} = \phi^{k}_{i,j}+\beta\cdot\omega_{i,j}(\hat{\phi}^{k+1}_{i,j}-\phi^{k}_{i,j}) \qquad 6.1.41$$

where

$\omega_{i,j}$ is the acceleration parameter
β is a dampening factor.

The values for ω were determined by the method described in section 6.1.1 and summarized in Appendix C. This method of determining the values for ω is intended for problems with Dirichlet boundary conditions. The effects of the Neuman boundary conditions in this problem are not taken into account when determining ω [8]. The introduction of β in Equation 6.1.41 was required to reduce the numerical oscillations that occured with over-estimated ω's. A value of 0.7 for β was found to produce convergence in all cases considered.

At very high Reynolds numbers the Navier-Stokes equations become increasingly unstable. This instability causes their solution to become impractical and uneconomical [38]. Consequently, at Reynolds numbers above 400, the potential flow solution was used to approximate the flow field. This solution was obtained by setting the vorticity, (or $f_{i,j}$), to zero in Equation 6.1.37. The resulting difference equation was solved using the block-successive overrelaxation algorithm defined by Equation 6.1.17.

When solving Equations 6.1.33, it is necessary to have boundary conditions that specify all values of ψ and ζ (or f) on the boundary completely enclosing the region of flow. This region is formed by:

- the surface of the collector
- the axis of symmetry
- the region were the influence of the collector is zero (which is at infinite distance from the collector).

This final boundary at infinity is numerically impractical. Therefore, it is approximated by assuming that there is a spherical wall enclosing the collector where, for all intents and purposes, the influence of the collector is zero and straight flow is imposed. The work of LeClaire et. al. [27], who provided extensive data on the effect of the location of this wall, was used in this investigation as a guideline.

Strictly speaking, when considering the flow of a fluid around a fluid body, the flow field around the collector is coupled to the circulation inside the fluid body. This coupling is through the boundary conditions that must be imposed at the fluid-fluid interface. These conditions are:

- the continuity of tangential velocity
- the continuity of tangential stress.

LeClair et. al. [28] considered solutions of the Navier-Stokes equations under these conditions. Fortunately, as pointed out by Wu [53], the affect of internal circulation in the air-water droplet system contributes negligibly to particle collection. Therefore, the collector can be modelled as a rigid body.

The boundary conditions for a rigid body for both ψ and ζ are:

For $\alpha_1 = 1$ (the axis of symmetry)

$$\psi = 0 \text{ and } \zeta = 0$$

For $\alpha_1 = I$ (the axis of symmetry)

$$\psi = 0 \text{ and } \zeta = 0$$

For $\alpha_2 = 1$ (the collector surface)

$$\psi = 0 \text{ and } \zeta = -\frac{1}{h_3} D^2 \psi$$

For $\alpha_2 = J$ (the outer wall at r_∞)

$$\psi = \frac{1}{2} y^2 \text{ and } \zeta = 0$$

The boundary condition for ζ at α_2 equal to 1 was determined by a second-order method given by Preyet [38]. It takes the following form in general orthogonal coordinates:

$$\zeta_{i,1} = \frac{\psi_{i,3} - 8\psi_{i,2}}{2\Delta\alpha_2^2 h_2^2 h_3} \qquad 6.1.42$$

Application of Equation 6.1.42 induced oscillations into the numerical solution which reduced the rate of convergence. This problem was alleviated by relaxing the boundary values of $\varsigma_{i,j}$ with Equation 6.1.23, by taking β to be 0.3. In addition, the convergence rate was increased by applying Equation 6.1.42 only every four iterations.

6.1.3 Convective Diffusion Algorithm

The convective diffusion equation, Equation 3.4.3, was solved using a method similar to that of Woo et. al. [52] to produce both temperature and vapor distributions around the collector. This equation was approximated by using second-order central difference operators to give the following difference equation:

$$a_j^- \eta_{i,j-1} + a_i^- \eta_{i-1,j} - a^o \eta_{i,j} + a_i^+ \eta_{i+1,j} + a_j^+ \eta_{i,j+1} = 0 \qquad 6.1.43$$

where

$$a_j^- = \frac{h_1 h_3}{h_2} - \frac{1}{2}\cdot\frac{\partial}{\partial\alpha_2}\left[\frac{h_1 h_3}{h_2}\right] - \frac{Npe}{4}\cdot\Delta_1^o \psi_{i,j}$$

$$a_i^- = \frac{h_2 h_3}{h_1} - \frac{1}{2}\cdot\frac{\partial}{\partial\alpha_1}\left[\frac{h_2 h_3}{h_1}\right] + \frac{Npe}{4}\cdot\Delta_2^o \psi_{i,j}$$

$$a^o = 2\left[\frac{h_2 h_3}{h_1} + \frac{h_1 h_3}{h_2}\right]$$

$$a_i^+ = \frac{h_2 h_3}{h_1} + \frac{1}{2}\cdot\frac{\partial}{\partial\alpha_1}\left[\frac{h_2 h_3}{h_1}\right] - \frac{Npe}{4}\cdot\Delta_2^o \psi_{i,j}$$

$$a_j^+ = \frac{h_1 h_3}{h_2} + \frac{1}{2}\cdot\frac{\partial}{\partial\alpha_2}\left[\frac{h_1 h_3}{h_2}\right] + \frac{Npe}{4}\cdot\Delta_1^0\psi_{i,j}$$

and

$$\frac{\partial}{\partial\alpha_i}\left[\frac{h_j}{h_i h_3}\right] = \frac{h_j}{h_i h_3}\left[\frac{1}{h_j}\cdot\frac{\partial}{\partial\alpha_i}(h_j)-\frac{1}{h_i}\cdot\frac{\partial}{\partial\alpha_i}(h_i)-\frac{1}{h_3}\cdot\frac{\partial}{\partial\alpha_i}(h_3)\right]$$

$$= \frac{h_j}{h_i h_3}\left[\frac{\Delta_i^0 h_j}{h_j} - \frac{\Delta_i^0 h_i}{h_i} - \frac{\Delta_i^0 h_3}{h_3}\right]$$

with

Npe = (Nre·Nsc) for mass transfer
= (Nre·Npr) for heat transfer

Equation 6.1.43 was solved in the same manner as Equations 6.1.37 and 6.1.38. The Guass-Siedel iterate was obtained according to Equation 6.1.40. This value was accelerated using Equation 6.1.41. Again, the introduction of β was required because the derivative boundary conditions are neglected when determining the $\omega_{i,j}$'s. An adequate value of β was found to be 0.7.

The boundary conditions used to solve Equations 3.4.3 are:

For $\alpha_1 = 1$ (the axis of symmetry)

no heat or mass transfer or $\frac{\partial\eta}{\partial\alpha_1}$

For $\alpha_1 = I$ (the axis of symmetry)

no heat or mass transfer or $\frac{\partial\eta}{\partial\alpha_1}$

For $\alpha_2 = 1$ (the collector surface)

$$\eta = 1$$

For $\alpha_2 = J$ (the the outer wall at r_∞)

$$\eta = 1$$

The first two derivative boundary conditions must be approximated by second-order one-sided operators as follows:

$$\Delta_1^{++}\eta_{1,j} = 0 \qquad 6.1.44$$

$$\Delta_2^{--}\eta_{I,j} = 0 \qquad 6.1.45$$

The boundary values obtained by Equations 6.1.45 were dampened using Equation 6.1.22 to increase the solution convergence rate. As in the application of Equation 6.1.42, a higher convergence rate was obtained by applying the above equations only every fourth iteration.

6.2 Particle Dynamic Equations

To determine the collection efficiency of a collector, Equations 4.2.3 must be integrated with Equations 4.2.6 and 4.2.7 to yield the limiting trajectory of a particle. These equations were integrated numerically using a variable step-size algorithm to accommodate the large change in the radius of curvature of the trajectory as the particle approaches the collector. The fifth-order variable step-size Runge-Kutta-Fehlberg method provided the means of integrating Equations 4.2.3 [6].

The Fehlberg method was chosen because it modifies the step-size automatically when required. Additionally, the number of function evaluations is kept to a minimum. At a given time level, the step-size is controlled by an estimate of the error which is determined from the difference between an approximation of order four and five. An algorithm using arbitrary fourth and fifth-order approximations would require ten function evaluations while the Fehlberg algorithm requires only six [6].

The particle equations of motion, Equations 4.2.3, require initial conditions for integration. These conditions are provided by assuming that the particle is initially suspended in the gas stream at the outer boundary wall (at $\alpha_2 = J$). A position along the outer wall must be determined such that the particle will just graze the collector. These boundary conditions take the following form in general orthogonal coordinates:

At $t = 0$

$\alpha_1 = \alpha_1^*$ (the value of α_1 for the limiting trajectory)

$\alpha_2 = J$ (the outer wall located at r_∞)

$v_1 = \frac{1}{h_1 h_3} \cdot \frac{\partial \psi}{\partial \alpha_2}$ (the initial particle velocity in the α_1 direction)

$v_2 = \frac{-1}{h_2 h_3} \cdot \frac{\partial \psi}{\partial \alpha_1}$ (the initial particle velocity in the α_2 direction)

The limiting trajectory was determined by first assuming minimum and maximum values of α_1^* and applying the method of bisection to reduce the interval to an acceptable level. With reference to Figure 6.2, the extreme values of α_1^* are adjusted according to the following criteria:

- if the particle strikes the collector at α_2 less than or equal to 1, the value of α_1^* minimum is set equal to α_1^*.
- if the particle does not hit the collector, the value of α_1^* maximum is set equal to α_1^*

In this work, it is assumed that the particle is collected when it strikes the collector surface.

Integration of Equation 4.2.3 requires some variables to be interpolated between grid nodes as shown in Figure 6.3. This interpolation is accomplished by using the following bilinear approximation:

$$\hat{\phi}_{\alpha_1,\alpha_2} = A_1+A_2\cdot\alpha_1+A_3\cdot\alpha_2+A_4\cdot\alpha_1\alpha_2 \qquad 6.2.1$$

where

$\hat{\phi}_{\alpha_1,\alpha_2}$ = a continuous representation of the discrete varible ϕ.

The four constants, A_i, are determined from the known values of ϕ at the four corner points shown in Figure 6.3. It should be recognized that the location of the grid cell is defined by the indices of the corner point (i,j). The boundary conditions employed in the grid

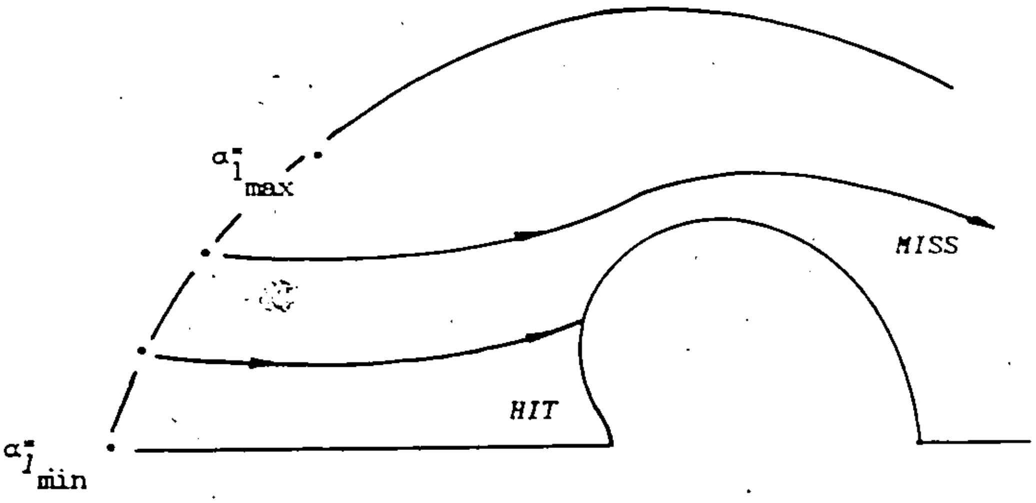

FIGURE 6.2: Trajectory Integration Procedure

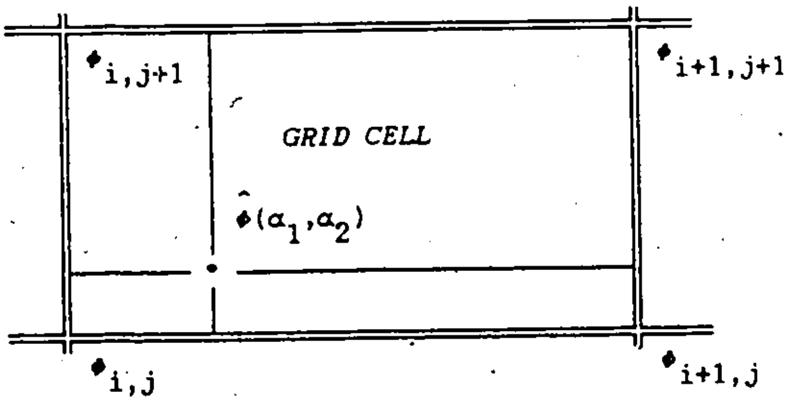

FIGURE 6.3: Grid Point Interpolation

construction algorithm defined in section 6.1.1 allow 'i' and 'j' to be determined by:

$$\begin{aligned} i &= \mathrm{int}(\ \alpha_1\ ) \\ j &= \mathrm{int}(\ \alpha_2\ ) \end{aligned} \qquad 6.2.2$$

Once the value of α_1 has been obtained, the corresponding value of the 'y' coordinate is required to determine the collection efficiency according to Equation 3.1.1. This value was determined by linear interpolation on the outer boundary.

7. RESULTS AND DISCUSSIONS

This investigation provides a comprehensive analysis of single drop collection of particles by:

- inertial impaction
- phoretic forces
- wake capture.

Emphasis has been focussed on a particle size range known as the Greenfield gap ($0.1\mu m \leq r_p \leq 2\mu m$) because no single collection mechanism is dominant in this region. Temperatures were chosen to model typical countercurrent scrubber operating conditions in which the gas is fully saturated.

Several separate and distinct fluid mechanics problems were solved before particle trajectory calculations were performed. The preliminary calculations involved:

- the generation of a grid to represent the computational domain
- the establishment of the fluid velocity profile around the collector
- the determination of the temperature and vapor distributions around the collector.

To ensure the integrity of the model as a whole, each step of the calculation procedure was validated before computing the final particle trajectories.

7.1 Orthogonal Grids

Orthogonal grids were generated for droplets of various sizes ranging between 50μm and 4000μm in radius. Droplets with radii, a_o, smaller than 620μm are essentially sphereical in shape. Significant deformation does not occur until the droplet radius is larger than 1800μm [41,42]. The grid generation procedure provides a convenient method of effectively distributing grids lines around these droplets for better resolution of the flow field and temperature/vapor distributions. The shapes of the deformed droplets were determined by the semi-empirical equation provided by Pruppacher and Pitter [42]. The method used to generate the grid system was described in Section 6.1.1. The algorithm was formulated into a FORTRAN 77 computer program which is provided in Appendix D.

The different packing configurations were obtained through the use of the packing fuctions $f(\alpha_1)$ and $g(\alpha_2)$ described in Chapter 5. Table 7.1 provides a summary of the various grid dimensions and packing functions used in this study.

Where possible, the α_1 grid lines where packed close to the axis of symmetry of the collector by letting:

$$f(\alpha_1) = z_2(\alpha_1) \quad ; \text{ with } a = 5.0.$$

The objective was to account for the large deviations in the fluid and particle trajectories in this region close to the collector. This

TABLE 7.1: Grid Dimensions and Packing Functions

Nre	I × J	r_∞	$f(\alpha_1)$	$g(\alpha_2)$
1.54	69 × 79	70.0	$z_2(\alpha_1)$; a=5.00	$z_1(\alpha_2)$
30	69 × 79	70.0	$z_2(\alpha_1)$; a=5.00	$z_1(\alpha_2)$
100	69 × 71	12.0	$z_2(\alpha_1)$; a=5.00	$z_1(\alpha_2)$
200	61 × 79	12.0	$z_1(\alpha_1)$	$z_1(\alpha_2)$
300	61 × 79	12.0	$z_1(\alpha_1)$	$z_4(\alpha_2)$; a=0.20
400	69 × 79	12.0	$z_1(\alpha_1)$	$z_1(\alpha_2)$; a=0.20
3600†	69 × 71	12.0	$z_2(\alpha_1)$; a=5.00	$z_1(\alpha_2)$
4900†	69 × 71	12.0	$z_2(\alpha_1)$; a=5.00	$z_1(\alpha_2)$

† potential flow conditions were solved for this Nre

packing configuration also allows more accurate modelling of the formation of standing eddies behind the collector. At higher Reynolds numbers, (Nre > 100), the standing eddy extended into the region of sparce α_1 grid lines. To alleviate this loss of resolution, no packing was used in the α_1 direction for Nre greater than 100 when the viscous fluid model was used.

Packing in the α_2 direction was generally not required on account of the polar coordinate conformal mapping equations, Equations 6.1.24. No packing in the α_2 direction would result in an approximately equal distribution of α_2 coordinate lines in the log-polar, ln(r), plane. Consequently, the α_2 grids lines will be naturally distributed in an exponential fashion with the highest density occurring near the collector. For Reynolds ~~numbers~~ of 300 and 400, additional packing close to the collector surface was used to model the steep gradients that occur as a result of the thin viscous boundary layer [26,51]. This was accomplished by letting:

$$g(\alpha_2) = z_4(\alpha_2) \quad ; \text{ with } a = 0.2.$$

The iterative nature of the procedure required an initial guess for the entire solution. This initial guess was based on the multi-directional transfinite interpolation technique provided by Thompson et. al. [49]. The iteration process was carried out until a convergence criterion between successive iterations was satisfied. This criterion is given by:

$$\max \left| \phi_{i,j}^{n+1} - \phi_{i,j}^{n} \right| \le 10^{-4} \qquad 7.1.1$$

In the literature, there appears to be no standard for evaluating the error in an orthogonal grid. A measure of the angle of intersection between nodes might be a viable estimate, but this can be misleading. The true angle of intersection of grids lines would be a function of the type of differencing used to determine the grid scale factors, h_i. That is to say, if a problem is solved with second order differencing and the grid scale factors (h_i) are determined by second order differencing, then adjacent grid points must be connected with second order curves. This approach would result in the measurement of the angle of intersection of two parabolas at a node.

A second measure of the error in a grid can be determined from the transformation metric tensor discussed in Chapter 5. It was emphasized that the diagonal terms in this tensor are zero if the transformation is orthogonal. The determinant of the metric tensor is given by:

$$g_{\text{orthogonal}} = g_{11} \cdot g_{22}$$

The determinant of the metric tensor for a general transformation is:

$$g_{\text{general}} = g_{11} \cdot g_{22} - (g_{12})^2$$

Since these quantities should be equal if the transformation is orthogonal, a simple measure of the error for an orthogonal grid could be:

$$R = \max\left[\left|1 - \frac{g_{\text{general}}}{g_{\text{orthogonal}}}\right|\right] \quad 7.1.6$$

where

R = the residual of the transformation

Figure 7.1 provides an example of an orthogonal grid for a deformed droplet with an equivalent volume radius, a_o, equal to 3000μm. The α_1 grid lines are packed close to the axis of symmetry to allow more accurate modelling of the fluid flow in the depression that exists in the front of the collector. Using Equation 7.1.6 as a measure of the deviation from orthogonality, the grid is very nearly orthogonal because:

- R = 0.0003.

This orthogonality is also confirmed by a simple visual inspection of Figure 7.1.

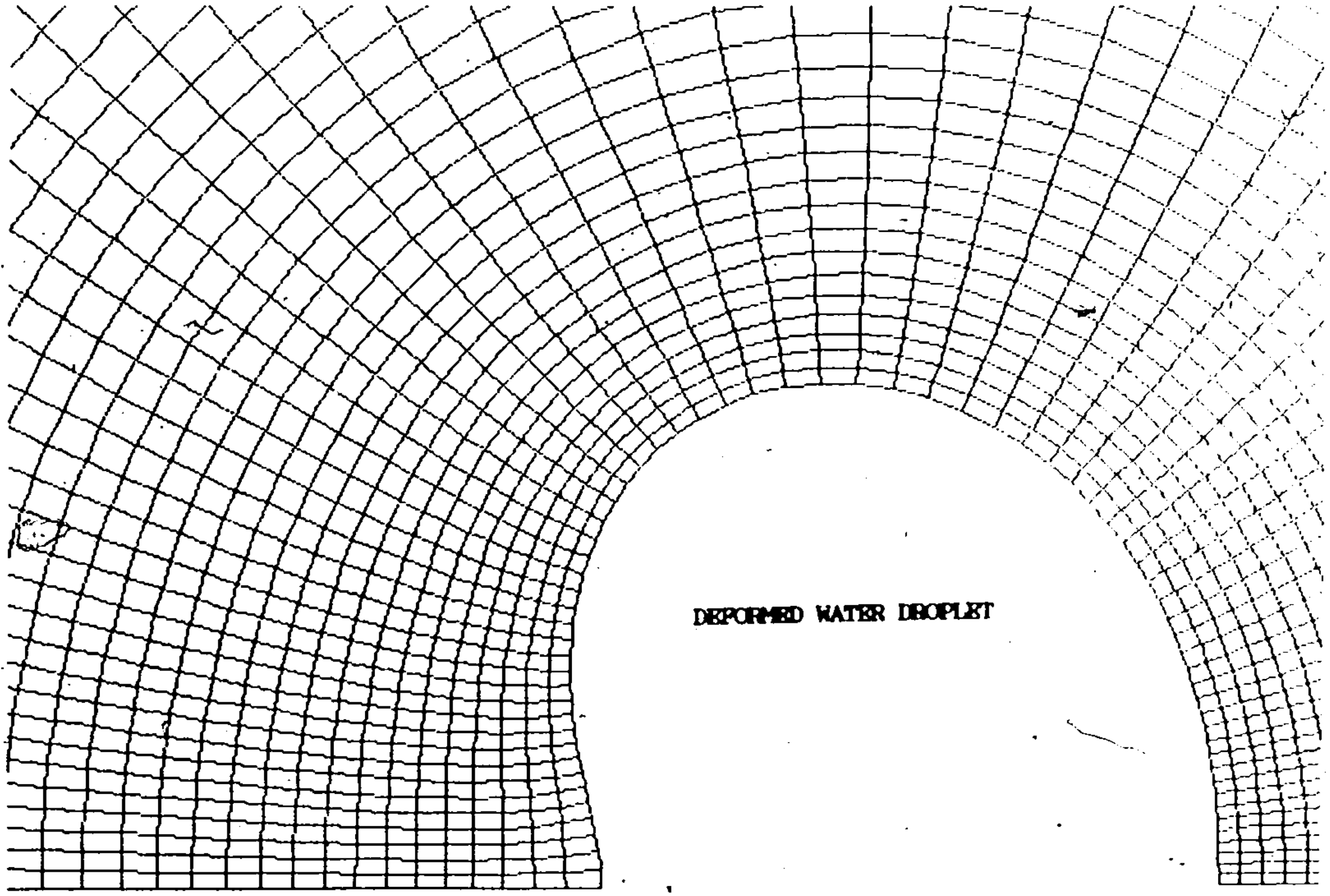

FIGURE 7.1: An Orthogonal Grid for a Deformed Droplet

7.2 Collector Flow Field

The governing equations for viscous flow around a rigid body are the Navier-Stokes equations which were discussed in Section 4.3. These equations were used to determine the flow fields around collectors for Reynolds numbers between 1.54 and 400. This range in Reynolds numbers corresponds to collector diameters varying between 100μm and 1240μm assuming that they are falling at their terminal velocities [42]. For Reynolds numbers above 400, potential flow conditions have been shown to provide good approximations to the flow fields when determining particle collection [3,9].

7.2.1 Viscous Flow Model

The method used to solve the equations governing viscous flow was the hybrid first-second order algorithm described in Section 6.1.2. This algorithm utilizes the upwinded hybrid method proposed by Peyret [38] to produce the difference equations which are solved by the variable relaxation factor method of Ehrlich [8]. It was formulated into a FORTRAN 77 computer program which is provided in Appendix E.

As described in Chapter 6, the first order algorithm was use to produce an initial guess for the second order algorithm. This process was accomplished by:

- assuming the potential flow solution for an initial guess

- running the code with $\epsilon^k_{i,j}$ determined from equations 6.1.39. (this converges very quickly)
- after a finite number of iterations (100 was used in all cases) $\epsilon^k_{i,j}$ was set to zero to obtain a second order accurate solution.

The final solution was obtained when the convergence criterion, Equation 7.1.1, was reached.

The algebraic difference equations representing the vorticity component of the Navier-Stokes equations, Equation 6.1.38, required severe underrelaxation. This condition was expected since it has been reported extensively in the literature [26,38,51]. The application of the varying relaxation factor parameter, $\omega_{i,j}$, produced much faster convergence when compared to the methods of block-successive relaxation and standard relaxation with a single relaxation parameter ($\omega_{optimum}$). The solution of Equations 6.1.37 and 6.1.38 would require that $\omega_{i,j}$ be updated every iteration because the coefficients of Equation 6.1.38 are not linear. This continual updating would consume a great deal of computer time. However, it was found that little of the optimal convergence behavior was lost when the $\omega_{i,j}$ values were updated every 10th iteration.

The ability of Ehrlich's method to adapt to the varing flow conditions is illustrated in Table 7.2. This table provides the maximum and minimum values of $\omega_{i,j}$ at convergence. Clearly demonstrated is the reduction of the minimum relaxation factor with increasing Reynolds number. This decrease occurs because the Reynolds

TABLE 7.2: Calculated Maximum and Minimum Relaxation Factors for Navier-Stokes Algorithm

Nre		ω_{max}	ω_{min}	LeClair‡
1.54	ψ	1.912	1.339	1.000
	ζ	1.856	0.811	——
30	ψ	1.912	1.339	1.000
	ζ	1.871	0.068	0.050
100†	ψ	1.913	1.378	1.000
	ζ	1.897	0.197	0.100
200	ψ	1.916	1.584	1.000
	ζ	1.916	0.111	0.100
300	ψ	1.920	1.601	1.000
	ζ	1.917	0.076	0.070
400	ψ	1.920	1.601	1.000
	ζ	1.915	0.057	0.020

† r_{∞} was reduced at this point

‡ determined by trial and error

number is the coefficient of the operator $L^{\circ}(\psi_{i,j})$. Increasing Reynolds numbers mean that the off diagonal terms produced by $L^{\circ}(\psi_{i,j})$ are becoming larger. As a result, the iteration matrix becomes poorly conditioned (loss of diagonal dominance). In addition, Table 7.2 illustrate good agreement between the trial and error values of $\omega_{optimum}$ found by LeClair [26] for similar grid sizes. Since the variable relaxation method requires no trial and error guessing by the user it is ideal as a general numerical model.

For the Reynolds number range considered for the viscous flow model, it is accepted that water droplets at their terminal velocities are still spherical [41,42]. Figure 7.2 illustrates the solution for the stream function, ψ, for a 433μm radius droplet at its terminal Reynolds number of 200. Clearly depicted in this illustration is the formation of a standing eddy or vortex behined the sphere. A complete set of solutions for the stream function, ψ, and the vorticity function, ζ, for various Reynolds numbers can be found in Appendix I.

The flow model can be validated by comparison with rigorous studies of viscous flow around spheres. Unfortunately, it is not possible to compare entire flow fields but several important parameters can be used establish the validity of the solution. These paremeters are:

- **The angle of flow separation (θ_s).**

  The angle of flow separation is defined as the angle at which the line of zero ψ intersects the sphere surface as shown in Figure 7.2. This angle can be determined from the point where the surface vorticity distribution is zero [51].

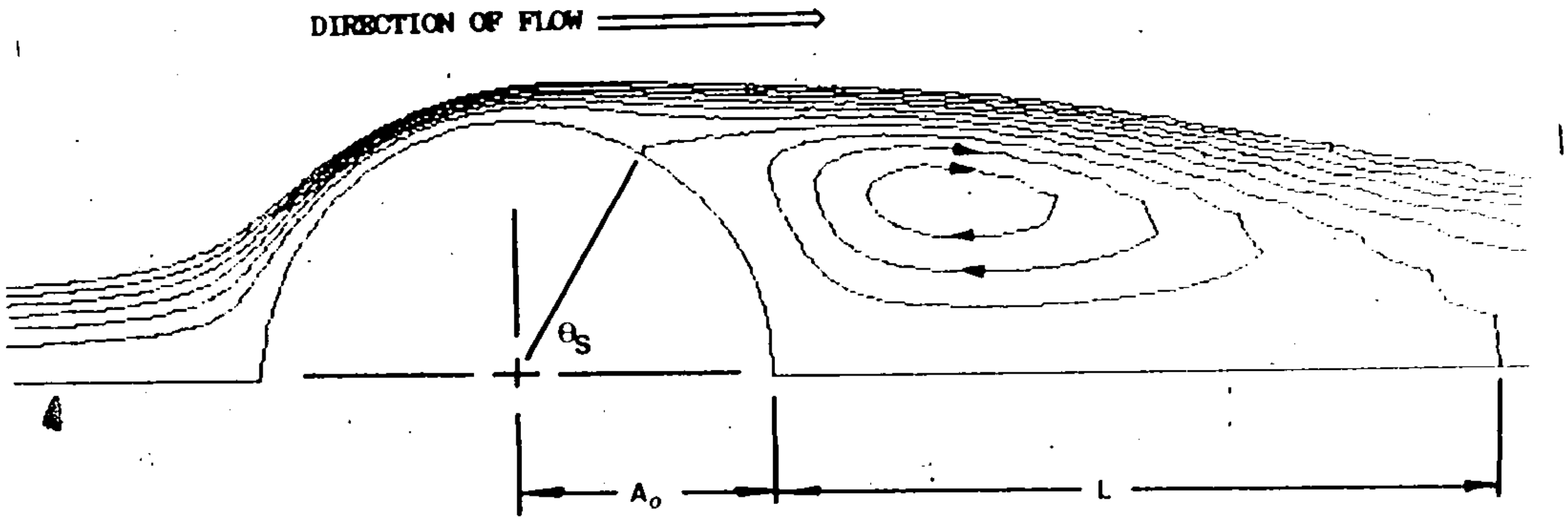

FIGURE 7.2: Viscous Flow Around A Sphere for a Reynolds Number of 200

- The vortex length (L).

  The vortex length is defined as the distance from the rear stagnation point where there is zero velocity along the x-axis [51]. This length is nothing more than the distance from the rear of the collector to the point where ψ equal to zero intersects the x-axis as shown in Figure 7.2.

- The surface vorticity distribution ($\zeta_{surface}$).

  This is the distribution of ζ along the surface of the sphere.

Experimentally and numerically determined results can be found for these quantities in the open literature.

Tables 7.3 and 7.4 are comparisons of the angles of flow separation, θ_s, and vortex lengths, L, determined in this study with numerical and experimental results cited by LeClair [26]. Figure 7.3 illustrates the results of current $\zeta_{surface}$ determinations. A comparison of typical $\zeta_{surface}$ evaluations with rigorous data of LeClair [27] is provided in Figure 7.4. The good agreement between the data validates the present model and demonstrates its ability to predict the complex flow behavior behind a collector. It is interesting to note that the waviness in the surface vorticity, $\zeta_{surface}$, at higher Reynolds numbers is not the result of numerical instabilities. Rather, it indicates the beginning of the formation of a secondary vortex behind the collector. This behavior has been observed experimentally. However, an extremely high resolution grid would be needed to document this phenomena numerically for a sphere [51].

TABLE 7.3: Comparison of Angle of Flow Separation (θ_s)

	Angle of Separation (θ_s)			
Nre	Present	Garner†‡	Tenada†‡	LeClaire†
30	23°	———	———	26°
100	51°	———	51°	53°
200	60°	61°	———	63°
300	66°	65°	———	67°
400	70°	71°	———	73°

† These data sets were taken from LeClaire [26]
‡ Experimental data

TABLE 7.4: Comparison of Vortex Length (L)

	Eddy Length [$L/2A_e$]		
Nre	Present	Taneda†‡	LeClaire†
30	0.154	0.125	0.155
100	0.882	0.920	0.950

† These data sets were taken from LeClaire [26]
‡ Experimental data

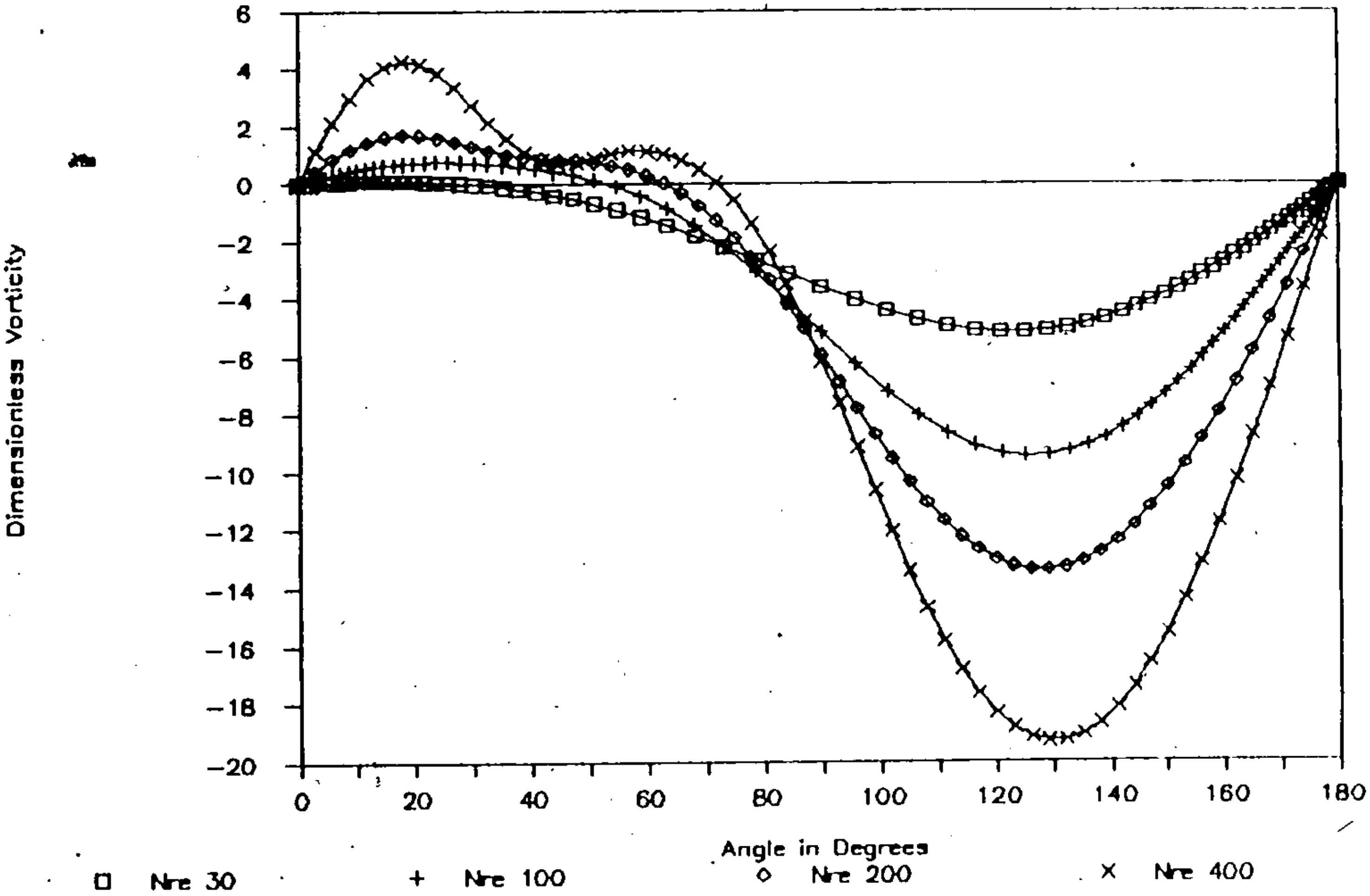

FIGURE 7.3: Calculated Surface Vorticity ($\zeta_{surface}$)

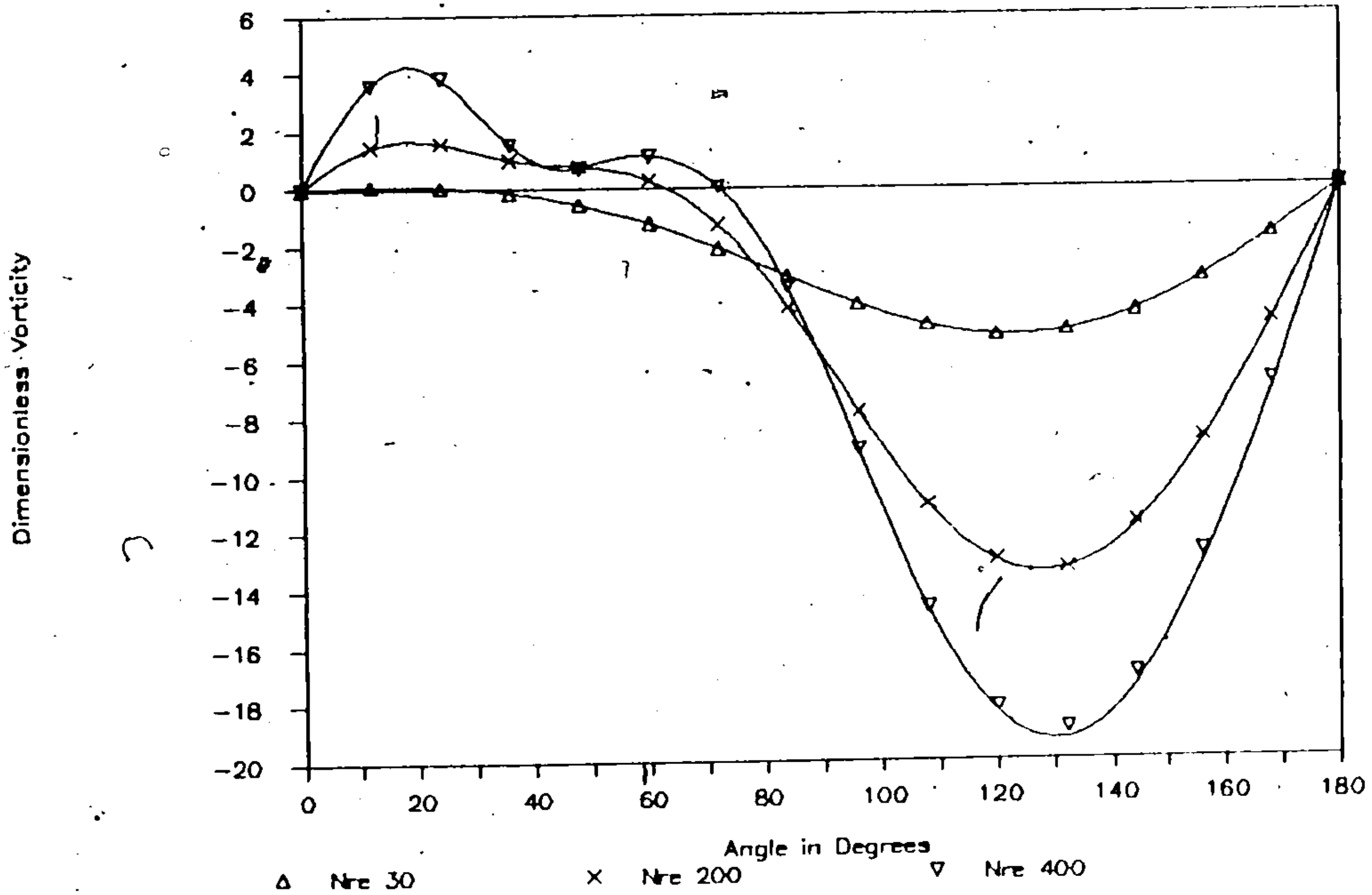

FIGURE 7.4: Comparison of $\zeta_{surface}$ with data of LeClair [27]

7.2.2 Potential Flow Model

The method used to solve the equations governing potential or irrotational flow involved the block-successive relaxation algorithm described in Section 6.1.2. Similar to the viscous model, the potential model employs the variable relaxation factor method to solve the algebraic difference equations. This algorithm was formulated into a FORTRAN 77 computer program that is provided in Appendix F.

The solutions to problems involving irrotational flows are known to be straight forward and rarely present numerical difficulties [38]. With straight flow as an initial guess to the velocity field, the iterative procedure described in Section 6.1.2 converged very rapidly. The system never required more than 350 iterations to reach the convergence criterion defined by Equation 7.1.1.

This portion of the model was validated by comparison to the analytical solution for potential flow around a sphere:

$$\psi = \frac{1}{2}\left[r^2 - \frac{1}{2r}\right]\sin^2(\theta) \qquad 7.2.1$$

The results of the numerical solution agreed with the analytical results within 0.1% everywhere in the solution domain.

