NASA Technical Memorandum 103721

A Finite Element Model of Conduction, Convection, and Phase Change Near a Solid/Melt Interface

Larry A. Viterna
Lewis Research Center
Cleveland, Ohio

January 1991

A FINITE ELEMENT MODEL OF CONDUCTION, CONVECTION, AND PHASE CHANGE NEAR A SOLID/MELT INTERFACE

Larry A. Viterna
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

Detailed understanding of heat transfer and fluid flow is required for many aerospace thermal systems. These systems often include phase change and operate over a range of accelerations or effective gravitational fields.

An approach to analyzing such systems is presented which requires the simultaneous solution of the conservation laws of energy, momentum, and mass, as well as an equation of state. The variable property form of the governing equations are developed in two-dimensional Cartesian coordinates for a Newtonian fluid.

A numerical procedure for solving the governing equations is presented and implemented in a computer program. The Galerkin form of the finite element method is used to solve the spatial variation of the field variables, along with an implicit Crank-Nicolson time marching algorithm. Quadratic Lagrangian elements are used for the internal energy and the two components of velocity. Linear Lagrangian elements are used for the

pressure.

The location of the solid/liquid interface as well as the temperatures are determined from the calculated internal energy and pressure. This approach is quite general in that it can describe heat transfer without phase change, phase change with a sharp interface, and phase change without an interface.

Analytical results from this model are compared to those of other researchers studying transient conduction, convection, and phase change and are found to be in good agreement. The numerical procedure presented requires significant computer resources, but this is not unusual when compared to similar studies by other researchers. Several methods are suggested to reduce the computational times.

ACKNOWLEDGMENTS

I would especially like to thank the co-chairmen Professor Gene E. Smith and Dr. Robert Siegel for their help in completing and critiquing this work. I also wish to acknowledge the assistance of all the members of my doctoral committee.

TABLE OF CONTENTS

NOMENCLATURE

a	thermal diffusivity
C	specific heat (also capacitance matrix)
D	geometric dimension
e	internal energy
Fo	Fourier number
g	acceleration (also gravity)
i	node number
j	time step increment
Ja	Jakob number
k	thermal conductivity
K	stiffness matrix
L	latent heat
$\hat{n}$	unit normal vector
$\hat{m}$	unit tangential vector
M	mass matrix
N	interpolation (shape) functions
Nu	Nusselt number
n	number of global nodes
P	pressure
Pr	Prantl number
q	heat flux
r	number of nodes per element
Ra	Rayleigh number
T	temperature
t	time
u	velocity in x-direction
v	velocity in y-direction
V	velocity
W	Gauss-Legendre weighting factors
x	coordinate in cartesian system
y	coordinate in cartesian system
β	coefficient of thermal expansion
η	transformed coordinate
μ	absolute viscosity
ρ	density
σ	normal stress
τ	shear stress (also dimensionless time)
ν	kinematic viscosity
ϕ	field variable
Φ	field value at a node
ζ	dimensionless location of phase interface
ξ	transformed coordinate

θ parameter in time-marching recursion algorithm (also dimensionless temperature)
Ψ prescribed nodal value
∇ divergence of a vector

Subscripts and superscripts

Ω classifies an area or volume integral
Σ classifies a surface integral
$*$ classifies an approximate value of field variable
l classifies liquid state
s classifies solid state
$_0$ classifies initial condition or reference state

Matrix notation

$\lfloor \rfloor$ single row matrix
$\{ \}$ single column matrix
$[ ]$ matrix

CHAPTER 1

INTRODUCTION

The need for general analytical tools for modeling heat transfer and fluid flow is increasing as man designs more complex thermal/fluid devices. This is particularly true in the aerospace industry where highly reliable systems must operate in environments where little or no supporting experimental data is available. Such systems often include phase change and operate over a range of accelerations or effective gravitational fields. Experimental investigations of fluid/thermal systems under low gravity conditions are difficult and expensive. Because of the time required for many phase change problems, most experimental studies are not possible in ground-based low-gravity facilities and must be done on Earth-orbiting laboratories. For these reasons a predictive analytical or numerical method would be very valuable.

The intent of this research is to develop a general purpose numerical approach and computer program for analyzing the heat transfer and fluid flow of materials undergoing phase change. Such an analytical tool would

significantly reduce the number of experiments required and aid in our understanding of the experimental results.

Many practical applications of such a computational tool exist, such as modeling cryogenic fluid management systems and analyzing advanced material processing and casting methods. Systems such as batteries and thermal management devices could be examined as they might have to withstand initial or inadvertent freezing in the low temperatures of space. Another application is in the analysis of designs for a thermal storage device to be used in the space power system on the National Aeronautics and Space Administration (NASA) Space Station Freedom or lunar base. As described by Klann[34], this space power system would collect and concentrate solar energy to heat the working fluid of a Brayton cycle heat engine. A latent heat thermal storage device would provide energy for the power system during dark periods of the Space Station orbit, (see Burns[10]).

Analytical approaches and numerical techniques for modeling thermal and fluid problems have been the subject of research and development for many years. Today, computer programs for modeling heat transfer by conduction are well developed and generally easy to apply. Until recently, computer programs for modeling fluid flow and its effect on convective heat transfer, however, had mostly been limited to empirical relationships based on known and simple geometries. Today, computational

convection using numerical methods such as finite differences and finite elements have become available to handle more complex flow geometries. Some of these are commercially available, however most are research oriented and limited in scope to particular applications. The additional complexities of having phase change phenomenon and materials that exhibit widely varying properties restricts the application of most present methods for such problems. Hence further work is needed to develop general purpose methods to analyze fluid/thermal problems with phase change.

In Chapter 2 a review is presented of the literature dealing with numerical solutions to thermal, fluid flow, and phase change problems. Chapters 3 and 4 cover the development of the governing equations and the numerical approach. Results and verification of the approach and model are presented in Chapter 5. The main body of the thesis ends with Chapter 6, in which concluding remarks and recommendations for further work are given. A computer program, PHASTRAN, developed during this research is discussed and presented in the appendices.

CHAPTER 2

LITERATURE REVIEW

Much research has been devoted to the analysis of materials undergoing phase change because of its association with many applications. The food, metallurgical, and semi-conductor industries are important examples. More recently, there has been an interest in modeling these processes in the space environment.

Many examples of research are in the literature for modeling phase change and fluid flow. For a detailed discussion on related subjects the reader is referred to: Stefan[58] and Lunardini[38], on the phase change problem; Carnahan, Luther, and Wilkes[11] on numerical methods including the finite difference method; Baker[4], Huebner and Thornton[32], Zienkiewicz[65] on the finite element method; Arpaci and Larsen[2] on convective heat transfer; and VanWylen and Sonntag[61] on thermodynamics.

To summarize the some of the most important works related to this research topic, first those papers related to numerical modeling of the phase change problem will be discussed followed by those related to the fluid flow. The classical analytical approach to the phase change

problem will not be discussed. Though elegant in its mathematical derivation, its application to problems of complex geometry and temperature dependent material properties is impractical. Some noteworthy papers on the classical analytical modeling of phase change include Budhia and Kreith[9], Siegel and Savino[54].

Most of the numerical models of phase change have involved the finite difference method, and to a lesser extent, the finite element method. This is not because of any superiority of the finite difference method, but rather the chronological development of the two methods. Indeed, most of the recent works have been devoted to the finite element approach.

Otis[46] solved the melting problem by dividing the region into finite space intervals. Temperature was assumed uniform within each volume element at any instant and the latent heat effect was modeled as a moving heat source. The method required a coordinate transformation in terms of a pseudo time variable and was limited to analyses of materials initially or finally at the fusion temperature.

Murray and Landis[41] suggested an approach by which the interface location was calculated by solving a differential equation for the velocity of the phase front. The differential equation was derived from an energy balance at the phase front. The temperature at the new front location was then set equal to the freezing

temperature.

Springer and Olson[56] used the Murray and Landis approach to track the phase front in two dimensions. Again the temperature at the phase front was set equal to the fusion temperature and the temperatures in the remainder of the solid and fluid was determined from a finite difference solution for heat conduction.

Shamsundar and Sparrow[53] used enthalpy as the dependent variable instead of temperature in a finite difference formulation. Because their formulation involved an integral approach to the energy balance, the method eliminated the need to explicitly track the interface. They maintained this was the best method for analysis of multidimensional conduction phase change. More discussion on this method follows in Chapter 3.

Only a few researchers have included the effects of natural convection in the fluid or radiative heat transfer. Such effects introduce nonlinearities in the field equations which require iterative solution procedures and increased computational times.

Tien[59] solved the phase change problem with natural convection included in the fluid. He used a finite difference formulation of the conservation laws using a vorticity and stream function form of the momentum equations. Again the Murray and Landis approach was used to track the phase front. Tien's numerical results compared favorably with experimental data on the freezing

of naphthalene.

Valle[60] also included natural convection in his solution but solved the problem using the finite element method. The conservation laws were developed in terms of the stream function and temperature. The latent heat effects and phase front motion were formulated implicitly in terms of an imbalance of the heat fluxes at the solid/liquid interface. This was one of the most detailed analyses of the phase change problem to date and included fluid flow, surface tension, and radiation effects. Valle compared his results to the work of Tien, however, and concluded that this approach did not seem to track the interface motion as effectively as approaches based on that of Murray and Landis.

More recently, several works have approached the phase change problem using moving and deforming finite element grids and/or coordinate transformation.

Ettouney and Brown[18] transformed the problem so that the melt and solid regions have fixed boundaries, of which the interface is one. This is an elegent approach which couples the interface shape and field variables allowing more efficient solution techniques. However, this approach, as with other moving mesh formulations, has the limitation of not being able to easily handle disappearance, merging or fragmentary distribution of phases.

Albert and O'Neill[1] used a method of transfinite

mappings in conjunction with a moving boundary-moving mesh finite element technique. Improvement in tracking the phase front was made compared to a fixed mesh approach. Again there is the limitation mentioned above for the Ettouney and Brown method which restricts its application.

Because of the high computational costs associated with modeling the phase change problem, some researchers have studied less numerically intensive schemes. Schneider[52] formulated the phase change problem using the finite difference technique along with a variation of the enthalpy method of Shamsundar and Sparrow[53]. Depending on the amount of movement of the interface, Schneider's algorithm adjusts the number of convergence iterations. If the interface only moves within one grid spacing, only one iteration is used to converge the nonlinearities. This significantly reduces the computational times but may also affect the accuracy, especially for materials with properties that vary rapidly near the interface.

The application of numerical methods to the modeling of fluid flow problems has made remarkable progress over the last 25 years. Initially, computer-based solutions used the finite difference method. Over the years, the finite difference method has provided solutions to many difficult flow problems including slow viscous flows, boundary layer flows and even variable property flows (thermo-hydrodynamic) flows. More recently, the finite element method has been developed to handle many of the

same problems. The finite element method has been shown to be particularly useful in problems involving complex geometries and boundary conditions. Baker[3], and Gallagher, et al.[21] contain many examples of the application of finite element to complex problems.

Early applications of the finite element method to some continuum problems often used variational methods to derive the finite element equations. The lack of exact variational forms of the Navier-Stokes equations, however, prevented the use of finite elements to practical flow problems. Later, the application of weighted residual methods broadened the application of finite elements to a variety of fluid problems.

Olson[44] applied a pseudo-variational approach to a two-dimensional incompressible formulation developed in terms of the stream function.

Baker[3] applied the weighted residual technique of Galerkin[22] to viscous incompressible flow. The Galerkin criteria, originally a nondiscretized approach is currently the most widely used method of formulating the finite element (discretized) equations.

Hood and Taylor[31] also used the Galerkin criteria and formulated the Navier-Stokes equations in three ways: the velocity/pressure formulation; the stream function and vorticity formulation; and the purely stream function formulation. Comparison of these three formulations suggests that the velocity/pressure formulation may have

several advantages. It is readily extended to three dimensions. Pressure, velocity, velocity gradient, and stress boundary conditions can be easily handled. And it appears to require less computational time than the other formulations.

Recently, more attention has been given to the considerations for obtaining good quality solutions to these nonlinear fluids problems over a wider range of conditions. Important aspects of this include proper choice of solution technique, element types, and mesh refinement.

Gartling, et al.[23] formulated the finite element equations in terms of velocity and pressure and studied the convergence properties of several solution algorithms, two element types, and several mesh refinements. Laminar flow between converging plane walls was used to represent a nonlinear problem. Of the solution techniques, they found that those which solved the full unsymmetric equation system were superior and more generally applicable than their symmetric counterparts. In particular, the Newton-Raphson procedure was the most rapidly convergent. No significant difference was found between an 8-node quadrilateral element and a 13-node quadrilateral element, with the 8-node being preferred because of its reduced complexity in formulation and use. Finally, adequate mesh refinement was required in the direction of most rapid variation of the solution field.

Ben-Sabar and Caswell[6] investigated the effect of the choice of boundary conditions on the problems where the ratio of convective to diffusive terms are large. They found that consistent use of the velocity and surface traction boundary conditions are necessary to delay the appearance of numerical instabilities with increasing Reynolds number.

Fletcher[20] developed an alternating direction implicit finite element method for flows where the convection terms dominate and applied the method to viscous compressible flow past a rectangular object. In comparison with an equivalent finite difference scheme, he found the finite element approach to be computationally more efficient.

Solutions to coupled fluid/thermal problems continue to be the subject of much research, particularly transient problems in three-dimensional space. Such problems often require the expenditure of significant computer resources. Though beyond the scope of this study, a number of efforts are directed at improved solution methods to solving large systems of nonlinear equations. The use of many of these methods will be dependent on the availability of new computer architectures providing vector and parallel processing capabilities.

CHAPTER 3

PROBLEM FORMULATION

Problem Statement

The problem selected is to analytically determine the transient temperatures, heat transfer rates, fluid velocities, and pressures in a pure substance or eutectic material undergoing phase change. The material can exist in solid and fluid states with variable properties satisfying a general equation of state model. It is contained in a vessel of arbitrary geometry such as is shown in Figure 3.1. Boundary conditions could include prescribed temperatures, heat flux, and fluid velocities. Flow in the fluid is induced due to a gravitational body force or accelerating reference frame and both inertial and viscous effects are included.

No surface free energy (surface tension) effects are included, nor is heat transfer by radiation. The effect of supercooling and mechanisms of nucleation or crystallization are also not considered. In addition, the fluid motion is restricted to laminar flow.

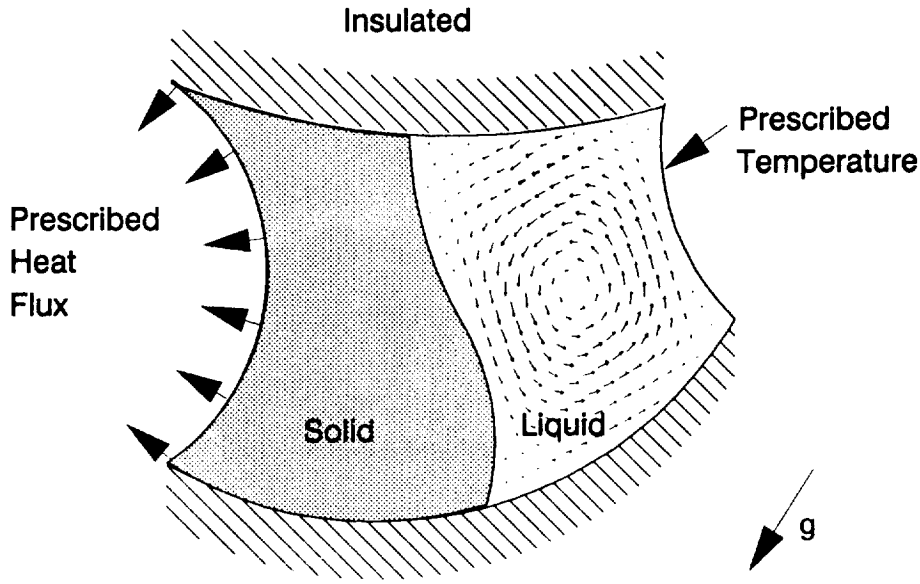

Figure 3.1 Example Phase Change System

Governing Equations

Problems in science and engineering can be classified as *Lagrangian* or *Eulerian* depending on the viewpoint or reference frame adopted. To formulate the governing equations, one of these two approaches must be adopted.

In the Lagrangian approach, all matter consist of particles which can be identified as they move through space. The independent variables in the Lagrangian system

are x_0, y_0, z_0 and t where x_0, y_0, z_0 are the coordinates which a specified fluid element passed through at time t_0.

In the Eulerian approach, processes are characterized by continua of field quantities. The independent variables are the spatial coordinates x, y, z and time. To derive the governing equations, we focus our attention on one area in space called a control volume. If we apply the governing laws of the problem to a differential control volume we obtain a set of governing differential equations. This is the approach with which most problems in fluid and thermal analysis are formulated and is the approach adopted here.

The solution to modeling the phase change problem includes solving the equations expressing the three physical laws of:

1. Conservation of energy
2. Conservation of momentum
3. Conservation of mass

as well as a thermodynamic equation of state.

Because this problem is dominated by thermal aspects, it is particularly important to consider the form of the energy equation. The conservation of energy is most commonly expressed in terms of temperature and specific heat. Such formulations are quite valid for single phase problems, however, they may be inappropriate for materials undergoing phase change at a discrete temperatures.

To further discuss this, let us consider two

situations, one in which a *sharp* interface is formed and the other in which a *mushy* region with no sharp interface will exist.

An example of a sharp interface might be a thin layer of water with its top and bottom sides insulated as shown in Figure 3.2. Suppose the water was initially at a

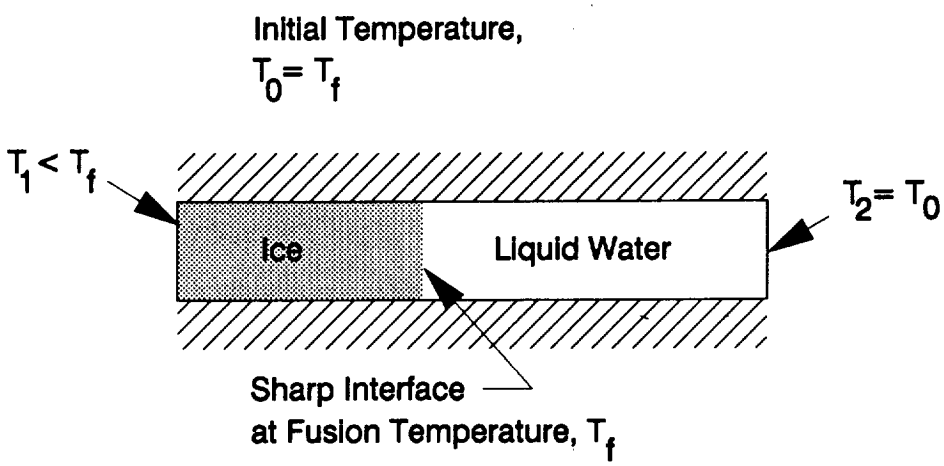

Figure 3.2 System Exhibiting a Sharp Interface During Phase Change

temperature above the freezing point and then one end of it is reduced to a temperature below the freezing point. Under these circumstances, a sharp interface will form which will separate the solid and liquid regions. In a volume element containing the interface however, the

specific heat is not easily defined. For such multi-phase problems, the energy equation is usually written separately for the solid and liquid phase regions. Since the interface is generally of unknown shape and position, numerical methods such as finite difference or finite element discretization encounter significant problems in handling the interface. Some numerical methods track the location of the interface. A heat balance can be formulated at the interface for more than one spatial coordinate. A differential equation is used to relate velocity of the interface to the heat absorption or removal. For three-dimensional phase change systems the interface is a surface and numerical methods for tracking the interface can become quite complicated.

The temperature and specific heat formulation also has difficulty when analyzing phase change where the interface is not sharp. White[62] justifies the existence of mushy regions with an example of the welding of two plates. A similar example would be to consider a thermally insulated plate with electrical connections at each end, as in Figure 3.3. When an electric current is passed through the plate it will heat up due to heat generation from internal resistance. Eventually, the temperature of the plate will reach the fusion (i.e. melting) temperature of the material. At this time the internal energy will equal the saturated value at that pressure and temperature and is equal to the product of

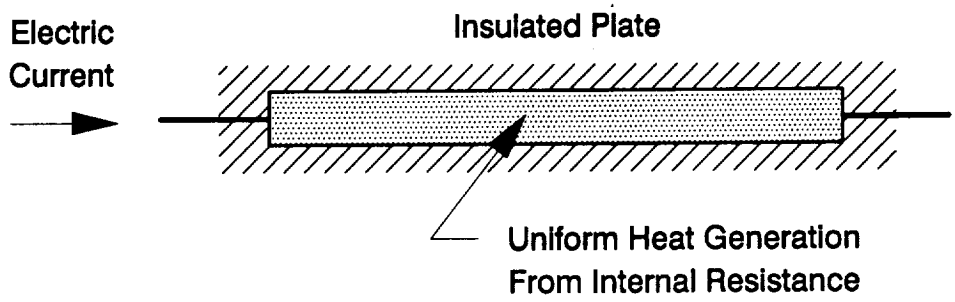

Figure 3.3 System Which Does Not Exhibit a Sharp Interface During Phase Change

the density, ρ, specific heat of the solid, C_s, and fusion temperature, T_f. This is shown graphically in Figure 3.4 as point 1. As time continues, the internal heat generation results in increased internal energy with no change in temperature. This continues until the amount of energy converted to heat equals the latent heat and the material becomes liquid. This is seen as point 2 in Figure 3.4. As time progresses further, temperature of the material resumes its increase governed by the specific heat in the liquid. From this example, it is apparent that with the temperature description of the problem, the continuous transition is lost during the phase change.

