SIMULATION OF ARBITRARY SHAPED BOUNDARIES FOR HEMODYNAMIC STUDIES

by

Roger K. DeBry

and

Harvey Greenfield

July 1973 UTEC-CSc-73-128

This research was supported in part by the University of Utah Computer Science Division and by the Advanced Research Projects Agency of the Department of Defense, monitored by Rome Air Development Center, Griffiss Air Force Base, New York 13440, under contract F30602-70-C-0300.

ACKNOWLEDGEMENTS

Appreciation is due to Dr. Steve Kelsey, Anthony Au and Dr. John Ahn for their suggestions and criticisms.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

PREFACE

Since the beginning of time, man has searched for answers to questions about himself and the world in which he lives. Out of this search have been born the many fields of science we know today. Most of these sciences had their beginnings with the ancient Babylonians, Egyptians, Phoenicians, and Greeks. By the 4th century B.C., Biology and Physics had begun, while Astronomy and Mathematics were well established sciences. Even applied sciences such as Medicine and Engineering had begun to emerge.

Often shrouded in mysticism, or dominated by religious and political thought, science seemed to advance or decline as a body. During periods of advancement in one field, similar gains are often found in others; progress in Astronomy and Biology, for example, was stimulated by the optical discoveries of the telescope and microscope. During the rise of Greek science, an interesting dichotomy appeared. Observation, which played a minor role in Astronomy and Physics, became prominent in Medicine. Hippocrates and Aristotle, the two Greeks of the period most interested in the life-sciences had little to do with Mathematics. Living things became exempt from mathematical treatment, and Medicine became the province of the non-mathematician [1]. Now, over 2000 years later, the medical scientist has found a very mathematical tool to aid him in his work - the digital computer.

Probably the first stored program electronic computer was the EDSAC developed at Cambridge University in 1949. Primarily designed to solve differential equations, early computers were strictly within

the realm of the mathematician. As computers became more powerful and easier to use, their use became more general in other disciplines. In 1956, the Air Research and Development Command of the U.S. Air Force sponsored an investigation to explore the possible uses of the computer in medical research [2].

In 1956, the National Aeronautics and Space Administration published a report on the uses of the computer in the medical field. By this time computers had been used in medical diagnosis, analog simulation of physical systems, handling of medical records, and teaching [3]. Since the time of that report, the use of the computer in medicine has been extended to include other administrative tasks, statistical analysis, and patient monitoring [4].

The use of the computer to understand the physical phenomena of the human body is still a relatively untouched field. There are many systems within the human body that cannot be completely understood using the physician's prime tool, observation. One of the most important of these is the human circulatory system. Although a great body of knowledge has been built up through the years, there are still a great many unknowns about the way blood flows through the body. Conventional means of measurement cannot answer these questions, because the act of introducing a measurement device into the system usually changes its characteristics.

By constructing a realistic mathematical model, the scientist can use the digital computer to see into previously unknown worlds, and find significant answers to problems regarding the circulatory system. The mathematician, with the aid of the digital computer, has

a great deal to offer the field of medicine. It is hoped that the kind of research represented here will be significant in bringing together these two disciplines.

ABSTRACT

The research presented here is part of a continuing project at the University of Utah, involved in the study of blood flow phenomena in the human body. These studies involve the solution of a system of non-linear, partial differential equations known as the Navier-Stokes equations. Analytic solutions to these equations exist only for a few special cases, therefore, numerical techniques have been developed for approximating them. A particular requirement for hemodynamic studies is that a solution technique exists which allows flexible, arbitrarily shaped, no-slip boundaries to be defined in the model being studied. The technique which is developed in this report allows that kind of definition. It is a finite difference technique, based on the Marker and Cell method developed at Los Alamos Scientific Laboratory.

Because such methods are typically large and cumbersome, the efficient implementation of this new technique was taken on as a concurrent investigation. The use of interactive computer graphics provides the user with the best approach to an efficient implementatopn, if it is applied in the right way.

Techniques for applying interactive computer graphics to the solution of hemodynamics problems involving flexible, arbitrarily

---

*This report reproduces a dissertation of the same title submitted to the Department of Electrical Engineering, University of Utah in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

shaped, no-slip boundaries are demonstrated for a particular problem. This problem involves the study of blood flow through a ball-type prosthetic heart valve. The results of this study agree closely with known experimental studies.

CHAPTER ONE

INTRODUCTION

The research reported on here is part of an on-going project at the University of Utah, committed to the study of blood flow phenomena in the human circulatory system [5]. Topics of current interest in this area include the formation of atherosclerotic plaques, and the problems associated with blood flow through artificial heart valves [6]. The numerical solution of this type of problem involves the solution of a system of non-linear partial differential equations. This set of equations, known as the Navier-Stokes equations is extremely difficult to solve. Analytic solutions exist for only a few well known special cases, where the proper assumptions reduce the equations to a solvable form.

In recent years, numerical techniques have been developed which, with the aid of the digital computer, can find solutions to a broader class of fluid problems. Even these techniques however, take advantage of special cases: straight walls, two-dimensional flow, symmetry, etc. They are typically finite difference techniques, and as such require large amounts of computer storage and large execution times. As the problem becomes more and more complex, the storage and run time requirements of the problem grow very, very fast. For example, in a recent work reported on by Hirt and Cook [7], a three-dimensional problem of limited resolution (15 computing cells on a side) required 64,000 words of high speed store on a CDC-7600 computer.

Although numerical techniques are becoming more powerful and more sophisticated, the trend seems to be for these techniques to rely heavily on the brute-force speed and large memory capacity of today's super-computers. They tend to neglect the more elegant features available on modern computing systems. It would seem that a certain level of sophistication would allow such problems to be run much more effectively. This does not mean that the goal here is to produce a faster piece of code, but rather one which is more efficient in its use of total machine resources. One way that this can be done is to give the user the capability to visually interact with his program.

Interactive computer graphics is certainly not a magic formula which will make a computation easier. In fact, if poorly designed, a programming system using interactive computer graphics may prove to be a serious handicap. Therefore, in suggesting the inclusion of an interactive graphics capability, the previously mentioned goal of program efficiency must be kept in mind.

The class of problems to be considered here involves the study of blood flow past artificial heart valves. In order that the problem be studied with as few simplifying assumptions as possible, a numerical technique of considerable power will be used. In addition, interactive computer graphics will be included as an integral part of the solution process. Considerable detail will be focused on keeping the entire program efficient in terms of the use of machine resources.

CHAPTER TWO

THE PHYSICAL PROBLEM

The heart is basically a four chambered pump. The major pumping muscle of the heart is the left ventricle, which supplies oxygenated blood to the entire body. The inflow valve to the left ventricle is the mitral valve, and it's outflow valve is the aortic valve. It is one of these two valves which is most often destroyed or impaired genitically or by disease, even though two other valves exist in the right side of the heart.

In the past few years, successful replacement of the natural valve with an artificial one has brought about the development of many different designs for artificial heart valves. Unfortunately, no one valve completely fulfills all of the criteria for an ideal replacement to the natural valve [8]. By studying the flow characteristics of currently available prosthetic heart valves, the results could provide valuable insight into the design of a better valve.

The physical model chosen to study first was the Starr-Edwards ball type valve. Because this is a very popular replacement for the natural aortic valve, some detailed study of its flow characteristics seems to be in order. The environment for this model is the heart valve test chamber developed by Weiting at the University of Texas [9]. This provides experimental results for later comparison and evaluation for the chamber was carefully designed, according to data derived from cadaver heart measurements.

The valve testing chamber shown in Figure 1 was designed by

Wieting such that the cross-sectional flow areas were analogous to those in the human heart.

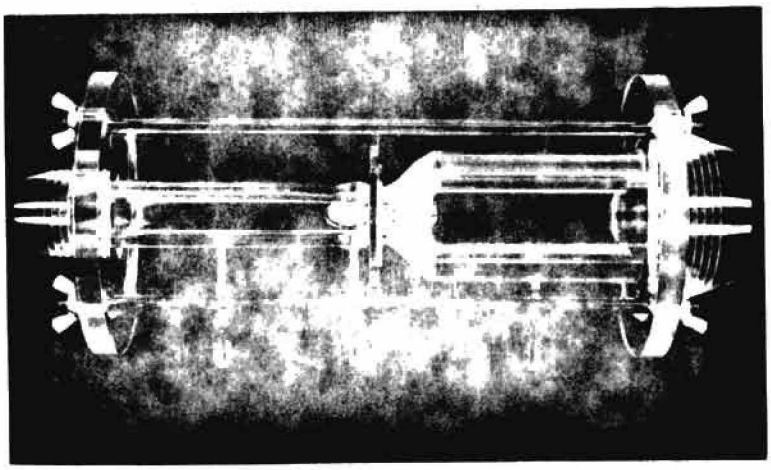


Figure 1

Wieting Heart Valve Rest Chamber

A cross-sectional view of the chamber is shown in Figure 2. In this drawing, a ball type valve such as the one to be studied is shown in the open position.

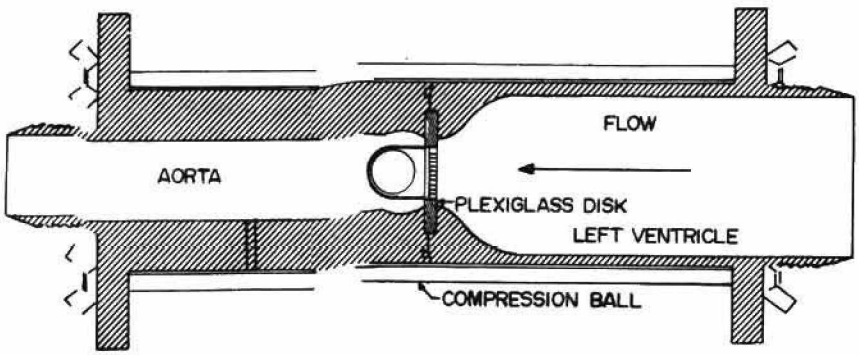


Figure 2

Cross Section of Chamber

Placing this cross section in an r-z reference frame, as shown in Figure 3, the coordinates of points along the wall of the chamber can be determined. These points then provide a description of the boundaries for the computer program. This list of points is contained in Table I.

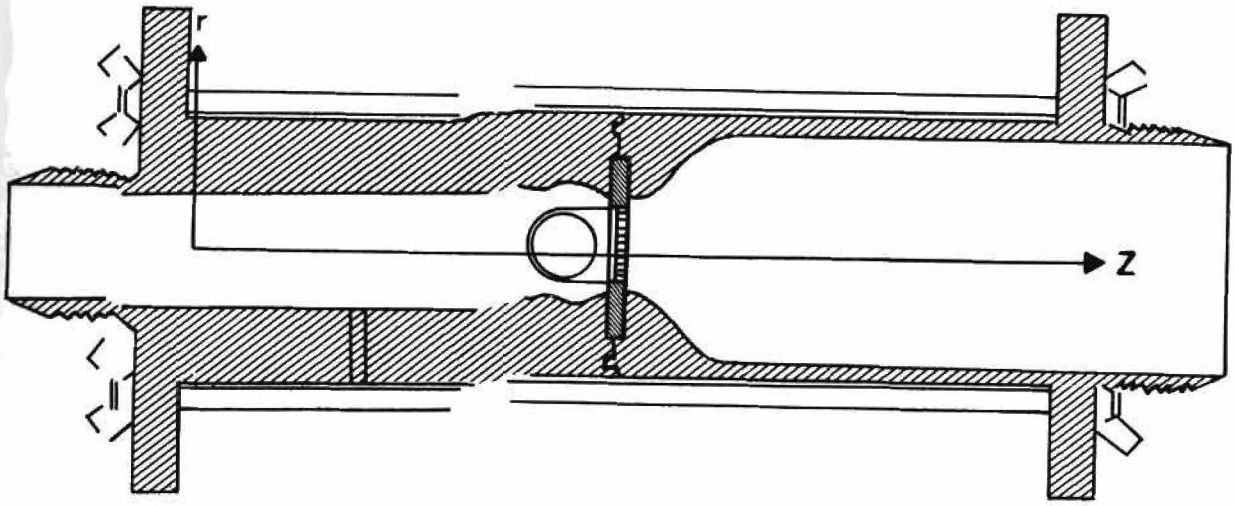


Figure 3

Coordinate Axes Used to Obtain Computer Input for Test Chamber Description

TABLE I

INPUT POINTS FOR BOUNDARY DESCRIPTION

COORDINATES IN MM

R	Z
0	0
1.7	0
1.7	6.8
1.58	7.31
1.56	7.56
1.58	7.82
1.62	8.07
1.736	8.53
1.736	8.78
1.683	9.04
1.524	9.42
0.89	9.48
0.846	9.74
0.90	9.99
1.397	10.10
1.46	10.35
1.65	10.73
2.984	12.13
3.175	12.51
3.2	12.89

TABLE I

Continued

R	Z
3.2	16.0
0	16.0
0	9.255
0.198	9.233
0.388	9.171
0.561	9.072
0.700	8.938
0.827	8.777
0.908	8.595
0.949	8.399
0.949	8.209
0.908	8.105
0.827	7.823
0.700	7.662
0.561	7.528
0.388	7.429
0.198	7.367
0	7.345
0	0

CHAPTER THREE

THE DEVELOPMENT OF THE MATHEMATICAL MODEL

A. THE EQUATIONS OF FLUID FLOW

The mathematical study of fluid dynamics is based upon the principles of conservation of mass, momentum and energy. These principles can be expressed in the form of partial differential equations. Derivation of these equations are found in standard fluid dynamics texts such as those by Lightfoot [10] or Batchelor [11]. For simple two-dimensional flow they are:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{1}$$

$$\rho \left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} \right) + \frac{\partial p}{\partial x} = 0, \tag{2}$$

$$\rho \left(\frac{\partial u}{\partial t} + \frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} \right) + \frac{\partial p}{\partial y} = 0. \tag{3}$$

Equation (1) is the continuity equation, and equations (2) and (3) are the Navier-Stokes equations for two-dimensional incompressible flow. In these equations, u and v are velocities in the x and y directions respectively, ρ is the density of the fluid, and p the pressure in the fluid.

Since blood is a viscous fluid, in our particular study involving incompressible flow, we must consider the effects of viscosity. This results in the set of equations:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{4}$$

$$\rho \left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} \right) + \frac{\partial p}{\partial x} = -\mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{5}$$

$$\rho \left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) + \frac{\partial p}{\partial y} - \mu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \tag{6}$$

where μ is the viscosity of the fluid.

Dividing equations (5) and (6) through by ρ, we get:

$$\frac{\partial u}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial \emptyset}{\partial x} = -\nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{7}$$

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial \emptyset}{\partial y} = -\nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right). \tag{8}$$

Here $\nu = \mu/\rho$, the kinematic viscosity of the fluid, and $\emptyset = p/\rho$.

In the more familiar vector form these equations are:

$$\nabla \cdot \vec{u} = 0 \tag{9}$$

$$\frac{\partial \vec{u}}{\partial t} = (\vec{u} \cdot \nabla)\, \vec{u} - \nabla \emptyset + \nu \nabla^2 \vec{u}. \tag{10}$$

The motion of the fluid is described in equation (10) in terms of:

1. $(\vec{u} \cdot \nabla)\vec{u}$ — The convection of momentum by fluid motion.
2. $-\nabla \emptyset$ — The momentum change due to pressure forces.
3. $\nu \nabla^2 \vec{u}$ — The diffusion of momentum by viscous forces.

Equations of this kind cannot be solved analytically. However, approximation methods have been devised which, with the aid of the modern computer, can realize extremely good solutions to fluid problems. One class of such methods is that of finite differences. The use of such techniques is discussed at length in Forsythe and Wasow [12], and Varga [13]. The techniques used by the present investigator are of this class.

B. FINITE DIFFERENCE TECHNIQUES

Fundamental to most numerical techniques for the solution of partial differential equations is the fact that the derivatives of a function can be expressed in terms of values of the function or differences in values of the function at different points in the region over which the function is defined. Given an arbitrary function $U = U(X)$, and assuming that U possesses a sufficient number of derivatives, the value of U at two points X and X + h can be related by the Taylor's series expansion

$$U(X+h) = U(X) + U'(X)\,h + \ldots$$

$$\ldots + \frac{U^{(n-1)}(X)}{(n-1)!}\,h^{n-1} + \frac{h^n}{n!}\,U^{(n)}(X_1) \qquad (11)$$

where X_1 lies between X and X+h. As h, the incremental distance between these two arbitrary points, is allowed to become very small, the high order terms of the expansion approach zero, and can be

lumped together as an error term. Now expand U_{i+1} and U_{i-1} in a Taylor series expansion about the central point U_i.

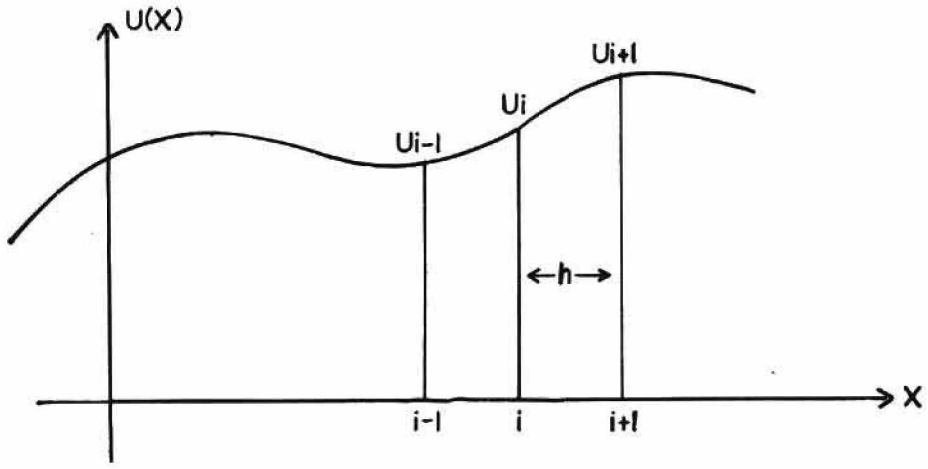


Figure 4

Using a Taylor's Series to Approximate
the Function U = U(x)

This results in the two equations:

$$U_{i-1} = U_i - h\frac{du}{dx} + h^2\frac{d^2u}{dx^2} - h^3\frac{d^3u}{dx^3} + 0\ [h^4] \qquad (12)$$

and

$$U_{i+1} = U_i + h\frac{du}{dx} + h^2\frac{d^2u}{dx^2} + h^3\frac{d^3u}{dx^3} + 0\ [h^4], \qquad (13)$$

where all derivatives are evaluated at the point X = i. Subtracting equation (13) from equation (12) gives the three point, central finite difference formula for the first order derivative of U at X = i as

$$\frac{du}{dx} = \frac{U_{i+1} - U_{i-1}}{2h} + 0\ [h^2]. \qquad (14)$$

Similar derivations give the forward difference formula

$$\frac{du}{dx} = \frac{U_{i+1} - U_i}{h} + 0\,[h], \tag{15}$$

and the backward difference formula

$$\frac{du}{dx} = \frac{U_i - U_{i-1}}{h} + 0\,[h]. \tag{16}$$

The usefulness of these difference formulae can be demonstrated by considering a simple differential equation:

$$\frac{dz}{dx} = F(x,z) \quad , z(x_o) = z_o. \tag{17}$$

The derivative $\frac{dz}{dx}$ at the point x_o can be approximated using the forward difference formula of equation (15) as:

$$\left.\frac{dz}{dx}\right|_{x=o} \approx \frac{z_1 - z_o}{h} \tag{18}$$

Applying this approximation to the given differential equation (17)

$$z(x_1) = z(x_o) + hF(x_o, y_o) \tag{19}$$

gives an approximation to the solution of equation (17) at the point X_1.

Clearly, successive applications of this difference scheme will result in solutions to the equation for all values of x for which the function is defined.

Extending this derivation to functions of two variables also results in the central difference forms useful in the solution of partial difference equations:

$$\frac{\partial u}{\partial x} = \frac{U_{i+1,j} - U_{i-1,j}}{2h} + O\,[\,h^2\,], \tag{20}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{h^2} + O[\,h^2\,], \tag{21}$$

Similar forms exist for $\frac{\partial u}{\partial y}$, $\frac{\partial^2 u}{\partial y^2}$, and $\frac{\partial^2 u}{\partial x \partial y}$.

An equation which frequently appears in studies of fluid dynamics is Poisson's equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = g\,(x,y). \tag{22}$$

Consider solving this equation over some region R with some appropriate boundary conditions along the boundary B. By superimposing a square mesh on the region R, as in Figure 5 , the finite difference forms of the derivatives at the point (X_1,Y_1) can be written

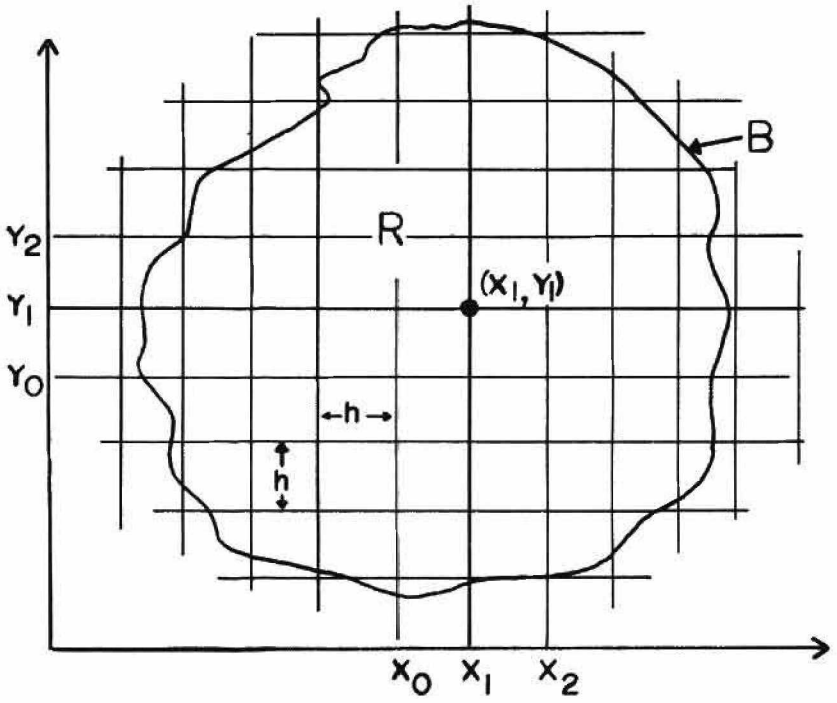


Figure 5

The Use of a Mesh in Deriving Finite Difference Equations

$$\frac{\partial^2 U}{\partial x^2} = \frac{U(X_0, Y_1) - 2U(X_1 Y_1) + U(X_2 Y_1)}{h^2} \tag{23}$$

and

$$\frac{\partial^2 U}{\partial y^2} = \frac{U(X_1, Y_0) - 2U(X_1, Y_1) + U(X_1 Y_2)}{h^2} \tag{24}$$

Using these approximations, equation (22) can be written as

$$\frac{U(X_o,Y_1) + U(X_2,Y_1) + U(X_1,Y_o) + U(X_1Y_2) - 4U(X_1,Y_1)}{h^2} \quad (25)$$

$$= g(X_1,Y_1).$$

An analogous equation may be written for every point interior to B, being careful to use the appropriate boundary conditions for points near B. This leads to a system of linear equations in the unknowns $U(X_i,Y_i)$ which can be solved for by some iterative technique. All of the methods discussed in this paper are based on this use of finite difference equations.

C. STREAM FUNCTION AND VORTICITY

In general, solutions to problems in fluid dynamics are difficult to obtain. The most popular technique for simplifying the equations has been the introduction of stream function and vorticity as primary variables.

In considering the flow of an incompressible fluid, the mass conservation equation

$$\frac{\partial p}{\partial t} + \nabla\cdot (\rho\vec{U}) = 0 \quad (26)$$

reduces to the statement that a vector divergence is zero. If the flow field is two-dimensional, this vector divergence is the sum of only two derivatives. From the mass conservation equation given previously in equation (1) it follows that udy-vdx is an exact differential,

say equal to $d\psi$. Then

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x} \tag{27}$$

In this way we have used the mass-conservation equation to replace the two dependent variables u,v by the single dependent variable ψ, which is a very valuable simplification in many cases of two-dimensional flow.

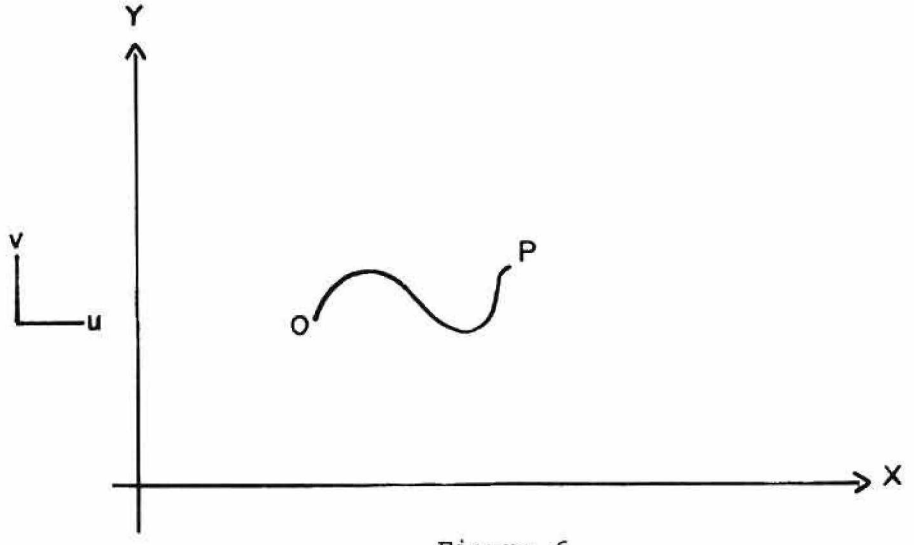


Figure 6

Derivation of Stream Function

The scalar function ψ is now defined by

$$\psi - \psi_o = \int (u dy - v dx), \tag{28}$$

where ψ_o is a constant and the line integral is taken along an arbitrary curve joining some reference point 0 to the point P with coordinates x,y, as shown in Figure (6). From this we can show that the flux of volume across the closed curve formed from any two paths joining 0 and P is necessarily zero when the enclosed region is wholly occupied

by incompressible flow. The flux then is independent of the path, and defines a function of the position of P. Since the flux of volume across any curve joining two points is equal to the difference between the values of ψ at these two points, it follows that ψ is constant along a streamline. ψ is therefore termed the stream function.

Another important concept in simplifying the equations is vorticity. The vorticity, or rotation, of a fluid is given by

$$\vec{\omega} = \nabla x \vec{u} \tag{29}$$

Again, restricting analysis to two-dimensional flow, there exists only one component of vorticity, namely

$$\omega z = \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} \ . \tag{30}$$

Almost all of the solution techniques that this author is acquainted with use stream function and vorticity to simplify the equations of fluid flow. A few of these techniques are outlined below.

D. SOME PREVIOUSLY USED NUMERICAL TECHNIQUES

One of the earliest attempts to solve numerically a problem of time dependent two-dimensional incompressible viscous flow was done by Fromm [14] at Los Alamos. Fromm's use of stream function and vorticity is representative of most numerical techniques.

Taking the curl of equation (10), and using the definition

of vorticity of equation (29), results in the vorticity transport equation

$$\frac{\partial \omega}{\partial t} = \nabla^2 \omega + \frac{\partial \omega}{\partial y}\frac{\partial \psi}{\partial x} - \frac{\partial \omega}{\partial x}\frac{\partial \psi}{\partial y} . \tag{31}$$

The definition of vorticity (29) can be written in terms of stream function

$$\omega = -\frac{1}{2}\nabla\psi . \tag{32}$$

This equation is a Poisson's equation, and can be solved by some iterative technique, using finite differences as outlined previously.

Fromm begins with some initial guess for his solution. Time is then advanced, and a finite difference form of equation (31) is used to compute new vorticities thoughout the mesh. Using the updated vorticites, equation (32) is solved for the ψ field. An improved Liebmann's method was used [15], taking advantage of any available advanced values of neighboring points. The iteration continues until the field settles down. Finally, new velocity components corresponding to the updated ψ values are calculated, and then used to obtain corrected vorticity values at walls and obstacles. This process is then repeated until some appropriate convergence criterion is satisfied.

Fromm's method suffered from a poor approximation of the boundary conditions, and the requirement for stability and accuracy of excessively small time steps. Steps to improve the application of the boundary conditions and the iterative procedures were taken by Pearson [16],

and Esch [17]. Stream function-vorticity methods still have several difficulties. First, the boundary conditions are difficult to apply, especially to a free surface. Second, extension to three-dimensions and cylindrical coordinates involves a great effort. These difficulties have been overcome by solving the Navier-Stokes equations in terms of the primary variables velocity and pressure. This work was done by Harlow et al. [18] at Los Alamos.

CHAPTER FOUR

THE DEVELOPMENT OF THE NUMERICAL METHOD

A. THE MARKER AND CELL TECHNIQUE

Developed at Los Alamos, the Marker and Cell [19] (MAC) technique for computing time-dependent, viscous, incompressible fluid flows in several space dimensions, was the first such method to use the primary variables velocity and pressure. The MAC technique utilizes the full Navier-Stokes equations, without the usual simplifying assumptions. The MAC methodology as described here forms the basis for later developments which were used in this research.

The Solution Technique

The MAC method uses two coordinate systems. The primary coordinate system covers the region of interest with a rectangular mesh, each cell of dimension δx by δy. If the cells are numbered by indices I and J, such that I counts the columns in the X direction and J the rows in the Y direction, then the field variables describing the flow field can be positioned as shown in Figure 7 .

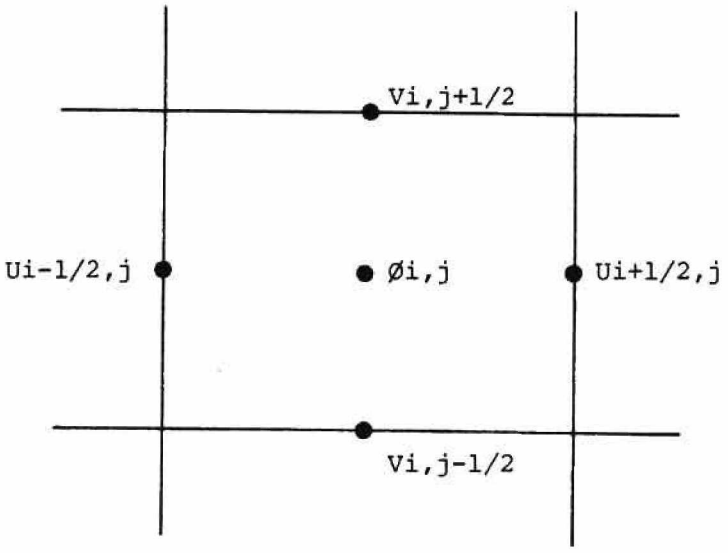


Figure 7

Location of Primary Variables in MAC Finite Difference Grid

The variables are placed as shown in order to maintain conservation. If the field variables are placed at the cell centers, the finite difference equation for pressure would require the involvement of the next layer of cells beyond that which immediately surrounds any central cell in order to attain rigorous finite-difference mass conservation. This means that the solution technique becomes much more complex, in addition to the inaccuracies introduced by using far-distant quantities.

Besides the coordinate system attached to the finite difference cells, there is a coordinate system of particles whose motions describe the trajectories of fluid elements. The marker particles serve two functions: first they show which cells are free surface cells. Second,

they show the motion of the fluid as it passes through the computing region.

The calculation is carried out by advancing the entire fluid configuration through a small, but finite, time increment. The results of each time cycle acts as the initial conditions for the next one, and the computation continues as long as the fluid motion is of interest. The steps in each cycle are outlined below:

1) The pressure for each cell is obtained by solving a finite-difference Poisson's equation, whose source term is a function of the velocities. This equation was derived subject to the requirement that the resulting momentum equations should produce a new velocity field that satisfies the incompressibility condition.
2) The full finite-difference Navier-Stokes equations are used to find the new velocities throughout the mesh.
3) The marker particles are moved to their new positions, using for their velocities simple interpolated values from the nearby cells.
4) Bookkeeping processes are accomplished related to the creation or destruction of surface cells, the input or output of particles, the advancement of a time counter, printing or plotting results, and numerous similar matters.

The Finite Difference Equations

Returning to the Navier-Stokes equations, the finite difference form of equation (9) can be written for MAC usage as

$$D_{ij} \equiv \frac{1}{\delta x} \left(u_{i+1/2,j} - u_{i-1/2,j}\right) + \frac{1}{\delta y} \left(v_{i,j+1/2} - v_{i,j-1/2}\right) \qquad (33)$$

The incompressibility condition then becomes

$$D_{ij} \equiv 0 \qquad (34)$$

which is required at every cell, at every time step. Rewriting equation (10) as

$$\frac{\partial \vec{u}}{\partial t} = -\nabla \cdot (\vec{u}\vec{u}) - \nabla \phi + \nu \nabla^2 \vec{u}, \qquad (35)$$

The finite difference equations can be written

$$\begin{aligned} \frac{1}{\delta t}\left(u^{n+1}_{i+1/2,j} - u_{i+1/2,j}\right) = & -\frac{1}{\delta x}\left[(u_{i,j})^2 - (u_{i+1,j})^2\right] \\ & - \frac{1}{\delta y}\left[(uv)_{i+1/2,j-1/2} - (uv)_{i+1/2,j+1/2}\right] \\ & + g_x - \frac{1}{\delta x}\left(\phi_{i,j} - \phi_{i+1,j}\right) \\ & + \nu\left[\frac{1}{\delta x^2}\left(u_{i+3/2,j} + u_{i-1/2,j} - 2u_{i+1/2,j}\right)\right. \\ & \left. + \frac{1}{\delta y^2}\left(u_{i+1/2,j+1} + u_{i+1/2,j-1} - 2u_{i+1/2,j}\right)\right] \end{aligned} \qquad (36)$$

and

$$\frac{1}{\delta t}\left(v^{n+1}_{i,j+1/2} - v_{i,j+1/2}\right) = \frac{1}{\delta x}\left[(uv)_{i-1/2,j+1/2} - (uv)_{i+1/2,j+1/2}\right]$$

$$+ \frac{1}{\delta y}\left[(v_{i,j})^2 - (v_{i,j+1})^2\right]$$

$$+ g_y + \frac{1}{\delta y}\left(\phi_{i,j} - \phi_{i,j+1}\right)$$

$$+ \nu\left[\frac{1}{\delta x^2}\left(v_{i+1,j+1/2} + v_{i-1,j+1/2} - 2v_{i,j+1/2}\right)\right.$$

$$\left. + \frac{1}{\delta y^2}\left(v_{i,j+3/2} + v_{i,j-1/2} - 2v_{i,j+1/2}\right)\right], \tag{37}$$

where g_x and g_y are the gravity components. Equations (36) and (37) are immediately available for computing new velocities, and in fact are the equations used in step (2) of the computing technique outlined previously.

Now it is only necessary to find an equation for the pressures. In order to do this, first define

$$Q_{ij} \equiv \frac{1}{\delta x^2}\left[(u_{i+1,j})^2 + (u_{i-1,j})^2 - 2(u_{i,j})^2\right]$$

$$+ \frac{1}{\delta y^2}\left[(v_{i,j+1})^2 + (v_{i,j-1})^2 - 2(v_{i,j})^2\right]$$

$$+ \frac{2}{\delta x \delta y}\left[(uv)_{i+1/2,j+1/2} + (uv)_{i-1/2,j-1/2} - (uv)_{i+1/2,j-1/2}\right.$$

$$\left. - (uv)_{i-1/2,j+1/2}\right]. \tag{38}$$

From equations (36) and (37) it follows that

$$\frac{1}{\delta t}\left(D_{i,j}^{n+1} - D_{i,j}\right) = -Q_{i,j} - \frac{1}{2}\left(\phi_{i+1,j} + \phi_{i-1,j} - 2\phi_{i,j}\right)$$

$$- \frac{1}{\delta y^2}\left(\phi_{i,j+1} + \phi_{i,j-1} - 2\phi_{i,j}\right)$$

$$+ \nu\left[\frac{1}{\delta x^2}\left(D_{i+1,j} + D_{i-1,j} - 2D_{i,j}\right) + \frac{1}{\delta y^2}\left(D_{i,j+1} + D_{i,j-1} - 2D_{i,j}\right)\right]. \quad (39)$$

By setting $D_i^{n+1}{}_j = 0$ in this equation, the fundamental equation for finding the pressures is

$$\frac{1}{\delta x^2}\left(\phi_{i+1,j} + \phi_{i-1,j} - 2\phi_{i,j}\right) + \frac{1}{\delta y^2}\left(\phi_{i,j+1} + \phi_{i,j-1} - 2\phi_{i,j}\right) = -R_{i,j} \quad (40)$$

where

$$R_{i,j} \equiv Q_{i,j} - \frac{D_{i,j}}{\delta t}$$

$$-\nu\left[\frac{1}{\delta x^2}\left(D_{i+1,j} + D_{i-1,j} - 2D_{i,j}\right) + \frac{1}{\delta y^2}\left(D_{i,j+1} + D_{i,j-1} - 2D_{i,j}\right)\right] \quad (41)$$

Equation (40) is a Poisson's equation, and must be satisfied by some iterative method.

Boundary Conditions

Rigid walls, inflow boundaries, and outflow boundaries are confined to follow cell boundaries. This means that all side or obstacle walls must be horizontal or vertical.

If the wall is to allow for Free-slip (i.e. there is no friction between the wall and the fluid), then whenever an equation calls for use of an exterior tangential velocity, the calculation simply uses the value of the tangential velocity at the image point back in the computing region. If an exterior normal velocity is required, the negative of the image value is used so that the normal component of the velocity in fact vanishes at the wall.

If the wall is no-slip (i.e. the tangential component of the fluid velocity = 0 along the wall), then the exterior tangential component must be the negative of the image point. In order that D_{ij} vanish for exterior cells, this requires the external normal velocity components to have the same value as they do at their image interior points.

Expressions relating the normal difference of the pressure to the normal component of the body force and the viscous diffusion of normal momentum are contained in the momentum equations. These expressions supply the needed boundary information for solution of the Ø-equation.

B. SIMPLIFIED MARKER AND CELL TECHNIQUE

It is the application of these boundary conditions in the Ø-equation which makes the MAC method unduly complicated. In addition, the solution of the Poisson's Equation is very difficult, and does not lend itself well to fast solution techniques. In order to overcome these difficulties, the Los Alamos Group modified the Marker and Cell method, making it much simpler to work with.

The procedure for each computational cycle as outlined by Amsden

and Harlow [20] is outlined below:

1. A tentative field of advanced-time velocities is calculated by using an arbitrary pressure field within the fluid, but with a pressure boundary condition at the free surface satisfying the normal stress condition. Correct velocity boundary conditions assure that this tentative velocity field contains the correct vorticity at every interior point in the fluid. The tentative velocities do not, however, have $\nabla\cdot\vec{u} = 0$.

2. The tentative velocities are modified to their final values so as to preserve the vorticity at every point. A potential function is employed, determined by the requirement that it convert the velocity field to one which satisfies the incompressibility condition everywhere.

To begin, the basic equations (7), (8), and (1) are rewritten to make use of cylindrical coordinates:

$$\frac{\partial u}{\partial t} + \frac{1}{r^{\alpha}} \frac{\partial r^{\alpha} u^2}{\mu \partial r} + \frac{\partial uv}{\partial z} = - \frac{\partial \phi}{\partial r} + g_r + \nu \frac{\partial}{\partial z} \left(\frac{\partial u}{\partial z} - \frac{\partial v}{\partial r} \right), \tag{42}$$

$$\frac{\partial v}{\partial t} + \frac{1}{r^{\alpha}} \frac{\partial r^{\alpha} uv}{\partial r} + \frac{\partial v^2}{\partial z} = - \frac{\partial \phi}{\partial z} + g_z - \frac{\nu}{r^{\alpha}} \frac{\partial}{\partial r} \left[r^{\alpha} \left(\frac{\partial u}{\partial z} - \frac{\partial v}{\partial r} \right) \right], \tag{43}$$

and

$$D \equiv \frac{1}{r^{\alpha}} \frac{\partial r^{\alpha} u}{\partial r} + \frac{\partial v}{\partial z} = 0 \ . \tag{44}$$

The velocity components, u and v, are respectively in the r and z directions, and the pressure, ϕ, is normalized to unit density. Plane cartesian coordinates have $\alpha=0$, and cylindered corrdinates have $\alpha= 1$.

Writing the finite difference forms of these quations, specifying the advance time velocities with a tilde:

$$\frac{\tilde{u}^{n+1}_{i+1/2,j} - u^{n}_{i+1/2,j}}{\delta t} =$$

$$\frac{r^{\alpha}_{i} u^{n}_{i+i/2,j} \; u^{n}_{i-1/2,j} - r^{\alpha}_{i+1} \; u^{n}_{i+3/2,j} \; u^{n}_{i+1/2,j}}{r^{\alpha}_{i+1/2} \delta r}$$

$$+ \frac{u^{n}_{i+1/2,j-1/2} \; v^{n}_{i+1/2,j-1/2} - u^{n}_{i+1/2,j+1/2} \; v^{n}_{i+1/2,j+1/2}}{\delta z}$$

$$+ \frac{\theta i,j - \theta i+1,j}{\delta r} + g_{r}$$

$$+ \nu \left[\frac{1}{\delta z^{2}} \left(u^{n}_{i+1/2,j+1} + u^{n}_{i+1/2,j-1} - 2u^{n}_{i+1/2,j} \right) \right.$$

$$\left. - \frac{1}{\delta r \delta z} \left(v^{n}_{i+1,j+1/2} - v^{n}_{i+1,j-1/2} - v^{n}_{i,j+1/2} + v^{n}_{i,j-1/2} \right) \right]$$

(45)

$$\frac{\tilde{v}^{n+1}_{i,j+1/2} - v^{n}_{i,j+1/2}}{\delta t} =$$

$$\frac{r^{\alpha}_{i-1/2}\, u^{n}_{i-1/2,j+1/2}\, v^{n}_{i-1/2,j+1/2} - r^{\alpha}_{i+1/2}\, u^{n}_{i+1/2,j+1/2}\, v^{n}_{i+1/2,j+1/2}}{r^{\alpha}_{i}\delta r}$$

$$+ \frac{v^{n}_{i,j+1/2}\, v^{n}_{i,j-1/2} - v^{n}_{i,j+3/2}\, v^{n}_{i,j+1/2}}{\delta z}$$

$$+ \frac{\theta i,j - \theta i,j+1}{\delta z} + g_{z}$$

$$- \frac{\nu}{r^{\alpha}_{i}\delta r}\left[r^{\alpha}_{i+1/2} \left(\frac{u^{n}_{i+1/2,j+1} - u^{n}_{i+1/2,j}}{\delta z} \right.\right.$$

$$\left. - \frac{v^{n}_{i+1,j+1/2} - v^{n}_{i,j+1/2}}{\delta r} \right)$$

$$\left. - r^{\alpha}_{i-1/2} \left(\frac{u^{n}_{i-1/2,j+1} - u^{n}_{i-1/2,j}}{\delta z} - \frac{v^{n}_{i,j+1/2} - v^{n}_{i-1,j+1/2}}{\delta r} \right) \right] , \qquad (46)$$

and

$$D^{n+1}_{i,j} \equiv \frac{r^{\alpha}_{i+1/2}\, u^{n+1}_{i+1/2,j} - r^{\alpha}_{i-1/2}\, u^{n+1}_{i-1/2,j}}{r^{\alpha}_{i}\delta r}$$

$$+ \frac{v^{n+1}_{i,j+1/2} - v^{n+1}_{i,j-1/2}}{\delta z} = 0 . \qquad (47)$$

The finite difference approximation can now be written for the vorticity as:

$$\omega^{n}_{i+1/2,j+1/2} \equiv \frac{u^{n}_{i+1/2,j+1} - u^{n}_{i+1/2,j}}{\delta z}$$

$$- \frac{v^{n}_{i+1,j+1/2} - v^{n}_{i,j+1/2}}{\delta r} . \qquad (48)$$

Equations (45) and (46) can be combined to give a vorticity transport equation. This equation can be shown to be independent of the pressure field, θ. This means that any field of pressure inserted into the Navier Stokes equations will assure that the resulting velocity field carries the correct vorticity. This velocity field, however, does not satisfy the condition that Di,j vanish for each cell. It is necessary then to convert the tilde velocities into a final velocity field so that Di,j = 0 for every cell. This must be done so that the vorticity already determined is preserved. This implies that the change in velocity be given by a gradient of a potential function, which will be called β. This can be written

$$U^{n+1}_{i+1/2,j} = \tilde{U}^{n+1}_{i+1/2,j} - \frac{1}{\delta r}\left(\beta_{i+1,j} - \beta_{i,j} \right) , \tag{49}$$

$$V^{n+1}_{i,j+1/2} = \tilde{V}^{n+1}_{i,j+1/2} - \frac{1}{\delta z}\left(\beta_{i,j+1} - \beta_{i,j} \right) , \tag{50}$$

Now, with equation (47) and equations (49) and (50), it follows that

$$D^{n+1}_{i,j} = \tilde{D}_{i,j} - \frac{1}{r^{\alpha}_{i}\delta r^2}\left[r^{\alpha}_{i+1/2}\left(\beta_{i+1,j} - \beta_{i,j} \right) - r^{\alpha}_{i-1/2}\left(\beta_{i,j} - \beta_{i-1,j} \right) \right] - \frac{1}{\delta_z{}^2}\left(\beta_{i,j+1} + \beta_{i,j-1} - 2\beta_{i,j} \right) . \tag{51}$$

Since it is required that $D^{n+1}_{i,j} \equiv 0$ for every cell, the value of $\beta_{i,j}$ for every cell can be expressed as

$$\beta_{i,j}^{h+1} = \frac{(1+\alpha)}{\left(\frac{2}{\delta_r{}^2} + \frac{2}{\delta_z{}^2}\right)} \left[- D_{i,j} + \frac{r_{i+1/2}\,\beta_{i+1,j}^{h} + r_{-1/2}\,\beta_{i-1,j}^{h+1}}{-\,r_i\,\delta r^2} + \frac{\beta_{i,j+1}^{h} + \beta_{i,j-1}^{h+1}}{\delta_z{}^2} \right] - \alpha\beta_{i,j}^{h}. \qquad (52)$$

Here, h indicates the iteration number, and α an over-relaxation parameter. The iteration sequence proceeds until appropriate convergence criteria are satisfied.

C. ARBITRARY BOUNDARY MARKER AND CELL TECHNIQUE

The restriction placed on most finite difference schemes, that the boundaries of theproblem must lie on the computational mesh, is most severe. In effect, this means that one must restrict his problem to having rectangular walls. In the particular application of the heart valve, it is certainly desirable to have boundaries of any arbitrary shape.

Mr. J.A. Viecelli developed a generalization of the Marker and Cell Technique at the Lawrence Radiation Laboratory [21]. Known as the Arbitrary Boundary Marker and Cell Technique (ABMAC) it treats the fluid boundary at an arbitrarily curved wall or obstacle as a free surface, to which a pressure is applied such taht the particles at the boundary move tangent to it. In order to calculate the pressure along the free surface, the usual MAC iteration formula is replaced by a simultaneous scheme proposed by Chorin [22,23].

Using centered differences in a cylindrical coordinate system, equation (7) can be written

$$u^{n+1}_{i+1/2,j} = u_{i+1/2,j} - \frac{\Delta t}{r^{\alpha}\Delta r}\left[(r^{\alpha}u^2)^{n}_{i+1j} - (r^{\alpha}u^2)^{n}_{ij}\right] - \frac{\Delta t}{\rho\Delta r}\left[P_{i+1j} - P_{ij}\right]$$

$$- \frac{\Delta t}{\Delta z}\left[(vu)^{n}_{i+1/2,j+1/2} - (vu)^{n}_{i+1/2,j-1/2}\right] - g_r\Delta t$$

$$+ \frac{\nu\Delta t}{\Delta z^2}\left[u^{n}_{i+1/2,j+1} - 2u^{n}_{i+1/2,j} + u^{n}_{i+1/2,j-1}\right]$$

$$- \nu\frac{\Delta t}{\Delta r\Delta z}\left[v^{n}_{i+1,j+1/2} - v^{n}_{i+1,j-1/2} - v^{n}_{i,j+1/2} + v^{n}_{i,j-1/2}\right] \tag{53}$$

By combining the old velocity, advection, body force, and viscous terms, and calling the resultant $\eta^{n}_{i+1/2j}$, equation (53) becomes

$$u^{n+1}_{i+1/2,j} = \eta^{n}_{i+1/2,j} - \frac{\Delta t}{\rho\Delta r}\left[P_{i+1j} - P_{ij}\right]. \tag{54}$$

Similarily the finite difference form of equation (8)

$$v^{n+1}_{i,j+1/2} = v^{n}_{i,j+1/2} - \frac{\Delta t}{\Delta z}\left[(v^2)^{n}_{i,j+1} - (v^2)^{n}_{ij}\right] - \frac{\Delta}{\rho\Delta z}\left[P_{ij+1} - P_{ij}\right]$$

$$- \frac{\Delta t}{r^{\alpha}\Delta r}\left[(r^{\alpha}uv)^{n}_{i+1/2,j+1/2} - (r^{\alpha}uv)^{n}_{i-1/2,j+1/2}\right] - g_z\Delta t$$

$$- \frac{\nu\Delta t}{r^{\alpha}\Delta r\Delta z}\left[\left(u^{n}_{i+1/2,j+1} - u^{n}_{i+1/2,j}\right)r^{\alpha}_{i+1/2} - \left(u^{n}_{i-1/2,j+1} - u^{n}_{i-1/2,j}\right)r^{\alpha}_{i-1/2}\right.$$

$$\left. - \frac{\nu\Delta t}{r^{\alpha}\Delta r^2}\left[\left(v^{n}_{ij+1/2} - v^{n}_{i-1,j+1/2}\right)r^{\alpha}_{i-1/2} - \left(v^{n}_{i+1,j+1/2} - v^{n}_{i,j+1/2}\right)r^{\alpha}_{i+1/2}\right]\right. \tag{55}$$

can be written as

$$v^{n+1}_{i,j+1/2} = \xi^{n}_{i,j+1/2} - \frac{\Delta t}{\rho\Delta z}\left[P_{ij+1} - P_{ij}\right], \tag{56}$$

where $\xi^n_{i,j+1/2}$ represents the old velocity, advection, body force, and viscous terms in the v-momentum equation.

According to Chorin's derivation, the pressure and the advanced time velocity fields may be solved for simultaneously using the relationship

$$P^{i+1}_{ij} = P^{i}_{ij} - \Delta\tau \left(\vec{\nabla}\cdot\vec{V}^{n+1}\right)^{i}_{ij} \quad . \tag{57}$$

The iteration scheme is as follows:

1. Equation (57) is used to compute a new pressure field.
2. The new pressure field is substituted into equations (54) and (56) to obtain the new velocities.
3. These new velocities are used to compute new values of the divergence.
4. Repeat the above sequence until some convergence criteria is reached. Note that $\eta^n_{i+1/2,j}$ and $\xi^n_{i,j+1/2}$ need not be recomputed at each interaction.

In order to maximize the time step, the iterates

$$\left(U^{n+1}_{i+1/2,j}\right)^{i+1}, \left(U^{n+1}_{i-1/2,j}\right)^{i+1}, \left(V^{n+1}_{i,j+1/2}\right)^{i+1} \text{ and } \left(V^{n+1}_{i,j-1/2}\right)^{i+1}$$

must be recomputed as soon as $P^{i+1}_{i,j}$ has been obtained for a cell, and before advancing to the next cell.

The pressures in boundary cells must be computed in a slightly different manner. Instead of adjusting the pressure proportional to the divergence or net flux out of a cell, it is adjusted proportional to the flux across the boundary relative to coordinates fixed in the boundary. This means that if liquid is flowing across the boundary the

pressure will be increased until the outflow stops. If liquid is tending to separate from the boundary the pressure will decrease until the liquid flows tangent to the boundary. This is accomplished through the use of the following equation in boundary cells.

$$P_{i,j}^{i+1} = P_{i,j}^{i} - \frac{\Delta\tau}{\delta}\left\{\left[\left(V_p^{n+1}\right)^i - \vec{V}_b\,(r,t)\right]\cdot\vec{n}\right\}_{i,j} \qquad (58)$$

Here $\vec{n}$ is the normal defining the boundary segment associated with the cell (i,j), $V_b(r,t)$ is the velocity of the midpoint of the segment, and $(\vec{V}_p^{n+1})^i$ is the liquid velocity at the midpoint of the segment. $\Delta\tau$ and δ are the relaxation parameter and the mesh width respectively. The inclusion of the velocity of the boundary segment means that the boundary not only can assume an arbitrary shape, but can move relative to the computing mesh as well.

The capability of a boundary to assume an arbitrary shape and to move within the computing mesh is of great use in the study of blood flow problems. The flexing leaflets of the natural heart valve plus the motion of the sinus of Valsalva in the aortic root area are examples of needed flow calculations over objects whose boundaries undergo large deformations. In turn, components of heart valve prostheses move about in the computing mesh, requiring the same capability in the computational technique.

Consider the computing region shown in Figure 8. Its shape corresponds to the in-vitro experimental chamber of Weiting [9]. The flow is from the left verticle, through the open valve, into the aorta. Since the geometry of the valve is symmetrical about a line drawn through its center, we can solve the problem by considering only one

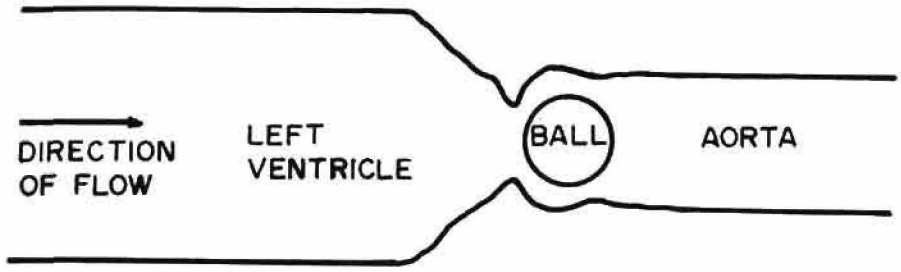


Figure 8

Mathematical Model of Aortic Valve
(Starr-Edwards 12-A)

half of the region. In Figure 9 the computational mesh has been placed over the half of the computing region we wish to consider, and it has been placed in an r-z coordinate frame.

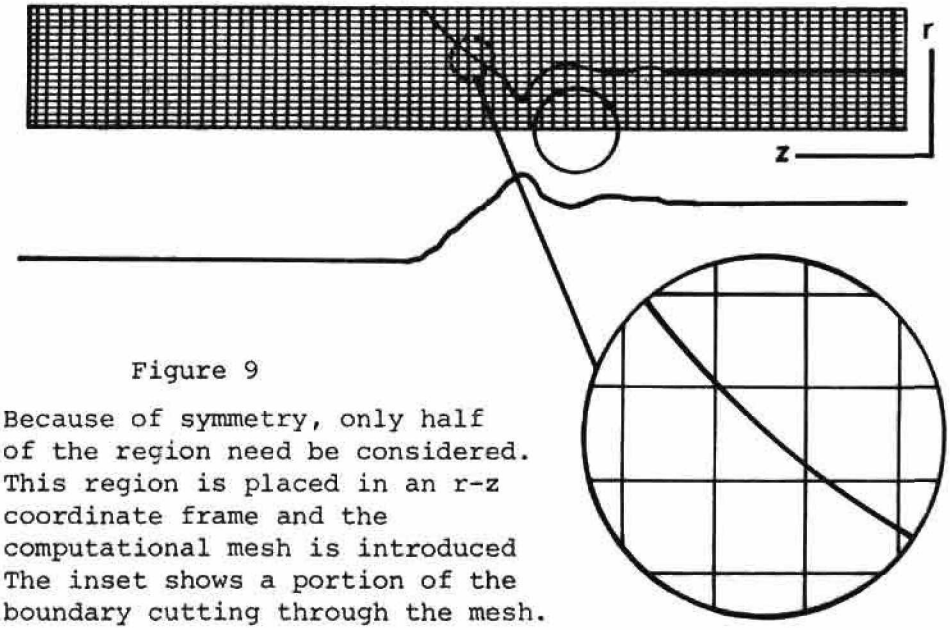


Figure 9

Because of symmetry, only half of the region need be considered. This region is placed in an r-z coordinate frame and the computational mesh is introduced The inset shows a portion of the boundary cutting through the mesh.

Looking at a portion of the boundary as it cuts through the computational mesh, we can visualize the ABMAC technique for describing an arbitrary, moving wall. In Figure 10a the boundary has been approximated in each cell by a straight line connecting the intersections of the boundary with the cell. The position of each segment is then

specified by a unit vector normal to the segment, with its base located at the mid-point of the segment. The convention is that the normal points towards the fluid, and to the left as one advances from the i^{th} to the $i+1^{st}$ boundary point. Also associated with each segment is a velocity vector, defining the motion of the wall. In this study, such velocity vectors would result from programming the pulsing motion of the walls, and the motion of the ball as a result of pressure changes in the system.