Figure 7.5 provides an illustration of the stream function, ψ, for the irrotational flow field around a deformed droplet with an equivalent volume radius, a_o, equal to 3000μm. This pattern

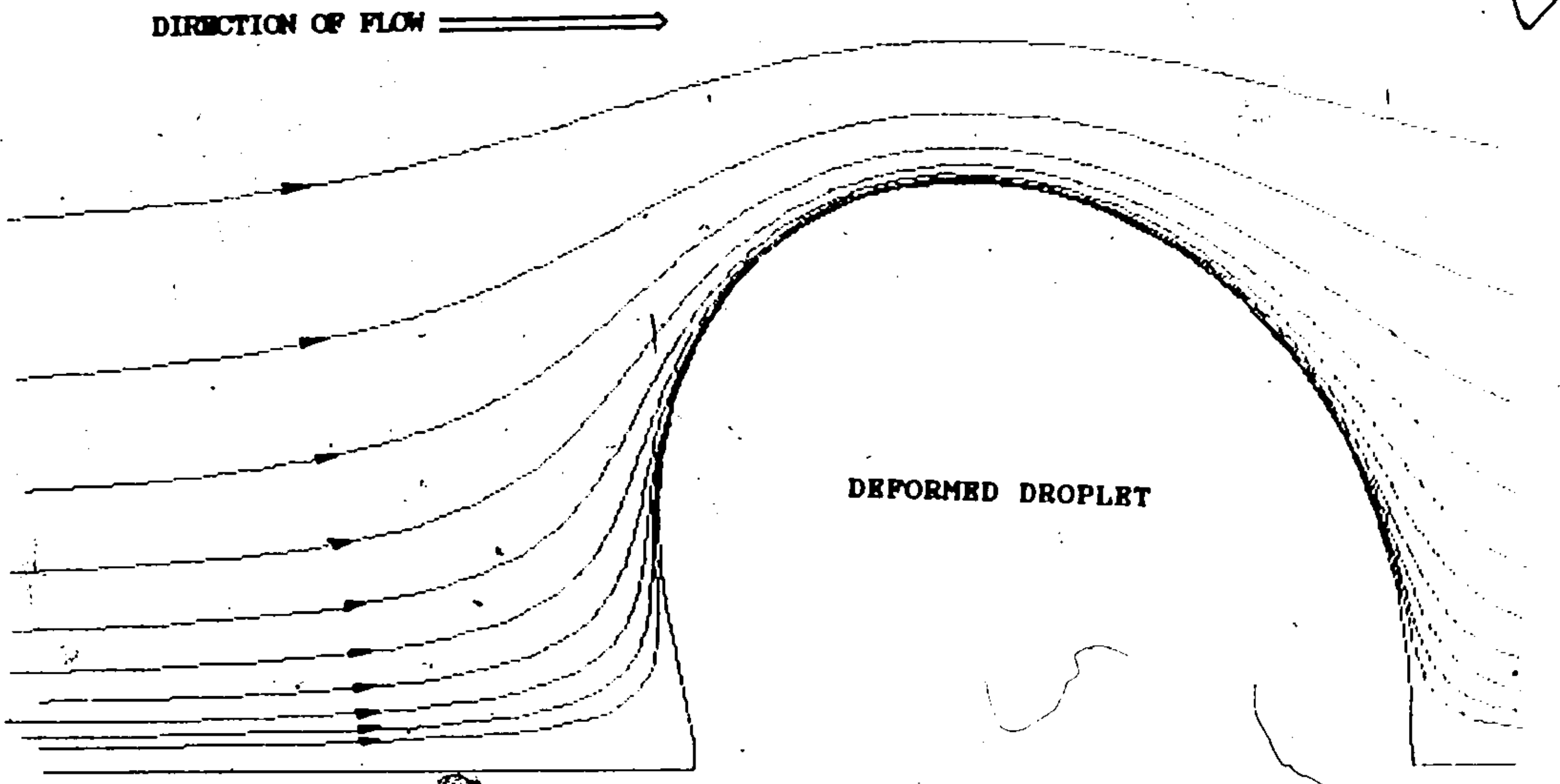

FIGURE 7.5: Calculated Potential Flow Solution for a Deformed Droplet

corresponds to a terminal Reynolds Number of 3600.

7.3 Convective Diffusion Model

The convective diffusion equation, Equation 4.4.3, was solved to provide temperature and vapor distributions around the collector. This equation was solved using the standard point relaxation method with variable relaxation factors as outlined in Section 6.1.3. The algorithm is provided in Appendix G in the form of a FORTRAN 77 computer program. Calculations were made for Reynolds numbers ranging between 1.54 and 400 with both the Prandtl number and Schmidt number set to 0.7.

The similarity between Equations 4.4.3 and 6.1.35a suggest that the numerical systems should behave similarly. As with the viscous flow model, severe underrelaxation was required, but the Ehrlich method adapted very well. Table 7.5 provides a summary of the maximum and minimum relaxation factors required to solve the linear system of algebraic equations that represent forced convection or forced diffusion. The data reinforce the ability of the variable relaxation method to automatically adapt and solve illconditioned matricies resulting from finite difference equations.

TABLE 7.5: Calculated Maximum and Minimum Relaxation Factors for the Convective Diffusion Algorithm

Nre		ω_{max}	ω_{min}
1.54	η	1.971	0.985
30	η	1.970	0.095
100†	η	1.998	0.268
200	η	1.990	0.155
300	η	1.995	0.106
400	η	1.992	0.081

† r_∞ was reduced at this point

Figure 7.6 is a contour plot in equal increments of the dimensionless temperature or vapor, η, for a 433μm radius droplet at its terminal Reynolds number of 200. Appendix I contains similar contour plots for the various Reynolds numbers considered in this work.

In Figure 7.6, the closely spaced contours at the front of the collector indicate high temperature gradients. As the fluid flows from the front of the sphere towards the rear along the surface, it is cooled by conduction from the collector. As a result, the contours spread out indicating a reduction in the temperature gradients towards the rear of the collector. The fluid leaves the collector at the point of flow separation, as shown in Figure 7.2. By this time it is

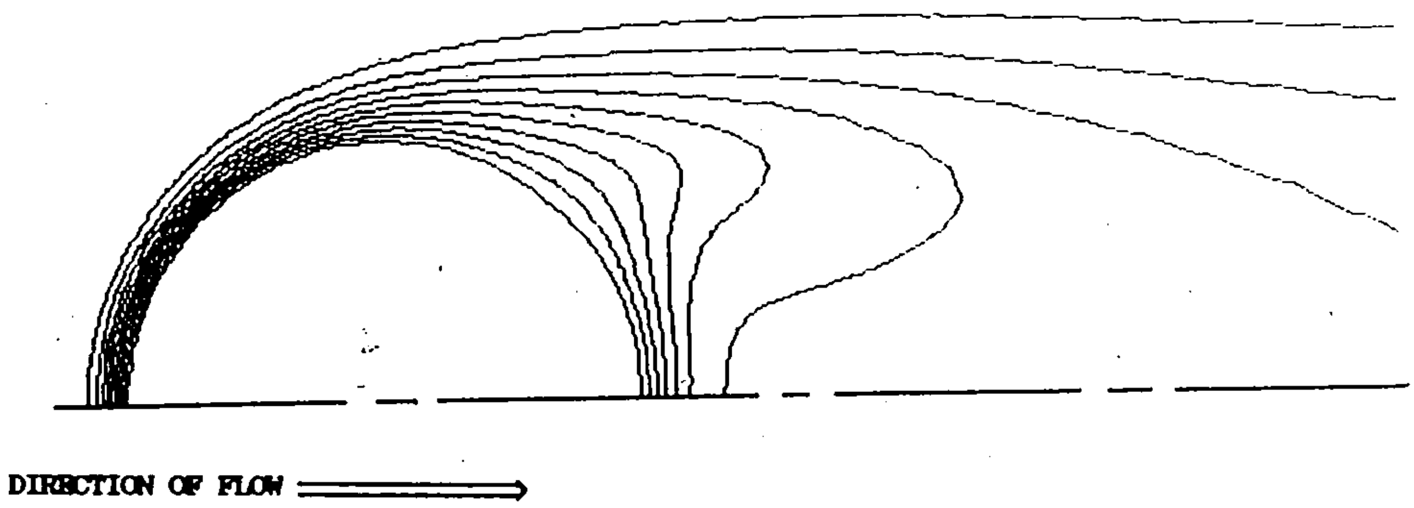

FIGURE 7.6: Calqualted Temperature/Vapor Profile Around a Droplet with a Reynolds Number of 200

well cooled. As a result, the contours are drawn out near the point of flow separation indicating very low temperature gradients.

High temperature gradients occur at the rear of the collector in the presence of a wake because the fluid is moving directly towards the rear of the body. Since the centre of the wake is well circulated it produces the uniform temperatures indicated by the large distances between the contours in Figure 7.6. As a result, there are low temperature gradients at the centre of the wake. It should be noted that the discussion pertaining to temperature gradients also applies to the vapor gradients surrounding a droplet on which condensation occurs.

To ensure that the solutions obtained for this portion of the model were valid, comparisons were made with pertinent data available in the literature. Woo and Hamielec [52] have made extensive numerical studies of the rates of evaporation and heat transfer from spheres which were shown to be in excellent agreement with experimental data. Their local Nusselt number, Nu_ℓ, or Sherwood number, Nsh_ℓ, was related to the dimensionless temperature or vapor gradient according to:

$$Nu_\ell = Nsh_\ell = -2\hat{n}\cdot\nabla\eta = -\frac{2}{h_2}\frac{\partial \eta}{\partial \alpha_2} \qquad 7.3.1$$

at the surface of the sphere. The results of current Nu_ℓ or Nsh_ℓ evaluations as functions of polar angle along the surface of the sphere are illustrated in Figure 7.7. Figure 7.8 provides a

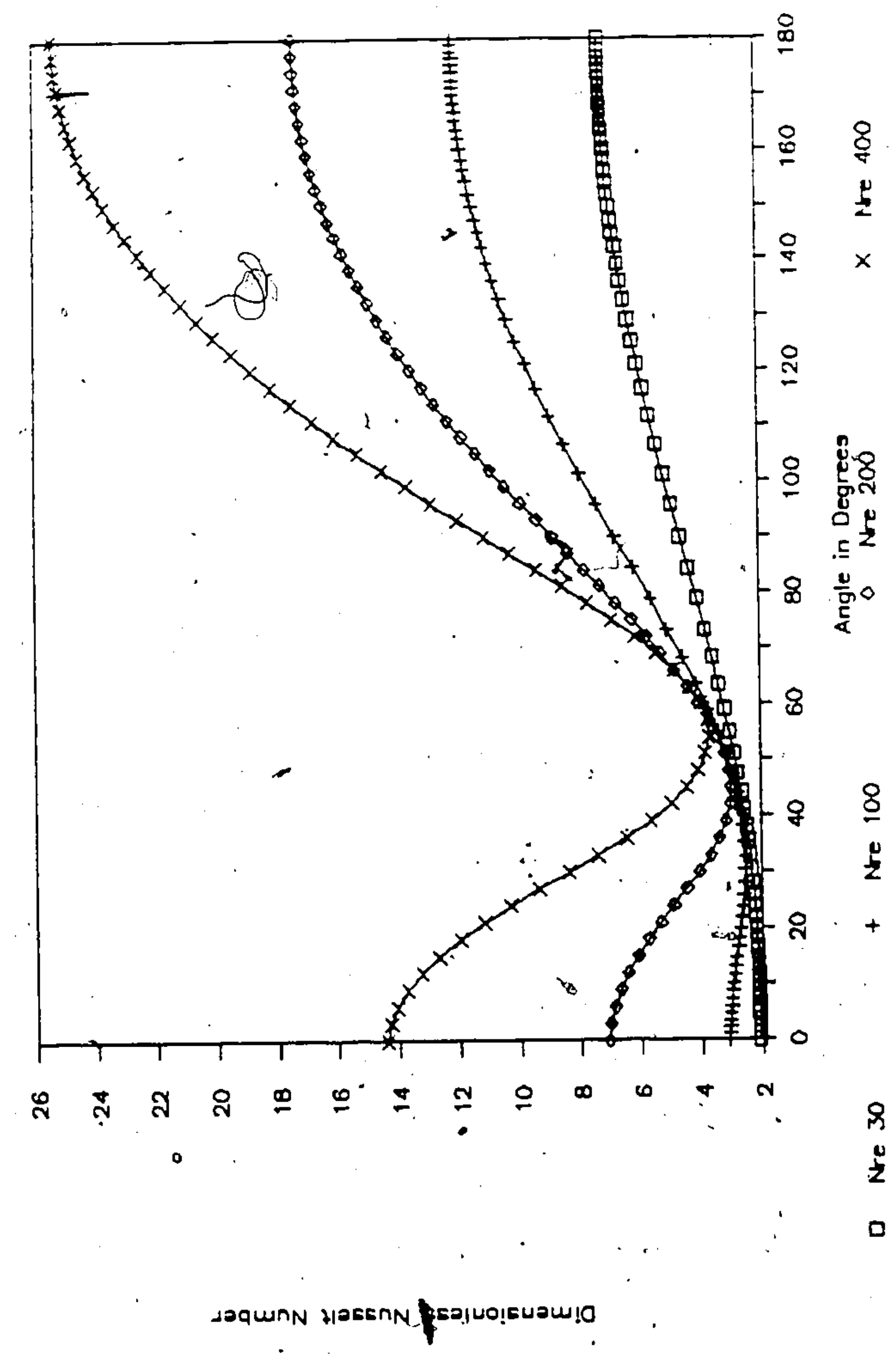

FIGURE 7.7: Computed Surface Nusselt Numbers (Nu_ℓ)

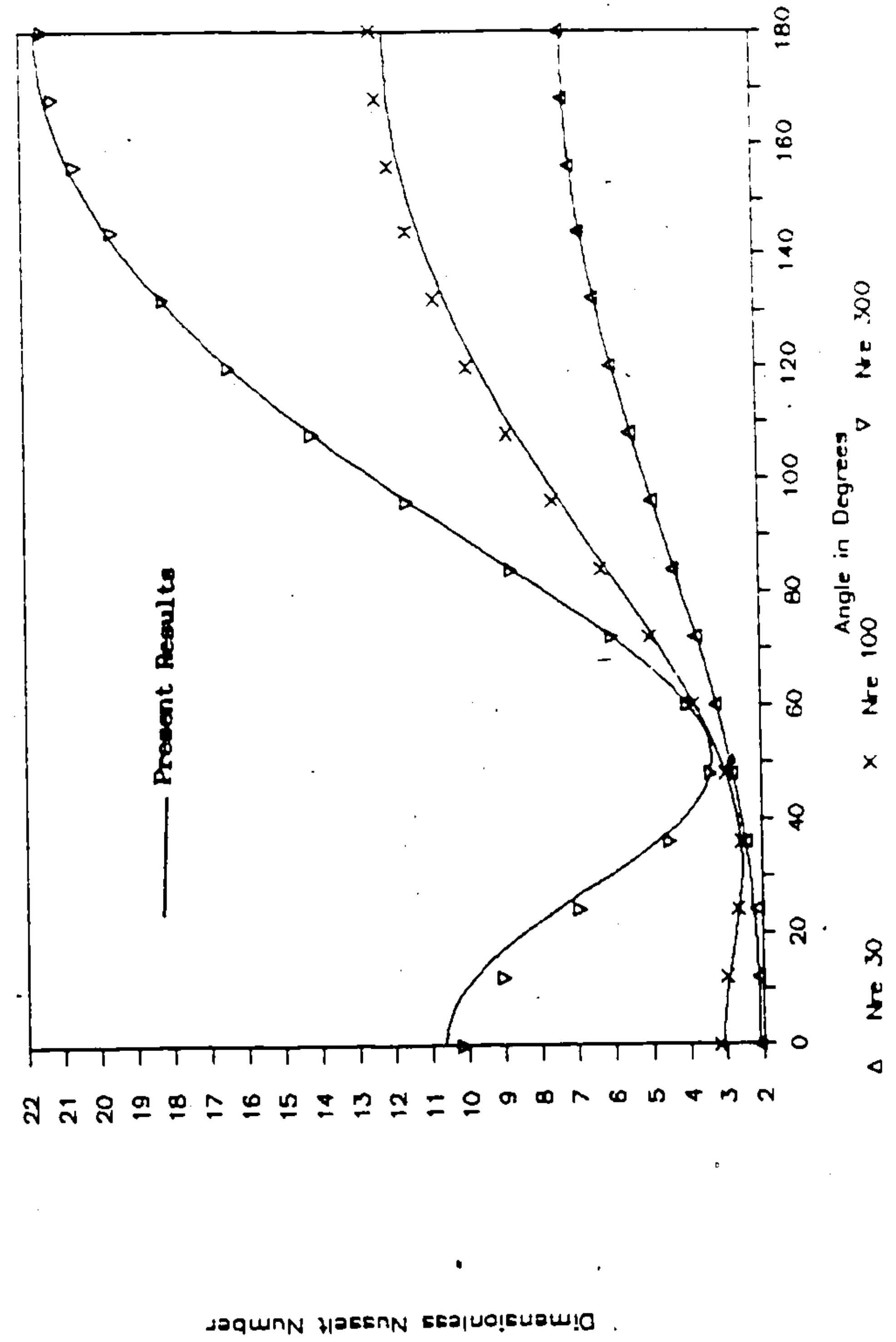

FIGURE 7.8: Comparison of Nu_ℓ with data of Woo [52]

comparison of the new data with the results of Woo and Hamielec [52]. The good agreement indicates that the model used in this investigation accurately predicts the heat transfer and condensation rates associated with water droplets.

7.4 Flux Depostion Model

The method used to solve for the flux deposition efficiency of fine particles involved trial and error calculation of particle trajectories. The trajectories were evaluated by integrating the generalized particle equations of motion with the variable step-size Runge-Kutta-Fehlberg method discussed in Section 6.3. This algorithm was formulated into a FORTRAN 77 computer program which is provided in Appendix H.

The calibration of this complex numerical model should, ideally, involve comparison with experimental data. However, evaluations of single droplet collection efficiencies for submicron particles by phoretic forces are not readily available in the literature. This scarcity of data has been reported by other authors [14,31]. The lack of experimental data forces the calibration to be done by comparison with existing numerical models.

The formulation of the inertial terms in the model can be tested by inserting the potential flow field and generating the well known solution for the case where flux forces are absent [22,24,25,32,37]. Figure 7.9 illustrates the calculated collection efficiencies for this case which are shown to be in excellent agreement with the data

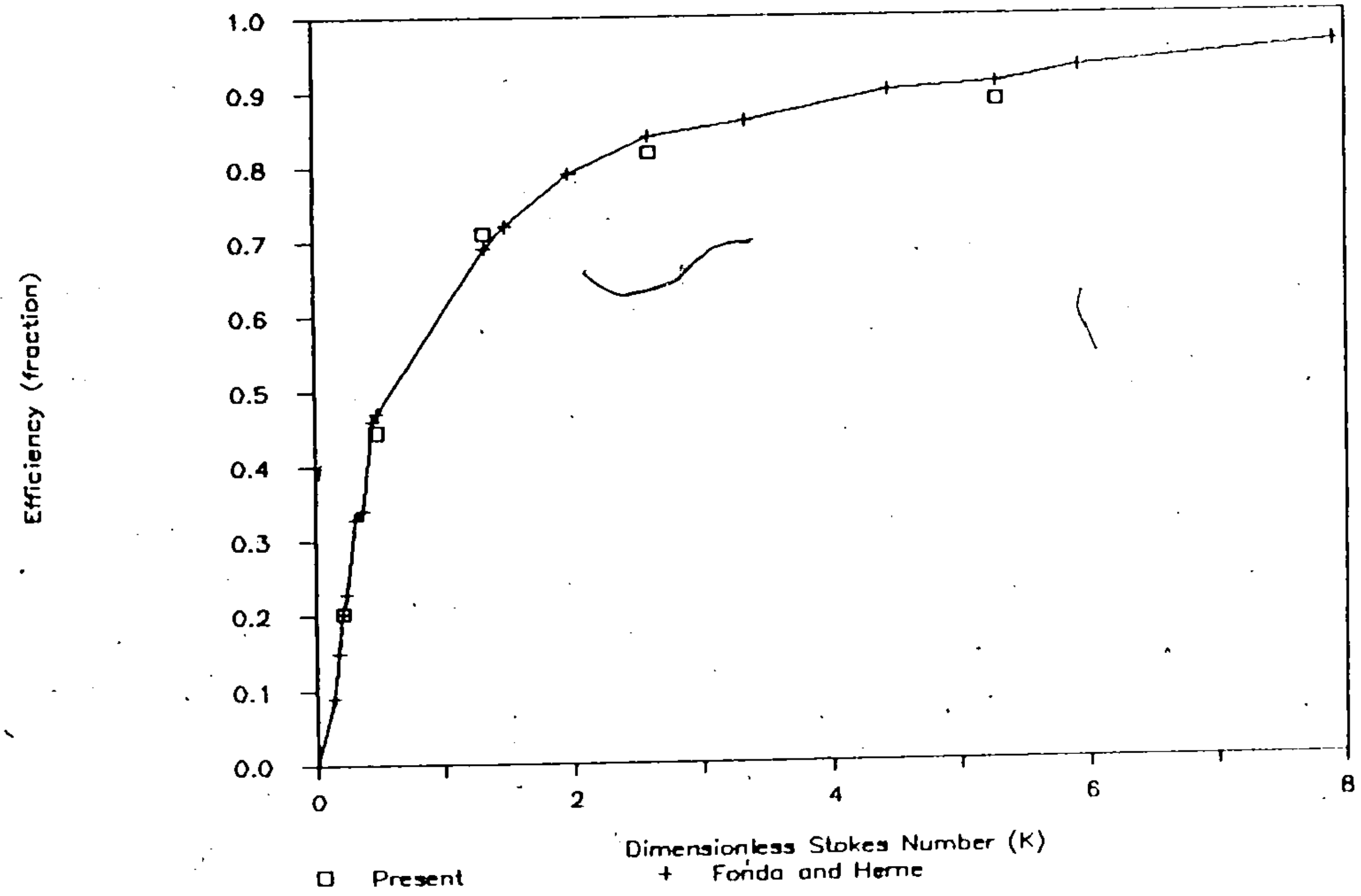

FIGURE 7.9: Potential Flow Collection Efficiencies for a Sphere

provided by Fonda and Herne [11].

Pilat and Prem [39] and Mehta [31] provided identical models for determining the collection of fine particulate matter by flux forces. Correct formulation of the flux terms in this current model can be tested by inserting the simplified flow field, temperature distribution, and vapor distribution assumed by these authors and recalculating their data. Figure 7.10 provides a comparison of their results with those of this investigation. The calculations were made for 100μm diameter droplets at 10°C and 60°C falling at their terminal velocities of 30cm/s in a gas at 65°C. Interception was included because of the small size of the collector.

Figure 7.10 shows excellent agreement between the recalculated and original data of Pilat and Prem. The data provided by Mehta clearly fall well below those of Pilat and Prem and this study. The differences must be due to coding errors since his fundamental assumptions and modelling approaches where identical to those of Pilat and Prem.

The good agreements in the comparisons made in Figures 7.9 and 7.10 indicate that the inertial and flux terms are correctly formulated in general orthogonal coordinates. Therefore, application of more accurate velocity, temperature, and vapor distributions will result in valid particulate matter deposition rates. Additionally, the agreement with the data of Pilat and Prem reinforces the fact that Brownian motion need not be included for the particle size range considered in this investigation when flux forces are present.

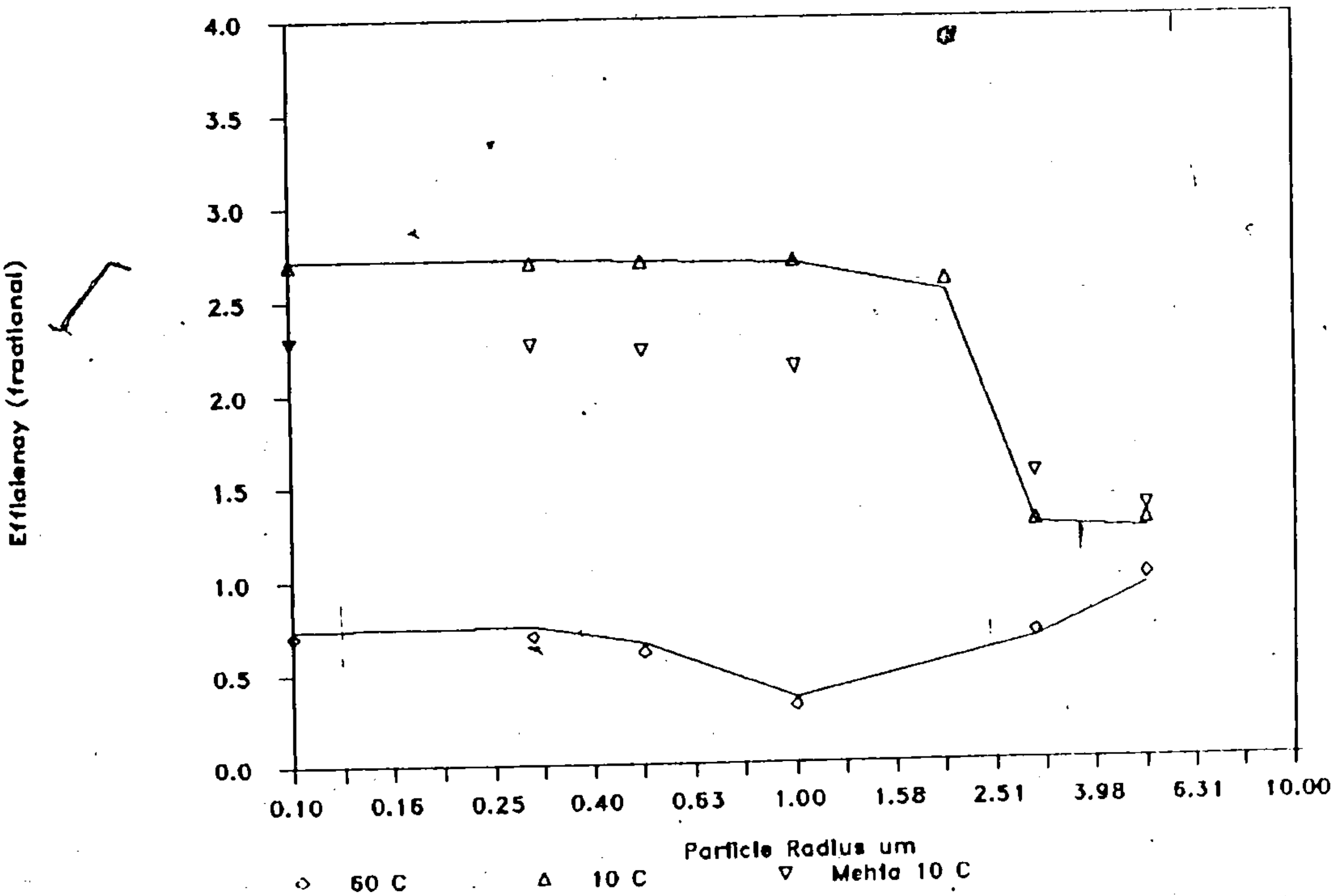

FIGURE 7.10: Recalculation of the Data of Pilat and Prem [30]

7.5 General Results

Particle collection efficiencies were calculated for a range of collector radii varying between 50μm and 4000μm. Accurate velocity, temperature, and vapor distributions were determined numerically as discussed in the previous sections. The particle sizes ranged between 0.04μm and 10μm to include particles in the 'Greenfield gap'. The mechanisms assumed to be responsible for particle collection were:

- for $a_o \leq 620\mu m$ - inertial impaction
  - thermophoresis
  - diffusiophoresis
  - wake capture
- for $a_o \geq 620\mu m$ - inertial impaction (with drop deformation)

The scrubber was assumed to operate with water at 10°C while the gas temperature varied between 20°C and 95°C.

The physical properties assumed for the calculations in this investigation were:

AIR

$$\mu_f = 1.817\times10^{-5}\ \mathrm{kg/m\cdot s}$$
$$\rho_f = 1.188\ \mathrm{kg/m^3}$$

k_f = 0.028514 J/m·s·°C
C_p = 1000 J/kg·°C
P_{total} = 1.0138 bar

PARTICLE

ρ_p = 1050 kg/m^3
k_p = 0.49 J/m·s·°C

The diffusivity of water in air [31] and the saturation water vapor pressure [36] were determined from the following empirical correlations:

$$\mathcal{D}_{12} = \frac{2.171\times10^{-5}}{P_{total}}\left[\frac{t_{ave}}{273}\right]^{1.75} \qquad 7.5.1$$

where

$\mathcal{D}_{12}$ ≡ [m^2/s]

t_{ave} ≡ [°K]

$$P_{sat} = P_c\cdot10^{[k(1-t_c/t)]} \qquad 7.5.2$$

where

$k = -5.1514\times10^{-9}t^3+9.9533\times10^{-6}t^2-6.2442\times10^{-3}t+4.39553$

t ≡ [°K]

t_c = 647.6°k

P_c = 220.9 bar

The general results of this investigation were obtained for the case where condensation occurs on cool water droplets moving in warm humid gas streams. To perform the calculations, the following simplifying assumptions were made:

1) The water droplet was assumed to be at a constant temperature during the time that a particle was passing the collector.
2) The amount of water condensing on the collector during the time that a particle was passing the collector was small compared to the droplet volume.

As a worst case, only 0.012 seconds (approximately) were required for a particle to pass a 100μm diameter droplet falling at its terminal velocity from a point where the gas is not disturbed (70 collector radii upstream).

Steady state conditions are implied in the above assumptions. However, this does not mean that the collector cannot change size or temperature as it moves down the length of the scrubber. Rather, the implication is that the conditions around the collector can be modelled in terms of a pseudo steady state system. Changes to the condition of the droplet are considered to be gradual so that it is always in equilibrium with its surroundings.

The physical aspects of the problem suggest that water condensing on the surface of the droplet will raise the surface temperature slightly due to the latent heat of vaporization. At the same time, a small amount of heat will be conducted to the droplet from the free gas stream. The rise in surface temperature should be reduced by the action of internal circulation inside the collector. In addition, if the droplet has a large enough mass, its temperature should remain constant during the shot time required for a particle to pass the collector.

Figures 7.11 to 7.16 depict the results of particle collection efficiency calculations for a water temperature of 10°C and a gas temperature range between 20°C and 95°C. From these illustrations, it is evident that particle collection efficiencies can be increased significantly by the actions of phoretic forces. The increases can be as large as several orders of magnitude and they are most prominent at lower Reynolds numbers. It is not surprising that efficiencies are greater than unity in some instances. This condition simply implies that some particles are collected from an area greater than the projected area of the collector due to the influence of flux forces.

The data presented in Figures 7.11 to 7.16 demonstrate only trends for the various parameters involved in determination of collection effiencies. Flux depostion rates are influenced primarily by:

- the complex flow behavior resulting from the formation of a vortex.
- the effects of changing Reynolds number
- the effects of increasing particle mass
- the role of droplet deformation.