Also assuming negligible diffusion of the thermal energy out of the electrodes, the material can exist at the fusion temperature in a two-phase state with no apparent sharp interface.

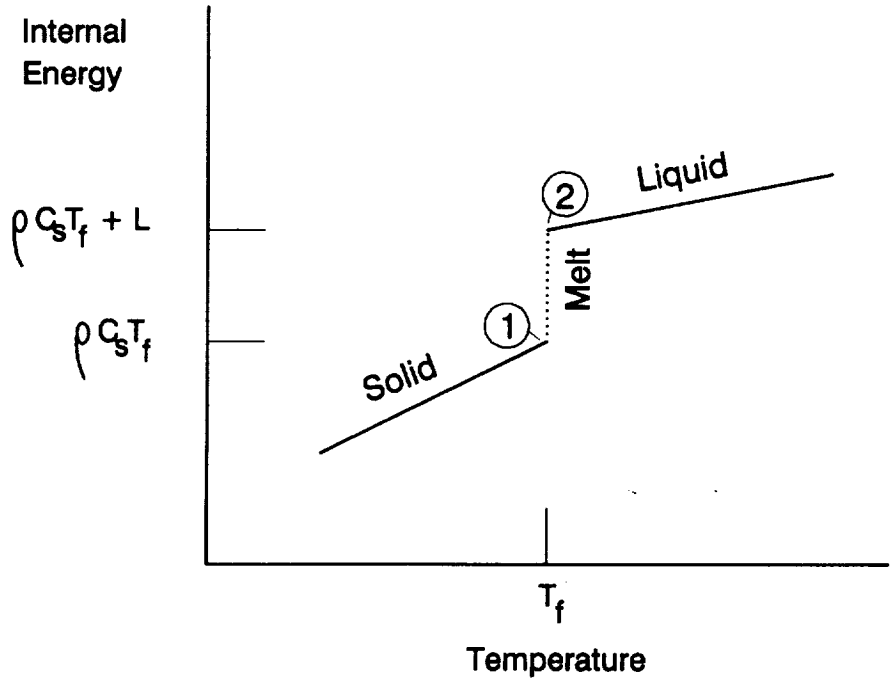

Figure 3.4 Internal Energy Versus Temperature for a Substance which Changes Phase at a Discrete Temperature

Many materials exhibit the behavior of phase change at a discrete temperature. To avoid the noncontinuous behavior of the product of temperature and specific heat, many formulations assume that the phase change occurs over a small but finite temperature range, see for example

Bamberger, et al.[5] This approach essentially defines an artificial specific heat for the volume containing the phase front. While these formulations retain temperature as the primary unknown, they may introduce significant errors in the results. Bonacina, et al.[8] demonstrated for example, that even in the one-dimensional case, the magnitude of the assumed range of phase change temperatures can affect the results significantly.

After studying the above examples, it is apparent that it is the energy in a given volume that is really of interest and not its temperature. Thus, an alternative to the temperature and specific heat formulation is to use internal energy as the primary unknown and compute the temperature from the internal energy. Shamsundar and Sparrow[53] were among the first to employ such an approach. They used an integral relation setting the rate of increase of the energy content in a arbitrary control volume equal to the net rate at which heat is conducted in through its surface. This relation was applicable whether or not the interfacial surface passes through the control volume. By assuming no fluid motion, pressure is independent of time, and they reformulated the problem in terms of enthalpy. Such a formulation turns out to be quite general in that it can describe heat conduction without phase change, phase change with a sharp interface, and phase change without an interface. Because the phase front is not explicitly tracked, the enthalpy formulation

avoids many of the numerical difficulties associated with fixed grid numerical methods, particularly in problems involving fragmented phases.

Using an Eulerian frame of reference the governing equations for this problem can now be presented. For a variable property, Newtonian fluid, neglecting internal heat generation, surface free energy, and radiation, the conservation laws in Cartesian coordinates are:

Conservation of energy:

$$\rho\frac{\partial e}{\partial t} + \rho u\frac{\partial e}{\partial x} + \rho v\frac{\partial e}{\partial y} + P\frac{\partial u}{\partial x} + P\frac{\partial v}{\partial y} = \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + \mu\Gamma(x,y,t) \tag{3.1}$$

in which Γ is the viscous dissipation function given by

$$\Gamma = 2\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right] + \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)^2 - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)$$

For the problems presented in Chapter 5, the natural convective velocities are slow and the vicosities are low. Under these conditions, the viscous dissipation terms may be neglected.

Conservation of momentum (Navier-Stokes equations):

$$\rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \rho v\frac{\partial u}{\partial y} = \rho g_x - \frac{\partial P}{\partial x} + \frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} \tag{3.2}$$

$$\rho\frac{\partial v}{\partial t} + \rho u\frac{\partial v}{\partial x} + \rho v\frac{\partial v}{\partial y} = \rho g_y - \frac{\partial P}{\partial y} + \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{xy}}{\partial y} \tag{3.3}$$

in which

$$\sigma_x = 2\mu\frac{\partial u}{\partial x} - \frac{2}{3}\mu\nabla\cdot V \tag{3.4}$$

$$\sigma_y = 2\mu\frac{\partial v}{\partial y} - \frac{2}{3}\mu\nabla\cdot V \tag{3.5}$$

$$\tau_{xy} = \tau_{yx} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{3.6}$$

Conservation of mass:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0 \tag{3.7}$$

Additional equations are required to evaluate the thermodynamic of state and material properties:

temperature, $T=T(e,P)$

density, $\rho=\rho(e,P)$

thermal conductivity, $k=k(e,P)$

viscosity, $\mu=\mu(e,P)$

If we can evaluate the state and the material properties explicitly we can reduce the number of equations to four and solve in terms of the basic unknowns u, v, P, and e. Specifying the state of a pure substance requires a minimum of two independent properties. When two phases of a pure substance exist together in equilibrium, the pressure and temperature are not independent and can therefore not be used to define the state. The two independent properties chosen as the basic field variables in the above equations are internal energy and pressure.

The initial conditions consist of specifying the velocities, pressure, and energy at time zero. The hydrodynamic boundary conditions specify either the velocity components or surface tractions. The thermal part of the problem requires the heat flux or internal energy be specified on the boundary. Temperature boundary conditions must therefore be converted to internal energy.

It should be noted that momentum and continuity equations as well as the convective transport terms in the energy equation are not required in that part of the solution domain which is in the solid state. The approach

to handling this problem is presented in the next chapter concerning the numerical method.

At this point, many formulations, for example Valle[60], perform a transformation with the fluid velocity variables into a streamfunction and vorticity formulation. This was not chosen here. This decision was due to the requirement that this formulation be easily extendable to three-dimensional space. The streamfunction-vorticity formulation is often applied to two-dimensional incompressible flows. It can, however, be applied to a broader class of problems. This is because the definitions for the dependent variable transformations are essentially vector identities. These transformations can therefore be applied to three-dimensional Cartesian coordinate system. Unfortunately, for three dimensions, six scalar components for the streamfunction and vorticity must be defined compared to the four (3 velocities and 1 pressure) used in the physical variable formulation. In addition, certain boundary conditions become difficult to apply. These difficulties have generally precluded application of streamfunction-vorticity to three-dimensional problems.

As given in the problem statement, the present analysis is restricted to laminar flow conditions. Fluid motion is characterized as *laminar* if the fluid flows in imaginary layers and there is no macroscopic mixing between adjacent fluid layers. A flow is said to be

turbulent, however, if such mixing occurs. It should be noted that the governing equations given above hold at any instant of time and apply to both laminar and turbulent flows. In a turbulent flow, however, the fluid velocities are fluctuating randomly about their mean values. Such a random variation in the field variables is nearly impossible to solve directly. The standard approach is to time-average the equations to obtain new ones which describe the temporally averaged field variables. Such an approach is beyond the scope of the present formulation, and the flow is assumed laminar.

CHAPTER 4

NUMERICAL APPROACH

The governing equations for the mass, momentum, and energy conservation given in the preceding chapter are represented by a system of nonlinear partial differential equations. These equations can describe some of the most interesting phenomenon in the fluid and thermal sciences. Unfortunately they are also some of the most difficult to solve.

With few exceptions, (see for example Graebel[27]), problems involving convection can not be solved by direct integration of the partial differential equations. For most problems we must resort to some numerical solution method. In the approach used here, the finite element method is used to solve the spatial problem along with a recursive time marching algorithm based on the finite difference method.

The finite element method is relatively new with most of its development occurring after 1960. There are several approaches to developing the finite element equations including the variational method, the method of weighted residuals, and the energy balance method. The

classical variational method is quite general, however it does require the existence of an exact variational form for the governing equations. For many problems, particularly in convective heat transfer, there are no exact variational forms. This requirement has limited the application of the variational method. Another procedure, the method of weighted residuals, or Galerkin method, does not require an alternate formulation of the physical problem and in fact can be applied to almost any well-posed system of differential equations. Oden[42] introduced a method by which the finite element equations can be developed from global energy considerations. This method has also proven very useful in the solution of many thermomechanical problems.

Of the methods mentioned above, the Galerkin method has proven to be the most general and is the method chosen for the formulation developed here.

The basic approach of the finite element method is to divide the solution domain up into a finite number of subdomains called elements. These elements are connected at node points on the element boundaries. The behaviors of the unknown field variables are then approximated within each element by continuous functions expressed in terms of nodal values of the field variables and their derivatives. Substitution of this approximation into the original differential equations and then integrating, results in some error or residual. In the Galerkin

method, linearly independent weighting functions are chosen such that the residual is required to vanish in some averaged sense over the entire solution domain. The resulting equations for each element are assembled into a set of coupled equations.

The coupled equations are then directly integrated in time to yield the nodal values of the field variables. This direct integration of the coupled equations uses a recursion technique based on the finite difference method. Approximations in the finite difference method, however, introduce numerical errors. Though these errors can be minimized as the time step used approaches zero, it is at the expense of increasing computational time. Large time steps, in contrast, can produce entirely unrealistic behavior, including nonphysical oscillations which can even become unstable. Development of the proper recursion technique is thus of great practical importance and is discussed more, later in this chapter.

Subdividing the Domain

The first step in applying the finite element method is to subdivide the solution domain into elements. The selection of proper element type is still somewhat of an art. Lower order polynomial elements are simplest to formulate, but more elements are required for good solution accuracy. Fewer higher order elements are needed

for the same accuracy but require increased computation time in the numerical integration of each element. In general, to model a complicated boundary, it is usually more efficient to use a large number of simple elements rather than a few complex ones. Thus, for most problems, elements with interpolating functions of order greater than 3 are seldom used.

In addition to computational efficiency, it is important that we select element types with interpolation functions that satisfy certain continuity and convergence requirements. This is necessary to ensure accuracy during integration and also that the approximate solution will converge to the correct solution with increasingly finer subdivisions (smaller elements). These requirements were given by Felippa and Clough[19] and verified by Oliveira[43]. Specifically, they can be stated as

1. The field variable ϕ and its derivatives up to one order less than the highest-order derivative of the element (weak form) equations must be continuous <u>at the element interfaces</u>.
2. The field variable ϕ and its derivatives up to the order of the highest-order derivative of the element (weak form) equations must be continuous <u>within the element</u>.

The first of these is known as the *compatibility*

requirement and the second as the *completeness* requirement. Compatibility requires that the field variable and its principal derivatives be the same at coincident nodes of neighboring elements. This ensures that there will be no contribution to the finite element equations from "gaps" at the element interfaces since the boundary integrals of each element will cancel. The completeness requirement ensures convergence to the correct solution when, in the limit, the element size shrinks to zero.

It is convenient to introduce a standard notation to describe the degree of continuity of a field variable at the element interfaces. If the field variable is continuous at the element interfaces, it is said to have C^0 continuity. If, in addition, the second derivatives are also continuous, there is C^1 continuity, and so on. By choosing the internal energy, pressure and velocity form of the governing equations only first derivatives of the field variables appear. Thus only elements which satisfy C^0 continuity are needed to satisfy the above requirements.

But other considerations may also influence the selection of proper element types. Several researchers modeling fluid flow have established that the interpolation functions for the velocity components must be at least one order higher than the pressure interpolation functions to prevent oscillations of the

field variable solution. Yamada, et al.[64] came to this conclusion by using a variational formulation. Olson and Tuann[45] showed that spurious rigid body modes in the solution appear when this criteria is violated. Other researchers who have supported this conclusion include Hood and Taylor[31] and Bercovier and Pironneau[7]. The restriction on the interpolation functions for the primitive variables arises from the uncoupled nature of the Navier-Stokes and continuity equations. This is because the continuity equation is simply a constraint on the velocities rather than an equation which fully couples velocities and pressure as the momentum equations do. A number of researchers modeling fluid flow using finite elements have concluded that quadratic interpolation functions for velocity and linear interpolation functions for pressure generally give the best performance[6]. An alternative approach to avoiding this problem is to uncouple the velocities and pressure by using a segregated method of solution. This is commonly done in finite difference formulations of the fluid equations, but it requires an additional convergence iteration to alternately satisfy the continuity and momentum equations.

Two useful sets of rectangular elements are the serendipity and Lagrangian families. The serendipity elements shown in Figure 4.1 contain only boundary nodes and their interpolation functions were derived by inspection. The Lagrangian elements shown in Figure 4.2

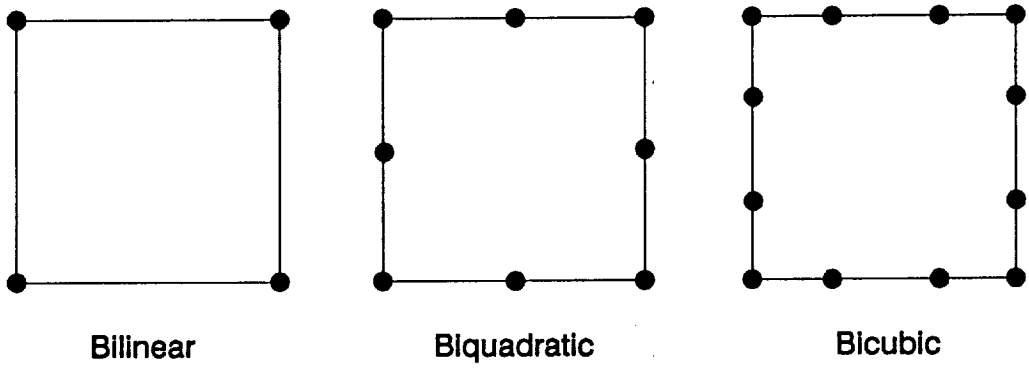

Figure 4.1 The Serendipity Family of Elements

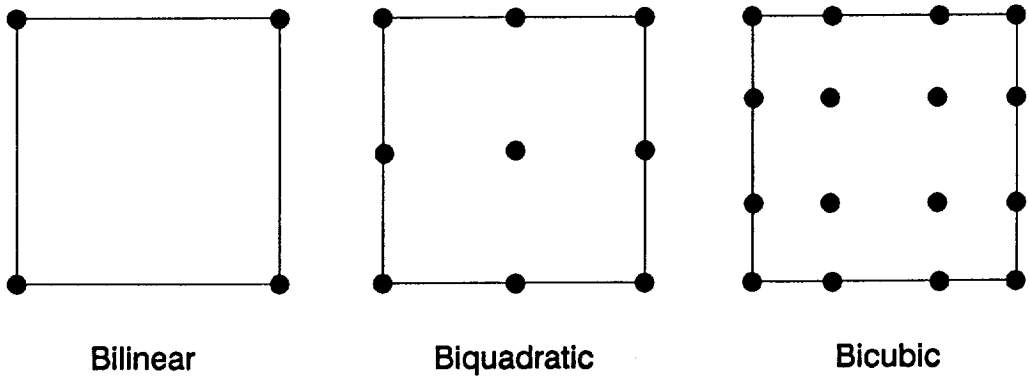

Figure 4.2 The Lagrangian Family of Elements

contain interior nodes and use the Lagrange polynomial as its interpolating function. Both the serendipity and Lagrangian element types have seen wide use in finite

element analysis.

As mentioned earlier, the geometry of the problem may also influence the selection of the element. One approach to modeling complex arbitrarily shaped boundaries is to use a body-fitted coordinate system. This approach can however add significantly to the modeling complexity. An alternative is to use curve-sided elements. *Isoparametric* elements are particularly useful as curve-sided elements. Isoparametric elements are elements whose geometry and field variable representations are described by polynomials of the same order. Using curve-sided elements, significantly fewer elements are usually required to fit a complex geometric boundary. Curved-sided isoparametric elements are commonly formed from either serindipity or Lagrangian elements.

Finally, other numerical considerations may also influence the selection of the proper element type. The numerical solution approach for the problem in this research requires the use of element mass lumping to prevent unrealistic oscillations in the field variables. This will be further discussed in the section on solving for the transient response. The use of element mass lumping has been shown by Gresho, et al.[28] to yield unstable solutions with the quadratic serindipity element under certain conditions. The Lagrangian biquadratic element however showed good accuracy and stability. During the course of this research, stability problems

were also observed when using the quadratic serindipity elements with element mass lumping.

For the reasons given above, the 4-node Lagrangian linear element was chosen for the pressure field and the 9-node Lagrangian biquadratic element was chosen for the energy and velocity fields.

Besides the element selection, the subdivision of the domain can have a significant influence on the solution. It is easiest to generate a uniform element mesh, however, this may not always provide the best representation of the field. Usually more elements should be placed in regions where the boundary is irregular. Also, in general, the elements used should be well proportioned, with the ratio of their largest dimensions to their smallest dimensions near unity. Nevertheless, it can be acceptable to use long thin elements if it is known that the field does not vary greatly in the elements lengthwise direction.

Provided that elements have been selected which satisfy the compatibility and completeness requirements, increasing the number of elements will provide improved solution accuracy. If there is an approximate solution to the problem, the finite element model accuracy can be improved by using a finer mesh in areas where high gradients are expected in the field variable. This increased accuracy is at the obvious expense of increased computational effort. It is generally good practice to obtain several solutions to a problem using an increasing

number of elements. By comparing results it can then be determined what is a sufficient number of elements for good solution accuracy.