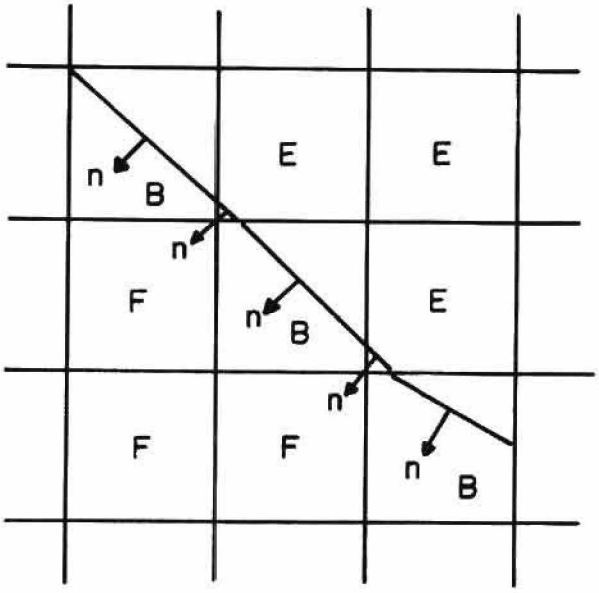


Figure 10a

Boundary is approximated by straight line segments connecting the intersections of the cell walls with the boundary. Unit vectors normal to the segment and pointing towards the fluid are positioned at the midpoint of each segment.

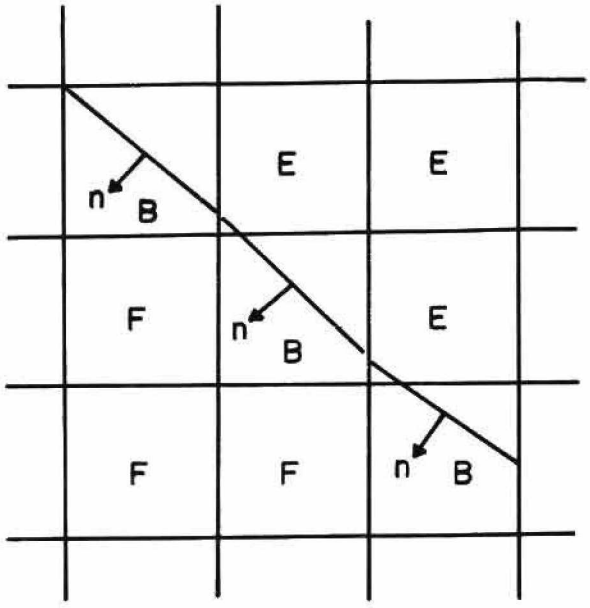


Figure 10b

When the liquid fraction of a cell is too small, the boundary flag is turned on in a neighboring cell. If that cell also contains a boundary segment, the two segments are replaced by one by removing the boundary intersection between the two cells.

Having determined in which cells the boundary lies, the cells are appropriately marked. Cells are marked as FULL, EMPTY, or BOUNDARY. The boundary is approximated by flagging a boundary cell as such, only if

the liquid fraction of the total cell is greater than a specified fraction, as shown in Figure 10a. If the liquid fraction is too small, the program determines the neighboring cell that the boundary segment normal points closest to, and turn on the flag for that cell. If that cell also contains a boundary segment, the two segments are replaced by a single segment, constructed by removing the boundary intersection between the two cells. This is illustrated in Figure 10b.

Finally, marker particles are introduced to visualize the fluid within the region. Figure 11 is an actual computer generated picture of the completely described computing region.

Figure 11

Completely Described Computing Region
Including Marker Particles

CHAPTER FIVE

INTERACTIVE COMPUTER GRAPHICS

The cathode ray tube display was used as a computer output device as early as 1956. The early Whirlwind computer at MIT used CRT displays to plot curves and graphs. However, the first significant use of a CRT display as an interactive device was demonstrated by Sutherland with his Sketchpad program [24]. Sketchpad exhibited the two basic functions of an interactive display device. First, it was used as an input/output device that could accept or display data in pictorial form. Second, it could be used to control the sequence of the program. Since this demonstration of interactive computer graphics, it has been found to be a valuable tool in many programming areas. Text editing [25], conversational mathematics programs [26], circuit design [27], mechanical design [28], and structural analysis [29], are some of the areas in which interactive computer graphics has been found to be useful.

Because fluid dynamics problems are extraordinarily complex, they tend to absorb the computational power of available computing systems. A single solution may consume several hours of CPU time on the most powerful of today's machines. As advances are being made in numerical methods, and in the understanding of the equations of fluid dynamics, concurrent investigations must be made in developing newer and faster computing techniques. The computer user needs to think of today's computer as more than just a very big, very fast calculator. There are many ways in which these machines can be used to create an effective

problem solving system. One such way is the addition of interactive computer graphics.

The role that interactive graphics can be expected to play in the study of fluid dynamics should be carefully evaluated before a large scale commitment to computer graphics is made. One area for consideration is the application of computer graphics to the display and interpretation of results. Digital plotters have been around for many, many years, helping computer users visualize their results. A picture may be worth a thousand words, but in the case of computer generated data, a graph or a plot may be worth over a thousand numbers. When working with a real world physical problem, the computer user wants to see a real world representation of his solution. He wants to see a picture, not rows and rows of numbers.

A system where this kind of graphics is used interactively has been proposed by Fromm and Schreiber [30]. Their approach has been to have the program which computed the solution to the fluid problem write a data set out onto a direct access storage device. Another program, running interactively can then read the data, and under user control provide a variety of interpretations of the data. A diagram of this scheme is shown in Figure 12.

Especially applicable to Marker and Cell type techniques is the making of motion pictures of the fluid simulations. A number of films have been made [31,32] showing the power of such techniques. As the marker particles are moved in each computational step, they are plotted and photographed. The resulting sequence provides a most graphic display of the fluid motion.

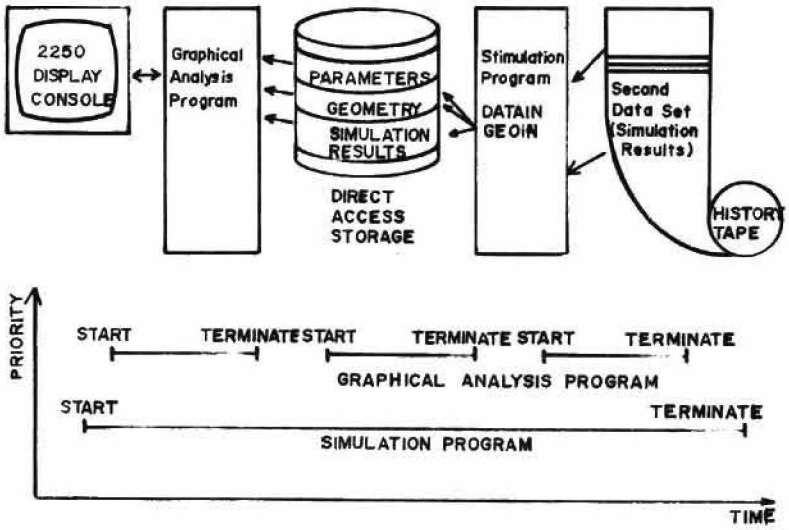


Figure 12

Graphic Analysis Program
(Fromm and Schreiber)

Another area in which it seems that graphics can easily be applied is the description and input of the problem to the computer. Still bound to ideas associated with batch processing, most problems are carefully drawn out by hand, measured and detailed and the resulting data punched into cards. It would seem much better to present the computer the data in a more natural way, where the user could sit down at an interactive graphics terminal, and draw the region in which a solution would be of interest, specifying the various parameters as the program requested them. In methods such as ABMAC where the specification of boundary segments and the correct flagging of computational cells is critical, the ability to quickly see the results of the input andspecify the appropriate changes would seem to be very desireable.

A third area worthy of attention is that of graphically monitoring and interacting with the executing fluid computation. Although this

idea has been considered by previous investigators [33,34], it was not developed to its full potential. The basic idea is very simple. In any complex numerical problem there are several factors which can affect the solution to that problem. In a finite difference problem, these typically are the mesh size, the time step, and the relaxation factor. Other factors which may be of importance are the convergence criteria, the differencing technique, and the proper use of boundary conditions.

In working with problems in fluid dynamics, it becomes very apparent that no two problems are alike. Given the same initial conditions, the same boundary conditions, but changing the geometry of the problem even slightly, may mean that a whole new set of convergence criteria may be necessary to solve the problem. Perhaps one problem converges very quickly but one diverges unless a different time step is used. Each problem must be considered as a separate entity. Certainly dumps of pertinent numbers can lead the user to the correct choice of conditions to allow him to reach the correct solution. The question is how can this process of getting into the solution space of the problem be made more efficient. The obvious choice seems to be interactive computer graphics.

The interactive computer graphics facilities at the University of Utah consist of a PDP-10 time shared computer running under Tenex, a PDP-9 computer, four Univac 1559 CRT displays, and the associated hardware interfaces. Figure 13 shows the basic configuration of the system.

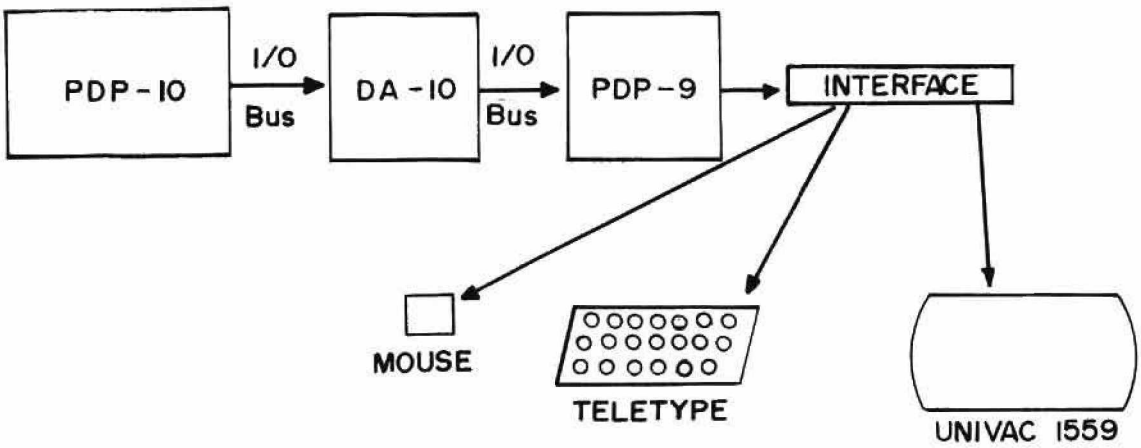


Figure 13

Configuration of Interactive Graphic System
at the University of Utah

The PDP-10 is a 36 bit machine. It has sixteen high speed integrated circuit registers which can be used as accumulators, normal memory, and/or index registers. The memory bus structure of the PDP-10 gives the central processor and high speed data channels simultaneous access to separate memory modules. The bus system allows each data channel to transmit full 36 bit words at speeds of up to one million words per second.

In conversational time sharing, up to 63 users at local and remote locations can simultaneously develop programs on remote consoles, and receive answers to mathematical or engineering problems in seconds. The time sharing monitor provides instantaneous response for the users, so that they can perform on-line composition, editing, and debugging of programs in FORTRAN IV, MACRO-10, BASIC, and AID. The monitor can handle any mixture of these languages and programs concurrently.

The PDP-10 at the University of Utah is configured with 96,000 words of core storage, 8 dec-tape drives, and 6 disk pack drives. Each disk pack is capable of holding over 5 million words of information.

Attached to the PDP-10 through the DA-10, a communications device, is the PDP-9. The PDP-9 is a single address, fixed word length (18 bits), binary computer. The configuration here has 8,192 words of memory. The function of the PDP-9 is to act as a satellite computer to the PDP-10. It processes interrupts from the displays gathers data to be sent to the PDP-10, and acts as a multiplexor for display information coming from the PDP-10.

The Univac 1559 is a high speed, buffered, line drawing display which was designed as a cooperative exercise between the University of Utah's Computer Science Department, and the Univac Division of Sperry Rand Corporation. The Screen of the 1559 has a useful viewing area of 10" by 10". Positions on the screen are specified in cartesian coordinates, in which the origin is placed at the bottom left hand corner of the screen. The top right hand corner is the point (1024, 1024).

Display files are held in the displays own memory, a 4096 by 16 bit core memory with a cycle time of 1.4 microseconds. The 1559 has its own program counter, called the list counter, which is used to access sequential display instructions.

All lines are drawn on the screen in a relative mode, i.e., they are defined by their length in the two axis directions, and are drawn from the position currently defined in the X and Y registers. The time taken to display a normal vector is 2 microseconds. Beam repositioning takes 32 microseconds or less, depending upon the distance involved.

Associated with each display is a mouse, and a teletype. The mouse is a graphical input device, consisting of a small plastic box in whose base two potentiometers are mounted. The mouse rests on two metal wheels, whose axes are horizontal and at right angles to each other. Each wheel is connected to one potentiometer. As the mouse is rolled around on a flat surface, its movement in two orthogonal directions is reocrded by the rotation of the potentiometers. This can be determined by applying a voltage across each potentiometer and sampling the outputs through analog-to-digital converters. Push buttons on the mouse give the user the ability to issue commands from the mouse under program control.

CHAPTER SIX

GRAPHICS INTERACTION FOR THE ABMAC PROCEDURE

Regardless of how sophisticated the application may be, if the graphics and interaction code are poorly handled it is difficult to justify their use. Whatever advantage may be gained through the addition of interactive graphics can be completely offset because of poor program design. In the kind of application being considered here, there are two glaring problems which must be considered. First, any problem involving finite differences is typically very large. In a time sharing environment such as the one being used here, any increase in core size becomes critical to the effective running of the machine. Second, fluid computations absorb a great deal of computing time. A solution may take more than 30 hours of computing time to complete. For this reason, the time required for interaction and graphics should not interfere a great deal with the actual computation time.

Work done by Carter [33] and Bennion [34] here at the University of Utah is typical of the approach which has been used in implementing interactive graphics in a large scale numerical application. Figure 14 illustrates the way in which this approach imbeds the interactive and graphics code within the computational program. This implies several drawbacks in light of the problems under consideration.

First, at each point in the computational program where one of these imbedded pieces of interactive graphics occurs, there must exist a sequence of code represented by the flow chart in Figure 15. This adds a certain amount of program space to the already large finite

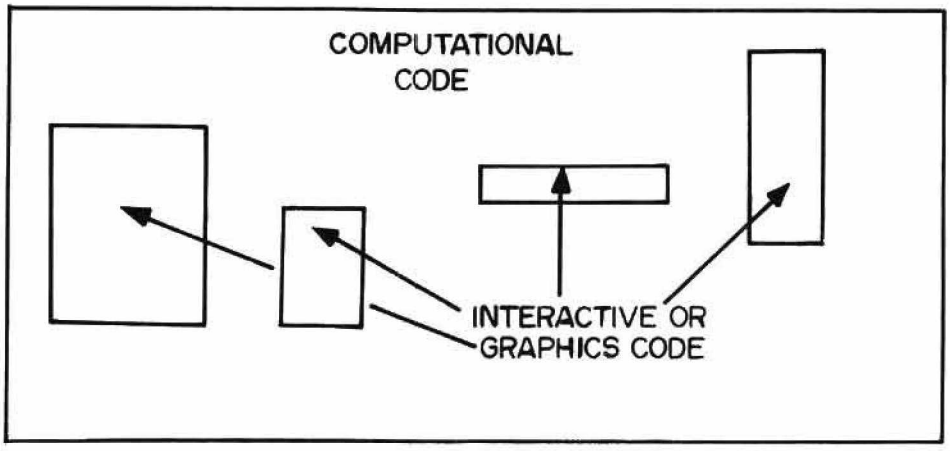


Figure 14
Schematic of Imbedded Interactive Graphics Code

difference code, and requires the program process this overhead each time that it is encountered. Suppose that some code of this nature was placed in an iterative loop, and that that loop was iterated through a hundred times per time cycle. That would represent a large amount of overhead in a program where computing time was already critical.

Second, consider what takes place when a display is required. Since dynamic events are being modelled, each time an additional piece of information is required to be displayed, a piece of code is needed to describe that display. This means that there will be an additional amount of program which will be proportional to the complexity of the display both in terms of space and the time needed to execute it. As the number and complexity of displays increases, the size of the

program grows, and the time required to display is taken from the total computation time.

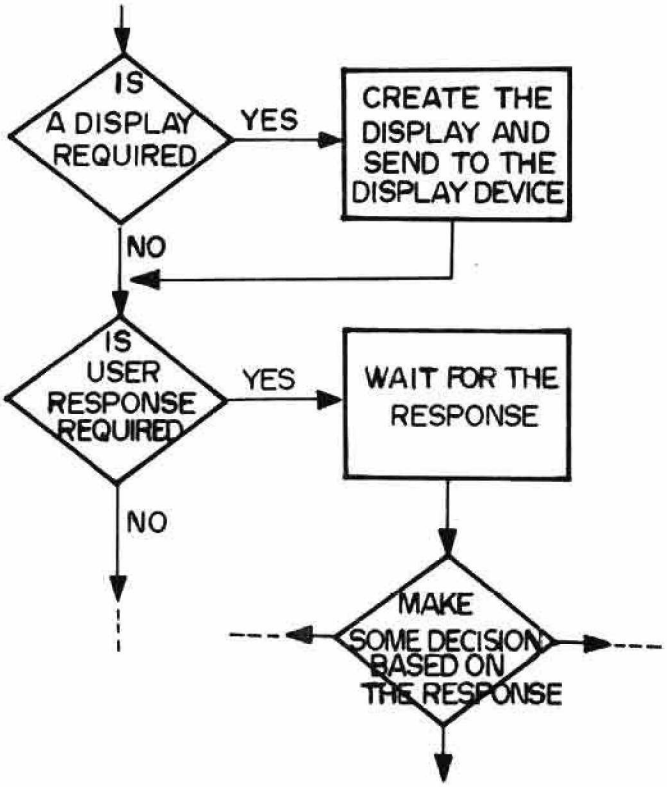


Figure 15

Flow Chart for Imbedded Interactive Graphics

Third, consider what occurs when some user interaction is required. Whenever the program expects a response from the user, it must stop and wait for that response. When the response is received, the program must make some decision or perform some branching based on the response. Again, what this means is that the more often interaction takes place, the more code the program needs, and the more time the program loses to computation.

The proposed solution to these problems is simple; remove the interactive graphics from the program. At first this may appear to be a contradictory statement. How can an interactive graphics system be written for solving a problem, when the computational program has no interactive graphics. By splitting up the program as shown in Figure 16 , the solution becomes easy.

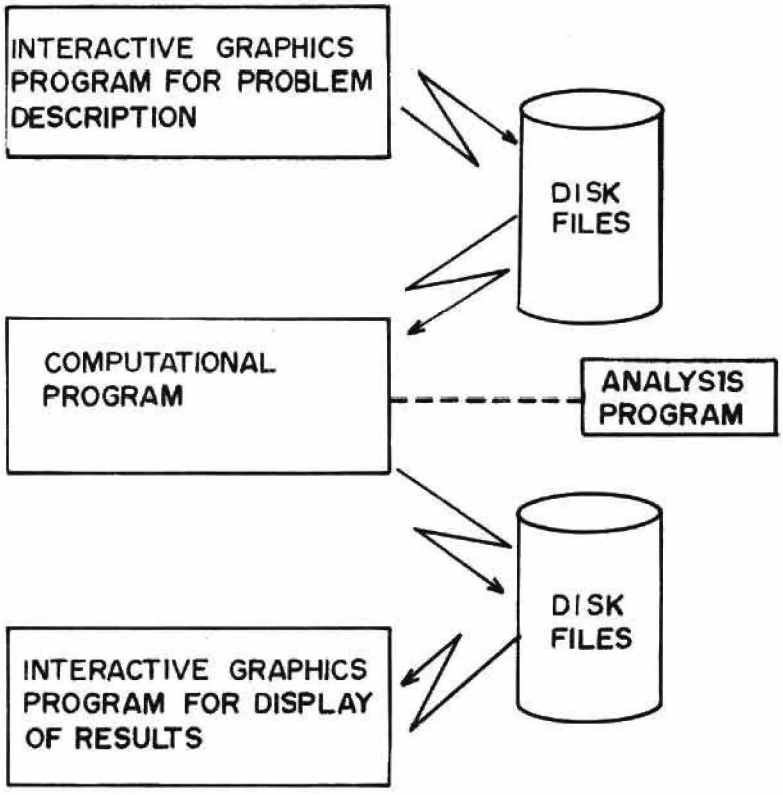


Figure 16

Formation of Supplemental Programs

This system allows the investigator a great deal of freedom in describing his problem to the computer. His input is not bound to certain fields on punched cards, but can be more naturally input as he manipulates figures on the face of the CRT. In this type of environment he can alter the problem description and have immediate visual feedback

of the changes he makes, and computation need not proceed until he is satisfied with the problem he has described. A term that is often heard in the computer community is "structured programming" [35]. In large software systems such as assemblers, compilers and operating systems, people try to structure their programs in a nice way. However, in the scientific world, it seems that this idea is often neglected. In the system under consideration, it seemed that a very highly structured program was a necessity. First, it is desirable to keep the computational routines and the display routines completely separate, and yet they must be able to communicate with each other very freely. Also, the graphics routines themselves must be structured in such a way as to allow the user the greatest possible degree of flexibility, and yet be simple to understand and use. These considerations pointed to the necessity of a modular program for implementing a graphical ABMAC technique.

In quantizing the program, it seemed appropriate to study the nature of the problem which it was intended to solve. Basically, the solution consists of three components; describing the problem, solving the problem, and displaying the solution. This is illustrated in Figure 17.

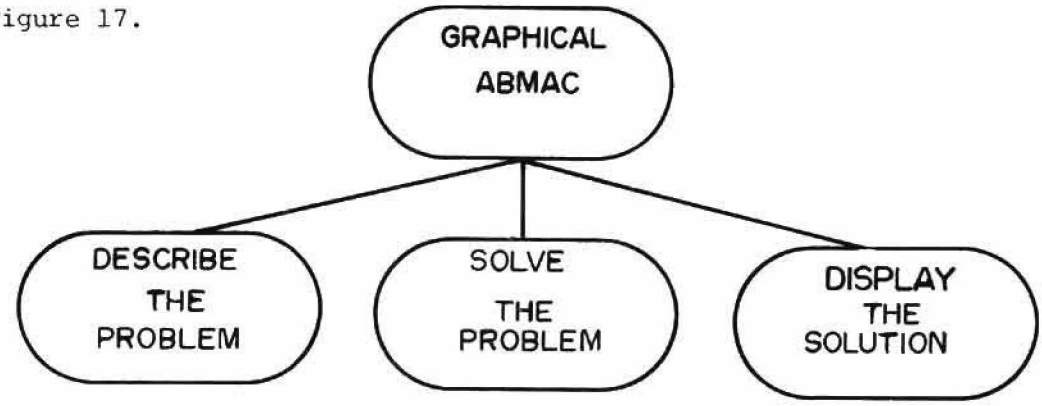


Figure 17

Basic Components of a Graphical ABMAC Program

A. DESCRIBING THE PROBLEM

The problem description can be broken down into four different components; setting the initial values of computational parameters, describing the computational grid, describing the geometry of the problem, and describing initial and boundary conditions of the fluid. These steps are shown in Figure 18.

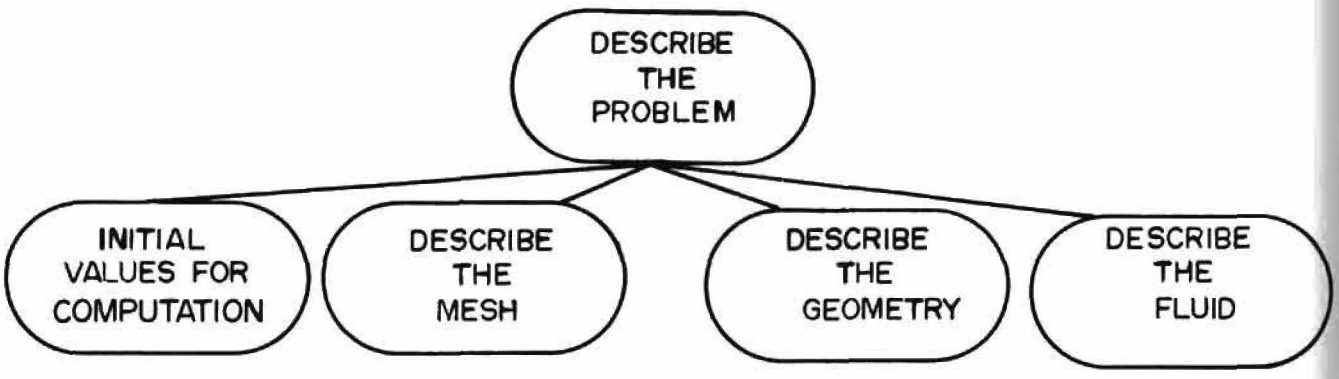


Figure 18
Components of Problem Description

Each of these components can be broken down even further into functional groups. It is at this level that the program was structured. For example, consider the mesh description. This requires four functional groups: a main or controlling program, a program to handle communications with the user through the teletype, a display program to give visual verification of entered values, and a routine to computer various scale factors and constants for later use. The actual grid is displayed in another set of programs.

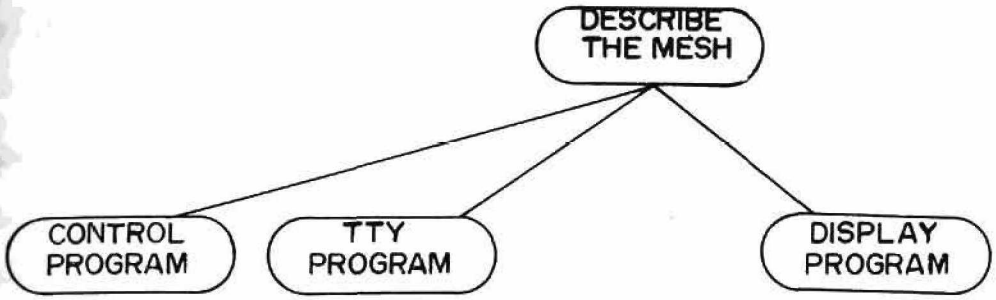


Figure 19

Components of Mesh Description

The entire system then takes on a tree-like structure. Putting the pieces together as described above gives the structure of Figure 20.

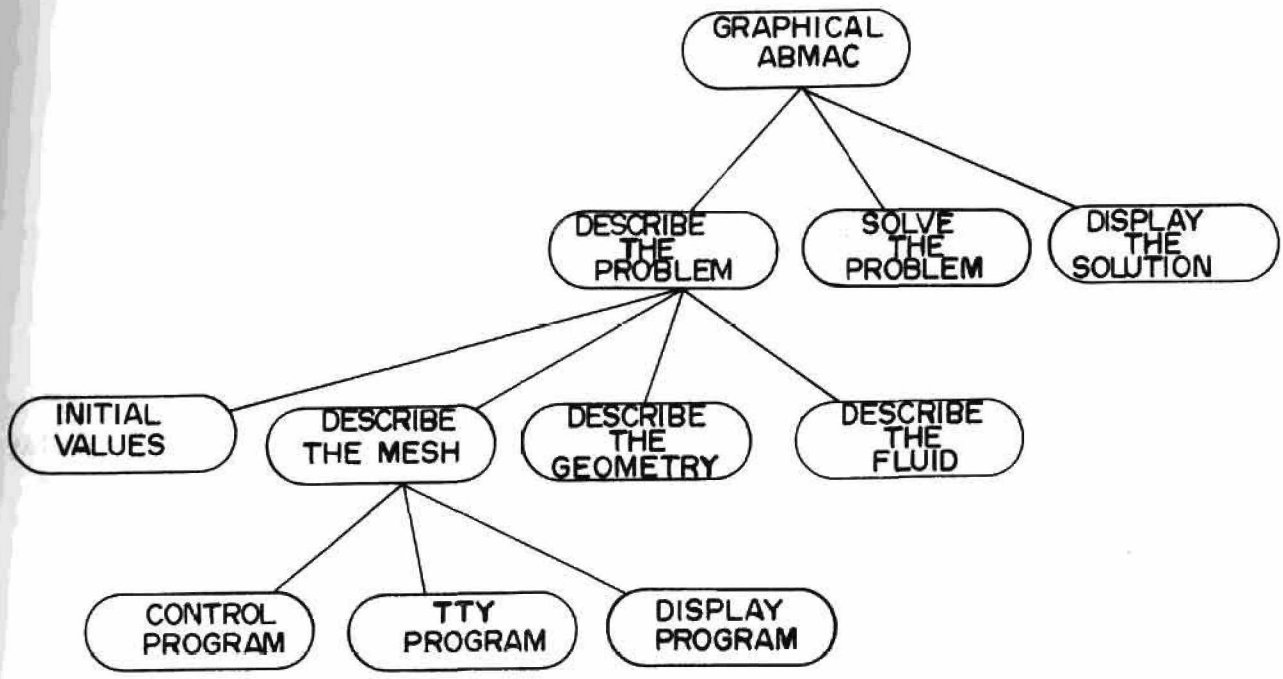


Figure 20

Tree Structure of Graphic ABMAC Components

This kind of structure makes the code simple, easy to understand, and interactive to a high degree. The following section will point out some of these ideas, as the basic components of the program are discussed in detail.

The Main Program

The main program in the input routines simply allows the user to choose the type of input he wants to use. After initializing the display system, the main program produced the display shown in Figure 21. By pointing to one of the boxes shown, the appropriate input program is called. "TTY" and "DSK" allow the user to input all of his data from the teletype or from a previously constructed disk file. If he points to "TERMINAL", the graphical input program is called and executed.

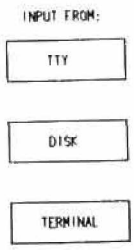


Figure 21

Initial Graphic ABMAC Display for Choosing Input Device

The Input Module

This module is the actual driving module for the input programs. It gives the user complete control of the input for his particular problem. In order to provide the user with some guide for constructing his problem description, the input module produces the display shown in

Figure 22. This display lists the various functions performed in a typical problem description. The user steps through this list by pointing to the function which he wishes to perform next. As he points to an item he pushes one of the switches on the mouse. This indicates to the program which function is desired. That program module is then called to be executed. When it is finished control returns to the input module. The basic structure of the input module is this:

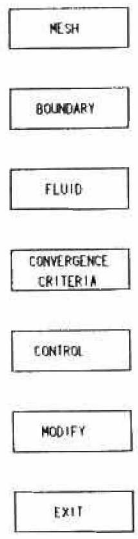


Figure 22

Computer Display for Graphic ABMAC Problem Description

```
1       CALL INPUT1
        CALL WMOUSE
        INY=(INY+128)/128
        GO TO (2,3,4,5,6,7,8)INY
        CALL CHECK
2       CALL OUTP
        CALL EXIT
3       CALL MOD
        GO TO 1
4       CALL CNTRL
        GO TO 1
5       CALL CONVG
        GO TO 1
6       CALL FLUID
        GO TO 1
7       CALL FIGURE
        GO TO 1
8       CALL MESH
        GO TO 1
        END
```

INPUT1 is the program which creates the display of Figure 20. WMOUSE is a program which halts the computer until one of the mouse switches is pushed. It then returns the coordinates of the mouse cursor in the variables INX and INY. This is how the mouse is used in pointing to objects on the display screen. By performing a transformation on INY and using a computed GO TO statement, the input module calls the appropriate subroutine. As each subroutine returns, a GO TO 1 is encountered which creates the list of functions again and waits for the next user response.

The light buttons produced by INPUT1 are created in a program called BOXES. The number of boxes desired is passed as an argument to the subroutine. It computes the correct starting position for each box, and then calls another subroutine named BOX which draws the actual box in that location.

The Mesh Description Module

This module allows the user to specify the computational mesh upon which he wishes to solve his problem. The way in which this is done illustrates another unique feature based on the structuring of the program. It is desireable to ask the user to input all of the required parameters, so that none are forgotten. Once this is done the program should allow him to selectively change any of the input values. This is accomplished using a driving routine for this module which looks like this:

```
      DO 1 I=1,5
1     CALL MESH1(I)
      CALL WMOUSE
      INY=(INY+128)/128
      IF(INY.GE.6)RETURN
      CALL MESH1(INY)
      END
```

and MESH1 looks like this:

```
      SUBROUTINE MESH1(J)
      GO TO (4,6,8,13,15),J
 4    TYPE 5
 5    FORMAT(1H ,'DR=',$)
      ACCEPT 3, DR
 3    FORMAT(F)
      GO TO 18
 6    TYPE 7
      .
      .
      .
18    CALL MESH2
      CALL SEND
      RETURN
      END
```

As the module executes, the following sequence of events occurs. MESH1 is called 5 times, each time with a different argument. This argument determines which of the parameters is to be requested from the teletype when the value is typed in. Control is then passed to MESH2. MESH2 puts up the display shown in Figure 23. Each time the display is produced, it shows the most recent values of the input mesh parameters.

Having gone through all of the mesh parameters in the DO LOOP, the driving routine calls WMOUSE in the same manner as in the control module. This way the user can point to a value he wants changed, and push the mouse switch. This calls MESH1 with the appropriate argument to change that value. If he points to the top box, control returns to the input module.

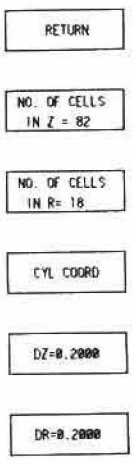


Figure 23

Computer Display for Graphic ABMAC Mesh Description

Figure Description Module

The purpose of this module is to allow the user to input the geometry of his problem in a natural way. Initially the module creates the display shown in Figure 24.

Pointing to INFLOW directs the program to accept the inflow boundary conditions of the problem. Pointing to BOUNDARIES directs the program to accept the geometry of the problem. A program called FIG3 controls the input on the geometry. Its first task is to create the display shown in Figure 25. This is a display of the computational mesh previously described. The coordinate axes, the number of cells in the mesh, Dr, and DZ are also displayed. The program is capable of accepting the input describing the figure either from the teletype or the mouse. As each point is entered, an arrow head is created and displayed, showing the location of the point. A completely described digure is shown in Figure 26. At the conclusion of executing this module, the completed figure is created and displayed without the mesh, as shown in Figure 27. This figure is seen to be equivalent to half of the region of interest shown in Figure 8. Because of symmetry only half of the region need be considered. If the display produces a satisfactory drawing, control returns to the input module.

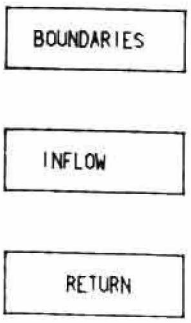


Figure 24
Computer Display for Graphic ABMAC Figure Description

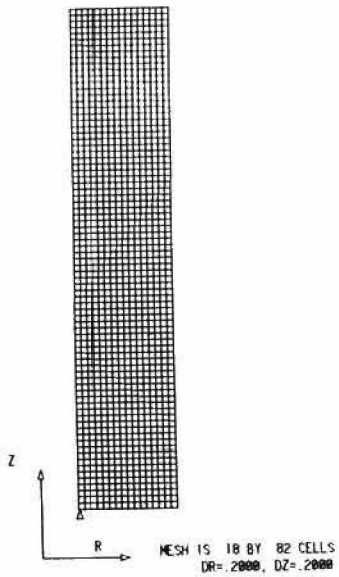


Figure 25
Computer Display of Computational Mesh

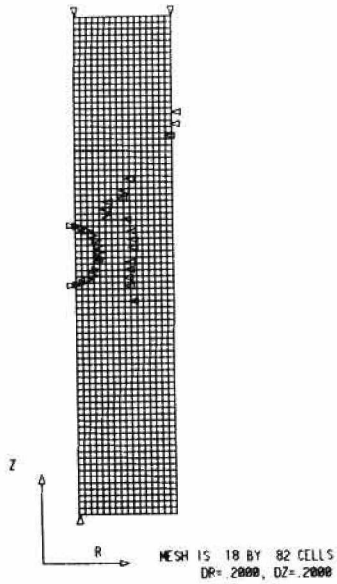
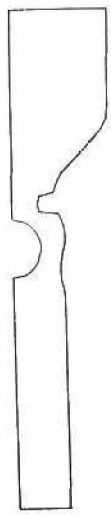


Figure 26

Computer Display of Computational Mesh Showing Points Input to Describe the Curved Boundary



RETURN

Figure 27

Computer Display of Final Figure Description

The Fluid Description Module

This module accepts the initial conditions which are to be imposed on the fluid. Using the display shown in Figure 28, the fluid description module operates in the same manner as the mesh description module. The variables input here include MU; the kinematic viscosity of the fluid, RHOA; the denisty of the fluid, NP, the density of the marker particles, V, NOT; the initial V-velocity, and U-NOT; the initial U-velocity.

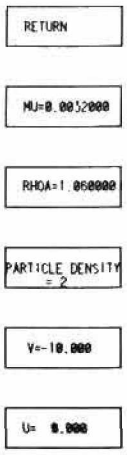


Figure 28

Computer Display for Graphic ABMAC Fluid Description

The Initialization Module

This module has been split into two different parts. The first handles the input of the problem parameters effecting convergence. The list of parameters is shown in Figure 29. DT is the time step, EPS1 is the convergence epsilon, and BETA is the relaxation constant. Input of these numbers is done just as in the mesh description module.

Figure 29

Computer Display for Initializing Convergence Parameters

The second portion of this module handles the input of those parameters effecting program control. These are shown in Figure 30. NDUMP is the number of cycles between creations of saved core images, NPIT is the number of pressure iterations allows per cycle, NSTOP is the number of cycles to be run, and NEDIT is the number of cycles between EDITS. FNAME is the name of the file to be created with the output of this program, and description is a short description of the program which has been described. Input is done in the same way as above.

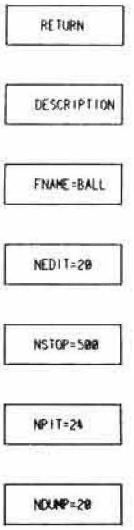


Figure 30

Computer Display for Initializing Control Parameters

When the EXIT function is requested, the input program calls upon two remaining subroutines. Although they serve important functions, they were not included as separate modules. The first of these is the CHECK routine. This produces the display shown in Figure 31. This is a complete description of the input problem. The mirror image of the description is rpoduced across the Z-AXIS, and the dimensions are labeled. Also shown are the initial and input velocities, and the text describing the problem. If this description is what the user wants, he pushes the switch on the MOUSE, and the various parameters describing the problem are written out onto a disk file. This is done

using the subroutine OUTP. This routine organizes the data and writes it out in the form expected by the ABMAC routines.

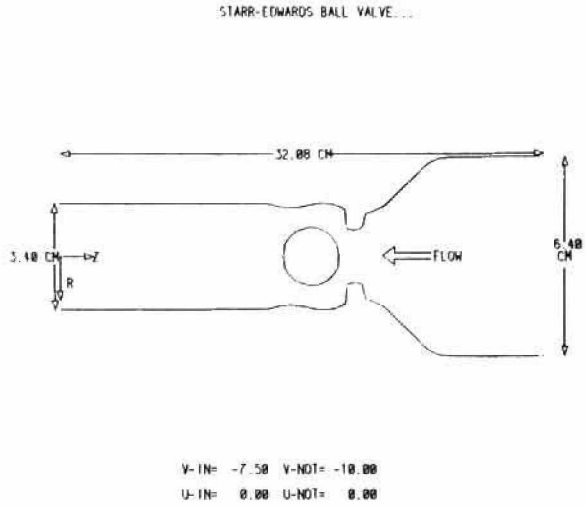


Figure 31
Computer Display of Complete Problem Description

B. SOLUTION OF THE PROBLEM

The use of interactive graphics in numerical problems was briefly discussed in the previous chapter. One area under consideration was the use of interactive computer graphics to monitor and interact with the computational program. The computational code in this program is based on Viecelli's ABMAC technique. Typical of finite difference methods, it is very large. The addition of an arbitrarily shaped boundary adds even more complexity to the program. If the design goals of an efficient program are to be met, this program cannot contain any code to do interactive graphics. It is still possible however, to graphically monitor and interact with the computational program.

The structure of the TENEX system on the PDP-10 allows two jobs to simultaneously share core. Using this facility, a program can be constructed which "SPIES" on the computational program, and visually displays what the computation is doing. This method of applying' interactive graphics has the following advantages:

1. The spy program is completely independent of the computational program. Small and compact, it adds no apparent load to the overall computer system.
2. The computational program, running independently of the spy program can run completely in the background, being detached from any I/O device.
3. The Spy program need only be run when desired. Its presence puts no additional load on the computation program.
4. The computation program has none of the overhead normally associated with graphics or interaction.
5. Since the data space of the computational program is available to the spy program, parameters effecting convergence or program control can be changed from the spy program.

The mechanics of core sharing are straightforward. During the initialization process in the computational program, the user has the option of requesting that a shareable file be created. At this point,

a file is created, and opened. Then pages from the fork containing the computation program are mapped into the file. The computational program is now free to run uninterrupted.

Whenever it is desired to examine the state of the computational program, the spy program is run. The spy program immediately maps into its own fork (or processing space) pages from the file created by the computational program. Whatever pages have been mapped in this way are actually now being shared by the two programs. This is illustrated in Figure 32.

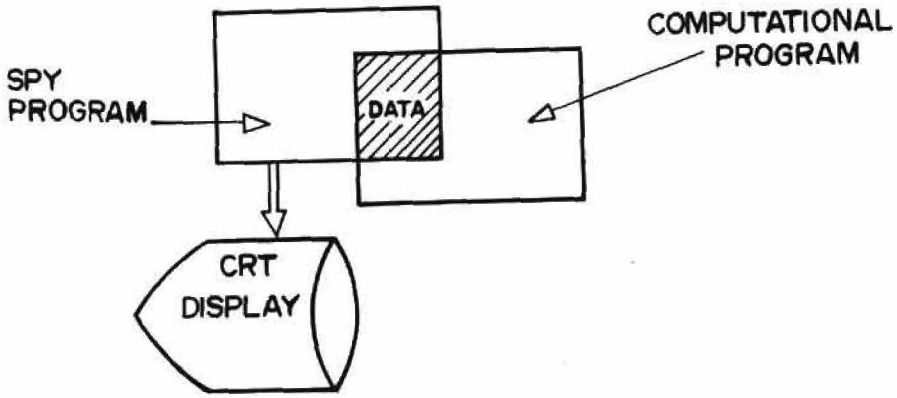


Figure 32

Block Diagram of Core-Sharing Used
to Achieve Interactive Graphics

The spy program then creates the display shown in Figure 33. This display presents to the user important information about the state of the computation. In the center of the display are shown the cycle number and the iteration number in that cycle. The display is dynamic in the sense that these numbers constantly change to reflect the current state of the computation.

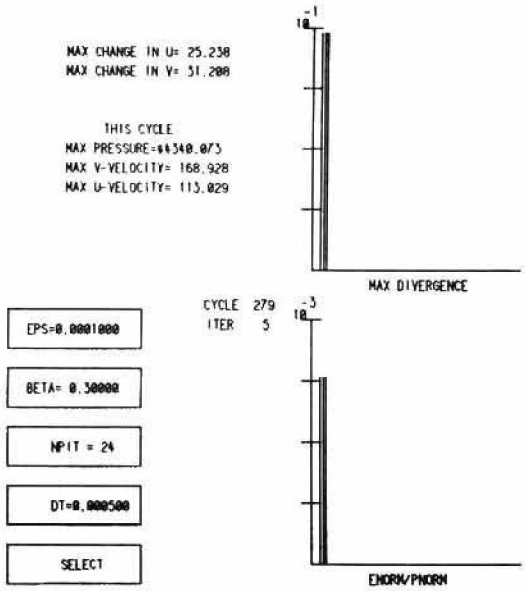


Figure 33

Computer Display of Current State of ABMAC Computation as Produced by Spy Program

In the upper left hand corner of the display are shown the maximum changes in the velocity components as measured during the current iteration. As the problem converges, these values will approach zero. In order to give additional meaning to the changes in velocity, the maximum values of the velocity components as well as of the pressure are displayed for the current cycle.

On the right hand side of the display are shown plots of the divergence and the error term for the pressure iteration. Bars are added to this plot at each iteration, so that the convergence or divergence of the iteration sequence can be monitored. The divergence is given in equation (1). As is shown, it is required that the divergence be equal to zero. In terms of the finite differences, this will never exactly be true. However, it should approach zero. The error term in the pressure iteration is

$$\text{ERROR:} \quad \text{PNORM}/\text{ENORM}, \tag{59}$$

where

$$\text{PNORM} = \sum_i P_i^2 \tag{60}$$

and

$$\text{ENORM} = \sum_i \Delta P_i^2 \tag{61}$$

The spy program is constructed so that it sleeps as long as there is nothing new to display. When a change occurs, it wakes up, makes the appropriate changes or additions to the display, and returns to a sleep state.

Along the left hand side of the display are five light buttons. The first four of these are parameters which affect the convergence of the program. By pointing at any one of these with the mouse, it is possible to change the value of that parameter in the computational program. EPS is the convergence epsilon, BETA is the relaxation constant, NPIT is the maximum number of allowable pressure iterations per cycle, and DT is the time step. The bottom light button, marked SELECT, gives the user some control over the course of the computation.

Pointing at the SELECT light button results in the display shown in Figure 34.

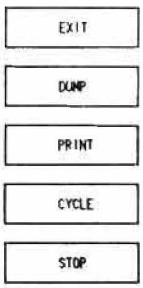


Figure 34

Computer Display of Spy Program Options

This display gives the user five options from which to choose. By pointing the mouse at one of the light buttons, flags are set in the computational program which set into motion the appropriate action. STOP sets a flag which causes the computational program to stop at the end of the current cycle. Before stopping, it will create a data file and a core image of itself. CYCLE sets a flag which causes the current iteration sequence to stop, and a new cycle to be started. PRINT causes a data file to be constructed at the end of the current cycle. DUMP causes the program to create a file which is a core image of itself. This file can be used as a starting place for continuing the program. EXIT halts the SPY PROGRAM without affecting the computational program.

These few capabilities provide all of the necessary interaction for this kind of a problem, and because of the way in which they are implemented, the computation procedure does not suffer from the problems associated with interactive graphics.

C. DISPLAYING THE SOLUTION

Using the computer to produce graphical images of a solution has become a well understood idea. Digital plotters, microfilm plotters, and CRT displays have given the computer user an important tool to help him solve his problems. Working with a visual representation of his solution he can gain valuable insight into his problem.

The program for displaying the solutions has three basic functions. The first is to read in the data from the disk files. Two files are required for displaying solutions. The first file which is needed is the file containing the problem description. This is used in drawing the curved boundaries for plotting solutions. The second file is the

data from the computational program. It should be noted here that all of the filenames are handled internally by the various programs. A master file name is all that is required in order for any program to get the required file from the disk.

The second function of the program is to compute the A matrix, the BNDRY matrix, and the SI matrix. The A matrix and the BNDRY matrix are matrices which are used in the countour plotting program. This program plots contours inside of arbitrarily shaped boundaries. The A matrix and the BNDRY matrix are used in specifying the boundary to the contour program.

The third function of the program is to produce displays of the solution space. Control over this function is similar in nature to the control of the input program, using the mouse and a computed GO TO statement to govern the branching. The display of figure 35 is drawn by the subroutine CNTRL1. These light buttons indicate the available functions for displaying the solutions to the problem. By pointing at one of these light buttons with the mouse, the appropriate type of solution will be displayed on the screen. Although the option for producing an isometric plot is shown, and a program for doing so is available, technical considerations made it impossible to include isometric plots of solutions in this report.

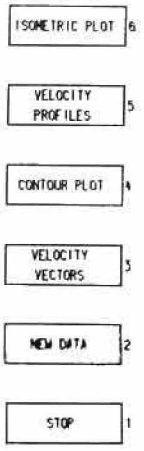


Figure 35

Computer Display of Output Functions.

Velocity Profiles

Velocity profiles can be drawn for any specified axial position. Initially the display shown in Figure 36 is drawn on the screen.

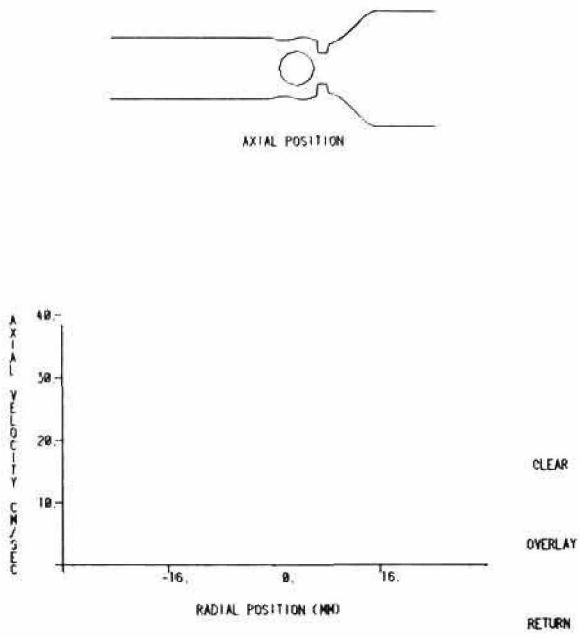


Figure 36

Computer Display of Basic Plot
For Velocity Vector Display

The horizontal and vertical axes are the radial position in millimeters, and the axial velocity in centimeters per second, respectively. On the right side of the screen are three light buttons. CLEAR causes the entire display to be erased, and the display of Figure 36 redrawn. OVERLAY allows more than one velocity profile to be displayed at the same time. RETURN returns control to the control portion of the main program. At the top of the display is a scaled down drawing of the geometry of the problem. As each velocity profile is drawn on the

display, a small arrow head indicates the axial position of that profile by pointing to the appropriate position on the scaled drawing. The display in figure 37 contains one velocity profile at the axial position

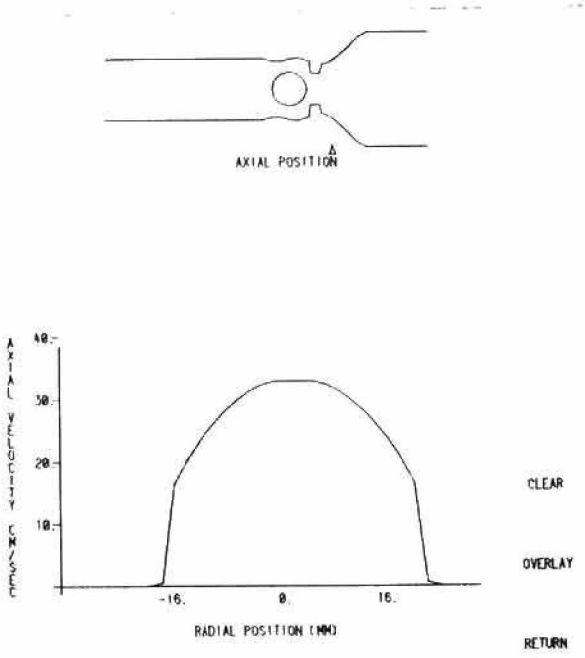


Figure 37

Computer Display of a Typical Velocity Profile

shown. All of the programs in this section were written to serve one purpose. That is to give the user a quick look into the solution of his problem. Because of this, they do not contain a lot of code required to produce "beautiful" pictures. For example, in the velocity profiles as displayed above, no data smoothing has been attempted. The plot shown merely connects existing data points. The display shown in Figure 38 illustrates the ability to produce a series of velocity profiles in the same display. This type of display is meaningful in

that it shows the development of the flow as it proceeds down the length of the tube.

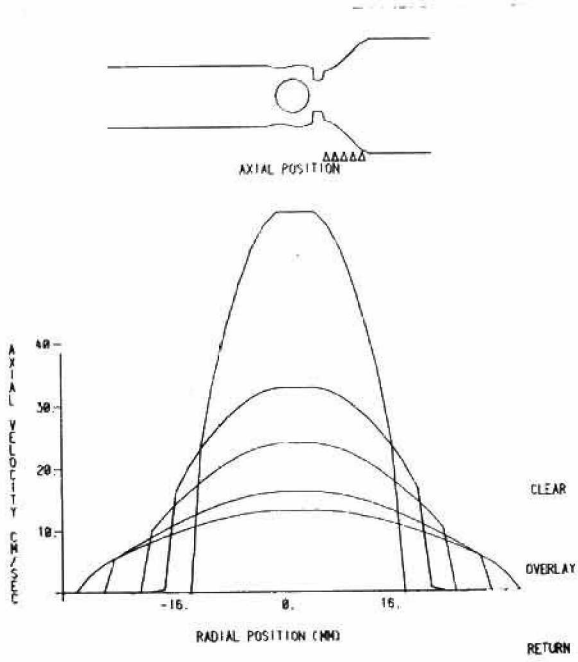


Figure 38

Computer Display of a Series of Velocity Profiles

Contour Plots

The initial display of the contour plotting program is shown in Figure 39. The user points the mouse at the light buttons on the right side of the display in order to choose the variables which he wants contoured. After doing this, he uses the mouse to outline the region he is interested in seeing onthe figure to the left of the light buttons. This allows him to "zoom in" on areas of particular interest. As the region of interest is narrowed down, the number of contour levels can be increased so that more detail is visible. Figures 40, 41, and 42 illustrate this effect on a contour plot of the stream function.

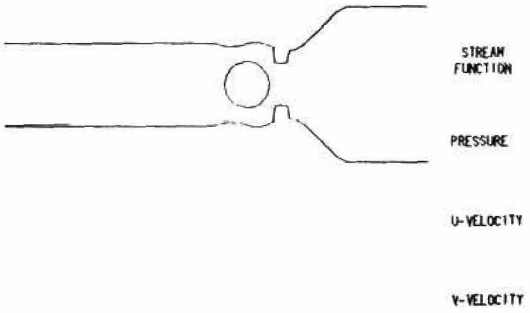


Figure 39

Initial Display for Contour Plotting

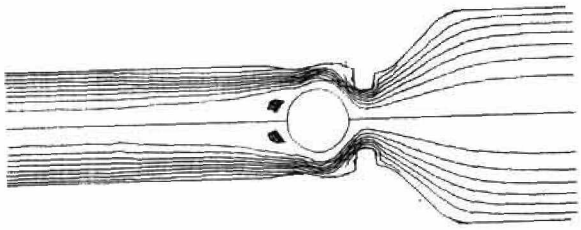


Figure 40

Computer Display of Stream Function

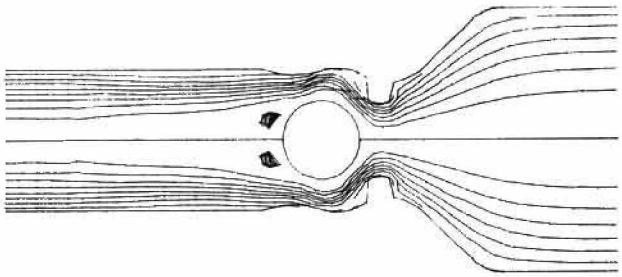


Figure 41

Computer Display of Same Stream
Function under a Different Window

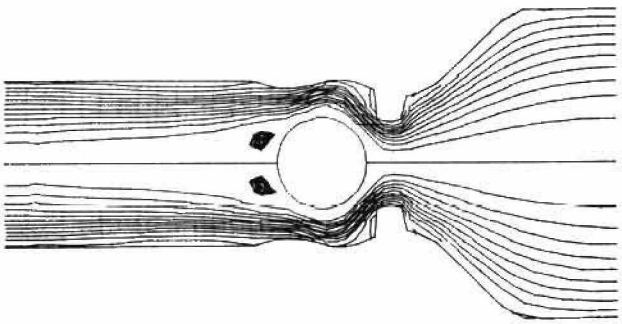


Figure 42

Computer Display of Same Stream Function
Under an Even Smaller Window, Illustrating the Zoom Effect

This zoom effect is done with a graphic technique known as "windowing." Windowing is the process of defining a region of interest and mapping it onto a particular area of the display screen. This technique is illustrated in Figure 43.

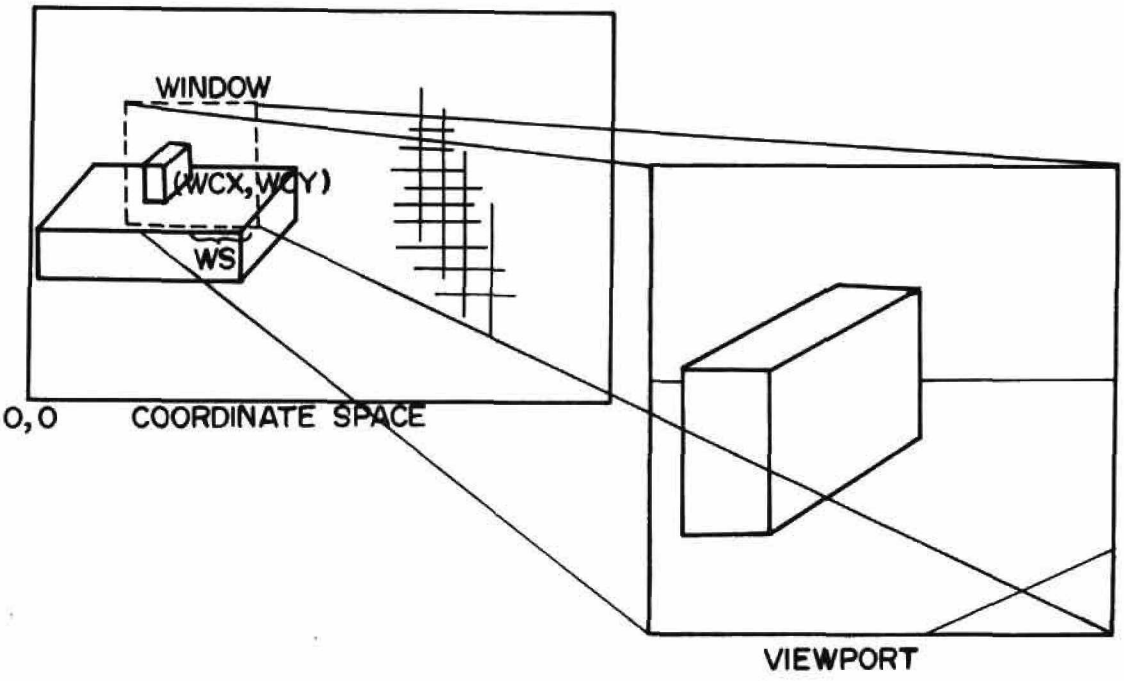


Figure 43
Mapping Function Performed by Windowing

If WCX and WCY are the coordinates in the coordinate system of the problem, of the center of the window, and WS is the measure from the center to one side, then the equations for the windowing transformation are

$$X_{vp} = \frac{1023}{2} \left(1 + \frac{X-WCX}{WS}\right) \quad , \tag{62}$$

and

$$Y_{vp} = \frac{1023}{2} \left(1 + \frac{Y-WCY}{WS}\right) \quad , \tag{63}$$

where the viewport is the entire face of the display screen. Also involved in producing this effect is a process known as "clipping." Clipping involves removing the lines outside of the region of interest from the display. In Figure 43, all of the points in the coordinate space undergo the transformations of equations (62) and (63). This means that the display will try to produce lines outside of its 1024 x 1024 addressable space. On the 1559, this produces only a maze of tangled lines crisscrossing the screen. The only way to display the picture then is to remove those lines not actually within the specified region of interest.