Clearly, several of these parameters influence one another, as

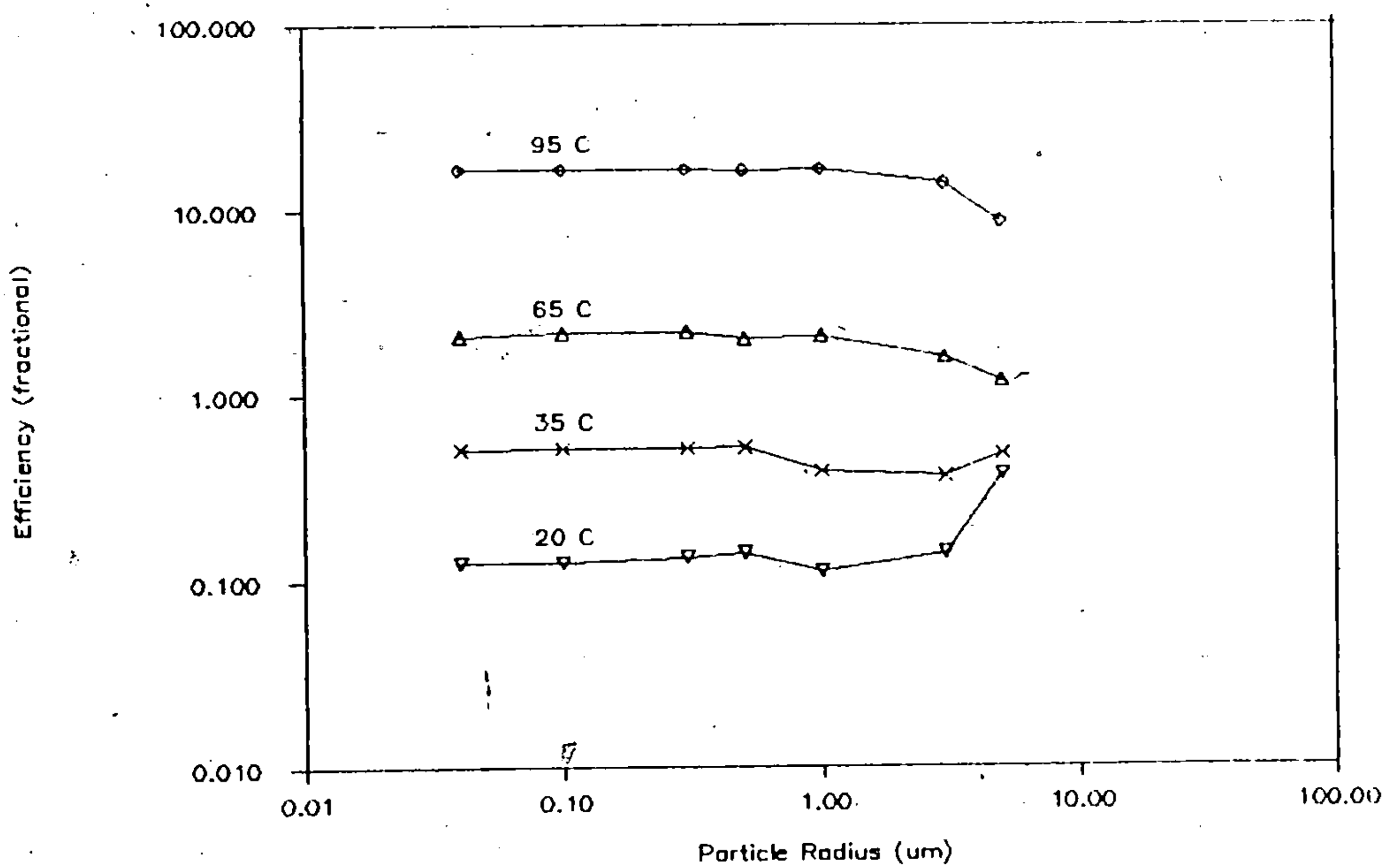

FIGURE 7.11: Flux Deposition for a Reynolds Number of 1.54

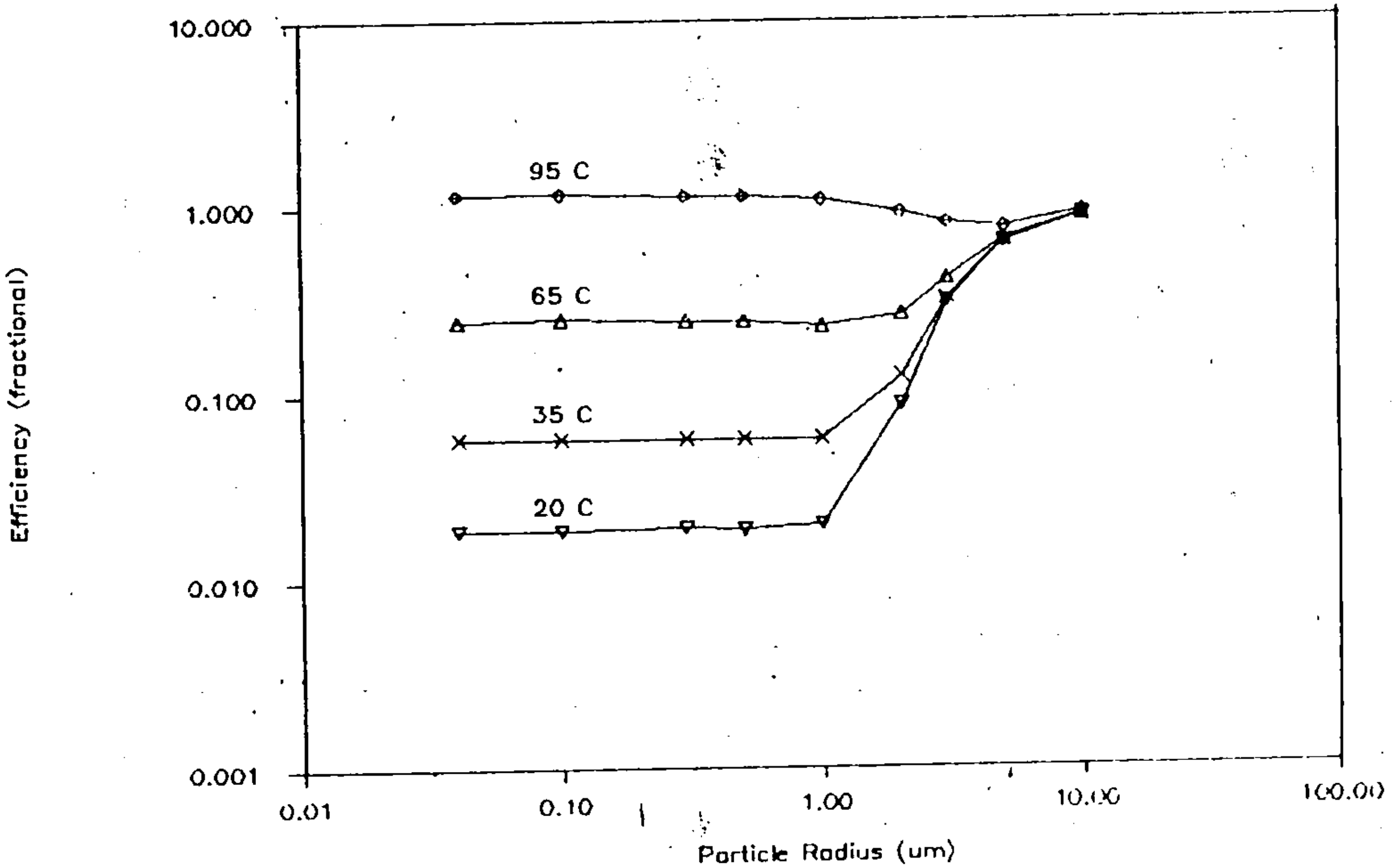

FIGURE 7.12: Flux Deposition for a Reynolds Number of 30

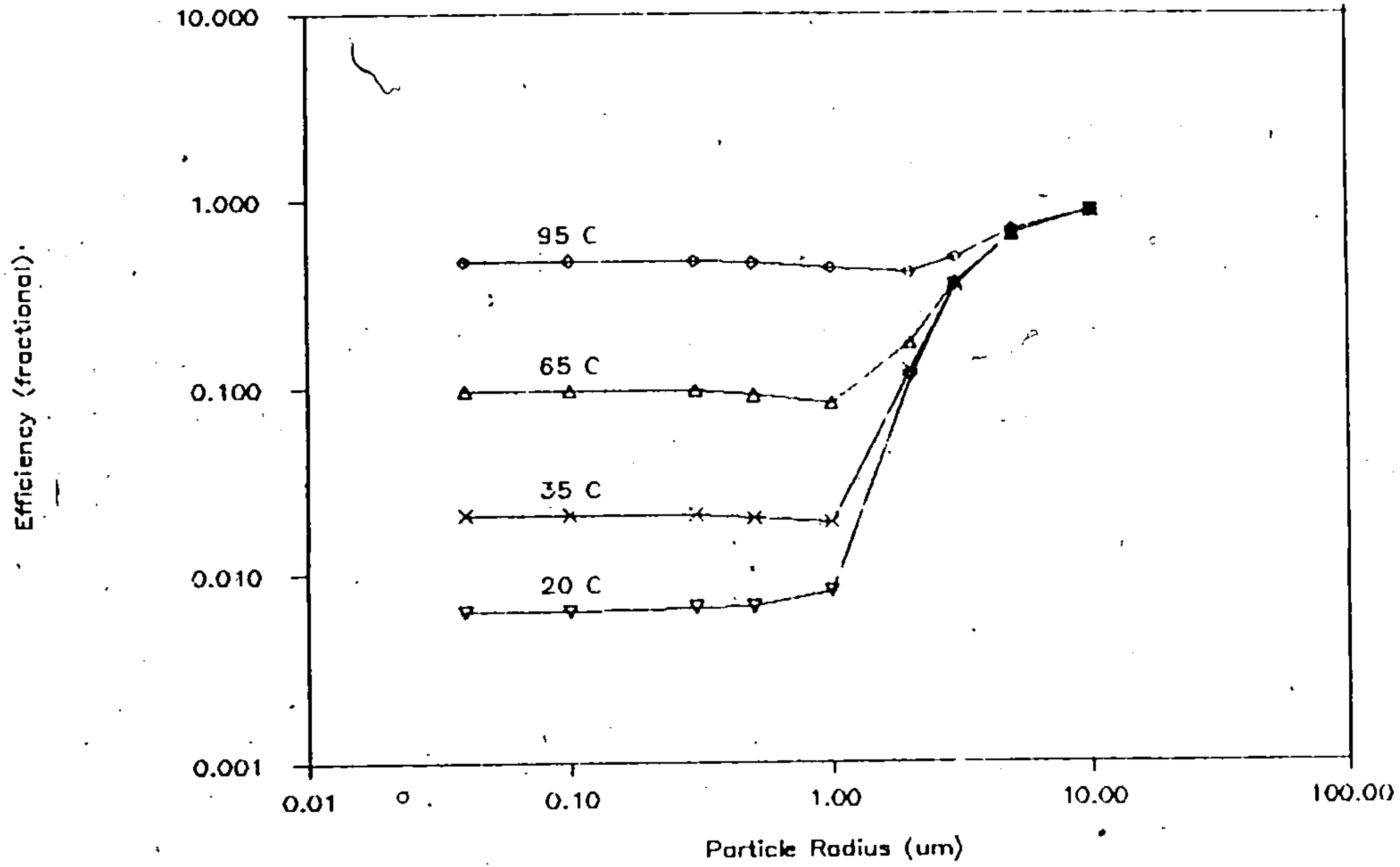

FIGURE 7.13: Flux Deposition for a Reynolds Number of 100

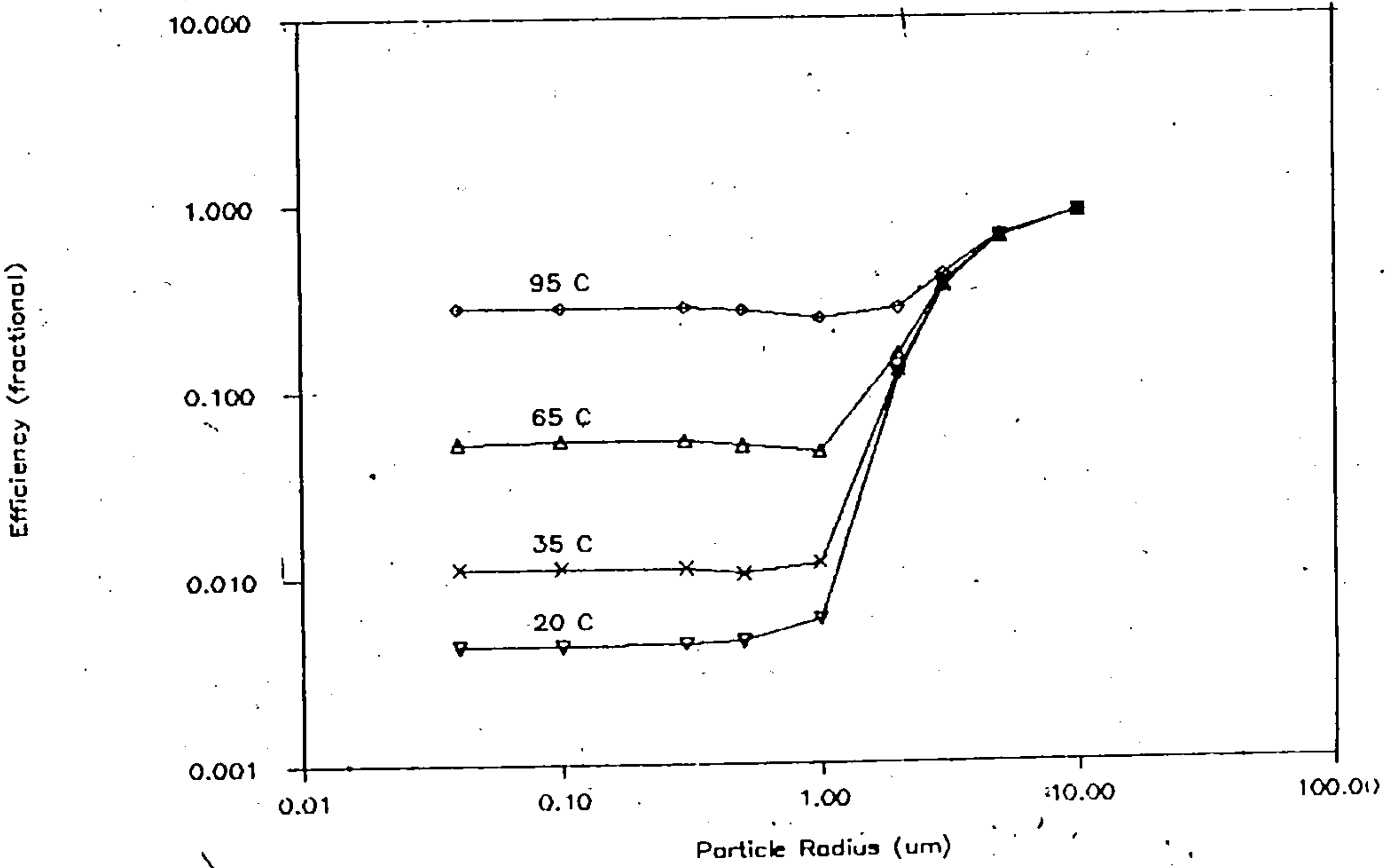

FIGURE 7.14: Flux Deposition for a Reynolds Number of 200

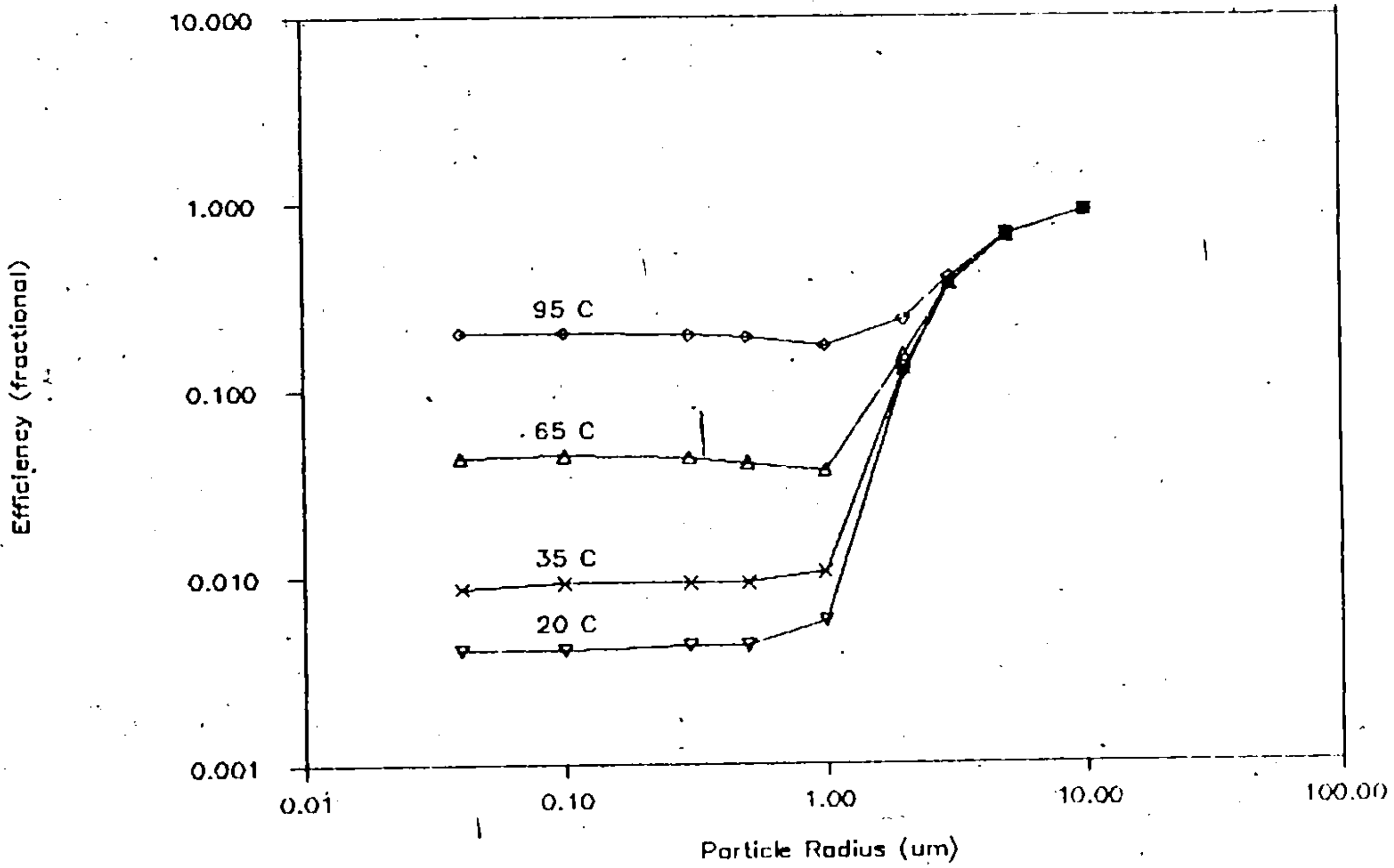

FIGURE 7.15: Flux Deposition for a Reynolds Number of 300

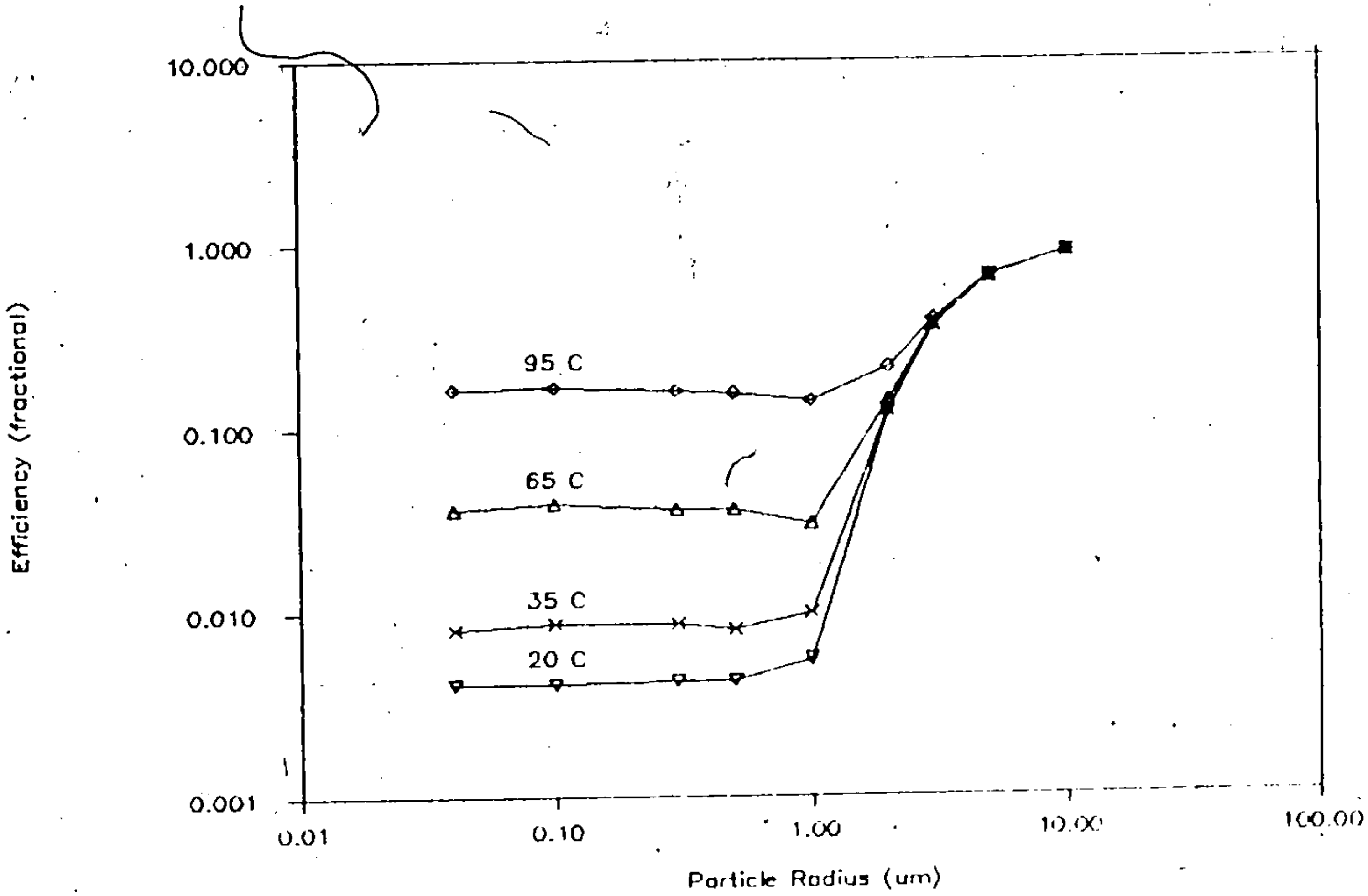

FIGURE 7.16: Flux Deposition for a Reynolds Number of 400

demonstrated by the effects of Reynolds number on vortex formation. However, each parameter has a distinct individual influence that merits further discussion. Droplet deformation does not affect flux deposition because of the extreme Reynolds numbers involved. However, it has been included because of the simplicity with which this generalized model could account for it.

7.5.1 Collection in the Wake

The collection mechanisms accounted for, as mentioned earlier, are inertial impaction, thermophoresis, diffusiophoresis, and wake capture. Wake capture is determined by predicting the complex flow behavior behind the collector as discussed in Section 7.2. The existence of wake capture has long been known to exist for fine particulate matter [14]. Neither the magnitude nor the mechanism of rear deposition under the influence of flux forces appears to have been discussed in the literature.

Figure 7.17 illustrates the wake capture of a $0.3\mu m$ radius particle by a droplet for a Reynolds number of 100 when the gas and droplet temperatures are 95°C and 10°C respectively. The same limiting trajectory is superimposed on the temperature/vapor distribution and the velocity field which is described in terms of the stream function, Ψ. From this representation, a definite mechanism can be visualized for wake capture when flux forces are weak in some regions and strong in others. The changes in the strengths of the flux forces are illustrated in Figure 7.18 in terms of the

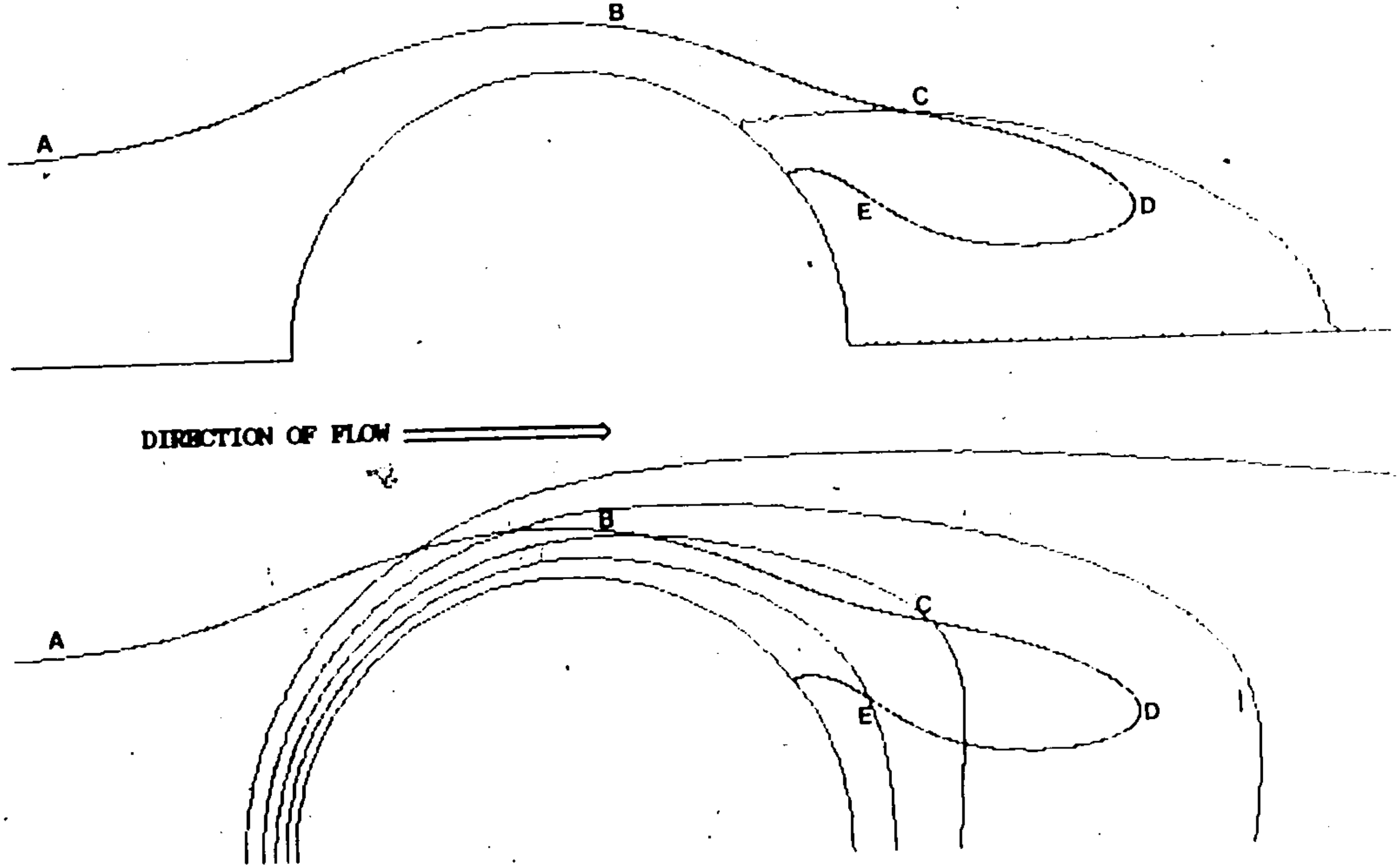

FIGURE 7.17: Rear Collision of a 0.3μm Particle

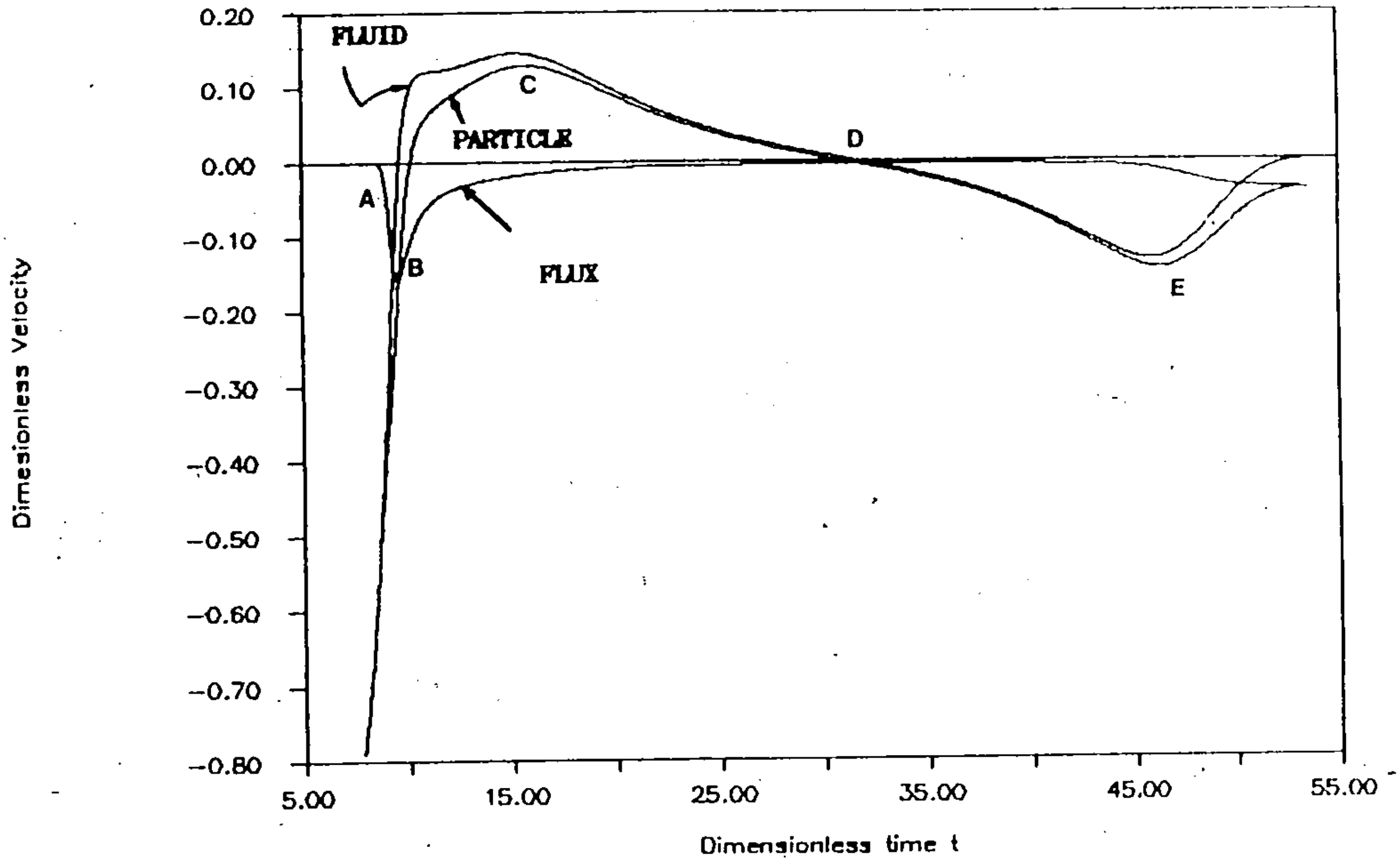

FIGURE 7.18: Radial Velocity Diagram for Rear Collision

dimensionless radial component of the flux velocity and the fluid velocity.

Referring to Figures 7.17 and 7.18, the flux velocities are small at point A and a particle is moved to the collector predominantly by the hydrodynamic effects. As the particle is carried close to the collector, high flux forces at point B help to move the particle towards the collector and into the wake region. Near the point of flow separation, as discussed earlier, the flux forces are reduced. As a result, the particle is carried away from the collector in the wake at point C. The total absence of flux forces in the wake causes the particle to follow the fluid, essentially, until it is pushed back towards the collector at point D. As the particle comes into close proximity of the collector, the high flux forces, resulting from the reverse flow, capture the particle near point E.

The sequential steps responsible for rear capture consist of the particle:

- being carried close to the collector by hydrodynamic forces.
- being pulled into the wake region by flux forces
- being moved around and back towards the collector by the hydrodynamic forces in the wake due to the total absence of flux forces
- finally being collected by the strong flux forces that exist near the collector surface due to the reverse flow of the fluid.

The presence of the wake has two effects relative to the capture of a particle. The wake can influence the particle motion directly through hydrodynamic forces and it can influence particle motion through its affects on the temperature/vapor distribution. In regions where the wake produces low temperature/vapor gradients, the flux forces will be reduced.

Resolution of the question of whether the presence of the wake actually enhances flux deposition of particles is very complicated. For higher Reynolds numbers, the low temperature/vapor gradients in the wake region are more pronounced as shown in Figure 7.6. If the wake was not present, the resulting higher and more uniform gradients could produce higher collection efficiencies than possible through the combined action of the wake and flux forces.

Figures 7.19 and 7.20 demonstrate another type of wake capture that can occur. The data sets are presented in a similar manner as the data shown in Figures 7.17 and 7.18 except that the particle size is increased to a 0.5μm radius. It is easy to imagine a small (nearly massless) particle being forced into a wake and circulating forever in the absence of other forces. The decaying spiral phenomenon appears to be the result of the complex interaction of a weak oscillating flux force and the fluid drag.

With reference to Figure 7.20, the magnitude of the oscillating flux velocity is generally larger when the particle is moving away from the collector (negative values are directed towards the collector). As a result there is a larger reduction in the particle

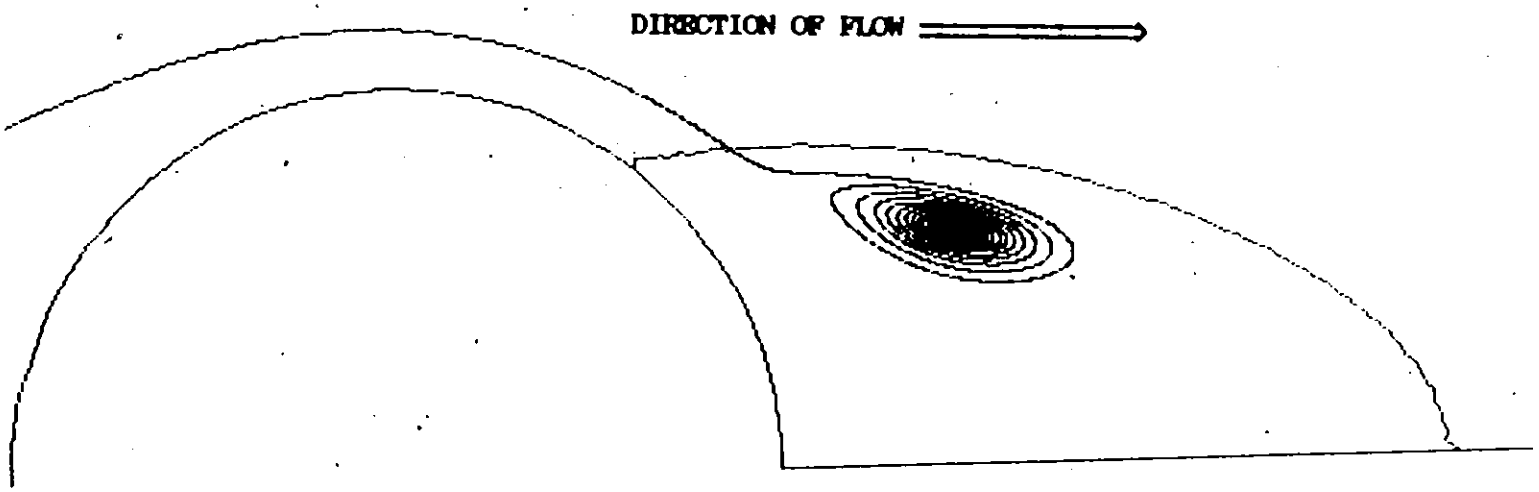

FIGURE 7.19: Wake Entrainment of a 0.5μm Particle

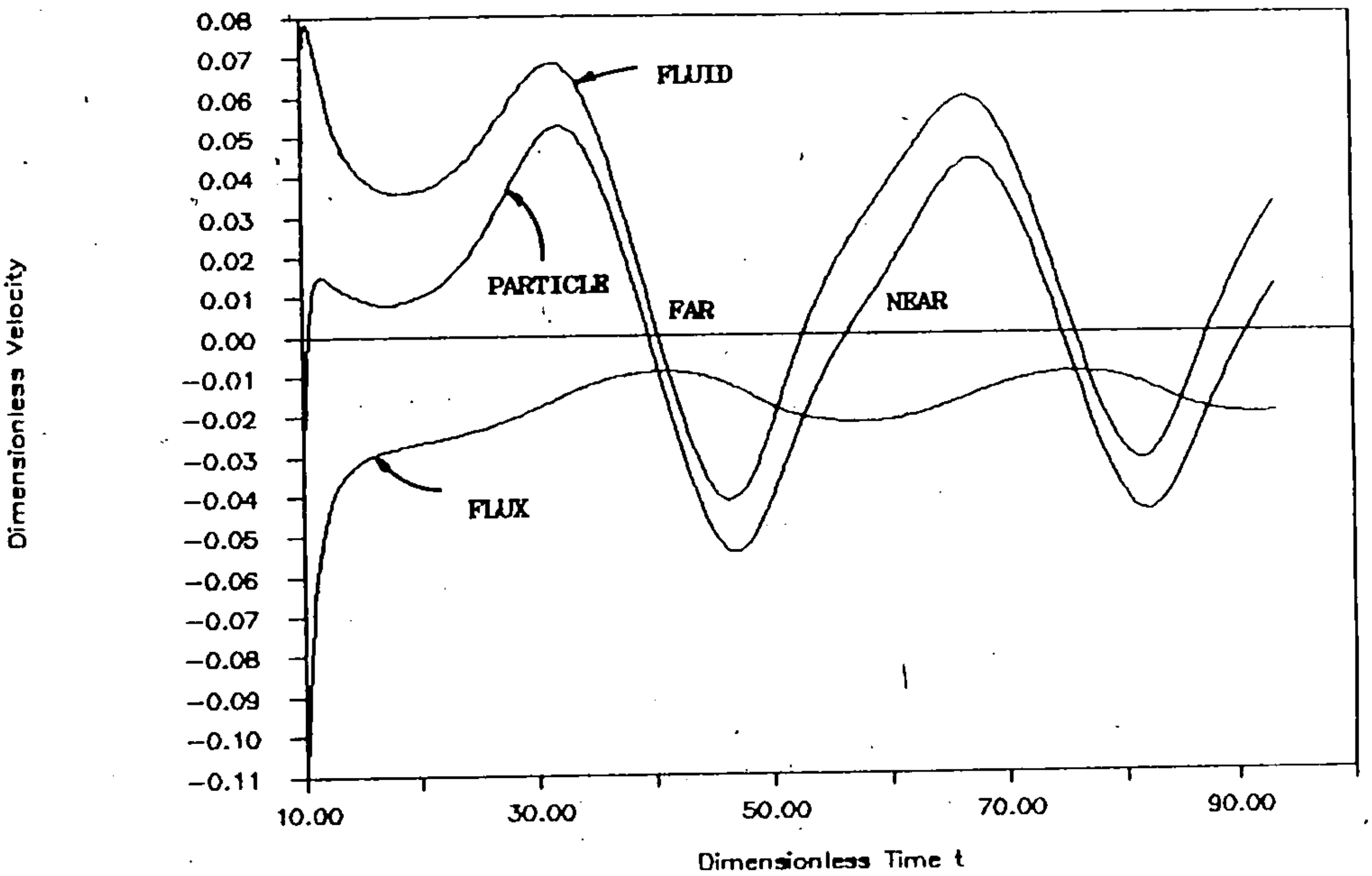

FIGURE 7.20: Radial Velocity Diagram for Wake Entrainment

velocity when it is being dragged away by the fluid than the increase when it is moving towards the collector. The reduction in particle velocity moves it closer to the centre of the vortex where the fluid velocities are lower. Imagining that the particle has now moved into a 'ring' of slower moving fluid, it appears that as the particle moves towards the collector its velocity is enhanced by the flux velocity which would normally pull the particle closer to the collector or into a 'ring' of higher fluid velocity. However, since the velocity enhancement is less when the particle moves towards the collector, the flux force is not strong enough to move the particle as far out of the slower 'ring' of fluid as it has been moved in earlier during the preceeding half cycle.

The net result of the cycling is a constant reduction of fluid velocity and particle velocity. In addition, although it is difficult to see from Figure 7.20, the flux force decays as the particle moves towards the well mixed centre of the vortex. Eventually, the particle stops in the centre of the vortex where there are no forces acting on it.

The mechanism of wake capture has been demonstrated in this investigation but its relative importance has not been considered yet. Figure 7.21 illustrates the ratio of the number of particles captured on the front of the collector to the total collected considering particles of 0.1μm and 0.5μm radius being collected on a droplet at 10°C from a gas at 65°C. The small particle sizes and relatively large temperature difference were chosen to promote rear capture.

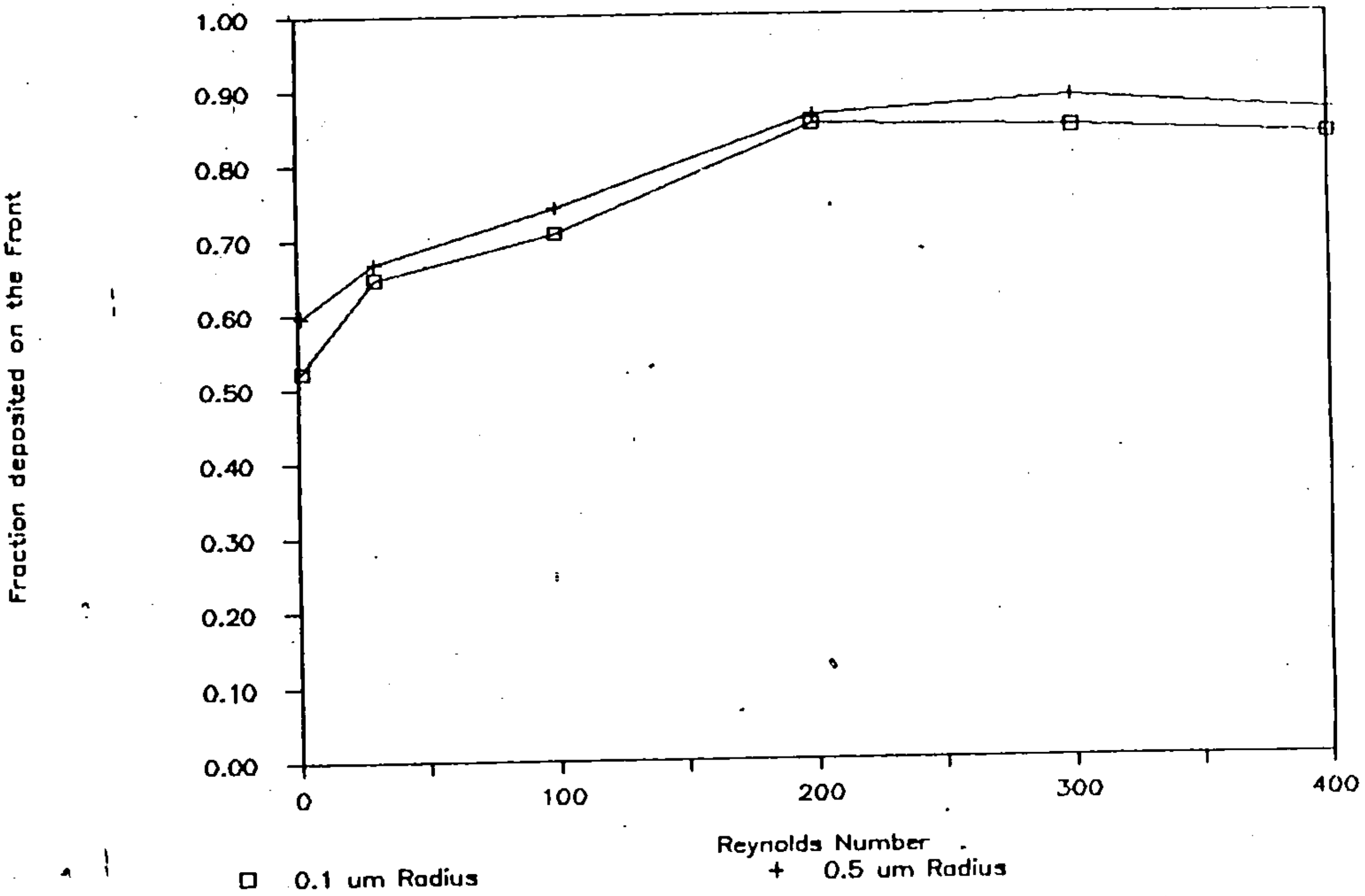

FIGURE 7.21: Fraction of Deposition Occurring on the Front of a Collector

When the Reynolds number is low, the fraction captured on the rear of the collector is as high as 50%. For higher Reynolds numbers most of the deposition occurs on the front of the collector. Although it has been shown that there is potential for the wake to capture particles, its relative importance to flux deposition is low because most of the particles do not reach the wake.

7.5.2 The Effects of Reynolds Number

The role that Reynolds number plays in producing higher collection efficiencies through inertial impaction has been well documented in the literature [3,9,24,48]. The effects of Reynolds number on flux deposition rates is not as straight forward because of the complex interaction of wake formation, increased inertial forces on the particle, and boundary layer thickness.

The collection efficiencies shown in Figures 7.11 through 7.16 appear to be relatively constant for particle radii, r_p, less than 1μm. Similar trends were noticed by Pilat and Prem [39]. This constant value decreases dramatically with increases in Reynolds number for a given temperature. This decrease is illustrated in Figure 7.22 which depicts the dependence of collection efficiency of small particles on Reynolds number for various temperatures. When considering an explanation for this behavior, two factors must be considered:

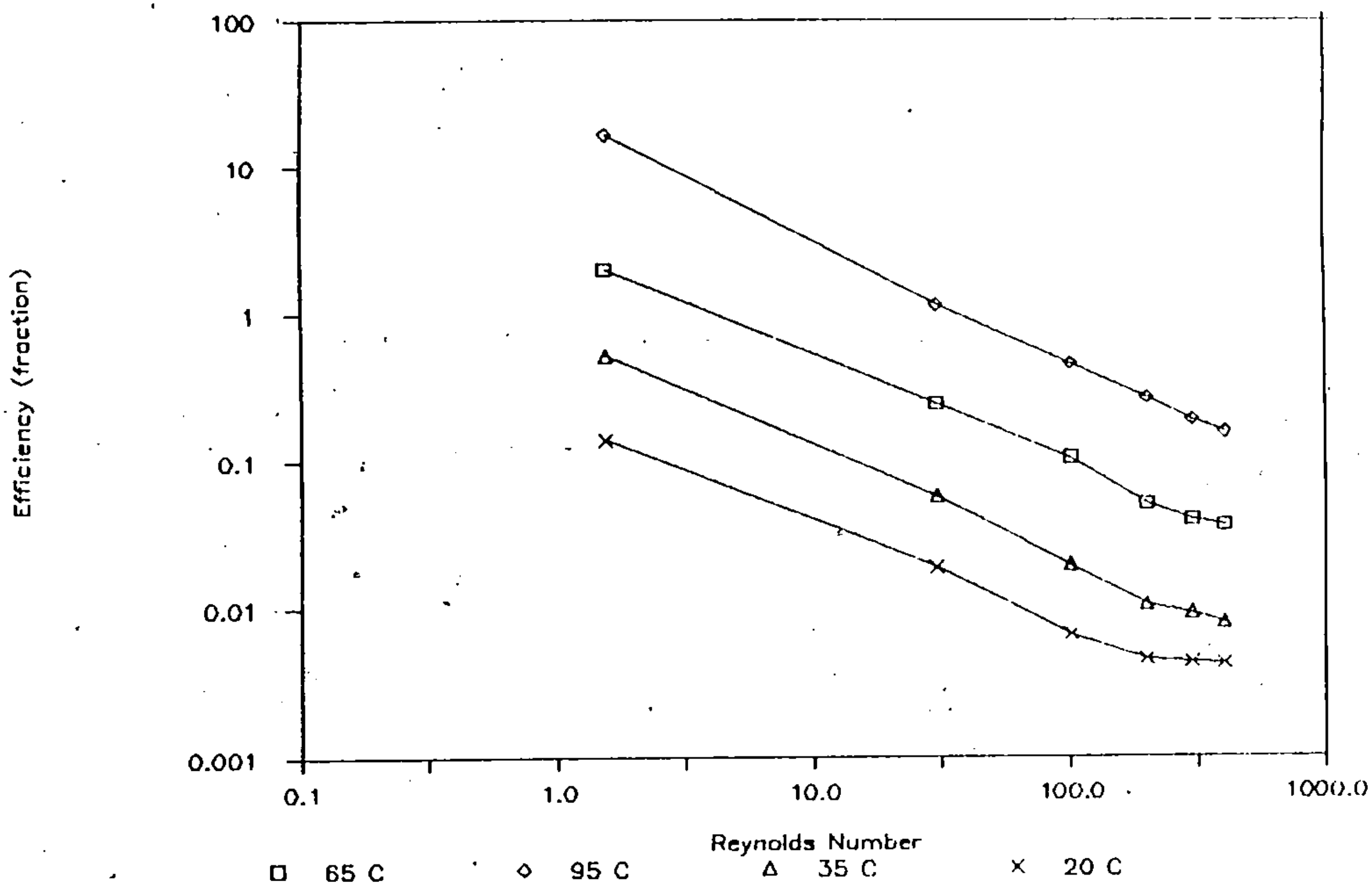

FIGURE 7.22: The Effects of Reynolds Number on Flux Deposition

- An increase in Reynolds number would reduce the thermal/diffusive boundary layer as shown in Figure 7.23. Consequently, flux deposition is retarded because the region in which the particle can be affected by the flux forces will be smaller.
- This same reduction in boundary layer thickness would increase the temperature/vapor gradients as shown in both Figures 7.7 and 7.23. Consequently, flux deposition should increase because of the increased flux forces in this boundary layer. Also, there is a greater inertial force moving the particle to the sphere at higher Reynolds numbers.

The data in Figure 7.22 appear to support the first of these considerations as the controlling factor that governs flux deposition.

An unexpected result that is present in Figure 7.22 is the remarkably linear relationship between the collection efficiency of small submicron particulate and the Reynolds number. This linear relationship on a log-log plot suggests a correlation of the form:

$$E = a \cdot Nre^{b} \qquad 7.5.3$$

where 'a' and 'b' are, generally, functions of the droplet and gas temperatures. It is clearly evident from Figure 7.22 that the slopes of the lines for different temperatures are relatively constant. This trend suggests that 'b' is a constant that is independent of the temperatures of the gas and the collector. Therefore, a generalized expression for the flux deposition of small particles can be written as a product of two functions according to:

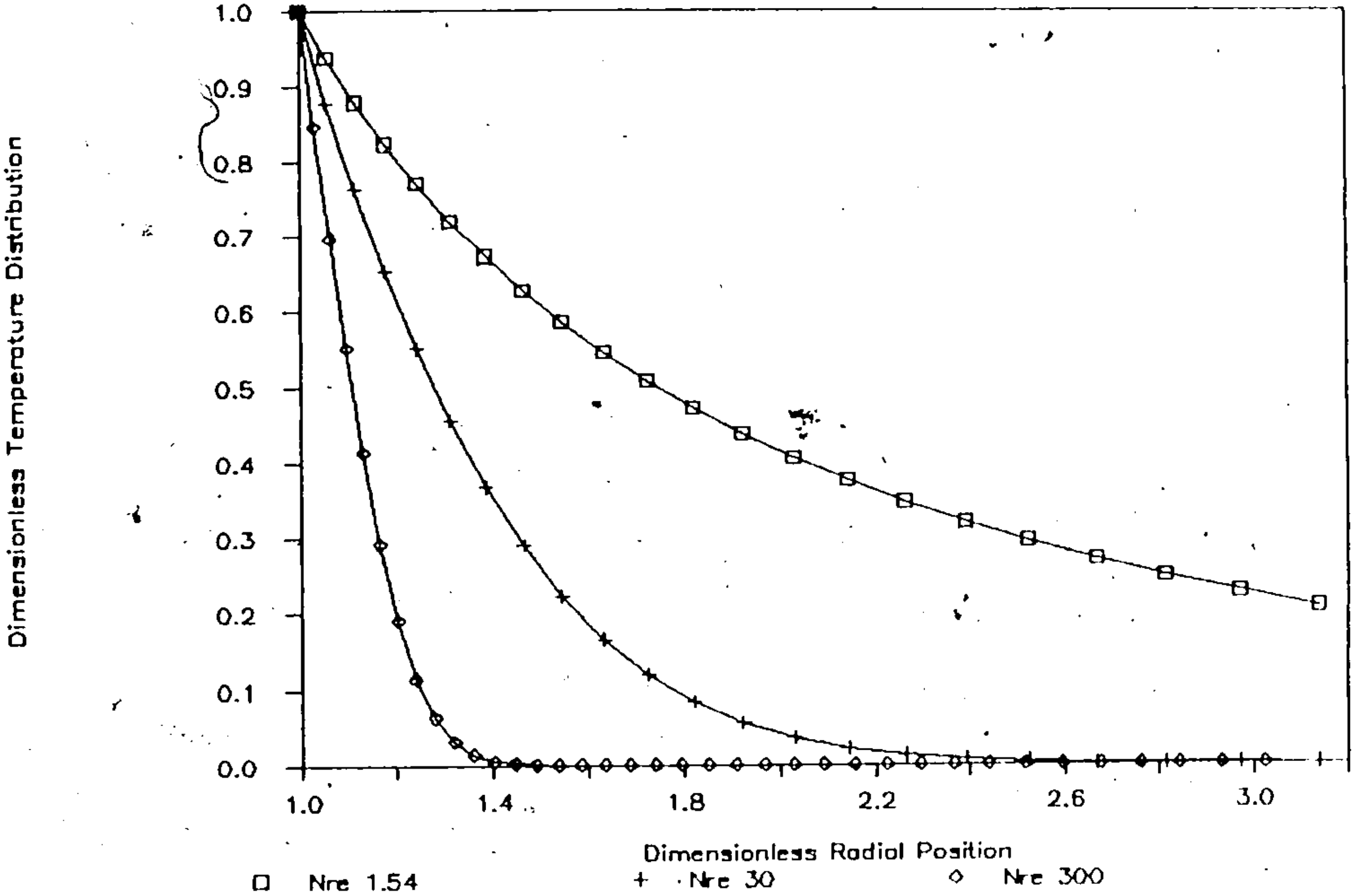

FIGURE 7.23: The Reduction of Thermal/Diffusive Boundary Layer

$$E = F(T_d, T_f) \times G(Nre) \qquad 7.5.4$$

where

$$G = Nre^{-0.78}$$

In Equation 7.5.4, the function F reflects the change in flux deposition due to an increase in the temperature difference between the gas and the collector. The function G represents the change in the deposition rate as a result of changes in Reynolds number. The simple form for the affect of Reynolds number on flux deposition can be explained through earlier findings. Since the amount of collection at the rear of the collector is small when the wake is most prominent, the simple expression for the affect of Reynolds number is the result of the reduction of the uniform portion of the thermal/diffusive boundary layer.