Once the element type has been chosen, the interpolating functions for both the linear and quadratic elements can now be developed. The Lagrange polynomial is defined by

$$L_k(x) = \prod_{\substack{m=0 \\ m \neq k}}^{n} \frac{x - x_m}{x_k - x_m}$$

$$= \frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} \qquad (4.1)$$

Using the 4-node rectangular element and local coordinate system defined by Figure 4.3 the variation of some field variable ϕ can be written as

$$\phi(\xi,\eta) = N_1(\xi,\eta)\Phi_1 + N_2(\xi,\eta)\Phi_2 + N_3(\xi,\eta)\Phi_3 + N_4(\xi,\eta)\Phi_4$$

where Φ represents the nodal values of the field variable and the interpolating functions N are given by

$$N_1(\xi,\eta) = L_1(\xi)L_1(\eta), \quad N_2(\xi,\eta) = L_2(\xi)L_2(\eta), \quad \text{etc.}$$

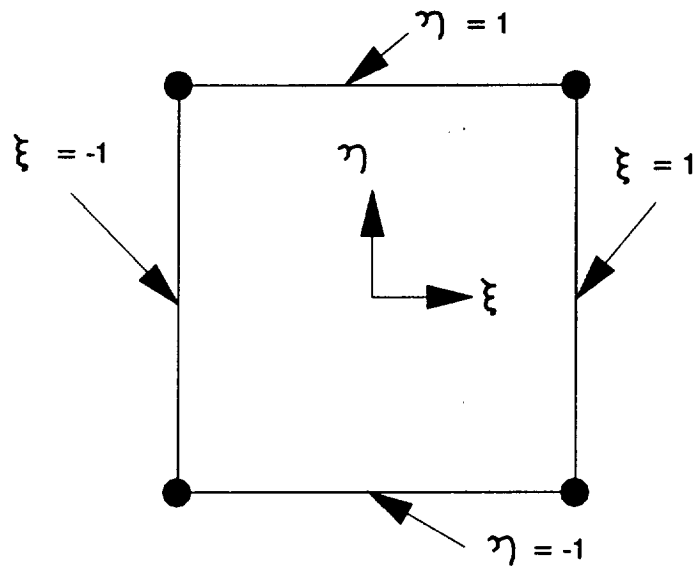

Figure 4.3 The 4-node Lagrangian Element and Coordinate System

These interpolation functions, formed as products of the Lagrange polynomial, are bilinear. The explicit expression for node 1 follows:

$$N_1(\xi,\eta) = L_1(\xi)L_1(\eta) = \frac{\xi-\xi_2}{\xi_1-\xi_2} \times \frac{\eta-\eta_4}{\eta_1-\eta_4} = \frac{\xi-1}{-1-1} \times \frac{\eta-1}{-1-1}$$

In this manner all of the linear Lagrangian interpolation

functions can be developed and are given by

$$N_1 = \frac{1}{4}(\xi\eta-\xi-\eta+1) \qquad N_2 = \frac{1}{4}(-\xi\eta+\xi-\eta+1)$$
$$N_3 = \frac{1}{4}(\xi\eta+\xi+\eta+1) \qquad N_4 = \frac{1}{4}(\xi\eta+\xi-\eta-1) \tag{4.2}$$

The quadratic Lagrangian interpolation functions can be developed similarly and are given by

$$N_1 = \frac{1}{4}(\xi\eta-\xi^2\eta-\xi\eta^2+\xi^2\eta^2) \qquad N_2 = \frac{1}{2}(-\eta+\eta^2+\xi^2\eta-\xi^2\eta^2)$$
$$N_3 = \frac{1}{4}(-\xi\eta-\xi^2\eta+\xi\eta^2+\xi^2\eta^2) \qquad N_4 = \frac{1}{2}(\xi+\xi^2-\xi\eta^2-\xi^2\eta^2)$$
$$N_5 = \frac{1}{4}(\xi\eta+\xi^2\eta+\xi\eta^2+\xi^2\eta^2) \qquad N_6 = \frac{1}{2}(\eta+\eta^2-\xi^2\eta-\xi^2\eta^2) \tag{4.3}$$
$$N_7 = \frac{1}{4}(-\xi\eta+\xi^2\eta-\xi\eta^2+\xi^2\eta^2) \qquad N_8 = \frac{1}{2}(-\xi+\xi^2+\xi\eta^2-\xi^2\eta^2)$$
$$N_9 = 1-\xi^2-\eta^2+\xi^2\eta^2$$

After selecting the element type, the solution domain is subdivided into a specified number of these elements. Fitting a curved boundary such as shown in Figure 4.4 could be done with many small elements, however in this case, a better fit would result if we could use curve-sided elements as in Figure 4.5. Ergatoudis et al.[17] were among the first to develop a general approach to creating such elements. Curved-sided elements are developed by

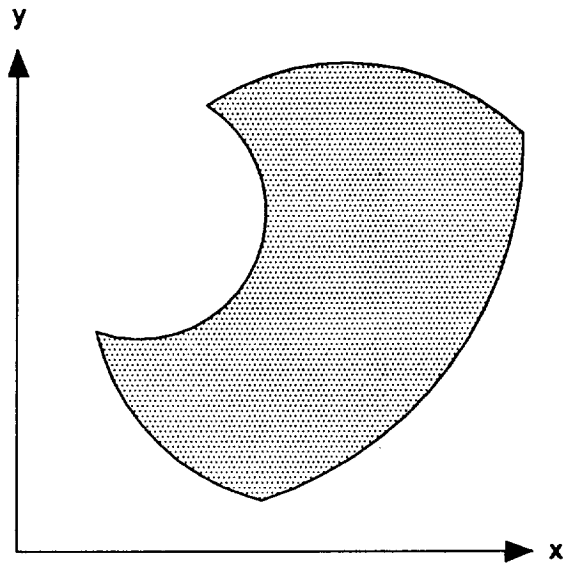

Figure 4.4 Example Solution Domain of Arbitrary Boundary

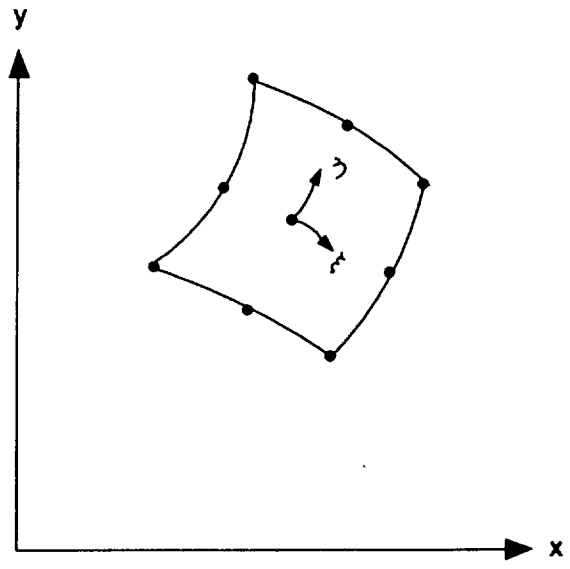

Figure 4.5 Curve-sided Element

transforming or mapping simple geometric shapes in some local coordinate system (ξ-η) into distorted shapes in the global Cartesian coordinate system (x-y). To construct a typical element such as is shown in Figure 4.5 we must start with a simpler "parent" element. Consider a parent element such as the 9-node quadratic Lagrangian element shown in Figure 4.6. The coordinates in the ξ-η plane may be transformed into the x-y plane using mapping functions of exactly the same form as the interpolation functions. These are given by

$$x = \sum_{i=1}^{9} N_i(\xi,\eta)x_i \qquad \text{and} \qquad y = \sum_{i=1}^{9} N_i(\xi,\eta)y_i \tag{4.4}$$

When making this transformation we must of course ensure that for every point in the local ξ-η coordinate system there is a unique corresponding point in global x-y coordinate system. If the transformation is not unique, the element can be greatly distorted causing unpredictable results on the solution.

This transformation technique can be useful in generating a set of element coordinates for a region in the solution domain such as that shown earlier. The region coordinates in the global cartesian system would become the x_i and y_i in equation (4.4). For a division of nine elements in the "parent" region shown in Figure 4.7,

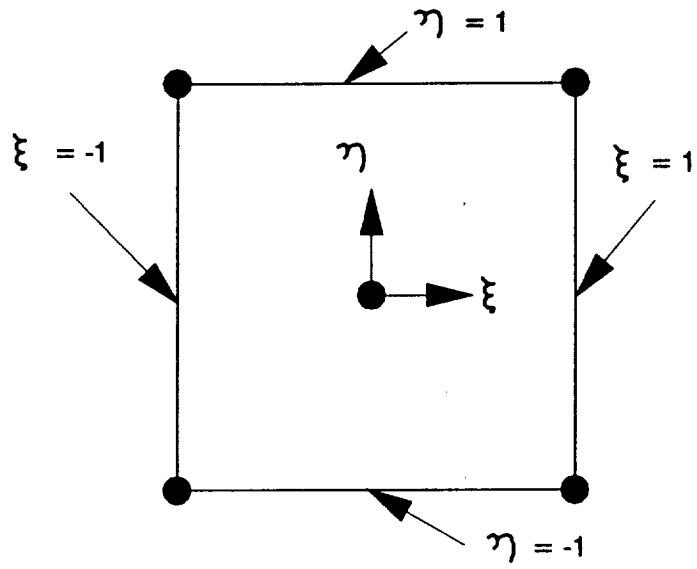

Figure 4.6 The 9-node Quadratic Lagrangian Element

the interpolation functions $N_i(\xi,\eta)$ are evaluated at the appropriate ξ and η having discrete values of -1, -0.667, 0.333, 0, -0.333, -0.667, and 1. The resulting elements in the global cartesian system are shown in Figure 4.8. This technique is quite useful for automatic element (grid) generation and is not restricted to equal numbers of divisions in the ξ and η directions. It is also possible to make the grid mesh finer in an area by slight shifts in the discrete ξ and η values given above. For further discussion on grid generation techniques see Zienkiewicz, et al.[65].

Before substituting the interpolating functions into the finite element equations, it is also necessary to

develop expressions for their derivatives. Following Huebner[32], the variation of some field variable ϕ within an element having r nodes is again expressed as

$$\phi(\xi,\eta) = \sum_{i=1}^{r} N_i(\xi,\eta)\ \Phi_i \tag{4.5}$$

The derivatives of the field variable can also be evaluated by

$$\frac{\partial\phi}{\partial x} = \sum_{i=1}^{r} \frac{\partial N_i}{\partial x}\ \Phi_i \qquad \text{and} \qquad \frac{\partial\phi}{\partial y} = \sum_{i=1}^{r} \frac{\partial N_i}{\partial y}\ \Phi_i \tag{4.6}$$

To evaluate the element matrices we must also express $\partial N_i/\partial x$ and $\partial N_i/\partial y$ in terms of local coordinates ξ and η. Applying the chain rule of differentiation yields

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \zeta} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = \left[J \right] \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} \tag{4.7}$$

where [J] is the Jacobian matrix. The Jacobian is evaluated using

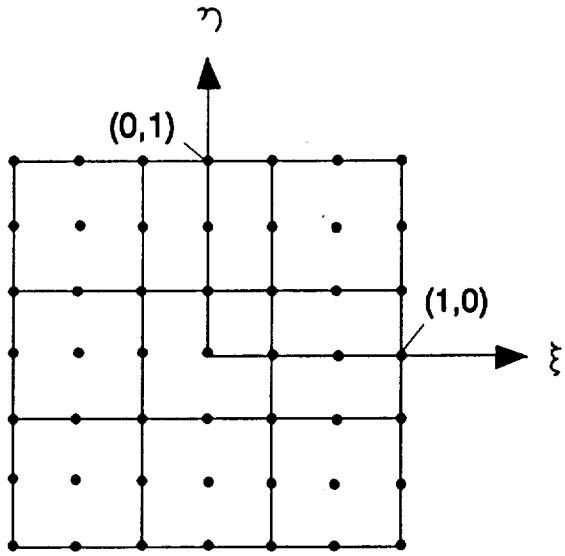

Figure 4.7 Nine-element "Parent" Region

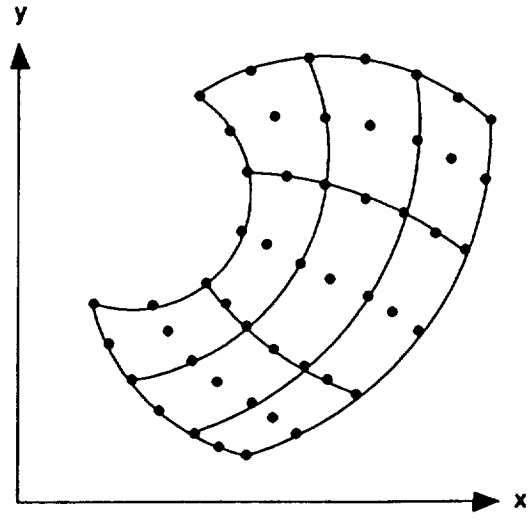

Figure 4.8 Curve-sided Elements in the Cartesian System

$$\left[J(\xi,\eta)\right] = \begin{bmatrix} \sum_{i=1}^{r} \frac{\partial N_i}{\partial \xi}(\xi,\eta)x_i & \sum_{i=1}^{r} \frac{\partial N_i}{\partial \xi}(\xi,\eta)y_i \\ \sum_{i=1}^{r} \frac{\partial N_i}{\partial \eta}(\xi,\eta)x_i & \sum_{i=1}^{r} \frac{\partial N_i}{\partial \eta}(\xi,\eta)y_i \end{bmatrix} \tag{4.8}$$

Rearranging, the derivatives of the shape functions in the two coordinate system are related by the inverse of the Jacobian as follows:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = \left[J \right]^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} \qquad \text{for} \quad i = 1,2,\ldots,r \tag{4.9}$$

From the above equations we can find the partial derivatives of the field variable in terms of the transformed coordinates ξ and η using

$$\begin{Bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{Bmatrix} = \left[J \right]^{-1} \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \cdots & \frac{\partial N_R}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \cdots & \frac{\partial N_R}{\partial \eta} \end{bmatrix} \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \vdots \\ \Phi_r \end{Bmatrix} \tag{4.10}$$

Finally, from advanced calculus we can express the differential area dx dy in terms of $d\xi\ d\eta$ using

$$dx\ dy = |J|\ d\xi\ d\eta \tag{4.11}$$

in which $|J|$ is the determinant of the Jacobian.

The validity of the element equations depends on the existence of the inverse of the Jacobian for each element. Also, the ξ-η to x-y coordinate mapping discussed earlier is unique only if the inverse of the Jacobian exists. A useful method for determining this uniqueness and the validity of the mapping is to evaluate the determinant of the Jacobian for all elements. If the sign of the determinant does not change throughout the solution domain, an acceptable mapping will be assured.

Formulating the Element Equations

To formulate the finite element equations from the governing equations we must apply the Galerkin method, substitute the interpolation functions for the field variables and their derivatives, and then perform the numerical integration on an element basis.

The velocity, pressure, and energy distribution within each element can be approximated by

$$u(x,y,t) = \lfloor N^u(x,y) \rfloor \{u(t)\} \tag{4.12}$$

$$v(x,y,t) = \lfloor N^{v}(x,y) \rfloor \{v(t)\} \tag{4.13}$$

$$P(x,y,t) = \lfloor N^{p}(x,y) \rfloor \{P(t)\} \tag{4.14}$$

$$e(x,y,t) = \lfloor N^{e}(x,y) \rfloor \{e(t)\} \tag{4.15}$$

Before applying the Galerkin method to the conservation of energy equation it is necessary to first linearize the nonlinear convective terms. Let u^* and v^* be an approximate solution to the velocity field and P^* be an approximate solution to the pressure field (such as the results from a previous iteration). Now applying the Galerkin method, the linearized energy equation yields

$$\int_{\Omega} N_i^e \left(\rho \frac{\partial e}{\partial t} + \rho u^* \frac{\partial e}{\partial x} + \rho v^* \frac{\partial e}{\partial y} + P^* \frac{\partial u^*}{\partial x} + P^* \frac{\partial v^*}{\partial y} - \frac{\partial T}{\partial x}\left(k \frac{\partial T}{\partial x}\right) - \frac{\partial T}{\partial y}\left(k \frac{\partial T}{\partial y}\right) \right) d\Omega = 0 \tag{4.16}$$

Integrating the last two conduction terms by parts using Green's theorem

$$\int_{\Omega} a(\nabla \cdot b)\ d\Omega = \int_{\Sigma} a(b \cdot \vec{n})\ d\Sigma - \int_{\Omega} b \cdot \nabla a\ d\Omega \tag{4.17}$$

$$\text{letting } a = N_i^* \qquad \vec{b} = k\frac{\partial T}{\partial x}\vec{n}_i + k\frac{\partial T}{\partial y}\vec{n}_j \tag{4.18}$$

the energy equation becomes

$$\int_{\Omega} N_i^* \left(\rho\frac{\partial e}{\partial t} + \rho u^*\frac{\partial e}{\partial x} + \rho v^*\frac{\partial e}{\partial y} + P^*\frac{\partial u^*}{\partial x} + P^*\frac{\partial v^*}{\partial y}\right) d\Omega \tag{4.19}$$

$$+ \int_{\Omega}\left(\frac{\partial N_i}{\partial x}\ k\frac{\partial T}{\partial x} + \frac{\partial N_i}{\partial y}\ k\frac{\partial T}{\partial y}\right) d\Omega$$

$$+ \int_{\Sigma} N_i^* \left(k\frac{\partial T}{\partial x}\vec{n}_i + k\frac{\partial T}{\partial y}\vec{n}_j\right) d\Sigma = 0$$

The surface integral in the equation above is the natural boundary condition and allows for the introduction of the prescribed heat flux, q, boundary condition.

Substituting the approximations for the field variables and rearranging, the energy equation for the interpolating function at node i becomes

$$\int_\Omega N_i^* \left(\rho \frac{\partial e}{\partial t}\right) d\Omega + \int_\Omega N_i^* \left(\left[\rho u^* \frac{\partial N}{\partial x}\right] + \left[\rho v^* \frac{\partial N}{\partial y}\right]\right) d\Omega \{e\} + \int_\Sigma N_i^* \, q \, d\Sigma - \int_\Omega N_i^* \left(P^* \frac{\partial u^*}{\partial x} + P^* \frac{\partial v^*}{\partial y}\right) d\Omega - \int_\Omega \left(\frac{\partial N_i}{\partial x} k \frac{\partial T}{\partial x}\right) + \left(\frac{\partial N_i}{\partial y} k \frac{\partial T}{\partial y}\right) d\Omega \tag{4.20}$$

Applying the Galerkin method now to the momentum equations it is necessary to also linearize the nonlinear convective terms, by again letting u^* and v^* be an approximate solution to the velocity field. Taking the x-direction momentum equation and applying Galerkin's criterion yields

$$\int_\Omega N_i^v \left(\rho \frac{\partial u}{\partial t} + \rho u^* \frac{\partial u}{\partial x} + \rho v^* \frac{\partial u}{\partial y} - \frac{\partial (\sigma_x - P)}{\partial x} - \frac{\partial \tau_{xy}}{\partial y} + \rho g_x\right) d\Omega = 0 \tag{4.21}$$

Integrating the viscous force terms by parts using Green's theorem

letting $\quad a = N_i^v \qquad \vec{b} = \sigma_x \vec{n}_i + \tau_{xy} \vec{n}_j$

yields

$$\int_{\Omega} N_i^v \left(\frac{\partial (\sigma_x - P)}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} \right) d\Omega = \int_{\Sigma} N_i^v \left((\sigma_x - P)\vec{n}_i + \tau_{xy}\vec{n}_j \right) d\Sigma - \int_{\Omega} \left((\sigma_x - P)\frac{\partial N_i}{\partial x} + \tau_{xy} \frac{\partial N_i}{\partial y} \right) d\Omega \tag{4.22}$$

Now defining

$$\overline{\sigma}_x = (\sigma_x - P)\vec{n}_i + \tau_{xy} \vec{n}_j \tag{4.23}$$

and introducing the velocity components with

$$\sigma_x = 2\mu \frac{\partial u}{\partial x} - \frac{2}{3}\mu \nabla \cdot V$$

$$\sigma_y = 2\mu \frac{\partial v}{\partial y} - \frac{2}{3}\mu \nabla \cdot V \tag{4.24}$$

$$\tau_{xy} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)$$

The x-momentum equation becomes

$$\int_\Omega N_i^v \left(\rho\frac{\partial u}{\partial t} + \rho u^*\frac{\partial u}{\partial x} + \rho v^*\frac{\partial u}{\partial y}\right) d\Omega$$
$$+ \int_\Omega \frac{\partial N_i}{\partial x}\left(2\mu\frac{\partial u}{\partial x} - \frac{2}{3}\mu\frac{\partial u}{\partial x} - \frac{2}{3}\mu\frac{\partial v}{\partial y}\right) d\Omega$$
$$+ \int_\Omega \frac{\partial N_i}{\partial y}\ \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) d\Omega - \int_\Omega \frac{\partial N_i}{\partial x}\ P\ d\Omega \qquad (4.25)$$
$$+ \int_\Omega N_i^v \rho g_x\ d\Omega = \int_\Sigma N_i\,\bar{\sigma}_x\ d\Sigma$$

Substituting the approximations for the field variables and rearranging, the x-momentum equation for node i of the element becomes

$$\int_\Omega N_i^v \left(\rho\frac{\partial u}{\partial t}\right) d\Omega$$
$$+ \int_\Omega \left(N_i^v \left\lfloor \rho u^*\frac{\partial N}{\partial x}\right\rfloor + N_i^v \left\lfloor \rho v^*\frac{\partial N}{\partial y}\right\rfloor\right) d\Omega \left\{u\right\}$$
$$+ \int_\Omega \left(\frac{4}{3}\frac{\partial N_i}{\partial x}\left\lfloor \mu\frac{\partial N}{\partial x}\right\rfloor + \frac{\partial N_i}{\partial y}\left\lfloor \mu\frac{\partial N}{\partial y}\right\rfloor\right) d\Omega \left\{u\right\}$$
$$+ \int_\Omega \left(-\frac{2}{3}\frac{\partial N_i}{\partial x}\left\lfloor \mu\frac{\partial N}{\partial y}\right\rfloor + \frac{\partial N_i}{\partial y}\left\lfloor \mu\frac{\partial N}{\partial x}\right\rfloor\right) d\Omega \left\{v\right\} \qquad (4.26)$$
$$+ \int_\Omega \frac{\partial N_i}{\partial x}\left\lfloor N^P\right\rfloor d\Omega \left\{P\right\} = \int_\Sigma \bar{\sigma}_x N_i^V\, d\Sigma - \int_\Omega N_i^v \rho g_x\ d\Omega$$

The integral over the surface Σ is the natural boundary condition and can be used to introduce surface tractions.