Figure 44 is a contour plot of the v-velocity component, and Figure 45 is a plot of the u-velocity component. In Figure 46, we have zoomed in slightly, and specified more contour levels, illustating the capability of producing more detail when it is required. Figure 47 is a contour of the pressure field.

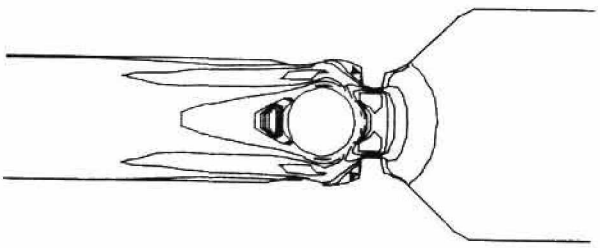


Figure 44

Contour Plot of V-Velocity Component

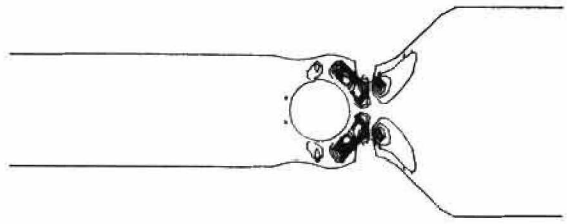


Figure 45

Contour Plot of U-Velocity Component

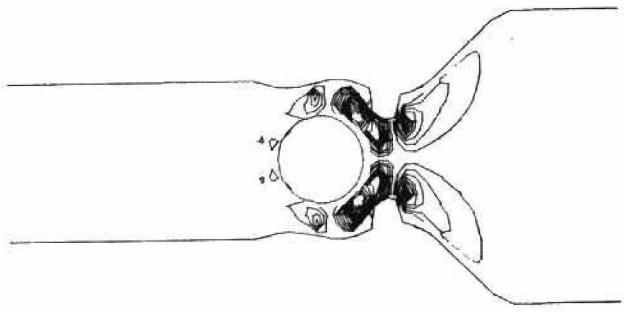


Figure 46

Contour Plot of U-Velocity Component With More Contour Levels Specified

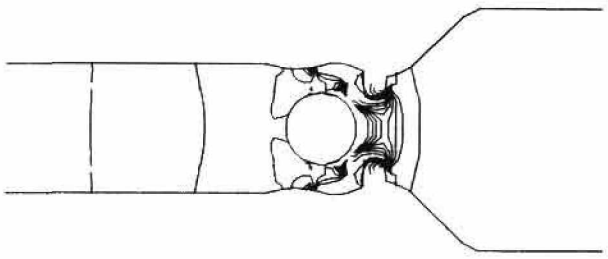


Figure 47
Contour Plot of Pressure

Velocity Vectors

Velocity vectors are produced by means of the subroutine VVECT. This program blanks out areas where the velocity is too small to produce a vector, so that some areas in the display will in fact appear blank. Figures 48 and 49 are displays of velocity vectors.

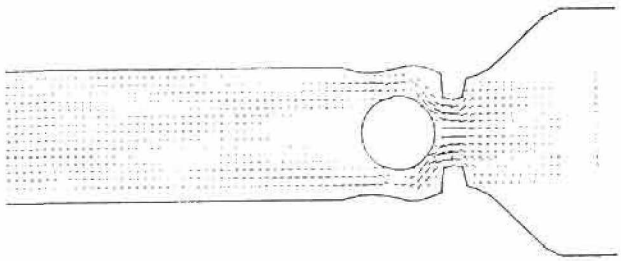


Figure 48
Computer Display of Velocity Vectors

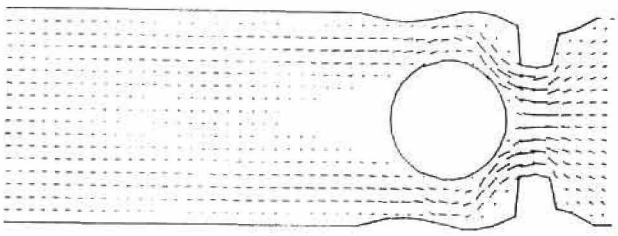


Figure 49

Same Display as Figure 48
With a Smaller Window

In the interactive graphics environment described in this chapter, the computer user can sit down at a conveniently placed graphics terminal. On the graphics display he can have his data presented in its natural form for both input and output. Decisions can be made at the display allowing several separate trials at a single session. Yet, this has been implemented with no appreciable load on the computer due to the addition of the interactive graphics.

CHAPTER SEVEN

RESULTS

A. NUMERICAL RESULTS

Several attempts were made in computing the flow about a ball type valve. The ABMAC technique as outlined in Chapter Four relies upon the assumption that a free-slip boundary condition exists at the arbitrarily shaped walls of the boundary and of the obstacle. As has been pointed out, in ABMAC these curved walls are treated as a free surface. In these surface cells, the pressure is computed according to the equation

$$P_{k,l}^{i+1} = P_{k,l}^{i} - \frac{\Delta\tau}{\delta} \{ [(V_p^{n+1})^{i} - V_b(r,t)] \cdot \hat{n} \}_{k,l} . \qquad (64)$$

Here $\hat{n}$ is the unit normal defining the boundary segment associated with cell (k,l), $(V_b(r,t))_{k,l}$ is the velocity of the midpoint of the segment, and V_p^{n+1} is the liquid velocity at the midpoint of the segment. According to this equation, the pressure is not adjusted proportional to the divergence of the cell, but rather proportional to the flux across the boundary. This certainly represents a free-slip boundary condition. There can be no flow across the boundary wall, instead the fluid flow is forced to be tangential to the boundary.

The velocity vectors in Figure 50 illustrate this effect. Note that there are velocity vectors right on the surface of the ball. As a result of this, the velocity directly behind the ball does not drop off as expected. However, a strange side effect is shown in Figure 51.

In this display, note how the velocities far downstream react. They actually approach zero at the center of the tube.

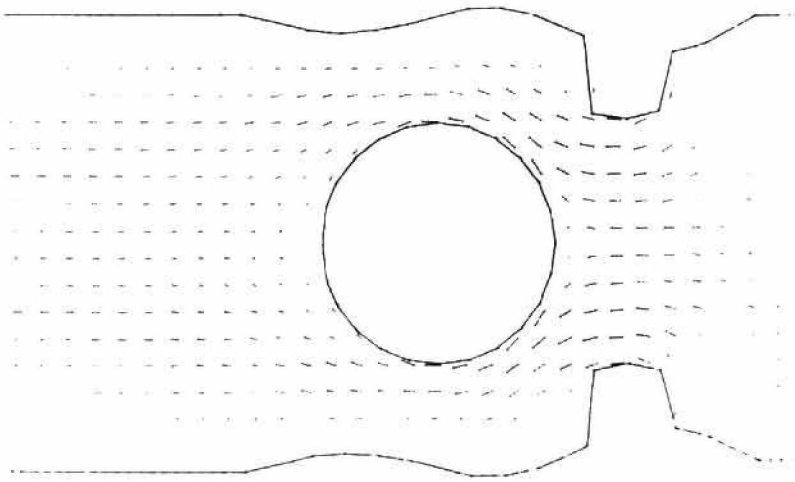


Figure 50

Computer Display of Velocity Vectors on
Ball with Free-Slip Boundary Condition

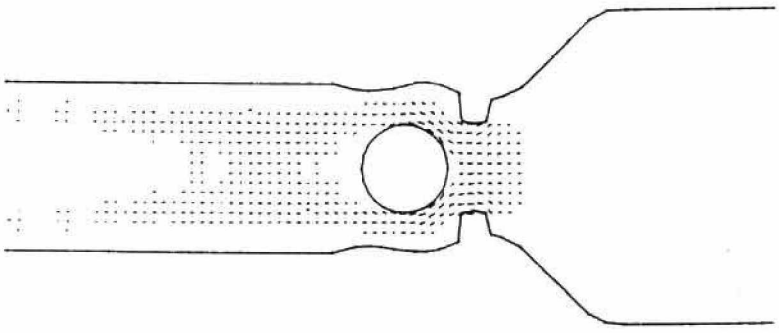


Figure 51

Computer Display of Downstream Anomoly
in Velocity Vectors with Free-Slip Boundary Condition

In order to pursue a more realistic solution, the case where the boundary is no-slip must be considered. By a no-slip boundary, we mean one on which both the normal and tangential velocity components are zero.

In the original MAC Technique, a no-slip boundary is specified as follows:

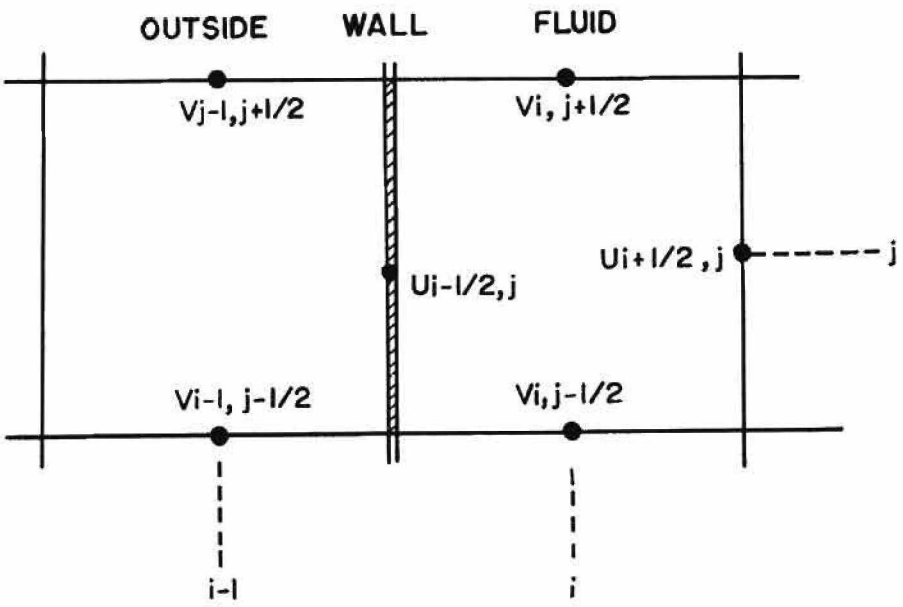


Figure 52
Specification of MAC No-Slip Boundary

$$U_{i-1/2,j} = 0 \qquad (65)$$

$$V_{i-1,j+1/2} = -V_{i,j+1/2} \qquad (66)$$

$$V_{i-1,j-1/2} = -V_{i,j-1/2} \ . \qquad (67)$$

This scheme assumes that the wall is coincident with some part of the computing mesh. If, however, the wall is represented by an arbitrary

segment cutting through the cell, it should still be possible to specify the across-the-wall velocities such that a no-slip boundary condition exists.

There appear to be two classes of boundary configurations that need to be considered. The first, represented in Figure 53, contains those instances where there is an empty cell only on one side of the cell containing the wall segment.

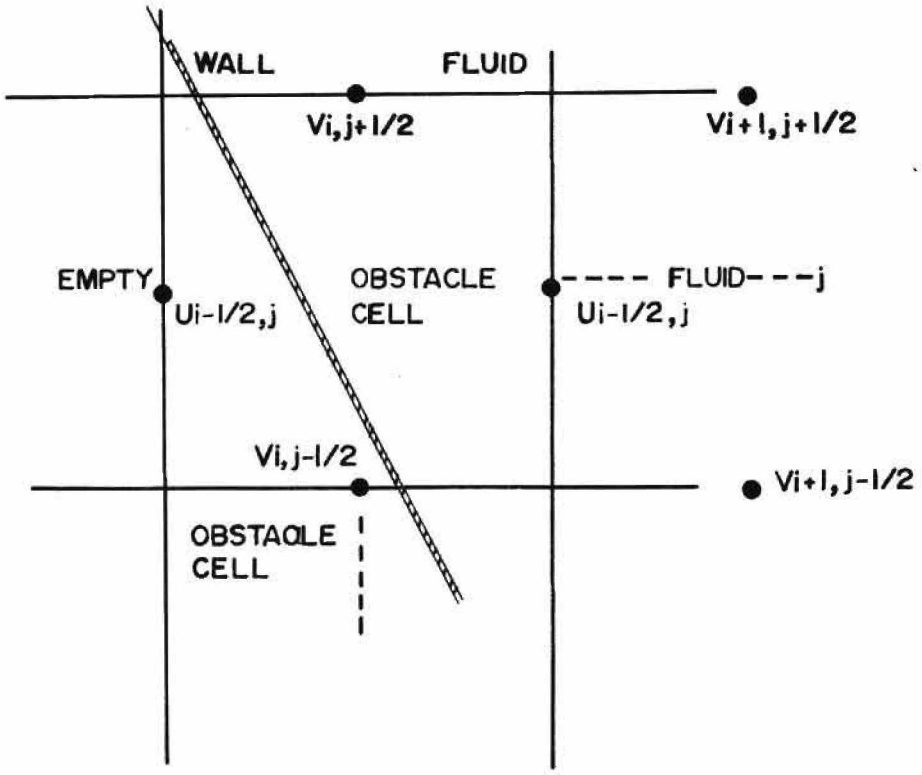


Figure 53

ABMAC Cell with Only One Empty Cell Bordering the Cell Containing the Boundary Segment

In this class of cells, the velocities $V_{i,j+1/2}$, $V_{i,j-1/2}$, and $U_{i-1/2,j}$ need to be computed such that a linear interpolation across the boundary will yield zero velocity components on the wall. In order to compute the required velocities, a linear interpolation is

used in the direction most nearly normal to the boundary segment. In Figure 54 the case under consideration is shown.

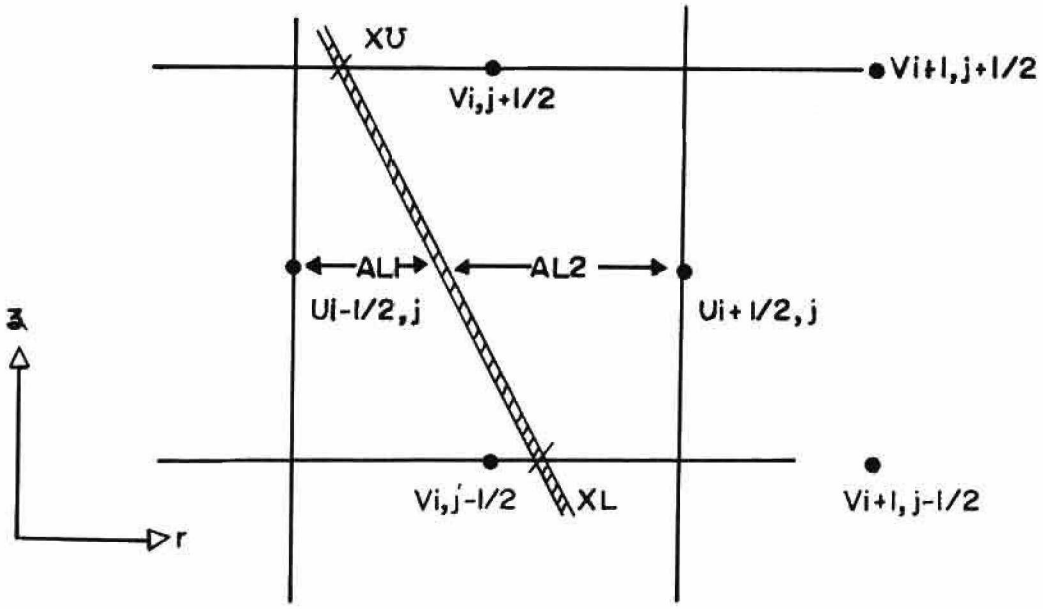


Figure 54

Linear Interpolation of Boundary Values for the Class of Cells Illustrated in Figure 53.

Since the midpoint of the segment is known, for $U_{i-1/2,j}$

$$U_{i-1/2,j} = -(U_{i+1/2,j} * AL1) / AL2, \tag{68}$$

where AL1 and AL2 are easily computed.

In order to compute $V_{i,j-1/2}$, the point XL where the wall cuts the mesh must be known. Given PNORX and PNORY, the midpoint of the segment, and DNORX and DNORY, the direction cosines of the segment, the point XL is given by

$$XL = (Z_{j-1/2} - PNORY) / TAN + PNORX, \tag{69}$$

where

$$TAN = DNORX / DNORY. \tag{70}$$

Now with XL known, $V_{i,j-1/2}$ can be computed as in equation (68). When the wall crosses the mesh outside of the unknown velocity as in the case of $V_{i,j+1/2}$, it should be noted that equation (68) still produces a correct velocity for that point.

This system works for all orientations of the boundary, where only one side of the boundary cell faces an empty cell. The equations for determining XU, XL, AL1, and AL2 are a little different for each case, but the general approach is the same.

The second class of problems are those in which an empty cell faces the obstacle cell on two sides. This is shown in Figure 55.

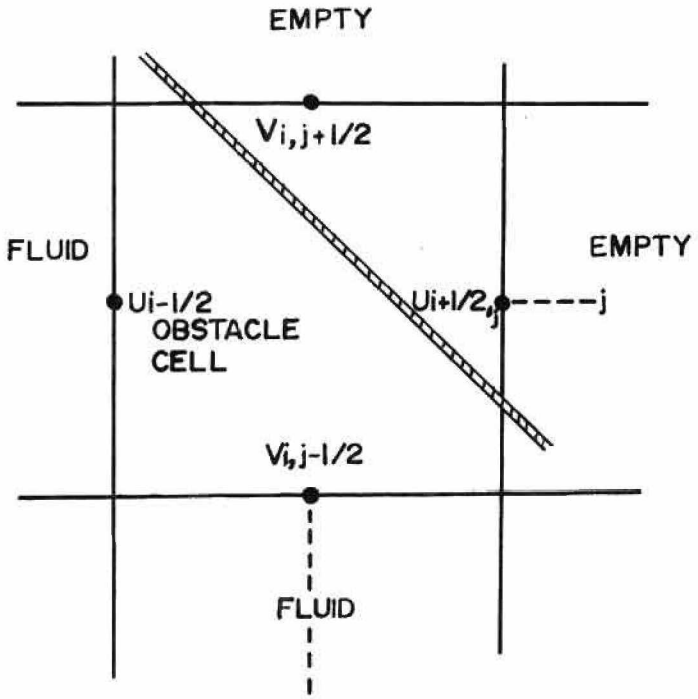


Figure 55

ABMAC Cell with Two Empty Cells Bordering the Cell Containing the Boundary Segment

In this case, $V_{i,j+1/2}$ and $U_{i+1/2,j}$ will be computed using the velocities directly across the cell. In order to do this, the lengths XL, and YL must be known, as shown in Figure 56.

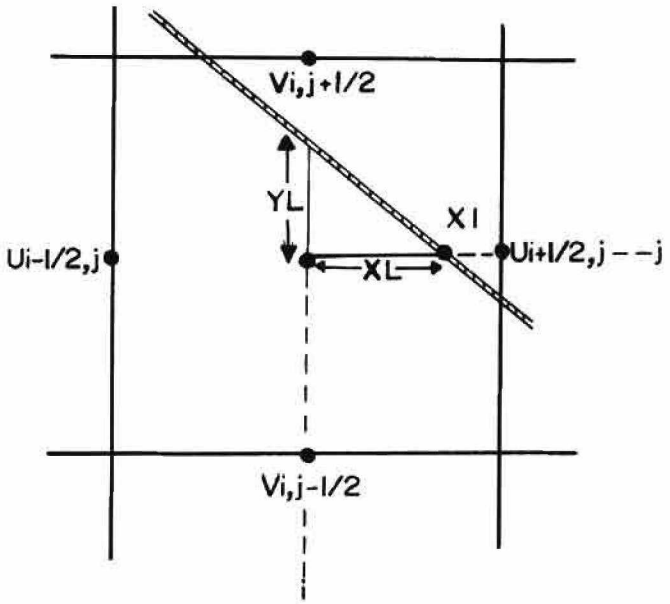


Figure 56

Computation of XL and YL for Determining the Boundary Values for the Class of Cells Illustrated in Figure 55

X1, the point where the boundary segment crosses the line $z = z_i$ is given as

$$X1 = (Z_j - PNORY) / TAN + PNORX, \qquad (71)$$

where PNORY, PNORX, and TAN have identical meanings as in equation (69). With X1 known, the lengths XL and YL can be computed and the velocities found using a linear interpolation formula.

Results using the no-slip boundary condition represent a far better solution than those with the free slip condition. The velocity profiles shown in Figures 57 and 58 are at positions upstream from the ball. Those in Figures 59, 60, and 61 are at positions downstream from the ball. These profiles represent a realistic simulation of the flow about a ball type valve.

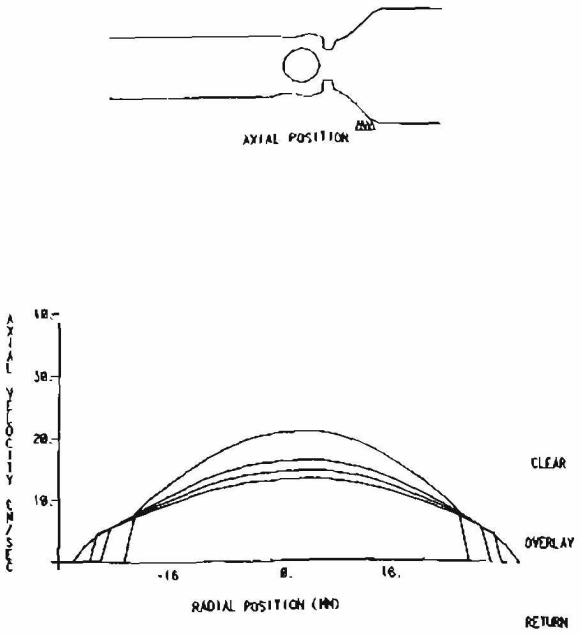


Figure 57

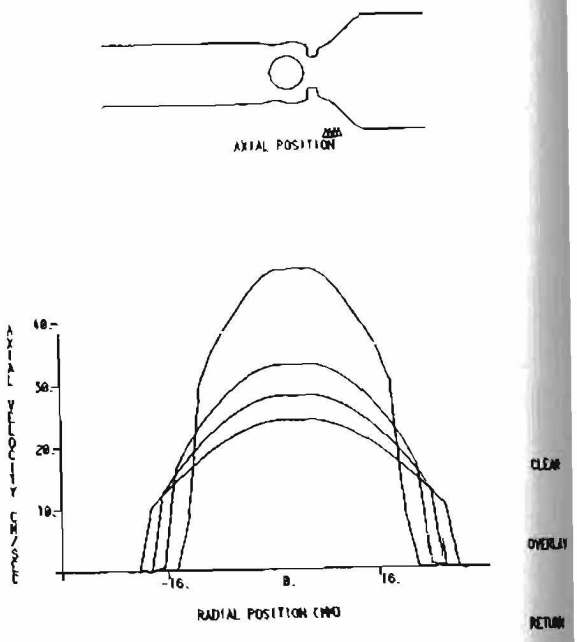


Figure 58

Upstream Velocity Profiles
For No-Slip Boundary Condition

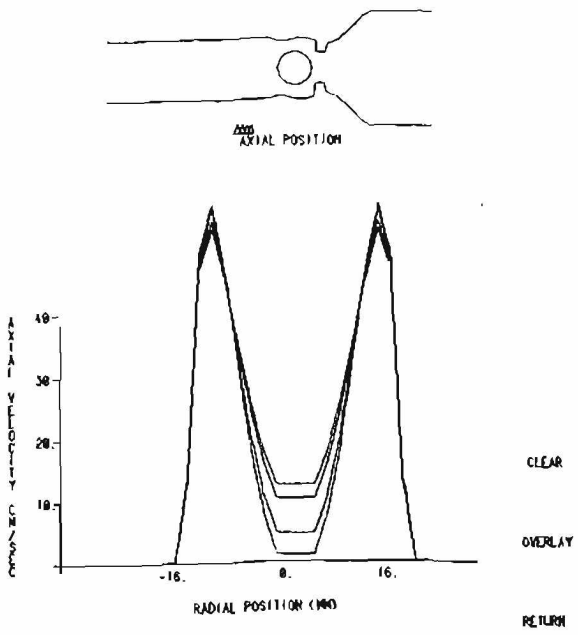


Figure 59

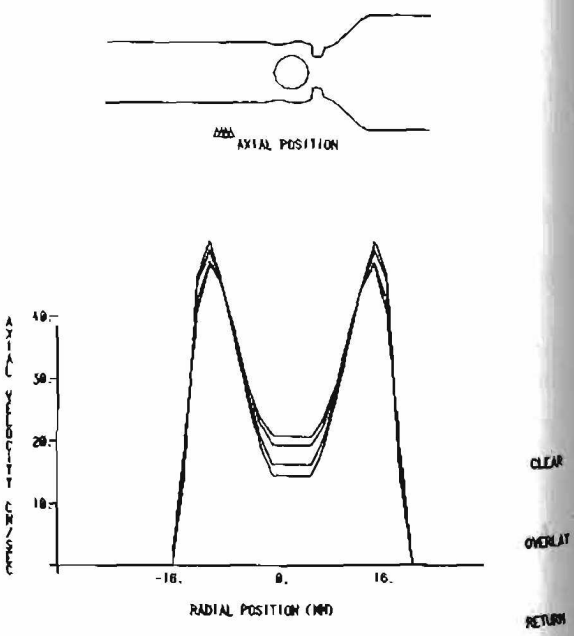


Figure 60

Downstream Velocity Profiles
for No-Slip Boundary Conditions

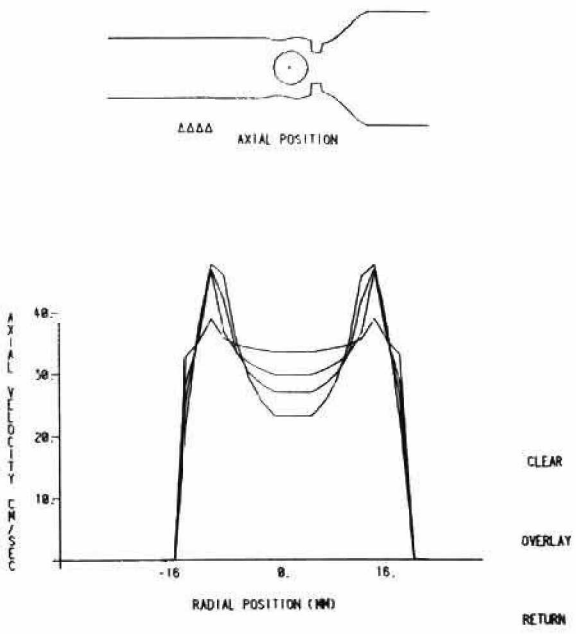


Figure 61

Downstream Velocity Profiles
for No-Slip Boundary Conditions

Figure 62 and 63 are comparisons of these velocity profiles with selected profiles produced experimentally by Wieting. The shapes of the profiles are very similar, but note that those produced numerically show slightly faster velocities.

The differences can be attributed to several factors. First, the numerical solution represented here is a steady state solution, i.e. given some initial conditions and a set of boundary conditions, the computer iterates on the given equations until the solution satisfies the given conditions. The experimental solution on the other hand is time dependent. The pulsatile motion of the fluid, the motion of the ball, and other time dependent factors will certainly make a difference. The exact moment at which the profiles were measured could in itself

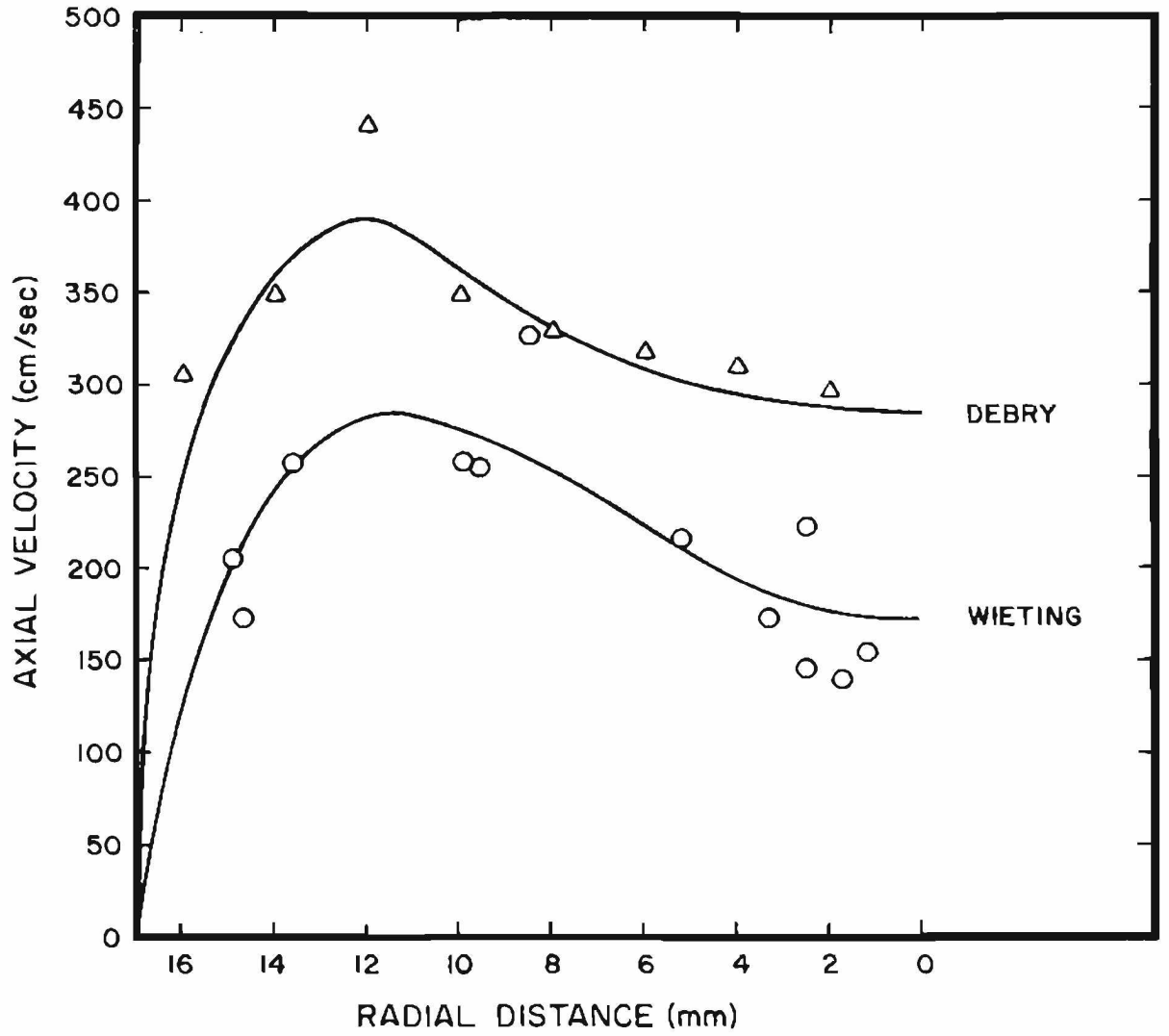


Figure 62
Comparison of Calculated Velocity Profiles with Experimental Data (Wieting), at position z = 79mm., downstream of sewing ring position.

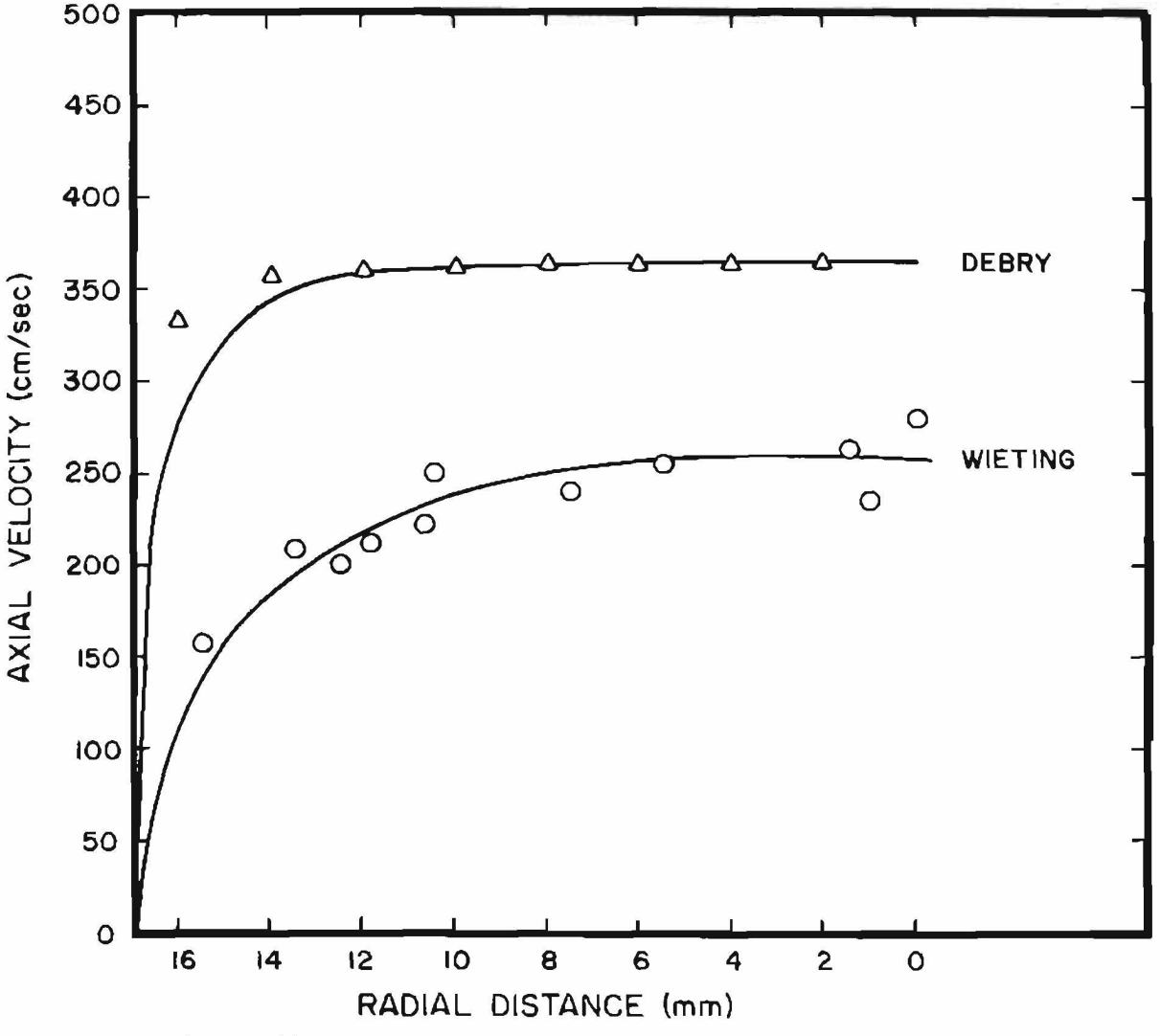


Figure 63
Comparison of Calculated Velocity Profiles with Experimental Data (Wieting), at position z = 68mm., downstream of sewing ring position.

produce different results. From a numerical standpoint, a more accurate solution could certainly be obtained if a finer computational mesh were used.

Figures 64, 65, 66, and 67 are contour plots of the stream-function, V-velocity component, U-velocity component, and pressure. Figure 68 shows the velocity vectors for the solution. These figures represent a realistic representation of the flow.

B. THE COMPUTER PROGRAM

In addition to producing numerical results, the goal of this research was to produce a series of programs which used the computer to its best advantage.

In order to make the ABMAC code more efficient for use on the PDP-10 time sharing system, several changes were made in the program.

The first major change involved separating the initialization portion of the program from the computational part. This is a significant change for many reasons. During the initialization procedure, the curved boundary segments are defined, fluid particles are created, and the cell flags are marked. This involves a great deal of time, program space, and temporary storage. By separating this code from the rest of the program, the amount of core required to run in dropped from 96K to 48K. Under our time sharing system, this means a considerable increase in efficiency.

During a long production run, the set-up time may be insignificant when compared to the total run time. However, testing and debugging pose an entirely different picture. Usually, interest lies only in the

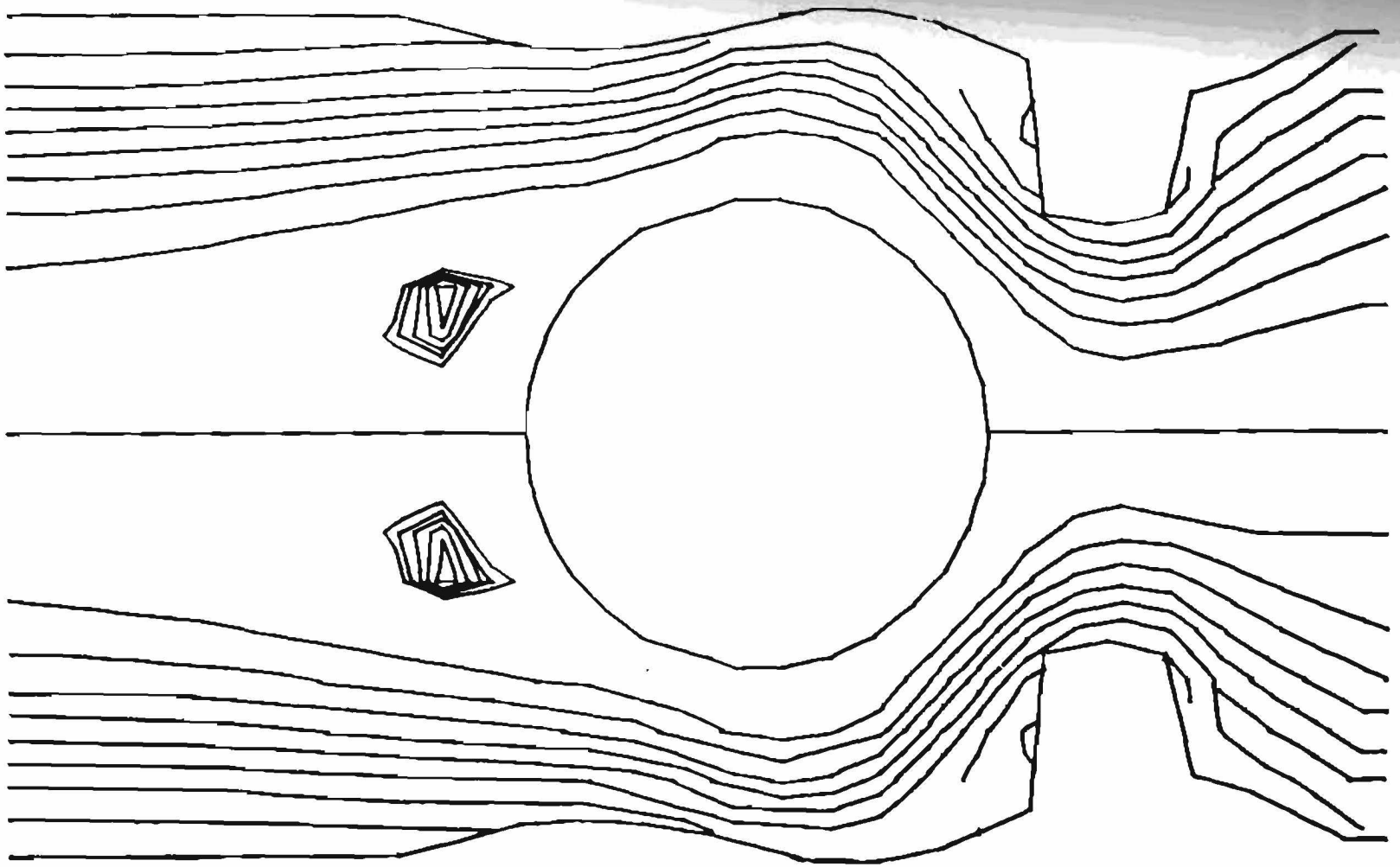


Figure 64

Computer Display of the Contour Plot of the Stream Function
for No-Slip Boundary Solution of Blood Flow About
a Starr-Edwards Ball Type Valve

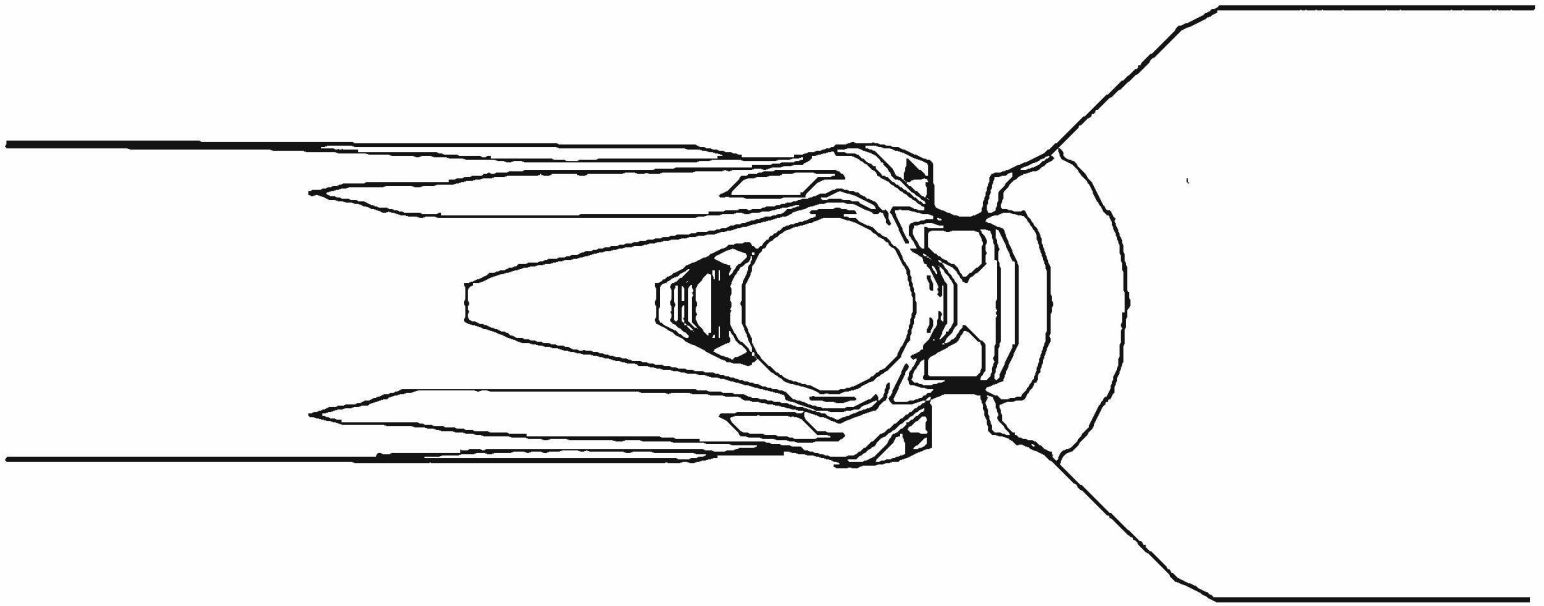


Figure 65

Computer Display of the Contour Plot of the V-Velocity Component
for No-Slip Boundary Solution of Blood Flow About
a Starr-Edwards Ball-Type Valve

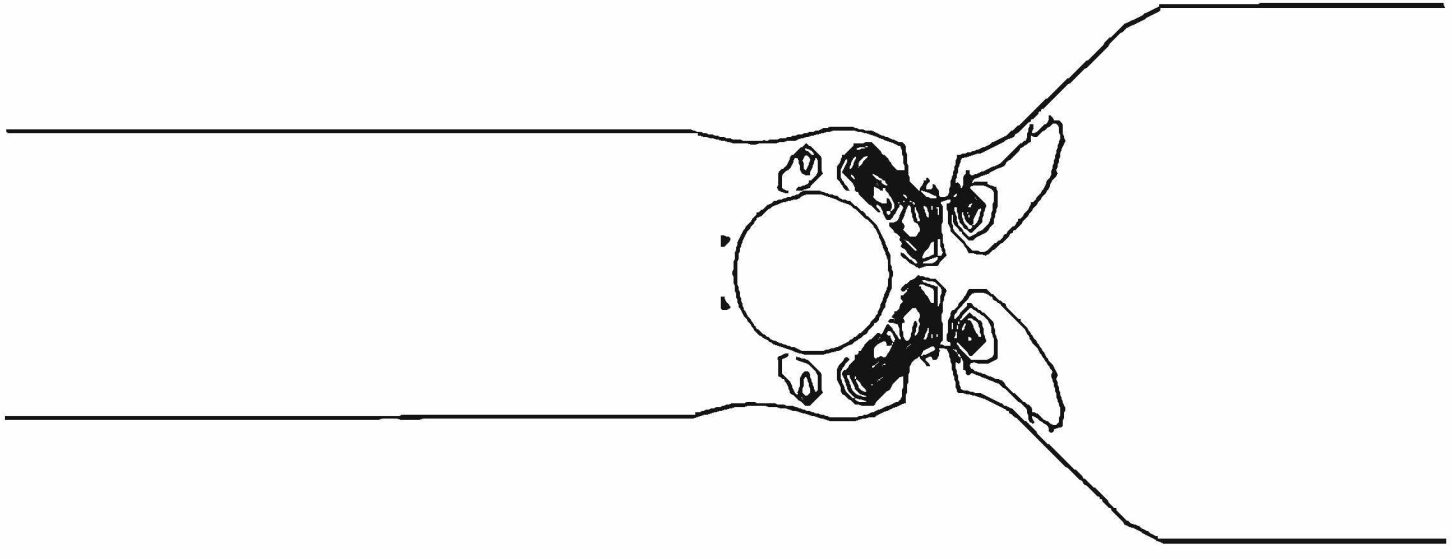


Figure 66

Computer Display of the Contour Plot of the U-Velocity Component
for No-Slip Boundary Solution of Blood Flow About
a Starr-Edwards Ball-Type Valve

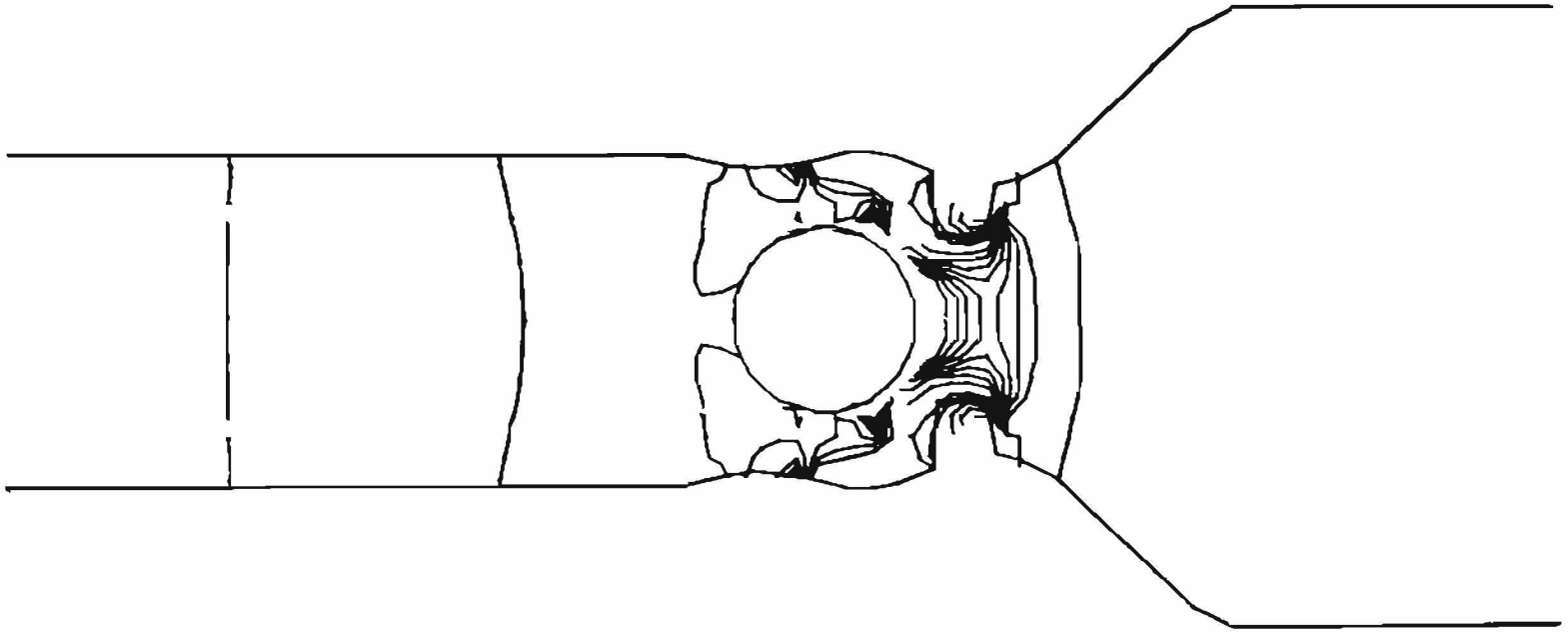


Figure 67

Computer Display of the Contour Plot of the Pressure Field
for No-Slip Boundary Solution of Blood Flow About a
Starr-Edwards Ball-Type Valve

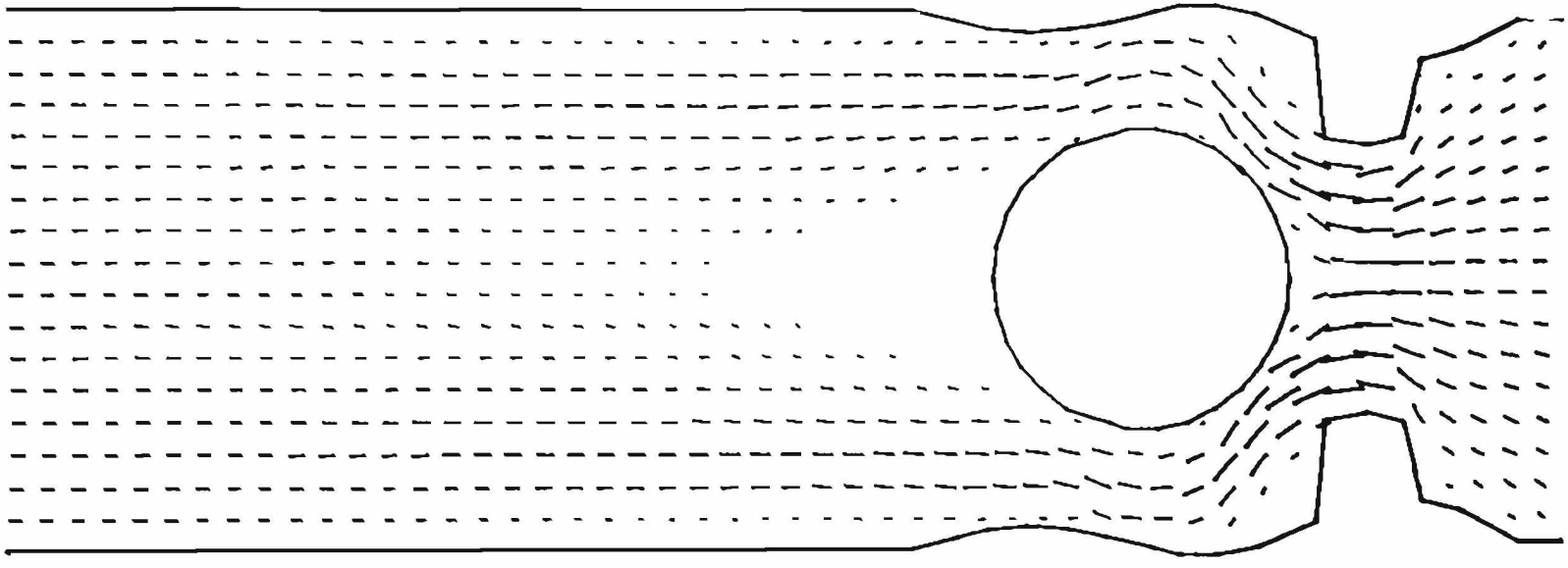


Figure 68

Computer Display of Velocity Vectors for No-Slip Boundary
Solution to Blood Flow About a
Starr-Edwards Ball-Type Valve

first one or two time cycles. In this frame of reference, the set up time becomes a large part of the total run time. Therefore, by removing that code from the rest of the program, the set-up for a particular test need only be done once. Each test can use the same stored values of initial data. The overall scheme of the initialization procedure is shown in Figure 69.

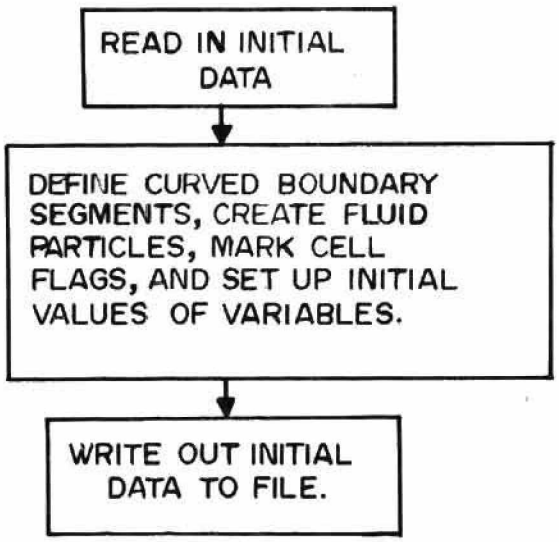


Figure 69

Block Diagram of Initialization Procedure
For Modified Version of ABMAC Code

The second change involved combining several pieces of code into common subroutines. The major changes in this respect involved the velocity computation code. The ABMAC technique requires a velocity computation at two distinct times. As the pressure is re-computed for each cell, the velocities are simultaneously changed for that cell. In addition, at the end of each pressure iteration, the entire field of

velocities is re-computed. Although computationally the same, the codes for these two distinct subroutines are different.

It was a simple matter, to combine these subroutines, using a check to determine which function to perform. Subroutine VEL55 for example is called with three arguments. If one of these arguments is zero, it indicates that the entire mesh is to be swept. If non zero, the arguments indicate which cell the velocities are to be computed in. The code to do this is as follows:

```
      SUBROUTINE VEL55 (KM, K, L)
      COMMON STATEMENTS
      .
      .
      .
      IF (KM, NE . 0) GO TO 10
      DO 1 L = 2, LMZ
      KMM = L  KMAX
      DO 1 k = 2, KMZ
      KM = K + KMM - KMAX
10    KT = KM + KMAX
      KB = KM - KMAX
```

The third change involved the determination of a system for setting and checking cell flags. The original ABMAC code, written in LRLTRAN for the CDC 7600, took advantage of several nice language features not generally available. In particular, LRLTRAN contains a byte specification and a structure specification. In ABMAC, the cell glags SUR, FULL, EMP, BND, and etc., were specified as 1 bit bytes. These flags were then combined into one word using the structure specification, and that set of flags was made into an array.

These 1 bit bytes could then be accessed by name. For example:

```
OB (KM) = 1,

FULL (KM + 1) = 0 .
```

Treating these quantities as truth values made it easy to check cell flags and make decisions based on their contents. For example:

```
TEST = FULL (KM). UN. (SUR (KM) . INT. (.NOT.COR(KM)))
IF (TEST) 20, 192.
```

In this illustration, are shown three other items peculiar only to LRLTRAN. .UN. is a bit or, .INT. is a bit by bit AND, and the statement

```
IF (TEST) 20, 192
```

means go to statement 20 if test is true, and go to statement 192 if it is false.

In order to simulate this scheme as efficiently as possible, a table was devised which marked the cells as powers of two. The flags are defined as

```
   SUR =     4
  FULL =     8
   EMP =    16
   BND =    32
    IN =    64
   OUT =   128
 FRSLP =   256
 NOSLP =   512
EMPBND =  1024
    OB =  2048
   COR =  4096
    OK =  8192
   GAS = 16384
   ARB = 32768
```

A cell then can be marked as full by writing

```
FLAG (KM) = LFAG (KM) . OR. FULL,
```

while checking the cell to see if it is full can be done by writing

```
IF ((FLAG(KM), AND.FULL).EQ.FULL) ...  .
```

It can be seen that while it takes more source code to write the same

thing, the code for the PDP-10 is just as efficient both space-wise and time-wise.