7.5.3 The Effect of Particle Mass

The calculated particle collection efficiencies in Figures 7.11 to 7.16 generally exhibit minima when particle radii are of the order of 1μm to 5μm. The existence of such minimum collection efficiencies for flux deposition has been reported in the literature [16,39]. Figure 7.24 illustrates the dependence of collection effiency on particle size for a droplet with a Reynolds number of 30 in a gas stream at various temperatures. The minimum collection efficiency at each temperature can be clearly identified. It is evident that there is a shift to smaller particle sizes as the temperature decreases.

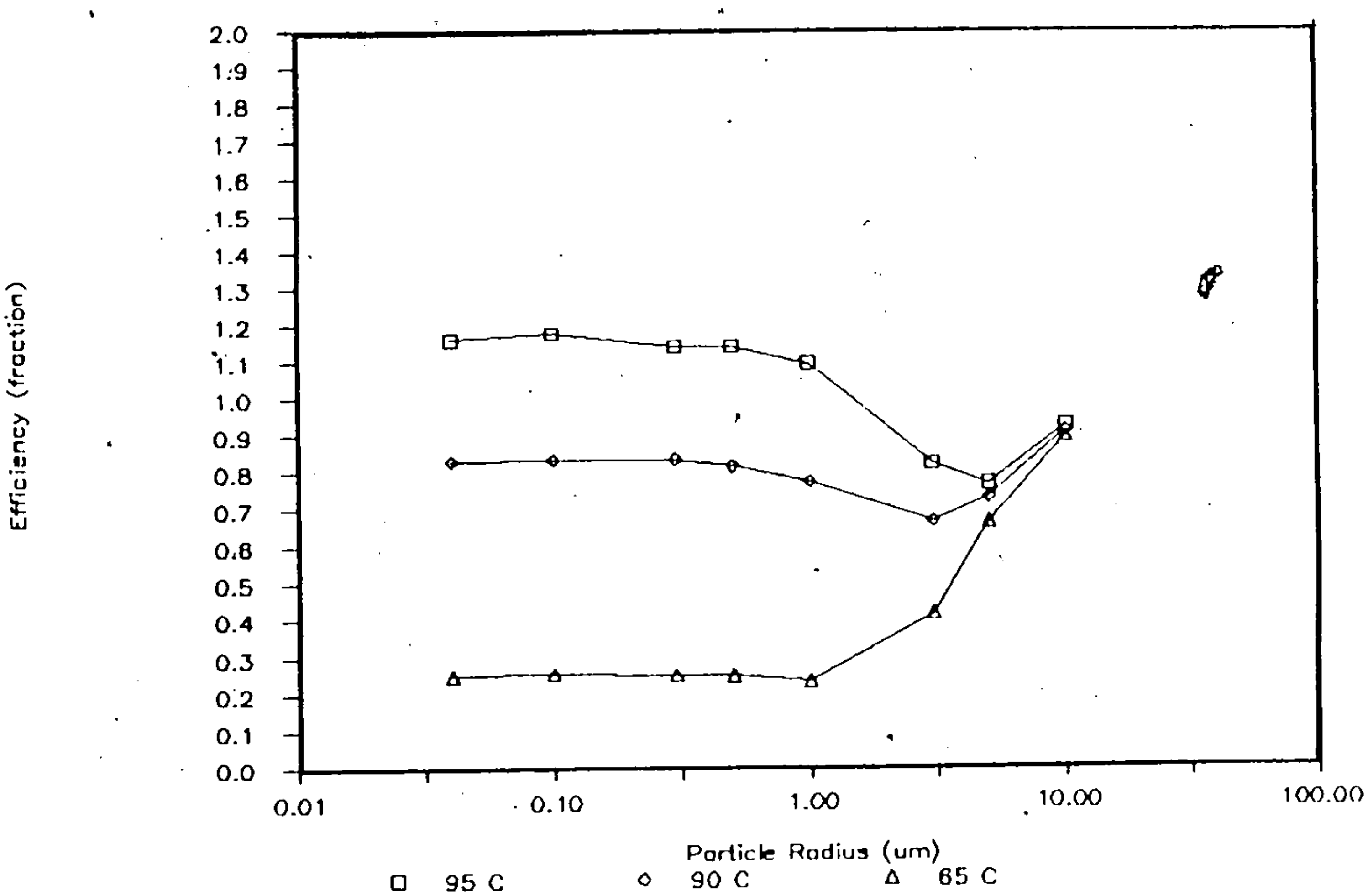

FIGURE 7.24: Illustrated Minimum Collection for Nre of 30

The minimum collection efficiency occurs as a result of increasing masses of particles with increasing sizes. This trend can be explained readily in terms of the illustration provided in Figure 7.25. The trajectories were computed for a 0.1μm, 5μm, and a 10μm radius particle released from the same point, 70 collector radii upstream. For a 170μm radius droplet at 10°C in a gas stream at 95°C, these particles are collected with efficiencies of 117%, 76%, and 92% respectively.

The smallest particle (0.1μm) is influenced by the flow of the gas stream most significantly of the three sizes as the fluid initially draws it up and away from the collector. However, as the particle approaches the collector its low mass offers little resistance to the strong radial flux forces. Consequently, it collides with the droplet. The net result of the particle having a low mass is a relatively high collection efficiency.

The 5μm particle, having a slightly larger mass, is not affected to the same extent as the smaller particle was by the sweeping tangential component of the gas velocity as it flows around the collector. This particle is actually brought into a region where the flux gadients are larger as indicated by its trajectory in Figure 7.25. However, its slightly larger mass offers enough resistance to acceleration in the radial direction by flux forces. This resistance allows the particle to be swept past the collector. As a result there is a decrease in collection efficiency for an increase in mass.

The largest particle (10μm) is affected very little by the

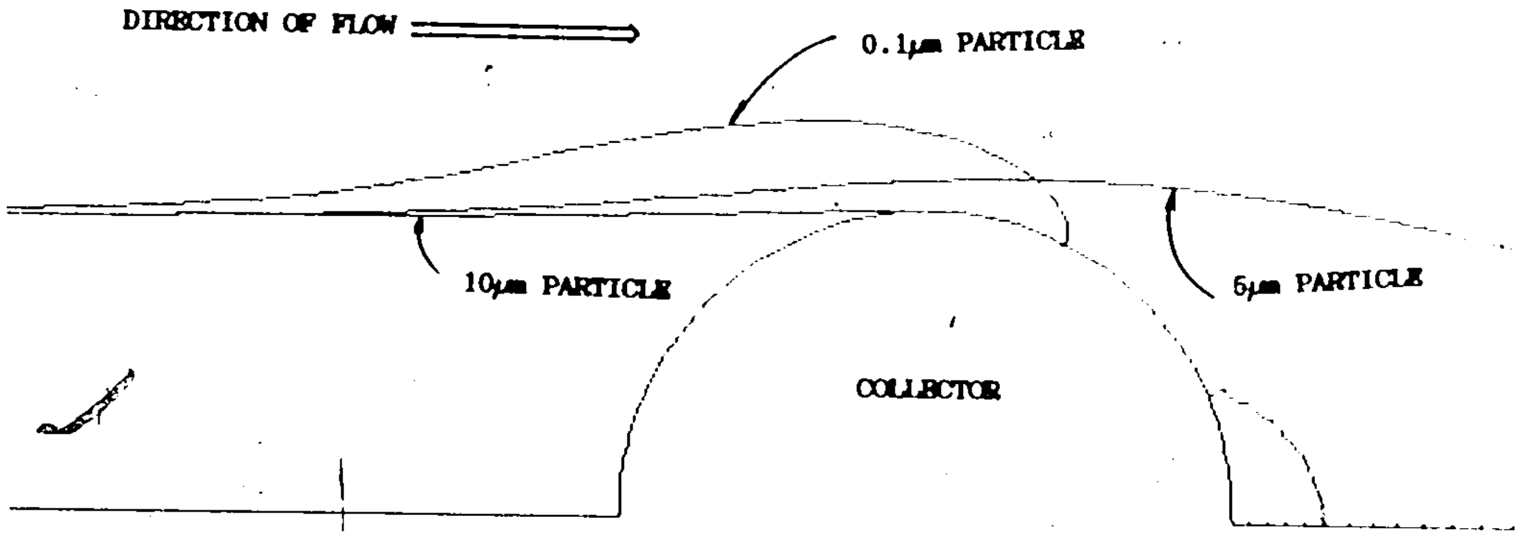

FIGURE 7.25: Trajectories of 0.1μm, 5μm, and 10μm Particles

sweeping tanjential component of the gas flow as indicated by its almost straight path in Figure 7.25. The increased inertia allows the large particle to be captured on the front of the collector. Consequently, the collection efficiency increases for larger particles.

The shift in the location of the minimum towards larger particle sizes with increasing gas temperatures is due to the increase in flux forces at the waist of the collector. This increased flux force would capture particles that were able to escape at a lower temperature. Consequently, the location of the minimum collection efficiency will shift to a larger particle size for an increase in temperature.

7.5.4 The Effect of Drop Deformation

Droplet deformation has been shown to be insignificant until the droplet radius, a_o, is approximately greater than 1800μm [41,42]. Droplets whose radii are greater than 1800μm start to form a concave depression at the front as illustrated in Figure 7.5. Strictly speaking, for such large drops the flow around the collector can no longer be considered steady because of the occurence of vortex shedding [51]. A time dependent three dimensional model would be needed to account for the fluctuations that would occur in the collection efficiency as a result of this vortex shedding. However, the benefits of such a comprehensive study, providing time dependent collection efficiencies, would be of limited practical use since only time averaged values of efficiency are required. Simplification of this complex behavior was achieved by assuming that potential flow conditions exist around the droplet. This assumption should produce values of collection efficiencies that approximate time averaged values. This approach is supported by comparisons between potential flow solutions and experimental data for high Reynolds numbers [44]. In addition, a review of the numerical solutions provided by Degani and Tardos [7] shows that fluctuations in the time dependent collection efficiencies are not very significant.

The primary collection mechanism for flows at high Reynolds numbers is inertial impaction due to the extremely small thermal/diffusive boundary layer. Figure 7.26 illustrates efficiencies calculated for water droplets whose radii are 3000μm (Nre ≃ 3600) and 4000μm (Nre ≃ 4900). Since the collection efficiency due to inertial impaction is a function of Stokes number (K) only, these calculations are compared to the potential flow solution for a spherical collector with (K) as the independent parameter.

Figure 7.26 shows that deformation of water droplets is responsible for a significant increase in the collection efficiency for large particles over the potential flow solution for a spherical collector. However, the greatest relative increase in collection efficiency is for particles smaller than 10μm in radius ($K < 2$). For example, the collection efficiency of a 20μm particle colliding with a 4000μm collector is 1.38 times higher than the potential flow value for a sphere of the same size. A 5μm particle is collected 1.68 times more efficiently by deformed collectors which is a larger relative

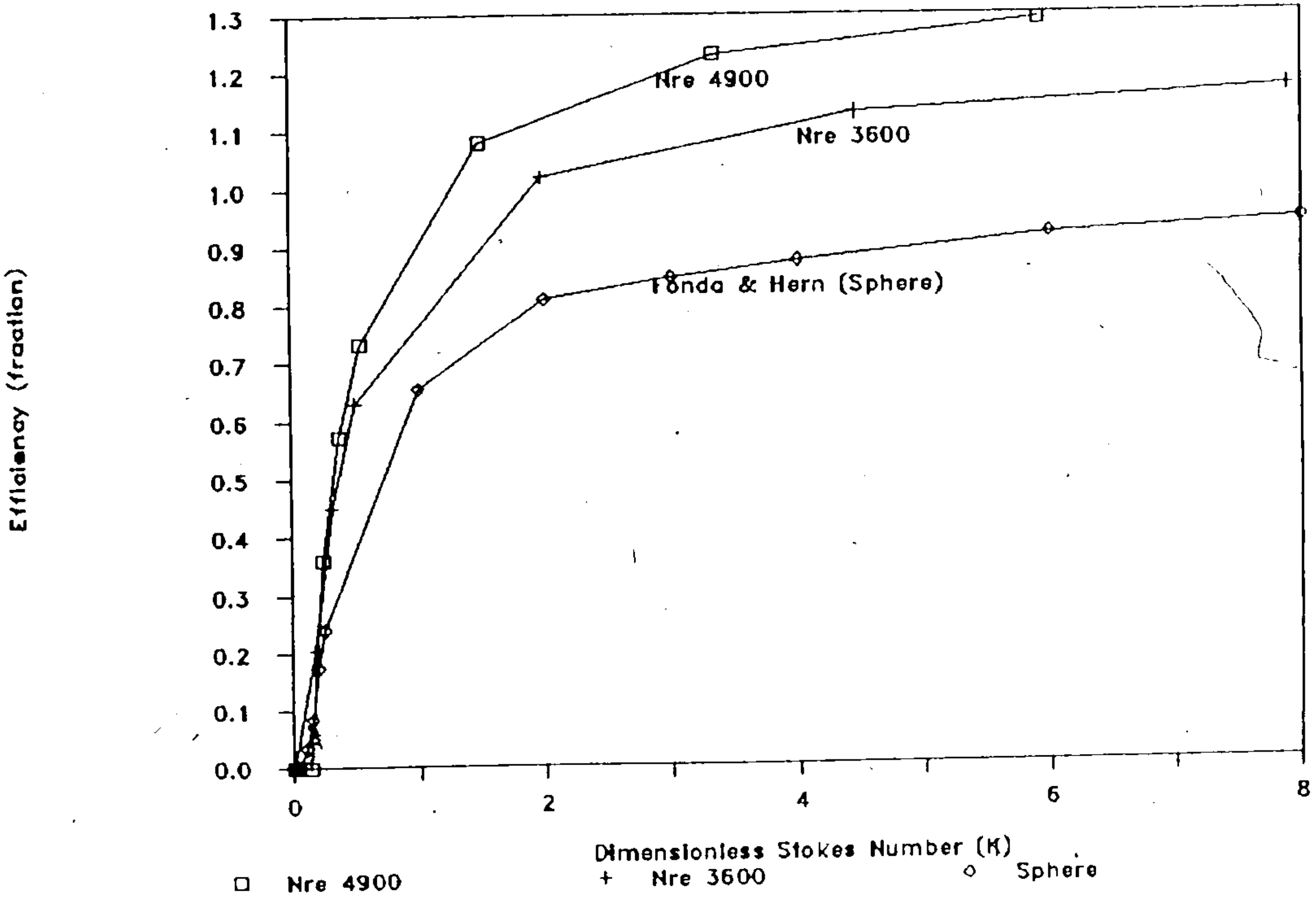

FIGURE 7.26: Collection Efficiency for Deformed Drops

increase in collection efficiency.

The explanation for this increase in efficiency is provided in Figure 7.27 which illustrates the trajectories of two particles. The larger, 20μm, particle is captured at the waist of the collector. Its increase in efficiency is due to the increased dimension of the collector at the waist. However, the smaller, 5μm, particle is collected on the protrusion at the front of the collector. Its improvement in collection efficiency is due to the increased difficulty that small particles have in escaping the blunt frontside of the collector. As illustrated by the streamlines in Figure 7.5, the fluid deviates insignificantly from a straight path until a point very close to the collector. This flow behavior indicates that there is a larger radial component of velocity (relative to the velocity around an undeformed droplet) created as the fluid flows into the depressed frontal area. This increased radial velocity reduces the time during which the tangential velocity can carry a particle past the collector. The net result is in an increase in collection efficiency.

7.6 Comparison with Existing Models

As pointed out in Section 7.4, the only data available in the literature for comparison purposes are those of Pilat and Prem [39]. Their results are useful for determining the effectiveness of complex modelling methods because of the simplicity of their model. In developing their model, Pilat and Prem assumed that:

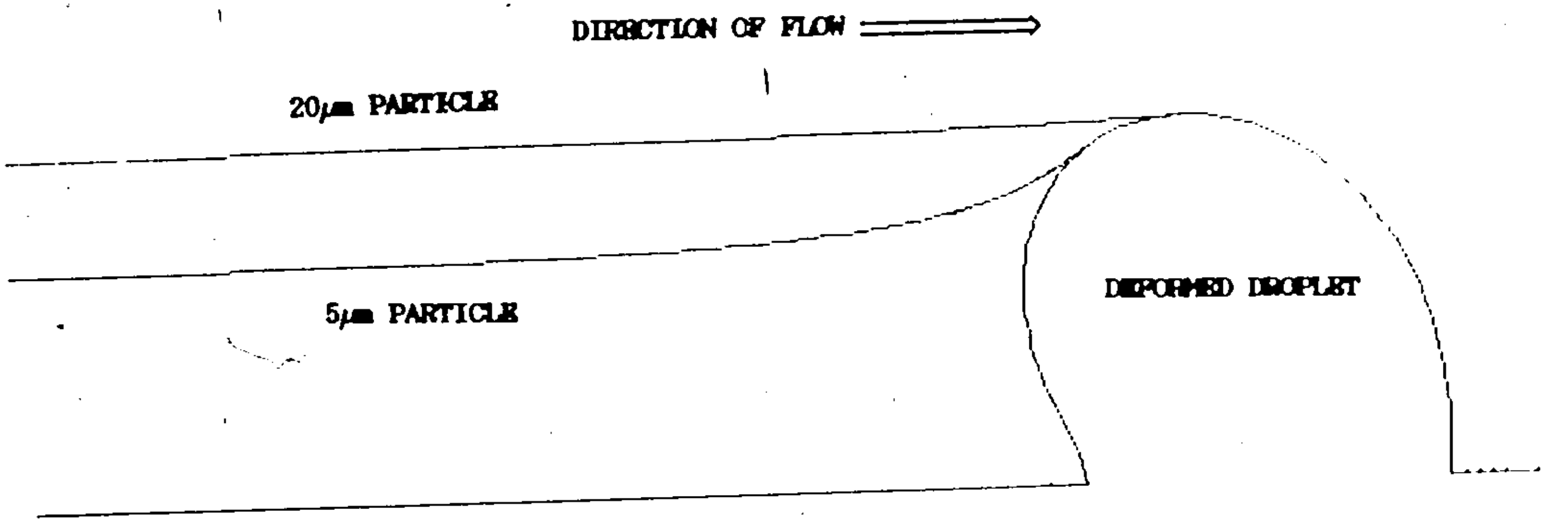

FIGURE 7.27: Particle Trajectories for a Deformed Drop

- potential flow conditions described the flow field around the collector
- inertial impaction, thermophoresis, diffusiophoresis, Brownian diffusion, and interception were responsible for the collection of particles
- the temperature, vapor, and particle distributions were linear in a thin boundary layer.

The calculations of Section 7.4 were repeated with the rigorous model proposed in this investigation. The results are shown in Figure 7.28. It is evident that Pilat and Prem generally overestimated collection efficiencies by as much as 70% for a water temperature of 10°C. There are two distinct reasons for the difference between the data of Pilat and Prem and this investigation.

The first relates to the assumption of potential flow by Pilat and Prem. They considered a collector diameter of 100μm. This size corresponds to a terminal Reynolds number of 1.54 which is much too low for potential flow conditions. Rigorous numerical determination of the flow field during this investigation (which is provided in Appendix I) indicates that it would be more realistic to assume viscous flow around the sphere. As a result of viscous flow conditions, lower collection efficiences than determined by Pilat and Prem are to be expected [3,9,25]. This decrease is due mainly to the reduction in the radial component of velocity. Since the particle is not driven to the surface as quickly, it experiences a longer contact with the tangential velocity that sweeps it away around the collector.

The second reason for the difference in the two collection

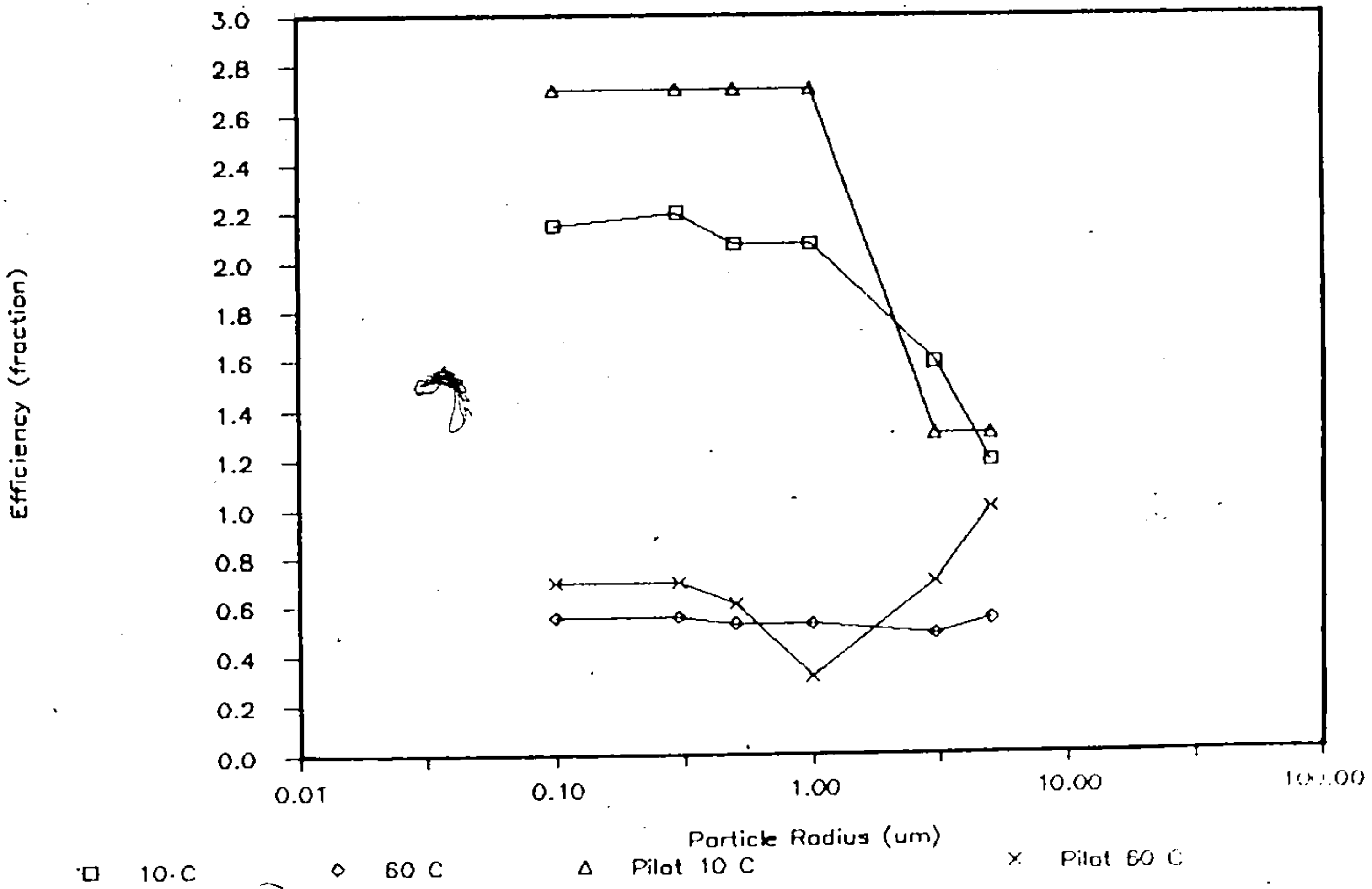

FIGURE 7.28: Rigorous Recalculation of the data of Pilat and Prem [39]

efficiencies is related to the assumed temperature/vapor distributions. Figure 7.29 compares the linear, symmetrical, profile assumed by Pilat and Prem with the numerically generated profile at the waist of the collector. The validity of the Pilat and Prem profile is very restricted because:

- the flux gradients are overestimated in the thin boundary layer
- the flux gradients do not exist outside the thin boundary layer

assumed to associated with the collector.

Overestimation of the flux gradients leads to an overestimation of particle collection. The exaggerated flux forces in the thin boundary layer, assumed by Pilat and Prem, would exert an overestimated flux force on a particle in this region. Consequently, a particle would be more easily captured relative to a particle that was influenced by the actual temperature/vapor distribution. Overestimation of flux gradients appears to be responsible for the differences in the data compared in Figure 7.28.

On the other hand, absence of flux gradients outside the boundary layer would suggest that the model of Pilat and Prem would underestimate flux deposition. As shown in Figure 7.29, the more acurate temperature/vapor distribution of the present model would predict flux forces outside the boundary layer assumed by Pilat and Prem. As a result, a particle could be collected from outside the

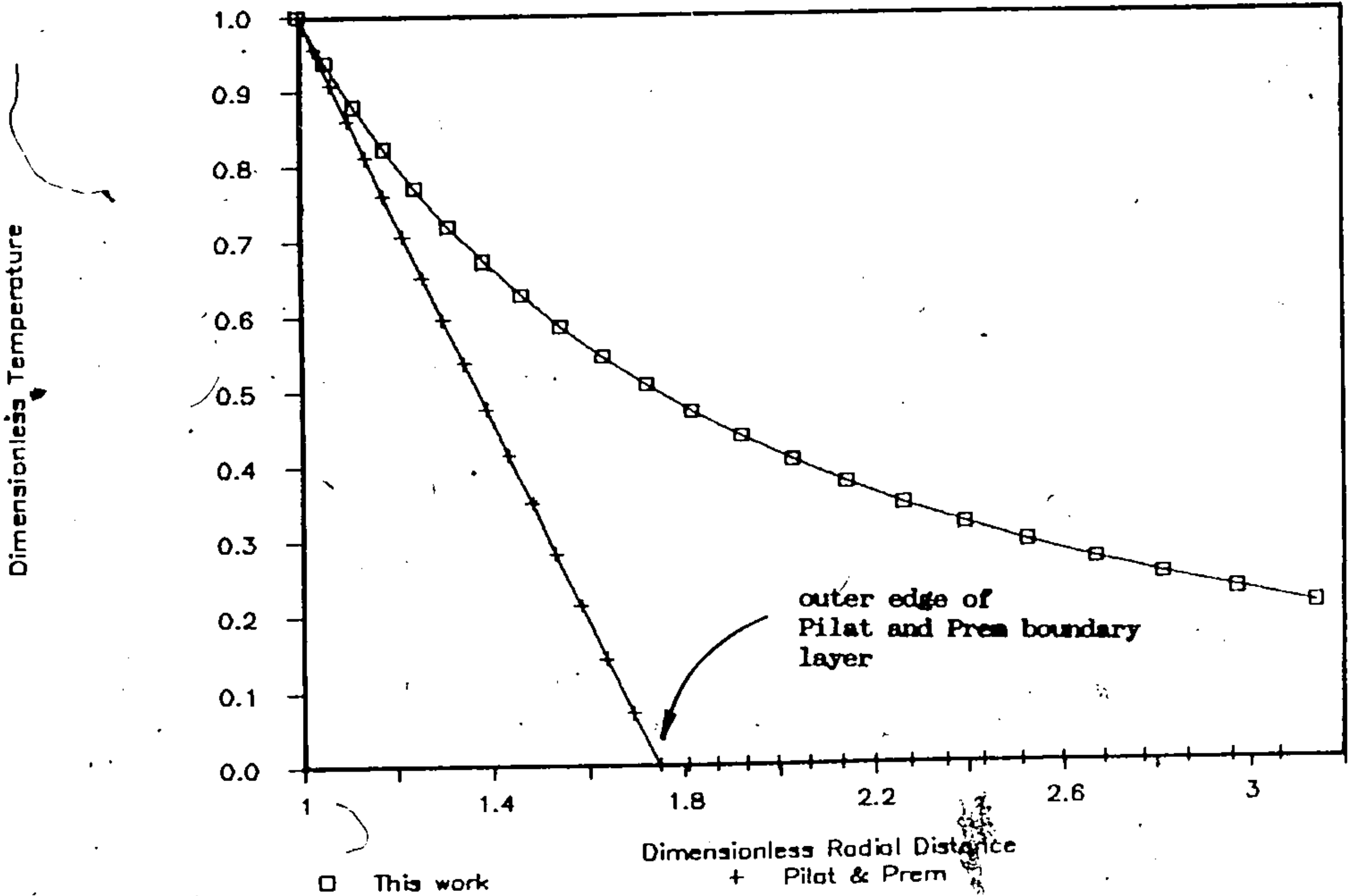

FIGURE 7.29: Low Reynolds Number Comparison of the Present Temperature Profile with the Simplified Profile of Pilat and Prem [39]

boundary layer if there was a high enough temperature difference between the droplet and the gas. Due to the absence of flux forces, it was impossible for collection to occur from the region outside the boundary layer in the Pilat and Prem study. Consequently, for the collector size studied by Pilat and Prem, the maximum possible collection efficiency, under any circumstance, by flux forces is essentially:

$$E = 1.7^2 \times 100 = 296\%$$

This value of 1.7 is nothing more than the dimensionless distance from the centre of the collector to the outer edge of the boundary layer as illustrated in Figure 7.29.

A specific example of a situation where the model of Pilat and Prem would underestimate collection efficiency severely is provided in Figure 7.11. The illustration provides data for the most extreme conditions of this investigation. For a gas temperature of 95°C, very fine particulate matter is collected theoretically with an efficiency close to 1600%. The model of Pilat and Prem would predict only 296%.

The high value predicted by the present model is related directly to the effects of diffusiophoresis. High vapor gradients are generated by the exponential dependence of gas saturation vapor pressure on gas temperature according to Equation 7.5.2. Clearly, this drop size would not maintain its collection efficiency for very long because the extreme conditions would cause it to increase in size and change in temperature. It should be reemphasized that the changing conditions are not a deficiency in this or any steady state model. The changes can be modelled as a series of psuedo steady state conditions along the length of the counter current scrubber.

The equations used by Pilat and Prem to predict the temperature distribution illustrated in Figure 7.29 was [39]:

$$\Delta X_h = \frac{2 \cdot a_o}{(2.+0.557 \cdot Nre^{0.5} Npr^{0.375})} \qquad 7.6.1$$

where

ΔX_h = is the effective film thickness due to heat transfer

However, Equation 7.6.1, as pointed out by Johnstone and Roberts [22], was developed originally for predicting overall heat transfer rates from spheres and not necessarily for evaluating temperature distributions. At low Reynolds numbers, when there is no well defined thermal boundary layer, Equation 7.6.1 predicts the correct temperature distribution only near the surface of the collector. The numerical model developed in this study would be needed for the evaluation of temperature gradients well removed from the sphere surface. Figure 7.29 shows there is excellent agreement between temperature gradients calculated using Equation 7.6.1 and the numerical procedure for the region vary close to the collector surface. These well developed boundary layers exist at higher Reynolds numbers.

Since well developed boundary layers are expected to occur at higher Reynolds numbers where temperature variations in the bulk of the fluid are small [4], Equation 7.6.1 should be applicable to the prediction of temperature gradients over a greater distance than at lower Reynolds numbers. This capability is demonstrated in Figure 7.30 which provides a comparison of the dimensionless temperature distributions, η, at the waist of the collector with a Reynolds number of 300 as predicted by the rigous method of this investigation and Equation 7.6.1. The data indicate that the model of Pilat and Prem would predict the collection of fine particles accurately at high Reynolds numbers. Unfortunately, at high Reynolds numbers the flux deposition of fine particulate matter becomes less important.

In short, the model of Pilat and Prem is applicable only in the high Reynolds number region where fine particle collection becomes relatively unimportant. For low Reynolds numbers more realistic determinations of velocity and temperature/vapor distributions are required.

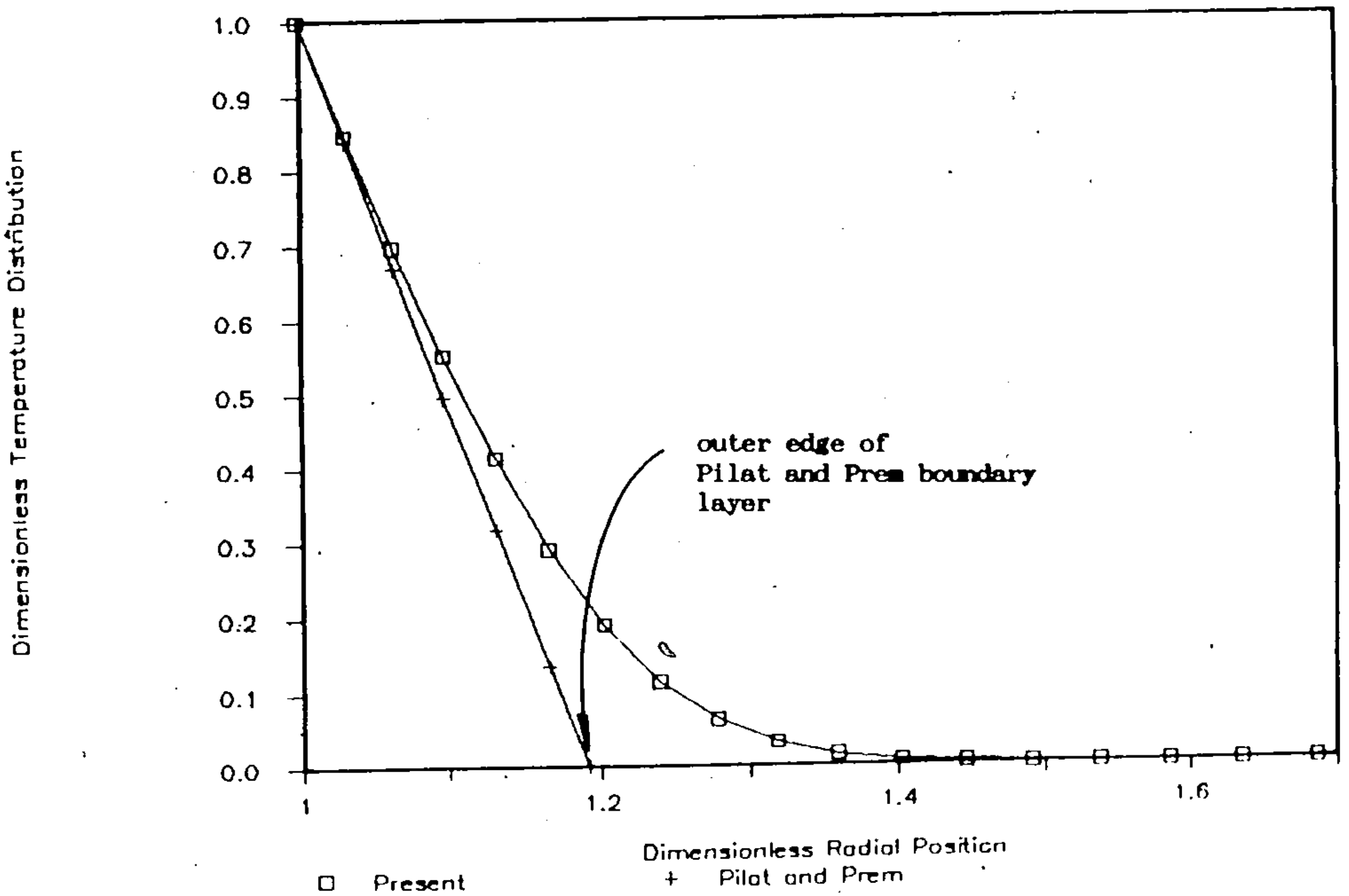

FIGURE 7.30: High Reynolds Number Comparison of the Present Temperature Profile with the Simplified Profile of Pilat and Prem [39]

8. CONCLUSIONS

The generalized model developed in this study has been shown to be accurate through a step-by-step calibration procedure. The numerical methods applied produce a quick reliable code which can be applied to microcomputer modelling since all calculations where performed on an IBM AT system with a math coprocessor. This configuation is used in industry quite widely.

The general results of this investigation indicate that flux forces can increase the collection of fine particulate material dramatically with increases in temperature of a saturated gas. Increases in Reynolds number decrease collection of fine particles significantly. The constancy in the collection efficiency of small particles by flux forces indicates that flux deposition can be related to Reynolds number through the following proportionality:

$$E_{flux} \alpha \; Nre^{-0.78}$$

Wake capture of fine particles was also taken into account by numerical simulation of the flow field of the collector. Although definite mechanisms for collision on the rear of the collector and capture in the wake itself were established, wake capture appears to be relatively unimportant as a collection mechanism for the operating conditions assumed in this investigation.

Generally, minimum collection efficiencies were observed over the entire range of temperatures and Reynolds numbers considered. These minima were attributed to the influence of particle masses on the relative magnitudes of the inertial force and the flux forces near the collector.

Estimation of the effect of drop deformation was obtained by numerically generating the potential flow field around a high Reynolds number collector. Analysis shows that collection of micron sized particles can be significantly enhanced. Increased collection of large particles ($r_p > 10\mu m$) is simply due to the increased dimension of the droplet at the waist. However, increased collection of smaller particles ($r_p < 10\mu m$) can be explained in terms of the difficulty they have in escaping from the concave depression at the front of the droplet.

The only model available in the literature for the assumed operating conditions is that of Pilat and Prem [39]. This model was developed using assumptions that are too restrictive for general considerations. Overestimations of collection efficiencies were observed at low Reynolds numbers and low temperature differences. They were attributed to:

- overestimation of hydrodynamic forces on the particle from the assumption of potential flow
- overestimation of flux gradients near the collector due to the assumption of a thin thermal and diffusive boundary layer

The underestimations by the Pilat and Prem model at low Reynolds numbers and high temperature differences were attributed to:

- underestimation of flux gradients well removed from the collector due to the assumption of a thin thermal and diffusive boundary layer

8.1 Recommendations for Future Study

This investigation has demonstrated that there is a need for the implementation of more complex modelling procedures if single droplet collection efficiencies are to be predicted accurately. Several deficiencies have been found in the literature pertaining to the collection of submicron and micron sized particles.

Previous evaluations of overall countercurrent scrubber performance characteristics were based on simplified single drop collection efficiency models such the one developed by Mehta [31]. Due to the inaccuracies that can be associated with simplified models, a rigorous reevaluation of optimum overall performance parameters is essential with emphasis focussed on fine particle collection.

In this investigation collection of particles smaller than 0.05μm in radius was not considered. Brownian motion is a dominant collection mechanism for particles of this size. Present models appear to consider the effects of Brownian motion to be limited to a thin boundary layer in which particle distributions are linear [31,39]. However, it has been shown that the distribution of particles is governed by [12]:

$$\frac{D\eta}{Dt} = \mathcal{D}_p \nabla^2 \eta \qquad 8.1.1$$

where

$\mathcal{D}_p$ = the particle diffusion coefficient

This convective diffusion equation can be solved readily by the methods discussed in this report. Consequently, accurate determination of the collection of extremely fine massless particles can be performed. However, since $\mathcal{D}_p$ is a function of particle size, a general model would be cumbersome, therefore future work should focus on the development of accurate design curves.

Fine particle collection by turbulent diffusion is known to occur in high speed flows [29,47], but very little data appear in the literature. Greenfield [15] is one of the few to have studied this mechanism. Unfortunately, as pointed out by Grover et. al. [16], the method for accounting for the effects of turbulence was very crude. The lack of sophistication is due to the relatively early time period when the investigation took place. It is only recently that reliable turbulent models have been developed [38]. Future studies should involve the application of much improved models, such as the (κ,ϵ) model or the Large-Eddy simulation model [38], to fine particle collection.

The Navier-Stokes model developed in this study was applied to essentially spherical collectors because drop deformation does not occur until the flow at the front of the collector is nearly potential in nature. It is still advantageous to use numerical grid generation for geometries that have analytical representations, but the code will be underutilized. The phenomenon of particulate film formation on cylindrical elements in filters is known to deform the collector significantly and cause an increase in collection efficiency [13]. When materials form a semipermeable film on the collector consideration must be given to flow through a porous medium. The code developed during this study can be applied readily to flows around deformed cylinders as discussed in Chapter 6. LeClair [26] has provided the theory for modelling flow through porous media at intermediate Reynolds numbers. It can be applied to the case of a semipermiable film on a cylinder using a modified version of the Navier-Stokes code developed in this study.

The final recommendation concerns the applicability of a generalized model to every day use. The rigorous model of this study was developed on a microcomputer. A combination of a number of techniques; such as the hybrid first-second order upwinded differencing scheme, the variable relaxation factor, and block relaxation; were applied to produce a quick and reliable code. The basis of all of the proceedures was the general orthogonal mapping strategy. However, other techniques that offer all the advantages of the ones used in this study are available. One of the most interesting schemes is the 'true' spectral method which is becoming more popular for fluid flow problems. It has been reported that this method is generally 10 to 30 times faster than the finite difference techniques or finite element methods when considering stream fucntion/vorticity function formulation of the Navier-Stokes equations [38]. Predictions with the rigorous particle collection model could benefit significantly from the apparent increase in speed that the 'true' spectral method provides.

References

1. Alias, E. I., An Evaluation of Wet Collector Performance for Particulate Removal Part III: Particle Collection by Thermophoresis and Diffusiophoresis, M.A.Sc. Thesis, University of Windsor, Windsor, Ontario, (1976).

2. Annis, B. K., and E. A. Mason, Theory of Thermophoresis of Sperical Particles, Journal of Aerosol Science, 6, pp. 105-117, (1975).

3. Beard, K. V., and S. N. Grover, Numerical Collision Efficiences for Small Raindrops Colliding with Micron Size Particles, Journal of the Atmospheric Sciences, 31, pp. 543-550, (Mar. 1974).

4. Bird, R. B., W. E. Stewart, and E. N. Lightfoot, Transport Phenomena, John Wiley & Sons, New York, New York, (1960).

5. Brock, J. R., The Thermal Force in the Transition Region, Journal of Colloid and Interface Science, 23, p. 448, (1967).

6. Burden, R. L., J. D. Faires, and A. C. Reynolds, Numerical Analysis, 2nd ed., PWS Publishers, Boston, Massachusetts, (1981).

7. Degani, D. D., and G. I. Tardos, Inertial Deposition of Small Particles on a Sphere at Intermediate and High Reynolds Numbers: A Time Dependent Study, Journal of the Air Pollution Control Association, 31, 9, (Sept. 1981).

8. Ehrlich, L. W., An Ad Hoc SOR Method, Journal of Computational Physics, 44, pp. 31-45, (1981).

9. Ellwood, K., A. W. Gnyp, C. C. St. Pierre, and S. Viswanathan, Development of Improved Single Drop Collection Efficiency Correlations for Microcomputer Modeling of Venturi Scrubber Performance, Proceedings: Sixth Symposium on the Transfer and Utilization of Particulate Control Technology, 1, (Nov. 1986).