The y momentum equation can be developed similarly and is given by

$$\int_\Omega N_i^v \left(\rho \frac{\partial v}{\partial t}\right) d\Omega$$

$$+ \int_\Omega \left(-\frac{2}{3} \frac{\partial N_i}{\partial y} \left[\mu \frac{\partial N}{\partial x}\right] + \frac{\partial N_i}{\partial x} \left[\mu \frac{\partial N}{\partial y}\right]\right) d\Omega \left\{u\right\} \qquad (4.27)$$

$$+ \int_\Omega \left(N_i^v \left[\rho u^* \frac{\partial N}{\partial x}\right] + N_i^v \left[\rho v^* \frac{\partial N}{\partial x}\right]\right) d\Omega \left\{v\right\}$$

$$+ \int_\Omega \left(\frac{4}{3} \frac{\partial N_i}{\partial y} \left[\mu \frac{\partial N}{\partial y}\right] + \frac{\partial N_i}{\partial x} \left[\mu \frac{\partial N}{\partial x}\right]\right) d\Omega \left\{v\right\}$$

$$+ \int_\Omega \frac{\partial N_i}{\partial y} \left[N^P\right] d\Omega \left\{P\right\} = \int_\Sigma \bar{\sigma}_y N_i^v \, d\Sigma - \int_\Omega \rho g_y N_i^v \, d\Omega$$

Finally, applying the Galerkin method to the conservation of mass equation (continuity), using weighting factors equal to the interpolating functions for pressure at each node i yields

$$\int_\Omega N_i^P \left[\frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y}\right] d\Omega = 0 \qquad (4.28)$$

Substituting the approximations for the field variables u and v yields

$$\int_\Omega N_i^P \left[\rho \frac{\partial N}{\partial x}\right] d\Omega \left\{u\right\} + \int_\Omega N_i^P \left[\rho \frac{\partial N}{\partial y}\right] d\Omega \left\{v\right\} = 0 \qquad (4.29)$$

It is important to note that all three conservation equations are applied throughout the solution domain. However, since there is no motion of the material in the

solid something must be done to prevent convective transport and influence of mass on the buoyant forcing function.

The simplest method for preventing fluid motion in the solid is to set the velocities u^* and v^* to zero and to increase the viscosity to some large value for those nodes which are in the solid state. As discussed earlier, during the phase change process, some materials develop a mushy region in which the interface between the phases is not sharp. If information is available on how the viscosity (and other properties) of the material varies as it undergoes phase change, it can be used in the material property data to improve the modeling of the flow near the phase front. For the cases studied in this research, the phase front was relatively sharp and the exact value of the assumed viscosity in the two phase region had little influence on the results. Also, because the results from the fluid equations are not applicable within the solid region, the exact influence of the viscosity in that region is unimportant.

The second problem alluded to above involves the influence of the density distribution within the solid on the overall buoyant forcing function. In reality the body forces on the solid are balanced by internal and boundary stresses. In this formulation, however, no such mechanism exists since no equations from solid mechanics were included. Because of this problem, incorrect body force

terms can develop if the density of the solid is significantly different from that of the liquid. To prevent this, the buoyant force term is first reformulated in terms of density change about a reference density such as

$$\int_{\Omega} \rho \left(1 - \frac{\rho}{\rho_0}\right) g \, N_i \, d\Omega$$

This approach is quite common in formulations of the Navier-Stokes equations which use the coefficient of thermal expansion. The reference density ρ_0 is taken to be the density of the fluid at the fusion temperature. To prevent the influence of the solid in the buoyant force terms, the density is set equal to the reference density for those nodes which are in the solid region. Note that this is done only in the evaluation of this integral while all other integrals are evaluated using the appropriate density for each state.

Because the fluid motion is influenced by buoyancy forces and the material properties vary with time, the energy, momentum, and mass equations are directly coupled and must be solved simultaneously. Two approaches have been used in the past to solve the steady solutions. Taylor and Ijam[57] solved the equations simultaneously. Gartling[24] used an algorithm in which the equations are segregated and the solution alternates between the them.

During the course of this research, both approaches were employed and evaluated. The method which solves the three equations simultaneously was found to require significantly more computer memory and computations in solving the equations. The alternating solution method required an iterative algorithm, however, this did not substantially change the overall algorithm. This is because iterations are required to satisfy the nonlinear terms in the Navier-Stokes equations. Based on these studies, the segregated approach to solving the energy and flow equations was adopted in the present analysis.

The energy, momentum, and mass equations above were given for the weighting functions at each node i in the element. By inspection, we can write the energy equation for all nodes of each element as

$$\left[C \right] \left\{ \dot{e} \right\} + \left[K_A^{\bullet} \right] \left\{ e \right\} = \left\{ R_q + R_P + R_C \right\} \qquad (4.30)$$

Similarly, we can write the momentum and continuity equations for all nodes of each element as

$$\left[\begin{array}{c|c|c} M & 0 & 0 \\ \hline 0 & M & 0 \\ \hline 0 & 0 & 0 \end{array}\right] \left\{\begin{array}{c} \dot{u} \\ \hline \dot{v} \\ \hline \dot{P} \end{array}\right\} \tag{4.31}$$

$$+ \left[\begin{array}{c|c|c} K_A^v + \frac{4}{3}K_{11} + K_{22} & K_{12} - \frac{2}{3}K_{21} & L_1 \\ \hline K_{21} - \frac{2}{3}K_{12} & K_A^v + \frac{4}{3}K_{22} + K_{11} & L_2 \\ \hline L_{c1} & L_{c2} & 0 \end{array}\right] \left\{\begin{array}{c} u \\ \hline v \\ \hline P \end{array}\right\}$$

$$= \left\{\begin{array}{c} R_x \\ \hline R_y \\ \hline 0 \end{array}\right\}$$

in which

$$\left[C \right] = \int_\Omega \left\{ N^* \right\} \left[\rho N^* \right] \, d\Omega$$

$$\left[M \right] = \int_\Omega \left\{ N^v \right\} \left[\rho N^v \right] \, d\Omega$$

$$\left[K_A^e \right] = \int_\Omega \left\{ \rho u^* N^e \right\} \left\lfloor \frac{\partial N^e}{\partial x} \right\rfloor d\Omega + \int_\Omega \left\{ \rho v^* N^e \right\} \left\lfloor \frac{\partial N^e}{\partial y} \right\rfloor d\Omega$$

$$\left[K_A^v \right] = \int_\Omega \left\{ \rho u^* N^v \right\} \left\lfloor \frac{\partial N^v}{\partial x} \right\rfloor d\Omega + \int_\Omega \left\{ \rho v^* N^v \right\} \left\lfloor \frac{\partial N^v}{\partial y} \right\rfloor d\Omega$$

$$\left[L_1 \right] = - \int_\Omega \left\{ \frac{\partial N^v}{\partial x} \right\} \left\lfloor N^p \right\rfloor d\Omega$$

$$\left[L_2 \right] = - \int_\Omega \left\{ \frac{\partial N^v}{\partial y} \right\} \left\lfloor N^p \right\rfloor d\Omega$$

$$\left[K_{11} \right] = \int_\Omega \left\{ \frac{\partial N^v}{\partial x} \right\} \left\lfloor \mu \frac{\partial N^v}{\partial x} \right\rfloor d\Omega$$

$$\left[K_{12} \right] = \int_\Omega \left\{ \frac{\partial N^v}{\partial y} \right\} \left\lfloor \mu \frac{\partial N^v}{\partial x} \right\rfloor d\Omega$$

$$\left[K_{21} \right] = \int_\Omega \left\{ \frac{\partial N^v}{\partial x} \right\} \left\lfloor \mu \frac{\partial N^v}{\partial y} \right\rfloor d\Omega$$

$$\left[K_{22} \right] = \int_{\Omega} \left\{ \frac{\partial N^v}{\partial y} \right\} \left\lfloor \mu \frac{\partial N^v}{\partial y} \right\rfloor \, d\Omega$$

$$\left[L_{C1} \right] = \int_{\Omega} \left\{ N^p \right\} \left\lfloor \rho \frac{\partial N^v}{\partial x} \right\rfloor \, d\Omega$$

$$\left[L_{C2} \right] = \int_{\Omega} \left\{ N^p \right\} \left\lfloor \rho \frac{\partial N^v}{\partial y} \right\rfloor \, d\Omega$$

$$\left\{ R_q \right\} = \int_{\Sigma} q \left\{ N^e \right\} \, d\Sigma$$

$$\left\{ R_p \right\} = - \int_{\Omega} N_i^e \left(P^* \frac{\partial u^*}{\partial x} + P^* \frac{\partial v^*}{\partial y} \right) \, d\Omega$$

$$\left\{ R_C \right\} = - \int_{\Omega} \left(\frac{\partial N_i}{\partial x} \; k \frac{\partial T}{\partial x} \right) + \left(\frac{\partial N_i}{\partial y} \; k \frac{\partial T}{\partial y} \right) \, d\Omega$$

$$\left\{ R_x \right\} = \int_{\Sigma} \bar{\sigma}_x \left\{ N^v \right\} \, d\Sigma - \int_{\Omega} \rho \left(1 - \frac{\rho}{\rho_0} \right) g_x \left\{ N^v \right\} \, d\Omega$$

$$\left\{ R_y \right\} = \int_{\Sigma} \bar{\sigma}_y \left\{ N^v \right\} \, d\Sigma - \int_{\Omega} \rho \left(1 - \frac{\rho}{\rho_0} \right) g_y \left\{ N^v \right\} \, d\Omega$$

To evaluate the finite element matrices requires integrating functions of the form

$$\int_{\Omega} f(x,y)\ d\Omega = \int_{-1}^{1}\int_{-1}^{1} f'(\xi,\eta)\ |J|\ d\xi\ d\eta \qquad (4.32)$$

The Jacobian is a function of ξ and η and cannot be explicitly evaluated because the coefficients are polynomials, thus some type of numerical integration must be used. The Gauss-Legendre method is chosen here because it requires relatively few sampling points to obtain a good degree of accuracy. This method involves evaluating the function at the sampling points and weighting the results as follows

$$\int_{-1}^{1}\int_{-1}^{1} f'(\xi,\eta)\ |J|\ d\xi\ d\eta \cong \sum_{i=0}^{n}\ \sum_{j=0}^{n} W_i W_j f'(\xi_i,\eta_j) \qquad (4.33)$$

Table 4.1 gives the location and weights for the Gauss-Legendre Quadrature up to order 4

Table 4.1 Location and Weights for Gauss-Legendre Quadrature to Order 4

Order	Location	Weight
n=1	±0.5773502691	1.0000000000
n=2	0.0000000000	0.8888888889
	±0.7745966692	0.5555555556
n=3	±0.3399810435	0.6521451548
	±0.8611363115	0.3478548451
n=4	0.0000000000	0.5688888889
	±0.5384693101	0.4786286704
	±0.9061798459	0.2369268850

Figure 4.9 shows an example of the location of these sampling points for a typical element using a Gauss-Legendre Quadrature of order 2.

To accurately evaluate the volume integrals, Gauss-Legendre integration of order 1 is required for the bilinear elements and order 2 for the biquadratic elements.

Assembling the System Equations

Once the behavior of each element has been developed, the overall system is modeled by assembling these element equations into a set of system equations. To do this, the element equations, which were evaluated at the nodes of

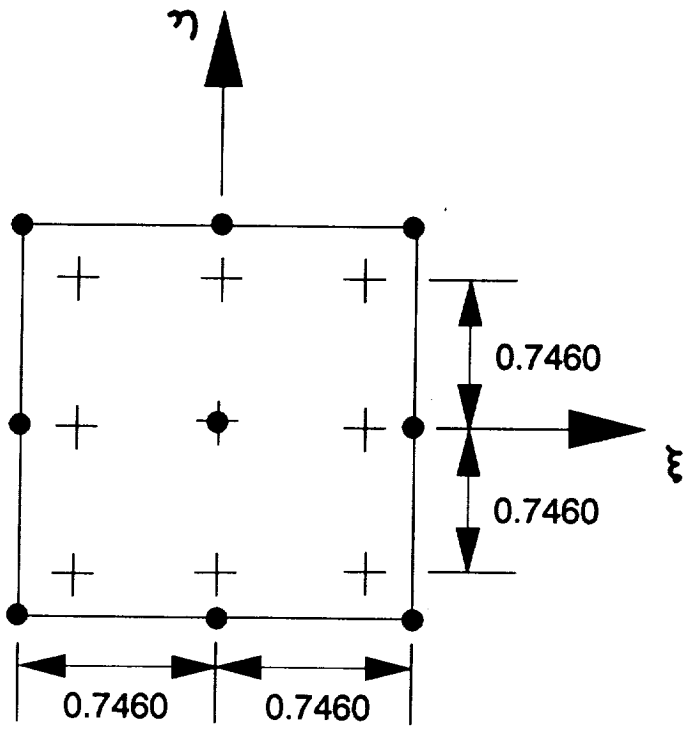

Figure 4.9 Location of the Gauss-Legendre Integration (Order 2) Sampling Points in a Typical Element

each element, must now be transformed into the global node numbering scheme. The numbering schemes for the quadratic Lagrangian element and the global node numbering scheme for an example four element region are shown in Figure 4.10. For the sake of explanation assume there is a single field variable at each node, the total number of system nodal variables is equal to the number of global nodes, n (i.e. 25 in Figure 4.10). The nine local element

nodes for the upper left element in Figure 4.10 correspond to the global nodes 11, 12, 13, 8, 3, 2, 1, 6, and 7. The

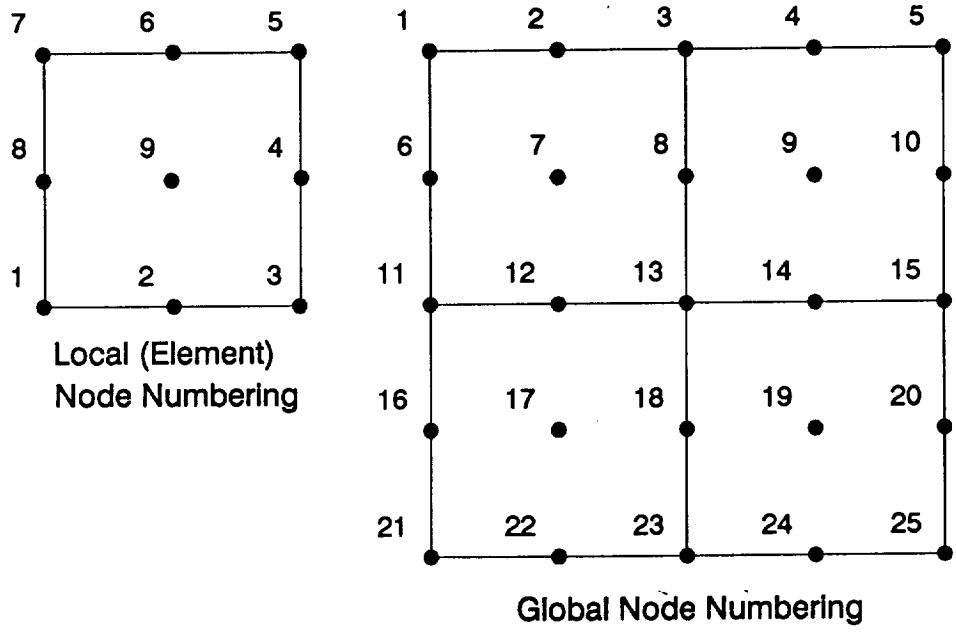

Figure 4.10 Transformation of the Local Node Numbering to the Global Node Numbering Scheme

node relationship is only slightly more complicated for the case where multiple field variables exist at the same geometric node location. The procedure for assembling the system equations is as follows:

1. For n global nodes, set up two n × n and one n × 1 null matrices (all zero entries) as the system

mass, stiffness and resultant matrices.

2. Take one element and use the relationship between the local and global node numbers to replace the indices in the element matrices with the corresponding global node numbers.
3. Insert those terms into the appropriate locations in the system matrices. If a term is inserted in a location where another term has already been placed, it should be added to the value at that location.
4. Repeat the procedure starting at step 2 for all of the elements.

The result will be a system of equations of the form

$$\overset{n\times n}{\left[\ M\ \right]}\overset{n\times 1}{\left\{\dot{\phi}\right\}} + \overset{n\times n}{\left[\ K\ \right]}\overset{n\times 1}{\left\{\phi\right\}} = \overset{n\times 1}{\left\{R(t)\right\}} \qquad (4.34)$$

where again ϕ is the unknown field variables, [M] the mass matrix, [K] the stiffness matrix, and {R} the resultant column matrix.

Solving for the Transient Response

The solution of the final set of simultaneous nonlinear ordinary differential equations is a formidable

task combining transient time integration with an iteration scheme at each time step to handle the nonlinear terms.

An approach to directly integrating these coupled equations in time is to use recursive algorithms based on the finite difference method. Let t_j be a typical time in the transient response such that

$$t_{j+1}=t_j+\Delta t \ , \quad \text{for } j=0,1,2,\ldots$$

A general family of algorithms can be developed by introducing a parameter θ such that

$$t_{j+\theta}=t_j+\theta\Delta t \ , \quad \text{for } 0\leq\theta\leq 1$$

The system equations at time $t_{j+\theta}$ can be written as

$$\left[M \right]\left\{\dot{\phi}\right\}_{j+\theta}+\left[K \right]\left\{\phi\right\}_{j+\theta}=\left\{R(t_{j+\theta})\right\} \tag{4.35}$$

Introducing the approximations

$$\left\{\dot{\phi}\right\}_{j+\theta} = \frac{\left\{\phi\right\}_{j+1} - \left\{\phi\right\}_{j}}{\Delta t}$$

$$\left\{\phi\right\}_{j+\theta} = (1-\theta)\left\{\phi\right\}_{j} + \theta\left\{\phi\right\}_{j+1} \tag{4.36}$$

$$\left\{R(t_{j})_{\theta}\right\} = (1-\theta)\left\{R\right\}_{j} + \theta\left\{R\right\}_{j+1}$$

Substituting these approximations into the system equations yields

$$\left[\theta[K] + \frac{1}{\Delta t}[M]\right]\left\{\phi\right\}_{j+1} = \left[(\theta-1)[K] + \frac{1}{\Delta t}[M]\right]\left\{\phi\right\}_{j} + (1-\theta)\left\{R\right\}_{j} + \theta\left\{R\right\}_{j+1} \tag{4.37}$$

Rearranging, a general recursion formula for calculating the unknown field variables $\{\phi\}_{j+1}$ at the end of the time step to the known values $\{\phi\}_{j}$ at the start of the time step is given by

$$\left[\,\bar{K}\,\right]\left\{\phi\right\}_{j+1} = \left\{\bar{R}\right\}_{j+1} \tag{4.38}$$

in which

$$\left[\bar{K} \right] = \theta \left[K \right] + \frac{1}{\Delta t} \left[M \right]$$

$$\left\{ \bar{R} \right\}_{j+1} = \left[(\theta - 1)[K] + \frac{1}{\Delta t}[M] \right] \left\{ \phi \right\}_j + (1-\theta) \left\{ R \right\}_j + \theta \left\{ R \right\}_{j+1}$$

This equation represents a family of popular time-marching algorithms. Table 4.2 describes some of the members of this family.

Table 4.2 Characteristics of Recursive Time-Marching Algorithms

Algorithm	θ	Accuracy	Stability
Euler or Forward Difference	0	1st Order	Conditional
Crank-Nicolson	1/2	2nd Order	Unconditional
Galerkin	2/3	1st Order	Unconditional
Backward Difference	1	1st Order	Unconditional

All of the algorithms given in Table 4.2 are first order accurate with the exception of the Crank-Nicolson method which is second order accurate. The terms *first order* and *second order* refer to the truncation errors in the finite difference approximations. First order accuracy means that the error is proportional to the first power of the time step Δt, and second order accuracy means

the error is proportional to the second power of Δt.

Many studies have been made to evaluate the relative accuracy of each of the methods given in Table 4.2. Perhaps one of the most relevant was performed by Hogge[29] in which he studied a nonlinear heat transfer problem. His detailed investigation of the relative accuracy of the various methods concluded that the Crank-Nicolson scheme ($\theta=1/2$) is indeed the most accurate of these methods. He did note however, that more sophisticated schemes spanning several time steps can give even better accuracy.