The final major change was one dictated by the heavy load this type of problem placed on the system. Since the problem is being run on a time-shared system, it must compete in time with other jobs. The scheduler on the PDP-10 is such that a long compute bound job is not favored. In order to take advantage of all possible time available on the machine, and yet not load the system, the following scheme was used.

The PDP-10 allows background jobs to be initiated using a system program called FIB. FIB queues up requests for background jobs, and runs them whenever the machine load average drops below a specified point. Jobs are run according to the requested run times, i.e., a job requesting two minutes of CPU time will run before one requesting four minutes. If a process can be broken up into discreet segments, such that each segment uses two minutes or less, the FIB program will use all available machine time to run these segments without loading the machine.

ABMAC runs through one time cycle in about 12 minutes. This provides a convenient place to segment the process. Instead of writing out a file at the end of each segment, and reading it in at the beginning of the next, the program maps a complete core image of itself onto a file called P.DMP. At the beginning of the next cycle the command RUN P.DMP is sufficient to start the next cycle with all of the proper data.

In the event that the machine crashes during this mapping operation, two different versions of P.DMP are written out, the program alternating

between them. This insures that there is always one complete, good version of P.DMP. on the system.

In order to keep the program running, the FIB request must generate the next FIB request as well as the next program segment. Also, it must be sure that it uses the correct version of P.DMP. The FIB request necessary to do this is shown here. The machine responses are underlined.

```
(1)      @ FIB
(2)      @@ <SYSTEM> EXEC. SAV
(3)      @@ CONSOLE-OUTPUT (TO FILE)  NIL.DAT  [NEW FILE]
(4)      @@ RUNTIME (MAXIMUM IN MINUTES)  2  [CONFIRM]
(5)      @@ COMMAND-TEXT (ENDING WITH ↑ Z)
(6)       * COPY O-REQUEST.INPUT (TO) REQUESTED-F-I-B. INPUT
(7)       * RUN GETIT
(8)       * P
(9)       * Y
(10)      * Y
(11)      * N
(12)      * ↑ Z
(13)      @
```

Line (1) calls up the FIB program. Line (2) tells FIB that it is to run the system EXEC routine. Line (3) specifies that any console output is to go to a file named NIL.DAT. Line (4) specifies a two minute maximum run time. Line (5) asks for a list of commands and/or text to be executed by the routine specified in line (2). Line (6) copies from a file O-REQUEST.INPUT, the next FIB request. This is put into the queue by writing it onto a file called REQUESTED-F-I-B.INPUT. Line (7) tells the system to run a program called GETIT. GETIT looks at the version numbers of P.DMP, selects the proper one, and maps it

onto core. Control is then passed to P.DMP. Lines (8), (9), (10), and (11) contain text which is used by P.DMP. Line (12) terminates the list, and line (13) returns to the EXEC.

CHAPTER EIGHT

CONCLUSIONS

The numerical results of this study illustrate the applicability of a unique numerical technique to an extremely important problem. The development of a method for specifying a flexible, arbitrarily shaped, no-slip boundary provides an important step in studying blood flow characteristics. The technique developed in this report is based on the Marker and Cell methodology produced at Los Alamos Scientific Laboratory. This method, using marker particles, provided a powerful method for defining free surfaces in a fluid flow study, as well as the ability to solve the problem in terms of the primitive variables of velocity and pressure. The marker particle concept allowed the definition of an arbitrarily shaped, flexible boundary in the flow problem, as demonstrated in the ABMAC method developed at the Lawrence Livermore Labs. Adding the no-slip boundary concept developed in this report to ABMAC provides a powerful tool for blood flow studies.

It should be noted that in the steady-state, confined flow solutions presented in this paper, the marker particles used originally in MAC have no significance, and in fact have been eliminated from the computation altogether. Therefore, no figures containing marker particles appear in the text. However, once the walls are allowed to flex, or the ball is allowed to move, the particles again become important in maintaining the definition of the walls.

In addition to the development and application of a new numerical method, this report has illustrated the kinds of techniques necessary

prime cause of some 800,000 heart attacks in the United States each year is an infarction of a coronary artery. Such an obstruction of the local circulation by an advanced plaque formation, thrombus or embolus is involved in a many-branched amplification of our present studies concerning arterial geometrics. It is realized that the inclusion of moving (pulsing), free boundaries and their computer graphics simulations into the problem will allow tethering, tapering and reflection effects; conceptual requirements that would aid in advancing the hypotheses of myocardial infarction mechanisms. Knowledge of such mechanisms may not be attainable by any other analytical means except by such resulting computer simulations.

BIBLIOGRAPHY

1. Hull, L.W.H., History and Philosophy of Science. London: Longmans, Green and Co., 1959.

2. Ledley, R.S., Use of Computers in Biology and Medicine. New York: McGraw Hill Co., 1965.

3. Townsend, J.C., Uses of Computers in the Medical Field. NASA Technical Report, 1965.

4. Whiteby, L.G., and Lutz, W., Principles and Practices of Medical Computing. London: Churchill Livingston Co., 1971.

5. Greenfield, H., and DeBry, R., An Application of Computer Graphics: Two Concurrent Investigations Within The Medical Field. Computer Science Technical Report UTEC-CSc-71-115, Salt Lake City, Utah, 1971.

6. Starr, A., and Edwards, M.L. "Mitral Replacement: Clinical Experience with a Ball Valve Prosthesis," Annals of Surgery, 154:726, 1961.

7. Hirt, C.W., and Cook, J.L., "Calculating Three-Dimensional Flows Around Structures and Over Rough Terrain." Journal of Computational Physics, Vol. 10, No. 2, October 1972, pp. 324-340.

8. Braunwald, N.S., and Detmer, D.E., "A Critical Analysis of the Status of Prosthetic Values and Homografts." Progress in Cardiovascular Diseases, Cardiovascular Surgery I., Vol. 11, No. 2, September 1968, p. 113.

9. Wieting, D.W., Dynamic Flow Characteristics of Heart Valves. Unpublished Doctoral Thesis, University of Texas, May 1969.

10. Bird, R.B., Steward, W.B., and Lightfoot, E.N., Transport Phenomena. New York: John Wiley and Sons, Inc., 1960.

11. Batchelor, G.K., An Introduction to Fluid Dynamics. London: Cambridge University Press, 1960.

12. Forsythe, G.E., and Wasow, W.A., Finite Difference Methods for Partial Differential Equations. New York: John Wiley and Sons, Inc., 1960.

13. Varga, R.S., Matrix Iterative Analysis. Englewood Cliffs, N.J.: Prentice-Hall, 1962.

14. Fromm, J.E., A Method For Computing Nonsteady, Incompressible, Viscous Fluid Flows. Los Alamos Scientific Laboratory Report LA-2910.

15. Liebman, H., Sitzungsberichte Der Bayerischer Akademie Der Wissenschaften, 1918.

16. Pearson, C.E., "A Computational Method for Viscous Flow Problems," J. Fluid Mech., Vol. 21, 1965, p. 116.

17. Esch, R.E., An Alternative Method of Handling Boundary Conditions, and Various Experiments in the Numerical Solution of Viscous Flow Problems. Sperry Rand Research Center, Sudbuey, Mass., SRRC-RR-64-64, 1964.

18. Harlow, F.H., "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with a Free Surface," The Physics of Fluids, Vol. 8, No. 12, December 1965, pp. 2182-2189.

19. Harlow, F.H., Welch, J.E., Shannon, J.P., and Daly, B.J., The MAC Method, Los Alamos Scientific Laboratory Report LA-3425.

20. Amsden, A., and Harlow, F.H., The SMAC Method - A Numerical Technique for Calculating Incompressible Fluid Flows, Los Alamos Scientific Laboratory Report LA-4370.

21. Viecelli, J.A., "A Method for Including External Boundaries in the MAC Incompressible Fluid Computing Technique," Journal of Computational Physics, Vol. 4, 1969. pp. 543-551.

22. Chorin, A.J., "Numerical Solution of the Navier-Stokes Equations," Journal of Computational Physics, Vol. 2, No. 12, 1967.

23. Chorin, A.J., The Numerical Solution of the Navier-Stokes Equations for an Incompressible Fluid. AEC Research and Development Report NYO-1480-82, November 1967.

24. Sutherland, I.E., "A Man-Machine Graphical Communication System," AFIPS Conference Proceedings, Vol. 23, 1963. pp. 329-346.

25. English, W.K., Engelbart, D.C., and Berman, M.L., "Display Selection Techniques for Text Manipulation," IEEE Transactions on Human Factors in Electronics. HFE-8 No. 1, March 1967, pp. 5-15.

26. Ruyle, A., Brackett, J.W., and Kaplow, R., "The Status of Systems for On-Line Mathematical Assistance," Proceedings of the 22nd National Conference of the ACM, 1967, pp. 471-476.

27. Waxman, J.S., "Automated Logic Design Techniques Applicable to Integrated Circuit Technology," AFIPS Conference Proceedings, Vol. 29, 1966, pp. 247-265.

28. Jacks, E.L., "A Laboratory for the Study of Graphical Man-Machine Communication," AFIPS Conference Proceedings, Vol. 26, Part 1, 1960, pp. 343-350.

29. Parmlee, R.P., Three-Dimensional Stress Analysis for Computer-Aided Design, Ph.D. Thesis, Dept. of Mechanical Engineering, M.I.T., 1966.

30. Fromm, J.A., and Schreiber, "System Aspects of Large Problem Computation and Display," IBM Systems Journal, Vol. 11, No. 1, 1972, pp. 41-55.

31. Computer Studies of Fluid Dynamics (Y-154), Film Produced by Los Alamos Scientific Laboratory, 1966.

32. Computer Fluid Dynamics (Y-204), Film produced by Los Alamos Scientific Laboratory, 1969.

33. Carter, T.R., "Laminar Fluid Flow from a Reservoir Up To and Through a Tube-Entry Region," Ph.D. dissertation, Department of Chemical Engineering, University of Utah, 1969.

34. Bennion, S.T., "A Method of Solution for Hydrodynamics and Radiation Diffusion as a Multi-Material Problem in One Dimension," Ph.D. dissertation. Department of Computer Science, University of Utah, 1971.

35. Dijkstra, E.W., Notes on Structured Programming, August 1969.

APPENDIX I

GRAPHICAL INPUT PROGRAM

Routine Name	Function
MAIN	Main program
MAIN1	Displays input selection
INPUT	Controls input
INPUT1	Displays input functions
MESH	Controls mesh input
MESH1	Accepts mesh input
MESH2	Displays mesh input
FIGURE	Controls figure input
FIG1	Displays input functions
FIG2	Displays mesh
FIG3	Accept figure input
FIG4	Displays figure
TTYIN	Does teletype input
FLUID	Controls fluid input
FLU1	Accepts fluid input
FLU2	Displays fluid input
CONVG	Controls parameter input
CONVG1	Accepts parameter input
CONVG2	Displays parameter input
CNTRL	Controls parameter input
CNTRL1	Accepts parameter input
CNTRL2	Displays parameter input
CHECK	Displays completed description
OUTP	Outputs completed description
ARO	Draws arrow heads
BOXES	Draws light buttons
BOX	Draws one box

```
      COMMON MESS(7)
      COMMON KMAX,LMAX,MU,RHOA,DR,DZ
      COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
      COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP,FNAME
      INTEGER FNAME
      REAL MU
      CALL SETDIS
      CALL MAIN1
      CALL WMOUSE
      INY=(INY+128)/128
      GO TO (5,2,1)INY
C
C     JUMP TABLE FOR INPUT TYPE
C     1-TTY
C     2-DISK
C     3-TERMINAL
C
1     NDEV1=5
      CALL READIT
      CALL EXIT
2     NDEV1=21
      TYPE 3
3     FORMAT(1H ,'INPUT FILENAME:',$)
      ACCEPT 4,FNAME
4     FORMAT(A5)
      FNAME=FNAME+32
      CALL IFILE(21,FNAME)
      CALL READIT
      CALL EXIT
5     NDEV1=0
      CALL INPUT
      CALL EXIT
      END
```

```
SUBROUTINE MAIN1
CALL BOXES(3)
CALL MOVETO(752,24)
CALL WRITE('TERMINAL<>')
CALL MOVETO(768,152)
CALL WRITE('DISK<>')
CALL MOVETO(768,280)
CALL WRITE('TTY<>')
CALL MOVETO(728,350)
CALL WRITE('INPUT FROM:<>')
CALL SEND
RETURN
END
```

```
      SUBROUTINE INPUT
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
1     CALL INPUT1
      CALL WMOUSE
      INY=(INY+128)/128
      GO TO (2,3,4,5,6,7,8)INY
C
C     JUMP TABLE FOR INPUT MODULES
C
2     CALL CHECK
      CALL WMOUSE
      CALL OUTP
      CALL EXIT
3     CALL MOD
      GO TO 1
4     CALL CNTRL
      GO TO 1
5     CALL CONVG
      GO TO 1
6     CALL FLUID
      GO TO 1
7     CALL FIGURE
      GO TO 1
8     CALL MESH
      GO TO 1
      END
```

```
SUBROUTINE INPUT1
CALL BOXES(7)
CALL MOVETO(768,24)
CALL WRITE('EXIT<>')
CALL MOVETO(750,152)
CALL WRITE('MODIFY<>')
CALL MOVETO(740,280)
CALL WRITE('CONTROL<>')
CALL MOVETO(732,422)
CALL WRITE('CONVERGENCE<>')
CALL MOVETO(748,390)
CALL WRITE('CRITERIA<>')
CALL MOVETO(760,536)
CALL WRITE('FLUID<>')
CALL MOVETO(746,664)
CALL WRITE('BOUNDARY<>')
CALL MOVETO(768,792)
CALL WRITE('MESH<>')
CALL SEND
RETURN
END
```

```
      SUBROUTINE MESH
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      DO 1 I=1,5
1     CALL MESH1(I)
2     CALL WMOUSE
      INY=(INY+128)/128
      IF(INY.GE.6) GO TO 3
      CALL MESH1(INY)
      GO TO 2
3     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MESH1(J)
      COMMON MESS(7)
      COMMON KMAX,LMAX,MU,RHOA,DR,DZ
      COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
      COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP
      GO TO (4,6,8,13,15)J
4     TYPE 5
5     FORMAT(1H ,'DR=',$)
      ACCEPT 3,DR
2     FORMAT(A1)
3     FORMAT(F)
      GO TO 18
6     TYPE 7
7     FORMAT(1H ,'DZ=',$)
      ACCEPT 3,DZ
      GO TO 18
8     TYPE 9
9     FORMAT(1H ,'RECT OR CYL COORD?',$)
      ACCEPT 2,NSYS
      IF(NSYS.EQ.'R')NEXP=0
      IF(NSYS.EQ.'C')NEXP=1
      GO TO 18
13    TYPE 14
14    FORMAT(1H ,'NO. OF CELLS IN R=',$)
      ACCEPT 12,KMAX
      GO TO 18
15    TYPE 16
16    FORMAT(1H ,'NO. OF CELLS IN Z=',$)
      ACCEPT 12,LMAX
      NYSP=800/(LMAX-2)
      NXSP=NYSP*DR/DZ
12    FORMAT(I)
18    CALL MESH2
      RETURN
      END
```

```
SUBROUTINE MESH2
COMMON MESS(7)
COMMON KMAX,LMAX,MU,RHOA,DR,DZ
COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
COMMON/MOUSE/INX,INY,INSW,SWSTAT
COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP
CALL BOXES(6)
CALL MOVETO(744,24)
CALL WRITE('DR=Δ.ΔΔΔΔ<>',DR)
CALL MOVETO(744,152)
CALL WRITE('DZ=Δ.ΔΔΔΔ<>',DZ)
CALL MOVETO(746,280)
IF(NEXP.EQ.0)CALL WRITE('RECT COORD<>')
IF(NEXP.EQ.1)CALL WRITE('CYL COORD<>')
CALL MOVETO(722,422)
CALL WRITE('NO. OF CELLS<>')
CALL MOVETO(734,390)
CALL WRITE('IN R=ΔΔΔ<>',KMAX)
CALL MOVETO(722,550)
CALL WRITE('NO. OF CELLS<>')
CALL MOVETO(734,518)
CALL WRITE('IN Z =ΔΔΔ<>',LMAX)
CALL MOVETO(768,664)
CALL WRITE('RETURN<>')
CALL SEND
RETURN
END
```

```
      SUBROUTINE FIGURE
      COMMON MESS(7)
      COMMON KMAX,LMAX,MU,RHOA,DR,DZ
      COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
      COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP
      CALL FIG1
10    CALL WMOUSE
      INY=(INY+128)/128
      GO TO (1,2,7)INY
1     RETURN
2     TYPE 3
3     FORMAT(1H ,'INFLOW VELOCITIES:')
      TYPE 4
4     FORMAT(1H ,'V-IN=',$)
      ACCEPT 5,VIN
5     FORMAT(F)
      TYPE 6
6     FORMAT(1H ,'U-IN=',$)
      ACCEPT 5,UIN
      GO TO 10
7     CALL FIG2
      CALL FIG3
      GO TO 10
      RETURN
      END
```

```
SUBROUTINE FIG1
CALL BOXES(3)
CALL MOVETO(768,24)
CALL WRITE('RETURN<>')
CALL MOVETO(740,152)
CALL WRITE('INFLOW<>')
CALL MOVETO(732,280)
CALL WRITE('BOUNDARIES<>')
CALL SEND
RETURN
END
```

```
      SUBROUTINE FIG2
      COMMON MESS(7)
      COMMON KMAX,LMAX,MU,RHOA,DR,DZ
      COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
      COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP
      KMAXZ=KMAX-2
      LMAXZ=LMAX-2
      MAXY=LMAXZ*NYSP+128
      MAXX=KMAXZ*NXSP+128
      DO 10 I=1,KMAX-1
      NXLINE=128+(I-1)*NXSP
      CALL MOVETO(NXLINE,128)
10    CALL VECTO(NXLINE,MAXY)
      DO 20 J=1,LMAX-1
      NYLINE=128+(J-1)*NYSP
      CALL MOVETO(128,NYLINE)
20    CALL VECTO(MAXX,NYLINE)
      CALL SEND
      CALL ARO(64,192,'U')
      CALL VEC(0,-128)
      CALL VEC(128,0)
      CALL ARO(207,49,'L')
      CALL MOVETO(150,60)
      CALL WRITE('R<>')
      CALL MOVETO(10,200)
      CALL WRITE('Z<>')
      CALL MOVETO(256,52)
      CALL WRITE('MESH IS ΔΔΔ BY ΔΔΔ CELLS<>',KMAX,LMAX)
      CALL MOVETO(325,25)
      CALL WRITE('DR=.ΔΔΔΔ, DZ=.ΔΔΔΔ<>',DR,DZ)
      CALL APND
      CALL ARO(128,128,'U')
      RETURN
      END
```

```
      SUBROUTINE FIG3
      COMMON MESS(7)
      COMMON KMAX,LMAX,MU,RHOA,DR,DZ
      COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
      COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP
      LOGICAL IMF
      LOGICAL TERM
      TERM=.TRUE.
      TYPE 101
101   FORMAT(1H ,'TTY OR MOUSE INPUT?',$)
      ACCEPT 102,IDEV
102   FORMAT(A1)
      IF(IDEV.NE.'M')TERM=.FALSE.
      IMF=.TRUE.
      SMX=(KMAX-2)*DR
      SMY=(LMAX-2)*DZ
      SX=DR/NXSP
      SY=DZ/NYSP
      MM=1
1     IF(TERM)CALL WMOUSE
      IF(.NOT.TERM)CALL TTYIN
      BPX(MM)=(INX-128)*SX
      BPY(MM)=(INY-128)*SY
      IF(BPX(MM).LT.DR)BPX(MM)=0.
      IF(BPY(MM).LT.DZ)BPY(MM)=0.
      IF(BPX(MM).GT.SMX)BPX(MM)=SMX
      IF(BPX(MM).LT.DR.AND.BPY(MM).LT.DZ)GO TO 6
      IF(BPY(MM).LT.DR)GO TO 2
      IF(BPY(MM).GT.SMY)GO TO 3
      IF(IMF)GO TO 4
      CALL ARO(INX,INY,'L')
      GO TO 5
2     CALL ARO(INX,INY,'U')
      GO TO 5
3     CALL ARO(INX,INY,'D')
      IMF=.FALSE.
      ISMX1=MM
      GO TO 5
4     CALL ARO(INX,INY,'R')
5     MM=MM+1
      GO TO 1
6     ISMX2=MM
      CALL FIG4
      RETURN
      END
```

```
      SUBROUTINE FIG4
      COMMON MESS(7)
      COMMON KMAX,LMAX,MU,RHOA,DR,DZ
      COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
      COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP
      CALL MOVETO(128,128)
      DO 1 I=1,ISMX2
      IX=(BPX(I)*NXSP)/DR+128
      IY=(BPY(I)*NYSP)/DZ+128
      CALL VECTO(IX,IY)
      CALL MOVETO(732,350)
      CALL WRITE('RETURN<>')
      CALL SEND
      RETURN
      END
```

```
SUBROUTINE TTYIN
COMMON MESS(7)
COMMON KMAX,LMAX,MU,RHOA,DR,DZ
COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
COMMON/MOUSE/INX,INY,INSW,SWSTAT
COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP
TYPE 1
FORMAT(1H ,'*')
ACCEPT 2,AINX,AINY
FORMAT(2F)
INX=128+(AINX*NXSP)/DR
INY=128+(AINY*NYSP)/DZ
RETURN
END
```

```
SUBROUTINE FLUID
COMMON/MOUSE/INX,INY,INSW,SWSTAT
DO 1 I=1,5
CALL FLU1(1)
CALL WMOUSE
INY=(INY+128)/128
IF(INY.GE.6)RETURN
CALL FLU1(INY)
GO TO 2
END
```

```
      SUBROUTINE FLU1(J)
      COMMON MESS(7)
      COMMON KMAX,LMAX,MU,RHOA,DR,DZ
      COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
      COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP
      REAL MU
      GO TO (1,4,6,9,11)J
1     TYPE 2
2     FORMAT(1H ,'INITIAL U VELOCITY=',$)
      ACCEPT 3,UNOT
3     FORMAT(F)
      GO TO 15
4     TYPE 5
5     FORMAT(1H ,'INITIAL V VELOCITY=',$)
      ACCEPT 3,VNOT
      GO TO 15
6     TYPE 7
7     FORMAT(1H ,'PARTICLE DENSITY=',$)
      ACCEPT 8,NP
8     FORMAT(I)
      GO TO 15
9     TYPE 10
10    FORMAT(1H ,'RHOA=',$)
      ACCEPT 3,RHOA
      GO TO 15
11    TYPE 12
12    FORMAT(1H ,'MU=',$)
      ACCEPT 3,MU
      GO TO 15
15    CALL FLU2
      RETURN
      END
```

```
SUBROUTINE FLU2
COMMON MESS(7)
COMMON KMAX,LMAX,MU,RHOA,DR,DZ
COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
COMMON/MOUSE/INX,INY,INSW,SWSTAT
COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP
REAL MU
CALL BOXES(6)
CALL MOVETO(732,24)
CALL WRITE('U=ΔΔΔ.ΔΔΔ<>',UNOT)
CALL MOVETO(740,152)
CALL WRITE('V=ΔΔΔ.ΔΔΔ<>',VNOT)
CALL MOVETO(704,279)
CALL WRITE('PARTICLE DENSITY<>')
CALL MOVETO(770,258)
CALL WRITE('=ΔΔ<>',NP)
CALL MOVETO(732,410)
CALL WRITE('RHOA=Δ.ΔΔΔΔΔΔΔ<>',RHOA)
CALL MOVETO(732,536)
CALL WRITE('MU=Δ.ΔΔΔΔΔΔΔ<>',MU)
CALL MOVETO(732,664)
CALL WRITE('RETURN<>')
CALL SEND
RETURN
END
```

```
      SUBROUTINE CONVG
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      DO 1 I=1,3
1     CALL CONVG1(I)
2     CALL WMOUSE
      INY=(INY+128)/128
      IF(INY.GE.4)RETURN
      CALL CONVG1(INY)
      GO TO 2
      END
```

```
      SUBROUTINE CONVG1(J)
      COMMON MESS(7)
      COMMON KMAX,LMAX,MU,RHOA,DR,DZ
      COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
      COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP
      GO TO (1,4,6)J
1     TYPE 2
2     FORMAT(1H ,'DT=',$)
      ACCEPT 3,DT
3     FORMAT(F)
      GO TO 8
4     TYPE 5
5     FORMAT(1H ,'CONVERGENCE EPS=',$)
      ACCEPT 3,EPS2
      GO TO 8
6     TYPE 7
7     FORMAT(1H ,'RELAXATION FACTOR=',$)
      ACCEPT 3,BETA
8     CALL CONVG2
      RETURN
      END
```

```
      SUBROUTINE CONVG2
      COMMON MESS(7)
      COMMON KMAX,LMAX,MU,RHOA,DR,DZ
      COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
      COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP
      CALL BOXES(4)
      CALL MOVETO(742,24)
      CALL WRITE('DT=Δ.ΔΔΔΔΔ<>',DT)
      CALL MOVETO(742,152)
      CALL WRITE('EPS2=Δ.ΔΔΔΔΔ<>',EPS2)
      CALL MOVETO(752,280)
      CALL WRITE('BETA=Δ.ΔΔΔ<>',BETA)
      CALL MOVETO(760,408)
      CALL WRITE('RETURN<>')
      CALL SEND
      RETURN
      END
```

```
      SUBROUTINE CNTRL
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      DO 1 I=1,6
1     CALL CNTRL1(I)
2     CALL WMOUSE
      INY=(INY+128)/128
      IF(INY.GE.7)RETURN
      CALL CNTRL1(INY)
      GO TO 2
      RETURN
      END
```

```
      SUBROUTINE CNTRL1(J)
      COMMON MESS(7)
      COMMON KMAX,LMAX,MU,RHOA,DR,DZ
      COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
      COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP,FNAME
      INTEGER FNAME
      GO TO (1,4,6,8,10,13)J
1     TYPE 2
2     FORMAT(1H ,'NO. OF CYCLES BETWEEN DUMPS=',$)
      ACCEPT 3,NDUMP
      GO TO 16
3     FORMAT(I)
4     TYPE 5
5     FORMAT(1H ,'NO. OF PRESSURE ITERATIONS=',$)
      ACCEPT 3,NPIT
      GO TO 16
6     TYPE 7
7     FORMAT(1H ,'STOP AT CYCLE-',$)
      ACCEPT 3,NSTOP
      GO TO 16
8     TYPE 9
9     FORMAT(1H ,'NO. OF CYCLES BETWEEN EDITS=',$)
      ACCEPT 3,NEDIT
      GO TO 16
10    TYPE 11
11    FORMAT(1H ,'OUTPUT FILENAME:',$)
      ACCEPT 12,FNAME
12    FORMAT(A5)
      GO TO 16
13    TYPE 14
14    FORMAT(1H ,'DESCRIPTION:',$)
      ACCEPT 15,(MESS(I),I=1,7)
15    FORMAT(7A5)
16    CALL CNTRL2
      RETURN
      END
```

```
SUBROUTINE CNTRL2
COMMON MESS(7)
COMMON KMAX,LMAX,MU,RHOA,DR,DZ
COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
COMMON/MOUSE/INX,INY,INSW,SWSTAT
COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP,FNAME
INTEGER FNAME
CALL BOXES(7)
CALL MOVETO(752,24)
CALL WRITE('NDUMP=ΔΔ<>',NDUMP)
CALL MOVETO(752,152)
CALL WRITE('NPIT=ΔΔ<>',NPIT)
CALL MOVETO(752,280)
CALL WRITE('NSTOP=ΔΔΔ<>',NSTOP)
CALL MOVETO(752,408)
CALL WRITE('NEDIT=ΔΔ<>',NEDIT)
CALL MOVETO(752,536)
CALL WRITE('FNAME=<>')
CALL WRITES(FNAME)
CALL MOVETO(752,664)
CALL WRITE('DESCRIPTION<>')
CALL MOVETO(760,792)
CALL WRITE('RETURN<>')
CALL SEND
RETURN
END
```

```
      SUBROUTINE CHECK
      COMMON MESS(7)
      COMMON KMAX,LMAX,MU,RHOA,DR,DZ
      COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
      COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP
      IY1=512-BPX(1)*NXSP/DR
      CALL MOVETO(150,IY1)
      DO 1 J=2,ISMX1-1
      IX=150+BPY(J)*NYSP/DZ
      IY=512-BPX(J)*NXSP/DR
1     CALL ZIPTO(IX,IY)
      IY1=512+BPX(1)*NXSP/DR
      CALL MOVETO(150,IY1)
      DO 2 J=2,ISMX1-1
      IX=150+BPY(J)*NYSP/DZ
      IY=512+BPX(J)*NXSP/DR
2     CALL ZIPTO(IX,IY)
      ISM=ISMX1+1
      IX1=150+BPY(ISM)*NYSP/DZ
      CALL MOVETO(IX1,512)
      DO 3 J=ISM+1,ISMX2-1
      IX=150+BPY(J)*NYSP/DZ
      IY=512+BPX(J)*NXSP/DR
3     CALL ZIPTO(IX,IY)
      CALL MOVETO(IX1,512)
      DO 4 J=ISM+1,ISMX2-1
      IX=150+BPY(J)*NYSP/DZ
      IY=512-BPX(J)*NYSP/DR
4     CALL ZIPTO(IX,IY)
      IYB=512-BPX(1)*NXSP/DR
      IYT=512+BPX(1)*NXSP/DZ
      CALL SEND
      CALL ARO(140,IYT,'U')
      CALL VECTO(140,525)
      CALL ARO(140,IYB,'D')
      CALL VECTO(140,499)
      CALL MOVETO(52,504)
      CM=BPX(1)*2.
      CALL WRITE('ΔΔ.ΔΔ CM<>',CM)
      CALL ARO(150,IYB,'D')
      CALL VECTO(150,512)
      LNG=512-IYB
      IXT=150+LNG
      CALL ARO(IXT,512,'L')
      CALL VECTO(150,512)
      CALL MOVETO(160,IYB+5)
      CALL WRITE('R<>')
      CALL MOVETO(IXT+12,504)
      CALL WRITE('Z<>')
      ISM1=ISMX1-1
      IYB=512-BPX(ISM1)*NXSP/DR
      IYT=512+BPX(ISM1)*NXSP/DR
      CALL ARO(985,IYT,'U')
      CALL VECTO(985,532)
      CALL ARO(985,IYB,'D')
```

```
      CALL VECTO(985,492)
      CALL MOVETO(955,522)
      CM=BPX(ISM1)*2.
      CALL WRITE('ΔΔ.ΔΔ<>',CM)
      CALL MOVETO(973,503)
      CALL WRITE('CM<>')
      IY1=IYT+20
      CALL ARO(950,IY1,'L')
      CALL VECTO(600,IY1)
      CALL ARO(150,IY1,'R')
      CALL VECTO(500,IY1)
      CM=BPY(ISMX1)*2.
      IY1=IY1-8
      CALL MOVETO(504,IY1)
      CALL WRITE('ΔΔ.ΔΔ CM<>',CM)
      IX1=BPY(ISM)*NXSP/DZ+300
      CALL MOVETO(IX1,517)
      CALL VEC(-60,0)
      CALL VEC(0,10)
      CALL VEC(-20,-15)
      CALL VEC(20,-15)
      CALL VEC(0,10)
      CALL VEC(60,0)
      CALL MOVE(5,-3)
      CALL WRITE('FLOW<>')
      CALL MOVETO(348,124)
      CALL WRITE('U-IN=ΔΔΔΔ.ΔΔ<>',UIN)
      CALL MOVETO(348,164)
      CALL WRITE('V-IN=ΔΔΔΔ.ΔΔ<>',VIN)
      CALL MOVETO(515,124)
      CALL WRITE('U-NOT=ΔΔΔΔ.ΔΔ<>',UNOT)
      CALL MOVETO(515,164)
      CALL WRITE('V-NOT=ΔΔΔΔ.ΔΔ<>',VNOT)
      CALL MOVETO(412,900)
      DO 5 J=1,7
5     CALL WRITES(MESS(J))
      CALL APND
      RETURN
      END
```

```
      SUBROUTINE OUTP
      COMMON MESS(7)
      COMMON KMAX,LMAX,MU,RHOA,DR,DZ
      COMMON NP,BPX(50),BPY(50),UNOT,VNOT,UIN,VIN
      COMMON DT,NEDIT,EPS2,BETA,NSTOP,NPIT,NDUMP,NEXP
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/ETC/ISMX1,ISMX2,NXSP,NYSP,FNAME
      DIMENSION IDT(2)
      REAL MU
      INTEGER FNAME
      CALL OFILE(23,FNAME)
      TYPE 1
1     FORMAT(1H ,'LINEFEED TO NEW PAGE,CR')
      ACCEPT 2,DUMMY
2     FORMAT(A1)
      CALL DATE(IDT)
      TYPE 3
3     FORMAT(1H ,T37,'ABMAC')
      TYPE 4
4     FORMAT(1H ,T35,'UNIVERSITY OF UTAH')
      TYPE 5,IDT(1),IDT(2)
5     FORMAT(1H ,T35,A5,A4)
      TYPE 6
6     FORMAT(1H-)
      TYPE 7,(MESS(I),I=1,7)
7     FORMAT(1H ,7A5)
      WRITE(23,7)(MESS(I),I=1,7)
      MOPT=0
      TYPE 8,KMAX,LMAX,NEXP,MOPT
8     FORMAT(1H ,4I)
      WRITE(23,8)KMAX,LMAX,NEXP,MOPT
      SMIN=.01
      EPS1=.2
      TYPE 9,SMIN,EPS1
9     FORMAT(1H ,2F)
      WRITE(23,9)SMIN,EPS1
      GZ=0.
      GR=0.
      TYPE 10,GZ,GR,MU,RHOA
10    FORMAT(1H ,4F)
      WRITE(23,10)GZ,GR,MU,RHOA
      TYPE 11,DR
11    FORMAT(1H ,F)
      WRITE(23,11)DR
      TYPE 11,DZ
      WRITE(23,11)DZ
      KA=2
      KB=KMAX-1
      LA=2
      LB=LMAX-1
      TYPE 12,NP,KA,KB,LA,LB
12    FORMAT(1H ,5I)
      WRITE(23,12)NP,KA,KB,LA,LB
      IZERO=0
      ZERO=0.
      SMALL=-0.0001
      ISM=ISMX1-1
```

```
      TYPE 13,ZERO,ZERO
      WRITE(23,13)ZERO,ZERO
13    FORMAT(1H ,2F)
      DO 14 J=1,ISMX2
      TYPE 13,BPX(J),BPY(J)
14    WRITE(23,13)BPX(J),BPY(J)
      TYPE 15,IZERO
15    FORMAT(1H ,I)
      WRITE(23,15)IZERO
      KA=1
      KB=KMAX
      LA=1
      LB=LMAX
      KIN=0
      KOUT=0
      TYPE 16,KA,KB,LA,LB,UNOT,VNOT,KIN,KOUT
16    FORMAT(1H ,4I,2F,2I)
      WRITE(23,16)KA,KB,LA,LB,UNOT,VNOT,KIN,KOUT
      KA=2
      KB=BPX(1)/DR+2
      LA=1
      LB=1
      KOUT=1
      TYPE 16,KA,KB,LA,LB,UNOT,VNOT,KIN,KOUT
      WRITE(23,16)KA,KB,LA,LB,UNOT,VNOT,KIN,KOUT
      KA=2
      KB=KMAX-1
      LA=LMAX
      LB=LMAX
      VVEL=VIN
      UVEL=UIN
      KOUT=0
      KIN=1
      TYPE 16,KA,KB,LA,LB,UVEL,VVEL,KIN,KOUT
      WRITE(23,16)KA,KB,LA,LB,UVEL,VVEL,KIN,KOUT
      TYPE 15,IZERO
      WRITE(23,15)IZERO
      NBP=2*(KMAX-2)
      XXA=.05
      YYA=DZ*(LMAX-2)+.05
      DDR=DR/2
      DDZ=0.
      TYPE 17,NBP,XXA,YYA,DDR,DDZ
17    FORMAT(1H ,I,4F)
      WRITE(23,17)NBP,XXA,YYA,DDR,DDZ
      TYPE 9,BPX(1),SMALL
      WRITE(23,9)BPX(1),SMALL
      DO 18 I=2,ISMX1-2
      TYPE 9,BPX(I),BPY(I)
18    WRITE(23,9)BPX(I),BPY(I)
      ITWO=2
      TYPE 19,BPX(ISM),BPY(ISM),ITWO
19    FORMAT(1H ,2F,I)
      WRITE(23,19)BPX(ISM),BPY(ISM),ITWO
      IONE=1
      ISM2=ISMX1+2
      ISM=ISMX2-1
```

```
      BPX(ISM2)=SMALL
      DO 20 I=ISM2,ISMX2-2
      TYPE 9,BPX(I),BPY(I)
20    WRITE(23,9)BPX(I),BPY(I)
      TYPE 19,BPX(ISM),BPY(ISM),IONE
      WRITE(23,19)BPX(ISM),BPY(ISM),IONE
      ICRT=1
      CRTRA=3.2
      CRTRB=6.
      CRTZA=16.
      CRTZB=.2885
      XMAX=.2885
      YMAX=.9
      TYPE 21,ICRT,CRTRA,CRTRB,CRTZA,CRTZB,XMAX,YMAX
21    FORMAT(1H ,I,6F)
      WRITE(23,21)ICRT,CRTRA,CRTRB,CRTZA,CRTZB,XMAX,YMAX
      TYPE 11,DT
      WRITE(23,11)DT
      TYPE 15,NEDIT
      WRITE(23,15)NEDIT
      TYPE 11,EPS1
      WRITE(23,11)EPS1
      TYPE 11,EPS2
      WRITE(23,11)EPS2
      TYPE 11,BETA
      WRITE(23,11)BETA
      NGEN='N'
      TYPE 22,NGEN
22    FORMAT(1H ,A1)
      WRITE(23,22)NGEN
      TYPE 11,SMIN
      WRITE(23,11)SMIN
      TYPE 15,NSTOP
      WRITE(23,15)NSTOP
      TYPE 15,NPIT
      WRITE(23,15)NPIT
      TYPE 15,NDUMP
      WRITE(23,15)NDUMP
      END FILE 23
      RETURN
      END
```

```
SUBROUTINE ARO(INX,INY,DIR)
INTEGER DIR
N1=5
N2=15
IF(DIR.EQ.'U')INY=INY-15
IF(DIR.EQ.'D')INY=INY+15
IF(DIR.EQ.'L')INX=INX-15
IF(DIR.EQ.'R')INX=INX+15
CALL MOVETO(INX,INY)
IF(DIR.EQ.'R'.OR.DIR.EQ.'U')N2=-N2
IF(DIR.EQ.'U'.OR.DIR.EQ.'D')GO TO 1
CALL VEC(0,N1)
CALL VEC(N2,-N1)
CALL VEC(-N2,-N1)
CALL VEC(0,N1)
CALL APND
RETURN
CALL VEC(-N1,0)
CALL VEC(N1,-N2)
CALL VEC(N1,N2)
CALL VEC(-N1,0)
CALL APND
RETURN
END
```

```
      SUBROUTINE BOXES(N)
      DO 1 I=1,N
      JPOS=(I*128)-128
      CALL MOVETO(704,JPOS)
1     CALL BOX
      RETURN
      END
```

```
SUBROUTINE BOX
CALL VEC(0,64)
CALL VEC(192,0)
CALL VEC(0,-64)
CALL VEC(-192,0)
RETURN
END
```