10. Epstein, N., and J. H. Masliyah, Numerical Study of Steady Flow Past Spheroids, Journal of Fluid Mechanics, 44, Part 3, pp. 493-512, (1970).

11. Fonda, A., and H. Herne, The Classical Computation of the Aerodynamic Capture of Particles by Spheres, International Journal of Air Pollution, 3, pp. 572-573, (1960).

12. Friedlander, S. K., Particle Diffusion in Low-Speed Flows, Journal of Colloid and Interface Science, 23, pp. 157-164, (1967).

13. Fuchs, N. A., The Mechanics of Aerosols, Pergamon-MacMillan, New York, New York, (1964).

14. Ganguli, A., An Evaluation of Wet Collector Performance for Particulate Removal Part IV: Analysis of Particulate Collection by Spherical Obstacles, M.A.Sc. Thesis, University of Windsor, Windsor, Ontario, (1976).

15. Greenfield, S. M., Rain Scavenging of Radioactive Particulate Matter from the Atmosphere, Journal of Meteorology, 14, pp. 115-125, (1957).

16. Grover, S. N., H. R. Pruppacher, and A. E. Hamielec, A Numerical Determination of the Efficiency with Which Spherical Aerosol Particles Collide with Spherical Water Drops Due to Inertial Impaction And Phoretic and Electrical Forces, Journal of the Atmospheric Sciences, 34, pp. 1655-1663, (Oct. 1977).

17. Hamielec, A. E., T. W. Hoffman, and L. L. Ross, Numerical Solution Of the Navier-Stokes Equation for Flow Past Spheres: Part 1, A.I.Ch.E. Journal, 13, 2, pp. 212-219, (Mar. 1967).

18. Hamielec, A. E., and A. I. Johnson, Viscous Flow Around Spheres at Intermediate Reynolds Numbers, Canadian Journal of Chemical Engineering, 40, pp. 41-45, (1962).

19. Haussling, H. J., and R. M. Coleman, A Method for Generating Orthogonal and Nearly Orthogonal Boundary-Fitted Coordinate Systems, Journal of Computational Physics, 43, pp. 373-381, (1981).

20. Hung, T., and T. D. Brown, An Implicit Finite-Difference Method for Solving the Navier-Stokes Equation Using Curvilinear Coordinates, Journal of Computational Physics, 23, pp. 343-363, (1977)..

21. Jenson, V. G., Viscous Flow Round a Sphere at Low Reynolds Numbers (<40), Proc. Roy. Soc., 249A, 497, pp.346-366, (1952).

22. Johnstone, H. F., and W. H. Roberts, Deposition of Aerosol Particles from Moving Gas Streams, Industrial and Engineering Chemistry, 41, pp. 2417-2423, (1949).

23. Lanczos, C., The Variational Principles of Mechanics, 4th ed., Dover Publications, New York, New York, (1970).

24. Langmuir, I., The Production of Rain by a Chain Reaction in Cumulus Clouds at Temperatures Above Freezing, Journal of Meteorology, 5, 5, pp. 175-192, (1948).

25. Langmuir, I., and K. B. Blodgett, A Mathematical Investigation of Water Droplet Trajectories, U.S. Army Air Forces, Tech. Rept. No. 5418.40, (1946).

26. LeClair, B. P., Viscous Flow in Multiparticle Systems at Intermediate Reynolds Numbers, Ph. D. Dissertation, McMaster University, Hamilton, Ontario, (1970).

27. LeClair, B. P., A. E. Hamielec, and H. R. Pruppacher, A Numerical Study of the Drag on a Sphere at Low and Intermediate Reynolds Numbers, Journal of the Atmospheric Sciences, 27, pp. 308-315, (March 1970).

28. LeClair, B. P., A. E. Hamielec, H. R. Pruppacher, and W. D. Hall, A Theoretical and Experimental Study of the Internal Circulation in Water Drops Falling at Terminal Velocity in Air, Journal of the Atmospheric Sciences, 29, pp. 728-740, (May 1972).

29. Leong, K. H., K. V. Beard, and H. T. Ochs, Laboratory Measurements of Particle Capture by Evaporating Cloud Drops, Journal of the Atmospheric Sciences, 39, pp. 1130-1140, (May 1982).

30. Lin, C. L., and S. C. Lee, Collision Efficiency of Water Drops in the Atmosphere, Journal of the Atmospheric Sciences, 32, pp. 1412-1418, (July 1975).

31. Mehta, J. V., A Computer Simulation of Countercurrent Flow Spray Scrubbers, M.A.Sc. Thesis, University of Windsor, Windsor, Ontario, (1977).

32. Micheal, D. H., and P. W. Norey, Particle Collision Efficiences for a Sphere, Journal of Fluid Mechanics, 37, pp. 565-575, (1969).

33. Mobley, C. D., and R. J. Stewart, Note: On the Numerical Generation of Boundary-Fitted Orthogonal Curvilinear Coordinate Systems, Journal of Computational Physics, 34, pp. 124-135, (1980).

34. Oliver, D. L. R., and J. N. Chung, Steady Flows Inside and Around a Fluid Sphere at Low Reynolds Numbers, Journal of Fluid Mechanics, 154, pp. 215-230, (1985).

35. Ortega, J. M., and W. C. Rheinbolt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, New York, (1970).

36. Pallady, P. H., and P. J. Henley, Evaluating Moist-Air Properties: Relative Humidity by Direct Calculation, Chemical Engineering, 91, 22, pg. 117, (Oct. 1984).

37. Pemberton, C. S., Scavenging Action of Rain on Nonwettable Particulate Matter Suspended in Atmosphere, International Journal of Air Pollution, 3, pp. 168-178, (1960).

38. Peyret, R., and T. D. Taylor, Computational Methods for Fluid Flow, Springer-Verlag, New York, New York, (1983).

39. Pilat, M. J., and A. Prem, Calculated Particle Collection Efficiencies of Single Droplets Including Inertial Impaction, Brownian Diffusion, Diffusiophoresis, and Thermophoresis, Atmospheric Environment, 10, pp. 13-19, (1976).

40. Pope, S. B., The Calculation of Turbulent Recirculating Flows in General Orthogonal Coordinates, Journal of Computational Physics, 26, pp. 197-217, (1978).

41. Pruppacher, H. R., and K. V., Beard, A Wind Tunnel Investigation of the Internal Circulation and Shape of Drops Falling at Terminal velocity in Air, Quart. J. R. Met. Soc., 96, pp 247-256, (1970),

42. Pruppacher, H. R., and R. L. Pitter, A Semi-Empirical Determination of the Shape of Cloud and Rain Drops, 28, pp. 86-94, (Jan. 1971).

43. Ranz, W. E., On Sprays and Spraying, Department of Chemical Engineering, The Pennsylvania State University, University Park, Pennsylvania, (1956).

44. Ranz, W. E., and J. B. Wong, Impaction of Dust and Smoke Particles on Surface and Body Collectors, Industrial and Engineering Chemistry, 44, pp. 1371-1381, (1952).

45. Ryskin, G., and L. G. Leal, Orthogonal Mapping, Journal of Computational Physics, 50, pp. 71-100, (1983).

46. Sokolnikoff, I. S., Tensor Analysis, 2nd ed., John Wiley & Sons, New York, New York, (1964).

47. Sparks, L. E., The Effect of Scrubber Operating and Design Parameters on the Collection of Particulate Air Pollutants, Ph. D. Dissertation, University of Washington, (1971).

48. Tardos, G. I., N. Abuaf, and C. Gutfinger, Dust Deposition in Granular Bed Filters: Theory and Experiments, Journal of the Air Pollution Control Association, 28, 4, pp. 354-362, (Apr. 1978).

49. Thompson, J. F., Z. U. A. Warsi, and C. W. Mastin, Numerical Grid Generation, North-Holland, New York, New York, (1985).

50. Waldmann, L., and K. H. Schmitt, Thermophoresis and Diffusiophoresis of Aerosols, Aerosol Science, Edited by C. N. Davies, Academic Press, London, (1966).

51. Woo, S. E., Simultaneous Free and Forced Convection Around Submerged Cylinders and Spheres, Ph. D. Dissertation, McMaster University, Hamilton, Ontario, (1971).

52. Woo, S., E., and A. E., Hamielec, A Numerical Method of Determining the Rate of Evaporation of Small Water Drops Falling at Terminal Velocity in Air, Journal of the Atmospheric Sciences, 28, pp. (1448-1454).

53. Wu, M., Scavenging of Atmospheric Particles by Water Drops, Ph. D. Dissertation, University of Missouri-Rolla, (1973).

NOMENCLATURE

$\bar{a}$	particle acceleration [dimensionless]
$\bar{A}$	particle acceleration [m/s^2]
a_o	volume radius of the collector [m]
C_f	Cunningham correction factor [dimensionless]
C_p	heat capacity of the fluid [J/kg·°C]
C_1	molar concentration of water vapor [$mole/m^3$]
$C_{1\infty}$	molar concentration of water vapor in the undisturbed gas stream [$mole/m^3$]
C_{1c}	molar concentration of water vapor on the surface of the collector [$mole/m^3$]
$\mathcal{D}_{12}$	diffusivity of water in air [m^2/s]
$\bar{e}_i$	unit vector in the direction of α_i increasing
E	droplet collection efficiency [dimensionless]
$\bar{F}_d$	diffusiophoretic force [$kg \cdot m/s^2$]
$\bar{F}_{drag}$	drag force [$kg \cdot m/s^2$]
$\bar{f}_e$	external force [dimensionless]
$\bar{F}_e$	external force [$kg \cdot m/s^2$]

$\bar{F}_t$	thermophoretic force [kg·m/s^2]
g_{ij}	transformation metric tensor element
h_i	grid scale factor
K	Stokes number [dimensionless]
k_f	thermal conductivity of the fluid [W/m·°C]
k_p	thermal conductivity of the particle [W/m·°C]
L	vortex length [m]
M_1	molecular weight of water [kg/mole]
M_2	molecular weight of air [kg/mole]
Nkn	Knudsen number [dimensionless]
Npe_h	Peclet number for heat transfer [dimensionless]
Npe_m	Peclet number for mass transfer [dimensionless]
Npr	Prandtl number [dimensionless]
Nr	interception parameter [dimensionless]
Nre	Reynolds number [dimensionless]
Nsc	Schmidt number [dimensionless]
Nsh_ℓ	Local Sherwood number [dimensionless]
Nu_ℓ	Local Nusselt number [dimensionless]
$\bar{r}$	position vector [dimensionless]
r_p	particle radius [m]
t	time [dimensionless]
T	time [s]

$\underline{T}$	iteration matrix
T_f	fluid temperature [°K]
$\bar{u}$	fluid velocity [dimensionless]
$\bar{U}$	fluid velocity [m/s]
$\bar{v}$	particle velocity [dimensionless]
$\bar{V}$	particle velocity [m/s]
$\bar{V}_d$	particle velocity due to diffusiophoresis [m/s]
$\bar{V}_t$	particle velocity due to thermophoresis [m/s]

Greek letters

α_i	transformed boundary conforming coordinates
β	dampening factor [dimensionless]
η	temperature or vapor distribution [dimensionless]
ρ_f	density of fluid [kg/m^3]
ρ_p	density of particle [kg/m^3]
γ_1	mole fraction of water vapor [dimensionless]
γ_2	mole fraction of air [dimensionless]
λ	mean free path length of the fluid molecules [m]
μ_f	viscosity of fluid [kg/m·s]
ω	acceleration factor [dimensionless]

ψ	stream function [dimensionless]
ζ	vorticity function [dimensionless]
θ_s	angle of flow separation [degrees]

Notation

Scalar quantities appear as normal faced characters (no bold) such as the fluid viscosity μ_f.

Vector quantities appear as bold faced characters with a bar over the top such as the particle velocity $\bar{\mathbf{v}}$.

Matrices appear as bold faced characters with a bar below such as the iteration matrix $\underline{\mathbf{T}}$.

Appendix A

General Orthogonal Differential Operators

This appendix describes the differential operators in generalized orthogonal coordinates required to transform partial differential equations from the physical plane to the computational plane.

Basic Definitions

f = a scalar function in the physical domain

$\bar{F} = F_1\bar{e}_1+F_2\bar{e}_2+F_3\bar{e}_3$ is a vector function in the physical domain. Derivatives of this function will be given in terms of the components of $\bar{F}$ in the direction of α_i increasing as opposed to the $\bar{i}$, $\bar{j}$, and $\bar{k}$ directions in the physical plane.

$$h_i = \left[\left[\frac{\partial x}{\partial\alpha_i}\right]^2+\left[\frac{\partial y}{\partial\alpha_i}\right]^2+\left[\frac{\partial z}{\partial\alpha_i}\right]^2\right]^{\frac{1}{2}} \quad i=1,2,3$$

are scale factors between the physical plane and the computational plane.

The Transformed Operators

The gradient of f

$$\bar{\nabla}f = \frac{\bar{e}_1}{h_1}\cdot\frac{\partial f}{\partial\alpha_1}+\frac{\bar{e}_2}{h_2}\cdot\frac{\partial f}{\partial\alpha_2}+\frac{\bar{e}_3}{h_3}\cdot\frac{\partial f}{\partial\alpha_3} \qquad \text{A.1}$$

The divergence of $\bar{F}$

$$\bar{\nabla}\cdot\bar{F} = \frac{1}{h_1h_2h_3}\left[\frac{\partial}{\partial\alpha_1}(h_2h_3F_1)+\frac{\partial}{\partial\alpha_2}(h_1h_3F_2)+\frac{\partial}{\partial\alpha_3}(h_1h_2F_3)\right] \qquad \text{A.2}$$

The curl of $\bar{F}$

$$\bar{\nabla}\times\bar{F} = \frac{1}{h_1h_2h_3}\begin{vmatrix} h_1\bar{e}_1 & h_2\bar{e}_2 & h_3\bar{e}_3 \\ \dfrac{\partial}{\partial\alpha_1} & \dfrac{\partial}{\partial\alpha_2} & \dfrac{\partial}{\partial\alpha_3} \\ h_1F_1 & h_2F_2 & h_3F_3 \end{vmatrix} \qquad \text{A.3}$$

The laplacian of f

$$\nabla^2 f = \frac{1}{h_1h_2h_3}\left[\frac{\partial}{\partial\alpha_1}\left[\frac{h_2h_3}{h_1}\cdot\frac{\partial f}{\partial\alpha_1}\right]+\frac{\partial}{\partial\alpha_2}\left[\frac{h_1h_3}{h_2}\cdot\frac{\partial f}{\partial\alpha_2}\right]+\frac{\partial}{\partial\alpha_3}\left[\frac{h_1h_2}{h_3}\cdot\frac{\partial f}{\partial\alpha_3}\right]\right] \qquad \text{A.4}$$

An additional identity that is useful when using the vorticity transport equation (4.3.6) under the assumption of axisymmetric or planar flow conditions is:

The curl(curl($f\bar{e}_3$))

$$\bar{\nabla}\times\bar{\nabla}\times(f\bar{e}_3) = -\frac{\bar{e}_3}{h_3}\cdot D^2(h_3 f) \qquad \text{A.5}$$

where $$D^2 = \frac{h_3}{h_1h_2}\left[\frac{\partial}{\partial\alpha_1}\left[\frac{h_2}{h_1h_3}\cdot\frac{\partial}{\partial\alpha_1}\right]+\frac{\partial}{\partial\alpha_2}\left[\frac{h_1}{h_2h_3}\cdot\frac{\partial}{\partial\alpha_2}\right]\right]$$

Appendix B

Particle Acceleration in General Orthogonal Coordinates

In general, the contravariant acceleration of a particle in the direction of increasing α_k, a_t^k, is given by [45]:

$$a_t^k = \frac{d^2\alpha_k}{dt^2} + \Gamma^k_{pq} \cdot \frac{d\alpha_p}{dt}\frac{d\alpha_q}{dt} \qquad \text{B.1}$$

where the Christoffel symbols, Γ^k_{pq}, are given by:

$$\Gamma^k_{pq} = \sum_{r=1}^{r=3} \frac{g^{kr}}{2}\left[\frac{\partial\, g_{pr}}{\partial\alpha_q} + \frac{\partial\, g_{qr}}{\partial\alpha_p} - \frac{\partial\, g_{pq}}{\partial\alpha_r}\right] \qquad \text{B.2}$$

Note: g^{ij} and g_{ij} are the contravariant and covariant metric tensor components respectively.

If the system is orthogonal then the off diagonal elements of the transformation metric tensor are zero. Therefore:

$$g^{kr} = 0 \ , \ k \neq r$$

This condition allows Equation B.2 to be wriiten as:

$$\Gamma^k_{pq} = \frac{1}{2g_{kk}}\left[\frac{\partial\, g_{pk}}{\partial\alpha_q} + \frac{\partial\, g_{qk}}{\partial\alpha_p} - \frac{\partial\, g_{pq}}{\partial\alpha_k}\right] \qquad \text{B.3}$$

As mentioned earlier, a_t^k is the contravariant component of acceleration. The physical component of acceleration, a^k, is defined by:

$$a^k = \sqrt{g_{kk}} \cdot a_\dagger^k = h_k \cdot a_\dagger^k \qquad \text{B.4}$$

From Equation 1, the two components of contravariant acceleration are:

$$a_\dagger^1 = \frac{d^2\,\alpha_1}{dt^2} + \frac{1}{g_{11}}\left[\frac{1}{2}\cdot\frac{\partial\,g_{11}}{\partial\alpha_1}\left[\frac{d\alpha_1}{dt}\right]^2 + \frac{\partial\,g_{11}}{\partial\alpha_2}\left[\frac{d\alpha_1}{dt}\cdot\frac{d\alpha_2}{dt}\right] - \frac{1}{2}\cdot\frac{\partial\,g_{22}}{\partial\alpha_1}\left[\frac{d\alpha_2}{dt}\right]^2\right] \qquad \text{B.5}$$

$$a_\dagger^2 = \frac{d^2\,\alpha_2}{dt^2} + \frac{1}{g_{22}}\left[-\frac{1}{2}\cdot\frac{\partial\,g_{11}}{\partial\alpha_2}\left[\frac{d\alpha_1}{dt}\right]^2 + \frac{\partial\,g_{22}}{\partial\alpha_1}\left[\frac{d\alpha_1}{dt}\cdot\frac{d\alpha_2}{dt}\right] + \frac{1}{2}\cdot\frac{\partial\,g_{22}}{\partial\alpha_2}\left[\frac{d\alpha_2}{dt}\right]^2\right] \qquad \text{B.6}$$

According to Equation B.4, the two components of physical acceleration are:

$$a^1 = h_1\frac{d^2\,\alpha_1}{dt^2} + \frac{\partial\,h_1}{\partial\alpha_1}\left[\frac{d\alpha_1}{dt}\right]^2 + 2\,\frac{\partial\,h_1}{\partial\alpha_2}\left[\frac{d\alpha_1}{dt}\cdot\frac{d\alpha_2}{dt}\right] - \frac{h_2}{h_1}\cdot\frac{\partial\,h_2}{\partial\alpha_1}\left[\frac{d\alpha_2}{dt}\right]^2 \qquad \text{B.7}$$

$$a^2 = h_2\frac{d^2\,\alpha_2}{dt^2} - \frac{h_1}{h_2}\cdot\frac{\partial\,h_1}{\partial\alpha_2}\left[\frac{d\alpha_1}{dt}\right]^2 + 2\,\frac{\partial\,h_2}{\partial\alpha_1}\left[\frac{d\alpha_1}{dt}\cdot\frac{d\alpha_2}{dt}\right] + \frac{\partial\,h_2}{\partial\alpha_2}\left[\frac{d\alpha_2}{dt}\right]^2 \qquad \text{B.8}$$

As an example, if cylindrical coordinates (r,θ,z) were used the following relationships would be obtained:

$$h_1 = 1 \text{ and } h_2 = r$$

$$a^1 = \frac{d^2 r}{dt^2} - r\left[\frac{d\theta}{dt}\right]^2$$

$$a^2 = r\cdot\frac{d^2 \theta}{dt^2} + 2\cdot\frac{dr}{dt}\cdot\frac{d\theta}{dt}$$

APPENDIX C

Optimal Acceleration Factor

Optimal acceleration parameters are presented for difference equations resulting from the use of a five-point stencil. These acceleration parameters $\omega_{i,j}$ are calculated by the method proposed by Ehrlich [8] which is based on the Jacobi spectral radius ρ_J of the linearized equation. For the five-point difference equation:

$$a_1x_{i+1,j}+a_2x_{i,j+1}+a_3x_{i-1,j}+a_4x_{i,j-1}-a_ox_{i,j} = b_{i,j} \qquad \text{C.1}$$

the Jacobi spectral radius is

$$\rho_J = \frac{2}{a_o}\left[\sqrt{a_1a_3}\cdot\cos\left(\frac{\pi}{I+1}\right)+\sqrt{a_2a_4}\cdot\cos\left(\frac{\pi}{J+1}\right)\right] \qquad \text{C.2}$$

where

I = maximum number of x's in the i direction
J = maximum number of x's in the j direction

The optimal acceleration parameter is obtained by:

letting $\rho_J = \rho_r + \sqrt{-1}\cdot\rho_i$

and defining

$$A = \rho_r^2+\rho_i^2$$
$$B = \rho_r^2-\rho_i^2$$
$$a = A^2-B^2$$
$$b = A^2-B$$

$$\varpi = \frac{1}{A^4-A^2B}\Big[[3b+(a+b^2)^{1/2}a^{1/3}[(a+b^2)^{1/2}-b]^{1/3}$$
$$-[3b-(a+b^2)]a^{1/3}[(a+b^2)+b]^{1/3}$$
$$+A^2+3B^2-4A^2B\Big]$$

Then

$$\omega_{i,j} = -\frac{[\varpi-(\varpi^2+4\varpi)^{1/2}]}{2}, \qquad \text{if } A^2 > B$$

$$\omega_{i,j} = -\frac{[\varpi+(\varpi^2+4\varpi)^{1/2}]}{2}, \qquad \text{if } A^2 < B$$

It should be pointed out that it is necessary for ρ_r to be less than one in magnitude for convergence [8].

Appendix D

Orthogonal Grid Generation Algorithm

```
************************************************************************
*                                                                      *
*     THIS PROGRAM GENERATES AN ORTHOGONAL COORDINATE SYSTEM IN AN     *
*     ARBITRARY DOMAIN                                                 *
*                                                                      *
*                    BY: KEVIN ELLWOOD                                 *
*                        DEPT. OF CHEMICAL ENGINEERING                 *
*                        UNIVERSITY OF WINDSOR                         *
*                        SEPT 1986                                     *
*                                                                      *
* NOTE: THIS PROGRAM CAN BE USED TO GENERATE COORDINATES DIRECTLY      *
*       FROM A CARTESIAN COORDINATE SYSTEM OR A POLAR COORDINATE       *
*       SYSTEM. THE PROBLEM OF THE DEFORMED DROPLET WAS FORMULATED     *
*       IN POLAR COORDINATE FORM.                                      *
*                                                                      *
************************************************************************

* VARIABLES:

*          X      - MATRIX OF X AS A FUNCTION OF I,J
*          Y      - MATRIX OF Y AS A FUNCTION OF I,J
*          NI     - TOTAL NO. OF XI (I) COORDINATE LINES
*          NJ     - TOTAL NO. OF ETA (J) COORDINATE LINES
*          TOL    - COMPUTATIONAL TOLERANCE
*          IMAX   - MAX. NO. OF ALLOWABLE ITERATIONS
*          A      - SEMI MAJOR AXIS OF ELLYPSE ENCLOSING THE DROPLET
*          B      - SEMI MINOR AXIS OF ELLYPSE ENCLOSING THE DROPLET

* SUBROUINES CALLED:

*          GRID   - GENERATES THE GRID

      parameter (pi=3.141593)
      dimension x(80,80),y(80,80),xi(80),eta(80)
      character file*40
      common a,b
      write(*,*) 'Enter max. no. of iterations:'
      read(*,*) imax
      write(*,*) 'Enter total no. of i grid lines:'
      read(*,*) ni
      write(*,*) 'Enter total no. of j grid lines:'
      read(*,*) nj
      write(*,*) 'Enter tolerance:'
      read(*,*) tol
      write(*,*) 'Radius of outer boundary:'
      read(*,*) a
      b=a

* INITIALIZE THE VALUES OF THE CORNER POINTS
```

```
      x(1,1)=pi
      x(1,nj)=pi
      x(ni,1)=0.0
      x(ni,nj)=0.0
      call ibound( pi,pi,y(1,1),y(1,nj) )
      call ibound( 0.,0.,y(ni,1),y(ni,nj) )
      call grid( x,y,ni,nj,tol,er,imax,it )

* RELATE THE CARTESIAN GRID TO THE POLAR COORDINATE GRID

      do 10 i=1,ni
         do 10 j=1,nj
            tx=exp( -y(i,j) )*cos( x(i,j) )
            ty=exp( -y(i,j) )*sin( x(i,j) )
            x(i,j)=tx
            y(i,j)=ty
10    continue

* SAVE THE GRID TO THE DISK

      write(*,*) 'Enter filename to store grid:'
      read(*,20) file
      open( unit=1,status='new',file=file,form='unformatted' )
      write(1) ni,nj,((x(i,j),i=1,ni),j=1,nj),
     &               ((y(i,j),i=1,ni),j=1,nj)
      close( unit=1,status='keep' )
      stop
20    format(a40)
      end

      subroutine grid( x,y,ni,nj,tol,er,imax,it )

* THIS SUBROUITNE CONFORMALLY MAPS AN ARBITRARY DOMAIN IN
* CARTESIAN (OR POLAR) COORDINATES INTO THE ORTHOGONAL COORDINATES
* (XI,ETA) OR (I,J)

* NOTE: THE CORNER POINTS OF (X,Y) MUST BE INITIALIZED

* THE RESULTING SIMULTANEOUS ELLIPTIC P.D.E'S ARE SOLVED BY
* BLOCK SUCCESSIVE OVERRELAXATION (B.S.O.R)

* VARIABLES:

*         X       - MATRIX OF X AS A FUNCTION OF I,J
*         Y       - MATRIX OF Y AS A FUNCTION OF I,J
*         NI      - TOTAL NO. OF XI (I) COORDINATE LINES
*         NJ      - TOTAL NO. OF ETA (J) COORDINATE LINES
*         TOL     - COMPUTATIONAL TOLERANCE
*         IMAX    - MAX. NO. OF ALLOWABLE ITERATIONS
*        IT     - NO. OF ACTUAL ITERATIONS
*        W      - ACCELERATION FACTOR FOR GRID VARIABLES
*        WS     - ACCELERATION FACTOR FOR CONFORMAL MODUAL (S)
*        S      - CONFORMAL MODUAL OF THE SYSTEM

* SUBROUINES CALLED:

*        JBOUND - CALCULATES Y=A(X) ALONG J = 1 & NJ
*        IBOUND - CALCULATES X=B(Y) ALONG I = 1 & NI
*        IPACK  - CALCULATES PACKING FUNCTION DERIVATIVES IN THE
*                 I COORDINATE STRETCH
*        JPACK  - CALCULATES PACKING FUNCTION DERIVATIVES IN THE
*                 J COORDINATE STRETCH
*        BANDIT - SOLVES TRIDIAGONAL MATRIX PRODUCED IN S.L.O.R.

      parameter (pi=3.141593)
      dimension x(80,80),y(80,80),g(80),gp(80),f(80),fp(80),ww(80)
      dimension fx(80),fy(80),a(80),b(80),c(80),xi(80),eta(80)
      common /accel/ cosx,cose
      errfn(er,unew,uold)=amax1( er,abs( unew-uold )/(abs( unew )+1.) )
      nim=ni-1
      njm=nj-1
      nim2=ni-2
      njm2=nj-2

* INITIALIZE VARIABLES FOR PACKING CONTROL

      do 5 i=1,ni
         rxi=float( i )
         call ipack( rxi,f0,f1,f2,ni )
         xi(i)=f0
         fp(i)=f1
         f(i)=f2/f1
5     continue
      do 10 j=1,nj
         reta=float( j )
         call jpack( reta,g0,g1,g2,nj )
         eta(j)=g0
         gp(j)=g1
         g(j)=g2/g1
10    continue

* INITIALIZE BOUNDARY BY LINEAR INTERPOLATION

      do 15 i=2,nim
         ri1=(xi(i)-xi(1))/(xi(ni)-xi(1))
         ri2=(xi(ni)-xi(i))/(xi(ni)-xi(1))
         x(i,1)=ri1*x(ni,1)+ri2*x(1,1)
         x(i,nj)=ri1*x(ni,nj)+ri2*x(1,nj)
         call ibound( x(i,1),x(i,nj),y(i,1),y(i,nj) )
```

```
15    continue
      do 20 j=2,njm
         rj1=(eta(j)-eta(1))/(eta(nj)-eta(1))
         rj2=(eta(nj)-eta(j))/(eta(nj)-eta(1))
         y(1,j)=rj1*y(1,nj)+rj2*y(1,1)
         y(ni,j)=rj1*y(ni,nj)+rj2*y(ni,1)
         call jbound( y(1,j),y(ni,j),x(1,j),x(ni,j) )
20    continue

* USE TRANSFINITE INTERPOLATION AS AN INITIAL GUESS

      do 25 i=1,ni
         ri1=(xi(i)-xi(1))/(xi(ni)-xi(1))
         ri2=(xi(ni)-xi(i))/(xi(ni)-xi(1))
         x11=ri1*x(ni,1)+ri2*x(1,1)
         y11=ri1*y(ni,1)+ri2*y(1,1)
         x1m=ri1*x(ni,nj)+ri2*x(1,nj)
         y1m=ri1*y(ni,nj)+ri2*y(1,nj)
         dxm=x(i,nj)-x1m
         dx1=x(i,1)-x11
         dym=y(i,nj)-y1m
         dy1=y(i,1)-y11
         do 25 j=1,nj
            rj1=(eta(j)-eta(1))/(eta(nj)-eta(1))
            rj2=(eta(nj)-eta(j))/(eta(nj)-eta(1))
            x1=ri1*x(ni,j)+ri2*x(1,j)
            y1=ri1*y(ni,j)+ri2*y(1,j)
            x2=rj1*dxm+rj2*dx1
            y2=rj1*dym+rj2*dy1
            x(i,j)=x1+x2
            y(i,j)=y1+y2
25    continue

* SET VECTORS FOR CALL TO TRID

      do 30 i=2,nim
         a(i-1)=1.+f(i)/2.
         c(i-1)=1.-f(i)/2.
30    continue

* MAKE AN INITIAL GUESS FOR THE CONFORMAL MODUAL

      sum=(fp(1)*(y(1,3)-y(1,1))+fp(ni)*(y(ni,3)-y(ni,1)))/4.
      do 35 i=2,nim
         sum=sum+fp(i)*(y(i,3)-y(i,1))/2.
35    continue
      s=sum/((x(ni,2)-x(1,2))*gp(2))

* SOLVE THE ELLIPTIC SYSTEM FOR X & Y USING B.S.O.R IN I DIRECTION
```

```
      cosx=cos( pi/(float( nim2 )+1.) )
      cose=cos( pi/(float( njm2 )+1.) )
      ws=0.5
      it=0
      er=10.*tol
40    if (( er.lt.tol ).or.( it.ge.imax )) go to 45
         wmax=0.0
         wmin=2.0
         snew=0.0
         erx=0.0
         ery=0.0
         erxs=0.0
         erys=0.0
         ers=0.0
         it=it+1
         a1=s*s
         do 50 j=2,njm
            jp=j+1
            jm=j-1
            cjp=1.-g(j)/2.
            cjm=1.+g(j)/2.
            gpj=gp(j)
            do 55 i=2,nim
               im=i-1
               del=(fp(i)/gpj/s)**2
               a2=(fp(i)/gpj)**2
               b1=-a1*f(i)
               b2=-a2*g(j)
               call omega( a1,a2,b1,b2,0.0,w )
               wmax=amax1( wmax,w )
               wmin=amin1( wmin,w )
               ww(im)=w
               b(im)=-2.*(1.+del)
               fx(im)=-del*(cjm*x(i,jm)+cjp*x(i,jp))
               fy(im)=-del*(cjm*y(i,jm)+cjp*y(i,jp))
55          continue
            fx(1)=fx(1)-a(1)*x(1,j)
            fy(1)=fy(1)-a(1)*y(1,j)
            fx(nim2)=fx(nim2)-c(nim2)*x(ni,j)
            fy(nim2)=fy(nim2)-c(nim2)*y(ni,j)
            call bandit( a,b,c,fx,nim2 )
            call bandit( a,b,c,fy,nim2 )
            sum=(fp(1)*(y(1,jp)-y(1,jm))+fp(ni)*(y(ni,jp)-y(ni,jm)))/4.

* RELAX THE GRID POINTS ALONG GRID LINE J

            do 60 i=2,nim
               im=i-1
               xij=ww(im)*fx(im)+(1.-ww(im))*x(i,j)
               yij=ww(im)*fy(im)+(1.-ww(im))*y(i,j)
               erx=errfn( erx,xij,x(i,j) )
               ery=errfn( ery,yij,y(i,j) )
               x(i,j)=xij
               y(i,j)=yij
               sum=sum+fp(i)*(y(i,jp)-y(i,jm))/2.
60          continue
            snew=snew+sum/gpj/(x(ni,j)-x(1,j))
50       continue

* UPDATE COORDINATES ORTHOGONAL TO BOUNDARIES

         w=0.5
         do 65 i=2,nim
            xj1=-s*gp(1)/fp(i)*(y(i+1,1)-y(i-1,1))/2.
            xjn=-s*gp(nj)/fp(i)*(y(i+1,nj)-y(i-1,nj))/2.
            x1=(4.*x(i,2)-x(i,3)-2.*xj1)/3.
            xn=(4.*x(i,njm)-x(i,njm2)+2.*xjn)/3.
            x1=w*x1+(1.-w)*x(i,1)
            xn=w*xn+(1.-w)*x(i,nj)
            erxs=errfn( erxs,x1,x(i,1) )
            erxs=errfn( erxs,xn,x(i,nj) )
            x(i,1)=x1
            x(i,nj)=xn
65       continue
         do 70 j=2,njm
            yi1=-fp(1)/gp(j)/s*(x(1,j+1)-x(1,j-1))/2.
            yin=-fp(ni)/gp(j)/s*(x(ni,j+1)-x(ni,j-1))/2.
            y1=(4.*y(2,j)-y(3,j)-2.*yi1)/3.
            yn=(4.*y(nim,j)-y(nim2,j)+2.*yin)/3.
            y1=w*y1+(1.-w)*y(1,j)
            yn=w*yn+(1.-w)*y(ni,j)
            erys=errfn( erys,y1,y(1,j) )
            erys=errfn( erys,yn,y(ni,j) )
            y(1,j)=y1
            y(ni,j)=yn
70       continue

* UPDATE MONOTONIC VARIATION ON BOUNDARIES

         do 75 i=2,nim
            call ibound( x(i,1),x(i,nj),y(i,1),y(i,nj) )
75       continue
         do 80 j=2,njm
            call jbound( y(1,j),y(ni,j),x(1,j),x(ni,j) )
80       continue

* COMPUTE NEW AVERAGE CONFORMAL MODULE

         snew=snew/float( njm2 )
         snew=ws*snew+(1.-ws)*s
         ers=errfn( ers,snew,s )
         er=amax1( erx,ery,ers,erxs,erys )
         s=snew
         write(*,*) '============ iteration ',it,'============='
         write(*,*) 'local error in x transformation:',erx
         write(*,*) 'local error in y transformation:',ery
         write(*,*) 'local error in conformal modual:',ers
         write(*,*) 'local error on x surface       :',erxs
         write(*,*) 'local error on y surface       :',erys
         write(*,*) '                   global error:',er
         write(*,*) 'omega -max. :',wmax
         write(*,*) '      -min. :',wmin
      go to 40
45    return
      end

      subroutine bandit( a,b,c,f,n )

* This subroutine solves the tridiagonal system b(a,b,c) = f

* The solution is overwritten into f

      dimension a(n),b(n),c(n),f(n),x(100)

* Perform forward elimination

      np=n+1
      nm=n-1
      x(1)=c(1)/b(1)
      f(1)=f(1)/b(1)
      do 5 j=2,nm
         z=1./(b(j)-a(j)*x(j-1))
         x(j)=c(j)*z
         f(j)=(f(j)-a(j)*f(j-1))*z
5     continue
      z=1./(b(n)-a(n)*x(nm))
      f(n)=(f(n)-a(n)*f(nm))*z

* Perform backward substitution

      do 10 j1=2,n
         j=np-j1
         f(j)=f(j)-x(j)*f(j+1)
10    continue
      return
      end

      subroutine omega( a1,a2,b1,b2,c,w )

* This subroutine evaluates the acceleration factor at
* each grid point by the method of Erlich

* In this version complex roots were ingnored
*     only purely imaginary of real roots were considered

      common /accel/ cosx,cose
      d=abs( c-2.*(a1+a2) )
      f=4.*a1*a1-b1*b1
      g=4.*a2*a2-b2*b2
      c1=sqrt( abs( f ) )/d
      c2=sqrt( abs( g ) )/d
      rj=c1*cosx+c2*cose
      if (( f.ge.0. ).and.( g.ge.0. )) then
         w=2./(1.+sqrt( 1.-rj*rj ))
      else
         w=2./(1.+sqrt( 1.+rj*rj ))
      end if
      return
      end

      subroutine ibound( x1,x2,y1,y2 )

* This subroutine defines the surface of the collector and
* the out wall in polar coordinates at j=1 and nj

      real c(10)
      common a,b
      data c/-7089.e-5,0.0,-21070.e-5,6482.e-5,-1023.e-5,-283.e-5,
     #127.e-5,-15.e-5,-57.e-5,22.e-5/
      r=1.+c(1)
      do 5 n=3,10
         r=r+c(n)*cos( float( n-1 )*x1 )
5     continue
      y1=alog( 1./r )
      r=a*b/sqrt( a*a*sin( x2 )**2+b*b*cos( x2 )**2 )
      y2=alog( 1./r )
      return
      end

      subroutine jbound( y1,y2,x1,x2 )

* This subroutine defines the boundaries at i=1 and ni
* in this case it is the axis of symmetry of the collector
* in polar coordinates

      x1=3.1415926535
      x2=0.
      return
      end
      subroutine ipack( xi,f0,f1,f2,ni )

* This subroutine supplies the packing function in the i
* direction (along the surface of the body)

      data a/5.0/
      asinh(x)=alog( x+sqrt( x*x+1.0 ) )
      r=float( ni+1 )/2.0
      q=float( ni-1 )/2.0
      u=a/q*(xi-r)
      f0=r+q/asinh( a )*asinh( u )
      f1=a/asinh( a )/sqrt( u*u+1. )
      f2=-a*a*u/q/asinh( a )/( u*u+1. )**1.5
      return
      end

      subroutine jpack( eta,g0,g1,g2,nj )

* This subroutine supplies the packing function in the j
* direction (perpendicular to the surface of the body)


      g0=eta
      g1=1.0
      g2=0.0
      return
      end
```