Table 4.2 also characterizes the stability of the various methods. Stability means that the computed response does not oscillate and grow without bounds unrealistically. Stability is ensured for $\theta \geq 1/2$. All of the methods given in Table 4.2 are unconditionally stable except the Euler forward difference method. For methods where $\theta<1/2$ a stable solution results only for time steps less than some critical value. It should be noted however that the selection of time step is important even for methods with $\theta \geq 1/2$. Though the computed response with one of these methods will not grow unrealistically without bound, it may exhibit spurious oscillations and decreased accuracy with a very large time step. With either method it is good practice to solve the integration with several different time steps and compare the results.

In addition to accuracy and stability, other considerations effect the selection of time integration

algorithm. The two common approaches to solving the overall system of equations are the explicit forward difference scheme and the implicit one parameter θ schemes.

The explicit forward difference scheme computes the field variables at time t_{j+1} from a set of uncoupled system equations. It does however require a lumped mass matrix. The term *lumped* is used to differentiate it from the original (or *consistent*) mass matrix. Lumped matrices are formed by assigning each node an amount of mass which can be attributed to that location. The most common approach to forming a lumped mass matrix is to sum the coefficients of the rows of the consistent mass matrix and use these as terms along the diagonal.. The explicit forward difference scheme using a lumped mass matrix may result in a significant computational savings over implicit schemes because the field variable can be computed without solving the system of simultaneous equations at each iteration. Again it has the disadvantage of only conditional stability with selection of the time step. It also requires a constant time step throughout the solution.

The implicit "θ" algorithms compute the field variables from a coupled set of system equations. The time step for the implicit algorithms again is not restricted by the stability constraint discussed earlier. In addition, the time step can be varied throughout the

transient solution. This is a significant advantage of the implicit algorithms for this research problem. During the solution of a phase change problem, the maximum allowable time step may vary significantly. This is because the solution domain may be all solid, multi-phase, or fluid. Each of these situations could have very different rates of response. For example, during the time the material is all solid and only the energy equation is important, the allowable time step might be significantly larger than when fluid is present and results from the momentum equations become important.

Implicit algorithms permit either lumped or consistent mass matrices. However, the choice of lumped versus consistent mass matrices is not always be easily resolved. Considering their formulation, consistent mass matrices are thought to be more accurate. Many researchers, however, have found insignificant loss of accuracy using the lumped approach. In fact, virtually all finite difference formulations use the lumped mass approach. Emery, et al.[16] found that the consistent approach could sometimes even predict unrealistic oscillations in the temperature distributions. This was most often observed near areas of sharp transients. The lumped approach however gave solutions which were intuitively obvious. During the course of this research, I also observed unrealistic oscillations in the field variables while using a consistent mass matrix approach.

This was particularly evident near the phase change front where the material properties varied greatly.

Because of the considerations and observations discussed above, the lumped mass, implicit Crank-Nicolson scheme was adopted for this research.

After application of the recursive time integration approach, the result will be a reduced set of system equations of the form

$$\left[K \right]\left\{\phi\right\} = \left\{R\right\} \tag{4.39}$$

It is now necessary to account for any boundary conditions which were not already applied as natural boundary conditions. In particular, these include any prescribed value boundary conditions. Usually, at least one and sometimes more than one nodal value must be prescribed to make the system equations nonsingular and provide a unique solution. There are several ways to apply these prescribed boundary conditions and modify the set of system equations. The one chosen here is relatively straightforward and can be described best by example. Suppose there are only four system equations given by

$$\begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix} \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{Bmatrix} = \begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{Bmatrix}$$

Consider applying prescribed nodal values specified as

$$\phi_1 = \Psi_1 \ , \qquad \phi_3 = \Psi_3$$

The modified system equations will become

$$\begin{bmatrix} k_{22} & k_{24} \\ k_{42} & k_{44} \end{bmatrix} \begin{Bmatrix} \phi_2 \\ \phi_4 \end{Bmatrix} = \begin{Bmatrix} r_2 - k_{21}\Psi_1 - k_{23}\Psi_3 \\ r_4 - k_{41}\Psi_1 - k_{43}\Psi_3 \end{Bmatrix} .$$

Note that once these prescribed boundary conditions have been applied, the number of equations and nodal unknowns to be solved for is reduced since it is not necessary to solve for the prescribed values. For problems with many prescribed boundary values, such as flow inside a containment vessel, the size of the system equations can be reduced significantly. From a computational standpoint, this can result in a substantial decrease in the time to invert the large system matrix.

Note also that since the dependent variable in the energy equation is internal energy, prescribed temperature boundary conditions must be handled with an additional

step. This step is simply to apply the equation of state model to convert the prescribed temperature value to a prescribed internal energy value.

Before summarizing the overall numerical approach, some discussion is warranted on the method for determining the state and material properties used throughout the solution domain. There were several requirements for this equation of state and material property model. First, the method should be able to describe reasonably complex material property characteristics. Second, the amount of input data to describe these material properties should be minimal. And finally, the method should be computationally efficient. Usually, either tabular data interpolation or "curve fitting" approaches are used for such models. Since two independent material properties are used (e.g. pressure and internal energy) the tabular data approach requires double interpolation and the "curve" in the second approach is really a surface.

Because of the requirements discussed above, a surface fitting approach to the material state properties was developed. In such an approach, the thermodynamic surface representing the dependent property as a function of the two independent properties is represented by one or more regions as shown in Figure 4.11. These regions are described by quadrilaterals defined by values at eight points along their sides. The point representing the independent properties is projected onto one of these

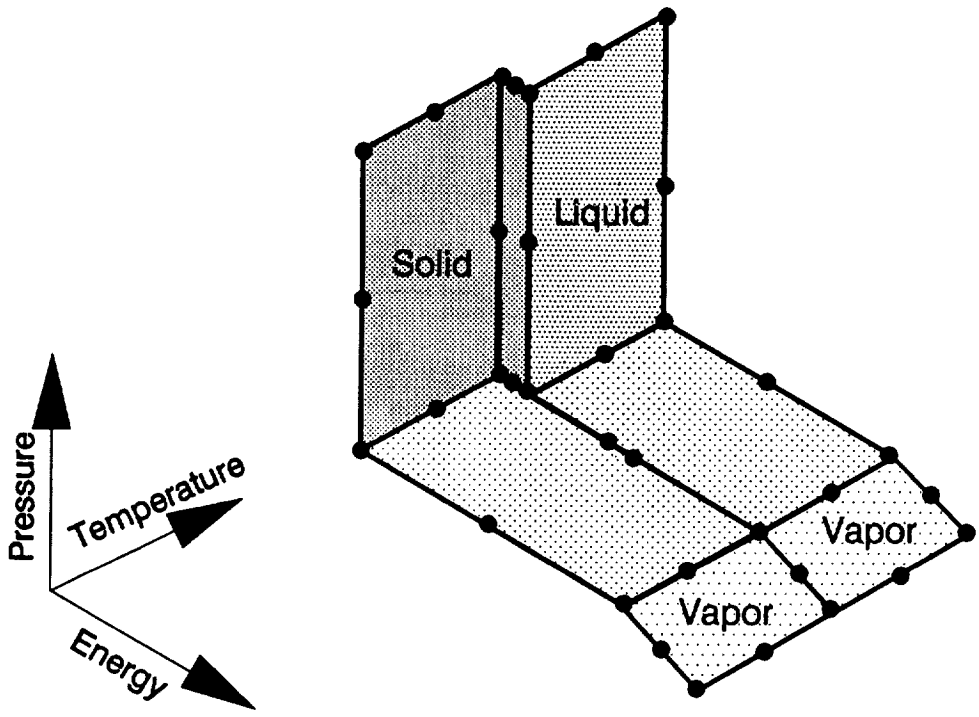

Figure 4.11 Surface Fitting of the Material State Properties

quadrilaterals. A double quadratic regression analysis of that quadrilateral is then used to yield the value of the dependent property.

The overall numerical solution procedure described in this chapter is summarized in Table 4.3. This numerical solution procedure is implemented in the computer program, PHASTRAN, which is described in Appendix A and listed in Appendix B.

Table 4.3 Overall Solution Procedure

```
Initial calculations
    Read input data
    Generate element coordinates
    Initialize field variables
    Evaluate material state properties
At each time step
    Increment time
    Iterate to converge nonlinear terms
        Form the element equations
        Assemble [M], [K], and {R} system matrices
        Modify system matrices to form [K̄] and {R̄}
        Apply the prescribed boundary conditions
        Solve the simultaneous equations
        Update the material state properties
        Set u* and v* to zero in the solid
    Check for convergence of the field variables
Advance to the next time step
```

CHAPTER 5

RESULTS AND VERIFICATION

Direct experimental verification of the multi-dimensional phase change problem is difficult at best. Verification is especially difficult for containment vessels with complicated shapes and nonuniform boundary conditions. Even for very simple geometries, verification would rely on results from other numerical methods or the very few experimental observations of the combined transient effects that do exist. For this research, an alternative but indirect approach was chosen in which the individual phenomena are verified with simple geometries for which there are known and well established solutions. This approach resulted in a number of test cases which are summarized in Table 5.1. These cases are presented individually in the remainder of this chapter. Table 5.1 characterizes each case by the geometric space (1-dimensional or 2-dimensional) of the problem, the thermodynamic state of the material, and the principle phenomena of interest. Note that even though the particular case can be characterized as 1-dimensional, the 2-dimensional analysis was used.

The last case gives the solution to the general problem combining all the phenomena of interest. This case demonstrates the ability of this analysis approach to solving a realistic problem representing several interacting fluid/thermal phenomena. It could also serve as a test case for comparison of similar analyses which other researchers may be developing.

A discussion of the computer resource usage is given at the end of the chapter.

Table 5.1 Summary of Cases

Case	Space	Material State	Phenomena
1	1-D	Solid	Conduction, with Prescribed Temperatures
2	2-D	Solid-Liquid	Phase Change, by Conduction only with Prescribed Temperatures
3	2-D	Liquid	Buoyancy-Driven Convection with Prescribed Temperatures
4	2-D	Solid-Liquid	Phase Change, by Conduction and Buoyancy Driven Convection

Case 1: Conduction Only

As was discussed in Chapter 3, the formulation of the energy equation is based on internal energy instead of the more common temperature and specific heat approach. To ensure the validity of such an approach, Case 1 considers a simple transient problem for conduction heat transfer. For this case, material properties are constant throughout the solution domain and the material always remains in the solid state. Figure 5.1 describes the solution domain as well as the boundary and initial conditions. The problem consists of a slab of material initially at a uniform temperature, T_o. At time zero the surface temperature of one side is suddenly changed to T_1. Note that since the upper and lower sides are insulated, this problem is actually one dimensional.

Exact solutions to the problem of Case 1 are widely available, for example Kreith[35]. The results of such solutions are often presented in terms of a nondimensional temperature versus the Fourier number defined by

$$F_o = \frac{at}{(2D)^2}$$

where a is the thermal diffusivity defined by

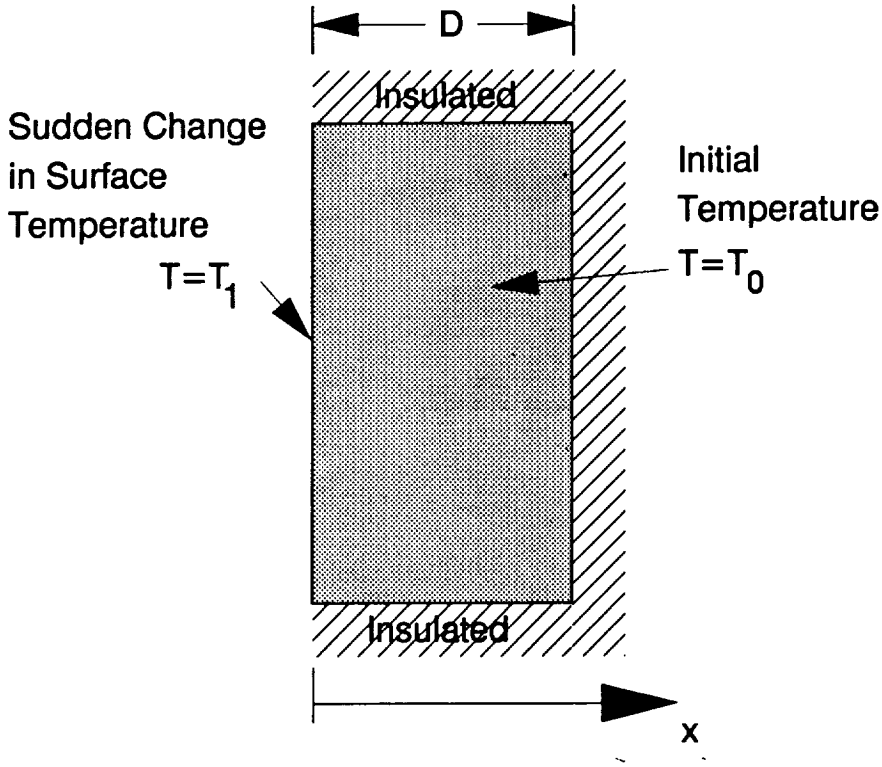

Figure 5.1 Description of Case 1

$$a = \frac{k}{\rho C}$$

and D is the thickness of the material.

Though this problem is really one-dimensional, for convenience, the solution domain was discretized into a total of 25 equal elements, with 5 element divisions along each of the x and y directions. The time step used corresponds to a Fourier number of 0.0005. Figure 5.2

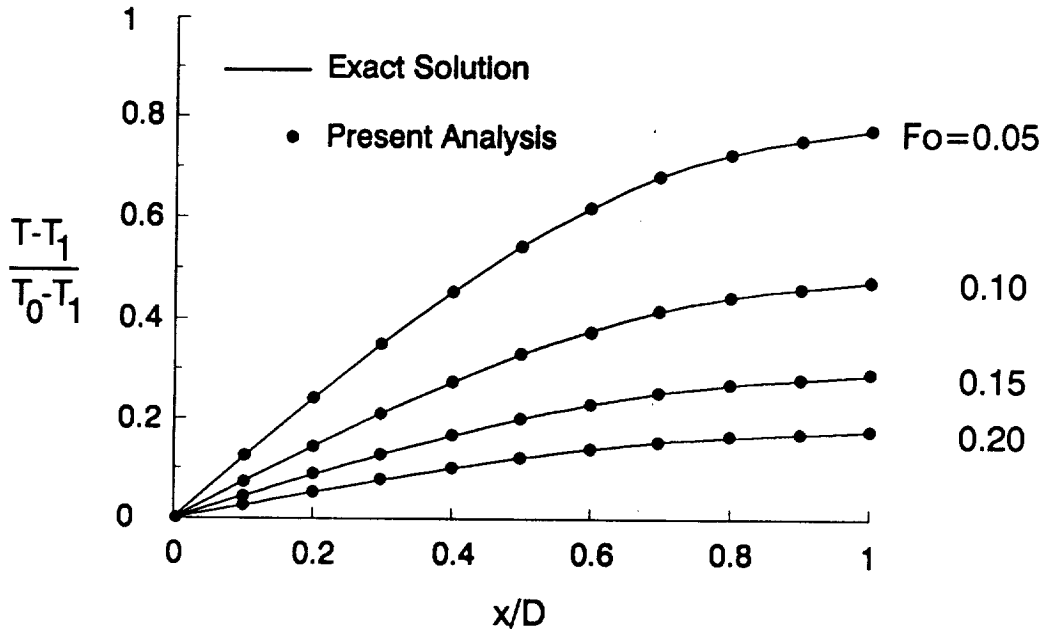

Figure 5.2 Calculated Temperatures for Case 1

shows the calculated temperature distributions at four times. These values are within one percent of the exact values for this problem. During the course of this research, many other similar problems involving conduction heat transfer were solved. These included cases with both prescribed temperatures and/or prescribed heat flux. Results from these cases as well as those of other investigators (see for example White[62]) has confirmed the validity of the internal energy formulation of the energy equation in calculating transient conduction heat

transfer.

Case 2: Phase Change by Conduction

Several problems were studied involving phase change by conduction. For 1-dimensional space, exact solutions exist for prescribed temperature boundary conditions as well as cases with precribed heat flux conditions. Results from the present formulation for such cases showed excellent agreement with exact solutions. A more complex case is that of multi-dimensional phase change. Case 2 models 2-dimensional phase change by conduction with prescribed temperature boundary conditions. Specifically the problem consists of a prism of square cross-section which is initially in the liquid state at the fusion temperature. Prescribed temperatures which are lower than the fusion temperature are applied to the surface of the prism, and it solidifies with time. Due to the symmetry of the problem, only one quarter of the cross-section need be considered with two boundaries maintained at the prescribed temperature as shown in Figure 5.3. For convenience the problem can be described by

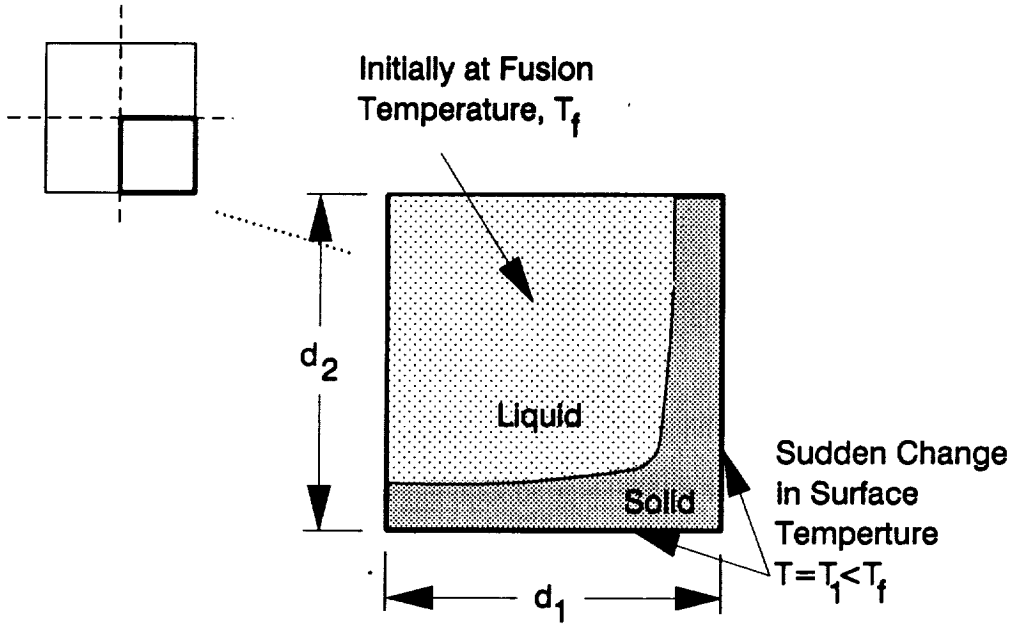

Figure 5.3 Description of Case 2

$$\lambda^* = \frac{(L/C_s)}{T_f - T_1}$$ dimensionless latent heat parameter

$$\theta_f = \frac{(T_f - T_1)}{(T_0 - T_1)}$$ dimensionless fusion temperature

$$\tau = \frac{a_s t}{D^2}$$ dimensionless time

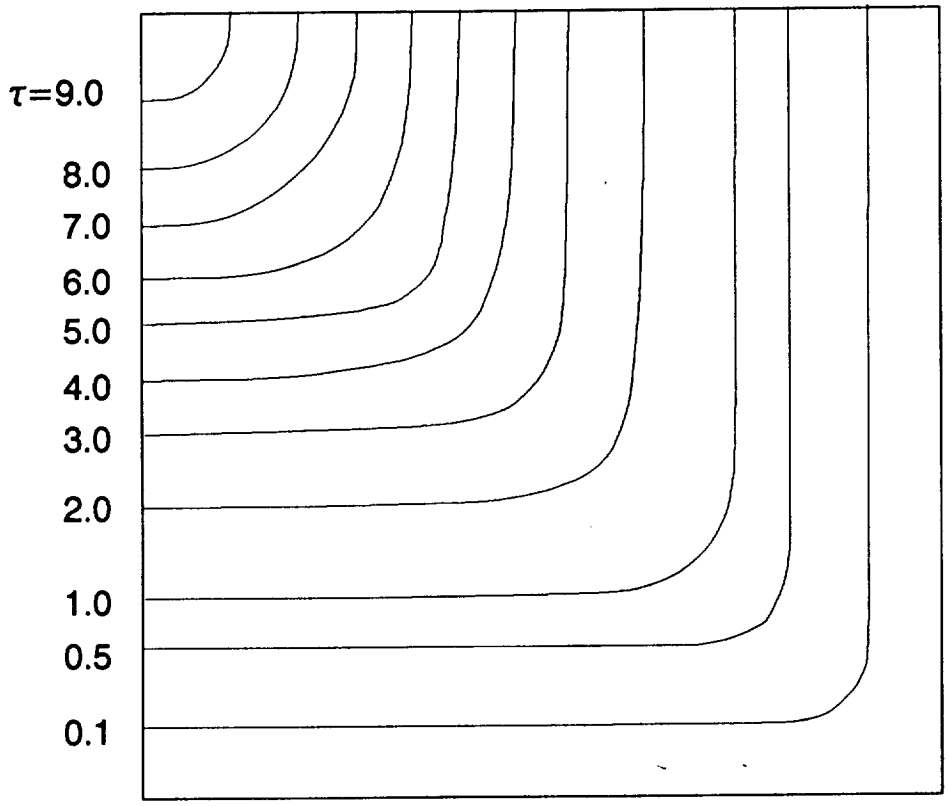

Figure 5.4 Calculated Loci of Interface for Case 2

$\zeta_j = \frac{l_j}{D}$ dimensionless interface location

in which D is a convenient reference length and the subscript s denotes properties of the solid.