APPENDIX II

THE COMPUTATION INITIALIZATION PROGRAM

Routine Name	Function
MAIN	Establish common areas
MACMAIN	Control initialization, writes out data file
GEN	Marks cells and initializes
GENB	Computes normals describing boundary segments
REFLAG	Readjusts cell flags in surface
EDIT	Lists initial values for all problem variables

```
      COMMON NVAC,NGEN,UAC,VAC,TM1,TM2,TM3
      COMMON NCYCLE,NSTOP,NEDTM,NEDIT,OPE
      COMMON TIME,EDTIM,EDDT,NPIT
      COMMON BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
      COMMON SMIN,EPS1,NBMAX,UBTO,VBTO,UBND,VBND
      COMMON UM,MU,RHOA,GR,GZ,G,ISEC
      COMMON VIN,UIN,DKP,DLP,NBP,DDR,DDZ
      COMMON NEXP,NEXPP,NPART,NDIV
      COMMON NOR1D,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      COMMON KDMN,LDMN,KDMX,LDMX,MESS(7)
      COMMON ICRT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
      COMMON ITLST,IDZ,IZ,GAMA,PGAS,PGASZ,PATB,GASV,GASVZ
      COMMON NF,NG,NIT,H,XG,HH,HV,GG,FUDGE,HTAL,CRAX,DELZ
      COMMON BPX(300),PTX(300),RMP(300),RNP(300),XB(300)
      COMMON BPY(300),PTY(300),ZMP(300),ZNP(300),YB(300)
      COMMON DS(300),IBP(300)
      COMMON R(150),Z(150),RB(150),ZB(150)
      COMMON DRB(150),DZB(150)
      COMMON M(2500)
      COMMON RP1(5000),RP2(5000)
      COMMON UAVG(2500),VAVG(2500)
      COMMON P(2500),U(2500),ETA(2500),V(2500),PSI(2500)
      COMMON DNORX(2500),DNORY(2500),PNORX(2500),PNORY(2500)
      COMMON NPRTA(2500),NPRTB(2500),MPRTA(2500),MPRTB(2500)
      COMMON SNC(2500),DR,DZ
      COMMON NBC(300),IFX(300)
      COMMON NDEV1,NDEV2
      COMMON/FLAG1/FLAG1
      REAL MU
      INTEGER AND,OR,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB
      DATA INTRF,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB/1,2,4,8,16,32,
     *  64,128,256,512,1024,2048,4096,8192,16384,32768/
      DATA MASK1,MASK2,MASK3,MASK4,MASK5,MASK6,
     *  MASK7,MASK8/077777777757,077777773777,
     *  077777677777,077777777773,077777777767,
     *  077777737777,077777757777,077777767777/
      DOUBLE PRECISION U,V,PSI,ETA,P
      DOUBLE PRECISION UMAX,VMAX
      PI=3.1415926
      INTEGER FLAG1
      FLAG1=1
      CALL MACMAIN
      END
      SUBROUTINE MACMAIN
      COMMON NVAC,NGEN,UAC,VAC,TM1,TM2,TM3
      COMMON NCYCLE,NSTOP,NEDTM,NEDIT,OPE
      COMMON TIME,EDTIM,EDDT,NPIT
      COMMON BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
      COMMON SMIN,EPS1,NBMAX,UBTO,VBTO,UBND,VBND
      COMMON UM,MU,RHOA,GR,GZ,G,ISEC
      COMMON VIN,UIN,DKP,DLP,NBP,DDR,DDZ
      COMMON NEXP,NEXPP,NPART,NDIV
      COMMON NOR1D,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      COMMON KDMN,LDMN,KDMX,LDMX,MESS(7)
      COMMON ICRT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
```

```
      COMMON ITLST,IOZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
      COMMON NF,NG,NIT,H,XG,HH,HV,GG,FUDGE,NTAL,CRAX,DELZ
      COMMON BPX(300),PTX(300),RMP(300),RNP(300),XB(300)
      COMMON BPY(300),PTY(300),ZMP(300),ZNP(300),YB(300)
      COMMON DS(300),IBP(300)
      COMMON R(150),Z(150),RB(150),ZB(150)
      COMMON DRB(150),DZB(150)
      COMMON M(2500)
      COMMON RP1(5000),RP2(5000)
      COMMON UAVG(2500),VAVG(2500)
      COMMON P(2500),U(2500),ETA(2500),V(2500),PSI(2500)
      COMMON DNORX(2500),DNORY(2500),PNORX(2500),PNORY(2500)
      COMMON NPRTA(2500),NPRTB(2500),MPRTA(2500),MPRTB(2500)
      COMMON SNC(2500),DR,DZ
      COMMON NBC(300),IPX(300)
      COMMON NDEV1,NDEV2
      COMMON/FLAG1/FLAG1
      REAL MU
      INTEGER AND,OR,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     * FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB
      DATA INTRF,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     * FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB/1,2,4,8,16,32,
     * 64,128,256,512,1024,2048,4096,8192,16384,32768/
      DATA MASK1,MASK2,MASK3,MASK4,MASK5,MASK6,
     * MASK7,MASK8/077777777757,077777773777,
     * 077777677777,077777777773,077777777767,
     * 077777737777,077777757777,077777767777/
      DOUBLE PRECISION U,V,PSI,ETA,P
      DOUBLE PRECISION UR,UL,VB,VT,UBAR,VBAR
      DOUBLE PRECISION UTOP,VLEFT,VRGHT
      DOUBLE PRECISION TEMPL,TEMPR,UBOT
      DOUBLE PRECISION UMAX,VMAX
      PI=3.1415926
      NTAL=0
      TYPE 2
2     FORMAT(1H ,'NEW GENERATOR?',$)
      ACCEPT 4,NIXE
3     FORMAT(I)
4     FORMAT(A1)
      IF(NIXE.EQ.'N') GO TO 7
      TYPE 5
5     FORMAT(1H ,'INPUT FROM:',$)
      ACCEPT 6,NDEV1
6     FORMAT(A3)
      IF(NDEV1.EQ.'TTY') NDEV1=5
      IF(NDEV1.EQ.'DSK') NDEV1=21
      IF(NDEV1.EQ.21)GO TO 601
      GO TO 605
601   TYPE 602
602   FORMAT(1H ,'INPUT FILE:',$)
      ACCEPT 603,NME
603   FORMAT(A5)
      NME=NME+32
      CALL IFILE(21,NME)
605   CALL GEN
      NEDTM=0
      NLIMT=0
```

```
      NDMPT=1
      DPE=1.
      GO TO 11
7     TYPE 8,NCYCLE
8     FORMAT(1H ,'READY AT CYCLE ',I3)
      TYPE 9
9     FORMAT(1H ,'CHANGE RUN TIME SYSTEM?')
      ACCEPT 10,NYS
10    FORMAT(A1)
      IF(NYS.EQ.'N') GO TO 33
      NDEV1=5
11    TYPE 12
12    FORMAT(1H ,'DT=',$)
      READ(NDEV1,13) DT
      TYPE 24
13    FORMAT(F)
      TYPE 14
14    FORMAT(1H ,'NO. OF CYCLES BETWEEN EDITS=',$)
      READ(NDEV1,15)NEDIT
      TYPE 24
15    FORMAT(I)
      TYPE 16
16    FORMAT(1H ,'BOUNDARY SENSING PARAMETER=',$)
      READ(NDEV1,13) EPS1
      TYPE 24
      TYPE 17
17    FORMAT(1H ,'CONVERGENCE EPSILON=',$)
      READ(NDEV1,13) EPS2
      TYPE 24
      TYPE 18
18    FORMAT(1H ,'OVER-RELAXATION FACTOR=',$)
      READ(NDEV1,13) BETA
      TYPE 24
      TYPE 19
19    FORMAT(1H ,'BACKWARDS DIFFERENCING?')
      READ(NDEV1,20) NGEN
      TYPE 24
      IF(NGEN.EQ.'Y') NGEN=0
      IF(NGEN.EQ.'N') NGEN=1
20    FORMAT(A1)
      TYPE 21
21    FORMAT(1H ,'FULL CELL EPSILON=',$)
      READ(NDEV1,13) SMIN
      TYPE 24
      TYPE 22
22    FORMAT(1H ,'STOP AT CYCLE=',$)
      READ(NDEV1,15) NSTOP
      TYPE 24
23    CONTINUE
24    FORMAT(1H )
      TYPE 25
25    FORMAT(1H ,'PRESSURE ITERATIONS=',$)
      READ(NDEV1,15) NPIT
      TYPE 24
      TYPE 26
26    FORMAT(1H ,'NO. OF CYCLES BETWEEN DUMPS=',$)
      READ(NDEV1,15) NDUMP
```

```
      TYPE 24
      KMAXZ=KMAX-1
      LMAXZ=LMAX-1
      KMAXZZ=KMAX-2
      LMAXZZ=LMAX-2
      NGRID=KMAX*LMAX
      DTMAX=DR**2*DZ**2/(DR**2+DZ**2)
      DTMAX=.4*DTMAX/(MU+.000001)*RHOA
      DT=.5*(DT+DTMAX)-.5*ABS(DT-DTMAX)
      DTMAX=DT
      RELAX=.2*RHOA*DR**2/DT
      NSQR=2
      TEV=.25
      TAU=2.0
      EV=.5
      IF(G)26,28,29
26    CRAX=RHOA
      GO TO 30
29    CRAX=G*Z(LMAX)
      CRAX=1./CRAX
30    GG=G
      IF(NIXE.EQ.'N') GO TO 33
      CALL EDIT
32    CONTINUE
33    CONTINUE
      NME=NME+2
      CALL OFILE(22,NME)
      WRITE(22)NGEN
       WRITE(22)NCYCLE,NSTOP,NEDTM,NEDIT,OPE
       WRITE(22)TIME,EDTIM,EDDT,MPIT
       WRITE(22)BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
       WRITE(22)SMIN,EPS1,NBMAX,UBTO,VBTO,UBND,VBND
       WRITE(22)UM,MU,RHOA,GR,GZ,G,ISEC
       WRITE(22)VIN,UIN,DKP,DLP,NBP,DDR,DDZ
       WRITE(22)NEXP,NEXPP,NPART,NDIV
       WRITE(22)NGRID,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
       WRITE(22)KDMN,LDMN,KDMX,LDMX,MESS
      WRITE(22)ICRT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
       WRITE(22)ITEST,IOZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
       WRITE(22)NF,NG,NIT,H,XG,HH,HV,GO,FUDGE,NTAL,CRAX,DELZ
       WRITE(22)R,Z,RB,ZB
       WRITE(22)DRB,DZB
       WRITE(22)M
       WRITE(22)P,U,ETA,V,PSI
       WRITE(22)DNORX,DNORY,PNORX,PNORY
      WRITE(22)DR,DZ
      END FILE 22
      RETURN
      END
```

```
      SUBROUTINE GEN
      COMMON NVAC,NGEN,UAC,VAC,TM1,TM2,TM3
      COMMON NCYCLE,NSTOP,NEDTM,NEDIT,OPE
      COMMON TIME,EDTIM,EDDT,NPIT
      COMMON BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
      COMMON SMIN,EPS1,NBMAX,UBTO,VBTO,UBND,VBND
      COMMON UM,MU,RHOA,GR,GZ,G,ISEC
      COMMON VIN,UIN,DKP,DLP,NBP,DDR,DDZ
      COMMON NEXP,NEXPP,NPART,NDIV
      COMMON NGRID,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      COMMON KDMN,LDMN,KDMX,LDMX,MESS(7)
      COMMON ICRT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
      COMMON ITEST,IOZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
      COMMON NF,NG,NIT,H,XG,HH,HV,GG,FUDGE,NTAL,CRAX,DELZ
      COMMON BPX(300),PTX(300),RMP(300),RNP(300),XB(300)
      COMMON BPY(300),PTY(300),ZMP(300),ZNP(300),YB(300)
      COMMON BS(300),IBP(300)
      COMMON R(150),Z(150),RB(150),ZB(150)
      COMMON DRB(150),DZB(150)
      COMMON M(2500)
      COMMON RP1(5000),RP2(5000)
      COMMON UAVG(2500),VAVG(2500)
      COMMON P(2500),U(2500),ETA(2500),V(2500),PSI(2500)
      COMMON DNORX(2500),DNORY(2500),PNORX(2500),PNORY(2500)
      COMMON NPRTA(2500),NPRTB(2500),MPRTA(2500),MPRTB(2500)
      COMMON SNC(2500),DR,DZ
      COMMON NBC(300),IPX(300)
      COMMON NDEV1,NDEV2
      COMMON/FLAG1/FLAG1
      REAL MU
      INTEGER AND,OR,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB
      DATA INTRF,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB/1,2,4,8,16,32,
     *  64,128,256,512,1024,2048,4096,8192,16384,32768/
      DATA MASK1,MASK2,MASK3,MASK4,MASK5,MASK6,
     *  MASK7,MASK8/0777777777757,0777777773777,
     *  0777777677777,0777777777773,0777777777767,
     *  0777777737777,0777777757777,0777777767777/
      DOUBLE PRECISION UMAX,VMAX
      DOUBLE PRECISION U,V,PSI,ETA,P
      PI=3.1415926
      LOGICAL TEST
      PI=3.1415926
      TYPE 1001
      TYPE 1009
1001  FORMAT(1H ,'HEADING')
      READ(NDEV1,1)(MESS(I),I=1,7)
1     FORMAT(7A5)
2     FORMAT(4I)
3     FORMAT(2F)
4     FORMAT(4F)
      TYPE 1002
      TYPE 1009
1002  FORMAT(1H ,'KMAX,LMAX,NEXP,MOPT')
      READ(NDEV1,2)KMAX,LMAX,NEXP,MOPT
      TYPE 1003
```

```
      TYPE 1009
1003  FORMAT(1H ,'SMIN,EPS1')
      READ(NDEV1,3)SMIN,EPS1
      TYPE 1004
      TYPE 1009
1004  FORMAT(1H ,'GZ,GR,MU,RHOA')
      READ(NDEV1,4)GZ,GR,MU,RHOA
      G=SQRT(GZ**2+GR**2)
      NEXPP=NEXP+1
      KMAXZ=KMAX-1
      KMAXZZ=KMAX-2
      LMAXZ=LMAX-1
      LMAXZZ=LMAX-2
      NCYCLE=0
      TIME=0.
      NDIV=0
      CUT=.0000001
      UM=MU
      NGRID=KMAX*LMAX
      TYPE 1005
C     GENERATE COORDINATES, AND CONSTANTS
C
1005  FORMAT(1H ,'DR')
      READ(NDEV1,5)DR
      TYPE 1006
      TYPE 1009
1006  FORMAT(1H ,'DZ')
      READ(NDEV1,5)DZ
5     FORMAT(F)
      RB(1)=1./10.0**10
      DO 6 K=1,KMAXZ
6     RB(K+1)=RB(K)+DR
      ZB(1)=0.
      DO 7 L=1,LMAXZ
7     ZB(L+1)=ZB(L)+DZ
      R(1)=-.5*RB(2)
      Z(1)=-.5*ZB(2)
      DO 8 K=2,KMAX
8     R(K)=(RB(K)+RB(K-1))*0.5
      DO 9 L=2,LMAX
9     Z(L)=(ZB(L)+ZB(L-1))*0.5
      DO 10 K=1,KMAXZ
10    DRB(K)=R(K+1)-R(K)
      DO 11 L=1,LMAXZ
11    DZB(L)=Z(L+1)-Z(L)
      DO 12 K=1,NGRID
      P(K)=0.
      U(K)=0.
      V(K)=0.
      ETA(K)=0.
      PSI(K)=0.
      UAVG(K)=0.
      VAVG(K)=0.
      M(K)=0
      DNORX(K)=0.
      DNORY(K)=0.
      PNORX(K)=0.
```

```
12      PNORM(K)=0.
        UBND=0.
        VBND=0.
        DO 13 L=2,LMAX2
        KMM=(L-1)*KMAX
        DO 13 K=2,KMAX2
        KM=K+KMM
13      M(KM)=(M(KM).OR.EMP)
C       READ IN A SET OF POINTS SPECIFYING A CLOSED CURVE, LYING
C       IN THE REGION BOUNDED BY KA,KB,LA,LB. PARTICLES ARE
C       GENERATED AT DENSITY NP**2 INSIDE THIS AREA. ANY NUMBER
C       OF CLOSED REGIONS ALLOWED.
C
        N=0
        TYPE 1007
        TYPE 1009
1007    FORMAT(1H ,'NP,KA,KB,LA,LB')
14      READ(NDEV1,15)NP,KA,KB,LA,LB
15      FORMAT(5I)
        IF(NP) 31,31,16
16      TYPE 1008
1008    FORMAT(1H ,'BPX,BPY')
        TYPE 1009
1009    FORMAT(1H )
        READ(NDEV1,17)BPX(1),BPY(1)
17      FORMAT(2F)
        MM=2
18      READ(NDEV1,17)BPX(MM),BPY(MM)
        IF(BPX(1).EQ.BPX(MM).AND.BPY(1).EQ.BPY(MM)) GO TO 21
19      MM=MM+1
        GO TO 18
21      MMAX=MM-1
        DKP=DR/NP
        DLP=DZ/NP
        DO 28 L=LA,LB
        KMM=(L-1)*KMAX
        DO 28 K=KA,KB
        KM=K+KMM
        RA=RB(K-1)-0.5*DKP
        ZA=ZB(L-1)-0.5*DLP
        DO 28 LP=1,NP
        ZZ=ZA+DLP*LP
        DO 28 KP=1,NP
        RR=RA+DKP*KP
        ANGLE=0.
        DO 26 J=1,MMAX
        X1=BPX(J)-RR
        X2=BPX(J+1)-RR
        Y1=BPY(J)-ZZ
        Y2=BPY(J+1)-ZZ
        CROSSP=X1*Y2-X2*Y1
        FMAG=(X1**2+Y1**2)*(X2**2+Y2**2)
        FMAG=SQRT(FMAG)
        ALPHA=ASIN(CROSSP/FMAG)
        DOT=X1*X2+Y1*Y2
        IF(DOT)22,25,25
22      IF(ALPHA) 23,24,24
```

```
23      ANGLE=ANGLE-PI-ALPHA
        GO TO 26
24      ANGLE=ANGLE+PI-ALPHA
        GO TO 26
25      ANGLE=ANGLE+ALPHA
26      CONTINUE
        IF(ABS(ANGLE)-0.0001)28,27,27
27      N=N+1
        NPRTA(KM)=NPRTA(KM)+1
        M(KM)=(M(KM).OR.FULL)
        M(KM)=(M(KM).AND.MASK1)
        RP1(N)=RR
        RP2(N)=ZZ
28      CONTINUE
        NOLV=N
        NPART=N
        GO TO 14
31      CONTINUE
C       INPUT CELL VELOCITIES. KIN=0,KOUT=0 FOR INTERIOR CELLS.
C       FOR INFLOW, SET KIN=1, FOR OUTFLOW SET KOUT=1.
C
32      TYPE 1010
        TYPE 1009
1010    FORMAT(1H ,'KA,KB,LA,LB,UVEL,VVEL,KIN,KOUT')
        READ(NDEV1,33)KA,KB,LA,LB,UVEL,VVEL,KIN,KOUT
33      FORMAT(4I,2F,2I)
        IF(KA) 38,38,34
34      DO 37 L=LA,LB
        KMM=(L-1)*KMAX
        DO 37 K=KA,KB
        KM=K+KMM
        U(KM)=UVEL
        V(KM)=VVEL
        IF(KIN.EQ.0) GO TO 35
        UIN=UVEL
        VIN=VVEL
35      M(KM)=(M(KM).OR.(KIN*64))
        M(KM)=(M(KM).OR.(KOUT*128))
37      CONTINUE
        GO TO 32
C       SPECIFY A LINE OF INPUT PARTICLES LYING JUST OUTSIDE
C       INPUT BOUNDARY. NBP= NUMBER OF PARTICLES, XXA AND
C       YYA ARE THE COORDINATES OF THE FIRST POINT. DDR AND
C       DDZ GIVE THE SPACING.
C
38      TYPE 1011
        TYPE 1009
1011    FORMAT(1H ,'NBP,XXA,YYA,DDR,DDZ')
        READ(NDEV1,39)NBP,XXA,YYA,DDR,DDZ
39      FORMAT(I,4F)
        IF(NBP)43,43,40
40      MM=NBP+1
        NPART=NPART+NBP
        DO 41 N=NPART,MM,-1
        RP1(N)=RP1(N-NBP)
41      RP2(N)=RP2(N-NBP)
        RP1(1)=XXA
```

```
      RP2(1)=YYA
      K=(XXA+DR)/DR+1
      L=(YYA+DZ)/DZ+1
      KM=KT(L-1)*KMAX
      NBC(I)=KM
      NPRTA(KM)=NPRTA(KM)+1
      DO 42 N=2,NOP
      XXA=XXA+DR
      YYA=YYA+DZ
      RP1(N)=XXA
      RP2(N)=YYA
      K=(XXA+DR)/DR+1
      L=(YYA+DZ)/DZ+1
      KM=KT(L-1)*KMAX
      NPRTA(KM)=NPRTA(KM)+1
42    NBC(N)=KM
      NOLV=NPART
43    CONTINUE
      CALL GENB
C     SET NOBB AND NCORB FLAGS.
C
      LOGICAL TEST
      DO 57 L=2,LMAX2
      KTT=L*LMAX
      DO 57 K=2,KMAX2
      KT=KTT+K
      KM=KT-KMAX
      KB=KM-KMAX
      ITFX=1
      IF((M(KM).AND.OB).EQ.OB) GO TO 44
      GO TO 57
44    IF(DNORX(KM))45,49,47
45    TEST=(PTX(ISEC).EQ.RB(K)).OR.(PTX(1).EQ.RB(K))
      IF(ITFX.EQ.0) TEST=.FALSE.
      IF(((M(KM+1).AND.OB).EQ.OB).OR.((M(KM+1).AND.FULL).EQ.FULL)
     *  .OR.TEST) GO TO 49
      M(KM+1)=(M(KM+1).OR.COR)
      GO TO 49
47    TEST=(PTX(ISEC).EQ.RB(K-1)).OR.(PTX(1).EQ.RB(K-1))
      IF(ITFX.EQ.0) TEST=.FALSE.
      IF(((M(KM-1).AND.OB).EQ.OB).OR.((M(KM-1).AND.FULL).EQ.FULL)
     *  .OR.TEST) GO TO 49
      M(KM-1)=(M(KM-1).OR.COR)
49    IF(DNORY(KM)) 50,54,52
50    TEST=(PTY(ISEC).EQ.ZB(L)).OR.(PTY(1).EQ.ZB(L))
      IF(ITFX.EQ.0) TEST=.FALSE.
      IF(((M(KT).AND.OB).EQ.OB).OR.((M(KT).AND.FULL).EQ.FULL)
     *  .OR.TEST) GO TO 54
      M(KT)=(M(KT).OR.COR)
      GO TO 54
52    TEST=(PTY(ISEC).EQ.ZB(L-1)).OR.(PTY(1).EQ.ZB(L))
      IF(ITFX.EQ.0) TEST=.FALSE.
      IF(((M(KB).AND.OB).EQ.OB).OR.((M(KB).AND.FULL).EQ.FULL)
     *  .OR.TEST) GO TO 54
      M(KB)=(M(KB).OR.COR)
54    CONTINUE
      IF(ITFX.EQ.1) GO TO 55
```

```
      GO TO 57
55    TEST=((M(KB).AND.COR).EQ.COR).OR.((M(KT).AND.COR).EQ.COR)
     *  .OR.((M(KM-1).AND.COR).EQ.COR).OR.((M(KM+1).AND.COR).EQ.COR)
      IF(((M(KM).AND.OB).EQ.OB).AND.(.NOT.TEST)) GO TO 56
      GO TO 57
56    IFX=0
      GO TO 44
57    CONTINUE
C     TURN OFF "OB" FLAG IN CONVEX RIGHT ANGLE BOUNDARIES.
C
      LOGICAL TA,TB,TC,TD
      DO 59 L=2,LMAX2
      KTT=L*KMAX
      DO 59 K=2,KMAX2
      KT=KTT+K
      KM=KT-KMAX
      KB=KM-KMAX
      TA=(((M(KB).AND.OB).EQ.OB).AND.((M(KM-1).AND.OB).EQ.OB)
     *  .AND.((M(KT).AND.COR).NE.COR).AND.((M(KM+1).AND.COR).NE.COR))
      TB=(((M(KB).AND.OB).EQ.OB).AND.((M(KM+1).AND.OB).EQ.OB)
     *  .AND.((M(KM-1).AND.COR).NE.COR).AND.((M(KT).AND.COR).NE.COR))
      TC=(((M(KM+1).AND.OB).EQ.OB).AND.((M(KT).AND.OB).EQ.OB)
     *  .AND.((M(KM-1).AND.COR).NE.COR).AND.((M(KB).AND.COR).NE.COR))
      TD=(((M(KM-1).AND.OB).EQ.OB).AND.((M(KT).AND.OB).EQ.OB)
     *  .AND.((M(KM+1).AND.COR).NE.COR).AND.((M(KB).AND.COR).NE.COR))
      TEST=((M(KM).AND.OB).EQ.OB).AND.(TA.OR.TB.OR.TC.OR.TD)
      IF(TEST) GO TO 58
      GO TO 59
58    M(KM)=(M(KM).AND.MASK2)
59    CONTINUE
      TYPE 1012
      TYPE 1009
1012  FORMAT(1H ,'ICRT,CRTRA,CRTRB,CRTZA,CRTZB,XMAX,YMAX')
      READ(NDEV1,60)ICRT,CRTRA,CRTRB,CRTZA,CRTZB,XMAX,YMAX
      TYPE 1013
      TYPE 1009
1013  FORMAT(1H ,'KDMN,KDMX,LDMN,LDMX')
      READ(NDEV1,61)KDMN,KDMX,LDMN,LDMX
60    FORMAT(I,6F)
61    FORMAT(4I)
      ITEST=0
      PASS=PASS2
      DASV=0.
      CALL REFLAG
      DO 65 L=2,LMAX2
      KL=(L-1)*KMAX+1
      KR=L*KMAX
      IF((M(KL).AND.COR).EQ.COR)GO TO 63
      M(KL)=(M(KL).OR.BND)
63    IF((M(KR).AND.COR).EQ.COR) GO TO 65
      M(KR)=(M(KR).OR.BND)
65    CONTINUE
      DO 69 K=1,KMAX
      K1=K+LMAX2*KMAX
      K12=K+(LMAX2-1)*KMAX
      KB=K1-KMAX
      IF((M(K).AND.COR).EQ.COR) GO TO 67
```

```
      M(K)=(M(K).OR.BND)
67    IF((M(KT).AND.COR).EQ.COR) GO TO 69
      M(KT)=(M(KT).OR.BND)
69    CONTINUE
      IOZ=1
      NTAL=0
      RETURN
      END
```

```
      SUBROUTINE GENB
      COMMON NVAC,NGEN,UAC,VAC,TM1,TM2,TM3
      COMMON NCYCLE,NSTOP,NEDTM,NEDIT,OPE
      COMMON TIME,EDTIM,EDDT,NPIT
      COMMON BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
      COMMON SMIN,EPS1,NBMAX,UBTO,VBTO,UBND,VBND
      COMMON UM,MU,RHOA,GR,GZ,G,ISEC
      COMMON VIN,UIN,DKP,DLP,NBP,DDR,DDZ
      COMMON NEXP,NEXPP,NPART,NDIV
      COMMON NGRID,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      COMMON KDMN,LDMN,KDMX,LDMX,MESS(7)
      COMMON ICRT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
      COMMON ITEST,IOZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
      COMMON NF,NG,NIT,H,XG,HH,HV,GG,FUDGE,NTAL,CRAX,DELZ
      COMMON BPX(300),PTX(300),RMP(300),RNP(300),XB(300)
      COMMON BPY(300),PTY(300),ZMP(300),ZNP(300),YB(300)
      COMMON DS(300),IBP(300)
      COMMON R(150),Z(150),RB(150),ZB(150)
      COMMON DRB(150),DZB(150)
      COMMON M(2500)
      COMMON RP1(5000),RP2(5000)
      COMMON UAVG(2500),VAVG(2500)
      COMMON P(2500),U(2500),ETA(2500),V(2500),PSI(2500)
      COMMON DNORX(2500),DNORY(2500),PNORX(2500),PNORY(2500)
      COMMON NPRTA(2500),NPRTB(2500),MPRTA(2500),MPRTB(2500)
      COMMON SNC(2500),DR,DZ
      COMMON NBC(300),IPX(300)
      COMMON NDEV1,NDEV2
      COMMON/FLAG1/FLAG1
      REAL MU
      INTEGER AND,OR,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB
      DATA INTRF,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB/1,2,4,8,16,32,
     *  64,128,256,512,1024,2048,4096,8192,16384,32768/
      DATA MASK1,MASK2,MASK3,MASK4,MASK5,MASK6,
     *  MASK7,MASK8/077777777757,077777773777,
     *  077777677777,077777777773,077777777767,
     *  077777737777,077777757777,077777767777/
      DOUBLE PRECISION UMAX,VMAX
      DOUBLE PRECISION U,V,PSI,ETA,P
      PI=3.1415926
      DEPSX=.05*DR
      DEPSY=.05*DZ
1     N=1
      TYPE 1001
1001  FORMAT(1H ,'XB,YB,NSEG',$)
      TYPE 2001
2001  FORMAT(1H )
2     READ(NDEV1,3) XB(N),YB(N),NSEG
3     FORMAT(2F,I)
      IF(NSEG)88,4,5
4     N=N+1
      GO TO 2
5     NBMAX=N-1
      ISEC=0
      DO 32 I=1,NBMAX
```

```
      IF(XB(I+1)-XB(I))10,14,6
6     K1=(XB(I)+.000001)/DR+2
      A=(YB(I+1)-YB(I))/(XB(I+1)-XB(I))
      KDK=1
      MM=0
      K2=KMAX
      B=YB(I)-A*XB(I)
      DO 8 K=K1,K2,KDK
      IF(RB(K)-XB(I+1)-0.000001)7,7,9
7     MM=MM+1
      RMP(MM)=RB(K)
      ZMP(MM)=A*RMP(MM)+B
8     CONTINUE
9     KLINE=MM
      GO TO 15
10    K1=(XB(I)-.000001)/DR+1
      A=(YB(I+1)-YB(I))/(XB(I+1)-XB(I))
      K2=1
      B=YB(I)-A*XB(I)
      KDK=-1
      MM=0
      DO 12 K=K1,K2,KDK
      IF(RB(K)-XB(I+1)+.000001) 13,11,11
11    MM=MM+1
      RMP(MM)=RB(K)
12    ZMP(MM)=A*RMP(MM)+B
13    KLINE=MM
      GO TO 15
14    KLINE=0
15    CONTINUE
      IF(YB(I+1)-YB(I))20,24,16
16    L1=(YB(I)+.000001)/DZ+2
      A=(XB(I+1)-XB(I))/(YB(I+1)-YB(I))
      L2=LMAX
      B=XB(I)-A*YB(I)
      LDL=1
      N=0
      DO 18 L=L1,L2,LDL
      IF(ZB(L)-YB(I+1)-.000001)17,17,19
17    N=N+1
      ZNP(N)=ZB(L)
18    RNP(N)=A*ZNP(N)+B
19    LLINE=N
      GO TO 25
20    L1=(YB(I)-.000001)/DZ+1
      A=(XB(I+1)-XB(I))/(YB(I+1)-YB(I))
      L2=1
      B=XB(I)-A*YB(I)
      LDL=-1
      N=0
      DO 22 L=L1,L2,LDL
      IF(ZB(L)-YB(I+1)+.000001)23,21,21
21    N=N+1
      ZNP(N)=ZB(L)
22    RNP(N)=ZNP(N)*A+B
23    LLINE=N
      GO TO 25
```

```
24      LLINE=0
25      CONTINUE
        DO 26 J=1,LLINE
        JJ=J+KLINE
        RMP(JJ)=RNP(J)
26      ZMP(JJ)=ZNP(J)
        JMAX=KLINE+LLINE
        DO 27 J=1,JMAX
27      DS(J)=(RMP(J)-XB(I))**2+(ZMP(J)-YB(I))**2
        DO 30 N=1,JMAX
        TEST=10.0**5
        MTRANS=0
        DO 29 MM=1,JMAX
        IF(DS(MM)-TEST)28,28,29
28      TEST=DS(MM)
        MTRANS=MM
29      CONTINUE
        RNP(N)=RMP(MTRANS)
        ZNP(N)=ZMP(MTRANS)
        DS(MTRANS)=10.0**7
30      CONTINUE
        J1=ISEC+1
        J2=ISEC+JMAX
        N=0
        DO 31 J=J1,J2
        N=N+1
        PTX(J)=RNP(N)
        PTY(J)=ZNP(N)
31      CONTINUE
        ISEC=ISEC+JMAX
32      CONTINUE
        I=1
33      I=I+1
34      IF(ABS(PTX(I)-PTX(I-1))-DEPSX)35,35,38
35      IF(ABS(PTY(I)-PTY(I-1))-DEPSY)36,36,38
36      ISEC=ISEC-1
        DO 37 J=I,ISEC
        PTX(J)=PTX(J+1)
        PTY(J)=PTY(J+1)
37      CONTINUE
        IF(I-ISEC)34,34,39
38      IF(I-ISEC)33,39,39
39      CONTINUE
        ISECZ=ISEC-1
        DEP=(PTX(ISEC)-PTX(ISECZ))**2+(PTY(ISEC)-PTY(ISECZ))**2
        DEP=SQRT(DEP)
        IF(DEP-.2*DR)41,42,42
41      ISEC=ISECZ
42      CONTINUE
        DEP=(PTX(2)-PTX(1))**2+(PTY(2)-PTY(1))**2
        DEP=SQRT(DEP)
        IF(DEP-.2*DR)43,44,44
43      I=1
        GO TO 45
44      I=0
45      I=I+1
        K=(.5*(PTX(I+1)+PTX(I)))/DR+2
```

```
      L=(1.000001*PTY(I))/DZ+1
      IF((PTY(I).EQ.PTY(I+1)).AND.(PTY(I).EQ.ZB(L)))GO TO 46
      GO TO 49

46    IF(PTX(I+1)-PTX(I))48,48,47
47    KM=L*KMAX+K
      KB=KM-KMAX
      UNORX(KM)=0.
      UNORY(KM)=1.
      PNORX(KM)=.5*(PTX(I)+PTX(I+1))
      PNORY(KM)=PTY(I)
      M(KM)=(M(KM).OR.OB)
      M(KB)=(M(KB).OR.COR)
      GO TO 53
48    KM=(L-1)*KMAX+K
      KT=KM+KMAX
      UNORX(KM)=0.
      UNORY(KM)=-1.
      PNORX(KM)=.5*(PTX(I)+PTX(I+1))
      PNORY(KM)=PTY(I)
      M(KT)=(M(KT).OR.COR)
      M(KM)=(M(KM).OR.OB)
      GO TO 53
49    L=(.5*(PTY(I+1)+PTY(I)))/DZ+2
      K=(1.000001*PTX(I))/DR+1
      IF((PTX(I).EQ.PTX(I+1)).AND.(PTX(I).EQ.RB(K)))GO TO 50
      GO TO 40
50    IF(PTY(I+1)-PTY(I))52,52,51
51    KM=(L-1)*KMAX+K-1
      UNORX(KM)=-1.
      UNORY(KM)=0.
      PNORX(KM)=PTX(I)
      PNORY(KM)=.5*(PTY(I+1)+PTY(I))
      M(KM)=(M(KM).OR.OB)
      M(KM+1)=(M(KM+1).OR.COR)
      GO TO 53
52    KM=(L-1)*KMAX+K
      UNORX(KM)=1.
      UNORY(KM)=0.
      PNORX(KM)=PTX(I)
      PNORY(KM)=.5*(PTY(I+1)+PTY(I))
      M(KM-1)=(M(KM-1).OR.COR)
      M(KM)=(M(KM).OR.OB)
53    CONTINUE
      GO TO 83
40    CONTINUE
      DX=PTX(I+1)-PTX(I)
      XBAR=.5*(PTX(I)+PTX(I+1))
      DY=PTY(I+1)-PTY(I)
      YBAR=.5*(PTY(I)+PTY(I+1))
      K=XBAR/DR+1
      L=YBAR/DZ+1
      SUM=-YBAR*DX
      IF(PTX(I+1)-RB(K))57,54,57
54    IF(PTY(I)-ZB(L))55,56,55
55    SUM=SUM-ZB(L)*DR-(PTX(I)-RB(K+1))*ZB(L+1)
      GO TO 64
```

```
56      SUM=SUM-(PTX(I)-RB(K))*ZB(L)
        GO TO 64
57      IF(PTY(I+1)-ZB(L))59,58,59
58      SUM=SUM-(RB(K+1)-PTX(I+1))*ZB(L)-(PTX(I)-RB(K+1))*ZB(L+1)
        GO TO 64
59      IF(PTX(I+1)-RB(K+1))63,60,63
60      IF(PTY(I)-ZB(L+1))61,62,61
61      SUM=SUM+ZB(L+1)*DR-(PTX(I)-RB(K))*ZB(L)
        GO TO 64
62      SUM=SUM-(PTX(I)-RB(K+1))*ZB(L+1)
        GO TO 64
63      SUM=SUM-(RB(K)-PTX(I+1))*ZB(L+1)-(PTX(I)-RB(K))*ZB(L)
64      CONTINUE
        KPRM=K+1
        LPRM=L+1
        IF(SUM-SMIN)65,79,79
65      IF(ABS(DY)-ABS(DX))69,69,66
66      IF(DY)67,67,68
67      KPRM=K+2
        GO TO 72
68      KPRM=K
        GO TO 72
69      IF(DX)70,70,71
70      LPRM=L
        GO TO 72
71      LPRM=L+2
72      CONTINUE
        KKM=L*KMAX+K+1
        M(KKM)=(M(KKM).OR.COR)
        XAVG=.5*(PTX(I+2)+PTX(I+1))
        YAVG=.5*(PTY(I+2)+PTY(I+1))
        KAVG=XAVG/DR+2
        LAVG=YAVG/DZ+2
        IF((KPRM.EQ.KAVG).AND.(LPRM.EQ.LAVG)) GO TO 73
        GO TO 75
73      ISEC=ISEC-1
        IZ=I+1
        DO 74 J=IZ,ISEC
        PTX(J)=PTX(J+1)
        PTY(J)=PTY(J+1)
74      CONTINUE
        GO TO 78
75      XAVG=.5*(PTX(I)+PTX(I-1))
        YAVG=.5*(PTY(I)+PTY(I-1))
        KAVG=XAVG/DR+2
        LAVG=YAVG/DZ+2
        KCEL=(LAVG-1)*KMAX+KAVG
        KCEL=KCEL+NPRTB(KCEL)
        KMPT=(LPRM-1)*KMAX+KPRM
        IF(KMPT.EQ.KCEL) GO TO 76
        GO TO 79
76      ISEC=ISEC-1
        IZ=I
        I=I-1
        DO 77 J=IZ,ISEC
        PTX(J)=PTX(J+1)
        PTY(J)=PTY(J+1)
```

```
77      CONTINUE
76      CONTINUE
        XBAR=.5*(PTX(I+1)+PTX(I))
        YBAR=.5*(PTY(I+1)+PTY(I))
        DX=PTX(I+1)-PTX(I)
        DY=PTY(I+1)-PTY(I)
79      CONTINUE
        FMAG=SQRT(DX**2+DY**2)
        IF(FMAG)80,80,82
80      TYPE 81
81      FORMAT(1H ,'FMAG=0')
        CALL EXIT
82      CONTINUE
        KM=(LPRM-1)*KMAX+KPRM
        KAVG=XBAR/DX+2
        LAVG=YBAR/DZ+2
        KBAR=(LAVG-1)*KMAX+KAVG
        KSTOR=KM-KBAR
        NPRTB(KBAR)=KSTOR
        M(KM)=(M(KM).OR.OB)
        DNORX(KM)=-DY/FMAG
        DNORY(KM)=DX/FMAG
        PNORX(KM)=XBAR
        PNORY(KM)=YBAR
85      IF(1-ISEC+1)45,84,84
84      CONTINUE
        DO 87 N=1,NPART
        X1=RP1(N)
        X2=RP2(N)
        L=X2/DZ+2
        K=X1/DX+2
        KBAR=(L-1)*KMAX+K
        KSTOR=(NPRTB(KBAR))
        KM=KBAR+KSTOR
        IF((DNORX(KM).NE.0).OR.(DNORY(KM).NE.0)) GO TO 85
        GO TO 87
85      PDOTN=(XR-PNORX(KM))*DNORX(KM)+(XZ-PNORY(KM))*DNORY(KM)
        IF(PDOTN-EPS1)86,86,87
86      M(KM)=(M(KM).OR.ARB)
87      CONTINUE
5500    FORMAT(1H ,2F)
        IF(NSW-2)88,1,1
88      CONTINUE
5501    FORMAT(1H ,3F)
5502    FORMAT(1H ,4F)
        RETURN
        END
```

```
      SUBROUTINE REFLAG
      COMMON NVAC,NGEN,UAC,VAC,TM1,TM2,TM3
      COMMON NCYCLE,NSTOP,NEDTM,NEDIT,DPE
      COMMON TIME,EDTIM,EDDT,NPIT
      COMMON BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
      COMMON SMIN,EPS1,NDMAX,UBTO,VBTO,UBND,VBND
      COMMON UM,MU,RHOA,GR,GZ,G,ISEC
      COMMON VIN,UIN,DKP,DLP,NBP,DDR,DDZ
      COMMON NEXP,NEXPP,NPART,NDIV
      COMMON NGRID,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      COMMON KDMN,LDMN,KDMX,LDMX,MESS(7)
      COMMON ICRT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
      COMMON ITEST,IOZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
      COMMON NF,NG,NIT,H,XG,HH,HV,GG,FUDGE,NTAL,CRAX,DELZ
      COMMON BPX(300),PTX(300),RMP(300),RNP(300),XB(300)
      COMMON BPY(300),PTY(300),ZMP(300),ZNP(300),YB(300)
      COMMON DS(300),IBP(300)
      COMMON R(150),Z(150),RB(150),ZB(150)
      COMMON DRB(150),DZB(150)
      COMMON M(2500)
      COMMON RP1(5000),RP2(5000)
      COMMON UAVG(2500),VAVG(2500)
      COMMON P(2500),U(2500),ETA(2500),V(2500),PSI(2500)
      COMMON DNORX(2500),DNORY(2500),PNORX(2500),PNORY(2500)
      COMMON NPRTA(2500),NPRTB(2500),MPRTA(2500),MPRTB(2500)
      COMMON SNC(2500),DR,DZ
      COMMON NBC(300),IPX(300)
      COMMON NDEV1,NDEV2
      COMMON/FLAG1/FLAG1
      REAL MU
      INTEGER AND,OR,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  EPSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB
      DATA INTRF,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  EPSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB/1,2,4,8,16,32,
     *  64,128,256,512,1024,2048,4096,8192,16384,32768/
      DATA MASK1,MASK2,MASK3,MASK4,MASK5,MASK6,
     *  MASK7,MASK8/O777777777757,O777777773777,
     *  O777777677777,O777777777773,O777777777767,
     *  O777777737777,O777777757777,O777777767777/
      DOUBLE PRECISION UMAX,VMAX
      DOUBLE PRECISION U,V,PSI,ETA,P
      PI=3.1415926
      LOGICAL TA,TB,TC,TV,TH,TEST
      LOGICAL ANY,AA,AB,AC,AD
1     DO 23 L=2,LMAXZ
      KMM=L*KMAX
      DO 23 K=2,KMAXZ
      KT=K+KMM
      KM=KT-KMAX
      KB=KM-KMAX
      IF((M(KM).AND.SUR).EQ.SUR) GO TO 2
      GO TO 13
2     IF(NPRTA(KM))23,3,23
3     IF((M(KM).AND.OB).EQ.OB) GO TO 4
      GO TO 5
4     TEST=((M(KM+1).AND.COR).EQ.COR).OR.((M(KM+1).AND.SUR).EQ.SUR
     *  .OR.((M(KM+1).AND.FULL).EQ.FULL).OR.((M(KM+1).AND.BND).EQ.BN
```

```
      TEST=TEST.AND.(((M(KM-1).AND.COR).EQ.COR).OR.
     *  (((M(KM-1).AND.SOR).EQ.SOR).OR.((M(KM-1).AND.FULL).EQ.FULL)
     *  .OR.((M(KM-1).AND.BND).EQ.BND))
      TEST=TEST.AND.(((M(KT).AND.COR).EQ.COR).OR.
     *  ((M(KT).AND.SOR).EQ.SOR).OR.((M(KT).AND.FULL).EQ.FULL)
     *  .OR.((M(KT).AND.BND).EQ.BND))
      TEST=TEST.AND.(((M(KB).AND.COR).EQ.COR).OR.
     *  ((M(KB).AND.SOR).EQ.SOR).OR.((M(KB).AND.FULL).EQ.FULL)
     *  .OR.((M(KB).AND.BND).EQ.BND))
      IF(TEST) GO TO 23
5     M(KM)=(M(KM).OR.EMP)
      M(KM)=(M(KM).AND.MASK3)
      M(KM)=(M(KM).AND.MASK4)
      P(KM)=0.0
      IF((M(KM+1).AND.EMP).EQ.EMP) GO TO 6
      GO TO 7
6     CONTINUE
      U(KM)=0.0
7     IF((M(KM-1).AND.EMP).EQ.EMP) U(KM-1)=0.0
9     IF((M(KT).AND.EMP).EQ.EMP) V(KM)=0.0
11    IF((M(KB).AND.EMP).EQ.EMP) V(KB)=0.0
      GO TO 23
13    IF((M(KM).AND.EMP).EQ.EMP) GO TO 14
      GO TO 23
14    P(KM)=0.0
      NNN=NPRTA(KM)
      IF(NNN)15,23,15
15    M(KM)=(M(KM).OR.SOR)
      M(KM)=(M(KM).AND.MASK1)
      J=KM
      VBAR=VAVG(J)
      VBAR=VBAR/NNN
      UBAR=UAVG(J)
      UBAR=UBAR/NNN
      IF(((M(KM+1).AND.EMP).EQ.EMP).OR.((M(KM+1).AND.OUT).EQ.OUT))
     *  GO TO 16
      GO TO 17
16    U(KM)=UBAR
17    IF(((M(KM-1).AND.EMP).EQ.EMP).OR.((M(KM-1).AND.OUT).EQ.OUT))
     *  U(KM-1)=UBAR
19    IF(((M(KT).AND.EMP).EQ.EMP).OR.((M(KT).AND.OUT).EQ.OUT))
     *  V(KM)=VBAR
21    IF(((M(KB).AND.EMP).EQ.EMP).OR.((M(KB).AND.OUT).EQ.OUT))
     *  V(KB)=VBAR
23    CONTINUE
      DO 47 L=2,LMAX2
      KMM=L*KMAX
      DO 47 K=2,KMAX2
      KT=KT+KMM
      KM=KT-KMAX
      KB=KM-KMAX
      IF(NPRTA(KM)) 24,47,24
24    IF((M(KM).AND.FULL).EQ.FULL) GO TO 25
      GO TO 30
25    IF((M(KM+1).AND.EMP).EQ.EMP) GO TO 29
      GO TO 26
26    IF((M(KM-1).AND.EMP).EQ.EMP) GO TO 29
```

```
      IF((M(KT).AND.EMP).EQ.EMP) GO TO 29
      IF((M(KB).AND.EMP).EQ.EMP) GO TO 29
      GO TO 47
29    M(KM)=(M(KM).OR.SUR)
      M(KM)=(M(KM).AND.MASK5)
      P(KM)=0.0
      GO TO 47
30    IF((M(KM).AND.OB).EQ.OB) GO TO 47
      IF((M(KM).AND.SUR).EQ.SUR) GO TO 32
      GO TO 47
32    SGP=0.0
      P(KM)=0.0
      N=0
      IF((M(KM+1).AND.EMP).EQ.EMP) GO TO 47
      IF((M(KM+1).AND.FULL).EQ.FULL) GO TO 34
      GO TO 35
34    SGP=SGP+P(KM+1)
      N=N+1
35    IF((M(KM-1).AND.EMP).EQ.EMP) GO TO 47
      IF((M(KM-1).AND.FULL).EQ.FULL) GO TO 37
      GO TO 38
37    SGP=SGP+P(KM-1)
      N=N+1
38    IF((M(KT).AND.EMP).EQ.EMP) GO TO 47
      IF((M(KT).AND.FULL).EQ.FULL) GO TO 40
      GO TO 41
40    SGP=SGP+P(KT)
      N=N+1
41    IF((M(KB).AND.EMP).EQ.EMP) GO TO 47
      IF((M(KB).AND.FULL).EQ.FULL) GO TO 43
      GO TO 44
43    SGP=SGP+P(KB)
      N=N+1
44    M(KM)=(M(KM).OR.FULL)
      M(KM)=(M(KM).AND.MASK6)
      M(KM)=(M(KM).AND.MASK4)
      IF(N)45,45,46
45    P(KM)=0.0
      GO TO 47
46    P(KM)=SGP/N
47    CONTINUE
      DO 66 L=2,LMAX2
      KTT=L*KMAX
      DO 66 K=2,KMAX2
      KT=KTT+K
      KM=KT-KMAX
      KB=KM-KMAX
      IF((M(KM).AND.OB).EQ.OB) GO TO 48
      GO TO 66
48    AA=((M(KM).AND.OK).EQ.OK)
      TA=((M(KM+1).AND.OB).EQ.OB).AND.((M(KM+1).AND.EMP).EQ.EMP)
      TB=((M(KT).AND.OB).EQ.OB).AND.((M(KT).AND.EMP).EQ.EMP)
      TC=((M(KM-1).AND.OB).EQ.OB).AND.((M(KM-1).AND.EMP).EQ.EMP)
      TV=((M(KB).AND.OB).EQ.OB).AND.((M(KB).AND.EMP).EQ.EMP)
      TEST=TA.OR.TB.OR.TC.OR.TV
      M(KM)=(M(KM).AND.MASK7)
      IF(((M(KM).AND.OB).EQ.OB).AND.((M(KM).AND.ARB).EQ.ARB).AND.
```

```
     *   (.NOT.TEST)) M(KM)=(M(KM).OR.OK)
         TEST=.FALSE.
         IF(DNORX(KM))49,51,50
49       IF((M(KM-1).AND.EMP).EQ.EMP) TEST=.TRUE.
         GO TO 51
50       IF((M(KM+1).AND.EMP).EQ.EMP) TEST=.TRUE.
51       IF(DNORY(KM))52,54,53
52       IF((M(KB).AND.EMP).EQ.EMP) TEST=TEST.OR..TRUE.
         GO TO 54
53       IF((M(KT).AND.EMP).EQ.EMP) TEST=TEST.OR..TRUE.
54       IF(TEST.AND.((M(KM).AND.OK).EQ.OK)) M(KM)=(M(KM).AND.MASK7)
         IF((M(KM).AND.OK).EQ.OK) GO TO 56
         P(KM)=0.
         GO TO 68
56       TEST=(AA.EQ..FALSE.).AND.((M(KM).AND.OK).EQ.OK)
         IF(TEST) GO TO 57
         GO TO 68
57       N=0
         SP=0.
         IF(((M(KT).AND.OK).EQ.OK).OR.((M(KT).AND.FULL).EQ.FULL))
     *   GO TO 58
         GO TO 59
58       SP=SP+P(KT)
         N=N+1
59       IF(((M(KB).AND.OK).EQ.OK).OR.((M(KB).AND.FULL).EQ.FULL))
     *   GO TO 60
         GO TO 61
60       SP=SP+P(KB)
         N=N+1
61       IF(((M(KM+1).AND.OK).EQ.OK).OR.((M(KM+1).AND.FULL).EQ.FULL))
     *   GO TO 62
         GO TO 63
62       SP=SP+P(KM+1)
         N=N+1
63       IF(((M(KM-1).AND.OK).EQ.OK).OR.((M(KM-1).AND.FULL).EQ.FULL))
     *   GO TO 64
         GO TO 65
64       SP=SP+P(KM-1)
         N=N+1
65       IF(N)68,68,66
66       P(KM)=SP/N
68       CONTINUE
         CALL VELUS
         DO 71 L=2,LMAXZ
         KMM=(L-1)*KMAX
         DO 71 K=2,KMAXZ
         KM=K+KMM
         IF(((M(KM).AND.EMP).EQ.EMP).OR.((M(KM).AND.SUR).EQ.SUR))
     *   GO TO 69
         GO TO 71
69       IF((M(KM).AND.ARB).EQ.ARB) GO TO 71
         P(KM)=PAMB
71       CONTINUE
         RETURN
         END
```

```
      SUBROUTINE EDIT
      COMMON NVAC,NGEN,UAC,VAC,TM1,TM2,TM3
      COMMON NCYCLE,NSTOP,NEDTM,NEDIT,OPE
      COMMON TIME,EDTIM,EDDT,NPIT
      COMMON BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
      COMMON SMIN,EPS1,NBMAX,UBTO,VBTO,UBND,VBND
      COMMON UM,MU,RHOA,GR,GZ,G,ISEC
      COMMON VIN,UIN,DKP,DLP,NBP,DDR,DDZ
      COMMON NEXP,NEXPP,NPART,NDIV
      COMMON NGRID,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      COMMON KDMN,LDMN,KDMX,LDMX,MESS(7)
      COMMON ICRT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
      COMMON ITEST,IOZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
      COMMON NF,NG,NIT,H,XG,RH,HV,GG,FUDGE,NTAL,CRAX,DELZ
      COMMON BPX(300),PTX(300),KMP(300),KNP(300),XB(300)
      COMMON BPY(300),PTY(300),ZMP(300),ZNP(300),YB(300)
      COMMON DS(300),IBP(300)
      COMMON R(150),Z(150),RB(150),ZB(150)
      COMMON DRB(150),DZB(150)
      COMMON M(2500)
      COMMON RP1(5000),RP2(5000)
      COMMON UAVG(2500),VAVG(2500)
      COMMON P(2500),U(2500),ETA(2500),V(2500),PSI(2500)
      COMMON DNORX(2500),DNORY(2500),PNORX(2500),PNORY(2500)
      COMMON NPRTA(2500),NPRTB(2500),MPRTA(2500),MPRTB(2500)
      COMMON SNC(2500),DR,DZ
      COMMON NBC(300),IPX(300)
      COMMON NDEV1,NDEV2
      COMMON/FLAG1/FLAG1
      REAL MU
      INTEGER AND,OR,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB
      DATA INTRF,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB/1,2,4,8,16,32,
     *  64,128,256,512,1024,2048,4096,8192,16384,32768/
      DATA MASK1,MASK2,MASK3,MASK4,MASK5,MASK6,
     *  MASK7,MASK8/07777777777,07777777773777,
     *  07777777677777,07777777777773,07777777777767,
     *  07777777737777,07777777757777,07777777677777/
      DOUBLE PRECISION U,V,PSI,ETA,P
      DOUBLE PRECISION VMAX,UMAX
      PI=3.1415926
      WRITE(22,1)
1     FORMAT(1H ,'EDIT')
2     FORMAT(1H ,'L=',I5,5X,'K,ETA,PSI,U,V,P,X OR R,UPART,VPART
4     FORMAT(1H ,'CYCLE',I8,' TIME',E,' DT',E,' NPART',
     *  I10,' PAMB',F)
5     WRITE(22,4)NCYCLE,TIME,DT,NPART,PAMB
      WRITE(22,7)RHOA,MU
7     FORMAT(1H ,5X,'RHOA=',E,' MU=',E)
      WRITE(22,8)
8     FORMAT(1H ,'SURFACE PARTICLES')
      MM=0
      DO 14 N=1,NPART
      XX=RP1(N)
      YY=RP2(N)
      K=XX/DR+2
```

```
      L=YY/DZ+2
      KM=KT(L-1)*KMAX
      IF((M(KM).AND.SUR).EQ.SUR) GO TO 9
      GO TO 14
9     MM=MM+1
      GO TO (10,11,12),MM
10    X1=XX
      Y1=YY
      GO TO 14
11    X2=XX
      Y2=YY
      GO TO 14
12    X3=XX
      Y3=YY
      MM=0
13    FORMAT(1H ,9X,2E,2X,2E,2X,2E)
      WRITE(22,13)X1,Y1,X2,Y2,X3,Y3
14    CONTINUE
      DO 27 L=LOMN,LOMX
      KTT=(L-1)*KMAX
      WRITE(22,15)(MESS(I),I=1,7)
15    FORMAT(1H ,7A5)
      WRITE(22,16)Z(L),NCYCLE,TIME,MU,RHOA
16    FORMAT(1H ,'Z=',E,' CYCLE=',I,' TIME=',E,' MU=',
     *  E,'RHO=',E)
      WRITE(22,2)L
      DO 21 K=KOMN,KOMX
      KP=K+KTT
      UPART=UAVG(KP)
      VPART=VAVG(KP)
      NP=NPRTA(KP)
      IF(NP)17,17,18
17    UPART=0.
      VPART=0.
      GO TO 19
18    UPART=UPART/NP
      VPART=VPART/NP
19    TEST=(U(KP).NE.0).OR.(V(KP).NE.0)
      IF(TEST) GO TO 20
      GO TO 21
20    CONTINUE
      WRITE(22,3)K,ETA(KP),PSI(KP),U(KP),V(KP),P(KP),R(K),
     *  UPART,VPART
3     FORMAT(1H ,I4,8E15.8)
21    CONTINUE
      IF(OPE.GT.0) GO TO 22
      GO TO 27
22    WRITE(22,23)L
23    FORMAT(1H ,'L=',I5,'K,FULL,SUR,EMP,OB,OK,ARB,COR,BND,IN,OUT,G
     *  EMPBND,SNC,NPT,KMPR,DNORX,DNORY,PNORX,PNORY,UBND,VBND')
      DO 26 K=KOMN,KOMX
      KM=K+KTT
      TEST=(NPRTA(KM).NE.0).OR.(AND(M(KM),COR).EQ.COR)
     *  .OR.(AND(M(KM),OB).EQ.OB).OR.(AND(M(KM),SUR).EQ.SUR)
     *  .OR.(AND(M(KM),FULL).EQ.FULL).OR.(AND(M(KM),BND).EQ.BND)
     *  .OR.(AND(M(KM),IN).EQ.IN).OR.(AND(M(KM),OUT).EQ.OUT)
     *  .OR.(.NOT.(AND(M(KM),EMP).EQ.EMP))
```

```
      IF(TEST) GO TO 24
      GO TO 26
24    MA=0
      IF(AND(M(KM),FULL).EQ.FULL) MA=1
      MB=0
      IF(AND(M(KM),SUR).EQ.SUR) MB=1
      MC=0
      IF(AND(M(KM),EMP).EQ.EMP) MC=1
      MD=0
      IF(AND(M(KM),OB).EQ.OB) MD=1
      ME=0
      IF(AND(M(KM),OR).EQ.OR) ME=1
      MF=0
      IF(AND(M(KM),ARB).EQ.ARB) MF=1
      MG=0
      IF(AND(M(KM),COR).EQ.COR) MG=1
      MH=0
      IF(AND(M(KM),BND).EQ.BND) MH=1
      MI=0
      IF(AND(M(KM),IN).EQ.IN) MI=1
      MJ=0
      IF(AND(M(KM),OUT).EQ.OUT) MJ=1
      MK=0
      IF(AND(M(KM),GAS).EQ.GAS) MK=1
      ML=0
      IF(AND(M(KM),EMPBND).EQ.EMPBND) ML=1
      MM=SNC(KM)
      NA=NPRTA(KM)
      NB=NPRTB(KM)
      WRITE(22,25)K,MA,MB,MC,MD,ME,MF,MG,MH,MI,MJ,MK,ML,
     *  MM,NA,NB,DNORX(KM),DNORY(KM),PNORX(KM),PNORY(KM)
25    FORMAT(1H ,I5,1X,3I1,1X,4I1,1X,3I1,1X,3I1,1X,2I3,
     *  4F)
26    CONTINUE
27    CONTINUE
      TYPE 28,NCYCLE
      WRITE(22,28)NCYCLE
28    FORMAT(1H ,'EDIT COMPLETE FOR CYCLE',I4)
      END FILE 22
      RETURN
      END
```