Appendix E

The Navier-Stokes Algorithm

```
************************************************************************
*                                                                      *
*     This program solves the steady, incompressible, viscous flow     *
*     over an arbitrary body that is defined in the program NETWORK.   *
*                                                                      *
*                     By: KEVIN ELLWOOD                                *
*                         DEPT. OF CHEMICAL ENGINEERING                *
*                         UNIVERSITY OF WINDSOR                        *
*                         1987                                         *
*                                                                      *
* NOTE: This program is set up for axisymmetric flows but planar       *
*       flows can be solved by setting h3 = 1                          *
*                                                                      *
************************************************************************

* VARIABLES:

*          X       - MATRIX OF X AS A FUNCTION OF I,J
*          Y       - MATRIX OF Y AS A FUNCTION OF I,J
*          NI      - TOTAL NO. OF ALFA1 (I) COORDINATE LINES
*          NJ      - TOTAL NO. OF ALFA2 (J) COORDINATE LINES
*          TOL     - COMPUTATIONAL TOLERANCE
*          IMAX    - MAX. NO. OF ALLOWABLE ITERATIONS
*          S       - MATRIX CONTAINING STREAM FUNCTION
*          V       - MATRIX CONTAINING VORTICITY FUNCTION
*          H'S     - ARE THE GRID SCALE FACTORS

* SUBROUINES CALLED:

*          SOLVE   - SOLVES THE NAVIER-STOKES EQUATIONS
*          SCALE   - CALCULATES THE GRID SCALE FACTORS
*          BOUND   - ASSIGN DIRICHLET BOUNDARY CONDITIONS
*          INTER   - ASSIGNS AN INITIAL GUESS

      program navier
      external para,zero,rel,dif
      dimension  s(80,80), v(80,80), x(80,80), y(80,80)
      dimension h1(80,80),h2(80,80),h3(80,80)

************************************************************************
*                                                                      *
* For axisymmetr[illegible]flow -----> h3(i,j) = y(i,j)                *
* For planar flow -[illegible]-------> h3(i,j) = 1.0                   *
*                                                                      *
************************************************************************

      equivalence ( h3,y )
      character file*40,pick*40
      data iflag,tol,re,max,ier,iup,nu,io/1,5.e-4,30.,150,1,5,1,2/
      call cls
10    if ( iflag.ne.1 ) stop
         write(*,*) '    ╔════════════════════════════════════════════╗'
         write(*,*) '    ║ 1. │ Read grid data                        ║'
         write(*,*) '    ║ 2. │ Initialize solution from file         ║'
         write(*,*) '    ║ 3. │ Initialize solution to 0.0            ║'
         write(*,*) '    ║ 4. │ Initialize solution to straight flow  ║'
         write(*,*) '    ║ 5. │ Set parameters (relaxation etc...)    ║'
         write(*,*) '    ║ 6. │ Solve system                          ║'
         write(*,*) '    ║ 7. │ Save system                           ║'
         write(*,*) '    ║ d. │ Execute a dos command                 ║'
         write(*,*) '    ║ x. │ Exit program                          ║'
         write(*,*) '    ╚════════════════════════════════════════════╝'
         write(*,*) '          Enter choice:'
         read(*,20) pick
         call cls
         if ( pick(1:1).eq.'1' ) then
            write(*,*) 'Enter grid filename:'
            read(*,20) file
            open( unit=1,status='old',file=file,form='unformatted' )
            read(1) ni,nj,((x(i,j),i=1,ni),j=1,nj),
     &                     ((y(i,j),i=1,ni),j=1,nj)
            close( unit=1,status='keep' )
            call scale( x,y,h1,h2,ni,nj )
            call cls
         end if
         if ( pick(1:1).eq.'2' ) then
            write(*,*) 'Enter filename for initialization:'
            read(*,20) file
            open( unit=2,status='old',file=file,form='unformatted' )
            read(2) re,mn,jj,((s(i,j),i=1,mn),j=1,jj),
     &                        ((v(i,j),i=1,mn),j=1,jj)
            if (( mn.ne.ni ).or.( jj.ne.nj )) then
               write(*,*) 'Warning... initialization file mismatch'
            end if
            close( unit=2,status='keep' )
            call cls
         end if
         if ( pick(1:1).eq.'3' ) then
            call inter( s,v,x,y,ni,nj,zero )
            call cls
         end if
         if ( pick(1:1).eq.'4' ) then
            call inter( s,v,x,y,ni,nj,para )
            call cls
         end if
         if ( pick(1:1).eq.'5' ) then
            write(*,*) 'Enter the order of approximation (1 or 2):'
            read(*,*) io
            write(*,*) 'Enter tolerance:'
            read(*,*) tol
            write(*,*) 'Enter update frequency for omega:'
            read(*,*) iup
            write(*,*) 'Enter update frequency for surface vorticity:'
            read(*,*) nu
            write(*,*) 'Enter surface relaxation factor:'
            read(*,*) w1
            write(*,*) 'Enter reynolds number:'
            read(*,*) re
            write(*,*) 'Error --> 1. relative or 2. absolute (1 or 2):'
            read(*,*) ier
            call cls
         end if
         if ( pick(1:1).eq.'6' ) then
            call bound( s,v,x,y,ni,nj )
            if ( ier.eq.1 ) then
               call solve( io,s,v,h1,h2,h3,ni,nj,re,tol,nu,iup,w1,rel )
            else
               call solve( io,s,v,h1,h2,h3,ni,nj,re,tol,nu,iup,w1,dif )
            end if
            call cls
         end if
         if ( pick(1:1).eq.'7' ) then
            write(*,*) 'Enter filename to save solution:'
            read(*,20) file
            open( unit=2,status='new',file=file,form='unformatted' )
            write(2) re,ni,nj,((s(i,j),i=1,ni),j=1,nj),
     &                       ((v(i,j),i=1,ni),j=1,nj)
            close( unit=2,status='keep' )
            call cls
         end if
         if (( pick(1:1).eq.'d' ).or.( pick(1:1).eq.'D' )) then
            pause 'Enter dos command'
         end if
         if (( pick(1:1).eq.'x' ).or.( pick(1:1).eq.'x' )) iflag=0
         call cls
      go to 10
20    format(a40)
      end

      subroutine solve( io,s,v,h1,h2,h3,ni,nj,re,tol,nu,iup,w1,errfn )

************************************************************************
*                                                                      *
* This subroutine solves the Navier-Stokes equation                    *
*                                                                      *
************************************************************************

      dimension  s(80,80), f(80,80), v(80,80),wss(80,80),wfs(80,80)
      dimension h1(80,80),h2(80,80),h3(80,80),key(8),nis(2),nif(2)
      parameter (pi=3.141593)
      common cosx,cose
      data alfa/0.7/
      njm=nj-1
      nim=ni-1
      nim2=ni-2
      njm2=nj-2
      nis(1)=2
      nis(2)=3
      nif(1)=int( nim/2 )*2
      nif(2)=int( ni/2 )*2-1
      cosx=cos( pi/(float( nim2 )+1.) )
      cose=cos( pi/(float( njm2 )+1.) )

* Initializing modified vorticity function f ; f = -vorticity/h3

      do 5 j=1,nj
         do 5 i=2,nim
            f(i,j)=-v(i,j)/h3(i,j)
5     continue
      it=0
      er=10.*tol
      wsmax=0.0
      wsmin=2.0
      key(4)=0
10    if (( er.lt.tol ).or.( key(4).eq.1 )) go to 15
         key(4)=0
         iaccel=mod(it,iup)
         it=it+1
         ers=0.0
         erf=0.0
         if ( iaccel.eq.0 ) then
            wfmax=0.0
            wfmin=2.0
         end if

* Update f on the axis of symetry using L'Hopitals rule because v/h3 = 0/0

         do 40 j=2,njm

* Update f on i=1 by forward differencing

            h3i=4.*h3(2,j)-h3(3,j)
            vi=4.*h3(2,j)*f(2,j)-h3(3,j)*f(3,j)
            f(1,j)=vi/h3i

* Update f on i=ni by backward differencing

            h3i=h3(nim2,j)-4.*h3(nim,j)
            vi=h3(nim2,j)*f(nim2,j)-4.*h3(nim,j)*f(nim,j)
            f(ni,j)=vi/h3i
40       continue
         do 50 l=1,2
         do 20 j=2,njm
            mj=mod( j+l-1,2 )+1
            jp=j+1
            jm=j-1
            do 25 i=nis(mj),nif(mj),2
               ip=i+1
               im=i-1
               ha=h1(i,j)
               hb=h2(i,j)
               hc=h3(i,j)
               h1i=(h1(ip,j)-h1(im,j))/2.
               h2i=(h2(ip,j)-h2(im,j))/2.
               h3i=(h3(ip,j)-h3(im,j))/2.
               h1j=(h1(i,jp)-h1(i,jm))/2.
               h2j=(h2(i,jp)-h2(i,jm))/2.
               h3j=(h3(i,jp)-h3(i,jm))/2.
               h213=hb/ha/hc
               h123=ha/hb/hc
               a3=ha*hb*hc
               h213i=h2i/hb-h1i/ha-h3i/hc
               h123j=h1j/ha-h2j/hb-h3j/hc

* Calculating coeficients of difference equations for stream function

               ajm=h123*(1.-0.5*h123j)
               aim=h213*(1.-0.5*h213i)
               ao=2.*(h213+h123)
               aip=h213*(1.+0.5*h213i)
               ajp=h123*(1.+0.5*h123j)

* Calculating local acceleration factors by the method of L.W. Ehrlich

               if ( it.eq.1 ) then
                  call omega( ajm,aim,ao,aip,ajp,w )
                  wss(i,j)=w
                  wsmax=amax1( wsmax,wss(i,j) )
                  wsmin=amin1( wsmin,wss(i,j) )
               end if
               ws=wss(i,j)
               sum=ajm*s(i,jm)+aim*s(im,j)+aip*s(ip,j)+ajp*s(i,jp)
               sij=(sum-a3*f(i,j))/ao
               sij=ws*sij+(1.-ws)*s(i,j)
               ers=errfn( ers,sij,s(i,j) )
               s(i,j)=sij
               si=(s(ip,j)-s(im,j))/2.
               sj=(s(i,jp)-s(i,jm))/2.
```

```
* Calculating coeficients of difference equations for vorticity function

          if ( io.eq.1 ) then
            ei=sign( 1.,sj )
            ej=sign( 1.,-si )
          else
            ei=0.0
            ej=0.0
          end if
          ajm=ajm*h3(i,jm)**2-re/4.*si*(1.+ej)
          aim=aim*h3(im,j)**2+re/4.*sj*(1.+ei)
          ao=2./h123+2./h213+re/2.*(ei*sj-ej*si)
          aip=aip*h3(ip,j)**2-re/4.*sj*(1.-ei)
          ajp=ajp*h3(i,jp)**2+re/4.*si*(1.-ej)

* Calculating local acceleration factors by the method of L.W. Ehrlich

          if ( iaccel.eq.0 ) then
            call omega( ajm,aim,ao,aip,ajp,w )
            wfs(i,j)=w
            wfmax=amax1( wfmax,wfs(i,j) )
            wfmin=amin1( wfmin,wfs(i,j) )
          end if
          wf=wfs(i,j)*alfa
          sum=ajm*f(i,jm)+aim*f(im,j)+aip*f(ip,j)+ajp*f(i,jp)
          fij=sum/ao
          fij=wf*fij+(1.-wf)*f(i,j)
          erf=errfn( erf,fij,f(i,j) )
          f(i,j)=fij
25      continue
20    continue
50    continue

* Update vorticity on body surface

      wsurf=amin1( wfmin,w1 )
      if ( mod( it-1,mu ).eq.0 ) then
      ersurf=0.0
      vmax=0.0
      do 35 i=2,nim
        vor=(8.*s(i,2)-s(i,3))/2./(h2(i,1)*h3(i,1))**2
        vor=wsurf*vor+(1.-wsurf)*f(i,1)
        ersurf=errfn( ersurf,vor,f(i,1) )
        f(i,1)=vor
        vort=-vor*h3(i,1)
        if ( abs( vort ).gt.abs( vmax ) ) then
          vmax=vort
        end if
35    continue
      end if
```

```
         call statkb( key )
         er=amax1( ers,erf,ersurf )
         write(*,200) it
         write(*,210) ers,erf,ersurf,er
         write(*,220) wsmax,wfmax,wsmin,wfmin
         write(*,*) 'Max. surface vorticity:',vmax
      go to 10

* Calculate actual vorticity

15    do 45 j=1,njm
         do 45 i=2,nim
            v(i,j)=-f(i,j)*h3(i,j)
45    continue
      return
200   format(1x,14('='),' iteration no. ',i3,1x,14('='))
210   format(1x,'local error in stream function   :',e11.4,/,
     &1x,'local error in vorticity function:',e11.4,/,
     &1x,'local error in surface vorticity :',e11.4,/,22x,
     &'global error:',e11.4,/)
220   format(15x,'stream func.  vorticity func.',/,1x,
     &'omega - max.:    ',f7.4,8x,f7.4,/,1x,
     &'      - min.:    ',f7.4,8x,f7.4,/)
      end

      subroutine bound( s,v,x,y,ni,nj )

********************************************************************
*                                                                  *
* This subroutine assigns the Dirichlet boundary conditions        *
*                                                                  *
********************************************************************

      dimension s(80,80),v(80,80),x(80,80),y(80,80)

* Assume the fluid is in steady parallel flow

      do 5 j=1,nj
         s(1,j)=0.
         s(ni,j)=0.
         v(1,j)=0.
         v(ni,j)=0.
5     continue
      do 10 i=1,ni
         s(i,1)=0.
         s(i,nj)=0.5*y(i,nj)**2
         v(i,nj)=0.
10    continue
      return
      end
      subroutine inter( s,v,x,y,ni,nj,func )

**********************************************************************
*                                                                    *
* This subroutine assigns the value of 'func' as an initial guess   *
*                                                                    *
**********************************************************************

      dimension s(80,80),v(80,80),x(80,80),y(80,80)
      do 5 i=2,ni-1
         do 5 j=2,nj-1
            xx=x(i,j)
            yy=y(i,j)
            s(i,j)=func( xx,yy )
            v(i,j)=0.0
5     continue
      return
      end

      subroutine scale( x,y,h1,h2,ni,nj )
      dimension x(80,80),y(80,80),h1(80,80),h2(80,80)

*********************************************************************
*                                                                   *
* This subroutine calculates the scale factors H1 & H2 from         *
* the orthogonal grid defined by X & Y                              *
*                                                                   *
*********************************************************************

      do 5 j=1,nj
         xi=(-3.*x(1,j)+4.*x(2,j)-x(3,j))/2.
         yi=(-3.*y(1,j)+4.*y(2,j)-y(3,j))/2.
         h1(1,j)=sqrt(xi*xi+yi*yi)
         do 10 i=2,ni-1
            xi=(x(i+1,j)-x(i-1,j))/2.
            yi=(y(i+1,j)-y(i-1,j))/2.
            h1(i,j)=sqrt( xi*xi+yi*yi )
10       continue
         xi=(3.*x(ni,j)-4.*x(ni-1,j)+x(ni-2,j))/2.
         yi=(3.*y(ni,j)-4.*y(ni-1,j)+y(ni-2,j))/2.
         h1(ni,j)=sqrt( xi*xi+yi*yi )
5     continue
      do 15 i=1,ni
         xj=(-3.*x(i,1)+4.*x(i,2)-x(i,3))/2.
         yj=(-3.*y(i,1)+4.*y(i,2)-y(i,3))/2.
         h2(i,1)=sqrt( xj*xj+yj*yj )
         do 20 j=2,nj-1
            xj=(x(i,j+1)-x(i,j-1))/2.
            yj=(y(i,j+1)-y(i,j-1))/2.
```

```
               h2(i,j)=sqrt( xj*xj+yj*yj )
20          continue
            xj=(3.*x(i,nj)-4.*x(i,nj-1)+x(i,nj-2))/2.
            yj=(3.*y(i,nj)-4.*y(i,nj-1)+y(i,nj-2))/2.
            h2(i,nj)=sqrt( xj*xj+yj*yj )
15       continue
         return
         end

         subroutine cls
         write(*,10)
10       format('1')
         return
         end

         function para( x,y )
         para=0.5*y**2
         return
         end

         function zero( x,y )
         zero=0.0
         return
         end

         subroutine omega( ajm,aim,ao,aip,ajp,w )
************************************************************************
* .                                                                    *
* This subroutine calculates the acceleration factors for each         *
* grid point                                                           *
*                                                                      *
************************************************************************
         complex u
         common cosx,cose
         data t/.3333333/
         f=aim*aip
         g=ajm*ajp
         u=2.*(csqrt( cmplx( f ) )*cosx+csqrt( cmplx( g ) )*cose)/ao
         ur=real( u )
         ui=aimag( u )
         aa=ur*ur+ui*ui
         aa2=aa*aa
         bb=ur*ur-ui*ui
         a=aa2-bb*bb
         b=aa2-bb
         c=sqrt( abs( a+b*b ) )
         e=(3.*b+c)*(abs(a*(c-b)))**t-(3.*b-c)*(abs(a*(c+b)))**t
         bar=(e+aa2+3.*bb*bb-4.*aa2*bb)/(aa2**2-aa2*bb)
```

```
if ( aa2.gt.bb ) then
   w=-(bar-sqrt( bar**2+4.*bar ))/2.
else
   w=-(bar+sqrt( bar**2+4.*bar ))/2.
end if
return
end

function rel( er,unew,uold )
rel=amax1( er,abs( unew-uold )/(abs( unew )+1.))
return
end

function dif( er,unew,uold )
dif=amax1( er,abs( unew-uold ) )
return
end
```

Appendix F

Potential Flow Algorithm

```
************************************************************************
*                                                                      *
*     This program solves the steady, irrotational flow over an        *
*     arbitrary body that is defined in the program NETWORK.           *
*                                                                      *
*                       By: KEVIN ELLWOOD                              *
*                           DEPT. OF CHEMICAL ENGINEERING              *
*                           UNIVERSITY OF WINDSOR                      *
*                           1987                                       *
*                                                                      *
* NOTE: This program is set up for axisymetric flows but planar        *
*       flows can be solved by setting h3 = 1                          *
*                                                                      *
************************************************************************

* VARIABLES:

*          X       - MATRIX OF X AS A FUNCTION OF I,J
*          Y       - MATRIX OF Y AS A FUNCTION OF I,J
*          NI      - TOTAL NO. OF ALFA1 (I) COORDINATE LINES
*          NJ      - TOTAL NO. OF ALFA2 (J) COORDINATE LINES
*          TOL     - COMPUTATIONAL TOLERANCE
*          IMAX    - MAX. NO. OF ALLOWABLE ITERATIONS
*          S       - MATRIX CONTAINING STREAM FUNCTION
*          H'S     - ARE THE GRID SCALE FACTORS

* SUBROUINES CALLED:

*          SOLVE   - SOLVES THE NAVIER-STOKES EQUATIONS
*          SCALE   - CALCULATES THE GRID SCALE FACTORS
*          BOUND   - ASSIGN DIRICHLET BOUNDARY CONDITIONS
*          INTER   - ASSIGNS AN INITIAL GUESS

      external para,zero,rel,dif
      dimension  s(80,80), x(80,80), y(80,80)
      dimension h1(80,80),h2(80,80),h3(80,80)
      equivalence ( h3,y )
      character file*40,pick*40
      data iflag,tol,re,max,v,ier/1,5.e-4,30.,150,0.0,1/
      call cls
10    if (iflag.ne.1) stop
        write(*,*) '    ┌────┬──────────────────────────────────────┐'
        write(*,*) '    │ 1. │ Read grid data                       │'
        write(*,*) '    │ 2. │ Initialize solution from file        │'
        write(*,*) '    │ 3. │ Initialize solution to 0.0           │'
        write(*,*) '    │ 4. │ Initialize solution to straight flow │'
        write(*,*) '    │ 5. │ Set parameters (relaxation etc...)   │'
        write(*,*) '    │ 6. │ Solve system                         │'
        write(*,*) '    │ 7. │ Save system                          │'
      write(*,*) '       | x. | Exit program                    |'
      write(*,*) '       |____|________________________________|'
      write(*,*) '              Enter choice:'
      read(*,20) pick
      call cls
      if (pick(1:1).eq.'1') then
         write(*,*) 'Enter grid filename:'
         read(*,20) file
         open( unit=1,status='old',file=file,form='unformatted' )
         read(1) ni,nj,((x(i,j),i=1,ni),j=1,nj),
     &                 ((y(i,j),i=1,ni),j=1,nj)
         close( unit=1,status='keep' )
         call scale( x,y,h1,h2,ni,nj )
         call cls
      end if
      if (pick(1:1).eq.'2') then
         write(*,*) 'Enter filename for initialization:'
         read(*,20) file
         open( unit=2,status='old',file=file,form='unformatted' )
         read(2) re,mm,jj,((s(i,j),i=1,mm),j=1,jj)
         if ((mm.ne.ni).or.(jj.ne.nj)) then
            write(*,*) 'Warning... initialization file mismatch'
         end if
         close( unit=2,status='keep' )
         call cls
      end if
      if (pick(1:1).eq.'3') then
         call inter( s,x,y,ni,nj,zero )
         call cls
      end if
      if (pick(1:1).eq.'4') then
         call inter( s,x,y,ni,nj,para )
         call cls
      end if
      if (pick(1:1).eq.'5') then
         write(*,*) 'Enter tolerance:'
         read(*,*) tol
         write(*,*) 'Enter max. no. iterations:'
         read(*,*) max
         write(*,*) 'Enter reynolds number:'
         read(*,*) re
         write(*,*) 'Error --> 1. relative or 2. absolute (1/2)'
         read(*,*) ier
         call cls
      end if
      if (pick(1:1).eq.'6') then
         call bound( s,x,y,ni,nj )
         if ( ier.eq.1 ) then
            call solve( s,h1,h2,h3,ni,nj,tol,max,it,rel )
         else
```

```
               call solve( s,h1,h2,h3,ni,nj,tol,max,it,dif )
            end if
            call cls
         end if
         if (pick(1:1).eq.'7') then
            write(*,*) 'Enter filename to save solution:'
            read(*,20) file
            open( unit=2,status='new',file=file,form='unformatted' )
            write(2) re,ni,nj,((s(i,j),i=1,ni),j=1,nj),
     &                        ((v,i=1,ni),j=1,nj)
            close( unit=2,status='keep' )
            call cls
         end if
         if ((pick(1:1).eq.'x').or.(pick(1:1).eq.'x')) iflag=0
         call cls
      go to 10
20    format(a40)
      end

      subroutine solve( s,h1,h2,h3,ni,nj,tol,max,it,errfn )

*************************************************************************
*                                                                       *
*  This subroutine solves the difference equations representing        *
*  potential flow by the block-successive overrelaxation method        *
*  with each grid point having its own relaxation factor                *
*                                                                       *
*************************************************************************

      dimension s(80,80),h1(80,80),h2(80,80),h3(80,80)
      dimension a(100),b(100),c(100),f(100),ws(80,80)
      common cosx,cose
      data pi/3.1415926/
      njm=nj-1
      nim=ni-1
      nim2=ni-2
      njm2=nj-2
      wsmax=0.0
      wsmin=2.0
      cosx=cos( pi/(float( nim2 )+1.) )
      cose=cos( pi/(float( njm2 )+1.) )

* Scan grid for local accelation factors

      do 25 j=2,njm
         jp=j+1
         jm=j-1
         do 25 i=2,nim
            ip=i+1
            im=i-1
            ha=h1(i,j)
            hb=h2(i,j)
            hc=h3(i,j)
            h1i=(h1(ip,j)-h1(im,j))/2.
            h2i=(h2(ip,j)-h2(im,j))/2.
            h3i=(h3(ip,j)-h3(im,j))/2.
            h1j=(h1(i,jp)-h1(i,jm))/2.
            h2j=(h2(i,jp)-h2(i,jm))/2.
            h3j=(h3(i,jp)-h3(i,jm))/2.
            h213=hb/ha/hc
            h123=ha/hb/hc
            h213i=h2i/hb-h1i/ha-h3i/hc
            h123j=h1j/ha-h2j/hb-h3j/hc
            ajm=h123*(1.-0.5*h123j)
            aim=h213*(1.-0.5*h213i)
            ao=2.*(h213+h123)
            aip=h213*(1.+0.5*h213i)
            ajp=h123*(1.+0.5*h123j)
            ws(i,j)=omega( ajm,aim,ao,aip,ajp )
            wsmax=amax1( wsmax,ws(i,j) )
            wsmin=amin1( wsmin,ws(i,j) )
25    continue
      it=0
      er=10.*tol
20    if (( er.lt.tol ).or.( it.gt.max )) return
        it=it+1
        er=0.0
        do 30 j=2,njm
          jp=j+1
          jm=j-1
          do 35 i=2,nim
            ip=i+1
            im=i-1
            ha=h1(i,j)
            hb=h2(i,j)
            hc=h3(i,j)
            h1i=(h1(ip,j)-h1(im,j))/2.
            h2i=(h2(ip,j)-h2(im,j))/2.
            h3i=(h3(ip,j)-h3(im,j))/2.
            h1j=(h1(i,jp)-h1(i,jm))/2.
            h2j=(h2(i,jp)-h2(i,jm))/2.
            h3j=(h3(i,jp)-h3(i,jm))/2.
            h213=hb/ha/hc
            h123=ha/hb/hc
            h213i=h2i/hb-h1i/ha-h3i/hc
            h123j=h1j/ha-h2j/hb-h3j/hc
            ajm=h123*(1.-0.5*h123j)
            aim=h213*(1.-0.5*h213i)
            ao=2.*(h213+h123)
            aip=h213*(1.+0.5*h213i)
```

```
              ajp=h123*(1.+0.5*h123j)
              a(im)=aim
              b(im)=-ao
              c(im)=aip
              f(im)=-ajm*s(i,jm)-ajp*s(i,jp)-aim*s(im,j)-aip*s(ip,j)
              f(im)=(f(im)+ao*s(i,j))*ws(i,j)
35         continue

* Correct f for boundary conditions at i=1 and i=ni

             f(1)=f(1)-a(1)*s(1,j)
             f(nim2)=f(nim2)-c(nim2)*s(ni,j)

* Solve the tridiagonal matrix

            call bandit( a,b,c,f,nim2 )

* Relax solution along the entire gridline

            do 40 i=2,nim
               im=i-1
               psi=s(i,j)+f(im)
               er=errfn( er,psi,s(i,j) )
               s(i,j)=psi
40          continue
30       continue
         write(*,200) it,max
         write(*,210) 'stream   ',er,wsmax,wsmin
      go to 20
      return
200   format(1x,11('='),'iteration no. ',i3,' of ',i3,11('='))
210   format(1x,a9,' function  - local error:',e11.4,/,21x,'- omega',
     &' max. ',f7.3,/,29x,'min. ',f7.3)
      end

      subroutine bound( s,x,y,ni,nj )
      dimension s(80,80),x(80,80),y(80,80)

* Assume the fluid is in steady parallel flow

      do 5 j=1,nj
         s(1,j)=0.
         s(ni,j)=0.
5     continue
      do 10 i=1,ni
         s(i,1)=0.
         s(i,nj)=0.5*y(i,nj)**2
10    continue
      return
      end
      subroutine inter( s,x,y,ni,nj,func )
      dimension s(80,80),x(80,80),y(80,80)
      do 5 i=2,ni-1
         do 5 j=2,nj-1
            xx=x(i,j)
            yy=y(i,j)
            s(i,j)=func( xx,yy )
5     continue
      return
      end

      subroutine scale( x,y,h1,h2,ni,nj )
      dimension x(80,80),y(80,80),h1(80,80),h2(80,80)

* This subroutine calculates the scale factors h1 & h2 from
*    the orthogonal grid defined by x & y

      do 5 j=1,nj
         xi=(-3.*x(1,j)+4.*x(2,j)-x(3,j))/2.
         yi=(-3.*y(1,j)+4.*y(2,j)-y(3,j))/2.
         h1(1,j)=sqrt( xi*xi+yi*yi )
         do 10 i=2,ni-1
            xi=(x(i+1,j)-x(i-1,j))/2.
            yi=(y(i+1,j)-y(i-1,j))/2.
            h1(i,j)=sqrt( xi*xi+yi*yi )
10       continue
         xi=(3.*x(ni,j)-4.*x(ni-1,j)+x(ni-2,j))/2.
         yi=(3.*y(ni,j)-4.*y(ni-1,j)+y(ni-2,j))/2.
         h1(ni,j)=sqrt( xi*xi+yi*yi )
5     continue
      do 15 i=1,ni
         xj=(-3.*x(i,1)+4.*x(i,2)-x(i,3))/2.
         yj=(-3.*y(i,1)+4.*y(i,2)-y(i,3))/2.
         h2(i,1)=sqrt( xj*xj+yj*yj )
         do 20 j=2,nj-1
            xj=(x(i,j+1)-x(i,j-1))/2.
            yj=(y(i,j+1)-y(i,j-1))/2.
            h2(i,j)=sqrt( xj*xj+yj*yj )
20       continue
         xj=(3.*x(i,nj)-4.*x(i,nj-1)+x(i,nj-2))/2.
         yj=(3.*y(i,nj)-4.*y(i,nj-1)+y(i,nj-2))/2.
         h2(i,nj)=sqrt( xj*xj+yj*yj )
15    continue
      return
      end
      subroutine bandit( a,b,c,f,n )

* This subroutine solves the tridiagonal system B( a,b,c ) = f

* The solution is overwritten into f

      dimension a(n),b(n),c(n),f(n),x(100)

* Perform forward elimination

      np=n+1
      nm=n-1
      x(1)=c(1)/b(1)
      f(1)=f(1)/b(1)
      do 5 j=2,nm
         z=1./(b(j)-a(j)*x(j-1))
         x(j)=c(j)*z
         f(j)=(f(j)-a(j)*f(j-1))*z
5     continue
      z=1./(b(n)-a(n)*x(nm))
      f(n)=(f(n)-a(n)*f(nm))*z

* Perform backward substitution

      do 10 j1=2,n
         j=np-j1
         f(j)=f(j)-x(j)*f(j+1)
10    continue
      return
      end


      subroutine cls
      write(*,10)
10    format('1')
      return
      end

      function para( x,y )
      para=0.5*y**2
      return
      end

      function zero( x,y )
      zero=0.0
      return
      end
      function omega( ajm,aim,ao,aip,ajp )

* This subroutine evaluates the acceleration factor for each
* grid point by the method of Ehrlich [8]

      complex u
      common cosx,cose
      data t/.3333333/
      f=aip*aim
      g=ajp*ajm
      u=2.*(csqrt( cmplx( f ) )*cosx+csqrt( cmplx( g ) )*cose)/ao
      ur=real( u )
      ui=aimag( u )
      aa=ur*ur+ui*ui
      aa2=aa*aa
      bb=ur*ur-ui*ui
      a=aa2-bb*bb
      b=aa2-bb
      c=sqrt( abs( a+b*b ) )
      e=(3.*b+c)*(abs(a*(c-b)))**t-(3.*b-c)*(abs(a*(c+b)))**t
      bar=(e+aa2+3.*bb*bb-4.*aa2*bb)/(aa2**2-aa2*bb)
      if ( aa2.gt.bb ) then
         omega=-(bar-sqrt( bar**2+4.*bar ))/2.
      else
         omega=-(bar+sqrt( bar**2+4.*bar ))/2.
      end if
      return
      end

      function rel( er,unew,uold )
      rel=amax1( er,abs( unew-uold )/(abs( unew )+1.))
      return
      end

      function dif( er,unew,uold )
      dif=amax1( er,abs( unew-uold ) )
      return
      end
```

Appendix G

The Convective Diffusion Algorithm

```
***********************************************************************
*                                                                     *
*     This program solves the steady convective diffusion equation    *
*     for an arbitrarily shape body that is defined in the program    *
*     NETWORK.  The stream fuction is read in and could be the        *
*     result of a potential calculation from program POTENT or        *
*     a viscous flow calculation form program NAVIER.                 *
*                                                                     *
*                     By: KEVIN ELLWOOD                               *
*                         DEPT. OF CHEMICAL ENGINEERING               *
*                         UNIVERSITY OF WINDSOR                       *
*                         1987                                        *
*                                                                     *
* NOTE: This program is set up for axisymetric flows but planar       *
*       flows can be solved by setting h3 = 1                         *
*                                                                     *
***********************************************************************

* VARIABLES:

*        X      - MATRIX OF X AS A FUNCTION OF I,J
*        Y      - MATRIX OF Y AS A FUNCTION OF I,J
*        NI     - TOTAL NO. OF ALFA1 (I) COORDINATE LINES
*        NJ     - TOTAL NO. OF ALFA2 (J) COORDINATE LINES
*        TOL    - COMPUTATIONAL TOLERANCE
*        S      - MATRIX CONTAINING STREAM FUNCTION
*        T      - TEMPERATURE OF VAPOR DISTRIBUTION
*        H'S    - ARE THE GRID SCALE FACTORS

* SUBROUINES CALLED:

*        SOLVE  - SOLVES THE CONVECTIVE DIFFUSION EQUATION
*        SCALE  - CALCULATES THE GRID SCALE FACTORS
*        BOUND  - ASSIGN DIRICHLET BOUNDARY CONDITIONS
*        LINEAR - ASSIGNS AN INITIAL GUESS

      external para,zero,rel,dif
      dimension  t(80,80), s(80,80), x(80,80), y(80,80)
      dimension h1(80,80),h2(80,80),h3(80,80)
      equivalence ( h3,y )
      character file*40,pick*40
      data iflag,tol,pe,max,ier/1,5.e-4,30.,150,1/
      call cls
10    if ( iflag.ne.1 ) stop
        write(*,*) '       ══════════════════════════════════════════ '
        write(*,*) '      │ 1. │ Read grid and stream function data  │'
        write(*,*) '      │ 2. │ Initialize solution from file       │'
        write(*,*) '      │ 3. │ Initialize solution to be linear    │'
        write(*,*) '      │ 4. │ Set parameters (tolerance etc...)   │'
      write(*,*) '     ║ 5.│ Solve system                       ║'
      write(*,*) '     ║ 6.│ Save system                        ║'
      write(*,*) '     ║ x.│ Exit program                       ║'
      write(*,*) '     ╚═══╧════════════════════════════════════╝'
      write(*,*) '           Enter choice:'
      read(*,20) pick
      call cls
      if ( pick(1:1).eq.'1' ) then
         write(*,*) 'Enter grid filename:'
         read(*,20) file
         open( unit=1,status='old',file=file,form='unformatted' )
         read(1) ni,nj,((x(i,j),i=1,ni),j=1,nj),
     &                 ((y(i,j),i=1,ni),j=1,nj)
         close( unit=1,status='keep' )
         call scale( x,y,h1,h2,ni,nj )
         write(*,*) 'Enter stream function filename:'
         read(*,20) file
         open( unit=1,status='old',file=file,form='unformatted' )
         read(1) re,nn,jj,((s(i,j),i=1,nn),j=1,jj)
         if (( nn.ne.ni ).or.( jj.ne.nj )) then
            write(*,*) 'Warning... initialization file mismatch'
         end if
         close( unit=1,status='keep' )
         call cls
      end if
      if ( pick(1:1).eq.'2' ) then
         write(*,*) 'Enter filename for initialization:'
         read(*,20) file
         open( unit=2,status='old',file=file,form='unformatted' )
         read(2) pe,nn,jj,((t(i,j),i=1,nn),j=1,jj)
         if (( nn.ne.ni ).or.( jj.ne.nj )) then
            write(*,*) 'Warning... initialization file mismatch'
         end if
         close( unit=2,status='keep' )
         call cls
      end if
      if ( pick(1:1).eq.'3' ) then
         call bound( t,ni,nj )
         call linear( t,ni,nj )
      end if
      if ( pick(1:1).eq.'4' ) then
         write(*,*) 'Enter tolerance:'
         read(*,*) tol
         write(*,*) 'Enter frequency to update boundaries:'
         read(*,*) nu
         write(*,*) 'Enter boundary relaxation factor:'
         read(*,*) wb
         write(*,*) 'Enter dampening factor:'
         read(*,*) alf
         write(*,*) 'Enter peclect number:'
```

```
            read(*,*) pe
            write(*,*) 'Error --> 1. relative or 2. absolute (1 or 2):'
            read(*,*) ier
            call cls
         end if
         if ( pick(1:1).eq.'5' ) then
            call bound( t,ni,nj )
            if ( ier.eq.1 ) then
               call solve( t,s,h1,h2,h3,ni,nj,pe,tol,nu,wb,it,alf,rel )
            else
               call solve( t,s,h1,h2,h3,ni,nj,pe,tol,nu,wb,it,alf,dif )
            end if
            call cls
         end if
         if ( pick(1:1).eq.'6' ) then
            write(*,*) 'Enter filename to save solution:'
            read(*,20) file
            open( unit=2,status='new',file=file,form='unformatted' )
            write(2) pe,ni,nj,((t(i,j),i=1,ni),j=1,nj)
            close( unit=2,status='keep' )
            call cls
         end if
         if (( pick(1:1).eq.'x' ).or.( pick(1:1).eq.'x' )) iflag=0
         call cls
      go to 10
20    format(a40)
      end

      subroutine solve( t,s,h1,h2,h3,ni,nj,pe,tol,nu,wb,it,alf,errfn )

************************************************************************
*                                                                      *
* This subroutine solves the convective diffusion equation             *
* by the variable relaxation factor by Erhlich [8]                     *
*                                                                      *
************************************************************************

      dimension  t(80,80), s(80,80), w(80,80)
      dimension h1(80,80),h2(80,80),h3(80,80),key(8),njs(2),njf(2)
      common cosx,cose
      njm=nj-1
      nim=ni-1
      nim2=ni-2
      njm2=nj-2
      njs(1)=2
      njs(2)=3
      njf(1)=int( njm/2 )*2
      njf(2)=int( nj/2 )*2-1
      cosx=cos( pi/(float( ni )+1.) )
      cose=cos( pi/(float( njm2 )+1.) )
      it=0
      er=10.*tol
      wmax=0.0
      wmin=2.0

* calculate acceleration factors for the temperatures

      do 20 j=2,njm
         jp=j+1
         jm=j-1
         do 25 i=2,nim
            ip=i+1
            im=i-1
            ha=h1(i,j)
            hb=h2(i,j)
            hc=h3(i,j)
            h231p=h2(ip,j)*h3(ip,j)/h1(ip,j)
            h231m=h2(im,j)*h3(im,j)/h1(im,j)
            h132p=h1(i,jp)*h3(i,jp)/h2(i,jp)
            h132m=h1(i,jm)*h3(i,jm)/h2(i,jm)
            h231i=0.5*(h231p-h231m)
            h132j=0.5*(h132p-h132m)
            h231=hb*hc/ha
            h132=ha*hc/hb
            b1=0.5*h231i-pe/8.*(s(i,jp)-s(i,jm))
            b2=0.5*h132j+pe/8.*(s(ip,j)-s(im,j))
            ajm=h132-b2
            aim=h231-b1
            ao=2.*(h231+h132)
            aip=h231+b1
            ajp=h132+b2
            w(i,j)=omega( ajm,aim,ao,aip,ajp )
            wmax=amax1( wmax,w(i,j) )
            wmin=amin1( wmin,w(i,j) )
            w(i,j)=w(i,j)*alf
25       continue
20    continue
      key(4)=0
10    if (( er.lt.tol ).or.( key(4).eq.1 )) go to 15
         key(4)=0
         it=it+1
         er=0.0
         if ( mod( it-1,mu ).eq.0 ) erb=0.0
         do 50 l=1,2
         do 30 j=njs(l),njf(l),2
            jp=j+1
            jm=j-1
            do 35 i=2,nim
               ip=i+1
               im=i-1
               ha=h1(i,j)
               hb=h2(i,j)
               hc=h3(i,j)
               h231p=h2(ip,j)*h3(ip,j)/h1(ip,j)
               h231m=h2(im,j)*h3(im,j)/h1(im,j)
               h132p=h1(i,jp)*h3(i,jp)/h2(i,jp)
               h132m=h1(i,jm)*h3(i,jm)/h2(i,jm)
               h231i=0.5*(h231p-h231m)
               h132j=0.5*(h132p-h132m)
               h231=hb*hc/ha
               h132=ha*hc/hb
               b1=0.5*h231i-pe/8.*(s(i,jp)-s(i,jm))
               b2=0.5*h132j+pe/8.*(s(ip,j)-s(im,j))
               ajm=h132-b2
               aim=h231-b1
               ao=2.*(h231+h132)
               aip=h231+b1
               ajp=h132+b2
               sum=ajm*t(i,jm)+aim*t(im,j)+aip*t(ip,j)+ajp*t(i,jp)
               tij=sum/ao
               tij=w(i,j)*tij+(1.-w(i,j))*t(i,j)
               er=errfn( er,tij,t(i,j) )
               t(i,j)=tij
35          continue

* Update boundaries for zero flux

            if ( mod( it-1,nu ).eq.0 ) then
               t1=(4.*t(2,j)-t(3,j))/3.
               t1=wb*t1+(1.-wb)*t(1,j)
               tn=(4.*t(nim,j)-t(nim2,j))/3.
               tn=wb*tn+(1.-wb)*t(ni,j)
               erb=errfn( erb,t1,t(1,j) )
               erb=errfn( erb,tn,t(ni,j) )
               t(1,j)=t1
               t(ni,j)=tn
            end if
30       continue
50       continue
         write(*,200) it
         write(*,*) 'local error in temperature:',er
         write(*,*) 'local error in boundary   :',erb
         write(*,*) 'omega    -max.:',wmax
         write(*,*) '         -min.:',wmin
         er=amax1( er,erb )
         call statkb( key )
      go to 10
15    return
200   format(1x,14('='),' iteration no. ',i3,1x,14('='))
      end
      subroutine bound( t,ni,nj )
      dimension t(80,80)

* the equations are made dimensionless with respect to the surface temp.

      do 10 i=1,ni
         t(i,1)=1.0
         t(i,nj)=0.0
10    continue
      return
      end

      subroutine linear( t,ni,nj )
      dimension t(80,80)
      do 5 j=2,nj-1
         rj1=float(j-1)/float(nj-1)
         rj2=float(nj-j)/float(nj-1)
         do 5 i=1,ni
            t(i,j)=rj1*t(i,nj)+rj2*t(i,1)
5     continue
      return
      end

      subroutine scale( x,y,h1,h2,ni,nj )
******************************************************************
*                                                                *
* This subroutine solves evaluates the grid scale factors        *
* The grid is defined as the ordered pair x & y                  *
*                                                                *
******************************************************************

      dimension x(80,80),y(80,80),h1(80,80),h2(80,80)

* THIS SUBROUTINE CALCULATES THE SCALE FACTORS H1 & H2 FROM
* THE ORTHOGONAL GRID DEFINED BY X & Y

      do 5 j=1,nj
         xi=(-3.*x(1,j)+4.*x(2,j)-x(3,j))/2.
         yi=(-3.*y(1,j)+4.*y(2,j)-y(3,j))/2.
         h1(1,j)=sqrt(xi*xi+yi*yi)
         do 10 i=2,ni-1
            xi=(x(i+1,j)-x(i-1,j))/2.
            yi=(y(i+1,j)-y(i-1,j))/2.
            h1(i,j)=sqrt( xi*xi+yi*yi )
10       continue
         xi=(3.*x(ni,j)-4.*x(ni-1,j)+x(ni-2,j))/2.
         yi=(3.*y(ni,j)-4.*y(ni-1,j)+y(ni-2,j))/2.
         h1(ni,j)=sqrt( xi*xi+yi*yi )
```

```
5     continue
      do 15 i=1,ni
         xj=(-3.*x(i,1)+4.*x(i,2)-x(i,3))/2.
         yj=(-3.*y(i,1)+4.*y(i,2)-y(i,3))/2.
         h2(i,1)=sqrt( xj*xj+yj*yj )
         do 20 j=2,nj-1
            xj=(x(i,j+1)-x(i,j-1))/2.
            yj=(y(i,j+1)-y(i,j-1))/2.
            h2(i,j)=sqrt( xj*xj+yj*yj )
20       continue
         xj=(3.*x(i,nj)-4.*x(i,nj-1)+x(i,nj-2))/2.
         yj=(3.*y(i,nj)-4.*y(i,nj-1)+y(i,nj-2))/2.
         h2(i,nj)=sqrt( xj*xj+yj*yj )
15    continue
      return
      end

      subroutine cls
      write(*,10)
10    format('1')
      return
      end

      function omega( ajm,aim,ao,aip,ajp )

*********************************************************************
*                                                                   *
* This subroutine calculates the acceration factors for each        *
* grid point                                                        *
*                                                                   *
*********************************************************************

      complex u
      common cosx,cose
      data t/.3333333/
      f=aip*aim
      g=ajp*ajm
      u=2.*(csqrt( cmplx( f ) )*cosx+csqrt( cmplx( g ) )*cose)/ao
      ur=real( u )
      ui=aimag( u )
      aa=ur*ur+ui*ui
      aa2=aa*aa
      bb=ur*ur-ui*ui
      a=aa2-bb*bb
      b=aa2-bb
      c=sqrt( abs( a+b*b ) )
      e=(3.*b+c)*(abs(a*(c-b)))**t-(3.*b-c)*(abs(a*(c+b)))**t
      bar=(e+aa2+3.*bb*bb-4.*aa2*bb)/(aa2**2-aa2*bb)
      if ( aa2.gt.bb ) then
         omega=-(bar-sqrt( bar**2+4.*bar ))/2.
```

```
else
   omega=-(bar+sqrt( bar**2+4.*bar ))/2.
end if
return
end

function rel( er,unew,uold )
rel=amax1( er,abs( unew-uold )/(abs( unew )+1.))
return
end

function dif( er,unew,uold )
dif=amax1( er,abs( unew-uold ) )
return
end
```