This problem has been studied by several researchers in the past including Poots[50], Lazaridis[37] and Crowley[14]. The numerical data used here is the same as was used by

those researchers. These values are $\lambda^*=1.5613$, $\theta_f=1$, and $d_1=d_2=4$, where d_1 and d_2 are the normalized dimensions of the quarter square section.

Because the density is constant throughout the solution domain, no effects of flow in the liquid were considered and only the energy equation was solved. For this problem, a total of 49 equal square elements were used with a time step corresponding to a τ of 0.005. Figure 5.4 shows the calculated interface at various times during the solidification. The interface locations are also presented in terms of fraction of solidified matter along the diagonal and at the insulated boundaries in Figures 5.5 and 5.6, respectively. The calculated results compare quite well with the results from other researchers. Though no exact solution to this problem exists, a similarity solution for an infinite medium is also given. The calculated results of the present analysis for the finite medium compare favorably with the infinite medium analysis initially. At later times, the infinite medium solution predicts a slightly faster solidification. This is expected since end effects in the infinite medium allow for heat conduction out of the corner, while ends for the finite medium are effectively insulated.

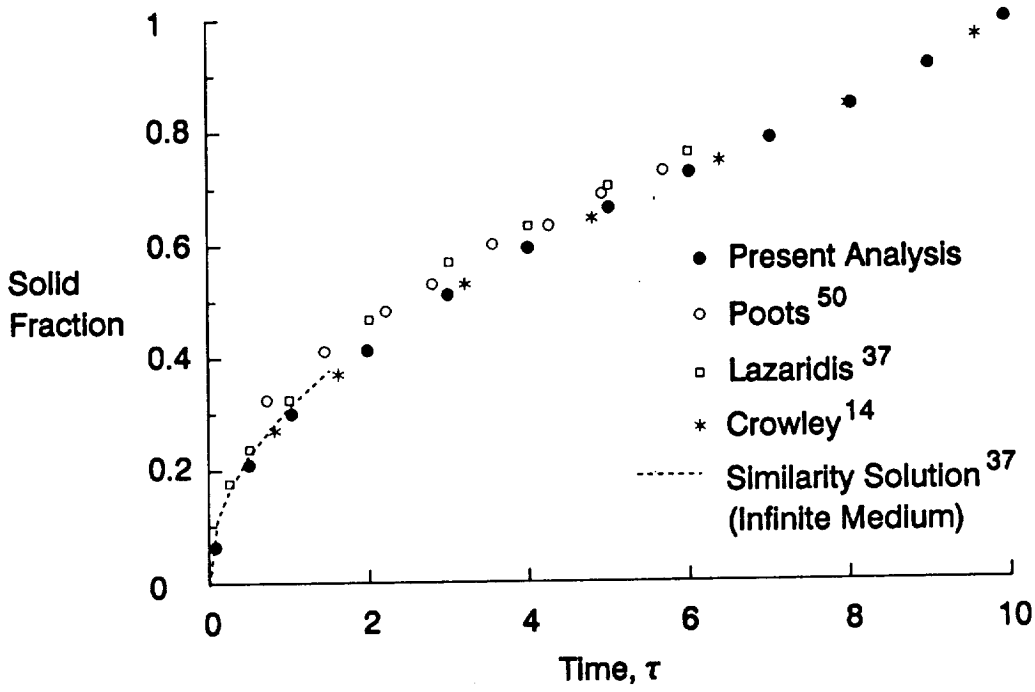

Figure 5.5 Calculated Interface Location Along the Diagonal for Case 2

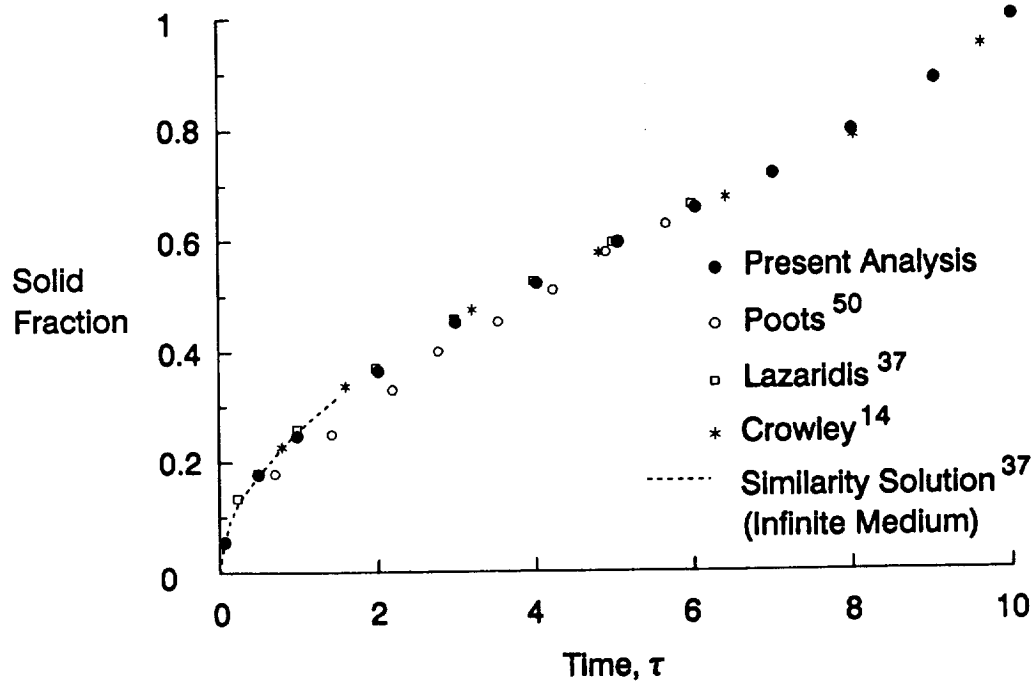

Figure 5.6 Calculated Interface Location Along the Center Line for Case 2

Case 3: Buoyancy-Driven Convection

Flow inside a square cavity is one of the simplest problems in convection and is often used to test the validity of fluid analysis methods. Case 3 is an example of buoyancy driven flow in a square cavity. Figure 5.7 describes the solution domain and boundary conditions. For this case, one side of the square cavity is maintained at a constant temperature of T_1. The opposite side is at a higher temperature T_2. The velocities of the fluid are prescribed to zero at the container wall. The Prandtl number of the fluid was chosen to be 1.

The dimensionless Rayleigh number, defined by

$$Ra = \frac{g\beta(T_2 - T_1)D^3}{\nu a}$$

is used to characterize the flow. The coefficient of thermal expansion, β, in the definition of the Rayleigh number above is defined by

$$\beta = \left(1 - \frac{\rho}{\rho_0}\right)\frac{1}{\Delta T}$$

where the 0 subscript denotes some reference state, and ΔT is the change in temperature from that reference state.

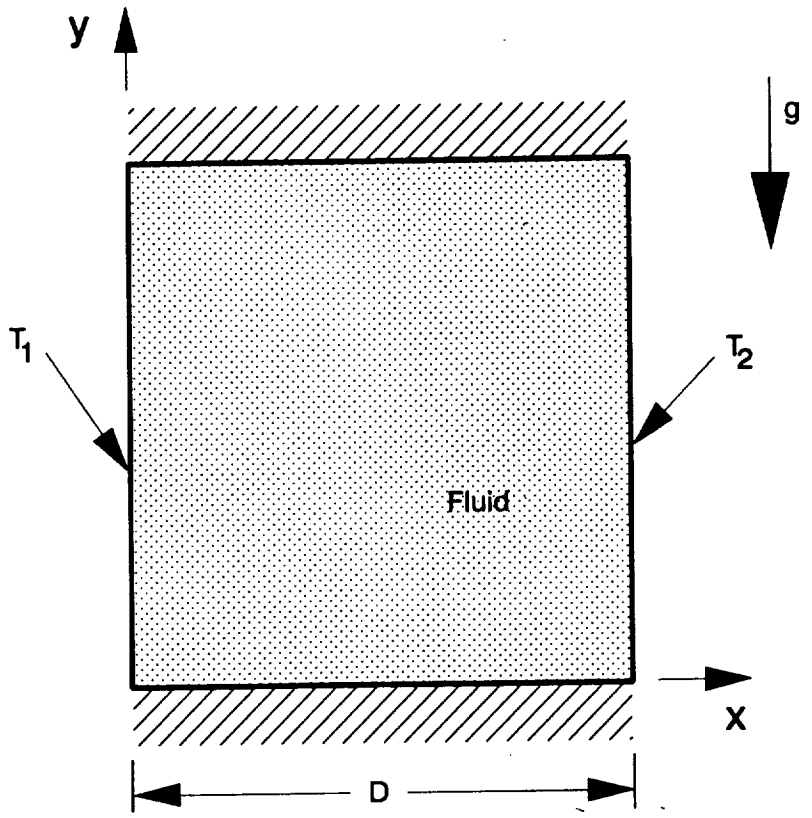

Figure 5.7 Description of Case 3

For this problem, a total of 49 equal sized square elements were used. Figures 5.8, 5.9, and 5.10 show the calculated fluid velocity fields and temperatures at Rayleigh numbers of 10^3, 10^4, and 10^5. No cases were calculated for higher Rayleigh numbers, however, no stability problems were observed at a Rayleigh number of 10^5. The calculated results compare well qualitatively with similar analyses by Pepper and Cooper[49]. Pepper and Cooper also compiled data from the literature for the buoyancy driven cavity flow problem described above. This

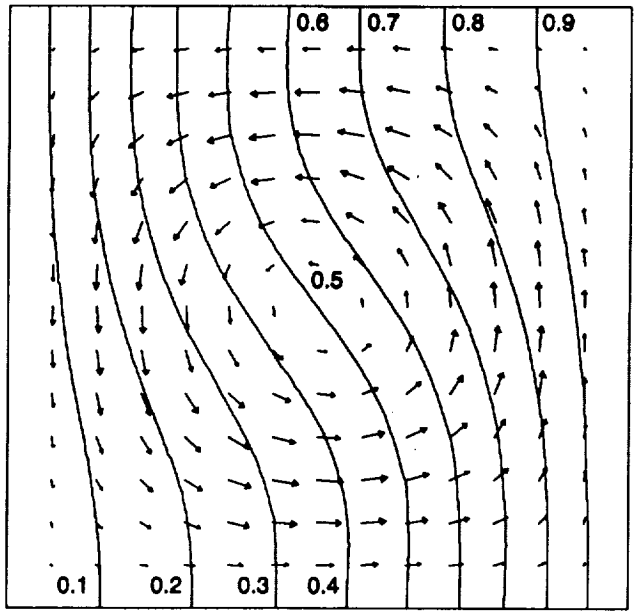

Figure 5.8 Calculated Fluid Velocity Vectors and Normalized Temperature Contours for Case 3 at a Rayleigh Number of 10^3

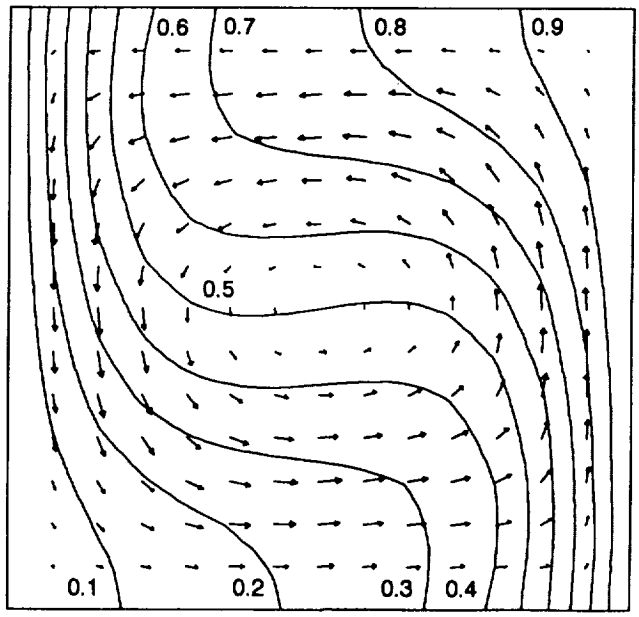

Figure 5.9 Calculated Fluid Velocity Vectors and Normalized Temperture Contours for Case 3 at a Rayleigh Number of 10^4

data is presented in terms of an average Nusselt number given by

$$Nu \equiv \int_0^D \left.\frac{\partial T}{\partial x}\right|_{x=0} dy$$

The Nusselt number above is dimensionless by normalizing with respect to the wall temperatures and letting the dimension D be 1. Results from the present analysis are given in Table 5.2 and graphically in Figure 5.11 along with the results from other researchers.

Table 5.2 Calculated Nusselt Numbers for Case 3

Ra	Nu
10^3	1.11
10^4	2.40
10^5	5.17

Figure 5.11 shows excellent agreement between the present analysis and previously published data.

In addition to this problem, several other cases involving free and forced convection were studied during this research. One case considered driven cavity flow (forced convection) without buoyancy. In this case three

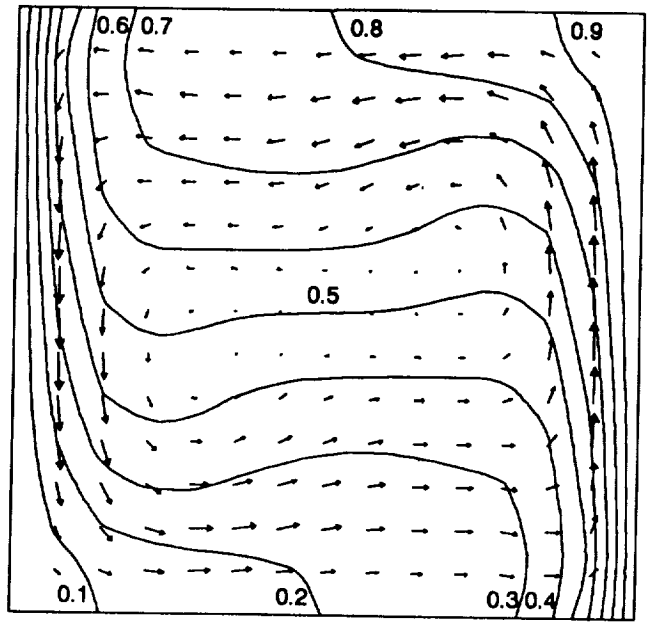

Figure 5.10 Calculated Fluid Velocity Vectors and Normalized Temperature Contours for Case 3 at a Rayleigh Number of 10^5

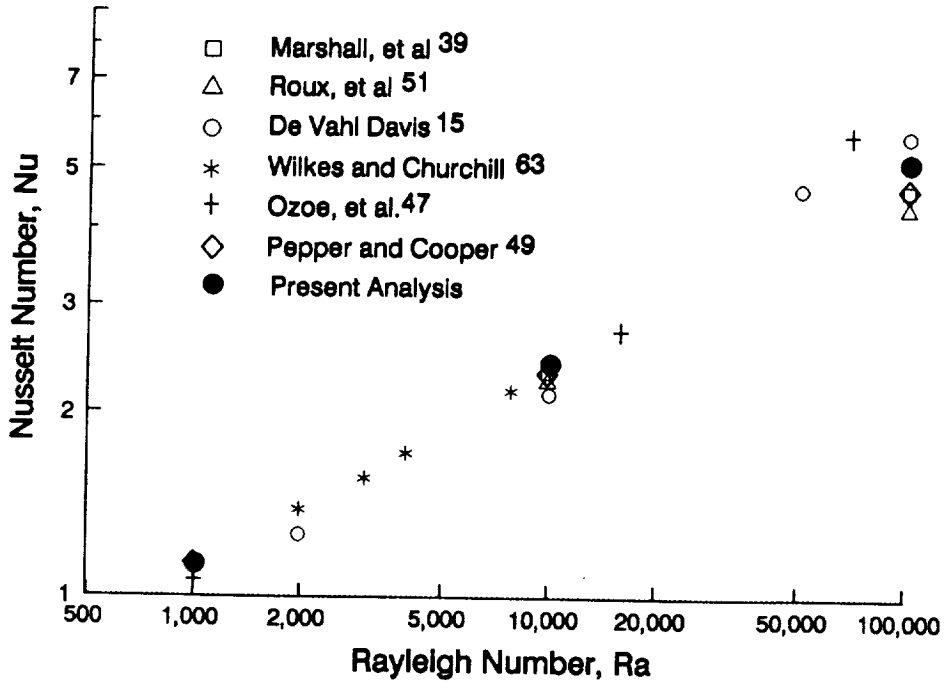

Figure 5.11 Average Nusselt Number Versus Rayleigh Number for Case 3

sides of the square cavity are maintained at a constant temperature of T_1. The fourth side is at a temperature T_2, and moves with a velocity V. The predicted velocity and temperature contours compared well qualitatively with similar analyses by Chen, et al.[13].

Case 4: Combined Conduction, Convection, and Phase Change

The last case gives the solution to the general problem combining conduction, convection, and phase change. Figure 5.12 shows the solution domain along with the boundary and initial conditions. The problem consists of a material inside a container of square cross-section. The top and bottom of the container are maintained at constant temperatures above and below the fusion temperature of the material. Since the container is initially at rest and not subject to a gravitational field, there is no motion in the liquid phase. The initial temperature distribution varies linearly between T_1 and T_2 and flat phase front has formed perpendicular to the direction of the temperature gradient. At time zero, the material is subjected to an acceleration causing buoyancy-driven convection in the fluid. This convective flow changes the heat transfer and thus affects the location of the phase front.

Table 5.3 gives the numerical values for the boundary

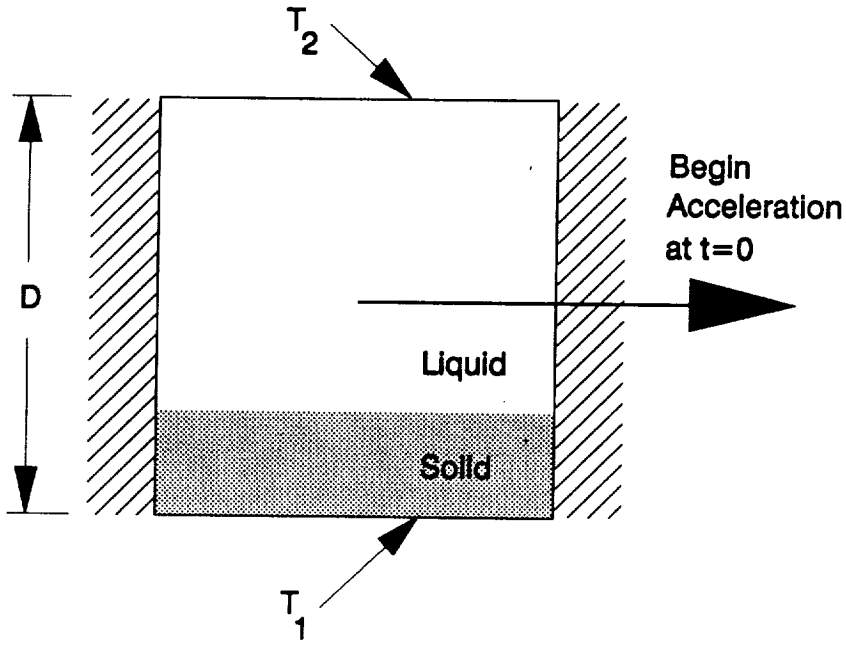

Figure 5.12 Description of Case 4

conditions, material properties, and geometry of Case 4.

The complexity of this problem is apparent from the number of physical constants listed in Table 5.3. Even if dimensional analysis were applied to this problem, over ten dimensionless groups would result. That many dimensionless numbers would not significantly improve the description of the problem, thus only the conventional dimensionless parameters are given here.