APPENDIX III

THE COMPUTATIONAL PROGRAM

Routine Name	Function
MAIN	Read in data from initialization program, compute ETA and PSI values, controls other computation.
PRES	Pressure Iteration
VEL	Velocity computation
VEL55	Free surface velocities
NOSLIP	Takes care of no-slip boundary condition

```
      COMMON NVAC,NGEN,UAC,VAC,TM1,TM2,TM3
      COMMON NCYCLE,NSTOP,NEDTM,NEDIT,OPE
      COMMON TIME,EDTIM,EDDT,NPIT,NITER
      COMMON BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
      COMMON SMIN,EPS1,NBMAX,UBTO,VBTO,UBND,VBND
      COMMON UM,NU,RHOA,GR,GZ,G,ISEC
      COMMON VIN,UIN,UKE,DLP,NBP,DDR,DDZ
      COMMON NEXP,NEXPP,NPART,NDIV
      COMMON NGRID,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      COMMON KDMN,LDMN,KDMX,LDMX,MESS(7)
      COMMON ICRT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
      COMMON ITEST,IOZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
      COMMON NF,NG,NIT,H,XG,HH,HV,GG,FUDGE,NTAL,CRAX,DELZ
      COMMON R(150),Z(150),RB(150),ZB(150)
      COMMON DRB(150),DZB(150)
      COMMON M(2500)
      COMMON P(2500),U(2500),ETA(2500),V(2500),PSI(2500)
      COMMON DNORX(2500),DNORY(2500),PNORX(2500),PNORY(2500)
      COMMON DR,DZ
      COMMON NDEV1,NDEV2
      COMMON/FLAG1/FLAG1
      COMMON/MDTN/MDTN
      COMMON/FLGS/TTY,NDMP,NEDT,NSTP,NPIM
      COMMON/STUFF/ERRORS,NAME,DIVS,VCHM,UCHM
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/LOOK/PMX,UMX,VMX,PMX1,UMX1,VMX1
      COMMON/NX/UTMAX
      LOGICAL TTY,NDMP,NEDT,NSTP,NPIM
      LOGICAL NF1B
      REAL MU
      INTEGER AND,OR,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     * FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB
      DATA INTRF,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     * FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB/1,2,4,8,16,32,
     * 64,128,256,512,1024,2048,4096,8192,16384,32768/
      DATA MASK1,MASK2,MASK3,MASK4,MASK5,MASK6,
     * MASK7,MASK8/077777777757,077777773777,
     * 077777677777,077777777773,077777777767,
     * 077777757777,077777757777,077777767777/
      CALL PSEUD
      TYPE 301
301   FORMAT(1H ,'PROCESS OR EXAMINE?',$)
      ACCEPT 302,IDO
302   FORMAT(A1)
      IF(IDO.EQ.'E')CALL EXAMIN
      TYPE 1
1     FORMAT(1H ,'ADMAC')
      TYPE 2
2     FORMAT(1H ,'SAVED CORE?',$)
      ACCEPT 3,NYS
3     FORMAT(A1)
      IF(NYS.EQ.'Y') GO TO 32
      FLAG1=0
      TYPE 4
4     FORMAT(1H ,'INPUT FILE=',$)
      ACCEPT 5,NAME
5     FORMAT(A5)
```

```
      NAME=NAME+32
      NAME1=NAME+2
      CALL IFILE(22,NAME1)
      READ(22)NGEN
      READ(22)NCYCLE,NSTOP,NEDTM,NEDIT,OPE
      READ(22)TIME,EDTIM,EDUT,NPIT
      READ(22)BETA,EPS2,DT,VMAX,UMAX,RELAX,OUT
      READ(22)SMIN,EPSI,NBMAX,UBTO,VBTO,UBND,VBND
      READ(22)UM,MU,RHOA,GR,GZ,G,ISEC
      READ(22)VIN,UIN,DKP,DLP,NBP,DDR,DDZ
      READ(22)NEXP,NEXPP,NPART,NDIV
      READ(22)NGRID,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      READ(22)KDMN,LDMN,KDMX,LDMX,MESS
      READ(22)ICRT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
      READ(22)ITEST,IOZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
      READ(22)MT,NG,NIT,H,XG,HH,HV,GG,FUDGE,NTAL,CRAX,DELZ
      READ(22)R,Z,RB,ZB
      READ(22)DRB,DZB
      READ(22)M
      READ(22)F,U,ETA,V,PSI
      READ(22)DNORX,DNORY,PNORX,PNORY
      READ(22)DR,DZ
      DTMAX=DR**2*DZ**2/(DR**2+DZ**2)
      DTMAX=.4*DTMAX/(MU+.000001)*RHOA
      DT=.5*(DT+DTMAX)-.5*ABS(DT-DTMAX)
      DTMAX=DT
      RELAX=.2*RHOA*DR**2/DT
      TYPE 6
6     FORMAT(1H ,'UPDATE?',$)
      ACCEPT 5,NYS
      IF(NYS.EQ.'N')GO TO 32
      NAME2=NAME+6
      CALL IFILE(23,NAME2)
      READ(23)M
      READ(23)U,V,P
      READ(23)DT,RHOA,TIME,NCYCLE,MU,EPS1
      READ(23)KDMN,KDMX,LDMN,LDMX,DR,DZ,KMAX,LMAX,DTMAX
      TYPE 7
7     FORMAT(1H ,'NPIT=',$)
      ACCEPT 8,NPIT
8     FORMAT(I)
      TYPE 9
9     FORMAT(1H ,'NDMPT=',$)
      ACCEPT 8,NDMPT
      TYPE 10
10    FORMAT(1H ,'NEDIT=',$)
      ACCEPT 8,NEDIT
      NEDTM=NCYCLE+NEDIT
32    CONTINUE
      TYPE 800
      NFIB=.FALSE.
800   FORMAT(1H ,'FIB RUN?',$)
      ACCEPT 801,NYS
801   FORMAT(A1)
      IF(NYS.EQ.'N')GO TO 802
      NFIB=.TRUE.
802   CONTINUE
```

```
      TYPE 11
11    FORMAT(1H ,'CREATE SHARE FILE?',$)
      ACCEPT 3,NYS
      IF (NYS.EQ.'Y')CALL FILE
      TTY=.TRUE.
      NP1M=.FALSE.
      NUMP=.FALSE.
      NEDT=.FALSE.
      NSTP=.FALSE.
      TFV=.25
      TWO=2.0
      NSQR=2
      FV=.5
33    CONTINUE
      UM=MU*DT
      DO 34 L=1,NGRID
      PSI(L)=0.0
      ETA(L)=0.0
34    CONTINUE
      DO 46 L=2,LMAXZ
      KMM=(L-1)*KMAX
      IF(NGEN .EQ. 0) GO TO 36
35    UR=TFV*(U(KMM+1)+U(KMM+2))**NSQR*RHOA*R(2)
      GO TO 39
36    UBAR=FV*(U(KMM+1)+U(KMM+2))*RHOA*R(2)
      IF(UBAR)38,37,37
37    UR=UBAR*U(KMM+1)
      GO TO 39
38    UR=UBAR*U(KMM+2)
39    DO 46 K=2,KMAXZZ
      KM=K+KMM
      UL=UR
      IF(NGEN .EQ. 0) GO TO 41
40    UR=TFV*(U(KM)+U(KM+1))**NSQR*RHOA*R(K+1)
      GO TO 44
41    UBAR=FV*(U(KM)+U(KM+1))*RHOA*R(K+1)
      IF(UBAR)42,43,43
42    UR=UBAR*U(KM+1)
      GO TO 44
43    UR=UBAR*U(KM)
44    IF(((M(KM).AND.EMP).NE.EMP).AND.((M(KM+1).AND.EMP)
     *  .NE.EMP)) GO TO 45
      GO TO 46
45    ETA(KM)=ETA(KM)+RHOA*U(KM)
      ETA(KM)=ETA(KM)-DT*((UR-UL)/(DRB(K)*RB(K)))
46    CONTINUE
      DO 61 K=2,KMAXZZ
      VT=0.0
      DO 61 L=2,LMAXZ
      KT=K+L*KMAX
      KM=KT-KMAX
      KU=KM-KMAX
      VL=VR
      VBAR=FV*(V(KM)+V(KM+1))
      IF(((M(KM).AND.EMP).EQ.EMP).AND.((M(KM+1).AND.EMP)
     *  .EQ.EMP)) GO TO 47
      GO TO 50
```

```
47      IF(((M(KT).AND.EMP).NE.EMP).AND.((M(KT+1).AND.EMP)
     *  .NE.EMP)) GO TO 49
        VT=0.0
        GO TO 61
49      VT=RHOA*U(KT)*VBAR
        GO TO 61
50      IF(((M(KM).AND.EMP).NE.EMP).AND.((M(KM+1).AND.EMP)
     *  .NE.EMP)) GO TO 55
        IF(((M(KT).AND.EMP).NE.EMP).AND.((M(KT+1).AND.EMP)
     *  .NE.EMP)) GO TO 52
        GO TO 61
52      IF(VBAR)53,54,54
53      VT=RHOA*U(KT)*VBAR
        GO TO 61
54      VT=RHOA*VBAR*U(KM)
        GO TO 61
55      IF(VBAR)56,58,58
56      IF(((M(KT).AND.EMP).EQ.EMP).AND.((M(KT+1).AND.EMP)
     *  .EQ.EMP)) GO TO 58
        VT=RHOA*VBAR*U(KT)
        GO TO 59
58      VT=RHOA*VBAR*U(KM)
59      IF(((M(KB).AND.OUT).EQ.OUT).OR.((M(KT).AND.OUT)
     *  .EQ.OUT)) GO TO 61
60      ETA(KM)=ETA(KM)-DT*((VT-VB)/DZ)
61      CONTINUE
        DO 73 K=2,KMAXZ
        IF(NOEN.EQ.0) GO TO 63
62      VT=TFV*(V(K)+V(K+KMAX))**NSQR*RHOA
        GO TO 66
63      VBAR=FV*(V(K)+V(K+KMAX))*RHOA
        IF(VBAR)64,65,65
64      VT=VBAR*V(K+KMAX)
        GO TO 66
65      VT=VBAR*V(K)
66      DO 73 L=2,LMAXZZ
        KT=K+L*KMAX
        KM=KT-KMAX
        VB=VT
        IF(NOEN.EQ.0) GO TO 68
67      VT=TFV*(V(KM)+V(KT))**NSQR*RHOA
        GO TO 71
68      VBAR=FV*(V(KM)+V(KT))*RHOA
        IF(VBAR)69,70,70
69      VT=VBAR*V(KT)
        GO TO 71
70      VT=VBAR*V(KM)
71      IF(((M(KM).AND.EMP).NE.EMP).AND.((M(KT).AND.EMP)
     *  .NE.EMP)) GO TO 72
        GO TO 73
72      PS1(KM)=PS1(KM)+RHOA*V(KM)
        PS1(KM)=PS1(KM)-DT*((VT-VB)/DZB(L))
73      CONTINUE
        DO 86 L=2,LMAXZZ
        KMM=L*KMAX
        QR=0.0
        DO 86 K=2,KMAXZ
```

```
      KT=K+KMM
      KM=KT-KMAX
      UL=UR
      UBAR=FV*(U(KM)+U(KT))*RB(K)
      IF(((M(KM).AND.EMP).EQ.EMP).AND.((M(KT).AND.EMP)
     *  .EQ.EMP)) GO TO 74
      GO TO 77
74    IF(((M(KM+1).AND.EMP).NE.EMP).AND.((M(KT+1).AND.EMP)
     *  .NE.EMP)) GO TO 76
      UR=0.0
      GO TO 88
76    UR=RHOA*UBAR*V(KM+1)
      GO TO 88
77    IF(((M(KM).AND.EMP).NE.EMP).AND.((M(KT).AND.EMP)
     *  .NE.EMP)) GO TO 82
      IF(((M(KM+1).AND.EMP).NE.EMP).AND.((M(KT+1).AND.EMP)
     *  .NE.EMP)) GO TO 79
      GO TO 88
79    IF(UBAR) 80,81,81
80    UR=RHOA*UBAR*V(KM+1)
      GO TO 88
81    UR=RHOA*UBAR*V(KM)
      GO TO 88
82    IF(UBAR) 83,85,85
83    IF(((M(KM+1).AND.EMP).EQ.EMP).AND.((M(KT+1).AND.EMP)
     *  .EQ.EMP)) GO TO 85
84    UR=RHOA*UBAR*V(KM+1)
      GO TO 86
85    UR=RHOA*UBAR*V(KM)
86    IF(((M(KM+1).AND.OUT).EQ.OUT).OR.((M(KM-1).AND.OUT)
     *  .EQ.OUT)) GO TO 88
      PS1(KM)=PS1(KM)-DT*((UR-UL)/(DR*R(K)))
88    CONTINUE
      IF(MU) 100,106,89
89    DO 97 L=2,LMAXZ
      KTT=L*KMAX
      DO 97 K=2,KMAXZZ
      KT=K+KTT
      KM=KT-KMAX
      KB=KT-KMAX
      IF(((M(KM).AND.EMP).NE.EMP).AND.((M(KM+1).AND.EMP)
     *  .NE.EMP))GO TO 90
      GO TO 97
90    IF(((M(KB).AND.EMP).EQ.EMP).AND.((M(KB+1).AND.EMP)
     *  .EQ.EMP)) GO TO 91
      GO TO 92
91    UBOT=U(KM)
      GO TO 93
92    UBOT=U(KB)
93    IF(((M(KT).AND.EMP).EQ.EMP).AND.((M(KT+1).AND.EMP)
     *  .EQ.EMP)) GO TO 94
      GO TO 95
94    UTOP=U(KM)
      GO TO 96
95    UTOP=U(KT)
96    ETA(KM)=ETA(KM)+UM*((UTOP+UBOT-TWO*U(KM))/
     *  DZ**NSQR)
```

```
      ETA(KM)=ETA(KM)-UM*((V(KM+1)-V(KB+1)-V(KM)+V(KB))/
     *  (DRB(K)*DZ))
97    CONTINUE
      DO 105 K=2,KMAXZ
      DO 105 L=2,LMAXZZ
      KT=KBL*KMAX
      KB=KT-KMAX
      IF(((H(KM).AND.EMP).NE.EMP).AND.((H(KT).AND.EMP)
     *  .NE.EMP)) GO TO 98
      GO TO 105
98    IF(((H(KM-1).AND.EMP).EQ.EMP).AND.((H(KT-1).AND.EMP)
     *  .EQ.EMP)) GO TO 99
      GO TO 100
99    VLEFT=V(KM)
      GO TO 101
100   VLEFT=V(KM-1)
101   IF(((H(KM+1).AND.EMP).EQ.EMP).AND.((H(KT+1).AND.EMP)
     *  .EQ.EMP)) GO TO 102
      GO TO 103
102   VRGHT=V(KM)
      GO TO 104
103   VRGHT=V(KM+1)
104   TEMPR=(U(KT)-U(KM))/DZB(L)-(VRGHT-V(KM))/DRB(K)
      TEMPL=(U(KT-1)-U(KM-1))/DZB(L)-(V(KM)-VLEFT)/DRB(K-1)
      PSI(KM)=PSI(KM)-UM*((TEMPR*RB(K)-TEMPL*RB(K-1))
     *  /(DR*R(K)))
105   CONTINUE
106   CONTINUE
107   CONTINUE
      DO 204 K=2,KMAXZ
      PSI(K)=PSI(K+4*KMAX)-2.*PSI(K+3*KMAX)+2.*PSI(K+KMAX)
204   CONTINUE
      LPS=LPS2
      CALL PPES(LPS)
      DO 120 I=1,NGRID
      PMX1=AMAX1(ABS(P(I)),PMX1)
      UMX1=AMAX1(ABS(U(I)),UMX1)
120   VMX1=AMAX1(ABS(V(I)),VMX1)
      NCYCLE=NCYCLE+1
      TIME=TIME+DT
      IF(VMAX) 110,110,109
109   DT=EPS1/VMAX
110   CONTINUE
      IF(DT-DTMAX)112,112,111
111   DT=DTMAX
112   CONTINUE
      RELAX=BETA*RHOA*DR**2/DT
      IF(NEQ)60 TO 805
      IF(NSTP)GO TO 119
      IF(.NOT.NDMP)GO TO 115
      CALL ABSAV
      GO TO 117
115   IF(NCYCLE-NDMPT)118,116,116
116   CALL ABSAV
117   NDMP=.FALSE.
      NDMPT=NDMPT+NDUMP
118   IF(.NOT.NEDT)GO TO 123
```

```
      CALL EDIT
      NEDIT=.FALSE.
      GO TO 33
123   IF(NCYCLE-NEDTM)33,121,121
121   NEDTM=NEDTM+NEDIT
      CALL EDIT
      GO TO 33
119   CALL EDIT
      CALL ABSAV
      CALL EXIT
805   IF(NCYCLE-NEDTM)807,806,806
806   NEDTM=NEDTM+NEDIT
      TYPE 823,NCYCLE
823   FORMAT(1H ,'EDIT AT CYCLE ',I3)
      CALL EDIT
807   CALL ABSAV
      CALL EXIT
      RETURN
      END
```

```
      SUBROUTINE PRES(EPS)
      COMMON NVAC,NGEN,UAC,VAC,TM1,TM2,TM3
      COMMON NCYCLE,NSTOP,NEDTM,NEDIT,DPE
      COMMON TIML,EDTIM,EDDT,NPIT,NITER
      COMMON BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
      COMMON SMIN,EPS1,NBMAX,UBTO,VBTO,UBND,VBND
      COMMON UM,MU,RHOA,GR,GZ,G,ISEC
      COMMON VIN,UIN,DKP,DLP,NBP,DDR,DDZ
      COMMON NEXP,NEXPP,NPART,ND1V
      COMMON NGRID,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      COMMON KDMN,LDMN,KDMX,LDMX,MESS(7)
      COMMON ICRT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
      COMMON ITEST,IOZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
      COMMON NF,NG,NI,H,XG,HP,HV,GG,FUDGE,NTAL,CRAX,DELZ
      COMMON R(150),Z(150),RB(150),ZB(150)
      COMMON DRB(150),DZB(150)
      COMMON M(2500)
      COMMON P(2500),U(2500),ETA(2500),V(2500),PSI(2500)
      COMMON DNORX(2500),DNORY(2500),PNORX(2500),PNORY(2500)
      COMMON DR,DZ
      COMMON NDEV1,NDEV2
      COMMON/FLAG1/FLAG1
      COMMON/FLGS/TTY,NUMP,NEDT,NSTP,NPIM
      COMMON/STUFF/ERRORS,NAME,DIVS,VCHM,UCHM
      LOGICAL TTY,NUMP,NEDT,NSTP,NPIM
      REAL MU
      INTEGER AND,OR,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     * FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB
      DATA INTRF,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     * FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB/1,2,4,8,16,32,
     * 64,128,256,512,1024,2048,4096,8192,16384,32768/
      DATA MASK1,MASK2,MASK3,MASK4,MASK5,MASK6,
     * MASK7,MASK8/077777777757,077777773777,
     * 077777677777,077777777773,077777777767,
     * 077777737777,077777757777,077777767777/
      PI=3.1415926
      MAST=43012
      MPLS=43016
      LOGICAL TEST
      LOGICAL TA,TB,TC,TV,TM
      NEX=0
      NIL=0
      CALL VEL(2,NIL,NIL,NIL,1)
      MM=0
      NITER=0
      NITER=NITER+1
      ENORM=0.
      PNORM=0.
      DIVS1=0
      TEMPS1=0
      DO 44 L=2,LMAXZ
      KTT=L*KMAX
      DO 44 K=2,KMAXZ
      KP=K
      LP=L
      NT=KTT+KP
      NM=NT-KMAX
```

```
      NB=NM-KMAX
      IF(((M(NM).AND.FULL).EQ.FULL).AND.((M(NM).AND.OB).NE.OB))
     *  GO TO 4
      IF(M(NM).EQ.43012)GO TO 4
      IF(M(NM).EQ.43016)GO TO 4
      GO TO 5
4     DIV=(U(NM)*RB(K)-U(NM-1)*RB(K-1))/
     *  (R(K)*DR)
      DIV=DIV+(V(NM)-V(NB))/DZ
      GO TO 39
5     CONTINUE
      IF((M(NM).AND.OK).EQ.OK) GO TO 6
      GO TO 43
6     DLLR=DR
      DLLZ=DZ
      IF(PNORY(NM)-Z(LP)) 7,7,8
7     LB=LP-1
      GO TO 9
8     CONTINUE
      LB=LP
9     KT=LB*KMAX+KP
      KM=(LB-1)*KMAX+KP
      SX=(R(KP)-PNORX(NM))/DLLR
      SY=(Z(LB)-PNORY(NM))/DLLZ
      IF(U(KT-1))11,10,11
10    U1=U(NM-1)
      GO TO 12
11    U1=U(KT-1)
12    IF(U(KT) )14,13,14
13    U2=U(NM)
      GO TO 15
14    U2=U(KT)
15    IF(U(KM-1))17,16,17
16    U3=U(NM-1)
      GO TO 18
17    U3=U(KM-1)
18    IF(U(KM)) 20,19,20
19    U4=U(NM)
      GO TO 21
20    U4=U(KM)
21    UP=(.5+SX)*(.5-SY)*U1+(.5-SX)*(.5-SY)*U2
      UP=UP+(.5+SX)*(.5+SY)*U3+(.5-SX)*(.5+SY)*U4
      IF(PNORX(NM)-R(KP)) 22,22,23
22    KB=KP-1
      GO TO 24
23    KB=KP
24    KU=(LP-2)*KMAX+KB
      KM=(LP-1)*KMAX+KB
      SX=(RB(KB)-PNORX(NM))/DLLR
      SY=(Z(LP)-PNORY(NM))/DLLZ
      IF(V(KM)) 26,25,26
25    V1=V(NM)
      GO TO 27
26    V1=V(KM)
27    IF(V(KM+1)) 29,28,29
28    V2=V(NM)
      GO TO 30
```

```
29      V2=V(NM+1)
30      IF(V(ND)) 32,31,32
31      V3=V(NB)
        GO TO 33
32      V3=V(KD)
33      IF(V(KD+1)) 35,34,35
34      V4=V(NB)
        GO TO 36
35      V4=V(KD+1)
36      VP=(.5+SX)*(.5-SY)*V1+(.5-SX)*(.5-SY)*V2
        VP=VP+(.5+SX)*(.5+SY)*V3+(.5-SX)*(.5+SY)*V4
        DUT=DNORX(NM)*UP+DNORY(NM)*VP
38      DIV=DUT/DR
39      IF(ABS(DIV).LT.1.0E-20)DIV=0.0
        TEMP=RLLAX*DIV
        IF(DIVS1.GT.ABS(DIV))GO TO 7022
        DIVS1=ABS(DIV)
        KMLOC=K
        LMLOC=L
7022    P(NM)=P(NM)-TEMP
40      IF((M(NT).AND.OUT).EQ.OUT) GO TO 41
        GO TO 42
41      P(NM)=0.
        TEMP=0.
42      ENORM=ENORM+TEMP*TEMP
        PNORM=PNORM+P(NM)*P(NM)
43      CALL VEL(1,NM,K,L,0)
44      CONTINUE
        DO 45 L=2,LMAXZ
        KL=2+(L-1)*KMAX
        P(KL-1)=P(KL)-GR*RHOA*(R(2)-R(1))
45      CONTINUE
        DO 46 L=2,LMAXZZ
        KR=L*KMAX
        P(KR)=P(KR-1)+GR*RHOA*(R(KMAX)-R(KMAXZ))
46      CONTINUE
        DO 47 K=2,KMAXZ
        KT=K+(LMAXZ-1)*KMAX
        KTT=K+LMAXZ*KMAX
        KB=K+KMAX
        KBB=KB
        P(KTT)=P(KT)
        IF((M(K).AND.OUT).EQ.OUT)GO TO 47
        P(KBB)=P(KB)
47      CONTINUE
        CALL VEL(1,NIL,NIL,NIL,1)
        ERROP=SQRT(ENORM/(PNORM+.0000001))
        ERRORS=ERROR
        DIVS=DIVS1
        TYPE 57,NCYCLE
        TYPE 56,DIVS
        TYPE 7023,KMLOC,LMLOC
7023    FORMAT(1H ,'AT K=',I3,',L=',I3)
        TYPE 48,ERROR
57      FORMAT(1H ,'CYCLE ',I3)
56      FORMAT(1H ,'MAX DIVERGENCE=',F)
48      FORMAT(1H ,'ERROR=',F)
```

```
      IF(ERROR-EPS)49,55,55
      IF(.NOT.NPIM)GO TO 50
55    NPIM=.FALSE.
      RETURN
50    IF(NITER-NPIT)3,51,51
51    WRITE(1,57)NCYCLE
      WRITE(1,56)DIVS
      WRITE(1,48)ERROR
      END FILE 1
      CALL APPLE
49    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VEL(NA1,NA2,NA3,NA4,NA5)
      COMMON NVAC,NGEN,UAC,VAC,TM1,TM2,TM3
      COMMON NCYCLE,NSTOP,NEDTM,NEDIT,OPE
      COMMON TIME,EDTIM,EDDT,NPIT,NITER
      COMMON BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
      COMMON SMIN,EPS1,NBMAX,UBTO,VBTO,UBND,VBND
      COMMON UM,MU,RHOA,GR,GZ,G,ISEC
      COMMON V1N,UIN,DRP,DLP,NRP,DDR,DDZ
      COMMON NEXP,NEXPP,NPART,NDIV
      COMMON NGRID,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      COMMON KDMN,LDMN,KDMX,LDMX,MFSS(7)
      COMMON ICNT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
      COMMON ITEST,IUZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
      COMMON NF,NG,NIT,U,XG,HH,HV,GG,FUDGE,NTAL,CRAX,DELZ
      COMMON R(150),Z(150),RB(150),ZB(150)
      COMMON DRB(150),DZB(150)
      COMMON M(2500)
      COMMON P(2500),U(2500),ETA(2500),V(2500),PSI(2500)
      COMMON DNORX(2500),DNORY(2500),PNORX(2500),PNORY(2500)
      COMMON DR,DZ
      COMMON NDEV1,NDEV2
      COMMON/FLAG1/FLAG1
      COMMON/FLGS/TTY,NDMP,NEDT,NSTP,NPIM
      COMMON/STUFF/ERRORS,NAME,DIVS,VCHM,UCHM
      LOGICAL TTY,NDMP,NEDT,NSTP,NPIM
      REAL NU
      INTEGER AND,OR,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,UK,GAS,ARB
      DATA INTRF,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,UK,GAS,ARB/1,2,4,8,16,32,
     *  64,128,256,512,1024,2048,4096,8192,16384,32768/
      DATA MASK1,MASK2,MASK3,MASK4,MASK5,MASK6,
     *  MASK7,MASK8/077777777757,077777777377,
     *  077777677777,077777777773,077777777767,
     *  077777737777,077777757777,077777767777/
      PI=3.1415926
      NGIG=NA1
      KM=NA2
      K=NA3
      L=NA4
      KP=NA5
C
      FV=.5
C     THIS ROUTINE COMPUTES ALL OF THE CELL VELOCITIES.
C     THE PRESSURE GRADIENTS ARE CALCULATED FOR BOUNDARY
C     PRESSURES LOCATED AT THE MID POINTS OF THE LAGRANGE
C     BOUNDARY SEGMENTS. APPROXIMATE LINEAR INTERPOLATION
C     IS USED.
C
      LOGICAL TEST
      VMAX=0.0
      LMZ=LMAXZ
      KMZ=KMAXZ
      IF(KP.EQ.0)GO TO 73
      UCHM=0.
      VCHM=0.
      DO 39 L=2,LMZ
```

```
      KTT=L*KMAX
      DO 39 K=2,KMZ
      KT=K+KTT
      KM=KT-KMAX
      KB=KM-KMAX
      KTZ=KT+KMAX
      GO TO 74
73    KT=KM+KMAX
      KB=KM-KMAX
      KTZ=KT+KMAX
      KMZ=K
      LMZ=L
74    IF(((M(KM).AND.FULL).EQ.FULL).OR.(((M(KM).AND.SUR)
     *  .EQ.SUR).AND.((M(KM).AND.COR).NE.COR))) GO TO 1
      GO TO 39
C
C     U COMPONENT OF VELOCITY
C
1     IF((M(KM+1).AND.BND).EQ.BND) GO TO 20
2     IF(((M(KM+1).AND.EMP).EQ.EMP).OR.((M(KM+1).AND.COR)
     *  .EQ.COR)) GO TO 3
      GO TO 5
3     GO TO (20,4),NSIG
4     U(KM)=U(KM)-GR*DT
      GO TO 20
5     XC=R(K)
      IF(((M(KM).AND.OR).EQ.OR).AND.((M(KM-1).AND.COR)
     *  .EQ.COR)) GO TO 6
      GO TO 7
6     XC=PNORX(KM)
7     XA=R(K+1)
      IF(((M(KM+1).AND.OR).EQ.OR).AND.((M(KM+2).AND.COR)
     *  .EQ.COR)) GO TO 8
      GO TO 9
8     XA=PNORX(KM+1)
9     PA=P(KM)
      IF(((M(KM).AND.OR).EQ.OR).AND.((M(KT).AND.COR)
     *  .EQ.COR)) GO TO 10
      GO TO 11
10    P2=P(KM)
      P1=P(KB)
      Y2=PNORY(KM)
      Y1=Z(L-1)
      GO TO 13
11    IF(((M(KM).AND.OR).EQ.OR).AND.((M(KB).AND.COR)
     *  .EQ.COR)) GO TO 12
      GO TO 14
12    P2=P(KT)
      P1=P(KM)
      Y2=Z(L+1)
      Y1=PNORY(KM)
13    SLP=(P2-P1)/(Y2-Y1)
      ORD=P1-SLP*Y1
      XC=R(K)
      PA=SLP*Z(L)+ORD
14    PB=P(KM+1)
      IF(((M(KM+1).AND.OR).EQ.OR).AND.((M(KT+1).AND.COR)
```

```
     *   .EQ.COR)) GO TO 15
         GO TO 16
15       P2=P(KM+1)
         P1=P(KB+1)
         Y2=PNORY(KM+1)
         Y1=Z(L-1)
         GO TO 18
16       IF(((M(KM+1).AND.OK).EQ.OK).AND.((M(KB+1).AND.COR)
     *   .EQ.COR)) GO TO 17
         GO TO 19
17       P2=P(KT+1)
         P1=P(KM+1)
         Y2=Z(L+1)
         Y1=PNORY(KM+1)
18       SLP=(P2-P1)/(Y2-Y1)
         ORD=P1-SLP*Y1
         XA=R(K+1)
         PB=SLP*Z(L)+ORD
19       UOLD=U(KM)
         U(KM)=ETA(KM)+DT*((PA-PB)/(XA-XC))
         U(KM)=U(KM)/RHOA-GR*DT
         UCH=ABS(UOLD-U(KM))
         UCHM=AMAX1(UCH,UCHM)
C
C        V COMPONENT OF VELOCITY
C
20       IF((M(KT).AND.BND).EQ.BND) GO TO 39
21       IF(((M(KT).AND.EMP).EQ.EMP).OR.((M(KT).AND.COR)
     *   .EQ.COR)) GO TO 22
         GO TO 24
22       GO TO (39,23),NSIG
23       V(KM)=V(KM)-GZ*DT
         GO TO 39
24       YC=Z(L)
         IF(((M(KM).AND.OK).EQ.OK).AND.((M(KB).AND.COR)
     *   .EQ.COR)) GO TO 25
         GO TO 26
25       YC=PNORY(KM)
26       YA=Z(L+1)
         IF(((M(KT).AND.OK).EQ.OK).AND.((M(KTZ).AND.COR)
     *   .EQ.COR)) GO TO 27
         GO TO 28
27       YA=PNORY(KT)
28       PA=P(KM)
         IF(((M(KM).AND.OK).EQ.OK).AND.((M(KM+1).AND.COR)
     *   .EQ.COR)) GO TO 29
         GO TO 30
29       P2=P(KM)
         P1=P(KM-1)
         X2=PNORX(KM)
         X1=R(K-1)
         GO TO 32
30       IF(((M(KM).AND.OK).EQ.OK).AND.((M(KM-1).AND.COR)
     *   .EQ.COR)) GO TO 31
         GO TO 33
31       P2=P(KM+1)
         P1=P(KM)
```

```
      X1=RNORX(KM)
      X2=R(K+1)
32    SLP=(P2-P1)/(X2-X1)
      ORD=P1-SLP*X1
      YC=Z(L)
      PA=SLP*R(K)+ORD
33    P2=P(KT)
      IF(((M(KT).AND.OR).EQ.OR).AND.((M(KT+1).AND.COR)
     *  .EQ.COR)) GO TO 34
      GO TO 35
34    P2=P(KT)
      P1=P(KT-1)
      X2=RNORX(KT)
      X1=R(K-1)
      GO TO 37
35    IF(((M(KT).AND.OR).EQ.OR).AND.((M(KT-1).AND.COR)
     *  .EQ.COR)) GO TO 36
      GO TO 38
36    P2=P(KT+1)
      P1=P(KT)
      X2=R(K+1)
      X1=RNORX(KT)
37    SLP=(P2-P1)/(X2-X1)
      ORD=P1-SLP*X1
      YA=Z(L+1)
      PB=SLP*R(K)+ORD
38    VOLD=V(KM)
      V(KM)=RS1(KM)+DT*((PA-PB)/(YA-YC))
      V(KM)=V(KM)/RHOA-GZ*DT
      VCH=ABS(VOLD-V(KM))
      VCHM=AMAX1(VCH,VCHM)
      UMAX=ABS(U(KM))+ABS(V(KM))
      VMAX=FV*(ABS(VMAX-UMAX)+VMAX+UMAX)
39    CONTINUE
C
C     USE NOSLIP BOUNDARY CONDITIONS ON OB CELLS...
C
      IF(KP.EQ.0)GO TO 75
      CALL NOSLIP(NIL,NIL,NIL)
      GO TO 76
75    CALL NOSLIP(KM,K,L)
76    CONTINUE
C
C     BOUNDARY CONDITIONS ON VELOCITIES AT EDGE OF MESH.
C
      DO 60 K=2,KMAX2
      KT=K+(LMAX2-1)*KMAX
      KTB=KT-KMAX
      KTT=KT+KMAX
      KB=K
      KBT=KB+KMAX
      IF((M(KTT).AND.IN).EQ.IN) GO TO 49
      GO TO 50
49    V(KT)=VIN
      V(KTT)=V(KT)
      U(KTT)=0.
      GO TO 54
```

```
50      IF((M(KT1).AND.OUT).EQ.OUT) GO TO 51
        GO TO 52
51      V(KT)=V(KTB)
        U(KT1)=U(KT)
        V(KTT)=V(KTB)
        GO TO 54
52      IF((M(KTT).AND.BND).EQ.BND) GO TO 53
        GO TO 54
53      V(KT)=0.
        U(KTT)=U(KT)
        V(KTT)=-V(KTB)
54      IF((M(KB).AND.IN).EQ.IN) GO TO 55
        GO TO 56
55      V(KB)=VIN
        U(KB)=0.
        GO TO 60
56      IF((M(KB).AND.OUT).EQ.OUT) GO TO 57
        GO TO 58
57      V(K)=(P51(K)-DT*P(KBT)/DZB(1))/RHOA
        U(K)=U(KBT)
        GO TO 60
58      IF((M(KB).AND.BND).EQ.BND) GO TO 59
        GO TO 60
59      V(KB)=0.
        U(KB)=U(KBT)
60      CONTINUE
        DO 72 L=1,LMAX
        KK=(L-1)*KMAX
        LR=KK+KMAX2
        LL=2+KK
        IF((M(LL-1).AND.IN).EQ.IN) GO TO 61
        GO TO 62
61      U(LL-1)=UIN
        V(LL-1)=0.
        GO TO 66
62      IF((M(LL-1).AND.OUT).EQ.OUT) GO TO 63
        GO TO 64
63      U(LL-1)=U(LL)
        V(LL-1)=V(LL)
        GO TO 66
64      IF((M(LL-1).AND.BND).EQ.BND) GO TO 65
        GO TO 66
65      U(LL-1)=0.
        V(LL-1)=V(LL)
        GO TO 72
66      IF((M(LR+1).AND.IN).EQ.IN) GO TO 67
        GO TO 68
67      U(LR)=UIN
        U(LR+1)=U(LR)
        V(LR+1)=V(LR)
        GO TO 72
68      IF((M(LR+1).AND.OUT).EQ.OUT) GO TO 69
        GO TO 70
69      U(LR)=U(LR-1)
        U(LR+1)=U(LR)
        V(LR+1)=V(LR)
        GO TO 72
```

```
70      IF((M(LR+1).AND.BND).EQ.BND) GO TO 71
        GO TO 72
71      U(LR)=0.
        U(LR+1)=-U(LR-1)
        V(LR+1)=V(LR)
72      CONTINUE
        RETURN
        END
```

```
      SUBROUTINE VEL55(NA6,NA7,NA8)
      COMMON NVAC,NGEN,UAC,VAC,TM1,TM2,TM3
      COMMON NCYCLE,NSTOP,NEDTM,NEDIT,OPE
      COMMON TIME,EDTIM,EDDT,NPIT,NITER
      COMMON BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
      COMMON SMIN,EPS1,NBMAX,UBTO,VBTO,UBND,VBND
      COMMON UM,MU,RHOA,GR,GZ,G,1SEC
      COMMON VIN,UIN,DKP,DLP,NBP,DDR,DDZ
      COMMON NEXP,NEXPP,NPART,NDIV
      COMMON NGRID,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      COMMON KDMN,LDMN,KDMX,LDMX,MESS(7)
      COMMON ICYT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
      COMMON ITEST,IOZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
      COMMON NF,NG,NIT,N,XG,NP,NV,GG,FUDGE,NTAL,CRAX,DELZ
      COMMON R(150),Z(150),RB(150),ZB(150)
      COMMON DRB(150),DZB(150)
      COMMON M(2500)
      COMMON P(2500),U(2500),ETA(2500),V(2500),PSI(2500)
      COMMON DNORX(2500),DNORY(2500),PNORX(2500),PNORY(2500)
      COMMON DR,DZ
      COMMON NDEV1,NDEV2
      COMMON/FLAG1/FLAG1
      COMMON/FLGS/TTY,NDMP,NEDT,NSTP,NPIT
      LOGICAL TTY,NEDT,NDMP,NSTP,NPIT
      REAL MU
      INTEGER AND,OR,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB
      DATA INTRF,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB/1,2,4,8,16,32,
     *  64,128,256,512,1024,2048,4096,8192,16384,32768/
      DATA MASK1,MASK2,MASK3,MASK4,MASK5,MASK6,
     *  MASK7,MASK8/077777777757,077777773777,
     *  077777677777,077777777773,077777777767,
     *  077777737777,077777757777,077777767777/
      PI=3.1415926
      KM=NA6
      K=NA7
      L=NA8
      LOGICAL TEST
      LMZ=LMAXZ
      KMZ=KMAXZ
      IF(KM.NE.0)GO TO 10
      DO 1 L=2,LMZ
      KMM=L*KMAX
      DO 1 K=2,KMZ
      KT=K+KMM
      KM=KT-KMAX
      KB=KM-KMAX
      GO TO 11
10    KT=KM+KMAX
      KB=KM-KMAX
      KMZ=K
      LMZ=L
11    CONTINUE
      IF(((M(KM).AND.SUR).EQ.SUR).OR.
     *  ((M(KM).AND.OK).EQ.OK)) GO TO 2
      GO TO 1
```

```
2       IF((.NOT.(((M(KM+1).AND.EMP).EQ.EMP).OR.
     *  ((M(KT).AND.EMP).EQ.EMP).OR.((M(KB).AND.EMP).EQ.EMP)))
     *  .AND.(((M(KM-1).AND.EMP).EQ.EMP)
     *  .OR.((M(KM-1).AND.COR).EQ.COR)).AND.((M(KM-1).AND.BND).NE.BND)
     *  .AND.(.NOT.(((M(KM+1).AND.COR).EQ.COR).OR.((M(KT).AND.
     *  COR).EQ.COR).OR.((M(KB).AND.COR).EQ.COR)))) GO TO 3
        GO TO 4
3       U(KM-1)=R(K)*DR*((V(KM)-V(KB))/(RB(K-1)*DZ))
        U(KM-1)=U(KM-1)+U(KM)*(RB(K)/RB(K-1))
        GO TO 1
4       IF((.NOT.(((M(KM-1).AND.EMP).EQ.EMP).OR.
     *  ((M(KT).AND.EMP).EQ.EMP).OR.((M(KB).AND.EMP).EQ.EMP))).AND.
     *  (((M(KM+1).AND.EMP).EQ.EMP).OR.((M(KM+1).AND.COR).EQ.COR))
     *  .AND.((M(KM+1).AND.BND).NE.BND).AND.(.NOT.(((
     *  M(KM-1).AND.COR).EQ.COR).OR.((M(KT).AND.COR).EQ.COR)
     *  .OR.((M(KB).AND.COR).EQ.COR)))) GO TO 5
        GO TO 6
5       U(KM)=U(KM-1)*(RB(K-1)/RB(K))
        U(KM)=U(KM)-R(K)*DR*((V(KM)-V(KB))/(RB(K)*DZ))
        GO TO 1
6       IF((.NOT.(((M(KM-1).AND.EMP).EQ.EMP).OR.((M(KM+1).AND.EMP)
     *  .EQ.EMP).OR.((M(KT).AND.EMP).EQ.EMP))).AND.(((M(KB).AND.
     *  EMP).EQ.EMP).OR.((M(KB).AND.COR).EQ.COR)).AND.
     *  ((M(KB).AND.BND).NE.BND).AND.(.NOT.(((M(KM-1).AND.COR)
     *  .EQ.COR).OR.((M(KM+1).AND.COR).EQ.COR).OR.
     *  ((M(KT).AND.COR).EQ.COR)))) GO TO 7
        GO TO 8
7       V(KB)=V(KM)+DZ*((RB(K)*U(KM)-RB(K-1)*U(KM-1))
     *  /(R(K)*DR))
        GO TO 1
8       IF((.NOT.(((M(KM-1).AND.EMP).EQ.EMP).OR.((M(KM+1).AND.EMP).EQ.
     *  .OR.((M(KB).AND.EMP).EQ.EMP))).AND.(((M(KT).AND.EMP).EQ.EMP)
     *  .OR.((M(KT).AND.COR).EQ.COR)).AND.((M(KT).AND.BND).NE.BND)
     *  .AND.(.NOT.(((M(KM-1).AND.COR).EQ.COR).OR.((M(KM+1).AND.
     *  COR).EQ.COR).OR.((M(KB).AND.COR).EQ.COR)))) GO TO 9
        GO TO 1
9       V(KM)=V(KB)-DZ*((RB(K)*U(KM)-RB(K-1)*U(KM-1))
     *  /(R(K)*DR))
1       CONTINUE
        RETURN
        END
```

```
      SUBROUTINE NOSLIP(NA6,NA7,NA8)
      COMMON NVAC,NGEN,UAC,VAC,TM1,TM2,TM3
      COMMON NCYCLE,NSTOP,NEDTM,NEDIT,OPE
      COMMON TIME,EDTIM,EDDT,NPIT,NITER
      COMMON BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
      COMMON SMIN,EPS1,NBMAX,UBTO,VBTO,UBND,VBND
      COMMON UM,MU,RHOA,GR,GZ,G,ISEC
      COMMON VIN,UIN,DKP,DLP,NBP,DDR,DDZ
      COMMON NEXP,NEXPP,NPART,NDIV
      COMMON NGRID,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      COMMON KDMN,LDMN,KDMX,LDMX,MESS(7)
      COMMON ICRT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
      COMMON ITEST,IOZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
      COMMON NF,NG,NIT,H,XG,HH,HV,GG,FUDGE,NTAL,CRAX,DELZ
      COMMON R(150),Z(150),RB(150),ZB(150)
      COMMON DRB(150),DZB(150)
      COMMON M(2500)
      COMMON P(2500),U(2500),ETA(2500),V(2500),PSI(2500)
      COMMON DNORX(2500),DNORY(2500),PNORX(2500),PNORY(2500)
      COMMON DR,DZ
      COMMON NDEV1,NDEV2
      COMMON/FLAG1/FLAG1
      COMMON/MBTN/MBTN
      COMMON/FLGS/TTY,NDMP,NEDT,NSTP,NPIM
      COMMON/STUFF/ERRORS,NAME,DIVS,VCHM,UCHM
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/LOOK/PMX,UMX,VMX,PMX1,UMX1,VMX1
      COMMON/NX/DTMAX
      LOGICAL TTY,NDMP,NEDT,NSTP,NPIM
      LOGICAL NFIB
      REAL MU
      INTEGER AND,OR,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB
      DATA INTRF,CNTRB,SUR,FULL,EMP,BND,IN,OUT,
     *  FRSLP,NOSLP,EMPBND,OB,COR,OK,GAS,ARB/1,2,4,8,16,32,
     *  64,128,256,512,1024,2048,4096,8192,16384,32768/
      DATA MASK1,MASK2,MASK3,MASK4,MASK5,MASK6,
     *  MASK7,MASK8/07777777777757,07777777773777,
     *  07777776777777,07777777777773,07777777777767,
     *  07777777737777,07777777757777,07777777677777/
      INTEGER ES,EC
      DH=DR/2.
      EC=4112
      ES=4100
      KM=NA6
      K=NA7
      L=NA8
      IF(KM.NE.0)GO TO 95
      LMZ=LMAXZ
      KMZ=KMAXZ
      DO 100 L=2,LMZ
      KTT=(L-1)*KMAX
      DO 100 K=2,KMZ
      KM=KTT+K
      GO TO 96
95    LMZ=L
      KMZ=K
```

```
96      IF(M(KM+1).EQ.32.AND.M(KM).EQ.8)GO TO 98
        GO TO 99
98      U(KM)=0.
        V(KM+1)=-V(KM)
        GO TO 100
99      IF((M(KM).AND.OB).NE.OB)GO TO 100
        TAN=SIGN(DNORX(KM)/DNORY(KM),-DNORX(KM)*DNORY(KM))
        IF(DNORY(KM).EQ.0.0)TAN=10**20
        KT=KM+KMAX
        KB=KM-KMAX
        IF(M(KM+1).EQ.EC)GO TO 101
        IF(M(KM-1).EQ.EC)GO TO 102
        IF(M(KT).EQ.EC)GO TO 103
        IF(M(KB).EQ.EC)GO TO 104
        IF(M(KM+1).EQ.SC)GO TO 101
101     XU=(ZB(L)-PNORY(KM))/TAN+PNORX(KM)
        IF(M(KT).EQ.EC)GO TO 110
        IF(M(KB).EQ.EC)GO TO 111
        AL1=RB(K)-PNORX(KM)
        AL2=DR-AL1
        U(KM)=-(U(KM-1)*AL1)/AL2
        IF(TAN.GE.10**10)GO TO 112
        IF(((M(KT).AND.FULL).EQ.FULL).AND.(XU.GT.RB(K)))GO TO 114
        AL1=R(K)-XU
        AL2=DR-AL1
        V(KM)=-(V(KM-1)*AL1)/AL2
        GO TO 100
112     V(KM)=0.0
        GO TO 100
110     IF(ABS(TAN).LT.0.25)GO TO 103
        Y1=TAN*(R(K)-PNORX(KM))+PNORY(KM)
        YL=Y1-Z(L)
        XL=YL/TAN
        AL1=DH-XL
        AL2=DH+XL
        U(KM)=-(U(KM-1)*AL1)/AL2
        AL1=DH-YL
        AL2=DH+YL
        V(KM)=-(V(KB)*AL1)/AL2
        GO TO 100
111     YL=TAN*(R(K)-PNORX(KM))+PNORY(KM)
        YL=Y1-Z(L)
        XL=YL/TAN
        AL1=DH+XL
        AL2=DH-XL
        U(KM)=-(U(KM-1)*AL1)/AL2
        IF(((M(KT).AND.OB).EQ.OB).AND.((M(KM-1).AND.FULL)
     *  .EQ.FULL))GO TO 115
        IF(((M(KM-1).AND.OB).EQ.OB).AND.((M(KT).AND.FULL)
     *.EQ.FULL))GO TO 116
        AL1=DH+YL
        AL2=DH-YL
        V(KB)=-(V(KM)*AL1)/AL2
        GO TO 100
114     AL1=R(K+1)-XU
        AL2=DR-AL1
        V(KM+1)=-(V(KM)*AL1)/AL2
```

```
      GO TO 100
115   AL1=R(K)-XU
      AL2=DR-AL1
      V(KM)=-(V(KM-1)*AL1)/AL2
      GO TO 100
116   YR=TAN*(RB(K)-PNORX(KM))+PNORY(KM)
      AL1=YR-Z(L)
      AL2=DZ-AL1
      U(KM)=-(U(KT)*AL1)/AL2
      GO TO 100
102   XU=(ZB(L)-PNORY(KM))/TAN+PNORX(KM)
      IF(M(KB).EQ.EC)GO TO 120
      IF(M(KT).EQ.EC)GO TO 121
      AL1=PNORX(KM)-RB(K-1)
      AL2=DR-AL1
      U(KM-1)=-(U(KM)*AL1)/AL2
      AL1=XU-R(K)
      AL2=DR-AL1
      V(KM)=-(V(KM+1)*AL1)/AL2
      GO TO 100
120   Y1=TAN*(R(K)-PNORX(KM))+PNORY(KM)
      YL=Z(L)-Y1
      XL=YL/TAN
      AL1=DH+XL
      AL2=DH-XL
      U(KM-1)=-(U(KM)*AL1)/AL2
      AL1=DH-YL
      AL2=DH+YL
      V(KB)=-(V(KM)*AL1)/AL2
      GO TO 100
121   Y1=TAN*(R(K)-PNORX(KM))+PNORY(KM)
      YL=Z(L)-Y1
      XL=YL/TAN
      AL1=DH-XL
      AL2=DH+XL
      U(KM-1)=-(U(KM)*AL1)/AL2
      AL1=DH-YL
      AL2=DH+YL
      V(KM)=-(V(KB)*AL1)/AL2
      GO TO 100
103   YR=TAN*(RB(K)-PNORX(KM))+PNORY(KM)
      AL1=ZB(L)-PNORY(KM)
      AL2=DZ-AL1
      V(KM)=-(V(KB)*AL1)/AL2
      AL1=Z(L)-YR
      AL2=DZ-AL1
      U(KM)=-(U(KB)*AL1)/AL2
      GO TO 100
104   YR=TAN*(RB(K)-PNORX(KM))+PNORY(KM)
      AL1=PNORY(KM)-ZB(L-1)
      AL2=DZ-AL1
      V(KB)=-(V(KM)*AL1)/AL2
      AL1=YR-Z(L)
      AL2=DZ-AL1
      U(KM)=-(U(KT)*AL1)/AL2
      GO TO 100
100   CONTINUE
```

END

APPENDIX IV

SPY PROGRAM

Routine Name	Function
EXAMIN	Does display and interaction for Spy Program
SHARE	Sets up core-sharing process

```
      SUBROUTINE EXAMIN
      COMMON NVAC,NGEN,UAC,VAC,TM1,TM2,TM3
      COMMON NCYCLE,NSTOP,NEDTM,NEDIT,OPE
      COMMON TIME,EDTIM,EDDT,NPIT,NITER
      COMMON BETA,EPS2,DT,VMAX,UMAX,RELAX,CUT
      COMMON SRIN,EPSI,NBMAX,UBTO,VBTO,UBND,VBND
      COMMON UM,MU,RHOA,GR,GZ,G,ISEC
      COMMON VIN,UIN,DKP,DLP,NBP,DDR,DDZ
      COMMON NEXP,NEXPP,NPART,NDIV
      COMMON NGRID,KMAX,LMAX,KMAXZ,LMAXZ,KMAXZZ,LMAXZZ
      COMMON KDMN,LDMN,KDMX,LDMX,MESS(7)
      COMMON ICRT,XMAX,YMAX,CRTRA,CRTRB,CRTZA,CRTZB
      COMMON ITEST,IOZ,IZ,GAMA,PGAS,PGASZ,PAMB,GASV,GASVZ
      COMMON NF,NG,NIT,H,XG,HH,HV,GG,FUDGE,NTAL,CRAX,DELZ
      COMMON R(150),Z(150),RB(150),ZB(150)
      COMMON DRB(150),DZB(150)
      COMMON M(2500)
      COMMON P(2500),U(2500),ETA(2500),V(2500),PSI(2500)
      COMMON DNORX(2500),DNORY(2500),PNORX(2500),PNORY(2500)
      COMMON DR,DZ
      COMMON NDEV1,NDEV2
      COMMON/FLAG1/FLAG1
      COMMON/FLGS/TTY,NDMP,NEDT,NSTP,NPIM
      COMMON/STUFF/ERRORS,NAME,DIVS,PNORMS,TEMPS
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/LOOK/PMX,UMX,VMX,PMX1,UMX1,VMX1
      DIMENSION IP1(70),IP2(70),IP3(70)
      LOGICAL TTY,NDMP,NEDT,NSTP,NPIM
      CALL FORK
      CALL SETDIS
103   NIT=NITER/60*60
      NPLOTS=NITER
      NPLT1=NITER
1     CALL MOVETO(256,928)
      CALL WRITE('PREVIOUS CYCLE<>')
      CALL MOVETO(192,896)
      CALL WRITE('MAX PRESSURE=ΔΔΔΔΔ.ΔΔΔ<>',PMX)
      CALL MOVETO(192,864)
      CALL WRITE('MAX V-VELOCITY=ΔΔΔΔ.ΔΔΔ<>',VMX)
      CALL MOVETO(192,832)
      CALL WRITE('MAX U-VELOCITY=ΔΔΔΔ.ΔΔΔ<>',UMX)
      CALL MOVETO(256,768)
      CALL WRITE('THIS CYCLE<>')
      CALL MOVETO(192,736)
      CALL WRITE('MAX PRESSURE=ΔΔΔΔΔ.ΔΔΔ<>',PMX1)
      CALL MOVETO(192,704)
      CALL WRITE('MAX V-VELOCITY=ΔΔΔΔ.ΔΔΔ<>',VMX1)
      CALL MOVETO(192,672)
      CALL WRITE('MAX U-VELOCITY=ΔΔΔΔ.ΔΔΔ<>',UMX1)
      D=ALOG10(DIVS)
      IB=D+1
      D=ALOG10(ERRORS)
      ID=D+1
      CALL MOVETO(576,944)
      CALL WRITE('10<>')
      CALL MOVE(16,20)
      CALL WRITE('ΔΔ<>',IB)
```

```
      DO 5 J=1,4
      JPLT=544+J*100
      CALL MOVETO(592,JPLT)
      CALL VEC(32,0)
      CALL MOVETO(576,464)
      CALL WRITE('10<>')
      CALL MOVE(16,20)
      CALL WRITE('ΔΔ<>',ID)
      DO 5 J=1,4
      JPLT=64+J*100
      CALL MOVETO(592,JPLT)
      CALL VEC(32,0)
      CALL MOVETO(608,944)
      CALL VEC(0,-400)
      CALL VEC(352,0)
      CALL MOVETO(608,464)
      CALL VEC(0,-400)
      CALL VEC(352,0)
      CALL MOVETO(704,512)
      CALL WRITE('MAX DIVERGENCE<>')
      CALL MOVETO(704,32)
      CALL WRITE('ENORM/PNORM<>')
      CALL BOXES
      CALL MOVETO(186,56)
      CALL WRITE('SELECT<>')
      CALL MOVETO(170,152)
      CALL WRITE('DT=Δ.ΔΔΔΔΔΔ<>',DT)
      CALL MOVETO(170,248)
      CALL WRITE('NPIT<>')
      CALL WRITE(' =ΔΔΔ<>',NPIT)
      CALL MOVETO(128,344)
      CALL WRITE('BETA=ΔΔ.ΔΔΔΔΔ<>',BETA)
      CALL MOVETO(128,440)
      CALL WRITE('EPS=Δ.ΔΔΔΔΔΔΔΔ<>',EPS2)
      CALL MOVETO(427,480)
      CALL WRITE('CYCLE ΔΔΔΔ<>',NCYCLE)
      CALL SEND
      CALL MOVETO(430,448)
      CALL WRITE('ITER ΔΔΔΔ<>',NITER)
      CALL APND
      ITLPT=0
      IF(NITER.EQ.ITPLT)GO TO 7
      CALL DELETE
   60 ITERATIONS CAN BE DISPLAYED ON SCOPE.....
      IF((NITER-NIT).GT.60)GO TO 103
      IY=544+(ALOG10(DIVS)-ID+4)*100
      IF(IY.GT.1000)IY=1000
      IF(IY.LT.544)IY=544
      IX=608+(NITER-NIT)*5
      IP2(NPLOTS-NIT)=IY
      CALL MOVETO(IX,IY)
      CALL VECTO(IX,544)
      IY=64+(ALOG10(ERRORS)-ID+4)*100
      IF(IY.LT.64)IY=64
      IP3(NPLOTS-NIT)=IY
      CALL MOVETO(IX,IY)
      CALL VECTO(IX,64)
```

```
      CALL APND
      CALL MOVETO(450,448)
      CALL WRITE('ITER ΔΔΔΔ<>',NITER)
      CALL APND
      NPLOTS=NPLOTS+1
      ITPLT=NITER
      IF(NITER.EQ.1)GO TO 103
7     CALL MSEINT(MBT)
      IF(MBT.EQ.1)GO TO 8
      CALL SLEEP
      GO TO 6
8     CALL MOUSXY
      INY=(INY+96)/96
      GO TO (16,9,11,13,14),INY
9     TYPE 25
25    FORMAT(1H ,'DT=',$)
      ACCEPT 10,DT
10    FORMAT(F)
      GO TO 101
11    TYPE 12
12    FORMAT('NPIT=',$)
      ACCEPT 28,NPIT
28    FORMAT(I)
      GO TO 101
13    TYPE 24
24    FORMAT(1H ,'BETA=',$)
      ACCEPT 10,BETA
      RELAX=BETA*RHOA*DR**2/DT
      GO TO 101
14    TYPE 15
15    FORMAT(1H ,'EPS=',$)
      ACCEPT 10,EPS2
      GO TO 101
100   NIT=ITPLT+70
      NPLOTS=1
101   DO 102 I=NPLTI-NIT,NPLOTS-NIT-1
      IX=608+I*5
      IY=IP2(I)
      CALL MOVETO(IX,IY)
      CALL VECTO(IX,544)
      IY=IP3(I)
      CALL MOVETO(IX,IY)
102   CALL VECTO(IX,64)
      GO TO 1
16    CALL CLEAR
      CALL BOXES
      CALL MOVETO(186,56)
      CALL WRITE('STOP<>')
      CALL MOVETO(186,152)
      CALL WRITE('CYCLE<>')
      CALL MOVETO(186,248)
      CALL WRITE('PRINT<>')
      CALL MOVETO(186,344)
      CALL WRITE('DUMP<>')
      CALL MOVETO(186,440)
      CALL WRITE('EXIT<>')
      CALL SEND
```

```
      CALL WHERE(IX,IY)
      INY=(INY+96)/96
      GO TO (17,18,19,20,21),INY
17    NSTP=.TRUE.
      CALL EXIT
18    NPIM=.TRUE.
      GO TO 101
19    NEDT=.TRUE.
      GO TO 101
20    NDMP=.TRUE.
      GO TO 101
23    FORMAT(I)
21    TYPE 26
26    FORMAT(1H ,'NDMP=',$)
      ACCEPT 23,NDMP1
      IF(NDMP1.NE.0)NDMP=NDMP1
      CALL EXIT
      END
      SUBROUTINE BOXES
      DO 1 I=1,5
      JPLT=(I-1)*96+32
      CALL MOVETO(96,JPLT)
1     CALL BOX
      RETURN
      END
      SUBROUTINE BOX
      CALL VEC(224,0)
      CALL VEC(0,64)
      CALL VEC(-224,0)
      CALL VEC(0,-64)
      RETURN
      END
```

```
TITLE SHARE
        ENTRY FILE
        ENTRY FORK
FILE:   0
        HRROI   2,FNAME
        MOVSI   1,(1B1!1B17)
        GTJFN
        HALTF
        HRRZS   1
        MOVEM   1,FILJFN#
        MOVE    2,[4485!1B19!1B20!1B21!1B25!1B26]
        OPENF
        HALTF
        SETZ    5,
        MOVSI   3,(1B2!1B3!1B4)
        MOVSI   1,400000
FMAP:   HRRI    1,0(5)
        RPACS
        TLNN    2,(1B5)
        JRST    DONE
        MOVS    2,FILJFN
        HRRI    2,0(5)
        PMAP
        AOJA    5,FMAP
DONE:   MOVSI   1,(1B0)
        HRRI    1,FILJFN
        CLOSF
        JRA     16,0(16)
FORK:   0
        HRROI   2,FNAME
        MOVSI   1,(1B2!1B17)
        GTJFN
        HALTF
        HRRZS   1
        MOVEM   1,FILJFN
        MOVE    2,[4485!1B19!1B20!1B21!1B25!1B26!]
        OPENF
        HALTF
        SETZ    5,
        MOVSI   3,(1B2!1B3!1B4)
        MOVS    1,FILJFN
FMAP1:  HRRI    1,0(5)
        RPACS
        TLNN    2,(1B5)
        JRST    DONE1
        MOVSI   2,400000
        HRRI    2,0(5)
        PMAP
        AOJA    5,FMAP1
DONE1:  MOVSI   1,(1B0)
        HRRI    1,FILJFN
        CLOSF
        JRA     16,0(16)
FNAME:  ASCIZ/EXAM+FILE.SHARE/
        END
```