Appendix H

Flux Deposition Algorithm

```
* --------------------------------------------------------------------- *
*                                                                       *
* PROGRAM:                                                              *
*         Calculates particle collection efficiencies.                  *
*                                                                       *
* DESCRIPTION:                                                          *
*                                                                       *
* This program calculates the trajectory of particles moving towards    *
* bodies of arbitrary shapes.  The grid representing the body was       *
* generated previously using the progran 'NETWORK.FOR'.                 *
*                                                                       *
* The particle trajectory can be effected by:                           *
*                                                                       *
*                         1.) inertial effects of the fluid             *
*                         2.) thermophoretic forces                     *
*                         3.) diffusiophoretic forces.                  *
*                                                                       *
*              By: Kevin Ellwood                                        *
*                  Dept. of Chem. Eng.                                  *
*                  University of Windsor                                *
*                  1987                                                 *
*                                                                       *
* --------------------------------------------------------------------- *
      program collect
      parameter ( pi=3.141593,r=8314. )
      dimension   x( 80, 80),  y( 80, 80)
      dimension  h1( 80, 80), h2( 80, 80), h3( 80, 80)
      dimension  v1( 80, 80), v2( 80, 80)
      dimension   t( 80, 80),  v( 80, 80)
      dimension vf1( 80, 80),vf2( 80, 80)
      dimension rp(20),z(4)
      real kf,kp,kfp,mf,md,lamb,nre,nsc,npr,nkn,nr
      equivalence ( h3,y ),( ri,z(1) ),( rj,z(2) )
      character input$*40,output$*40,grid$*40,flow$*40,temp$*40,vap$*40
      external state1,state2
      common /prop/nkn,nre,nsc,nr,st,kfp,tfd,ptot,pwf,ped,md,mf,cun
      common /grid/x,y,/gspeed/h1,h2,/flux/vf1,vf2,/speed/v1,v2
      common /terp/ni,nj

* Input the physical data for the system

      open( unit=1,file=' ' )
      read(1,10) grid$
      read(1,10) flow$
      read(1,10) temp$
      read(1,10) vap$
      read(1,10) output$
      read(1,*)  mf,denf,viscf,kf,cpf,tf,u0,rh,ptot,tcf,pcf
      read(1,*)  md,td,a0,tcd,pcd
      read(1,*)  denp,kp
      read(1,*)  nsize,(rp(i),i=1,nsize)
      read(1,*)  hmax,hmin,tol
      close( unit=1 )

* Read in grid data

      write(*,*) 'Reading grid data...........'
      open( unit=1,status='old',file=grid$,form='unformatted' )
      read(1) ni,nj,((x(i,j),i=1,ni),j=1,nj),
     &              ((y(i,j),i=1,ni),j=1,nj)
      close( unit=1 )

* Calculate grid speeds

      write(*,*) 'Calculating scale factors...'
      call scale( x,y,h1,h2,ni,nj )

* Read in flow data

      write(*,*) 'Reading flow data...........'
      open( unit=1,status='old',file=flow$,form='unformatted' )
      read(1) re,mn,jj,((t(i,j),i=1,mn),j=1,jj)
      close( unit=1 )

* Calculate velocity profile

      write(*,*) 'Calculating fluid velocity..'
      call velocity( ni,nj,t,h1,h2,h3,v1,v2 )

* Read in temperature data

      write(*,*) 'Reading temperature data....'
      open( unit=1,status='old',file=temp$,form='unformatted' )
      read(1) pe,mn,jj,((t(i,j),i=1,mn),j=1,jj)
      close( unit=1 )

* Read in vapor data

      write(*,*) 'Reading vapor data..........'
      open( unit=1,status='old',file=vap$,form='unformatted' )
      read(1) pe,mn,jj,((v(i,j),i=1,mn),j=1,jj)
      close( unit=1 )

* Calculate related physical data

      tav=(td+tf)/2.0
      lamb=viscf/denf*sqrt( pi*mf/2./r/tf )
      nre=2.*a0*denf*u0/viscf
      dwf=2.171e-5*(tav/273.)**1.75/ptot
      nsc=viscf/denf/dwf
      npr=cpf*viscf/kf
      psd=psat( td,tcd,pcd )
      psf=psat( tf,tcd,pcd )
      pwf=psf*rh/100.
      kfp=kf/kp
      tfd=tf/td

      open( unit=1,file=output$ )
      write(1,*) '========================================='
      write(1,*) '      ... DIMENSIONLESS GROOUPS ...'
      write(1,*) 'Reynolds No. ',nre
      write(1,*) ' Schmidt No. ',nsc
      write(1,*) ' Prandtl No. ',npr
      write(1,*) '========================================='
      write(1,*) '       ... FLUID DATA ... (units are MKS)'
      write(1,*) 'Molecular wt. ',mf
      write(1,*) 'Density   air ',denf
      write(1,*) 'Viscosity air ',viscf
      write(1,*) 'Thermo. cond. ',kf
      write(1,*) 'Heat capacity ',cpf
      write(1,*) 'Temperature   ',tf,' K'
      write(1,*) 'Relative hum. ',rh,' %'
      write(1,*) 'Path length   ',lamb
      write(1,*) 'Sat. press.   ',psf,' bar'
      write(1,*) 'Total press.  ',ptot,' bar'
      write(1,*) 'Diffusivity   ',dwf
      write(1,*) '========================================='
      write(1,*) '          ... DROPLET DATA ...'
      write(1,*) 'Molecular wt. ',md
      write(1,*) 'Temperature   ',td
      ~~write(1,*)~~ 'Sat. press.   ',psd,' bar'
      write(1,*) 'Volume radius ',a0
      write(1,*) 'Terminal vel. ',u0
      write(1,*) '========================================='
      write(1,*) '          ... PARTICLE DATA ...'
      write(1,*) 'Density       ',denp
      write(1,*) 'Thermo. cond. ',kp
      write(1,*) '_________________________________________'

* Set bounds on i coordinate

      do 25 i=1,ni
         if ( y(i,nj-1).gt.1.5 ) go to 40
25    continue

40    imax=i
      imin=1

* Calculate particle trajectories
```

```
    do 30 i=1,nsize
      h=(hmax+hmin)/2.
      nr=rp(i)/a0
      nkn=lamb/rp(i)
      alfa=1.257+0.4*exp(-1.1/nkn)
      cun=1.0+alfa*nkn
      st=cun*denp*u0*rp(i)**2/9./viscf/a0
      iflag=0
      if ( st.lt.0.006 ) iflag=1
      ris=float( imin )
      rib=float( imax )
      write(1,*) 'Radius          ',rp(i)
      write(1,*) 'Cunningham No.',cun
      write(1,*) 'Stokes No.      ',st

* Calculate flux velocities due to temperature and vapor fluxes

      write(*,*) 'Calculating flux velocity...'
      call vflux( t,v,h1,h2,ni,nj,vf1,vf2 )

50    trial=(rib-ris)/2.0+ris
      time=0.0
      rj=float( nj-1 )
      ri=trial
      call interp( y,ri,rj,ystart )
      effic=ystart**2
      write(*,*) 'Y-start      ',ystart
      write(*,*) 'Efficiency   ',effic
      if ( (rib-ris).le.0.001 ) go to 200
      npoints=1

* Assume the particle in initially moving with the gas (imbedded)

      call interp( v1,ri,rj,vp1 )
      call interp( v2,ri,rj,vp2 )
      call interp( h1,ri,rj,ha )
      call interp( h2,ri,rj,hb )

* Initialize particle contravariant velocity components

      x(3)=vp1/ha
      x(4)=vp2/hb

* Integrate the particle equations of motion over 1 time step

100   if ( iflag.eq.0 ) then

* Call with state eqations for large stokes numbers
```

```
            call rk5( state1,time,h,4,z,tol,hmax,hmin )
         else

* Call with state equations for small stokes numbers

            call rk5( state2,time,h,2,z,tol,hmax,hmin )
         end if

* Check for the condition where the particle crosses the
* axis of symetry.  If it has crossed then the symetry property
* is used to transpose the particle into the propper domain.

         if ( ri.gt.float( ni ) ) then
            ri=2.*float( ni )-ri
            z(3)=-z(3)
         end if
         npoints=npoints+1
         call interp( x,ri,rj,xx )
         call interp( y,ri,rj,yy )

* If the particle has not passed the collector and has not hit
* it then the integration is continued.

         if (npoints.gt.9999) then
            rj=0
            write(1,*) 'The particle was trapped'
         end if
         if ( ( rj.gt.1.0 ).and.( xx.lt.7.0 ) ) go to 100
         if ( rj.le.1.0 ) then
            ris=trial
         else
            rib=trial
         end if
         go to 50
200      write(1,*) 'Y-start        ',ystart
         write(1,*) 'Efficiency     ',effic
         write(1,*) '-------------------------------------------'
30    continue

      close( unit=1 )
      stop
10    format(a40)
      end

      subroutine scale( x,y,h1,h2,ni,nj )
      dimension x(80,80),y(80,80),h1(80,80),h2(80,80)

* This subrouitne calculates the scale facrors h1 & h2 from
* the orthogonal grid defined by x & y.
```

```
      do 5 j=1,nj
         xi=(-3.*x(1,j)+4.*x(2,j)-x(3,j))/2.
         yi=(-3.*y(1,j)+4.*y(2,j)-y(3,j))/2.
         h1(1,j)=sqrt(xi*xi+yi*yi)

         do 10 i=2,ni-1
            xi=(x(i+1,j)-x(i-1,j))/2.
            yi=(y(i+1,j)-y(i-1,j))/2.
            h1(i,j)=sqrt( xi*xi+yi*yi )
10       continue

         xi=(3.*x(ni,j)-4.*x(ni-1,j)+x(ni-2,j))/2.
         yi=(3.*y(ni,j)-4.*y(ni-1,j)+y(ni-2,j))/2.
         h1(ni,j)=sqrt( xi*xi+yi*yi )
5     continue

      do 15 i=1,ni
         xj=(-3.*x(i,1)+4.*x(i,2)-x(i,3))/2.
         yj=(-3.*y(i,1)+4.*y(i,2)-y(i,3))/2.
         h2(i,1)=sqrt( xj*xj+yj*yj )

         do 20 j=2,nj-1
            xj=(x(i,j+1)-x(i,j-1))/2.
            yj=(y(i,j+1)-y(i,j-1))/2.
            h2(i,j)=sqrt( xj*xj+yj*yj )
20       continue

         xj=(3.*x(i,nj)-4.*x(i,nj-1)+x(i,nj-2))/2.
         yj=(3.*y(i,nj)-4.*y(i,nj-1)+y(i,nj-2))/2.
         h2(i,nj)=sqrt( xj*xj+yj*yj )
15    continue

      return
      end

      subroutine vflux( t,v,h1,h2,ni,nj,vf1,vf2 )

* This subroutine calculates the flux velocity in general
* orthogonal coordinates.

* Note: 1.) grad( t ) = ( t,i /h1 , t,j /h2 )
*       2.) 2nd order central differencing is used except on the
*           coordinate boundaries where second order one sided
*           differencing is used.

      dimension  t( 80, 80), v( 80, 80),vf1( 80, 80),vf2( 80, 80)
      dimension h1( 80, 80),h2( 80, 80)
      real nkn,nre,nsc,kfp,md,mf,nr
      common /prop/nkn,nre,nsc,nr,st,kfp,tfd,ptot,pwf,ped,md,mf,cun
      data cm,ct/.75,2.16/
      at=-1.5*(kfp+ct*nkn)/(1.+2.*kfp+2.*ct*nkn)/(1.+3.*cm*nkn)

      do 5 j=1,nj
         ti=(-3.0*t(1,j)+4.0*t(2,j)-t(3,j))/2.0
         vi=(-3.0*v(1,j)+4.0*v(2,j)-v(3,j))/2.0
         phit=(1.-tfd)/(tfd+t(1,j)*(1.-tfd))
         pw=pwf+v(1,j)*(psd-pwf)
         yw=pw/ptot
         ad=-sqrt( md )/(yw*sqrt( md )+(1.-yw)*sqrt( mf ))
         phid=(psd-pwf)/(ptot-pw)
         vf1(1,j)=2.*(at*phit*ti+ad*phid*vi/nsc)/nre/h1(1,j)

         do 10 i=2,ni-1
            ti=(t(i+1,j)-t(i-1,j))/2.0
            vi=(v(i+1,j)-v(i-1,j))/2.0
            phit=(1.-tfd)/(tfd+t(i,j)*(1.-tfd))
            pw=pwf+v(i,j)*(psd-pwf)
            yw=pw/ptot
            ad=-sqrt( md )/(yw*sqrt( md )+(1.-yw)*sqrt( mf ))
            phid=(psd-pwf)/(ptot-pw)
            vf1(i,j)=2.*(at*phit*ti+ad*phid*vi/nsc)/nre/h1(i,j)
10       continue

         ti=(t(ni-2,j)-4.0*t(ni-1,j)+3.0*t(ni,j))/2.0
         vi=(v(ni-2,j)-4.0*v(ni-1,j)+3.0*v(ni,j))/2.0
         phit=(1.-tfd)/(tfd+t(ni,j)*(1.-tfd))
         pw=pwf+v(ni,j)*(psd-pwf)
         yw=pw/ptot
         ad=-sqrt( md )/(yw*sqrt( md )+(1.-yw)*sqrt( mf ))
         phid=(psd-pwf)/(ptot-pw)
         vf1(ni,j)=2.*(at*phit*ti+ad*phid*vi/nsc)/nre/h1(ni,j)
5     continue

      do 15 i=1,ni
         tj=(-3.0*t(i,1)+4.0*t(i,2)-t(i,3))/2.0
         vj=(-3.0*v(i,1)+4.0*v(i,2)-v(i,3))/2.0
         phit=(1.-tfd)/(tfd+t(i,1)*(1.-tfd))
         pw=pwf+v(i,1)*(psd-pwf)
         yw=pw/ptot
         ad=-sqrt( md )/(yw*sqrt( md )+(1.-yw)*sqrt( mf ))
         phid=(psd-pwf)/(ptot-pw)
         vf2(i,1)=2.*(at*phit*tj+ad*phid*vj/nsc)/nre/h2(i,1)

         do 20 j=2,nj-1
            tj=(t(i,j+1)-t(i,j-1))/2.0
            vj=(v(i,j+1)-v(i,j-1))/2.0
            phit=(1.-tfd)/(tfd+t(i,j)*(1.-tfd))
            pw=pwf+v(i,j)*(psd-pwf)
            yw=pw/ptot
            ad=-sqrt( md )/(yw*sqrt( md )+(1.-yw)*sqrt( mf ))
```

```
              phid=(psd-pwf)/(ptot-pw)
              vf2(i,j)=2.*(at*phit*tj+ad*phid*vj/nsc)/nre/h2(i,j)
20         continue

           tj=(t(i,nj-2)-4.0*t(i,nj-1)+3.0*t(i,nj))/2.0
           vj=(v(i,nj-2)-4.0*v(i,nj-1)+3.0*v(i,nj))/2.0
           phit=(1.-tfd)/(tfd+t(i,nj)*(1.-tfd))
           pw=pwf+v(i,nj)*(psd-pwf)
           yw=pw/ptot
           ad=-sqrt( md )/(yw*sqrt( md )+(1.-yw)*sqrt( mf ))
           phid=(psd-pwf)/(ptot-pw)
           vf2(i,nj)=2.*(at*phit*tj+ad*phid*vj/nsc)/nre/h2(i,nj)
15      continue

        return
        end

        subroutine velocity( ni,nj,s,h1,h2,h3,v1,v2 )

* This subroutine calculates the velocity vector curl( 0,0,s ) = (v1,v2,0)
* in general orthogonal axisymmetric coordinates.

* Note: 1.) ( v1,v2 ) = ( s,j /h2/h3 , -s,i /h1/h3 )
*       2.) 2nd order central differencing is used exept on the
*           coordinate boundaries where second order one sided
*           differencing is used
*       3.) l'hopitals rules is used on the boundary as both the
*           scale factor h3 and s,i vanishes.

        dimension h1( 80, 80),h2( 80, 80),h3( 80, 80)
        dimension v1( 80, 80),v2( 80, 80), s( 80, 80)

        do 5 j=1,nj
           sii=s(1,j)-2.0*s(2,j)+s(3,j)
           h3i=(-3.0*h3(1,j)+4.0*h3(2,j)-h3(3,j))/2.0
           v2(1,j)=-sii/h3i/h1(1,j)

           do 10 i=2,ni-1
              si=(s(i+1,j)-s(i-1,j))/2.0
              v2(i,j)=-si/h3(i,j)/h1(i,j)
10         continue

           sii=s(ni-2,j)-2.0*s(ni-1,j)+s(ni,j)
           h3i=(h3(ni-2,j)-4.0*h3(ni-1,j)+3.0*h3(ni,j))/2.0
           v2(ni,j)=-sii/h3i/h1(ni,j)
5       continue

        do 15 i=2,ni-1
           sj=(-3.0*s(i,1)+4.0*s(i,2)-s(i,3))/2.0
           v1(i,1)=sj/h2(i,1)/h3(i,1)
```

```
         do 20 j=2,nj-1
            sj=(s(i,j+1)-s(i,j-1))/2.0
            v1(i,j)=sj/h2(i,j)/h3(i,j)
20       continue

         sj=(s(i,nj-2)-4.0*s(i,nj-1)+3.0*s(i,nj))/2.0
         v1(i,nj)=sj/h2(i,nj)/h3(i,nj)
15    continue

      do 25 j=1,nj
         v1(ni,j)=0.0
         v1(1,j)=0.0
25    continue

      return
      end

      function psat( t,tc,pc )
      real k

* This function calculates saturation pressures for a given t

* tc and pc are critical pressure and temperature
* t and tc = Kelvin, psat is in units of pc

      k=-5.1514e-9*t**3+9.9533e-6*t*t-6.2442e-3*t+4.39553
      psat=pc*10**(k*(1.0-tc/t))
      return
      end

      subroutine interp( s,ri,rj,sij )

* This subroutine produces an interpolated value sij for the
* function s at the point ri,rj.

* Bilinear interpolation is used between the 4 corner points.

      dimension s( 80, 80)
      common /terp/ni,nj
      i=int( ri )
      if ( i.lt.1 ) i=1
      if ( i.ge.ni ) i=ni-1
      ip=i+1
      j=int( rj )
      if ( j.lt.1 ) j=1
      if ( j.ge.nj ) j=ni-1
      jp=j+1
      aj1=s(ip,j)-s(i,j)
      bj1=s(i,j)-float( i )*aj1
      aj2=s(ip,jp)-s(i,jp)
      bj2=s(i,jp)-float( i )*aj2
      aa=aj2-aj1
      ba=aj1-float( j )*aa
      ab=bj2-bj1
      bb=bj1-float( j )*ab
      a=aa*rj+ba
      b=ab*rj+bb
      sij=a*ri+b
      return
      end

      subroutine idot( s,ri,rj,si )

* This subroutine produces an interpolated value si for the
* derivative of the function s (ie. s,i ) at the point ri,rj.

* Bilinear interpolation is used between the 4 corner points.

      dimension s( 80, 80)
      common /terp/ni,nj
      i=int( ri )
      if ( i.lt.1 ) i=1
      if ( i.ge.ni ) i=ni-1
      ip=i+1
      j=int( rj )
      if ( j.lt.1 ) j=1
      if ( j.ge.nj ) j=ni-1
      jp=j+1
      aj1=s(ip,j)-s(i,j)
      aj2=s(ip,jp)-s(i,jp)
      a=aj2-aj1
      b=aj1-float( j )*a
      si=a*rj+b
      return
      end

      subroutine jdot( s,ri,rj,sj )

* This subroutine produces an interpolated value si for the
* derivative of the function s (ie. s,i ) at the point ri,rj.

* Bilinear interpolation is used between the 4 corner points.

      dimension s( 80, 80)
      common /terp/ni,nj
      i=int( ri )
      if ( i.lt.1 ) i=1
      if ( i.ge.ni ) i=ni-1
      ip=i+1
      j=int( rj )
      if ( j.lt.1 ) j=1
      if ( j.ge.nj ) j=ni-1
      jp=j+1
      ai1=s(i,jp)-s(i,j)
      ai2=s(ip,jp)-s(ip,j)
      a=ai2-ai1
      b=ai1-float( i )*a
      sj=a*ri+b
      return
      end

      subroutine rk5( vector,t,h,n,x,tol,hmax,hmin )
      real k(6,4),fac(5,5),ft(5),sx(4),xdot(4),x(4)

* Assigning data to dumy variables

      data ft/.25,.375,.9230769,1.0,0.5/
      data fac/.25,.09375,.879381,2.032407,-.2962963,0.,.28125,
     &     -3.277196,-8.,2.,2*0.,3.320892,7.173489,-1.381676,
     &     3*0.,-.2058967,.4529727,4*0.,-.275/
      data w1,w2,w3,w4,w5/2.777778e-3,-2.994152e-2,-2.919989e-2,
     &     0.02,3.636364e-2/
      data v1,v2,v3,v4,v5/.1185185,.5189864,.5061315,-.18,3.636364e-2/
      di=amod( x(1),1.0 )
      dj=amod( x(2),1.0 )
      d=amin1( di,dj,1.0-di,1.0-dj )
      tsav=t
5     nflag=0
      call vector( t,n,x,xdot )

      do 10 i=1,n
         sx(i)=x(i)
         k(1,i)=h*xdot(i)
10    continue

      do 20 j=2,6
         jm=j-1
         t=tsav+ft(jm)*h

         do 30 i=1,n
            x(i)=sx(i)

            do 30 l=1,jm
               x(i)=x(i)+fac(jm,l)*k(l,i)
30       continue

         call vector( t,n,x,xdot )

         do 20 i=1,n
```

```
            k(j,i)=h*xdot(i)
20        continue

          r=0.0

          do 40 i=1,n
             rr=abs( w1*k(1,i)+w2*k(3,i)+w3*k(4,i)+w4*k(5,i)+w5*k(6,i) )/h
             r=amax1( r,rr )
40        continue

          del=0.84*(tol/r)**.25
          if (( r.le.tol ).or.( d.lt.0.05 )) then
             nflag=1
             t=tsav+h

             do 50 i=1,n
                x(i)=sx(i)+v1*k(1,i)+v2*k(3,i)+v3*k(4,i)+v4*k(5,i)+v5*k(6,i)
50           continue

          end if
          if ( d.lt.0.05 ) return

* Try to adjust h

          if ( del.le.0.1 ) then
             h=h*.1
          else
             h=amin1( del*h,4.*h )
          end if
          if ( h.gt.hmax ) h=hmax
          if ( h.lt.hmin ) then
             write(*,60) hmin,h
          end if
          if ( nflag.eq.1 ) return

          do 70 i=1,n
             x(i)=sx(i)
70        continue

          t=tsav
          go to 5
60        format(' Warning.. hmin exceeded hmin:',f7.4,' h:',f7.4)
          end

          subroutine state1( t,n,z,zdot )

* This subroutine provides the 4 coupled o.d.e's that describe the
* particles trajectory in the orthogonal coordinate system represented
* by the vectors hi(i,j). ( Developed for large Stokes numbers )
```

```
      dimension  h1( 80, 80), h2( 80, 80)
      dimension  v1( 80, 80), v2( 80, 80)
      dimension vf1( 80, 80),vf2( 80, 80)
      dimension   z( 4),zdot( 4)
      real nkn,nre,nsc,nr,kfp,md,mf
      common /gspeed/h1,h2,/flux/vf1,vf2,/speed/v1,v2
      common /prop/nkn,nre,nsc,nr,st,kfp,tfd,ptot,pwf,psd,md,mf,cun
      data cs/1.0/
      ri=z(1)
      rj=z(2)
      z3=z(3)*z(3)
      z4=z(4)*z(4)
      z34=2.*z(3)*z(4)

* First, all necessary quatities must be obtained from nodal values
* by bilinear interpolation.

      call interp( h1,ri,rj,ha )
      call interp( h2,ri,rj,hb )
      call interp( v1,ri,rj,ua )
      call interp( v2,ri,rj,ub )
      call interp( vf1,ri,rj,vfa )
      call interp( vf2,ri,rj,vfb )
      call idot( h1,ri,rj,h1i )
      call idot( h2,ri,rj,h2i )
      call jdot( h1,ri,rj,h1j )
      call jdot( h2,ri,rj,h2j )
      vp1=z(3)*ha
      vp2=z(4)*hb
      hab=ha/hb
      st2=2.*st

* Evaluate derivatives

      zdot(1)=z(3)
      zdot(2)=z(4)
      zdot(3)=(-z3*h1i-z34*h1j+z4*h2i/hab+(cs*(ua-vp1)+vfa)/st2)/ha
      zdot(4)=(z3*h1j*hab-z34*h2i-z4*h2j+(cs*(ub-vp2)+vfb)/st2)/hb
      return
      end

      subroutine state2( t,n,z,zdot )

* This subroutine provides the 2 coupled o.d.e's that describe the
* particles trajectory in the orthogonal coordinate system represented
* by the vectors hi(i,j). ( Developed for st --> 0.0)

      dimension  h1( 80, 80), h2( 80, 80)
      dimension  v1( 80, 80), v2( 80, 80)
      dimension vf1( 80, 80),vf2( 80, 80)
      dimension   z( 4),zdot( 4)
      real nkn,nre,nsc,nr,kfp,md,mf
      common /gspeed/h1,h2,/flux/vf1,vf2,/speed/v1,v2
      common /prop/nkn,nre,nsc,nr,st,kfp,tfd,ptot,pwf,psd,md,mf,cun
      data cs/1.0/
      ri=z(1)
      rj=z(2)

* First, all necessary quatities must be obtained from nodal values
* by bilinear interpolation.

      call interp( h1,ri,rj,ha )
      call interp( h2,ri,rj,hb )
      call interp( v1,ri,rj,ua )
      call interp( v2,ri,rj,ub )
      call interp( vf1,ri,rj,vfa )
      call interp( vf2,ri,rj,vfb )

* Evaluate derivatives

      zdot(1)=(ua+vfa)/ha
      zdot(2)=(ub+vfb)/hb
      return
      end

      function density( t,p,tc,pc,mw )
      real mw
      data r/0.08315/
      a=27./64.*r*r*tc/pc
      b=r*tc/8./pc
      vo=r*t/p
10    vn=b+r*t/(p+a/vo/vo)
      er=abs( (vn-vo)/vn  )
      vo=vn
      if ( er.gt.0.0001 ) go to 10
      density=mw/vn
      return
      end

      function viscos( t )

* Viscosity of gas ( t = k ) ( viscosity = Kg/m/s )

      viscos=1.71601e-6*t**.4946-1.02968e-5
      return
      end
```

Appendix I

Solutions for Viscous Flow and Convective Diffusion Equations

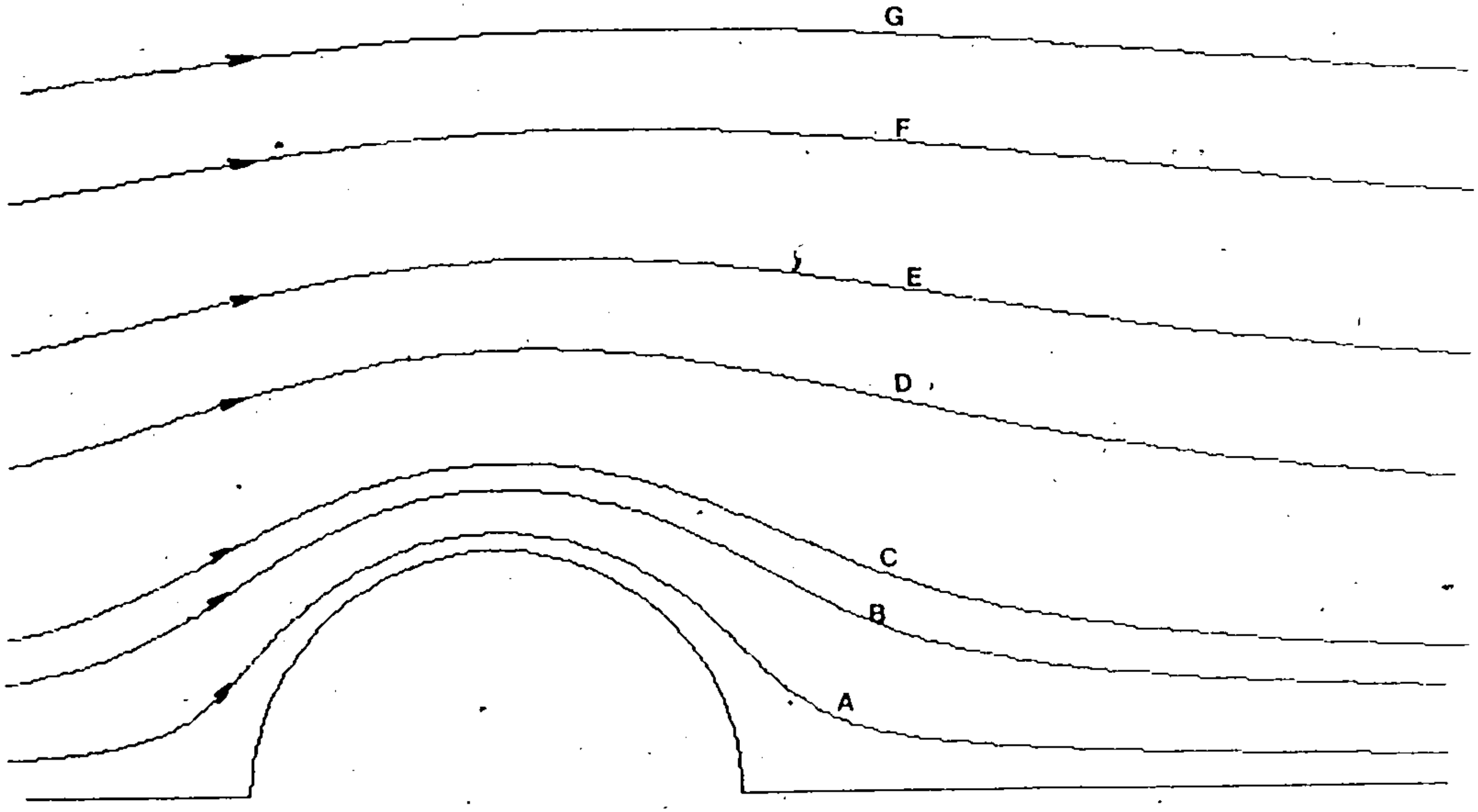

FIGURE I.1: Streamlines (ψ) for Reynolds Number of 1.54

Stream Function Values (A-G: .005, .05, .1, .5, 1, 2, 3)

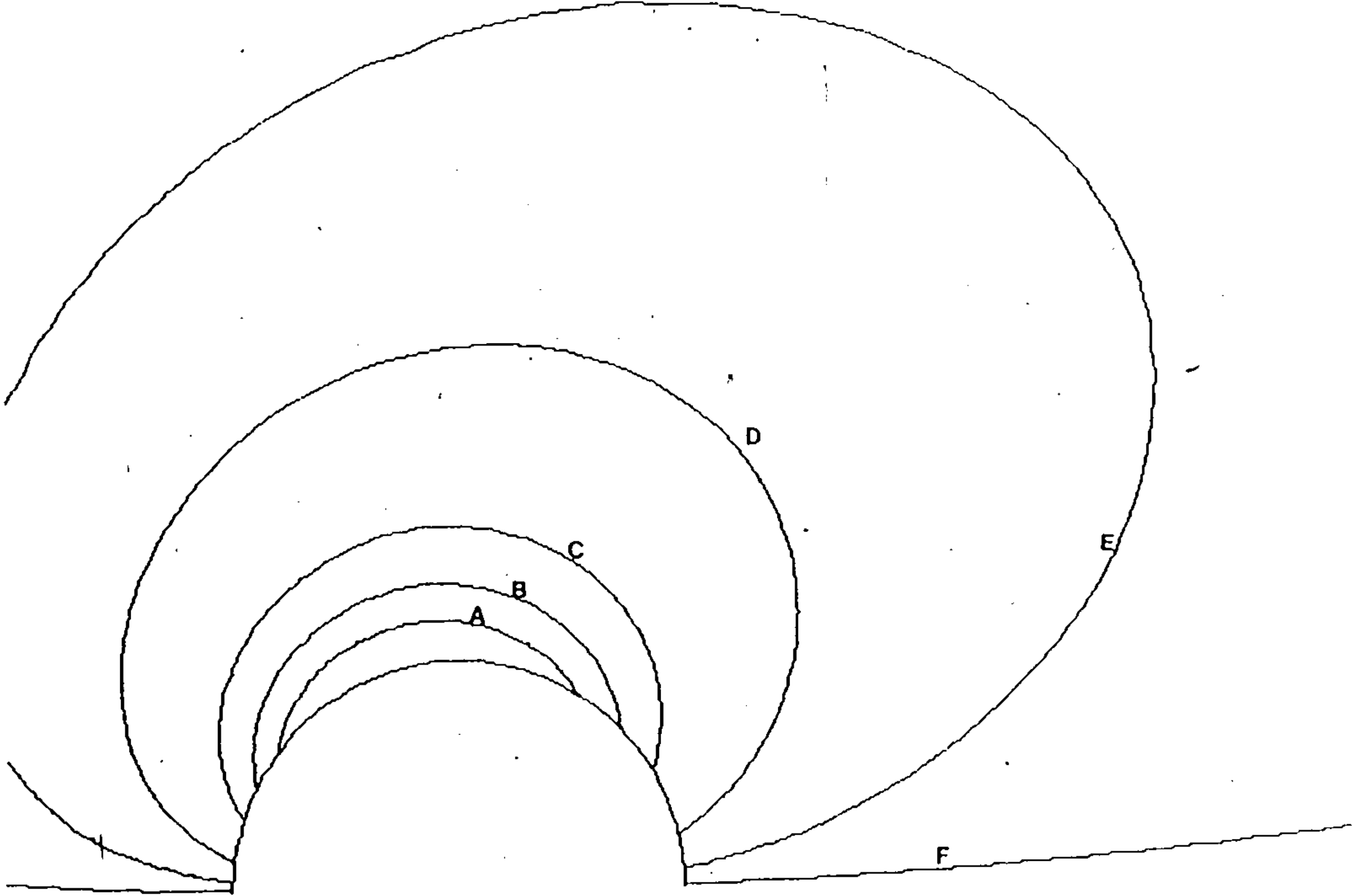

FIGURE I.2: Vorticity Contours (ζ) for Reynolds Number of 1.54

Vorticity Function Values (A-F: -1.3, -1, -.7, -.5, -.1, -.01)

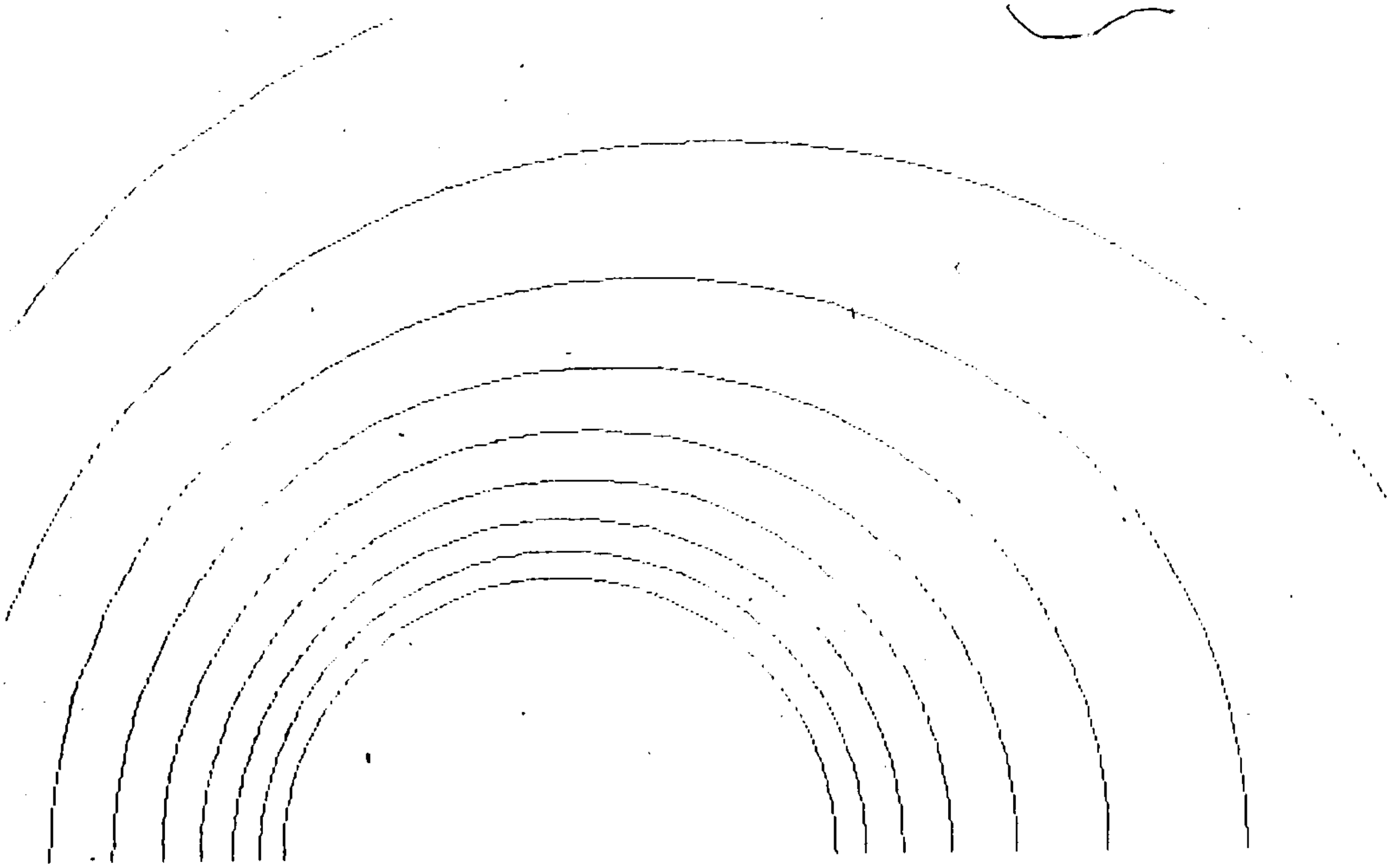

FIGURE I.3: Temperature/Vapor Contours (η) for Reynolds Number of 1.54

(10 Equal increments between 0 and 1)

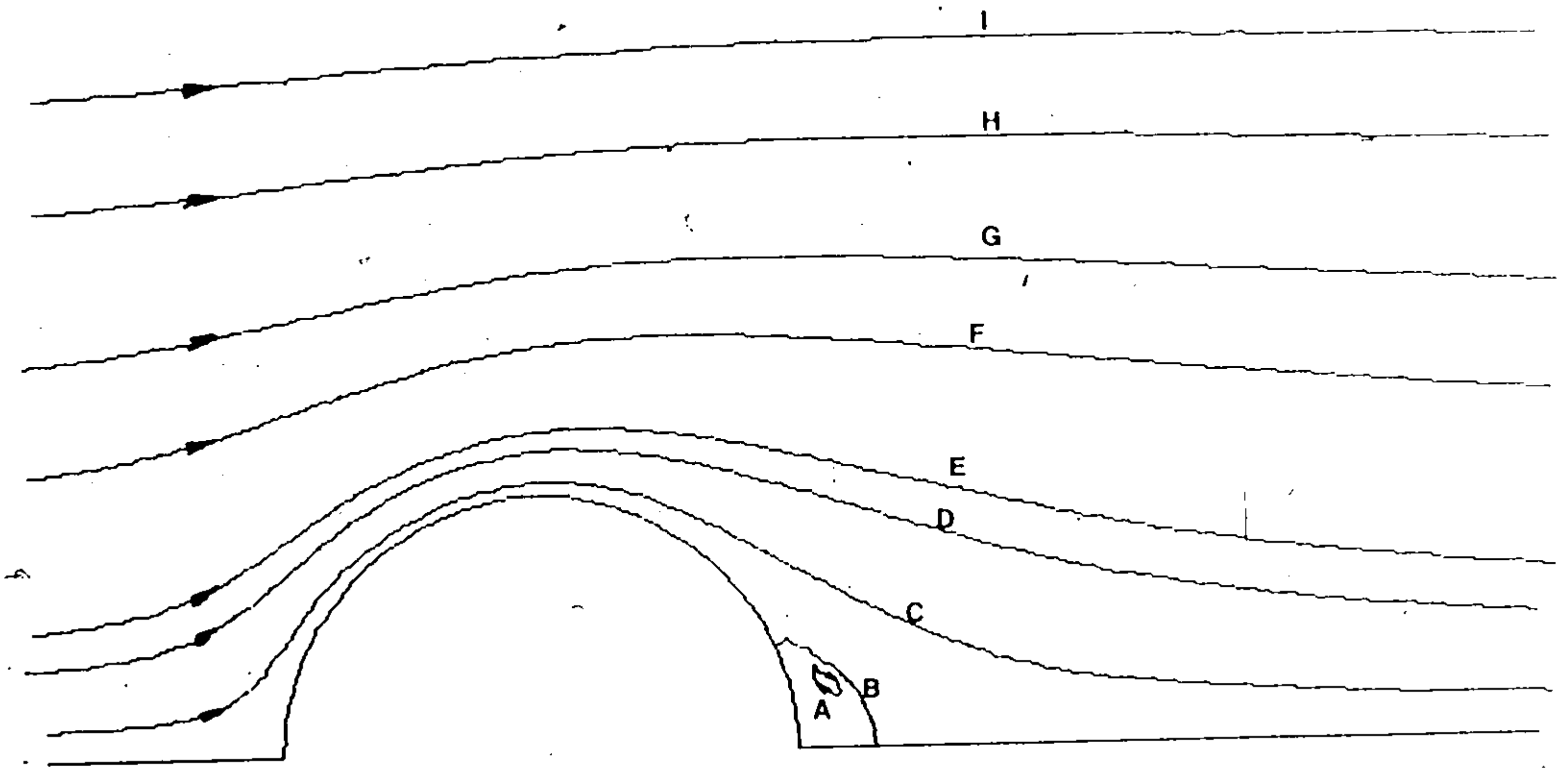

FIGURE I.4: Streamlines (ψ) for Reynolds Number of 30

Stream Function Values (A-I: -4.10^{-5}, 0.0, .005, 0.05, .1, .5, 1, 2, 3)