Table 5.3 Physical Properties and Conditions for Case 4

Symbol	Description	Value
T_f	Fusion Temperature	0.0°C
T_1	Prescribed Low Temperature	-0.5°C
T_2	Prescribed High Temperature	1.5°C
ρ_s	Solid density	1000 kg/m^3
ρ_l	Liquid density @ T_f	1000 kg/m^3
β	Liquid thermal expansivity	0.001 /°C
μ	Dynamic viscosity	0.001 kg/
C_s	Solid specific heat	1000 J/kg°C
C_l	Liquid specific heat	1000 J/kg°C
L	Latent heat of fusion	1000 J/kg
k_s	Solid thermal conductivity	1 W/m°C
k_l	Liquid thermal conductivity	1 W/m°C
g	Acceleration	0.02 m/s^2
D	Length of container side	0.1 m
t	time	s

The Rayleigh number, characterizing the buoyancy driven convection, and given by

$$Ra = \frac{g\beta(T_2 - T_f)D^3}{\nu a} = 3\times10^4$$

The Prandtl number, characterizing the fluid's ratio of momentum and thermal diffusivity, and given by

$$Pr = \frac{\nu}{a} = 1.0$$

The Jakob number, characterizing the material's ratio

of specific heat to latent heat capacity, and given by

$$Ja = \frac{C_s (T_f - T_1)}{L} = 0.5$$

In addition, we may define dimensionless temperatures given by

$$\theta_1 = \frac{T_1 - T_f}{T_2 - T_1} = -0.25$$

and

$$\theta_2 = \frac{T_2 - T_f}{T_2 - T_1} = 0.75$$

Finally, we may define a dimensionless time parameter given by

$$\tau = \frac{a_s t}{D^2}$$

For this problem a total of 49 equal square elements were used and the time step corresponded to a τ of 0.0002. Figures 5.13 through 5.23 show the calculated results for

Case 4 at various times during the phase change process. These results include normalized isotherms, velocity vectors, and the location of the liquid/solid phase front. The development of the fluid flow and its effect on the heat transfer can be seen at the early time steps. The interesting aspect of this example is the definite influence of the fluid flow on the heat transfer and the resulting movement of the phase front. Steady state is reached at τ =0.2 with a phase front significantly different from the flat front formed initially under conditions of no convective flow.

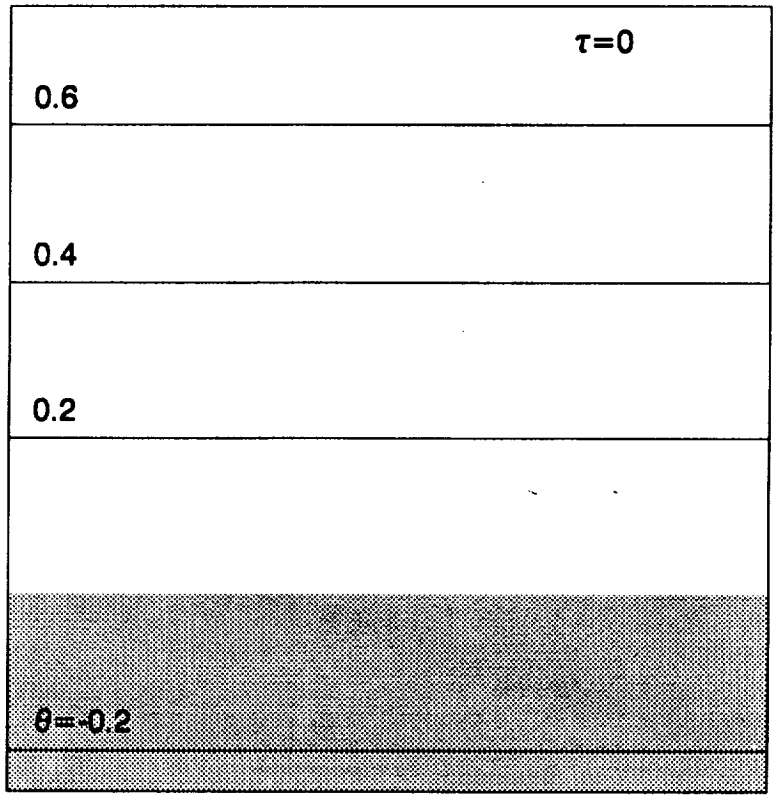

Figure 5.13 Normalized Isotherms, Velocity Vectors, and Phase Distribution for Case 4 at τ=0.0

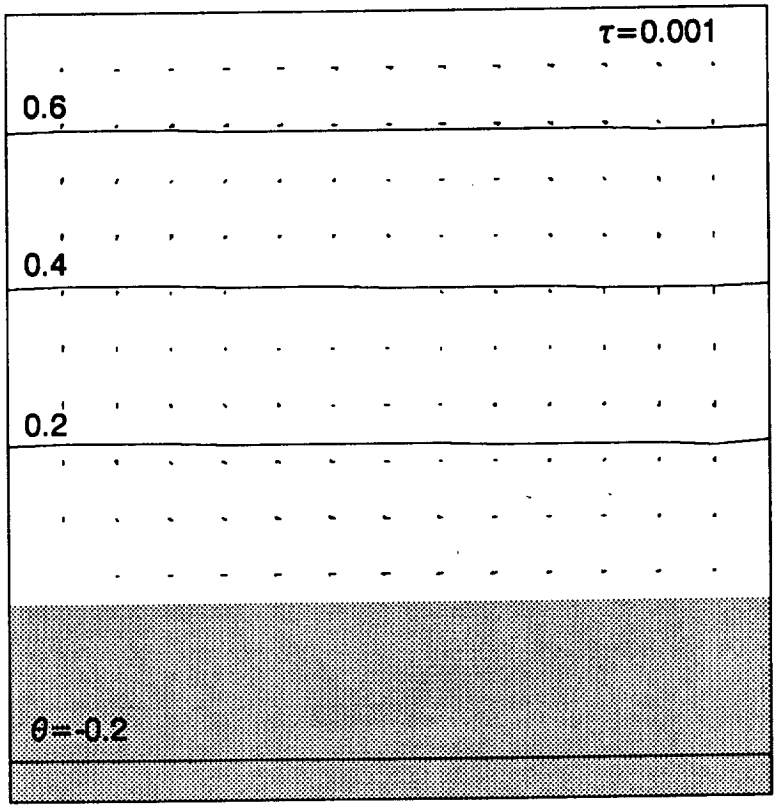

Figure 5.14 Normalized Isotherms, Velocity Vectors, and Phase Distribution for Case 4 at τ=0.001

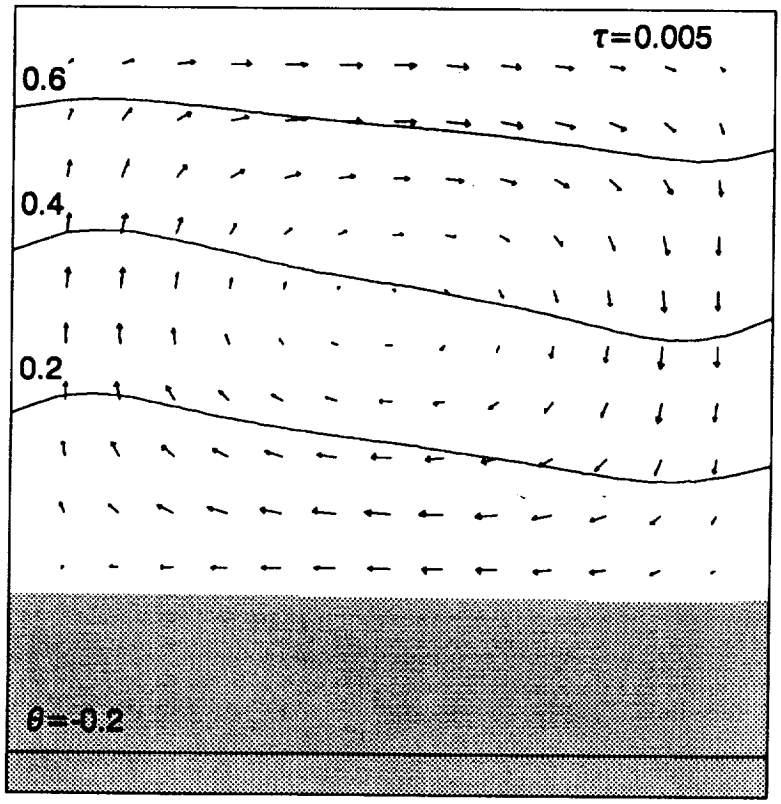

Figure 5.15 Normalized Isotherms, Velocity Vectors, and Phase Distribution for Case 4 at τ=0.005

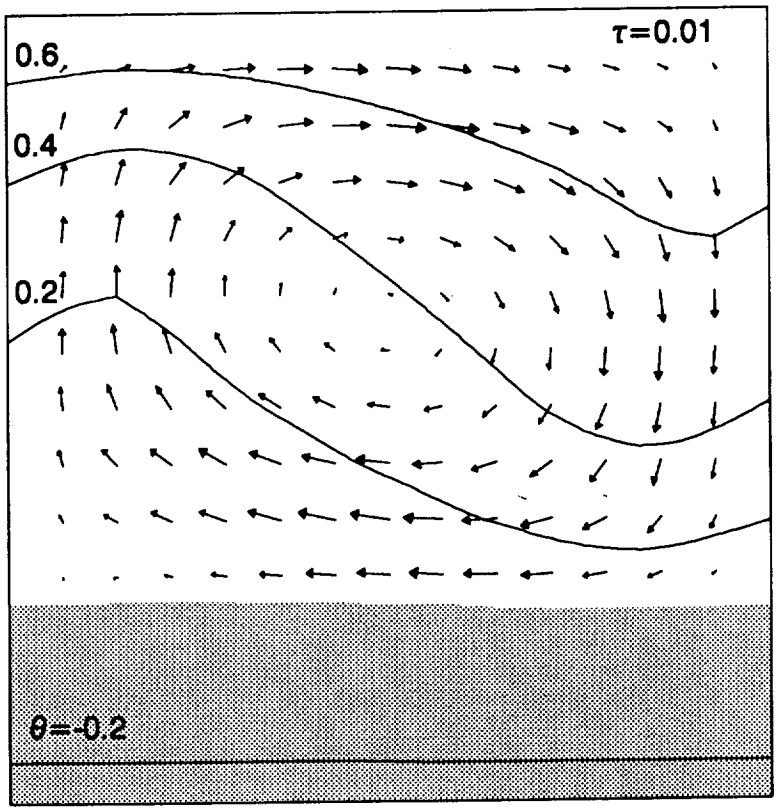

Figure 5.16 Normalized Isotherms, Velocity Vectors, and Phase Distribution for Case 4 at τ=0.01

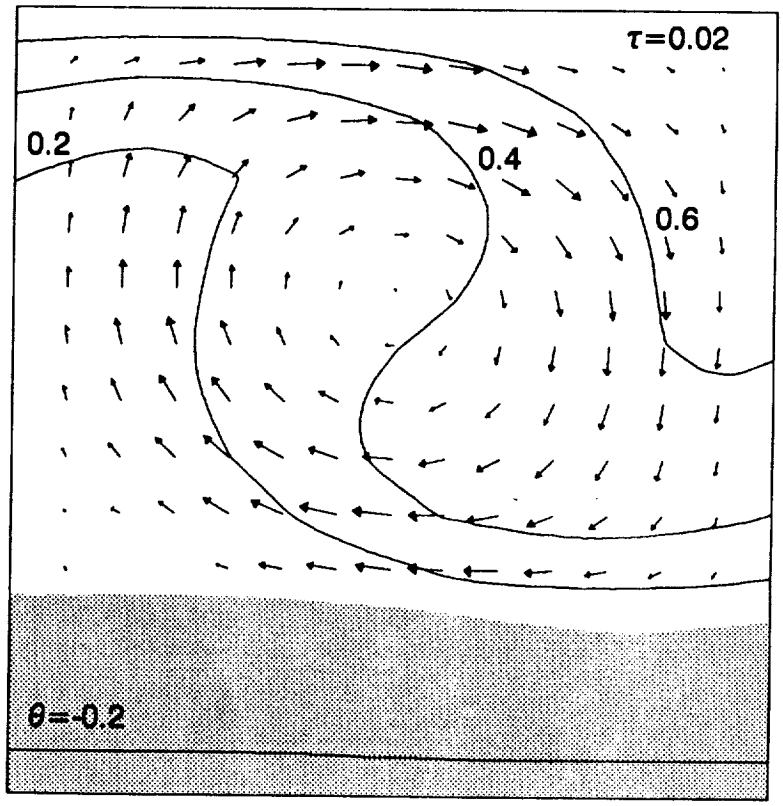

Figure 5.17 Normalized Isotherms, Velocity Vectors, and Phase Distribution for Case 4 at τ=0.02

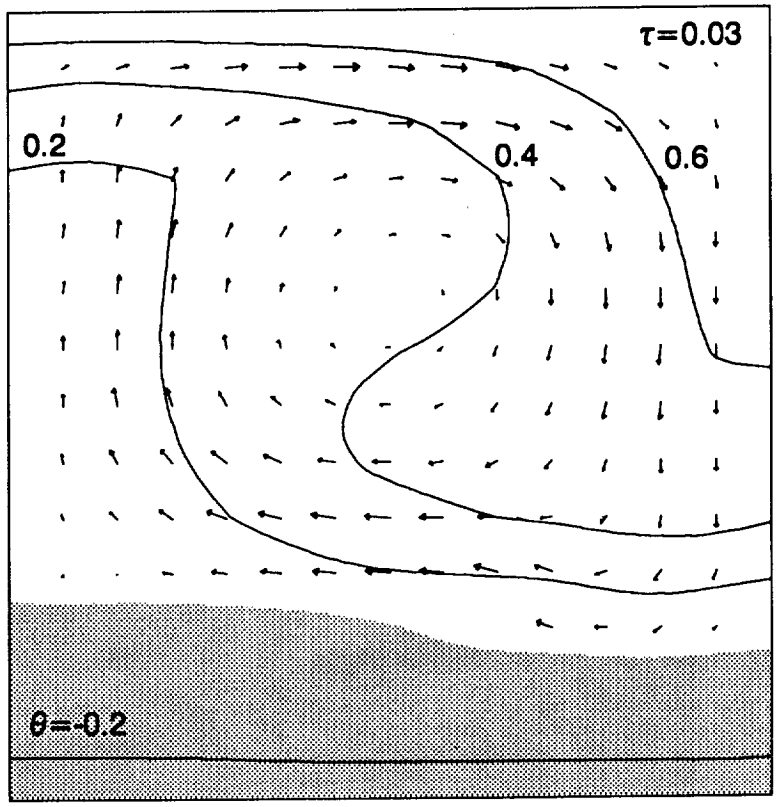

Figure 5.18 Normalized Isotherms, Velocity Vectors, and Phase Distribution for Case 4 at τ=0.03

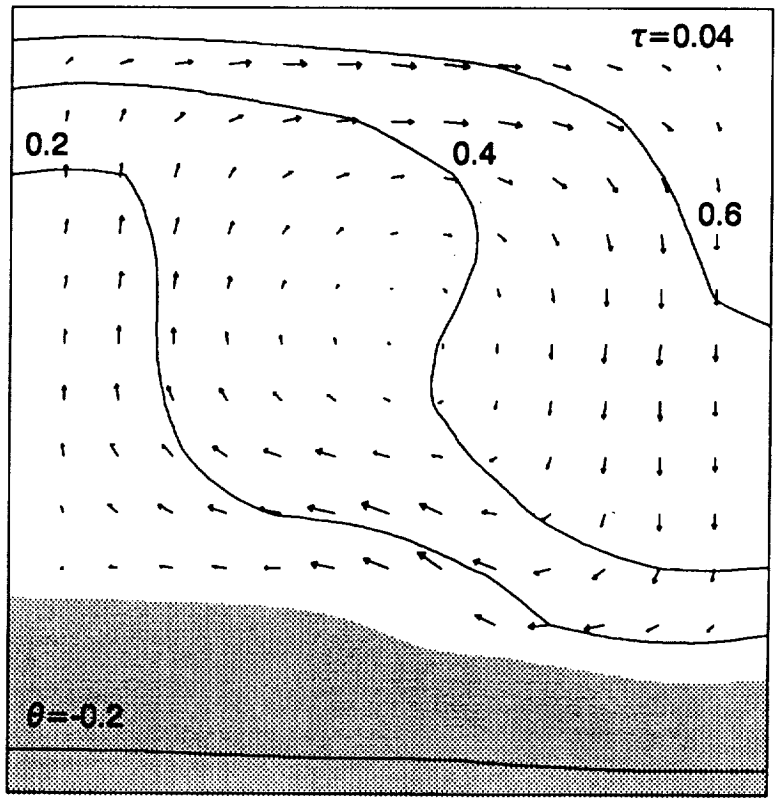

Figure 5.19 Normalized Isotherms, Velocity Vectors, and Phase Distribution for Case 4 at τ=0.04

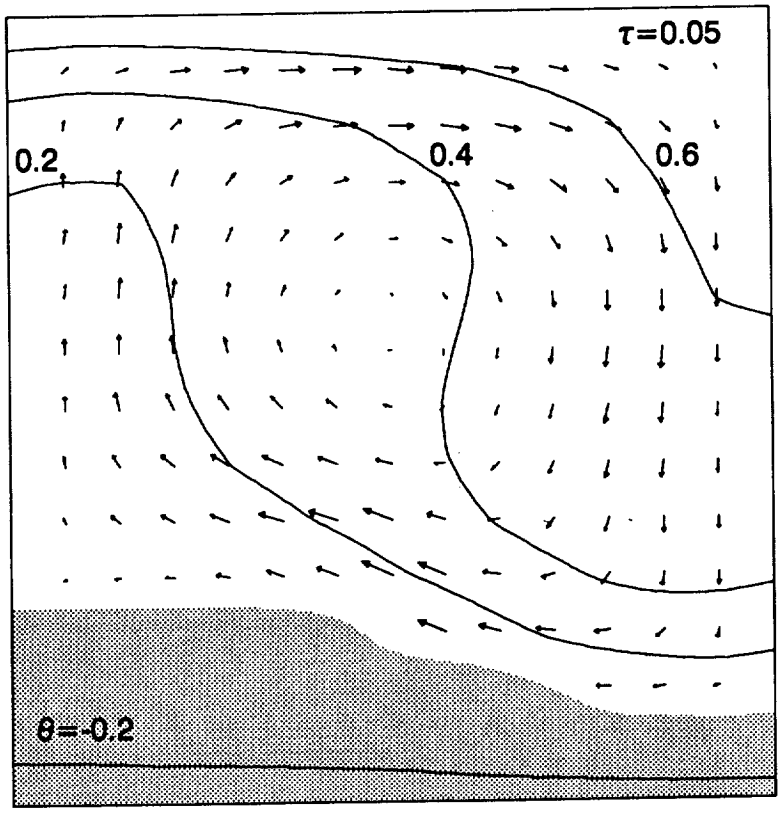

Figure 5.20 Normalized Isotherms, Velocity Vectors, and Phase Distribution for Case 4 at τ=0.05

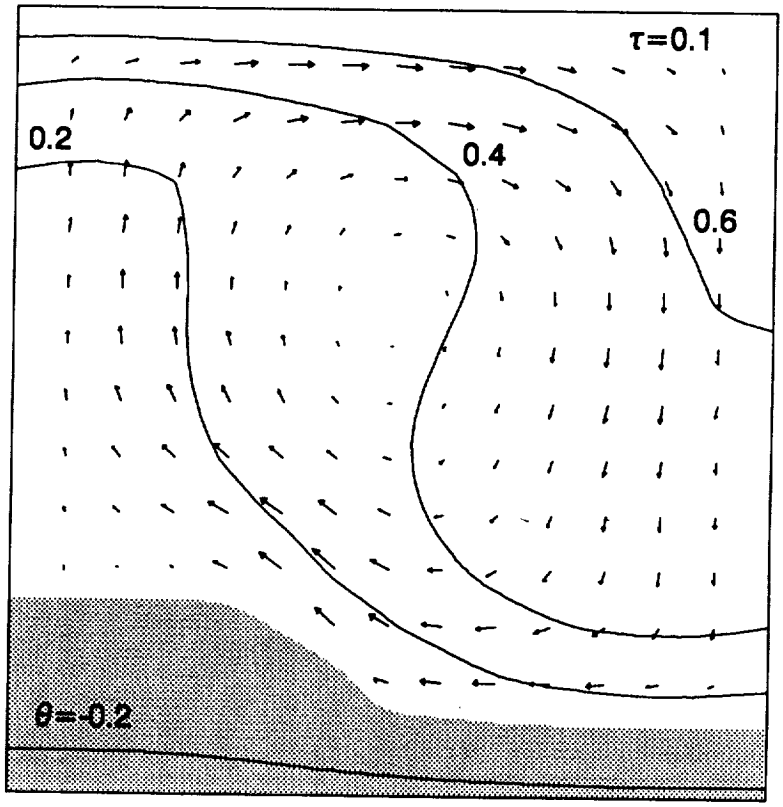

Figure 5.21 Normalized Isotherms, Velocity Vectors, and Phase Distribution for Case 4 at τ=0.1