APPENDIX V

OUTPUT PROGRAM

Routine Name	Function
MAIN	Read in data, compute A, BNDRY, and SI matrices. Controls other output functions.
CONTRL	Displays output functions
VPLOT	Draws velocity profiles
VVECT	Draws velocity vectors
CNTRL1	Displays axes for velocity profile
CNTRL2	Displays choices for contour plots
CONTOR	Contour plotting program

```
      INTEGER BLN(2),BNDRY
      DIMENSION UU(18,92),VV(18,92),PP(18,92),SI(18,92)
      DIMENSION BNDRY(18,92)
      DIMENSION A(18,92),U(2500),V(2500),P(2500)
      DIMENSION MESS(7)
      DIMENSION BPX(100),BPY(100)
      DIMENSION XB(100),YB(100),NSEQ(100)
      DIMENSION XBP(50,2),YBP(50,2)
      DIMENSION M(2500)
      DIMENSION R(150),Z(150),RB(150),ZB(150)
      DIMENSION XBB(50,2),YBB(50,2)
      DIMENSION ME(18,92)
      DIMENSION DS(300)
      DIMENSION PTX(300),PTY(300),ZNP(300),ZMP(300),RMP(300),RNP(3
      EQUIVALENCE (ME,M)
      EQUIVALENCE(XBB,XBP),(YBB,YBP)
      EQUIVALENCE(U,UU),(V,VV),(P,PP)
      COMMON/CLP/IN,IXP,IYP,MN
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/STUFF/DR,DZ,KMAX,LMAX,KDMN,LDMN,KDMX,LDMX,EPS1,DT
      COMMON/WINDOW/WS,WCX,WCY,XL
      COMMON/CONLVL/NCLS,NCL
      COMMON/BCON/BLN,XBP,YBP
      INTEGER EMP,BND,FULL,OB
      DATA EMP,BND,FULL,OB/16,32,8,2048/
      WS=8.0
      WCX=8.0
      NYS='N'
      WCY=8.0
105   TYPE 1001
1001  FORMAT(1H ,'DISPLAY=',$)
      ACCEPT 101,IDI
101   FORMAT(A4)
      TYPE 102
102   FORMAT(1H ,'INPUT FILE=',$)
      ACCEPT 103,NAME
103   FORMAT(A5)
      IF(IDI.EQ.'ARDS')IDI=1
      IF(IDI.EQ.'1559')IDI=2
      NAME=NAME+32
      CALL IFILE(22,NAME)
      READ(22,1)(MESS(I),I=1,7)
1     FORMAT(7A5)
      READ(22,2)KMAX,LMAX
2     FORMAT(2I)
      READ(22,3)SMIN,EPS1
3     FORMAT(2F)
      READ(22,4)U,D,MU,RHOA
4     FORMAT(4F)
      READ(22,5)DR
5     FORMAT(F)
      READ(22,5)DZ
      READ(22,6)D
6     FORMAT(I)
      READ(22,3)BPX(1),BPY(1)
      MM=2
7     READ(22,3)BPX(MM),BPY(MM)
```

```
      IF(BPX(1).EQ.BPX(MM).AND.BPY(1).EQ.BPY(MM))GO TO 8
      MM=MM+1
      GO TO 7
8     MM=MM-1
      DO 9 I=1,6
9     READ(22,60)D
      N=1
10    READ(22,11)XB(N),YB(N),NSEQ(N)
11    FORMAT(2F,1)
      IF(NSEQ(N).EQ.1)GO TO 12
      N=N+1
      GO TO 10
12    KMAXZZ=KMAX-2
      LMAXZZ=LMAX-2
      KMAXZ=KMAX-1
      LMAXZ=LMAX-1
      XBMAX=DR*KMAXZZ
      YBMAX=DZ*LMAXZZ
      N=0
      MM=0
13    N=N+1
      MM=MM+1
      XBP(MM,1)=YB(N)
      YBP(MM,1)=XB(N)
      IF(NSEQ(N).EQ.0)GO TO 13
      XBP(MM,1)=YB(N)
      YBP(MM,1)=XB(N)
      MM=MM+1
      XBP(MM,1)=YBMAX
      YBP(MM,1)=XBMAX
      BLN(1)=MM
      MM=0
14    MM=MM+1
      N=N+1
      XBP(MM,2)=YB(N)
      YBP(MM,2)=XB(N)
      IF(NSEQ(N).EQ.1)GO TO 15
      GO TO 14
15    XBP(MM,2)=YB(N)
      YBP(MM,2)=XB(N)
      BLN(2)=MM
      NAME=NAME+6
      CALL IFILE(23,NAME)
      READ(23)M
      READ(23)U,V,P
      READ(23)DT,RHOA,TIME,NCYCLE,MU,EPS1
      READ(23)KDMN,KDMX,LDMN,LDMX,DR,DZ,KMAX,LMAX
      KMAXZ=KMAX-1
      LMAXZ=LMAX-1
      KMAXZZ=KMAX-2
      LMAXZZ=LMAX-2
      DO 414 I=1,KMAX
      DO 414 J=1,LMAX
414   BNDRY(I,J)=2
      DO 411 L=2,LMAXZ
      NBND=1
      IF((ME(2,L).AND.EMP).EQ.EMP)GO TO 401
```

```
      IF((ME(2,L).AND.FULL).NE.FULL)GO TO 399
      NBND=2
399   BNDRY(1,L)=1
      GO TO 402
401   BNDRY(1,L)=2
402   DO 411 K=2,KMAX
      GO TO (403,405,407,409,411),NBND
403   IF((ME(K,L).AND.OB).EQ.OB)GO TO 404
      BNDRY(K,L)=2
      GO TO 411
404   BNDRY(K,L)=1
      NBND=2
      GO TO 411
405   IF((ME(K,L).AND.OB).NE.OB)GO TO 406
      BNDRY(K,L)=1
      NBND=3
      GO TO 411
406   BNDRY(K,L)=0
      NBND=3
      GO TO 411
407   IF((ME(K,L).AND.OB).EQ.OB.OR.(ME(K,L).AND.BND).EQ.BND)
     *  GO TO 408
      BNDRY(K,L)=0
      GO TO 411
408   BNDRY(K,L)=1
      NBND=4
      GO TO 411
409   IF((ME(K,L).AND.OB).EQ.OB.OR.(ME(K,L).AND.BND).EQ.BND)
     *  GO TO 410
      BNDRY(K,L)=2
      NBND=5
      GO TO 411
410   BNDRY(K,L)=1
      NBND=5
411   CONTINUE
      DO 412 K=1,KMAX
      IF((ME(K,1).AND.BND).NE.BND)GO TO 412
      BNDRY(K,1)=1
      IF((ME(K,LMAX).AND.BND).NE.BND)GO TO 412
      BNDRY(K,LMAX)=1
412   CONTINUE
      DO 413 K=1,KMAX
413   IF(BNDRY(K,2).EQ.2)BNDRY(K,1)=2
      IF(NYS.EQ.'N')GO TO 303
      TYPE 299
299   FORMAT(1H ,'ABMAC CALCULATION FOR PLOT ROUTINES')
      DO 301 L=1,LMAX
303   RB(1)=1./10.**10
      ZB(1)=0.
      KMAXZ=KMAX-1
      LMAXZ=LMAX-1
      DO 415 I=1,KMAXZ
415   RB(I+1)=RB(I)+DR
      DO 416 I=1,LMAXZ
416   ZB(I+1)=ZB(I)+DR
      DO 417 I=1,KMAX
      DO 417 J=1,LMAX
```

```
417     A(I,J)=DR
        DO 100 J=1,2
        NBMAX=BLN(J)-1
        ISEC=0
        DO 32 I=1,NBMAX
        IF(YBB(I+1,J)-YBB(I,J))20,24,16
16      L1=(YBB(I,J)+.000001)/DZ+2
        AA=(XBB(I+1,J)-XBB(I,J))/(YBB(I+1,J)-YBB(I,J))
        L2=LMAX
        B=XBB(I,J)-AA*YBB(I,J)
        LDL=1
        N=0
        DO 18 L=L1,L2,LDL
        IF(ZB(L)-YBB(I+1,J)-.000001)17,17,19
17      N=N+1
        ZMP(N)=ZB(L)
18      RMP(N)=AA*ZMP(N)+B
19      LLINE=N
        GO TO 25
20      L1=(YBB(I,J)-.000001)/DZ+1
        AA=(XBB(I+1,J)-XBB(I,J))/(YBB(I+1,J)-YBB(I,J))
        L2=1
        B=XBB(I,J)-AA*YBB(I,J)
        LDL=-1
        N=0
        DO 22 L=L1,L2,LDL
        IF(ZB(L)-YBB(I+1,J)+.000001)23,21,21
21      N=N+1
        ZMP(N)=ZB(L)
22      RMP(N)=ZMP(N)*AA+B
23      LLINE=N
        GO TO 25
24      LLINE=0
25      CONTINUE
        JMAX=LLINE
        DO 27 KJ=1,JMAX
27      DS(KJ)=(RMP(KJ)-XBB(I,J))**2+(ZMP(KJ)-YBB(I,J))**2
        DO 30 N=1,JMAX
        TEST=10.0**5
        MTRANS=0
        DO 29 MM=1,JMAX
        IF(DS(MM)-TEST)28,28,29
28      TEST=DS(MM)
        MTRANS=MM
29      CONTINUE
        RNP(N)=RMP(MTRANS)
        ZNP(N)=ZMP(MTRANS)
        DS(MTRANS)=10.0**7
30      CONTINUE
        J1=ISEC+1
        J2=ISEC+JMAX
        N=0
        DO 31 KJ=J1,J2
        N=N+1
        PTX(KJ)=RNP(N)
        PTY(KJ)=ZNP(N)
31      CONTINUE
```

```
      ISEC=ISEC+JMAX
32    CONTINUE
      I=1
33    I=I+1
34    IF(ABS(PTX(I)-PTX(I-1))-DEPSX)35,35,38
35    IF(ABS(PTY(I)-PTY(I-1))-DEPSY)36,36,38
36    ISEC=ISEC-1
      DO 37 KJ=I,ISEC
      PTX(KJ)=PTX(J+1)
      PTY(KJ)=PTY(J+1)
37    CONTINUE
      IF(I-ISEC)34,34,39
38    IF(I-ISEC)33,39,39
39    CONTINUE
      ISECZ=ISEC-1
      DEP=(PTX(ISEC)-PTX(ISECZ))**2+(PTY(ISEC)-PTY(ISECZ))**2
      DEP=SQRT(DEP)
      IF(DEP-.2*DR)41,42,42
41    ISEC=ISECZ
42    CONTINUE
      DEP=(PTX(2)-PTX(1))**2+(PTY(2)-PTY(1))**2
      DEP=SQRT(DEP)
      L=ISEC
      IF(NYS.EQ.'N')GO TO 54
      TYPE 53
53    FORMAT(1H ,///,' BOUNDARY-MESH INTERSECTIONS')
      DO 51 K=1,L
51    TYPE 52,K,PTX(K),K,PTY(K)
52    FORMAT(1H ,'PTX(',I2,')=',F,' PTY(',I2,')=',F)
54    DO 50 K=1,L
      II=(PTX(K)+.001)/DR+2
      JJ=(PTY(K)+.001)/DZ+2
      X1=PTX(K)-FLOAT(II-2)*DR
      Y1=PTY(K)-FLOAT(JJ-2)*DZ
      IF(X1.GT..000001)GO TO 47
      IF(Y1.LT.00005.OR.Y1.GT.0.199993)GO TO 49
      IF(BNDRY(II,JJ).EQ.1)GO TO 46
      A(II,JJ+1)=Y1
46    A(II,JJ)=DR-Y1
      GO TO 49
47    IF(X1.GE..19998)GO TO 49
      IF(BNDRY(II,JJ).EQ.1)A(II,JJ)=DR-X1
      IF(BNDRY(II+1,JJ).EQ.1)A(II+1,JJ)=X1
      IF(BNDRY(II-1,JJ).EQ.1)A(II-1,JJ)=DR+X1
1721  FORMAT(1H ,2I,F,I)
49    CONTINUE
50    CONTINUE
100   CONTINUE
      IF(NYS.EQ.'N')GO TO 312
      DO 310 K=1,KMAX
      DO 310 L=1,LMAX
      IF(ABS(A(K,L)-0.2).LE.0.0001)GO TO 310
310   CONTINUE
312   CONTINUE
      DO 915 L=1,LMAX
      SI(1,L)=0.5
      NBND=0
```

```
      IF(BNDRY(1,L).EQ.1)NBND=1
      DO 914 K=2,KMAX
      IF(NBND.GT.0)GO TO 912
      IF(BNDRY(K,L).EQ.1)NBND=1
911   SI(K,L)=0.5
      GO TO 914
912   IF(BNDRY(K,L).EQ.0)NBND=2
      IF(NBND.EQ.2.AND.BNDRY(K,L).EQ.2)GO TO 915
      SI(K,L)=SI(K-1,L)-0.5*A(K-1,L)*
     *  (VV(K,L)*RB(K)+VV(K-1,L)*RB(K-1))
914   CONTINUE
915   CONTINUE
900   CALL SETDIS
      CALL RECT1(6)
      CALL CONTRL
      CALL SEND
      IF(IDI.EQ.1)CALL AMOUSE
      IF(IDI.EQ.2)CALL WMOUSE
      INY=(INY+128)/128
      GO TO (106,107,109,111,108,111),INY
106   CALL EXIT
107   GO TO 105
108   CONTINUE
201   CALL CHGBIN
      CALL CLEAR
      CALL CNTRL1
200   CALL VPLOT(V)
      IF(IDI.EQ.1)CALL AMOUSE
      IF(IDI.EQ.2)CALL WMOUSE
      INY=(INY+128)/128
      GO TO (900,200,201),INY
109   CALL CHGBIN
      CALL CLEAR
      RB(1)=1./10.**10
      DO 309 K=1,KMAXZ
309   RB(K+1)=RB(K)+DR
      ZB(1)=0.
      DO 306 L=1,LMAXZ
306   ZB(L+1)=ZB(L)+DZ
      R(1)=-0.5*RB(2)
      Z(1)=-.05*ZB(2)
      DO 307 K=2,KMAX
307   R(K)=(RB(K)+RB(K-1))*.5
      DO 308 L=2,LMAX
308   Z(L)=(ZB(L)+ZB(L-1))*.5
      CALL VVECT(U,V,R,Z,M)
      CALL SEND
      IF(IDI.EQ.1)CALL AMOUSE
      IF(IDI.EQ.2)CALL WMOUSE
      INY=(INY+128)/128
      GO TO (900,300),INY
300   CALL CHGBIN
      CALL MOVETO(0,1000)
      CALL SEND
      CALL CHGASC
      TYPE 301
301   FORMAT(1H ,'WS=',$)
```

```
      ACCEPT 302,WS
302   FORMAT(F)
      TYPE 357
357   FORMAT(1H ,'WCX=',$)
      ACCEPT 304,WCX
304   FORMAT(F)
      TYPE 305
305   FORMAT(1H ,'WCY=',$)
      ACCEPT 304,WCY
      CALL CHGBIN
      GO TO 109
110   CONTINUE
111   CALL CLEAR
      SC=700/((LMAXZ-2)*DZ)
      CALL CNTRL2
      CALL WMOUSE
      INYGT=(INY+128)/128
      IF(INYGT.EQ.5)GO TO 900
      TYPE 113
113   FORMAT(1H ,'DEFINE REGION OF INTEREST')
      CALL WMOUSE
      WCX=(INX-128)/SC
      WCY=(INY-380)/SC+XL
      CALL WMOUSE
      WS=WCX-(INX-128)/SC
      TYPE 114
114   FORMAT(1H ,'POSITIVE CONTOUR LEVELS=',$)
      ACCEPT 115,NCLS
115   FORMAT(I)
      TYPE 116
116   FORMAT(1H ,'NEGATIVE CONTOUR LEVELS=',$)
      ACCEPT 115,NCL
      XL=0
      XW=0
      DO 117 I=1,BLN(1)
117   IF(YBP(I,1).GT.XL)XL=YBP(I,1)
      DO 121 J=1,2
      DO 119 I=1,BLN(J)
      IX=1023/2*(1+(XBP(I,J)-WCX)/WS)
      IY=1023/2*(1+(YBP(I,J)-WCY+XL)/WS)
      IF(I.EQ.1)GO TO 118
      CALL VECLP(IX,IY)
      GO TO 119
118   CALL MOVCLP(IX,IY)
      IONE=1
119   CONTINUE
      DO 121 I=1,BLN(J)
      IX=1023/2*(1+(XBP(I,J)-WCX)/WS)
      IY=1023/2*(1+(XL-YBP(I,J)-WCY)/WS)
      IF(I.EQ.1)GO TO 120
      CALL VECLP(IX,IY)
      GO TO 121
120   CALL MOVCLP(IX,IY)
121   CONTINUE
      CALL SEND
      GO TO (122,123,124,125,900),INYGT
122   CALL CONTOR(18,92,5.,VV,BNDRY,A,18,82)
```

```
      GO TO 126
123   CALL CONTOR(18,92,5.,UU,BNDRY,A,18,92)
      GO TO 126
124   CALL CONTOR(18,92,5.,PP,BNDRY,A,18,92)
      GO TO 126
125   CONTINUE
      XX=LMAX*DZ-DZ
      IMX=BLN(2)
      IX=1023/2*(1+(-WCX)/WS)
      IY=1023/2*(1+(XL-WCY)/WS)
      CALL MOVCLP(IX,IY)
      IX=1023/2*(1+(XBP(IMX,2)-WCX)/WS)
      CALL VECLP(IX,IY)
      IX=1023/2*(1+(XBP(1,2)-WCX)/WS)
      CALL MOVCLP(IX,IY)
      IX=1023/2*(1+(XX-WCX)/WS)
      CALL VECLP(IX,IY)
      CALL APND
      CALL CONTOR(18,92,5.,SI,BNDRY,A,18,92)
126   CALL APND
      CALL WMOUSE
      GO TO 111
      GO TO 900
      END
      SUBROUTINE RECT1(N)
      DO 10 I=1,N
      JPOS=(I*128)-128
      CALL MOVETO(704,JPOS)
10    CALL RECT
      RETURN
      END
      SUBROUTINE RECT
      CALL VEC(0,64)
      CALL VEC(192,0)
      CALL VEC(0,-64)
      CALL VEC(-192,0)
      RETURN
      END
      SUBROUTINE AMOUSE
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      CALL CHGASC
      TYPE 10
10    FORMAT(1H ,'*')
      ACCEPT 11,INY
11    FORMAT(I)
      INSW=INY
      INY=INY*128-128
      RETURN
      END
      SUBROUTINE CONTRL
      CALL MOVETO(770,24)
      CALL WRITE('STOP<>')
      CALL MOVETO(746,152)
      CALL WRITE('NEW DATA<>')
      CALL MOVETO(752,264)
      CALL WRITE('VECTORS<>')
      CALL MOVETO(746,296)
```

```
      CALL WRITE('VELOCITY<>')
      CALL MOVETO(722,408)
      CALL WRITE('CONTOUR PLOT<>')
      CALL MOVETO(746,520)
      CALL WRITE('PROFILES<>')
      CALL MOVETO(746,552)
      CALL WRITE('VELOCITY<>')
      CALL MOVETO(710,664)
      CALL WRITE('ISOMETRIC PLOT<>')
      CALL MOVETO(900,24)
      CALL WRITE('1<>')
      CALL MOVETO(900,152)
      CALL WRITE('2<>')
      CALL MOVETO(900,280)
      CALL WRITE('3<>')
      CALL MOVETO(900,408)
      CALL WRITE('4<>')
      CALL MOVETO(900,536)
      CALL WRITE('5<>')
      CALL MOVETO(900,664)
      CALL WRITE('6<>')
      RETURN
      END
      SUBROUTINE VPLOT(V)
      COMMON/WINDOW/WS,WCX,WCY,XL
      COMMON/STUFF/DR,DZ,KMAX,SZF
      COMMON/MOUSE/INX,INY,INSW,SWTAT
      DIMENSION V(2500)
      DIMENSION PV(50)
      KMAXZ=KMAX-1
      CALL CHGBIN
      CALL MOVETO(0,1000)
      CALL CHGASC
      TYPE 3
3     FORMAT(1H ,'PROFILE AT Z=',$)
      ACCEPT 4,PZ
      LPOS=200+(PZ*190)/((KMAX-2)*2*DR)-5
      CALL MOVETO(LPOS,820)
      CALL VEC(-10,0)
      CALL VEC(5,15)
      CALL VEC(5,-15)
4     FORMAT(F)
      L=PZ/DZ+1
      KTT=(L-1)*KMAX
      DO 6 K=1,KMAX
      KM=KTT+K
      PV(K+KMAX)=V(KM)
6     PV(KMAX-K+1)=V(KM)
      SC=350/(KMAX-2)
      MOVE=0
      DO 7 K=2,KMAX
      IF(PV(K)+10.00000.LT..0001)GO TO 7
      GO TO 8
7     CONTINUE
8     KST=K-1
      PV(KST)=0.
      KND=KMAX*2-KST+1
```

```
      PV(KND)=0.
12    DO 13 K=KST,KND
      IX=K*SC+120
      IY=-10.*PV(K)+128
      IF(MOVE.EQ.0)CALL MOVETO(IX,IY)
      CALL ZIPTO(IX,IY)
13    MOVE=1
      CALL APND
      RETURN
      END
      SUBROUTINE VVECT(U,V,R,Z,M)
      DIMENSION U(2500),V(2500),M(2500),R(150),Z(150)
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/WINDOW/WS,WCX,WCY,XL
      COMMON/STUFF/DR,DZ,KMAX,SZF,KDMN,LDMN,KDMX,LDMX,EPS1,DT
      COMMON/BOUN/BLN(2),XBP(50,2),YB (50,2)
      INTEGER BLN
      INTEGER FULL,SUR
      DATA FULL,SUR/8,4/
      DO 311 J=1,2
      DO 312 I=1,BLN(J)
      IX=1023/2*(1+(XBP(I,J)-WCX)/WS)
      IY=1023/2*(1+(YBP(I,J)-WCY+XL)/WS)
      IF(I.EQ.1)GO TO 345
      CALL VECLP(IX,IY+512)
      GO TO 312
345   CALL MOVCLP(IX,IY+512)
312   CONTINUE
      DO 313 I=1,BLN(J)
      IX=1023/2*(1+(XBP(I,J)-WCX)/WS)
      IY=1023/2*(1+(XL-YBP(I,J)-WCY)/WS)
      IF(I.EQ.1)GO TO 355
      CALL VECLP(IX,IY+512)
      GO TO 313
355   CALL MOVCLP(IX,IY+512)
313   CONTINUE
311   CONTINUE
      KKK=.5*(2.+KDMN)+.5*ABS(2.-KDMN)
      LLL=.5*(2.+LDMN)+.5*ABS(2.-LDMN)
      IONE=1
26    DO 375 K=KKK,KDMX
      DO 375 L=LLL,LDMX
      KM=K+(L-1)*KMAX
      KB=KM-KMAX
      IF(((M(KM).AND.SUR).EQ.SUR).OR.((M(KM).AND.FULL).EQ.FULL))
     *  GO TO 203
      GO TO 375
203   V1=.5*(V(KM)+V(KB))
      U1=.5*(U(KM)+U(KM-1))
      IF(ABS(U1)-.001)41,41,40
41    IF(ABS(V1)-.001)375,375,40
40    R1=R(K)
      Z1=Z(L)
      R2=R1+2.*DT*U1*DR/EPS1
      Z2=Z1+2.*DT*V1*DZ/EPS1
      R1=IONE*R1
      R2=IONE*R2
```

```
      R1=R1+XL
      R2=R2+XL
      IX1=1023/2*(1+(Z1-WCX)/WS)
      IX2=1023/2*(1+(Z2-WCX)/WS)
      IY1=512+1023/2*(1+(R1-WCY)/WS)
      IY2=512+1023/2*(1+(R2-WCY)/WS)
      IF((IX1-IX2.LE.1).AND.(IY1-IY2.LE.1))GO TO 375
      CALL MOVCLP(IX1,IY1)
      CALL VECLP(IX2,IY2)
      GO TO 375
375   CONTINUE
      IF(IONE.NE.1)GO TO 452
      IONE=-1
      GO TO 26
452   RETURN
      END
      SUBROUTINE CNTRL1
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/STUFF/DR,DZ,KMAX
      COMMON/BOUN/BLN(2),XBP(50,2),YBP(50,2)
      INTEGER BLN
      CALL MOVETO(120,512)
      CALL VECTO(120,116)
      CALL MOVETO(108,128)
      CALL VECTO(820,128)
      DO 1 J=1,4
      JPOS=128+J*100
      JN=J*10
      CALL MOVETO(120,JPOS)
      CALL VEC(-12,0)
      CALL MOVE(-30,-8)
      CALL WRITE('ΔΔ.<>',JN)
1     CONTINUE
      DO 2 I=1,4
      IPOS=120+I*175
      IN=-32+I*16
      CALL MOVETO(IPOS,128)
      CALL VEC(0,-12)
      CALL MOVE(-12,-16)
2     CALL WRITE('ΔΔΔ.<>',IN)
      CALL MOVETO(340,48)
      CALL WRITE('RADIAL POSITION (MM)<>')
      CALL MOVETO(32,512)
      CALL WRITE('A<>')
      CALL MOVETO(32,492)
      CALL WRITE('X<>')
      CALL MOVETO(32,472)
      CALL WRITE('I<>')
      CALL MOVETO(32,452)
      CALL WRITE('A<>')
      CALL MOVETO(32,432)
      CALL WRITE('L<>')
      CALL MOVETO(32,392)
      CALL WRITE('V<>')
      CALL MOVETO(32,372)
      CALL WRITE('E<>')
      CALL MOVETO(32,352)
```

```
      CALL WRITE('L<>')
      CALL MOVETO(32,332)
      CALL WRITE('O<>')
      CALL MOVETO(32,312)
      CALL WRITE('C<>')
      CALL MOVETO(32,292)
      CALL WRITE('I<>')
      CALL MOVETO(32,272)
      CALL WRITE('T<>')
      CALL MOVETO(32,252)
      CALL WRITE('Y<>')
      CALL MOVETO(32,212)
      CALL WRITE('C<>')
      CALL MOVETO(32,192)
      CALL WRITE('M<>')
      CALL MOVETO(32,172)
      CALL WRITE('/<>')
      CALL MOVETO(32,152)
      CALL WRITE('S<>')
      CALL MOVETO(32,132)
      CALL WRITE('E<>')
      CALL MOVETO(32,112)
      CALL WRITE('C<>')
      SC=190/((KMAX-2)*DR*2)
      DO 7 J=1,2
      DO 4 I=1,BLN(J)
      IX=XBP(I,J)*SC
      IY=YBP(I,J)*SC
      IF(I.EQ.1)GO TO 3
      CALL VECTO(IX+200,927+IY)
      GO TO 4
3     CALL MOVETO(IX+200,927+IY)
4     CONTINUE
      DO 6 I=1,BLN(J)
      IX=XBP(I,J)*SC
      IY=YBP(I,J)*SC
      IF(I.EQ.1)GO TO 5
      CALL VECTO(IX+200,927-IY)
      GO TO 6
5     CALL MOVETO(IX+200,927-IY)
6     CONTINUE
7     CONTINUE
      CALL MOVETO(416,800)
      CALL WRITE('AXIAL POSITION<>')
      CALL MOVETO(886,24)
      CALL WRITE('RETURN<>')
      CALL MOVETO(886,152)
      CALL WRITE('OVERLAY<>')
      CALL MOVETO(896,280)
      CALL WRITE('CLEAR<>')
      CALL SEND
452   RETURN
      END
      SUBROUTINE CNTRL2
      COMMON/MOUSE/INX,INY,INSW,SWSTAT
      COMMON/STUFF/DR,DZ,KMAX,LMAX
      COMMON/BOUN/BLN(2),XBP(50,2),YBP(50,2)
```

```
      INTEGER BLN
      CALL MOVETO(868,24)
      CALL WRITE('V-VELOCITY<>')
      CALL MOVETO(868,152)
      CALL WRITE('U-VELOCITY<>')
      CALL MOVETO(868,280)
      CALL WRITE('PRESSURE<>')
      CALL MOVETO(874,396)
      CALL WRITE('FUNCTION<>')
      CALL MOVETO(886,422)
      CALL WRITE('STREAM<>')
      SC=700/((LMAX-2)*DZ)
      DO 5 J=1,2
      DO 2 I=1,BLN(J)
      IX=XBP(I,J)*SC+128
      IY=YBP(I,J)*SC+380
      IF(I.EQ.1)GO TO 1
      CALL VECTO(IX,IY)
      GO TO 2
1     CALL MOVETO(IX,IY)
2     CONTINUE
      DO 4 I=1,BLN(J)
      IX=XBP(I,J)*SC+128
      IY=380-YBP(I,J)*SC
      IF(I.EQ.1)GO TO 3
      CALL VECTO(IX,IY)
      GO TO 4
3     CALL MOVETO(IX,IY)
4     CONTINUE
5     CONTINUE
      CALL SEND
      RETURN
      END
```

```
      SUBROUTINE CONTOR(M,N,XDIS,G,BND,A,MD,ND)
      DIMENSION G(MD,ND)
      DIMENSION XX(5),YY(5),B(200),X(200),
     *  Y(200),BND(MD,ND),A(MD,ND),F(5),S(5),T(5)
      DIMENSION XS(8000),YS(8000),XP(300),YP(300)
      COMMON/WINDOW/WS,WCX,WCY,XL
      COMMON/CONLVL/NCLS,NCL
      INTEGER BND
      SCALE=XDIS/FLOAT(M-1)
      G1=G(1,1)
      G2=G1
      DO 66 I=1,M
      DO 66 J=1,N
      IF(BND(I,J).GT.1)GO TO 66
      G1=AMIN1(G1,G(I,J))
      G2=AMAX1(G2,G(I,J))
66    CONTINUE
      NCL1=NCLS
      TEMP=G1
      G1=0.0
      GS=(G2-G1)/FLOAT(NCL1+1)
      B(1)=G1+GS
      DO 67 K=2,NCL1
67    B(K)=B(K-1)+GS
      G1=TEMP
      IF(G1.GE.0.0)GO TO 68
      TST=-G1/(NCL+1)
      NCL1=NCL+NCLS
      NCLN=NCLS+2
      B(NCLS+1)=G1+TST
      DO 69 K=NCLN,NCL1
69    B(K)=B(K-1)+TST
68    CALL CHGSIN
      CALL MOVETO(0,1000)
      CALL APND
      TYPE 70,G1,G2,(B(K),K=1,NCL1)
70    FORMAT(1H ,10X,'MIN GRID VALUE=' ,F,3X,
     *  'MAX GRID VALUE= ',F,/1H ,10X,'CONTOUR LEVELS'/
     *  (10X,4F))
      CALL CHGSIN
      DO 1 I=1,100
      X(I)=(I-1)*.2
1     Y(I)=(I-1)*.2
      M1=M-1
      N1=N-1
      LS=0
      DO 44 I=1,M1
      NS=1
2     JTEMP=1
      DO 5 J=NS,N1
      GO TO (3,4),JTEMP
3     IF(BND(I,J+1).GE.1.AND.BND(I+1,J+1).GE.1)GO TO 5
      NB=J
      JTEMP=2
      GO TO 5
4     IF(BND(I,J+1).EQ.0.OR.BND(I+1,J+1).EQ.0)GO TO 5
      NF=J
```

```
      NS=J+1
      IF(NS.GT.N)NS=N
      GO TO 6
 5    CONTINUE
      GO TO 44
 6    DO 43 J=NB,NF
      ICL=1
      IS=0
      IF(BND(I,J).GE.1)IS=IS+6
      IF(BND(I,J+1).GE.1)IS=IS+2
      IF(BND(I+1,J+1).GE.1)IS=IS+3
      IF(BND(I+1,J).GE.1)IS=IS+4
      IF(A(I+1,J)-1.)11,7,11
 7    IF(A(I,J)-1.)11,8,11
 8    IF(A(I+1,J+1)-1.)11,9,11
 9    IF(A(I+1,J)-1.)11,10,11
 10   IF(BND(I,J).EQ.2.OR.BND(I,J+1).EQ.2.OR.BND(I+1,J+1).
     * EQ.2.OR.BND(I+1,J).EQ.2)GO TO 11
      IS=1
      GO TO 13
11    IF(IS.GE.1.AND.IS.LE.15)GO TO 13
      IS=1
      KKKKK=KKKKK+1
      IF(KKKKK.GT.40)GO TO 13
      IF(ISEE.EQ.1)TYPE 12,I,J
 12   FORMAT(1H ,20X,31HWARNING... 'IS' IS OUT OF RANGE,
     * 3X,2HI=I3,3X,2HJ=I3)
 13   F(1)=G(I,J)
      F(2)=G(I,J+1)
      F(3)=G(I+1,J+1)
      F(4)=G(I+1,J)
      S(1)=X(I)
      S(2)=X(I)
      S(3)=X(I+1)
      S(4)=X(I+1)
      T(1)=Y(J)
      T(2)=Y(J+1)
      T(3)=Y(J+1)
      T(4)=Y(J)
      IN=4
      GO TO (28,14,15,16,17,18,19,20,26,21,27,24,23,43,43),IS
 14   F(2)=G(I,J+1)
      F(3)=G(I,J+1)
      F(4)=G(I+1,J+1)
      F(5)=G(I+1,J)
      S(3)=X(I+1)-A(I,J+1)
      S(5)=X(I+1)
      T(2)=Y(J)+A(I,J+1)
      T(4)=Y(J+1)
      T(5)=Y(J)
      IN=5
      GO TO 28
 15   F(3)=G(I+1,J+1)
      F(4)=G(I+1,J+1)
      F(5)=G(I+1,J)
      S(3)=X(I)+A(I+1,J+1)
      S(5)=X(I+1)
```

```
      T(4)=Y(J)+A(I+1,J+1)
      T(5)=Y(J)
      IN=5
      GO TO 28
16    F(4)=G(I+1,J)
      F(5)=G(I+1,J)
      S(5)=X(I)+A(I+1,J)
      T(4)=Y(J+1)-A(I+1,J)
      T(5)=Y(J)
      IN=5
      GO TO 28
17    F(2)=G(I,J+1)
      F(3)=G(I+1,J+1)
      T(2)=Y(J)+A(I,J+1)
      T(3)=Y(J)+A(I+1,J+1)
      IN=4
      GO TO 28
18    IF(BND(I,J+1).GE.1.AND.BND(I+1,J).GE.1)GO TO 25
      F(1)=G(I,J)
      F(5)=G(I,J)
      S(5)=X(I+1)-A(I,J)
      T(1)=Y(J+1)-A(I,J)
      T(5)=Y(J)
      IN=5
      GO TO 28
19    F(3)=G(I+1,J+1)
      F(4)=G(I+1,J)
      S(3)=X(I)+A(I+1,J+1)
      S(4)=X(I)+A(I+1,J)
      IN=4
      GO TO 28
20    F(1)=G(I,J)
      F(2)=G(I,J+1)
      S(1)=X(I+1)-A(I,J)
      S(2)=X(I+1)-A(I,J+1)
      IN=4
      GO TO 28
21    F(1)=G(I,J)
      F(4)=G(I+1,J)
      T(1)=Y(J+1)-A(I,J)
      T(4)=Y(J+1)-A(I+1,J)
      IN=4
      GO TO 28
22    ICL=4
23    F(1)=G(I,J)
      F(3)=A(I+1,J+1)*(G(I+1,J+1)-G(I,J+1))+G(I,J+1)
      S(3)=X(I)+A(I+1,J+1)
      T(1)=Y(J+1)-A(I,J)
      IN=3
      GO TO 28
24    F(1)=G(I,J+1)
      F(2)=G(I+1,J+1)
      F(3)=G(I+1,J)
      S(1)=X(I+1)-A(I,J+1)
      S(2)=X(I+1)
      T(1)=Y(J+1)
      T(3)=Y(J+1)-A(I+1,J)
```

```
      IN=3
      ICL=1
      GO TO 28
25    ICL=2
26    IF (BND(I,J).GE.1.AND.BND(I+1,J+1).GE.1) GOTO 22
      F(2)=G(I,J+1)
      F(3)=G(I+1,J)
      S(3)=X(I)+A(I+1,J)
      T(2)=Y(J)+A(I,J+1)
      T(3)=Y(J)
      IN=3
      GOTO 28
27    F(1)=G(I,J)
      F(2)=G(I+1,J+1)
      F(3)=G(I+1,J)
      S(1)=X(I+1)-A(I,J)
      S(2)=X(I+1)
      T(2)=Y(J)+A(I+1,J+1)
      T(3)=Y(J)
      IN=3
      ICL=1
28    DO 42 L=1,NCL1
      LL=0
      ISW=1
      I1=1
      I2=2
29    G1=F(I1)
      G2=F(I2)
      IF (B(L)-G1) 30,37,31
30    IF (B(L)-G2) 34,37,32
31    IF (G2-B(L)) 34,37,32
32    LL=LL+1
      IF (ABS(S(I2)-S(I1)) .LT. 0.0001) GOTO 33
      SLOPE=(T(I2)-T(I1))/(S(I2)-S(I1))
      IF (ABS(SLOPE).GT.1.0) GOTO 33
      XX(LL)=(S(I2)-S(I1))*(B(L)-G1)/(G2-G1)+S(I1)
      YY(LL)=(XX(LL)-S(I1))*SLOPE+T(I1)
      GOTO 34
33    SLOPE=(S(I2)-S(I1))/(T(I2)-T(I1))
      YY(LL)=(T(I2)-T(I1))*(B(L)-G1)/(G2-G1)+T(I1)
      XX(LL)=(YY(LL)-T(I1))*SLOPE+S(I1)
34    I1=I1+1
      I2=I2+1
      GOTO (35,36),ISW
35    IF (I2.LE.IN) GOTO 29
      I1=1
      I2=IN
      ISW=2
      GOTO 29
36    IF (LL.EQ.0) GOTO 42
      IF (LL.EQ.2) GOTO 39
37    IF(ISEE.EQ.1)TYPE 38,LL,B(L),I,J,(F(I1),I1=1,4)
38    FORMAT (' WARNING...',I2,F10.5,2I2,4F10.5)
      GOTO 42
39    IF (XX(1).LT.XX(2)) GOTO 40
      TEM=XX(1)
      XX(1)=XX(2)
```

```
      XX(2)=TEM
      TEM=YY(1)
      YY(1)=YY(2)
      YY(2)=TEM
40    LS=LS+1
      XS(LS)=XX(1)
      YS(LS)=YY(1)
      LS=LS+1
      XS(LS)=XX(2)
      YS(LS)=YY(2)
      IF (LS.LT. 8000) GOTO 42
      TYPE 41
41    FORMAT (' DIMENSION ON XS IS TOO SMALL IN SUB CONTOR')
      CALL EXIT
42    CONTINUE
      GOTO (43,24,43,27),ICL
43    CONTINUE
      GOTO 2
44    CONTINUE
      LSSAV=LS
      IONE=1
      ISW=-1
      EPS=.001
45    L=1
      IPS=1
      JJ=0
      IONE=1
      JJ=JJ+1
      XP(JJ)=XS(L)
      YP(JJ)=YS(L)
46    JJ=JJ+1
      IF (JJ.LT.300) GOTO 48
      TYPE 47
47    FORMAT (' DIMENSION ON XP,YP HAS BEEN EXCEEDED')
      CALL EXIT
48    XP(JJ)=XS(L+1)
      YP(JJ)=YS(L+1)
      K=0
49    K=K+1
      IF (K.EQ.L+1) GOTO 55
       IF (ABS(XS(K)-XS(L+1)).GT.EPS.OR.ABS(YS(K)-YS(L+1))
     *  .GT.EPS) GOTO 55
50    LS=LS-2
      IF (LS.LE.0) GOTO 56
      L1=L
      L=K-2
      K1=K
      IF ((-1.)**K.LT.0.0) GOTO 51
      TEM=XS(K-1)
      XS(K-1)=XS(K)
      XS(K)=TEM
      TEM=YS(K-1)
      YS(K-1)=YS(K)
      YS(K)=TEM
      L=K-3
      K1=K-1
51    IF (K.GT.L1+1) GOTO 52
```

```
      L=K1
52    IF (L.GT.0) GOTO 53
      L=1
53    DO 54 LL=L1,LS
      XS(LL)=XS(LL+2)
54    YS(LL)=YS(LL+2)
      GOTO (46,56),IPS
55    IF (K.LT.LS.AND.X(K).LT.FLOAT(M)) GOTO 49
      IPS=2
      L1=L
      LS=LS-2
      GOTO 53
56    J1=(JJ-1)*(1+ISW)/2+1
      CALL LNPLT(XP,YP,JJ)
      IX=1023/2*(1+(YP(J1)-WCX)/WS)
      IF(IONE.GT.1)GO TO 60
      IY=1023/2*(1+(XP(J1)-WCY+XL)/WS)
      GO TO 61
60    IY=1023/2*(1+(XL-XP(J1)-WCY)/WS)
61    CONTINUE
      CALL MOVCLP(IX,IY)
      DO 59 J=2,JJ
      IF (ISW.LT.0) GOTO 57
      J1=JJ+1-J
      GOTO 58
57    J1=J
58    IX=1023/2*(1+(YP(J1)-WCX)/WS)
      IF(IONE.GT.1)GO TO 62
      IY=1023/2*(1+(XP(J1)-WCY+XL)/WS)
      GO TO 63
62    IY=1023/2*(1+(XL-XP(J1)-WCY)/WS)
63    CONTINUE
      CALL VECLP(IX,IY)
59    CONTINUE
      IF(IONE.GT.1)GO TO 64
      IONE=IONE+1
      GO TO 56
64    CONTINUE
      IF(LS.LE.0)RETURN
      ISW=-ISW
      GOTO 45
      END
      SUBROUTINE PLOT(X,Y,N)
      COMMON/FIXT/DX,DY
      IF(N.LT.0)GO TO 1
      A=100.*(X+DX)
      B=100.*(Y+DY)
      NA=IFIX(A)
      NB=IFIX(B)
      IF(N.EQ.3)CALL MOVETO(NA,NB)
      IF(N.EQ.2)CALL VECTO(NA,NB)
      RETURN
1     DX=DX+X
      DY=DY+Y
      RETURN
      END
      SUBROUTINE FINI
```

```
      CALL APND
      RETURN
      END
      SUBROUTINE IDPLOT(XX,YY)
      COMMON/FIX1/DX,DY
      DX=2.0
      DY=4.2
      CALL SETDIS1
      WS=8.0
      WCX=6.0
      WCY=6.0
      RETURN
      END
      SUBROUTINE SYMBL4(XX,YY,HT,NOO,NNN,I)
      RETURN
      END
      SUBROUTINE NUMBER(XXX,YYY,HHH,RRR,ZZZ,IIIK)
      RETURN
      END
```

```
C       THIS SET OF SUBROUTINES PERFORMS CLIPPING ON
C       DATA PASSED TO IT THROUGH VECLP AND MOVCLP
C       CALLS. (4-20-72)
        SUBROUTINE VECLP(IX,IY)
        COMMON/CLP/IN,IXP,IYP,MN
        LOGICAL IN
        IF(.NOT.IN) GO TO 12
C       THE PREVIOUS POINT WAS INSIDE OF THE FIELD
C       OF VISION DEFINED BY THE SCOPE.
        N=0
        IF(IX.GT.1023)N=N+1
        IF(IX.LT.0)N=N+2
        IF(IY.GT.1023)N=N+3
        IF(IY.LT.0)N=N+6
        N=N+1
        S=(IY-IYP)/(IX-IXP)
        GO TO(1,3,5,4,2,4,6,8,6),N
C
C       ENTIRE LINE IS VISIBLE
C
1       CALL VECTO(IX,IY)
        GO TO 11
C
C       LINE GOES OUT TOP RIGHT HAND SIDE OF SCREEN
C
2       IXS=(1023-IYP)/S+IXP
        IF(IXS.GT.1023) GO TO 3
        CALL VECTO(IXS,1023)
        GO TO 10
3       IF(IY.EQ.IYP)GO TO 35
        IYS=(1023-IXP)*S+IYP
        CALL VECTO(1023,IYS)
        GO TO 10
35      CALL VECTO(1023,IY)
        GO TO 10
C
C       LINE GOES OUT TOP OR TOP LEFT HAND SIDE
C       OF SCREEN.
C
4       IF(IX.EQ.IXP)GO TO 45
        IXS=(1023-IYP)/S+IXP
        IF(IXS.LT.0)GO TO 5
        CALL VECTO(IXS,1023)
        GO TO 10
45      CALL VECTO(IX,1023)
        GO TO 10
C
C       LINE GOES OUT LEFT HAND SIDE OF SCREEN.
C
5       IF(IY.EQ.IYP)GO TO 55
        IYS=IYP-S*IXP
        CALL VECTO(0,IYS)
        GO TO 10
55      CALL VECTO(0,IY)
        GO TO 10
C
C       LINE GOES OUT OF BOTTOM OR BOTTOM LEFT CORNER.
```

```
C
6       IF(IX.EQ.IXP)GO TO 65
        IXS=IXP-IYP/S
        IF(IXS.LT.0)GO TO 7
        CALL VECTO(IXS,0)
        GO TO 10
65      CALL VECTO(IX,0)
        GO TO 10
7       IYS=IYP-IXP*S
        CALL VECTO(0,IYS)
        GO TO 10
C
C       LINE GOES OUT BOTTOM RIGHT HAND CORNER.
C
8       IYS=IYP+S*(1023-IXP)
        IF(IYS.LT.0)GO TO 9
        CALL VECTO(1023,IYS)
        GO TO 10
9       IXS=IXP-IYP/S
        CALL VECTO(IXS,0)
10      IN=.FALSE.
11      IXP=IX
        IYP=IY
        MN=N-1
        RETURN
C
C       LINE IS COMING INTO FIELD OF VISION.
C
12      N=0
        S=(IY-IYP)/(IX-IXP)
        IF(IX.GT.1023)N=N+1
        IF(IX.LT.0)N=N+2
        IF(IY.GT.1023)N=N+3
        IF(IY.LT.0)N=N+6
        N=N+1
        GO TO (13,23,23,23,23,23,23,23,23),N
13      GO TO (15,17,16,14,16,18,20,18),MN
14      IXS=(1023-IY)/S+IX
        IF(IXS.GT.1023)GO TO 15
        CALL MOVETO(IXS,1023)
        CALL VECTO(IX,IY)
        GO TO 22
C
C       LINE IS COMING FROM RIGHT SIDE OF SCREEN.
C
15      IF(IY.EQ.IYP)GO TO 155
        IYS=(1023-IX)*S+IY
        CALL MOVETO(1023,IYS)
        CALL VECTO(IX,IY)
        GO TO 22
155     CALL MOVETO(1023,IY)
        CALL VECTO(IX,IY)
        GO TO 22
C
C       LINE IS COMING FROM TOP OR TOP LEFT.
C
16      IF(IX.EQ.IXP)GO TO 166
```

```
      IXS=(1023-IY)/S+IX
      IF(IXS.LT.0)GO TO 17
      CALL MOVETO(IXS,1023)
      CALL VECTO(IX,IY)
      GO TO 22
166   CALL MOVETO(IX,1023)
      CALL VECTO(IX,IY)
      GO TO 22
C
C     LINE IS COMING FROM LEFT SIDE OF SCREEN.
C
17    IF(IY.EQ.IYP)GO TO 175
      IYS=IY-S*IX
      CALL MOVETO(0,IYS)
      CALL VECTO(IX,IY)
      GO TO 22
175   CALL MOVETO(0,IY)
      CALL VECTO(IX,IY)
      GO TO 22
18    IF(IX.EQ.IXP)GO TO 185
      IXS=IX-IY/S
      IF(IXS.LT.0)GO TO 19
      CALL MOVETO(IXS,0)
      CALL VECTO(IX,IY)
      GO TO 22
185   CALL MOVETO(IX,0)
      CALL VECTO(IX,IY)
      GO TO 22
19    IYS=IY-IX*S
      CALL MOVETO(0,IYS)
      CALL VECTO(IX,IY)
      GO TO 22
20    S=(IY-IYP)/(IX-IXP)
      IYS=IY+S*(1023-IX)
      IF(IYS.LT.0) GO TO 21
      CALL MOVETO(1023,IYS)
      CALL VECTO(IX,IY)
      GO TO 22
21    IXS=IX-IY/S
      CALL MOVETO(IXS,0)
      CALL VECTO(IX,IY)
22    IN=.TRUE.
23    MN=N-1
      IXP=IX
      IYP=IY
      RETURN
      END
      SUBROUTINE MOVCLP(IX,IY)
      COMMON/CLP/IN,IXP,IYP,MN
      LOGICAL IN
      IN=.TRUE.
      N=0
      IF(IX.GT.1023)N=N+1
      IF(IX.LT.0)N=N+2
      IF(IY.GT.1023)N=N+3
      IF(IY.LT.0)N=N+6
      N=N+1
```

```
GO TO (1,2,2,2,2,2,2,2,2),N
CALL MOVETO(IX,IY)
GO TO 3
IN=.FALSE.
IXP=IX
IYP=IY
MN=N-1
RETURN
END
SUBROUTINE DOTCLP(IX,IY)
COMMON/CLP/IN,IXP,IYP,MN
LOGICAL IN
N=0
IF(IX.GT.1023)N=N+1
IF(IX.LT.0)N=N+2
IF(IY.GT.1023)N=N+3
IF(IY.LT.0)N=N+6
N=N+1
GO TO (1,2,2,2,2,2,2,2,2),N
CALL DOTAT(IX,IY)
MN=N-1
IXP=IX
IYP=IY
RETURN
END
```

APPENDIX VI

MISCELLANEOUS PROGRAMS

Routine Name	Function
SAVCOR	Saves a core image of currently executing program
PSEUD	Sets up pseudo-interrupts
SLEEP	Dismisses process for specified number of ms.
GETIT	Determines latest version of saved core and gets it.
PRINT	Reads in data file and formats for line printer listing.
TYPE	Reads in data file and formats for teletype listing.
APFLE	Appends error function to a display file.

```
TITLE   SAVCOR
        ENTRY ABSAV
        EXTERNAL FLAG1
SAVREG: BLOCK 20
ABSAV:  0
        MOVEM   17,SAVREG+17
        MOVEI   17,SAVREG
        BLT     17,SAVREG+16
        MOVEI   1,400000
        GEVEC
        MOVEM   2,SAVLOC#
        HRLZI   1,1B18+1B35
        MOVE    3,FLAG1
        XORI    3,1
        MOVEM   3,FLAG1
        HRRO    2,RSTART(3)
        GTJFN
        HALTF
        HRRZM   1,JFNSAV#
        MOVEI   1,400000
        HRLI    2,1
        HRRI    2,RENTRY
        SEVEC
        HRLI    1,400000
        HRR     1,JFNSAV
        MOVE    2,[777760,,20]
        SAVE
RENTRY: MOVE    1,400000
        HRLI    2,1
        HRR     2,SAVLOC
        SEVEC
        HRLZI   17,SAVREG
        BLT     17,17
        JRA     16,(16)
RSTART: [ASCIZ/P.DMP;1/]
        [ASCIZ/P.DMP;2/]
        END
```

```
;       THIS PROGRAM ENABLES PSEUDO-INTERRUPTS TO
;       BE GENERATED. WHEN ^V IS TYPE IN, THE
;       INTERRUPT CHANGES THE SENSE OF THE
;       TTY FLAG.
        TITLE INTRPT
        ENTRY PSEUD
        EXTERNAL FLGS
        TTY=FLGS
SAVR    BLOCK 20
CHNTAB: BLOCK 44
LEVTAB: BLOCK 3
PCSAV:  BLOCK 3
PSEUD:  0
        MOVEM   17,SAVR+17
        MOVEI   17,SAVR
        BLT     17,SAVR+16
        HRLI    1,1
        MOVEM   1,CHNTAB
        HRRI    1,PCSAV
        MOVEM   1,LEVTAB
        MOVEI   1,400000
        HRLI    2,LEVTAB
        HRRI    2,CHNTAB
        SIR
        MOVEI   1,400000
        EIR
        MOVEI   1,400000
        MOVE    2,[400000,,0]
        AIC
        MOVE    2,[26,,0]
        ATI
        HRLZI   17,SAVR
        BLT     17,17
        JRA     16,(16)
MES:    ASCIZ\TTY FLAG CHANGED....\
LOC:    SETOM   TTY
        MOVEM   1,SAVR
        HRROI   1,MES
        PSOUT
        MOVE    1,SAVR
        DEBRK
        END
```

```
        TITLE SLEEP
;       DISMISSES THE PROCESS FOR SPECIFIED NUMBER OF MS.
        ENTRY   SLEEP
SLEEP:  0
        MOVEM   1,SAVE
        MOVEI   1,10000
        DISMS
        MOVE    1,SAVE
        JRA     16,(16)
        END
```

```
SEARCH STENEX

TITLE START RIGHT FIB JOB
LOC 200000
;THIS PROGRAM DETERMINES THE APPROPRIATE P.DMP FILE
;TO RUN FROM A FIB REQUEST.....
FNAM1:  ASCIZ/REQUESTED-F-I-B.INPUT/
FNAM2:  ASCIZ/P.DMP/
GETIT:  MOVE    1,[XWD 100001,-2]
        HRROI   2,FNAM1
        GTJFN
        HALTF
        MOVE    2,[XWD 1,7]
        MOVEI   3,2
        GTFDB
        HLRZ    2,2
        ROT     2,-1
        JUMPL   2,ODD
        MOVE    1,[XWD 100001,2]
        JRST    NXT
ODD:    MOVE    1,[XWD 100001,1]
NXT:    HRROI   2,FNAM2
        GTJFN
        HALTF
        HRLI    1,400000
        GET
        HRRZ    1,120
        JRST    0(1)
        HALTF
        END GETIT
```

```
      DIMENSION UU(18,82),VV(18,82),PP(18,82)
      DIMENSION A(18,82),U(2500),V(2500),P(2500)
      DIMENSION MESS(7)
      EQUIVALENCE (U,UU),(V,VV),(P,PP)
      TYPE 100
100   FORMAT(1H ,'INPUT FILE:',$)
      ACCEPT 101,NAME
101   FORMAT(A5)
      NAME=NAME+32
      CALL IFILE(22,NAME)
      READ(22,1)(MESS(I),I=1,7)
1     FORMAT(7A5)
      NAME=NAME+6
      CALL IFILE(23,NAME)
      READ(23)M
      READ(23)U,V,P
      READ(23)DT,RHOA,TIME,NCYCLE,MU,EPS1
      READ(23)KDMN,KDMX,LDMN,LDMX,DR,DZ,KMAX,LMAX
      WRITE(24,16)(MESS(I),I=1,7)
16    FORMAT(1H ,7A5)
      WRITE(24,17)NCYCLE
17    FORMAT(1H ,'CYCLE ',I)
      DO 20 L=1,LMAX
      WRITE(24,18)L
18    FORMAT(1H ,'L=',I3)
      WRITE(24,19)
19    FORMAT(1H ,'K       V             U             P')
      DO 20 K=1,KMAX
20    WRITE(24,21)K,UU(K,L),VV(K,L),PP(K,L)
21    FORMAT(1H ,I3,3X,3F)
      STOP
      END
```

```
      INTEGER BLN(2),BNDRY
      DIMENSION UU(18,82),VV(18,82),PP(18,82)
      DIMENSION ME(18,82)
      DIMENSION BNDRY(18,82)
      DIMENSION VZ(18,82),SI(18,82)
      DIMENSION DIMR(150),DIMZ(150)
      DIMENSION A(18,82),U(2500),V(2500),P(2500)
      DIMENSION MESS(7)
      DIMENSION BPX(100),BPY(100)
      DIMENSION XB(100),YB(100),NSEQ(100)
      DIMENSION XBP(50,2),YBP(50,2)
      DIMENSION M(2500)
      EQUIVALENCE(U,UU),(V,VV),(P,PP)
      EQUIVALENCE(M,ME)
      COMMON/CLP/IN,IXP,IYP,MN
      COMMON/STUFF/DR,DZ,KMAX,LMAX
      COMMON/WINDOW/WS,WCX,WCY,XL
      COMMON/CONLVL/NCLS,NCL
      INTEGER EMP,BND,FULL,OB
      DATA EMP,BND,FULL,OB/16,32,8,2048/
      TYPE 100
100   FORMAT(1H ,'INPUT FILE:',$)
      ACCEPT 101,NAME
      NAME=NAME+32
      CALL IFILE(22,NAME)
101   FORMAT(A5)
      READ(22,1)(MESS(I),I=1,7)
1     FORMAT(7A5)
      READ(22,2)KMAX,LMAX
2     FORMAT(2I)
      READ(22,3)SMIN,EPS1
3     FORMAT(2F)
      READ(22,4)D,D,MU,RHOA
4     FORMAT(4F)
      READ(22,5)DR
5     FORMAT(F)
      READ(22,5)DZ
      READ(22,6)D
6     FORMAT(I)
      READ(22,3)BPX(1),BPY(1)
      MM=2
7     READ(22,3)BPX(MM),BPY(MM)
      IF(BPX(1).EQ.BPX(MM).AND.BPY(1).EQ.BPY(MM))GO TO 8
      MM=MM+1
      GO TO 7
8     MM=MM-1
      DO 9 I=1,6
9     READ(22,6)D
      N=1
10    READ(22,11)XB(N),YB(N),NSEQ(N)
11    FORMAT(2F,I)
      IF(NSEQ(N).EQ.1)GO TO 12
      N=N+1
      GO TO 10
12    KMAXZZ=KMAX-2
      LMAXZZ=LMAX-2
      KMAXZ=KMAX-1
```

```
      LMAXZ=LMAX-1
      XBMAX=DR*KMAXZZ
      YBMAX=DZ*LMAXZZ
      N=0
      MM=0
13    N=N+1
      MM=MM+1
      XBP(MM,1)=YB(N)
      YBP(MM,1)=XB(N)
      IF(NSEQ(N).EQ.0)GO TO 13
      XBP(MM,1)=YB(N)
      YBP(MM,1)=XB(N)
      MM=MM+1
      XBP(MM,1)=YBMAX
      YBP(MM,1)=XBMAX
      BLN(1)=MM
      MM=0
14    MM=MM+1
      N=N+1
      XBP(MM,2)=YB(N)
      YBP(MM,2)=XB(N)
      IF(NSEQ(N).EQ.1)GO TO 15
      GO TO 14
15    XBP(MM,2)=YB(N)
      YBP(MM,2)=XB(N)
      BLN(2)=MM
      NAME=NAME+6
      CALL IFILE(23,NAME)
      READ(23)M
      READ(23)U,V,P
      READ(23)DT,RHOA,TIME,NCYCLE,MU,EPS1
      READ(23)KDMN,KDMX,LDMN,LDMX,DR,DZ,KMAX,LMAX
202   TYPE 16
16    FORMAT(1H ,'LOOK AT:',$)
      ACCEPT 17,IVAR
17    FORMAT(A1)
      IF(IVAR.EQ.' ')CALL EXIT
      TYPE 18
18    FORMAT(1H ,'START AT L=',$)
      ACCEPT 19,L1
19    FORMAT(I)
      TYPE 20
20    FORMAT(1H ,'LIST TO L=',$)
      ACCEPT 19,L2
      ACCEPT 203,NDUM
203   FORMAT(I)
      TYPE 201,NCYCLE
201   FORMAT(1H ,'CYCLE ',I3)
      IF(IVAR.EQ.'P')GO TO 24
      IF(IVAR.EQ.'V')GO TO 27
      DO 22 L=L1,L2
      TYPE 21,L
21    FORMAT(1H ,'L=',I3)
      DO 22 K=1,18
22    TYPE 23,K,UU(K,L)
23    FORMAT(1H ,'K=',I3,' U(K,L)=',F12.7)
      GO TO 202
```

```
24      DO 25 L=L1,L2
        TYPE 21,L
        DO 25 K=1,18
25      TYPE 26,K,PP(K,L)
26      FORMAT(1H ,'K=',I3,' P(K,L)=',F15.7)
        GO TO 202
27      DO 28 L=L1,L2
        TYPE 21,L
        DO 28 K=1,18
28      TYPE 29,K,VV(K,L)
29      FORMAT(1H ,'K=',I3,' V(K,L)=',F12.7)
        GO TO 202
        END
```

```
;       READ FROM FILE FOR01.DAT AND APPEND TO FILE
;       ERROR.DAT. DELETE FOR01.DAT WHEN DONE.....
        TITLE
        APPEND TO FILE
        .TYPE   APFILE
        .TERMINAL       APFILE
APFILE: BLOCK 23
FNAM1:  ASCIZ/FOR01.DAT/
FNAM2:  ASCIZ/ERROR.DAT;1/
JFN1:   0
        .MOVEI  17,SAVR+17
        MOVEI   17,SAVR
        BLT     17,SAVR+16
        GTJFN   2,FNAM1
        JRST    1,(1B2!1B17)
        OPENF
        HALTF
        .MOVEI  1,JFN17
        .MOVE   2,[444B5!1B19]
        SETSTS
        HALTF
        GTJFN   2,FNAM2
        JRST    1,(1B3!1B17)
        OPENF
        HALTF
        .MOVEM  1,JFN2#
        .MOVE   2,[4435!1B22]
        SETSTS
        HALTF
LOOP:   .MOVE   1,JFN1
        BIN
        MOVE    3,2
        SIOIS
        DONE    2,1000
        JRST    DONE
        .MOVE   2,3
        .MOVE   1,JFN2
        BOUT
        JRST    LOOP
DONE:   .RLJFN  1,JFN1
        .ELF
        HALTF
        MOVEI   17,SAVR
        BLT     17,17
        JRST    1@,(1@)
        END
```

APPENDIX VII

LISTING OF INPUT DATA FOR BALL VALVE

```
STARR-EDWARDS BALL VALVE(EXTENDED)
18,92,1,0
.01,.2,0.,0.
0.,0.,.0032,1.06
.2
.2
2,2,17,2,91
0.,0.
1.7,0.
1.7,8.8
1.58,9.31
1.56,9.56
1.58,9.82
1.62,10.07
1.736,10.53
1.736,10.78
1.683,11.04
1.524,11.42
0.89,11.48
0.846,11.74
0.90,11.99
1.397,12.10
1.46,12.35
1.65,12.73
2.984,14.13
3.175,14.51
3.2,14.89
3.2,18.
0.,18.
0.,11.255
.198,11.233
.388,11.171
.561,11.072
.700,10.938
.827,10.777
.908,10.595
.949,10.399
.949,10.209
.908,10.105
.827,9.823
.700,9.662
.561,9.528
.388,9.429
.198,9.367
0.,9.345
0.,0.
0
1,18,1,92,0.,-10.,0,0
2,10,1,1,0.,-10.,0,1
2,17,92,92,0.,-7.57,1,0
0
38,.05,22.45,.1
1.7,-.0001
1.7,8.8
1.58,9.31
1.56,9.56
1.58,9.82
```

```
1.62,10.07
1.736,10.53
1.736,10.76
1.685,11.04
1.524,11.42
0.89,11.46
0.846,11.74
0.9,11.99
1.397,12.19
1.46,12.35
1.65,12.73
2.984,14.13
3.175,14.51
3.202,14.59,2
-.0001,11.255
0.196,11.253
0.386,11.171
0.561,11.072
0.700,10.938
0.827,10.777
0.908,10.595
0.949,10.399
0.949,10.209
0.908,10.105
0.627,9.823
0.700,9.662
0.561,9.528
0.386,9.429
0.196,9.367
-0.0001,9.345,1
1,0.,3.2,6.,18.,.2885,.9
1,18,1,92
.0005
20
.1
.0001
.45
N
.01
20
20
4
```

APPENDIX VIII

LISTING OF DATA FOR BALL VALVE SOLUTION

STARR-EDWARDS BALL VALVE(EXTENDED)
CYCLE 277
L= 1

K	U	V	P
1	0.0000000	-35.2557037	0.0000000
2	-0.0075029	-35.2557037	0.0000000
3	-0.0131090	-35.3133917	0.0000000
4	-0.0182086	-35.3263162	0.0000000
5	-0.0226600	-35.3329953	0.0000000
6	-0.0265614	-35.3382938	0.0000000
7	-0.0304269	-35.3437446	0.0000000
8	-0.0342607	-35.3448391	0.0000000
9	-0.0341043	-34.7626600	0.0000000
10	0.0341043	0.0682824	0.0000000
11	0.0000000	0.0000000	0.0000000
12	0.0000000	0.0000000	0.0000000
13	0.0000000	0.0000000	0.0000000
14	0.0000000	0.0000000	0.0000000
15	0.0000000	0.0000000	0.0000000
16	0.0000000	0.0000000	0.0000000
17	0.0000000	0.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 2

K	U	V	P
1	0.0000000	-35.2409236	-25.9982292
2	-0.0075029	-35.2409236	-25.9982292
3	-0.0131090	-35.3010952	-26.1790384
4	-0.0182086	-35.3151243	-26.5270531
5	-0.0226600	-35.3228718	-27.0173922
6	-0.0265614	-35.3290867	-27.6058041
7	-0.0304269	-35.3348572	-28.2245919
8	-0.0342607	-35.3361804	-28.7877768
9	-0.0341043	-34.7583905	-29.2185578
10	0.0341043	0.0000000	-29.4657301
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 3

K	U	V	P
1	0.0000000	-35.2413002	-58.8076420
2	-0.0000038	-35.2413002	-58.8076420
3	0.0015547	-35.3034645	-58.9373298
4	0.0030556	-35.3178225	-59.1852829
5	0.0043301	-35.3254676	-59.5273599
6	0.0050547	-35.3311151	-59.9238522
7	0.0046078	-35.3355452	-60.3163153
8	0.0010023	-35.3332594	-60.6190503
9	0.0001551	-34.7578723	-60.6135242
10	-0.0001551	0.0000000	-60.6192805
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000

14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 4

K	U	V	P
1	0.0000000	-35.2408037	-91.6542172
2	-0.0005199	-35.2408037	-91.6542172
3	0.0005913	-35.3050075	-91.7389899
4	0.0018273	-35.3199217	-91.9006832
5	0.0030747	-35.3277832	-92.1191096
6	0.0041225	-35.3333227	-92.3628353
7	0.0048516	-35.3374459	-92.5882347
8	0.0066662	-35.3363112	-92.7424811
9	0.0002392	-34.7522557	-92.8500302
10	-0.0002392	0.0000000	-92.8584397
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 5

K	U	V	P
1	0.0000000	-35.2393445	-124.3235874
2	-0.0010812	-35.2393445	-124.3235874
3	-0.0004366	-35.3056680	-124.3757321
4	0.0005936	-35.3214685	-124.4758968
5	0.0019466	-35.3299677	-124.6103379
6	0.0034631	-35.3359006	-124.7581425
7	0.0050768	-35.3402894	-124.8953906
8	0.0063552	-35.3389189	-125.0059348
9	0.0003226	-34.7471148	-125.0681379
10	-0.0003226	0.0000000	-125.0796080
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 6

K	U	V	P
1	0.0000000	-35.2367852	-156.8759889
2	-0.0017108	-35.2367852	-156.8759889
3	-0.0015540	-35.3053719	-156.9040962
4	-0.0007146	-35.3224441	-156.9606797
5	0.0007568	-35.3320152	-157.0384786
6	0.0025955	-35.3386700	-157.1260919
7	0.0044356	-35.3433210	-157.2108832
8	0.0060378	-35.3418738	-157.2789331
9	0.0004056	-34.7424567	-157.3213567
10	-0.0004056	0.0000000	-157.3354588

11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 7

K	U	V	P
1	0.0000000	-35.2331959	-189.3525612
2	-0.0023061	-35.2331959	-189.3525612
3	-0.0026064	-35.3041831	-189.3615429
4	-0.0019422	-35.3228927	-189.3844683
5	-0.0003663	-35.3339322	-189.4205161
6	0.0017443	-35.3415921	-189.4663316
7	0.0039077	-35.3466470	-189.5146244
8	0.0057533	-35.3451062	-189.5568450
9	0.0004883	-34.7382482	-189.5867535
10	-0.0004883	0.0000000	-189.6029790
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 8