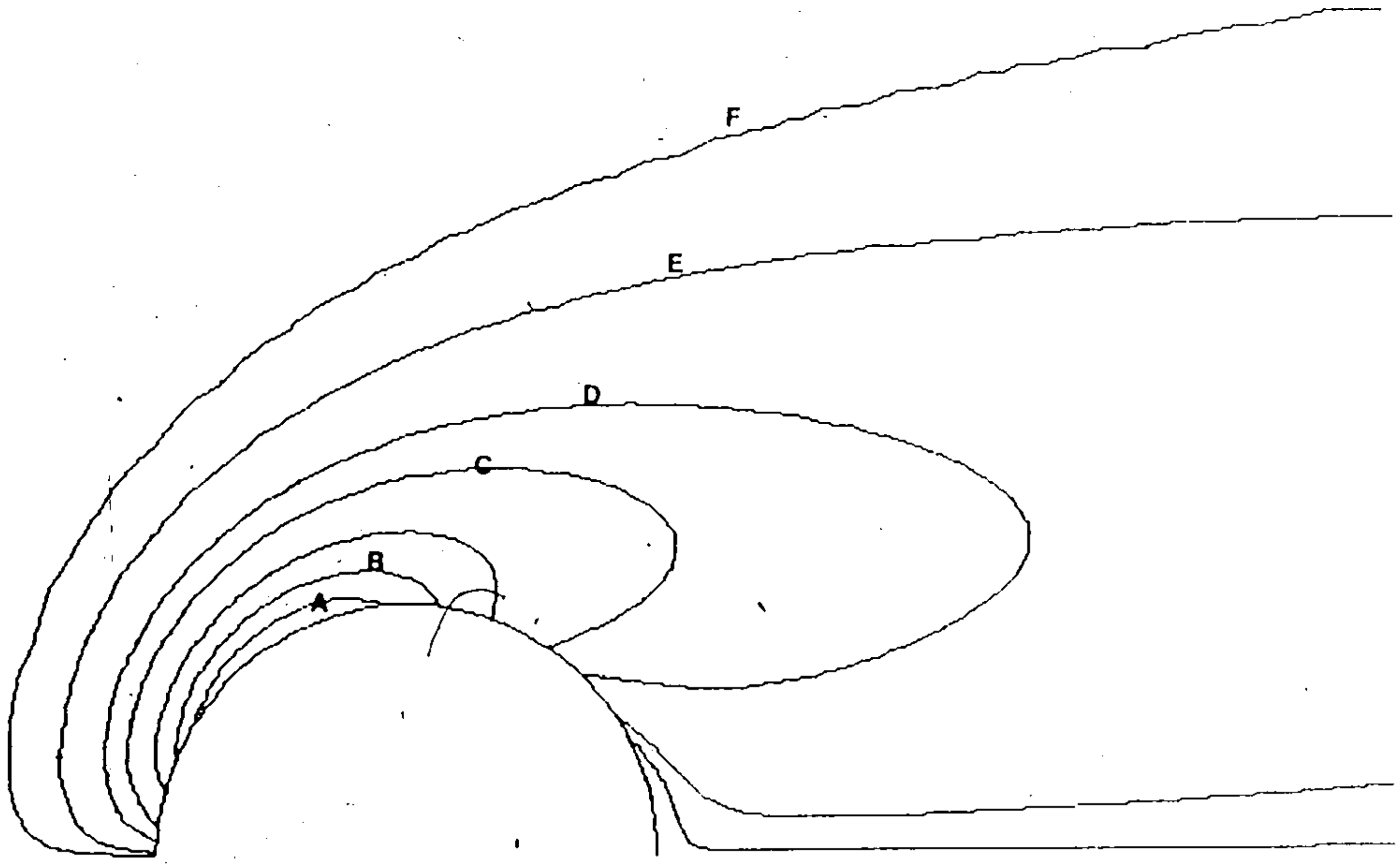

FIGURE I.5: **Vorticity Contours (ζ) for Reynolds Number of 30**

Vorticity Function Values (A-F: -4, -3, -1, -.5, -.1, -.01)

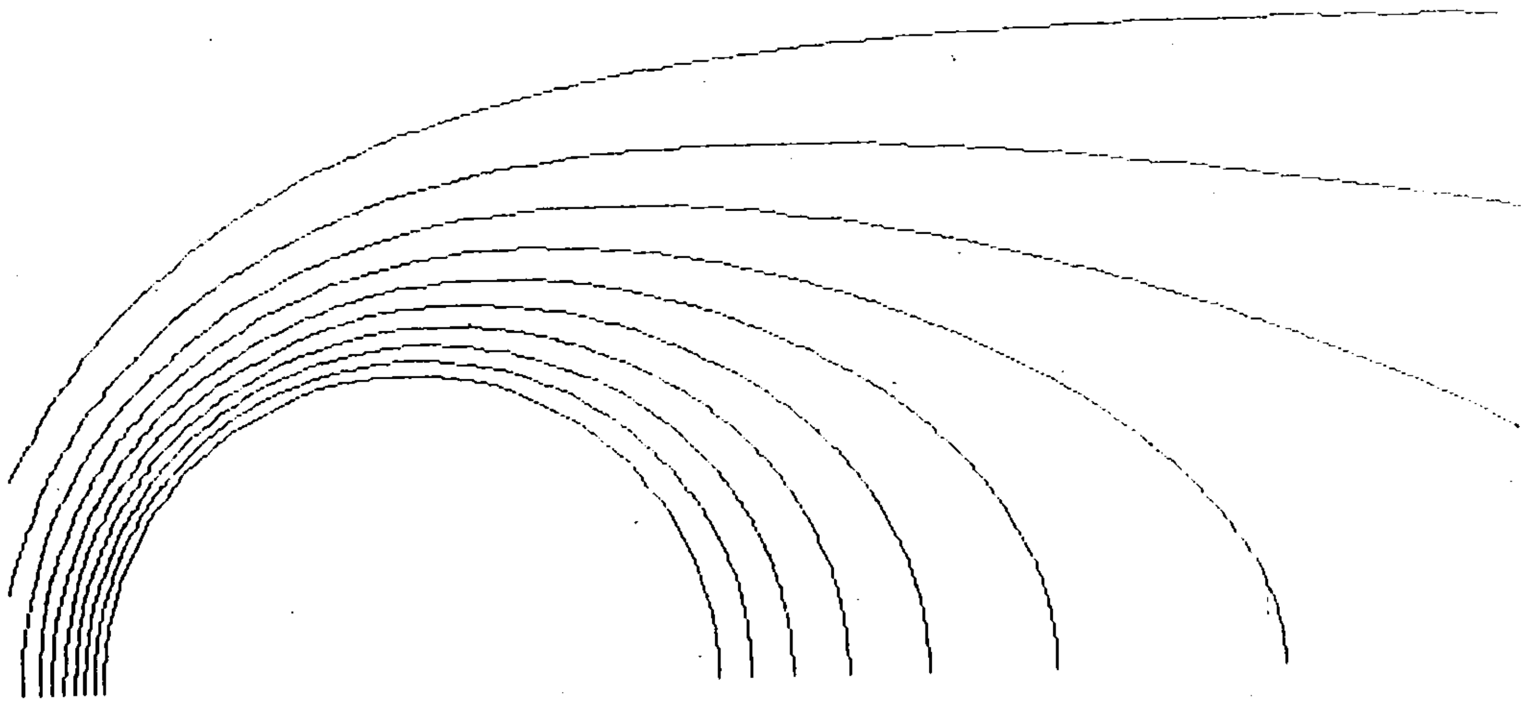

FIGURE I.6: Temperature/Vapor Contours (η) for Reynolds Number of 30

(10 Equal increments between 0 and 1)

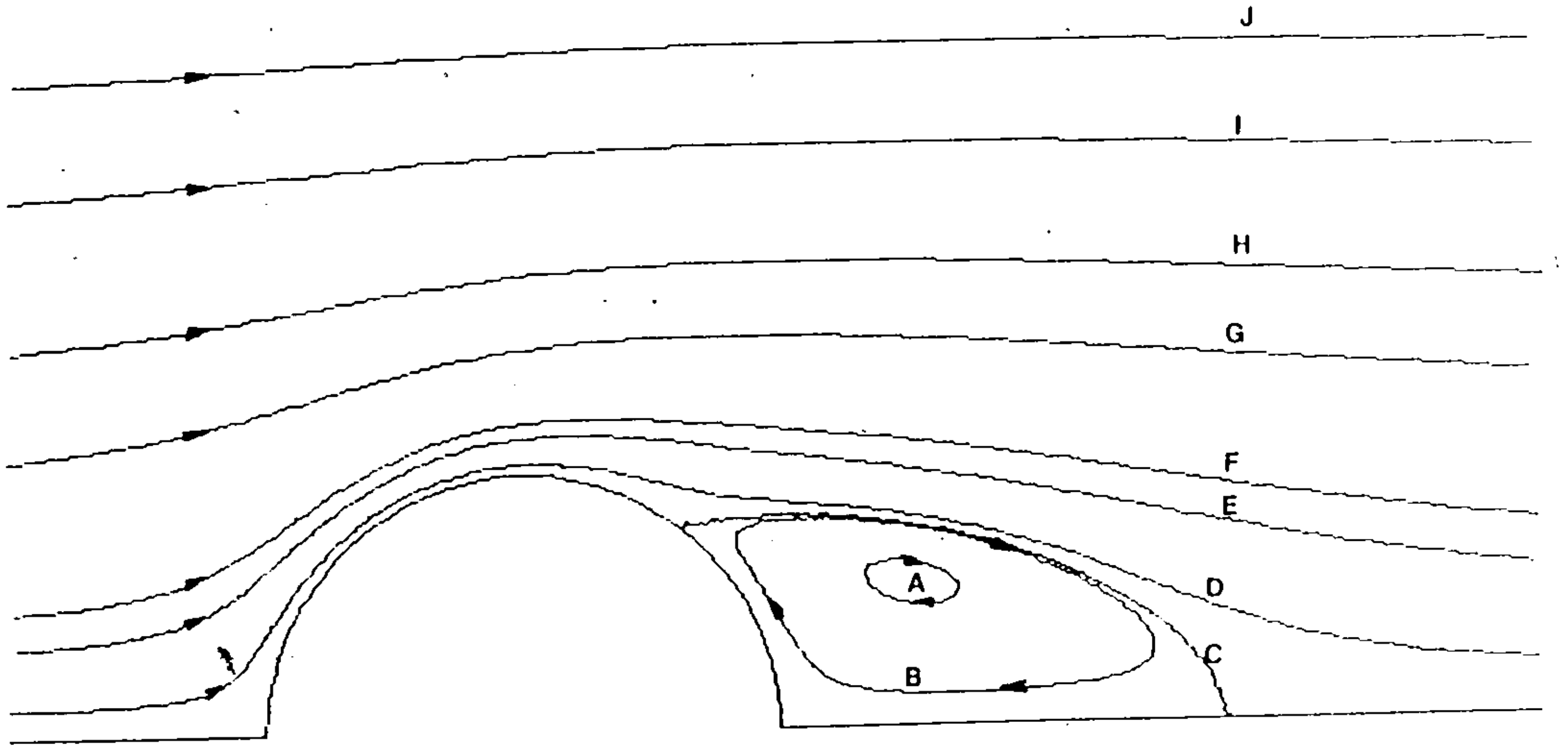

FIGURE I.7: **Streamlines (ψ) for Reynolds Number of 100**

Stream Function Values (A-J: -.01, -.001, 0.0, 0.005, 0.05, .1, .5, 1, 2, 3)

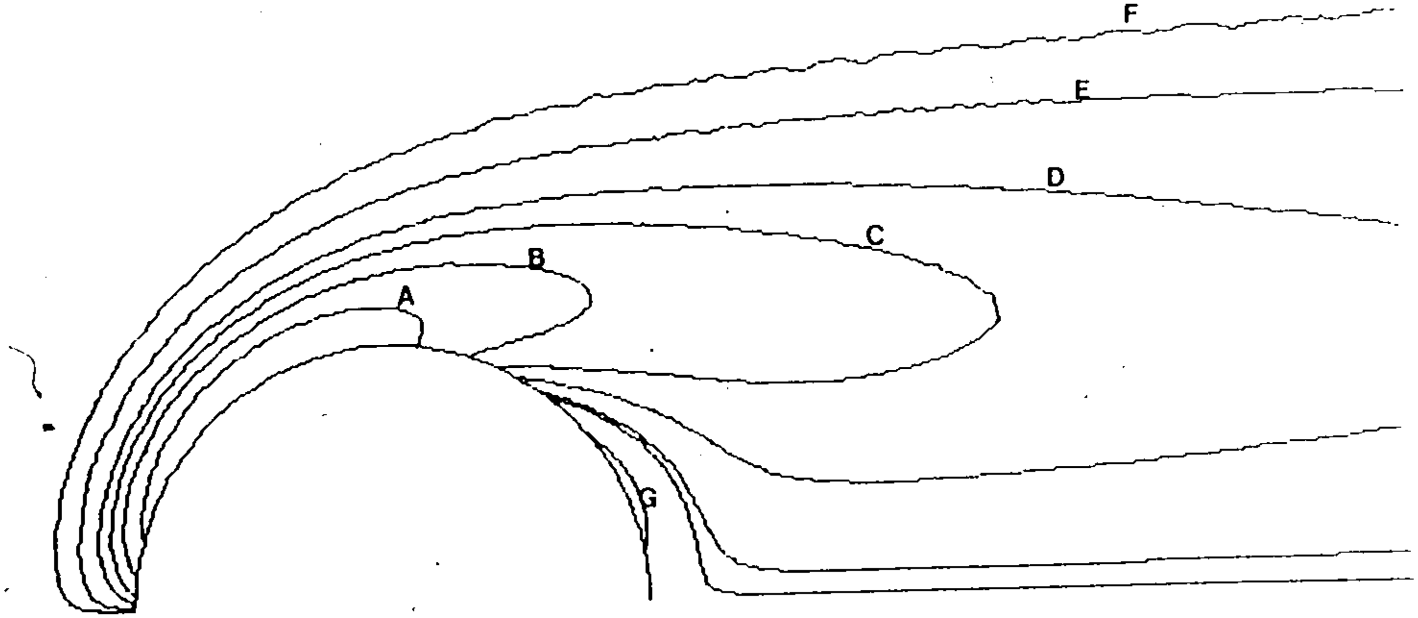

FIGURE I.8: Vorticity Contours (ζ) for Reynolds Number of 100

Vorticity Function Values (A-G: -4, -2, -1, -.5, -.1, -.01, .5)

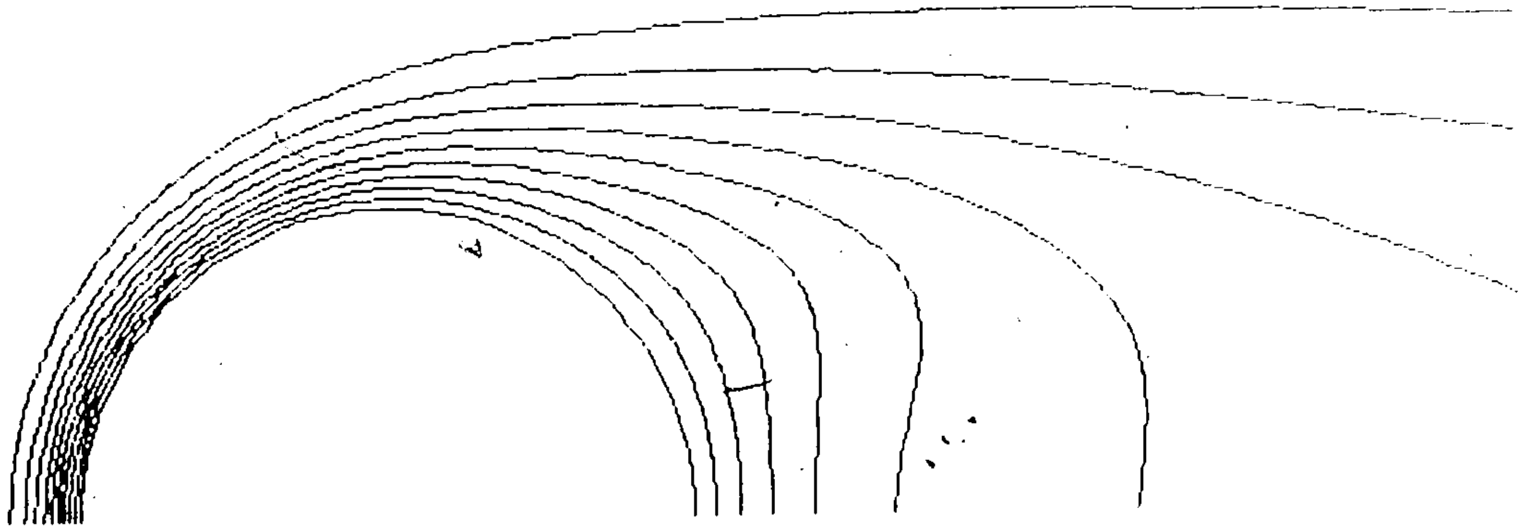

FIGURE I.9: Temperature/Vapor Contours (η) for Reynolds Number of 100

(10 Equal increments between 0 and 1)

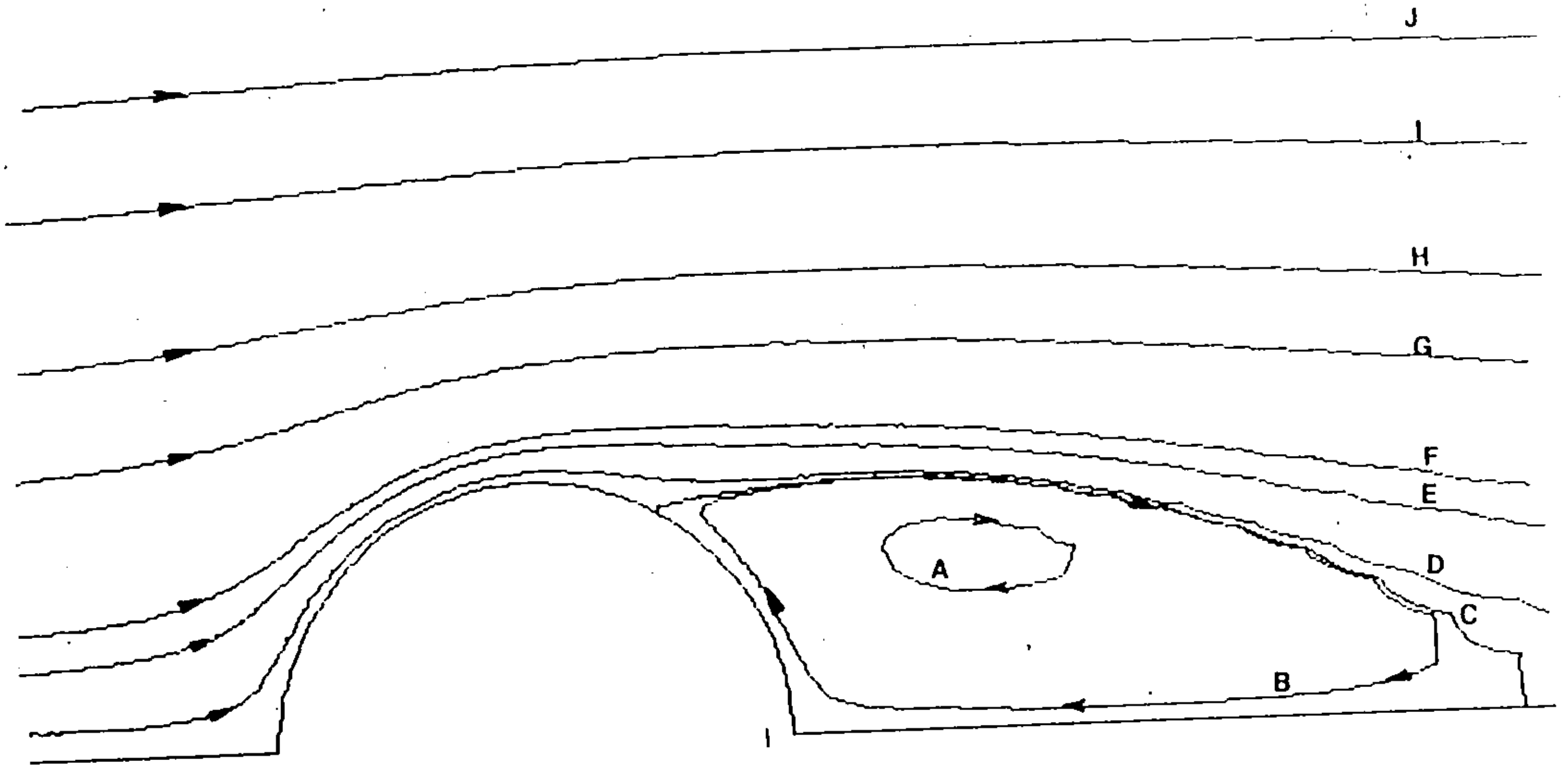

FIGURE I.10: Streamlines (ψ) for Reynolds Number of 200

Stream Function Values (A-J: -.03, -.001, .005, .05, .1, .5, 1, 2, 3)

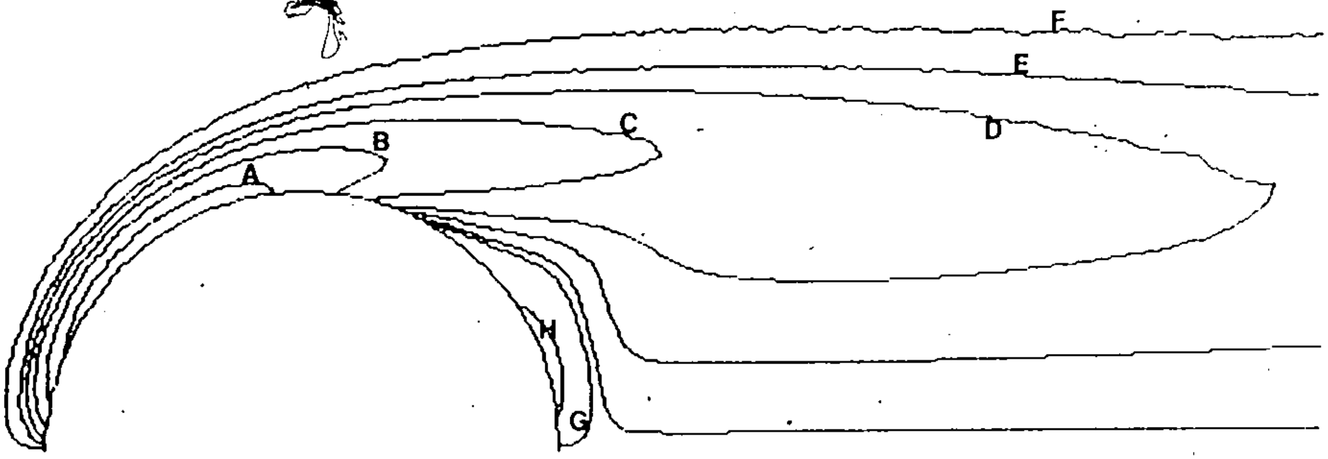

FIGURE I.11: Vorticity Contours (ζ) for Reynolds Number of 200

Vorticity Function Values (A-H: -8, -4, -2, -1, -.5, -.1, .1, 1)

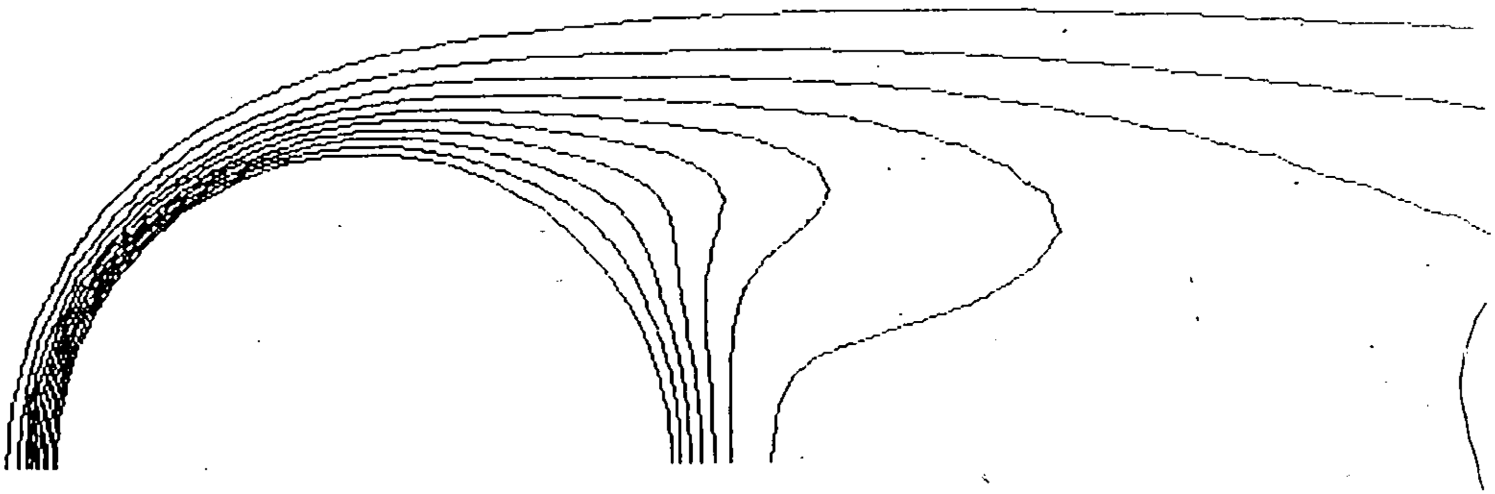

FIGURE I.12: Temperature/Vapor Contours (η) for Reynolds Number of 200

(10 Equal increments between 0 and 1)

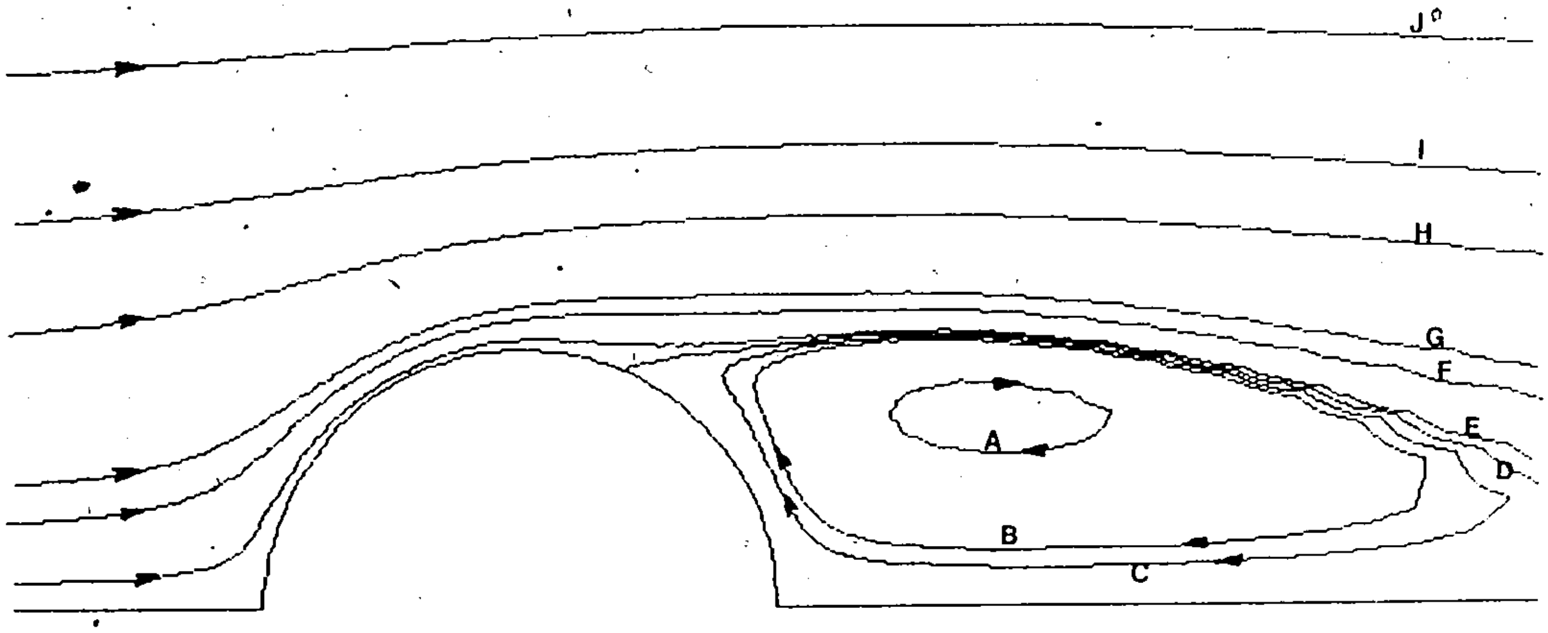

FIGURE I.13: Streamlines (ψ) for Reynolds Number of 300

Stream Function Values (A-J: -.05, -.01, 0.0, .005, .05, .1, .5, 1, 2)

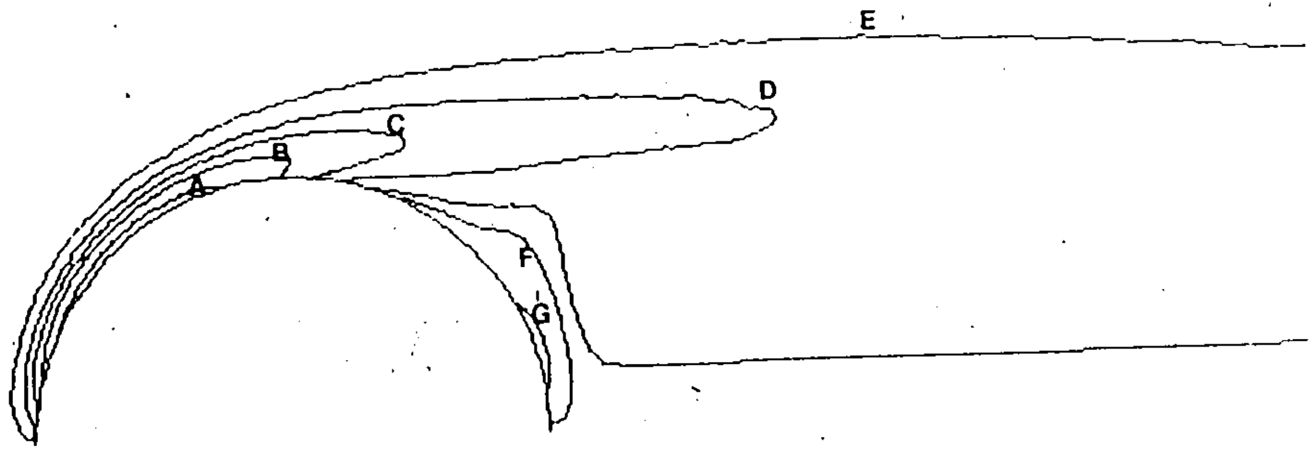

FIGURE I.14: Vorticity Contours (ζ) for Reynolds Number of 300

Vorticity Function Values (A–G: -12, -7, -4, -2, -.4, .3, 2)

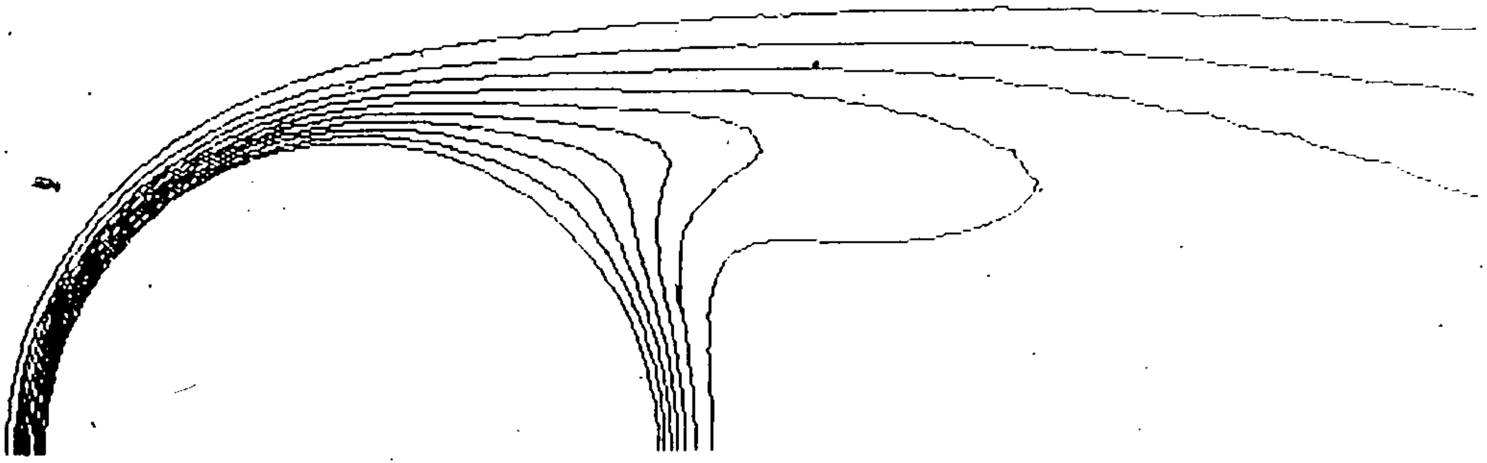

FIGURE 1.15: Temperature/Vapor Contours (η) for Reynolds Number of 300

(10 Equal increments between 0 and 1)

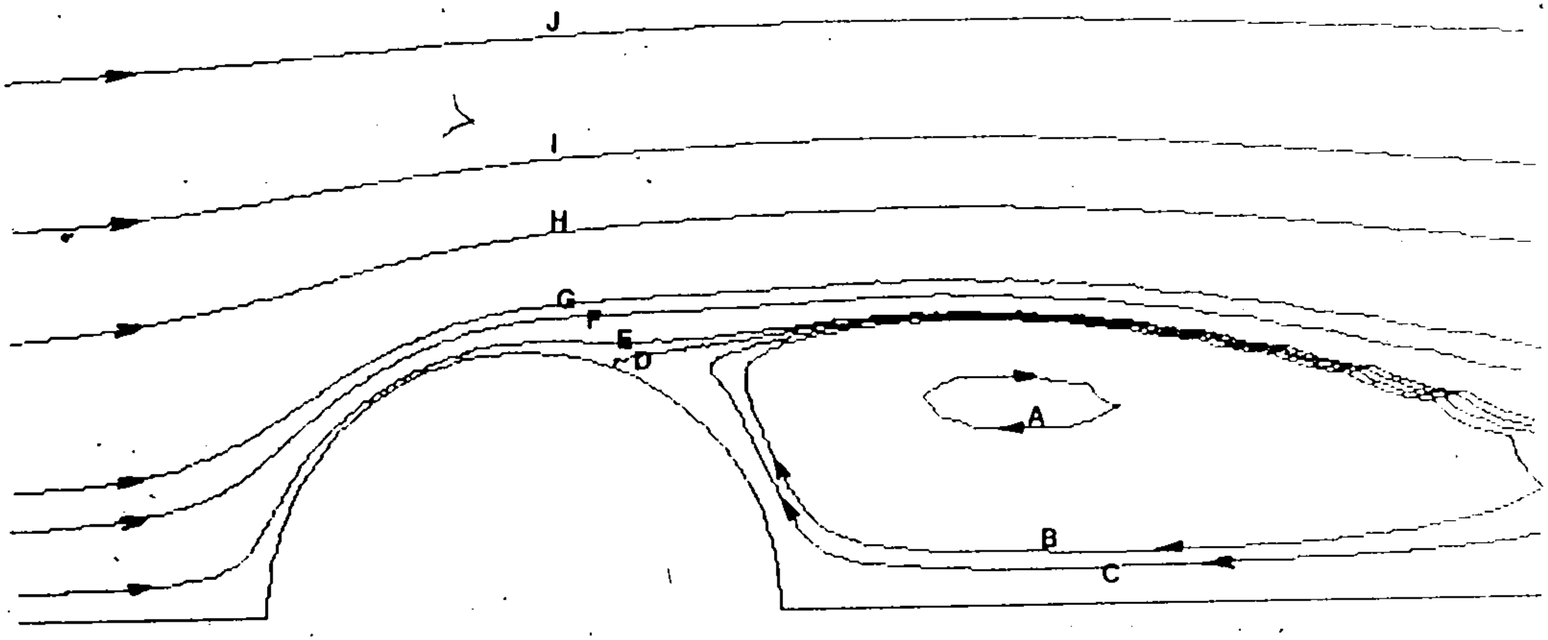

FIGURE I.16: Streamlines (ψ) for Reynolds Number of 400

Stream Function Values (A-J: -.07, -.01, -.005, 0.0, .005, .05, .1, .5, 1, 2)

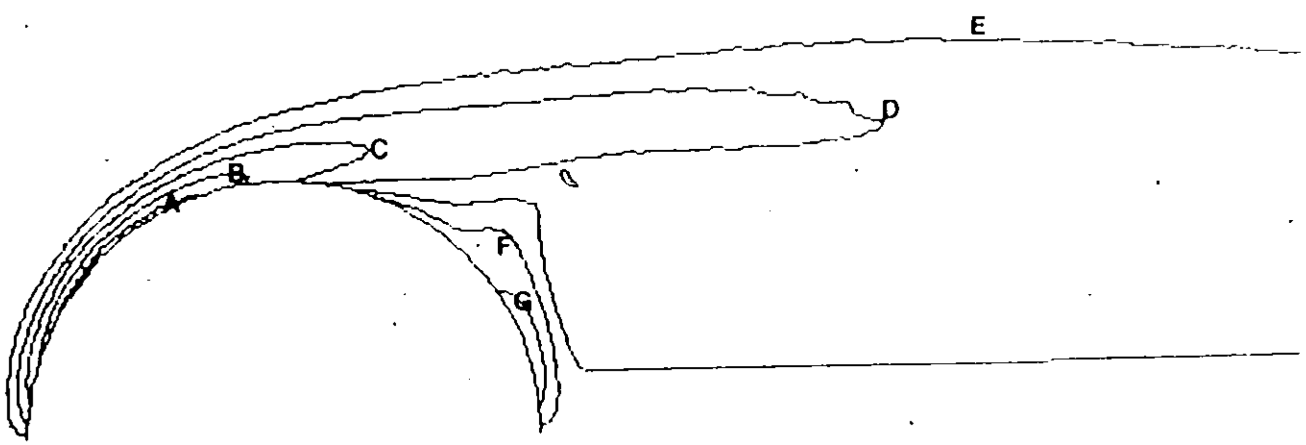

FIGURE I.17: Vorticity Contours (ζ) for Reynolds Number of 400

Vorticity Function Values (A–G: -15, -10, -5, -2, -.4, .5, 2)

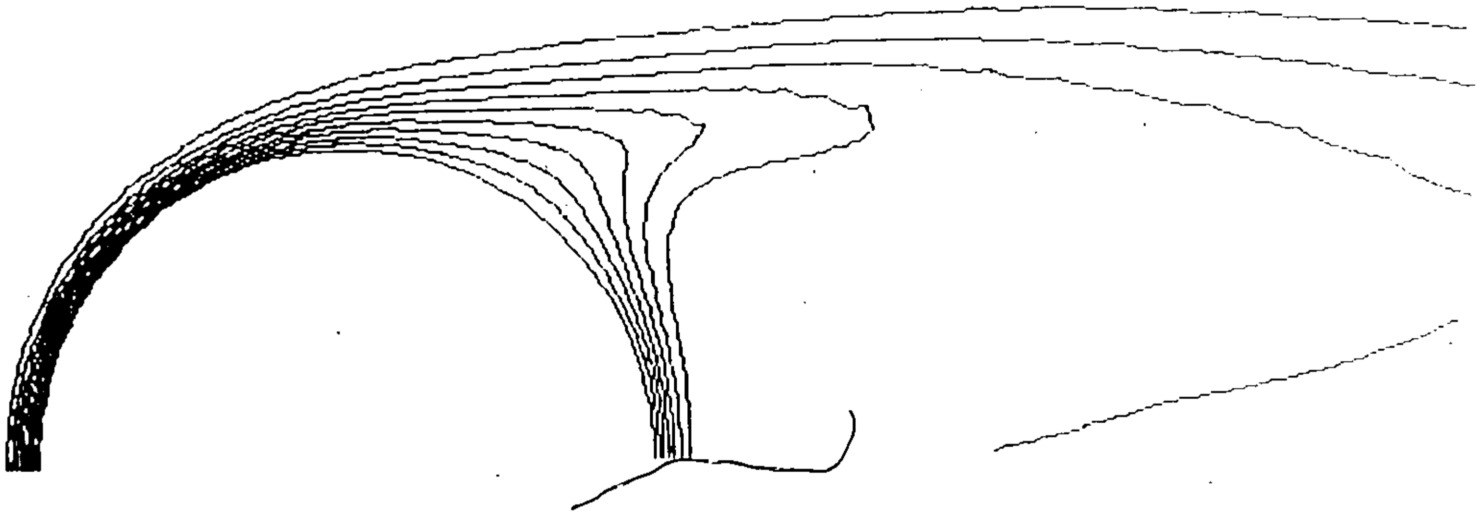

FIGURE I.18: Temperature/Vapor Contours (η) for Reynolds Number of 400

(10 Equal increments between 0 and 1)

VITA AUCTORIS

1962	Born in Windsor, Ontario, Canada, on January 17.
1981	Awarded Honours Secondary School Diploma from Vincent Massey Secondary School, Windsor, Ontario, Canada.
1985	Completed the Degree of Bachelor of Applied Science in Chemical Engineering at the University of Windsor, Windsor, Ontario, Canada.
1987	Candidate for the Degree of Master of Applied Science in Chemical Engineering at the University of Windsor, Windsor, Ontario, Canada.

AWARDS

1987	• NSERC Postgraduate Scholarship (PGS3) • Department of Chemical Engineering Fellowship at University of Michigan • Walter P. Murphy Fellowship at Northwestern University
1986	• NSERC Postgraduate Scholarship (PGS2)
1985	• NSERC Postgraduate Scholarship (PGS1) • Board of Governors Medal • Society of Chemical Industry Merit Award
1984	• Canadian Society for Chemical Engineering Prize • Reverend E.C. LeBel Award • Elizabeth Ellen Tilson Memorial Award • NSERC Summer Scholarship
1983	• NSERC Summer Scholarship
1982	• President's Role of Scholars • Guru P. Mather Memorial Award (Chem. Eng.)

PUBLICATIONS

Ellwood, K., A.W. Gnyp, C.C. St. Pierre, and S. Viswanathan, Development of Improved Single Drop Collection Efficiency Correlations for Microcomputer Modeling of Venturi Scrubber Performance, Proceedings: Sixth Symposium on the Transfer and Utilization of Particulate Control Technology, 1, (Nov. 1986).