K	U	V	P
1	0.0000000	-35.2292165	-221.7791439
2	-0.0025798	-35.2292165	-221.7791439
3	-0.0031120	-35.3026159	-221.7707601
4	-0.0025603	-35.3231080	-221.7639112
5	-0.0009363	-35.3358284	-221.7651927
6	0.0013630	-35.3446953	-221.7787509
7	0.0037333	-35.3502281	-221.8022016
8	0.0056511	-35.3484869	-221.8287797
9	0.0005705	-34.7343184	-221.8519454
10	-0.0005705	0.0000000	-221.8695157
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 9

K	U	V	P
1	0.0000000	-35.2266345	-254.1678143
2	-0.0019593	-35.2266345	-254.1678143
3	-0.0021293	-35.3020564	-254.1429256
4	-0.0015640	-35.3238367	-254.1079488
5	-0.0001432	-35.3378758	-254.0776287
6	0.0019433	-35.3478353	-254.0643033
7	0.0041479	-35.3538290	-254.0705991

8	0.0058341	-35.3517771	-254.0892553
9	0.0006522	-34.7303998	-254.1100872
10	-0.0006522	0.0000000	-254.1283411
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 10

K	U	V	P
1	0.0000000	-35.2290750	-286.5192001
2	0.0004745	-35.2290750	-286.5192001
3	0.0018432	-35.3052828	-286.4793879
4	0.0026255	-35.3265159	-286.4189599
5	0.0032137	-35.3402659	-286.3604803
6	0.0041310	-35.3505190	-286.3246132
7	0.0053742	-35.3568736	-286.3191259
8	0.0063303	-35.3545625	-286.3363367
9	0.0007330	-34.7261980	-286.3594410
10	-0.0007330	0.0000000	-286.3775831
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 11

K	U	V	P
1	0.0000000	-35.2426425	-318.8180928
2	0.0059613	-35.2426425	-318.8180928
3	0.0108315	-35.3169449	-318.7699024
4	0.0121493	-35.3335336	-318.6936048
5	0.0107562	-35.3432108	-318.6156344
6	0.0087228	-35.3516936	-318.5625193
7	0.0075305	-35.3581905	-318.5498644
8	0.0070213	-35.3561960	-318.5713035
9	0.0008126	-34.7215280	-318.6013107
10	-0.0008126	0.0000000	-318.6186370
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 12

K	U	V	P
1	0.0000000	-35.2760255	-351.0333714
2	0.0157937	-35.2760255	-351.0333714
3	0.0270152	-35.3437353	-350.9921883
4	0.0293990	-35.3484020	-350.9229612

5	0.0243600	-35.3470109	-350.8455013
6	0.0165756	-35.3495941	-350.7860047
7	0.0105188	-35.3557192	-350.7694159
8	0.0075182	-35.3556961	-350.7981715
9	0.0008908	-34.7165678	-350.8393545
10	-0.0008908	0.0000000	-350.8550045
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 13

K	U	V	P
1	0.0000000	-35.3393737	-383.1229997
2	0.0307032	-35.3393737	-383.1229997
3	0.0517871	-35.3937209	-383.1160930
4	0.0561562	-35.3756569	-383.0950414
5	0.0456579	-35.3523087	-383.0551249
6	0.0284295	-35.3418253	-383.0081974
7	0.0139093	-35.3462588	-382.9893213
8	0.0070234	-35.3516214	-383.0227335
9	0.0009674	-34.7122382	-383.0740531
10	-0.0009674	0.0000000	-383.0874567
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 14

K	U	V	P
1	0.0000000	-35.4410753	-415.0495161
2	0.0498085	-35.4410753	-415.0495161
3	0.0841169	-35.4741767	-415.1130323
4	0.0920375	-35.4201984	-415.1976858
5	0.0750737	-35.3605587	-415.2515461
6	0.0445843	-35.3259444	-415.2488638
7	0.0168991	-35.3254613	-415.2280693
8	0.0043263	-35.3419695	-415.2510804
9	0.0010419	-34.7105924	-415.2986449
10	-0.0010419	0.0000000	-415.3092359
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 15

K	U	V	P
1	0.0000000	-35.5820086	-446.8123731

2	0.0693563	-35.5820086	-446.8123731
3	0.1184604	-35.5874881	-446.9801788
4	0.1322621	-35.4856245	-447.2300340
5	0.1102661	-35.3746358	-447.4452887
6	0.0643802	-35.3008628	-447.5333807
7	0.0184666	-35.2884606	-447.5109917
8	-0.0018255	-35.3243276	-447.4878614
9	0.0011141	-34.7148474	-447.4913222
10	-0.0011142	0.0000000	-447.4988435
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 16

K	U	V	P
1	0.0000000	-35.7481584	-478.4891163
2	0.0818991	-35.7481584	-478.4891163
3	0.1430397	-35.7253037	-478.7695822
4	0.1653096	-35.5711285	-479.2150561
5	0.1440054	-35.3990287	-479.6478678
6	0.0853848	-35.2692086	-479.8861001
7	0.0179313	-35.2318600	-479.8701335
8	-0.0115784	-35.2967545	-479.7309269
9	0.0011837	-34.7283579	-479.6130364
10	-0.0011837	0.0000000	-479.6172832
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 17

K	U	V	P
1	0.0000000	-35.9045625	-510.2625419
2	0.0769643	-35.9045625	-510.2625419
3	0.1400217	-35.8624723	-510.6146117
4	0.1716546	-35.6667407	-511.2139736
5	0.1617890	-35.4383834	-511.8680452
6	0.1020792	-35.2403719	-512.3191704
7	0.0163551	-35.1584375	-512.3422821
8	-0.0218047	-35.2596991	-511.9657297
9	0.0012499	-34.7515583	-511.6049205
10	-0.0012499	0.0000000	-511.6059561
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 18

K	U	V	P
1	0.0000000	-35.9950975	-542.4087797
2	0.0439717	-35.9950975	-542.4087797
3	0.0892037	-35.9539613	-542.7357484
4	0.1254494	-35.7476365	-543.3447440
5	0.1400227	-35.4925394	-544.1168700
6	0.1036897	-35.2327429	-544.8094983
7	0.0194716	-35.0870207	-544.9520130
8	-0.0229936	-35.2185551	-544.1555534
9	0.0013123	-34.7760033	-543.4001316
10	-0.0013123	0.0000000	-543.3986862
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 19

K	U	V	P
1	0.0000000	-35.9514102	-575.2259125
2	-0.0231934	-35.9514102	-575.2259125
3	-0.0245368	-35.9386485	-575.4125793
4	0.0009593	-35.7702058	-575.8002777
5	0.0467369	-35.5468553	-576.4356558
6	0.0708051	-35.2715619	-577.2743081
7	0.0425161	-35.0707028	-577.6639660
8	0.0041674	-35.1854555	-576.2270172
9	0.0013701	-34.7752225	-574.9551733
10	-0.0013701	0.0000000	-574.9524985
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 20

K	U	V	P
1	0.0000000	-35.7093308	-608.9381878
2	-0.1224383	-35.7093308	-608.9381878
3	-0.2025060	-35.7522771	-608.9038257
4	-0.2160406	-35.6748153	-608.8500698
5	-0.1515240	-35.5606493	-608.9779590
6	-0.0281433	-35.3767302	-609.5923562
7	0.1159470	-35.2245048	-610.2796797
8	0.0898108	-35.1768613	-608.0556218
9	0.0014228	-34.6946239	-606.3213353
10	-0.0014228	0.0000000	-606.3169469
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000

16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 21

K	U	V	P
1	0.0000000	-35.2265522	-643.6794479
2	-0.2428349	-35.2265522	-643.6794479
3	-0.4274152	-35.3463557	-643.4273724
4	-0.5158106	-35.3996880	-642.8332706
5	-0.4720987	-35.4650758	-642.0731176
6	-0.2339792	-35.5382036	-641.7500381
7	0.2865550	-35.7516021	-642.3495193
8	0.2682112	-35.2029610	-639.5371188
9	0.0014710	-34.4476208	-637.7373715
10	-0.0014710	0.0000000	-637.7203320
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 22

K	U	V	P
1	0.0000000	-34.4952852	-679.5053636
2	-0.3671225	-34.4952852	-679.5053636
3	-0.6655240	-34.7058758	-679.1723673
4	-0.8572167	-34.9054176	-678.2025699
5	-0.8900878	-35.1844881	-676.4011388
6	-0.5712988	-35.6964756	-674.1460492
7	0.6005464	-36.9279411	-673.3530317
8	0.5585609	-35.2519568	-670.8787014
9	0.0015149	-33.9297220	-669.7760527
10	-0.0015149	0.0000000	-669.7063152
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 23

K	U	V	P
1	0.0000000	-33.5426595	-716.3895969
2	-0.4778434	-33.5426595	-716.3895969
3	-0.8767802	-33.8575944	-716.2130330
4	-1.1703163	-34.2044853	-715.3574754
5	-1.3113833	-34.6908428	-713.0187978
6	-0.9876788	-35.7666381	-708.3091666
7	1.0523841	-38.9757711	-703.7328859
8	0.9365956	-35.2910344	-703.2048483
9	0.0015558	-33.0590799	-703.5322343
10	-0.0015558	0.0000000	-703.3305843
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000

13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 24

K	U	V	P
1	0.0000000	-32.4163906	-754.2335917
2	-0.5647062	-32.4163906	-754.2335917
3	-1.0324574	-32.8597095	-754.4009988
4	-1.3867374	-33.3756447	-754.2470549
5	-1.5740538	-34.0774149	-752.7400632
6	-1.2785430	-35.7471529	-747.2712626
7	1.5185734	-41.7966399	-736.6449837
8	1.3138046	-35.3060547	-739.0522887
9	0.0015950	-31.8364639	-740.5057628
10	-0.0015950	0.0000000	-740.0949527
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 25

K	U	V	P
1	0.0000000	-31.1632540	-792.9777135
2	-0.6281805	-31.1632540	-792.9777135
3	-1.1317574	-31.7718003	-793.4378986
4	-1.4612365	-32.5296983	-794.0670703
5	-1.5387375	-33.5733995	-794.5629363
6	-1.1015646	-35.8929683	-792.8811689
7	1.7755537	-44.7370127	-777.4902587
8	1.5620841	-35.3512883	-781.0705011
9	0.0016342	-30.3822085	-781.6339438
10	-0.0016342	0.0000000	-781.0338442
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 26

K	U	V	P
1	0.0000000	-29.8191313	-832.7893470
2	-0.6737169	-29.8191313	-832.7893470
3	-1.1969293	-30.6274146	-833.2990263
4	-1.4699228	-31.7255236	-833.9636777
5	-1.2800676	-33.3725072	-835.2092807
6	-0.2122785	-36.7970038	-838.8087605
7	1.6861749	-46.7714906	-828.4534295
8	1.6064524	-35.5268575	-828.3775453
9	0.0016747	-28.8866356	-825.9410341

10	-0.0016747	0.0000000	-825.2897491
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 27

K	U	V	P
1	0.0000000	-28.4227613	-873.9951108
2	-0.6998858	-28.4227613	-873.9951108
3	-1.2361363	-29.4482660	-874.5191996
4	-1.4795000	-30.9412826	-874.6690606
5	-1.0983221	-33.3875921	-873.7977971
6	1.1044203	-39.0025448	-874.6188958
7	1.4006086	-47.2974978	-882.0260991
8	1.5081670	-35.8602356	-875.7069562
9	0.0017178	-27.4828948	-870.9813212
10	-0.0017178	0.0000000	-870.4006378
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 28

K	U	V	P
1	0.0000000	-27.0307504	-916.5451978
2	-0.6977563	-27.0307504	-916.5451978
3	-1.2204068	-28.2887425	-917.4204548
4	-1.4378544	-30.1945001	-917.8811987
5	-1.0275985	-33.4478721	-915.5198936
6	1.9169836	-42.0481306	-909.5184211
7	1.2468875	-46.9171774	-925.3187631
8	1.3992354	-36.2182587	-920.1577757
9	0.0017646	-26.1809323	-915.4815753
10	-0.0017646	0.0000000	-914.9829601
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 29

K	U	V	P
1	0.0000000	-25.6971717	-959.7103451
2	-0.6685963	-25.6971717	-959.7103451
3	-1.1325859	-27.2269551	-961.1601089
4	-1.2441632	-29.6099652	-963.0578239
5	-0.7824071	-33.6224221	-963.0086650
6	1.7579073	-44.6990877	-953.7183051

7	1.2351853	-46.6687743	-964.5730336
8	1.3334380	-36.5162920	-964.2826398
9	0.0018163	-24.9405013	-959.7451691
10	-0.0018163	0.0000000	-959.2961301
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 30

K	U	V	P
1	0.0000000	-24.4336925	-1002.6408315
2	-0.6336090	-24.4336925	-1002.6408315
3	-1.0189235	-26.2934852	-1004.3390517
4	-0.9480336	-29.2899411	-1007.2085403
5	-0.2140049	-34.1928314	-1010.7652706
6	0.9989842	-46.0016292	-1006.8379019
7	1.1474224	-47.0146373	-1006.8550755
8	1.3108338	-36.8710700	-1009.3036715
9	0.0018741	-23.7213038	-1003.9001287
10	-0.0018741	0.0000000	-1003.4738344
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 31

K	U	V	P
1	0.0000000	-23.2039552	-1045.2207227
2	-0.6168104	-23.2039552	-1045.2207227
3	-0.9522980	-25.4377551	-1046.7000727
4	-0.7476373	-29.1571893	-1049.0059902
5	0.3953287	-35.2879468	-1051.8103514
6	0.2838942	-45.9680870	-1059.2034322
7	0.8724453	-47.7106962	-1052.3784489
8	1.3487501	-37.5205923	-1054.6502147
9	0.0019397	-22.4668719	-1048.0564975
10	-0.0019397	0.0000000	-1047.6196699
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 32

K	U	V	P
1	0.0000000	-21.9601812	-1087.9624498
2	-0.6239120	-21.9601812	-1087.9624498
3	-0.9473307	-24.5935045	-1089.2295421

4	-0.6929861	-29.0861577	-1090.9900398
5	0.5456726	-36.5081489	-1092.8204518
6	-0.0649632	-45.4133928	-1102.9154391
7	0.4604113	-48.2745085	-1100.3299852
8	1.4751910	-38.6867214	-1100.0723534
9	0.0020147	-21.0946059	-1093.4634928
10	-0.0020147	0.0000000	-1092.9610164
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 33

K	U	V	P
1	0.0000000	-20.6628442	-1131.3066774
2	-0.6507896	-20.6628442	-1131.3066774
3	-0.9865006	-23.7150868	-1132.5740491
4	-0.7300576	-29.0021066	-1134.4149292
5	0.1784079	-37.3405188	-1137.1066750
6	-0.2307453	-45.0010747	-1144.8925131
7	0.0704051	-48.5636894	-1150.1829468
8	1.6720768	-40.4250029	-1147.0397238
9	0.0021016	-19.5387861	-1143.3270623
10	-0.0021016	0.0000000	-1142.7005057
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 34

K	U	V	P
1	0.0000000	-19.2816239	-1175.4457515
2	-0.6928425	-19.2816239	-1175.4457515
3	-1.0599559	-22.7669269	-1176.9867365
4	-0.8595240	-28.8216107	-1179.4761990
5	-0.4116542	-37.6096487	-1183.7304246
6	-0.4938803	-44.8210262	-1191.9400890
7	-0.1077552	-48.8978658	-1201.2344461
8	1.8136792	-42.4803708	-1199.4885050
9	0.0022030	-17.8510444	-1201.5936386
10	-0.0022030	0.0000000	-1200.8759428
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 35

K	U	V	P

1	0.0000000	-17.7929228	-1220.2209842
2	-0.7467107	-17.7929228	-1220.2209842
3	-1.1619476	-21.7188766	-1222.3403484
4	-1.0806561	-28.4575228	-1226.2305666
5	-1.0359859	-37.3549659	-1233.3510399
6	-0.8707759	-44.7773270	-1242.9710806
7	-0.0295833	-49.6601120	-1253.7553955
8	1.7072297	-44.3490993	-1260.5380502
9	0.0023214	-16.2629366	-1268.0795279
10	-0.0023214	0.0000000	-1267.4910810
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 36

K	U	V	P
1	0.0000000	-16.1709398	-1265.1441109
2	-0.8134976	-16.1709398	-1265.1441109
3	-1.2986379	-20.5333099	-1268.0530970
4	-1.3913201	-27.8301869	-1273.8465194
5	-1.5874746	-36.7364835	-1284.2539747
6	-1.2172970	-44.8390037	-1296.8277980
7	0.1142202	-50.8944385	-1309.7818059
8	1.3214640	-45.6698254	-1325.8278791
9	0.0024596	-15.0352031	-1335.6289317
10	-0.0024596	0.0000000	-1335.3836585
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 37

K	U	V	P
1	0.0000000	-14.3826196	-1309.6050781
2	-0.8968298	-14.3826196	-1309.6050781
3	-1.4814356	-19.1598065	-1313.4324041
4	-1.7785331	-26.8846549	-1321.4357619
5	-2.0173961	-35.9587729	-1335.0755398
6	-1.4314029	-45.0451475	-1351.8084821
7	0.1511175	-52.3638651	-1368.4610513
8	0.9112755	-46.5149581	-1389.4221981
9	0.0026190	-14.1906867	-1398.6589064
10	-0.0026190	0.0000000	-1398.6970719
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000

18	0.0000000	-10.0000000	0.0000000

L= 38

K	U	V	P
1	0.0000000	-12.3849273	-1353.1096515
2	-1.0016974	-12.3849273	-1353.1096515
3	-1.7180967	-17.5409313	-1357.8537155
4	-2.2012109	-25.6214842	-1368.0795070
5	-2.3440472	-35.1702794	-1384.9536565
6	-1.5254509	-45.4373851	-1406.3140240
7	0.0778529	-53.8390972	-1428.1783861
8	0.7445161	-47.2483639	-1451.1453596
9	0.0028005	-13.5022359	-1461.9771254
10	-0.0028005	0.0000000	-1462.1205118
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 39

K	U	V	P
1	0.0000000	-10.1221569	-1395.6821014
2	-1.1344352	-10.1221569	-1395.6821014
3	-2.0115207	-15.6196052	-1401.1555214
4	-2.6264815	-24.0829956	-1413.2172105
5	-2.6235399	-34.4271455	-1433.2521011
6	-1.6023830	-45.9928367	-1459.3802069
7	-0.1038604	-55.1862888	-1488.7393874
8	0.6535582	-48.0518036	-1515.7446687
9	0.0030033	-12.8991486	-1531.7628954
10	-0.0030033	0.0000000	-1531.9501658
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 40

K	U	V	P
1	0.0000000	-7.5282578	-1438.3790132
2	-1.3002142	-7.5282578	-1438.3790132
3	-2.3605570	-13.3437280	-1444.2135815
4	-3.0644949	-22.2984098	-1457.3631298
5	-2.9405016	-33.6974893	-1480.0064501
6	-1.8254875	-46.5824150	-1510.9451164
7	-0.5033405	-56.3007817	-1549.8224322
8	0.1696290	-48.7031136	-1583.7805208
9	0.0032254	-12.7482397	-1597.7448274
10	-0.0032254	0.0000000	-1598.0379891
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000

12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 44

K	U	V	P
1	0.0000000	6.4780799	-1679.5670290
2	-1.8887191	6.4780799	-1679.5670290
3	-4.1319691	0.3192238	-1690.4580642
4	-5.6841335	-11.3851216	-1704.3888245
5	-5.8823688	-28.4698833	-1720.3617704
6	-5.0891345	-46.3887230	-1743.6368067
7	-3.3509181	-59.5604934	-1768.6794933
8	-1.5857876	-52.4907507	-1777.0159294
9	0.0042447	-17.0354402	-1772.6088923
10	-0.0042447	0.0000000	-1772.2585660
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 45

K	U	V	P
1	0.0000000	14.6186573	-1804.0536579
2	-4.0747339	14.6186573	-1804.0536579
3	-4.7249329	3.8962002	-1802.3449604
4	-6.3056460	-7.6042862	-1827.0998870
5	-6.6548575	-26.2749799	-1825.3202040
6	-5.7921015	-45.8741847	-1831.8166819
7	-3.7414213	-60.7501090	-1840.5311045
8	-1.6915402	-54.1282587	-1828.9568324
9	0.0045110	-18.6245121	-1813.5686454
10	-0.0045110	0.0000000	-1813.0897035
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 46

K	U	V	P
1	0.0000000	14.9199745	-1925.3411752
2	-0.1552941	14.9199745	-1925.3411752
3	-8.5136396	15.1374444	-1996.7666876
4	-5.9754948	-7.2509249	-1997.8735831
5	-6.9940013	-23.4098312	-1951.2503271
6	-6.1945283	-45.2143043	-1937.7044451
7	-3.7856869	-62.2576088	-1932.6258127
8	-1.4567509	-56.0597859	-1898.1218387

9	0.0059688	-19.9962653	-1852.5101852
10	-0.0036950	0.0000000	-1852.8769959
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 47

K	U	V	P
1	0.0000000	2.4666596	-1886.8537296
2	6.2219077	2.4666596	-1886.8537296
3	10.8261753	4.8435224	-2029.6445165
4	-3.1583171	5.1933441	-2101.9010854
5	-6.6262764	-18.5505470	-2076.4189635
6	-6.2950439	-44.1162154	-2052.2238363
7	-3.4207452	-64.2549530	-2057.8970250
8	-0.4249512	-58.7659344	-2026.3216694
9	-0.0544809	-20.3420555	-1783.1510354
10	0.0000000	-10.0000000	0.0000000
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 48

K	U	V	P
1	0.0000000	-0.6890116	-1815.1779708
2	1.5726660	-0.6890116	-1815.1779708
3	4.3794715	0.0422457	-1962.4202375
4	0.7637725	7.7713452	-2180.5690464
5	-5.4736280	-11.6470918	-2196.4564596
6	-6.4636752	-41.8065302	-2157.4890849
7	-3.0613174	-66.7980894	-2230.1036168
8	3.0563267	-64.8898542	-2337.9675348
9	-0.1720533	-17.3166517	-2493.0419636
10	0.0000000	-10.0000000	0.0000000
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 49

K	U	V	P
1	0.0000000	-10.0000000	0.0000000
2	0.0000000	-10.0000000	0.0000000
3	-7.9295501	-10.0000000	0.0000000
4	7.7525625	-7.8859668	-2202.6059826
5	-3.1094193	-1.4554800	-2233.2870078

6	-7.6249190	-36.1053802	-2229.6427984
7	-4.6389894	-68.6761588	-2420.2963759
8	0.5314005	-69.7513148	-2746.6394801
9	-0.1014509	-16.7239975	-3003.3855189
10	0.0000000	-10.0000000	0.0000000
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 50

K	U	V	P
1	0.0000000	-10.0000000	0.0000000
2	0.0000000	-10.0000000	0.0000000
3	0.0000000	-10.0000000	0.0000000
4	0.8679453	-10.0000000	0.0000000
5	-1.3785837	0.8527263	-2340.5131374
6	-10.6997289	-25.4495509	-2329.2128046
7	-9.1591323	-68.4187494	-2536.8452494
8	-3.7352296	-74.1907001	-3034.1101859
9	0.6393637	-20.9045600	-3282.7533085
10	0.0000000	-10.0000000	0.0000000
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 51

K	U	V	P
1	0.0000000	-10.0000000	0.0000000
2	0.0000000	-10.0000000	0.0000000
3	0.0000000	-10.0000000	0.0000000
4	0.0000000	-10.0000000	0.0000000
5	3.8932115	-10.0000000	0.0000000
6	-14.0690284	-6.3685837	-2658.6879839
7	-15.9220656	-63.8478756	-2505.6760062
8	-7.7183429	-80.5835098	-3166.4481808
9	0.2538849	-28.3921167	-5356.4116408
10	0.0000000	-10.0000000	0.0000000
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 52

K	U	V	P
1	0.0000000	-10.0000000	0.0000000
2	0.0000000	-10.0000000	0.0000000

3	0.0000000	-10.0000000	0.0000000
4	0.0000000	-10.0000000	0.0000000
5	11.0363042	-10.0000000	0.0000000
6	-8.7338346	13.1279802	-2241.8237899
7	-24.8936653	-44.6384065	-2241.7419267
8	-34.4863281	-66.4308406	-1671.3421096
9	-4.6010386	-55.6848310	1349.2407699
10	0.0000000	17.2093526	0.0000000
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 53

K	U	V	P
1	0.0000000	-10.0000000	0.0000000
2	0.0000000	-10.0000000	0.0000000
3	0.0000000	-10.0000000	0.0000000
4	0.0000000	-10.0000000	0.0000000
5	-1.8498273	-10.0000000	0.0000000
6	0.6648684	10.7269044	-1939.8427431
7	-9.8963916	-33.2441505	-1934.1238823
8	-9.0742799	-65.8013768	-880.2858298
9	-6.9660420	-56.7313954	-41.0856515
10	10.8955994	-0.8981968	35.3639945
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 54

K	U	V	P
1	0.0000000	-10.0000000	0.0000000
2	0.0000000	-10.0000000	0.0000000
3	0.0000000	-10.0000000	0.0000000
4	0.0000000	-10.0000000	0.0000000
5	-6.2145130	-10.0000000	0.0000000
6	3.5503850	1.2398723	-1987.8413763
7	10.7053762	-41.7020530	-1805.8980119
8	3.7512957	-59.9670313	-675.4626115
9	-3.9708870	-49.0022462	454.0255521
10	2.6689086	-7.4749160	659.0322559
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 55

K	U	V	P
1	0.0000000	-10.0000000	0.0000000
2	0.0000000	-10.0000000	0.0000000
3	0.0000000	-10.0000000	0.0000000
4	0.0000000	-10.0000000	0.0000000
5	-5.7409237	91.2245519	0.0000000
6	14.2916791	-19.7608359	-5826.4608730
7	31.5379404	-63.1230831	-2226.7240894
8	22.9458012	-55.5735648	-11.3308048
9	2.2918534	-30.0384369	924.9372785
10	-1.1208065	-4.1346366	1077.7477969
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 56

K	U	V	P
1	0.0000000	-10.0000000	0.0000000
2	0.0000000	-10.0000000	0.0000000
3	0.0000000	-10.0000000	0.0000000
4	-91.7462833	66.7213745	0.0000000
5	57.7631138	-53.4464046	1611.7630203
6	87.6653433	-65.8341736	581.6073869
7	67.3591949	-56.9175430	-521.4857389
8	37.6721635	-33.9734089	239.1475593
9	2.2583450	2.7057737	758.0634360
10	-1.7911037	-0.1252179	1135.0159210
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 57

K	U	V	P
1	0.0000000	3.1215359	0.0000000
2	0.0000000	3.1215359	0.0000000
3	-66.2976310	-0.2734088	0.0000000
4	67.8119279	-67.7059501	4521.3531752
5	82.1846686	-89.2536588	1773.7367053
6	71.6161616	-72.3617412	76.5063718
7	47.4819653	-43.6177567	-680.0063182
8	-2.2319803	12.2523122	-276.3591897
9	-0.7900204	1.4577353	314.1016443
10	0.2892652	-1.1882713	330.8744174
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000

14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 61

K	U	V	P
1	0.0000000	-100.9282577	6299.4255920
2	-1.4269879	-100.9282577	6299.4255920
3	-6.9572740	-104.5676307	5401.5514304
4	-18.3950447	-110.3089037	3313.5332961
5	-24.4982339	-101.1141646	-268.0738796
6	23.5034473	-18.6341695	-3317.7078937
7	0.0000000	4.4151714	0.0000000
8	0.0000000	-10.0000000	0.0000000
9	0.0000000	-10.0000000	0.0000000
10	0.0000000	-10.0000000	0.0000000
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 62

K	U	V	P
1	0.0000000	-82.0268176	7249.2925127
2	-9.4524147	-82.0268176	7249.2925127
3	-24.2235413	-78.5732798	6937.6349498
4	-48.7850070	-71.1473010	6551.7301161
5	-71.0051447	-61.7820285	6843.1646426
6	-30.8816820	-47.4380437	9759.1740687
7	-2.1127798	-21.3554438	11156.9516855
8	-2.3212576	-9.4508944	11039.9224358
9	0.0000000	-10.0000000	0.0000000
10	0.0000000	-10.0000000	0.0000000
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 63

K	U	V	P
1	0.0000000	-61.0513538	8974.6271123
2	-10.4888464	-61.0513538	8974.6271123
3	-19.9424645	-58.9772038	9091.1351193
4	-26.6532000	-55.1183693	9374.7530927
5	-30.7295561	-49.5087102	9748.5064019
6	-28.8293471	-42.7287024	10250.0698092
7	-11.8039612	-34.6807394	10845.1576466
8	3.0989040	-23.6842545	10922.1556718
9	-8.4641266	1.9203038	10813.3572648
10	0.0000000	-10.0000000	0.0000000

11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 64

K	U	V	P
1	0.0000000	-48.0237648	10140.2689266
2	-6.5144645	-48.0237648	10140.2689266
3	-12.3783922	-46.8163989	10192.4043643
4	-17.0440353	-44.5687357	10306.2742884
5	-19.6211230	-41.6940223	10471.1466032
6	-19.4451826	-38.5640446	10651.4581042
7	-15.5734409	-35.3689339	10785.2859917
8	-7.5060854	-29.9761403	10758.5877674
9	3.6140152	-8.9402761	10557.2669330
10	0.0000000	-10.0000000	0.0000000
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 65

K	U	V	P
1	0.0000000	-39.2339587	10699.1912733
2	-4.3952355	-39.2339587	10699.1912733
3	-8.4488472	-38.4817481	10720.8591449
4	-11.8650601	-37.0899104	10767.6330517
5	-14.5764716	-35.2052815	10827.6770013
6	-16.7118967	-32.9520442	10889.1692507
7	-18.5021603	-30.3771906	10942.5710077
8	-18.3719682	-27.2695485	11003.0272539
9	-3.7242250	-22.1141746	11203.4913647
10	0.0000000	-1.7042921	0.0000000
11	0.0000000	-10.0000000	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 66

K	U	V	P
1	0.0000000	-32.7809941	11003.3453936
2	-3.2265659	-32.7809941	11003.3453936
3	-6.2917649	-32.2437845	11012.9590641
4	-9.0870317	-31.2188104	11034.5007415
5	-11.5596260	-29.7830068	11064.5493987
6	-13.7298098	-27.9716820	11100.0852222
7	-15.6395679	-25.7971689	11139.6524618

8	-17.1441833	-23.2427102	11185.6238479
9	-16.7556953	-20.2422004	11253.2191473
10	-0.9474240	-16.4704140	11385.5554285
11	0.0000000	-0.5778699	0.0000000
12	0.0000000	-10.0000000	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 67

K	U	V	P
1	0.0000000	-27.7731452	11186.8556152
2	-2.5038268	-27.7731452	11186.8556152
3	-4.9176803	-27.3558840	11192.3685792
4	-7.1682177	-26.5508436	11205.3072574
5	-9.2020882	-25.4102226	11224.4010466
6	-10.9661058	-23.9664044	11248.8709884
7	-12.3875807	-22.2522296	11278.1510251
8	-13.3515997	-20.2983048	11311.6719323
9	-13.6755399	-18.1159643	11348.4818051
10	-12.9286770	-15.6517979	11387.9755634
11	-0.1632654	-12.6533406	11459.0904238
12	0.0000000	0.1379119	0.0000000
13	0.0000000	-10.0000000	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 68

K	U	V	P
1	0.0000000	-23.8043524	11304.1952747
2	-1.9841709	-23.8043524	11304.1952747
3	-3.9018744	-23.4757982	11307.9379807
4	-5.6968657	-22.8357306	11316.6216324
5	-7.3193691	-21.9278596	11329.5559334
6	-8.7238844	-20.7788803	11346.1807864
7	-9.8679575	-19.4175338	11365.8829155
8	-10.7055454	-17.8776562	11387.9882293
9	-11.1680695	-16.1946934	11411.7673161
10	-11.1207076	-14.3875325	11436.4244538
11	-10.2417969	-12.4073720	11461.1223834
12	0.3643513	-9.9969536	11502.2087033
13	0.0000000	0.5214294	0.0000000
14	0.0000000	-10.0000000	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 69

K	U	V	P
1	0.0000000	-20.6370538	11380.9534843
2	-1.5833387	-20.6370538	11380.9534843
3	-3.1149578	-20.3776099	11383.6028176
4	-4.5505652	-19.8665647	11389.6015594

5	-5.8591222	-19.1397460	11398.4860974
6	-6.9850114	-18.2191715	11409.8714228
7	-7.9190730	-17.1280686	11423.3159023
8	-8.6305102	-15.8926735	11438.3040490
9	-9.0994525	-14.5432414	11454.2718540
10	-9.2925952	-13.1119853	11470.6656459
11	-9.1132488	-11.6174702	11487.0134304
12	-8.2658842	-10.0162219	11502.9529231
13	0.7333984	-8.0561981	11528.4317667
14	0.0000000	0.8250144	0.0000000
15	0.0000000	-10.0000000	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 70

K	U	V	P
1	0.0000000	-18.0966489	11431.6749881
2	-1.2698404	-18.0966489	11431.6749881
3	-2.4995453	-17.8909115	11433.5947233
4	-3.6544986	-17.4803033	11437.8273981
5	-4.7052887	-16.8942090	11444.0639066
6	-5.6258691	-16.1501907	11452.0404051
7	-6.3941801	-15.2665272	11461.4561694
8	-6.9932403	-14.2633086	11471.9659584
9	-7.4132065	-13.1623397	11483.1833663
10	-7.6514237	-11.9871075	11494.7089488
11	-7.6979968	-10.7625307	11506.1969991
12	-7.4833322	-9.5074728	11517.4483071
13	-6.7199975	-8.2015147	11528.4687402
14	0.9920661	-6.6570982	11544.5655334
15	0.0000000	1.0449895	0.0000000
16	0.0000000	-10.0000000	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 71

K	U	V	P
1	0.0000000	-16.0498023	11465.4234173
2	-1.0230351	-16.0498023	11465.4234173
3	-2.0149123	-15.8858262	11466.8521593
4	-2.9483720	-15.5536693	11469.9174213
5	-3.8004800	-15.0774768	11474.4127291
6	-4.5511453	-14.4710701	11480.1603864
7	-5.1835570	-13.7486484	11486.9593206
8	-5.6847005	-12.9256522	11494.5833134
9	-6.0459046	-12.0186059	11502.7834224
10	-6.2641085	-11.0443003	11511.2948726
11	-6.3439776	-10.0186003	11519.8451589
12	-6.2930600	-8.9561586	11528.1943726
13	-6.0865238	-7.8693524	11536.2145501
14	-5.5252249	-6.7534882	11544.0133522
15	1.1783590	-5.4968363	11555.5570955
16	0.0000000	1.5398376	0.0000000
17	0.0000000	-10.0000000	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 72

K	U	V	P
1	0.0000000	-14.3947914	11487.9666391

2	-0.8271110	-14.3947914	11487.9666391
3	-1.6298886	-14.2634893	11489.0603843
4	-2.3866137	-13.9931291	11491.3423109
5	-3.0790958	-13.6036849	11494.6745830
6	-3.6913754	-13.1060306	11498.9378984
7	-4.2100213	-12.5112165	11503.9989846
8	-4.6245217	-11.8311004	11509.7081783
9	-4.9276774	-11.0781722	11515.9030567
10	-5.1157004	-10.2650326	11522.4157023
11	-5.1882357	-9.4033047	11529.0752175
12	-5.1502294	-8.5014187	11535.7105183
13	-5.0145886	-7.5626191	11542.1624417
14	-4.7868238	-6.5887843	11548.3396250
15	-4.3494494	-5.5954350	11554.3175052
16	1.8793025	-4.6030451	11561.7124858
17	0.0000000	2.6211165	0.0000000
18	0.0000000	-10.0000000	0.0000000

L= 73

K	U	V	P
1	0.0000000	-13.0536282	11503.0158805
2	-0.6701973	-13.0536282	11503.0158805
3	-1.3212323	-12.9481023	11503.8748393
4	-1.9355136	-12.7270039	11505.6182752
5	-2.4983293	-12.4069893	11508.1537078
6	-2.9966639	-11.9966818	11511.4011365
7	-3.4193960	-11.5047559	11515.2712474
8	-3.7575274	-10.9404567	11519.6655999
9	-4.0044019	-10.3134021	11524.4779299
10	-4.1558959	-9.6330971	11529.5994590
11	-4.2102541	-8.9082081	11534.9239233
12	-4.1667993	-8.1455617	11540.3528820
13	-4.0256317	-7.3472012	11545.7878011
14	-3.8015499	-6.4993972	11551.1046285
15	-3.5800725	-5.5431495	11556.1740036
16	-3.6101837	-4.3246374	11561.1115033
17	1.6944709	-2.6211153	11570.2894079
18	0.0000000	-10.0000000	11570.2894079

L= 74

K	U	V	P
1	0.0000000	-11.9658354	11512.9904384
2	-0.5435331	-11.9658354	11512.9904384
3	-1.0718353	-11.8808354	11513.6790584
4	-1.5704015	-11.6995279	11515.0408382
5	-2.0271026	-11.4359188	11517.0130069
6	-2.4310685	-11.0969378	11519.5408958
7	-2.7727987	-10.6895314	11522.5643414
8	-3.0442734	-10.2211089	11526.0170336
9	-3.2390058	-9.6993890	11529.8287506
10	-3.3519788	-9.1320316	11533.9287120
11	-3.3794990	-8.5260227	11538.2493037
12	-3.3190366	-7.8866784	11542.7340875
13	-3.1672877	-7.2165781	11547.3576384
14	-2.9066201	-6.5167602	11552.1720174
15	-2.4556042	-5.7952241	11557.3410439
16	-1.5516379	-5.0900898	11562.9057894
17	0.3896617	-4.5245430	11568.5305121
18	0.0000000	3.9367294	11568.5305121

L= 75

K	U	V	P
1	0.0000000	-11.0838190	11519.4903250
2	-0.4406739	-11.0838190	11519.4903250
3	-0.8691338	-11.0153080	11520.0513537
4	-1.2732157	-10.8665556	11521.1331780
5	-1.6427492	-10.6494196	11522.6927708
6	-1.9684775	-10.3695763	11524.6920409
7	-2.2421045	-10.0327500	11527.0886133
8	-2.4563009	-9.6451327	11529.8364806
9	-2.6046133	-9.2133270	11532.8866551
10	-2.6811487	-8.7442234	11536.1892181
11	-2.6797261	-8.2449576	11539.6972309
12	-2.5916835	-7.7233733	11543.3696220
13	-2.4011193	-7.1897446	11547.1694181
14	-2.0780237	-6.6603784	11551.0375733
15	-1.5753820	-6.1622488	11554.8084984
16	-0.8494713	-5.7320824	11558.0343204
17	0.0000000	-5.3463104	11559.8470467
18	0.0000000	5.3463104	11559.8470467

L= 76

K	U	V	P
1	0.0000000	-10.3695257	11523.5903795
2	-0.3568480	-10.3695257	11523.5903795
3	-0.7038382	-10.3143420	11524.0525949
4	-1.0306127	-10.1925172	11524.9225433
5	-1.3285378	-10.0141196	11526.1696773
6	-1.5896364	-9.7838964	11527.7664299
7	-1.8065930	-9.5067158	11529.6818810
8	-1.9726913	-9.1880002	11531.8805474
9	-2.0816266	-8.8337934	11534.3248966
10	-2.1270861	-8.4508902	11536.9731123
11	-2.1019699	-8.0471987	11539.7800254
12	-1.9973142	-7.6325386	11542.6913398
13	-1.8016391	-7.2199691	11545.6318977
14	-1.5022744	-6.8273138	11548.4852838
15	-1.0905037	-6.4777223	11551.0618808
16	-0.5732014	-6.1917502	11553.0752140
17	0.0000000	-5.9007564	11554.1812527
18	0.0000000	5.9007564	11554.1812527

L= 77

K	U	V	P
1	0.0000000	-9.7920549	11526.0237391
2	-0.2884752	-9.7920549	11526.0237391
3	-0.5689713	-9.7476728	11526.4073509
4	-0.8325591	-9.6483009	11527.1122689
5	-1.0718598	-9.5024545	11528.1161415
6	-1.2799846	-9.3141634	11529.3980139
7	-1.4505148	-9.0876770	11530.9328335
8	-1.5774403	-8.8278821	11532.6921950
9	-1.6550238	-8.5404482	11534.6427570
10	-1.6775858	-8.2320355	11536.7454040
11	-1.6393173	-7.9106435	11538.9529179
12	-1.5343729	-7.5861118	11541.2044833
13	-1.3576743	-7.2706901	11543.4202860
14	-1.1067663	-6.9794127	11545.4928512
15	-0.7846621	-6.7295453	11547.2822013

16	-0.4029492	-6.5322851	11548.6185561
17	0.0000000	-6.2904820	11549.3181656
18	0.0000000	6.2904820	11549.3181656

L= 78

K	U	V	P
1	0.0000000	-9.3259967	11527.2979746
2	-0.2328103	-9.3259967	11527.2979746
3	-0.4591686	-9.2903559	11527.6181633
4	-0.6713104	-9.2097947	11528.1921374
5	-0.8629243	-9.0914112	11529.0031331
6	-1.0280969	-8.9386350	11530.0336416
7	-1.1612998	-8.7552045	11531.2632668
8	-1.2573726	-8.5455315	11532.6671424
9	-1.3115035	-8.3148415	11534.2148143
10	-1.3192454	-8.0693329	11535.8695882
11	-1.2766722	-7.8163855	11537.5862020
12	-1.1808063	-7.5647563	11539.3088693
13	-1.0304400	-7.3247220	11540.9675549
14	-0.8273466	-7.1080048	11542.4786257
15	-0.5776878	-6.9269793	11543.7446656
16	-0.2927353	-6.7870375	11544.6620184
17	0.0000000	-6.5735900	11545.1291698
18	0.0000000	6.5735900	11545.1291698

L= 79

K	U	V	P
1	0.0000000	-8.9502657	11527.7701552
2	-0.1876878	-8.9502657	11527.7701552
3	-0.3701853	-8.9216605	11528.0388417
4	-0.5406898	-8.8569000	11528.5077627
5	-0.6938329	-8.7617052	11529.1640657
6	-0.8245993	-8.6389612	11529.9929428
7	-0.9283364	-8.4919246	11530.9767919
8	-1.0007893	-8.3245070	11532.0936340
9	-1.0381656	-8.1413627	11533.3163674
10	-1.0372651	-7.9479834	11534.6119302
11	-0.9957333	-7.7507573	11535.9404619
12	-0.9124578	-7.5570107	11537.2544649
13	-0.7881525	-7.3749366	11538.4982462
14	-0.6260919	-7.2133489	11539.6093140
15	-0.4327693	-7.0809396	11540.5206131
16	-0.2175178	-6.9797220	11541.1679755
17	0.0000000	-6.7839476	11541.4925414
18	0.0000000	6.7839476	11541.4925414

L= 80

K	U	V	P
1	0.0000000	-8.6472858	11527.6949043
2	-0.1513538	-8.6472858	11527.6949043
3	-0.2985380	-8.6243298	11527.9218676
4	-0.4355802	-8.5728719	11528.3065038
5	-0.5579593	-8.4972399	11528.8384753
6	-0.6614826	-8.3997998	11529.5055817
7	-0.7423143	-8.2833374	11530.2923630
8	-0.7970464	-8.1512309	11531.1794477
9	-0.8228016	-8.0074913	11532.1436460
10	-0.8173983	-7.8567839	11533.1561799
11	-0.7795660	-7.7044217	11534.1838944
12	-0.7092485	-7.5563202	11535.1882326

13	-0.6079867	-7.4188184	11536.1259813
14	-0.4793141	-7.2984231	11536.9510400
15	-0.3290386	-7.2011718	11537.6177738
16	-0.1644883	-7.1271515	11538.0849252
17	0.0000000	-6.9430265	11538.3164520
18	0.0000000	6.9430265	11538.3164520

L= 81

K	U	V	P
1	0.0000000	-8.4024485	11527.2570896
2	-0.1223221	-8.4024485	11527.2570896
3	-0.2412428	-8.3840933	11527.4502459
4	-0.3515637	-8.3438778	11527.7673612
5	-0.4495333	-8.2847235	11528.2000537
6	-0.5316933	-8.2085167	11528.7379856
7	-0.5949209	-8.1175980	11529.3678391
8	-0.6365325	-8.0148043	11530.0733159
9	-0.6543918	-7.9034870	11530.8341133
10	-0.6470573	-7.7874832	11531.6265757
11	-0.6139451	-7.6710907	11532.4237878
12	-0.5555445	-7.5589596	11533.1949229
13	-0.4736257	-7.4559221	11533.9071984
14	-0.3714445	-7.3667352	11534.5269936
15	-0.2538301	-7.2955640	11535.0219915
16	-0.1264442	-7.2413605	11535.3654633
17	0.0000000	-7.0653289	11535.5341258
18	0.0000000	7.0653289	11535.5341258

L= 82

K	U	V	P
1	0.0000000	-8.2037932	11526.5919390
2	-0.0992684	-8.2037932	11526.5919390
3	-0.1956048	-8.1893918	11526.7582732
4	-0.2846461	-8.1587216	11527.0221169
5	-0.3633399	-8.1134018	11527.3762989
6	-0.4288472	-8.0549347	11527.8120647
7	-0.4786420	-7.9852545	11528.3180522
8	-0.5106165	-7.9066869	11528.8805711
9	-0.5231978	-7.8219426	11529.4830437
10	-0.5154544	-7.7340906	11530.1059122
11	-0.4872172	-7.6465201	11530.7277563
12	-0.4392086	-7.5628193	11531.3240821
13	-0.3731409	-7.4866398	11531.8703959
14	-0.2917897	-7.4214604	11532.3415520
15	-0.1989831	-7.3701662	11532.7148261
16	-0.0990155	-7.3310249	11532.9716849
17	0.0000000	-7.1611235	11533.0964821
18	0.0000000	7.1611235	11533.0964821

L= 83

K	U	V	P
1	0.0000000	-8.0419068	11525.8017118
2	-0.0809190	-8.0419068	11525.8017118
3	-0.1590481	-8.0312477	11525.9474638
4	-0.2310674	-8.0086581	11526.1694911
5	-0.2945059	-7.9748640	11526.4618332
6	-0.3470212	-7.9311244	11526.8172259
7	-0.3865597	-7.8790160	11527.2262628
8	-0.4114633	-7.8203885	11527.6777680
9	-0.4205690	-7.7573629	11528.1577716

10	-0.4132876	-7.6923189	11528.6508138
11	-0.3896894	-7.6278569	11529.1391062
12	-0.3505576	-7.5667040	11529.6049538
13	-0.2974279	-7.5116003	11530.0286124
14	-0.2325687	-7.4651256	11530.3917351
15	-0.1588761	-7.4293680	11530.6778374
16	-0.0794144	-7.4022768	11530.8735083
17	0.0000000	-7.2379837	11530.9669199
18	0.0000000	7.2379837	11530.9669199

L= 84

K	U	V	P
1	0.0000000	-7.9099767	11524.9647322
2	-0.0659719	-7.9099767	11524.9647322
3	-0.1290802	-7.9031390	11525.0948379
4	-0.1872597	-7.8872299	11525.2838964
5	-0.2384429	-7.8628866	11525.5279961
6	-0.2806648	-7.8312463	11525.8206176
7	-0.3122477	-7.7935560	11526.1543105
8	-0.3318897	-7.7512241	11526.5196684
9	-0.3387258	-7.7058472	11526.9056084
10	-0.3323935	-7.6592045	11527.2992785
11	-0.3130864	-7.6132244	11527.6871743
12	-0.2815654	-7.5699425	11528.0549536
13	-0.2391641	-7.5313973	11528.3879946
14	-0.1877025	-7.4995503	11528.6723274
15	-0.1292827	-7.4760854	11528.8959379
16	-0.0657674	-7.4591048	11529.0481160
17	0.0000000	-7.3016682	11529.1196416
18	0.0000000	7.3016682	11529.1196416

L= 85

K	U	V	P
1	0.0000000	-7.8037442	11524.1417224
2	-0.0531511	-7.8037442	11524.1417224
3	-0.1034842	-7.8006499	11524.2596279
4	-0.1500667	-7.7899908	11524.4233544
5	-0.1910588	-7.7732153	11524.6296423
6	-0.2248139	-7.7513375	11524.8733694
7	-0.2499803	-7.7252824	11525.1485753
8	-0.2655422	-7.6960626	11525.4477161
9	-0.2708679	-7.6648199	11525.7611993
10	-0.2657396	-7.6328270	11526.0793390
11	-0.2504011	-7.6014586	11526.3912390
12	-0.2255530	-7.5721890	11526.6855369
13	-0.1923356	-7.5465092	11526.9515226
14	-0.1522202	-7.5259484	11527.1781134
15	-0.1067337	-7.5120468	11527.3566480
16	-0.0566475	-7.5036231	11527.4788375
17	0.0000000	-7.3565535	11527.5357413
18	0.0000000	7.3565535	11527.5357413

L= 86

K	U	V	P
1	0.0000000	-7.7208541	11523.3792486
2	-0.0415035	-7.7208541	11523.3792486
3	-0.0807198	-7.7207811	11523.4874683
4	-0.1171924	-7.7140196	11523.6310783
5	-0.1492891	-7.7031311	11523.8079364
6	-0.1757006	-7.6888968	11524.0140381

7	-0.1953839	-7.6719458	11524.2440748
8	-0.2075765	-7.6529563	11524.4918997
9	-0.2118151	-7.6327047	11524.7501555
10	-0.2079886	-7.6120441	11525.0106464
11	-0.1963395	-7.5919137	11525.2649906
12	-0.1774879	-7.5733262	11525.5044336
13	-0.1523930	-7.5573492	11525.7204073
14	-0.1222729	-7.5451700	11525.9049955
15	-0.0883407	-7.5382672	11526.0512131
16	-0.0506767	-7.5365828	11526.1529135
17	0.0000000	-7.4056846	11526.2020874
18	0.0000000	7.4056846	11526.2020874

L= 87

K	U	V	P
1	0.0000000	-7.6594412	11522.7104659
2	-0.0307850	-7.6594412	11522.7104659
3	-0.0602323	-7.6611105	11522.8107693
4	-0.0876031	-7.6571893	11522.9391056
5	-0.1116710	-7.6507004	11523.0936120
6	-0.1314935	-7.6421598	11523.2704702
7	-0.1463080	-7.6319800	11523.4656754
8	-0.1555621	-7.6205850	11523.6742916
9	-0.1589324	-7.6084725	11523.8901722
10	-0.1563705	-7.5961639	11524.1069841
11	-0.1480981	-7.5842676	11524.3183012
12	-0.1346395	-7.5734272	11524.5169523
13	-0.1167815	-7.5643837	11524.6963250
14	-0.0955441	-7.5580220	11524.8509246
15	-0.0720314	-7.5556809	11524.9746973
16	-0.0462759	-7.5583673	11525.0632661
17	0.0000000	-7.4505761	11525.1102979
18	0.0000000	7.4505761	11525.1102979

L= 88

K	U	V	P
1	0.0000000	-7.6168504	11522.1573534
2	-0.0213895	-7.6168504	11522.1573534
3	-0.0422042	-7.6192346	11522.2522552
4	-0.0614645	-7.6173216	11522.3695087
5	-0.0784405	-7.6138604	11522.5069719
6	-0.0924778	-7.6092522	11522.6615715
7	-0.1030396	-7.6037644	11522.8304202
8	-0.1097363	-7.5976408	11523.0095136
9	-0.1123638	-7.5911587	11523.1939154
10	-0.1109050	-7.5846100	11523.3783173
11	-0.1055558	-7.5783450	11523.5577832
12	-0.0967357	-7.5727367	11523.7269114
13	-0.0850740	-7.5682627	11523.8806727
14	-0.0714183	-7.5655404	11524.0145969
15	-0.0568259	-7.5656470	11524.1244930
16	-0.0417713	-7.5701442	11524.2065425
17	0.0000000	-7.4911234	11524.2554369
18	0.0000000	7.4911234	11524.2554369

L= 89

K	U	V	P
1	0.0000000	-7.5899442	11521.7361162
2	-0.0135587	-7.5899442	11521.7361162
3	-0.0269989	-7.5924263	11521.8277584

4	-0.0395656	-7.5915825	11521.9362574
5	-0.0507305	-7.5899309	11522.0607753
6	-0.0600172	-7.5877935	11522.1995423
7	-0.0670674	-7.5852931	11522.3493921
8	-0.0716307	-7.5825397	11522.5072513
9	-0.0735927	-7.5796461	11522.6689289
10	-0.0729597	-7.5767538	11522.8304202
11	-0.0698966	-7.5740220	11522.9876275
12	-0.0647194	-7.5716362	11523.1366391
13	-0.0579239	-7.5698591	11523.2736367
14	-0.0502050	-7.5690673	11523.3954536
15	-0.0425443	-7.5700057	11523.4993892
16	-0.0358715	-7.5742467	11523.5820907
17	0.0000000	-7.5259706	11523.6375975
18	0.0000000	7.5259706	11523.6375975

L= 90

K	U	V	P
1	0.0000000	-7.5757933	11521.4594203
2	-0.0071873	-7.5757933	11521.4594203
3	-0.0151370	-7.5771961	11521.5468715
4	-0.0226231	-7.5766913	11521.6492238
5	-0.0292381	-7.5760490	11521.7654529
6	-0.0347284	-7.5753357	11521.8934166
7	-0.0389110	-7.5745538	11522.0309729
8	-0.0416669	-7.5737234	11522.1746760
9	-0.0429466	-7.5728637	11522.3212662
10	-0.0427739	-7.5720219	11522.4677632
11	-0.0412701	-7.5712410	11522.6108144
12	-0.0386569	-7.5705819	11522.7471600
13	-0.0352965	-7.5701428	11522.8742855
14	-0.0317401	-7.5700820	11522.9903283
15	-0.0288687	-7.5707712	11523.0951021
16	-0.0279569	-7.5733380	11523.1879550
17	0.0000000	-7.5531651	11523.2627402
18	0.0000000	7.5531651	11523.2627402

L= 91

K	U	V	P
1	0.0000000	-7.5700000	11521.3313634
2	-0.0030115	-7.5700000	11521.3313634
3	-0.0070247	-7.5700000	11521.4175108
4	-0.0103841	-7.5700000	11521.5178142
5	-0.0132068	-7.5700000	11521.6298523
6	-0.0154948	-7.5700000	11521.7520418
7	-0.0172148	-7.5700000	11521.8824270
8	-0.0183423	-7.5700000	11522.0177482
9	-0.0188646	-7.5700000	11522.1558633
10	-0.0188098	-7.5700000	11522.2935128
11	-0.0182404	-7.5700000	11522.4280889
12	-0.0172712	-7.5700000	11522.5574496
13	-0.0161034	-7.5700000	11522.6796391
14	-0.0150784	-7.5700000	11522.7943781
15	-0.0148804	-7.5700000	11522.9039016
16	-0.0172513	-7.5700000	11523.0147290
17	0.0000000	-7.5700000	11523.1482806
18	0.0000000	7.5700000	0.0000000

L= 92

K	U	V	P

1	0.0000000	-7.5700000	0.0000000
2	0.0000000	-7.5700000	11521.3313634
3	0.0000000	-7.5700000	11521.4175108
4	0.0000000	-7.5700000	11521.5178142
5	0.0000000	-7.5700000	11521.6298523
6	0.0000000	-7.5700000	11521.7520418
7	0.0000000	-7.5700000	11521.8824270
8	0.0000000	-7.5700000	11522.0177482
9	0.0000000	-7.5700000	11522.1558633
10	0.0000000	-7.5700000	11522.2935128
11	0.0000000	-7.5700000	11522.4280889
12	0.0000000	-7.5700000	11522.5574496
13	0.0000000	-7.5700000	11522.6796391
14	0.0000000	-7.5700000	11522.7943781
15	0.0000000	-7.5700000	11522.9039016
16	0.0000000	-7.5700000	11523.0147290
17	0.0000000	-7.5700000	11523.1482806
18	0.0000000	-10.0000000	0.0000000

APPENDIX IX

SAMPLE DIALOGUE FOR COMPUTATIONAL INPUT PROGRAM

```
NEW GENERATOR?Y

INPUT FROM:DSK

INPUT FILE:BALE

HEADING

KMAX,LMAX,NEXP,MOPT

SMALL,EPSI

GZ,GR,MU,RHOA

DR
DZ

NP,KA,KB,LA,LB

BPX,BPY
KA,KB,LA,LB,UVEL,VVEL,KIN,KOUT

KA,KB,LA,LB,UVEL,VVEL,KIN,KOUT

KA,KB,LA,LB,UVEL,VVEL,KIN,KOUT

KA,KB,LA,LB,UVEL,VVEL,KIN,KOUT

NBP,XXA,YYA,DDR,DDZ

XB,YB,NSEQ

XB,YB,NSEQ

ICRT,CRIRA,CRIRB,CRIZA,CRIZB,XMAX,YMAX

KOMI,KOMX,LOMN,LOMX

DT=

NO. OF CYCLES BETWEEN EDITS=

BOUNDARY SENSING PARAMETER=

CONVERGENCE EPSILON=

OVER-RELAXATION FACTOR=

BACKWARDS DIFFERENCING?

FULL CELL EPSILON=

STOP AT CYCLE-

PRESSURE ITERATIONS=

NO. OF CYCLES BETWEEN DUMPS=
```

APPENDIX X

SAMPLE DIALOGUE FOR ABMAC

```
PROCESS OR EXAMINE?P

ABMAC
SAVED CORE?N

INPUT FILE=BALE

UPDATE?N

FIB RUN?N

CREATE SHARE FILE?N
```