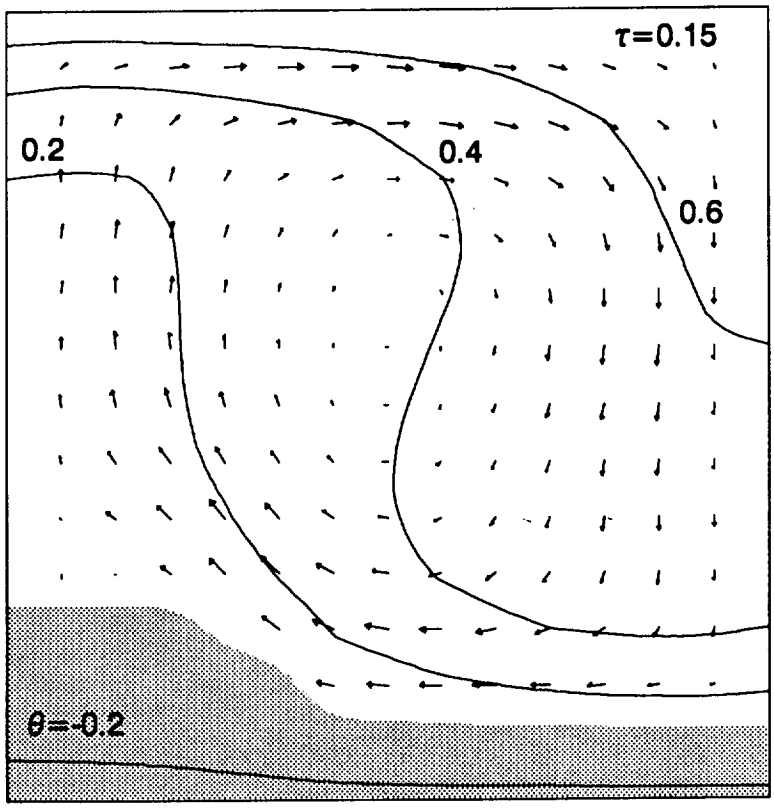

Figure 5.22 Normalized Isotherms, Velocity Vectors, and Phase Distribution for Case 4 at τ=0.15

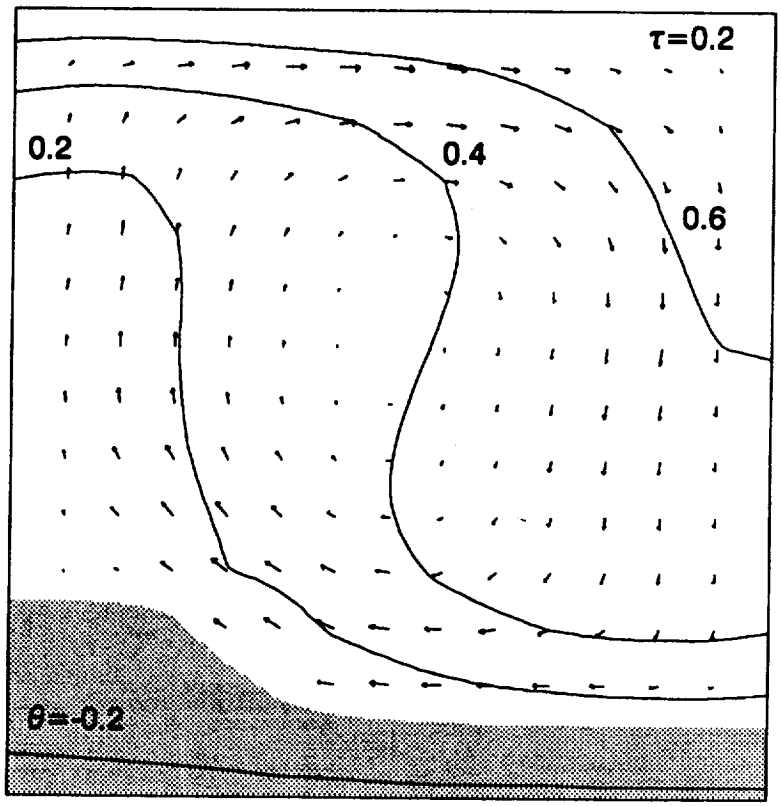

Figure 5.23 Normalized Isotherms, Velocity Vectors, and Phase Distribution for Case 4 at τ=0.2

Computer Resource Usage

Computer usage is of importance to most numerical modelers. Table 5.4 shows the computer time usage for each of the cases presented. The computer system used was an AMDAHL 5870 with an MVSXA operating system. It is important to note, that these cases were not fully optimized in terms of domain discretization or time step to provide minimum CPU times.

Table 5.4 CPU Times of Verification Cases

Case	CPU Time, seconds	CPU Time / # Time Steps
1	696	1.74
2	16470	8.24
3	15035	150.4
4	145700	145.7

The CPU times given in Table 5.4 are long and could result in significant expense on a pay for time computer system. The intended use of this analytical model, however, is in the aerospace industry and government with institutional computational facilities devoted to such tasks. Where no other similar analysis tool is available,

the comparison may be between these computer costs and the need for a space flight experiment, for example, the cost of which can also be substantial.

It should also be noted that these CPU times are not unusual for modeling phase change problems with convection. Schneider[52] cites researchers quoting CPU times of 50000 seconds on a CDC 6500 computer, for similar phase change problems. Further discussion on computer usage and possible areas of improvement can be found in the next chapter.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

The work presented develops a solution approach to the combined conservation laws of energy, momentum, and mass. This approach is quite general and provides a method to analyze a variety of fluid and thermal problems including those with phase change.

The finite element method, being an integral method provides a natural means to implementing the internal energy formulation of the phase change problem. Using elements with quadratic interpolation functions, the interface can be tracked quite accurately. Finite difference formulations to date only provide information on the interface location to within on mesh spacing. Results from those analyses show jagged interfaces (see Schneider[52]) that can influence convective flow in the liquid.

The analytic approach was implemented in a computer program and results were verified by investigating individual phenomenon and comparing with known solutions. In general, the approach yields solutions with good engineering accuracy. The predicted results for a problem

similar to that described in Chapter 3 were presented for comparison by other researchers who may develop similar analytical methods in the future.

One area of concern for the present approach is the high computer resource usage. The majority of the computer time is incurred in forming and then inverting the large set of system equations needed to converge the nonlinearities and achieve a valid solution.

One way of reducing the computer time requirements by improving the algorithm may be to incorporate a Newton-Raphson iteration scheme to converge for the nonlinearities. For a further discussion on the application of of such methods, see Geradin, et al.[25].

Another approach to reducing computational times would be to improve the simultaneous equation solver. The matrices representing the system equations are characteristically sparse with nonzero coefficients located close to the diagonal. The use of a banded matrix solver could significantly reduce computer time.

Fortunately, future trends will continue to reduce the computational times and costs. The operation of the computer program on a 3090 class computer with vector optimizing hardware should reduce CPU times by about an order of magnitude. This trend of increased computational capability of the hardware will continue in the future. Also the development of efficient large matrix solvers is undergoing much research and is a very important aspect of

numerical modeling of fluid/thermal systems.

There are several obvious extensions to this work. First, this approach can easily be modified to analyze 3-dimensional space systems. Nothing in the formulation should prevent this and only the available computational capabilities might present a restriction on the geometric complexity of the problem.

Further investigation is also warranted in the use of consistent mass matrices for the inertia terms in the energy and momentum equations. Though the lumped mass matrices are by far the most common, there is concern that the method of lumping might contribute to inaccuracies particularly for highly distorted curve-sided elements.

Though I personally believe that the internal energy or enthalpy method is the only practical method for modeling the phase change problem in 3-dimensional space, development of other methods is warranted. The work of Chang and Brown[12] is particularly interesting, although their moving boundary-moving mesh techniques would be very difficult to implement where the thermodynamic phases are fragmented.

For application to the space environment, other forces not addressed here can become important. Siegel[55] provides a good overview on the effects of reduced gravity on heat transfer. The incorporation of a surface free energy model would be especially useful, particularly for very low gravity conditions. Pearson[48], Labus[36], and

Merte[40] provide interesting approaches to handling the free surface problem, however, application to the finite element method remains fertile ground for research.

Finally, the incorporation of a turbulence model into the analysis could significantly extend its application.

APPENDICES

APPENDIX A

COMPUTER PROGRAMMING APPROACH AND DOCUMENTATION

This appendix discusses the general programming approach and implementation used in this research. Included is a description of the program functions and important global variables. This is followed by a flowchart of the major functions.

The programming language APL was chosen for its reduced programming development time, its inherent matrix manipulation capabilities, and because it is easily transported to various computer hardware systems. APL was originally developed by Iverson[33] as a general mathematical notation and later implemented as a computer programming language. A detailed description of the programming language APL is given by Gilman and Rose[26]. APL has several significant advantages over more "conventional" languages such as FORTRAN, PASCAL, etc. Because APL is a symbolic vector language the source code is typically at least two to three times shorter than most other languages and can be developed about four to ten times faster. Table A.1 shows a comparison between FORTRAN and APL programs to produce the sum of all the numbers greater than 50 in a set of real numbers. The APL program is significantly shorter. Several characteristics contribute to APL's brevity. APL processes aggregates of

data, or arrays, with much the same syntax as single numbers, and therefore essentially eliminates the need for structures such as loops. Data is maintained unformatted within the APL environment, so that no logical ties to the host operating system files is required. In addition, all numbers are stored and operations performed using double precision.

Table A.1 FORTRAN and APL Programs to Sum the Real Numbers In a Set That are Greater Than 50

FORTRAN	APL

`    DOUBLE PRECISION SUM,X`
`    READ(5,100) LIMIT`
`100 FORMAT(I10)`
`    SUM=0.0D0`
`200 DO 400 I=1,LIMIT`
`    READ(5,300) X`
`300 FORMAT(F10.5)`
`    IF(X.LE.50.) GO TO 200`
`400 SUM=SUM+X`
`    WRITE(6,500) SUM`
`500 FORMAT(F20.5)`
`    STOP`
`    END`
	`+/(X>50)/X`

Most APL systems today use an interpreter though some compilers are available. The interpreted versions lend themselves very well to program development because the environment is interactive, no compiling is required, and the debugging facilities work directly with the source

code. Compiled versions however can run significantly faster for some problems (particularly those that are highly iterative). For the problem presented in this research, however, the computer time usage is dominated by inversion of the large system matrix. This inversion is performed by a highly efficient APL primitive function and would not benefit significantly from using a compiled external function. This APL matrix inversion function will also take advantage of super computer class vector processing hardware, when available, to further improve computational performance.

The following pages contain descriptions of the program functions and important global variables. This is followed by the calling structure (flow chart) of the major functions.

FUNCTIONS

ADJSTEP
ADJUSTS TIME STEP BASED ON ABILITY TO CONVERGE AND FIELD VALUES
AGAIN
INTERATIVE SOLUTION (FOR NON-LINEARITIES) AT EACH TIME STEP
AINTCON
INTEGRATION CONSTANTS FOR USE WITH AINTGRT
AREA
CALCULATES AREAS AND VOLUMES OF ELEMENTS
BDY
EVALUATES R VECTOR FROM BOUNDARY AND INTERNAL CONDITIONS
BDY2
MODIFIES ELEMENT R MATRIX FOR BOUNDARY COND. TYPE 2 (PRESCRIBED FLUX)
BDY2ΔSUB
SUB-FUNCTION OF BDY2 TO PRESCRIBE EACH (ELEMENT, SIDE COMBINATION)
CHKCNV
CHECKS CONVERGENCE OF EF, UF, VF AND PF WITH LAST ITERATION VALUES
CHKCNVΔS1
CHECKS FIELD VALUES FOR CONVERGENCE
CHKEF
CHECKS ENERGY FIELD VALUES TO SEE IF WITHIN PROPERTY DATA RANGE
CHKINPRES
CHECK PRESRCRIBED PRESSURE NODE "PRNODE" SPECIFIED IN INPUT
CHKINPUT
CHECK INPUT PARAMETERS
CHKSS
CHECK IF STEADY STATE HAS BEEN REACHED
CHKTF
CHECKS TEMPERATURE FIELD VALUES IF WITHIN PRESCRIBED VALUES
CLEANUP
ERASES VARIABLES NAMED IN GLV EXCEPT THOSE WITH ALT. CHARACTER NAMES
COORXE
GENERATES XI-ETA COORDINATES OF THE NODES
CPUCHK
RETURNS A 0 IF CPU TIME LIMIT IS EXCEEDED
CPUTIME
RETURNS CPU SECONDS USED SINCE "TSTART" WAS ISSUED
DFN
CALCULATES DERIVATIVES OF FIELD VARIABLE AT THE ELEMENT NODES
DIAG
FORMS DIAGONAL MATRIX FROM A VECTOR X
EGYASN
ASSIGN ENERGY FIELD VALUES
EGYBAR
FORMS ENERGY EQUATIONS FOR SOLVING TRANSIENT RESPONSE
EGYEVL
EVALUATES ENERGY MATRICES
EGYMAT
CONSTRUCT THE ENERGY ELEMENT MATRICES
ENERGY
FORMS AND SOLVES THE TRANSIENT ENERGY EQUATION
FLDMATΔCM
FORMS CM MATRIX FOR FLDMAT

```
FLDMAT∆KA
     FORMS KA MATRIX FOR FLDMAT
FLDMAT∆KAE
     FORMS KAE MATRIX FOR FLDMAT
FLDMAT∆KNN
     FORMS K11 K12 K21 K22 MATRICIES FOR FLDMAT
FLDMAT∆LCN
     FORM LC1 AND LC2 MATRICIES FOR FLDMAT
FLDMAT∆LN
     FORMS L1 AND L2 MATRICIES FOR FLDMAT
FLDMAT∆MM
     FORMS MASS MATRIX FOR FLDMAT
FLDMAT∆R
     FORM R VECTORS FOR FLDMAT
FLDMAT∆RC
     FORMS RC VECTOR FOR FLDMAT
FLDMAT∆RP
     FORMS AND ASSIGNS RP MATRICIES FOR FLDMAT
FLDSLV
     SOLVES EQUATIONS AND ASSIGNS FIELD VALUES
FLDSLV∆STAR
     RETURNS FIELDS FROM LAST ITERATION
FLOW
     FORM AND SOLVE THE FLUID FLOW EQUATIONS
FLWASN
     ASSIGN FLUID FLOW FIELD VALUES
FLWBAR
     FORMS FLOW EQUATIONS FOR SOLVING TRANSIENT RESPONSE
FLWEVL
     EVALUATES FLUID FLOW ELEMENT MATRICES
FLWMAT
     CONSTRUCT THE FLUID FLOW MATRICES
FMFEQ∆1
     FORM ENERGY FIELD EQUATIONS
FMFEQ∆2
     FORM FIELD EQUATIONS
FRONTEND
     PERFORMS UPFRONT ONCE ONLY FUNCTIONS
GRAVVEC
     RETURNS GRAVITATIONAL BODY FORCE VECTOR
GRID∆ELN
     GENERATES QUADRATIC NODE NUMBERS (ELEMENT BASIS) FOR GRIDGEN
GRID∆LNODE
     GENERATES LINEAR NODE NUMBERS (ELEMENT BASIS) FOR GRIDGEN
GRID∆RSE
     GENERATES REGION SIDE ELEMENT NUMBERS
GRID∆RSLN
     GENERATES REGION SIDE LINEAR NODE NUMBERS FOR GRIDGEN
GRID∆RSN
     GENERATES REGION SIDE QUADRATIC NODE NUMBERS FOR GRIDGEN
GRIDGEN
     GENERATES ELEMENT NODE NUMBERING
INBDY
     SPECIFY REGION BOUNDARY CONDITIONS
```

```
INGEOM
    SPECIFY GEOMETRY INPUT PARAMETERS
INITIAL
    INITIALIZE VARIABLES
INITMP
    INITIALIZE MATERIAL PROPERTIES: CASE WHERE PRESSURE NODE PRESCRIBED
INITMPROP
    INITIALIZE MATERIAL PROPERTIES USING AVERAGE CONDITIONS
INITPRES
    INITIALIZE PRESSURE FIELD
INITTEMP
    INITIALIZE TEMPERATURE
INITVEL
    INITIALIZE FLUID VELOCITIES
INPLYGN
    FINDS IF POINTS XY ARE IN POLYGONS Q
INPROG
    SPECIFY PROGRAM OPERATION INPUT PARAMETERS
INPUT
    SPECIFY AND PRINT OUT INPUT PARAMETERS
INREG
    SPECIFY INITIAL R AND Z COORDINATES OF REGION
INTQR
    INTERPOLATES BY FINDING CLOSEST REGION AND USING DBL. QUAD. REGRESS.
INTQR∆REG
    CALLED BY INTQR, IT PERFORMS DOUBLE QUADRATIC REGRESSION BY REGION
INTQR∆UNN
    UNNORMALIZE A MATRIX "A" (NESTED ARRAY) WRT. RANGE
JACCHK
    CHECK IF DETERMINANT OF JACOBIAN HAS A SIGN REVERSAL
LSHAPE
    CALCULATES LINEAR SHAPE FUNCTIONS AND THEIR DERIVATIVES
LUMP
    LUMPS THE CAPACITANCE MATRIX BY ROWWISE SUMMATION
LX
    LATENT EXPRESSION FOR PHASTRAN
MAP
    MAP XI-ETA COORDINATES OF NODES INTO X-Y SYSTEM
PHASTRAN
    MAIN CONTROL FUNCTION FOR PHASE CHANGE ANALYSIS MODEL
PRSCRB
    PRESCRIBES FINITE ELEMENT EQUATIONS IN THE GLOBAL VARS. KBAR AND RBAR
PRSIDE
    MODIFIES PFEQ WHICH DEFINES PRESCRIBED BOUNDARY CONDTIONS
PRSIDE∆ADJ
    ADJUSTS ORIGINAL NODE POSITIONS IN A MATRIX FOR FIELD TYPE
PRSIDE∆FLD
    PRESCRIBES BOUNDARY CONDITIONS FOR A FIELD
PRSIDE∆TEMP
    CONVERTS TEMPERATURE BOUNDARY CONDITIONS TO ENERGY BC'S
PRSPRES
    PRESCRIBES PRESSURE NODEM
PRSP1
    PRESCRIBES PRESSURE NODEM: CASE WHERE TOTAL MASS CONSTRAINED
```

```
PRSP2
    PRESCRIBES PRESSURE NODEM: CASE OF OPEN SYSTEM
PRSVLC
    RETURNS LOCATIONS OF PRESCRIBED VALUES FOR THESE FIELD TYPES
QLGSHP
    CALCS. QUADRATIC LAGRANGIAN SHAPE FUNCTIONS AND DERIVATIVES
QUADRGXY
    PERFORMS QUADRATIC REGRESSION ANALYSIS IN 3 DIMENSIONS X Y Z
RCOND
    RETURNS SUB-VECTOR FOR INTERNAL CONDUCTION
RDUPL
    RETURNS A BOOLEAN FOR REDUCING DUPLICATE VALUES LEAVING ONLY THE 1ST
REDUCE
    REDUCES (BY SUMMATION) A VECTOR WITH MULTIPLE INDICIES
REMDUPEL
    REMOVES DUPLICATE ELEMENTS OF X
RESULTS
    DISPLAYS FIELD VARIABLES
RHOZERO
    CALCULATES THE REFERENCE DENSITY
RPRES
    FORMS PRESSURE RESULTANT VECTOR
SAVEFIELDS
    SAVES FIELDS AT EVERY DFN TIME STEP, PUTS IN NESTED ARRAY SAVEFLDS
SETLAST
    SETS VARIABLES FROM LAST TIME ITERATION
SETSOLID
    SETS VELOCITIES TO ZERO FOR NODES IN SOLID STATE
SETSTAR
    SETS EF, UF, VF AND PF AT LAST CONVERGENCE ITERATION
SIFAC
    CALCULATES THE SIDE INTEGRATION FACTOR FOR USE WITH SINTGRT
SINTCON
    INTEGRATION CONSTANTS FOR USE WITH SINTGRT
STATPRES
    CALCULATES THE STATIC PRESSURE DISTRIBUTION
STRIPE
    CREATES STRIPE LINE BORDER
TIMECHK
    RETURNS A 0 IF TIME LIMIT IS EXCEEDED
TIMESTEP
    TIME STEPPING FUNCTION
UPPROP
    UPDATES PROPERTIES ON GLOBAL NODE BASIS
UPPROPΔTEMP
    PLACES PRESCRIBED TEMPERATURES IN UPDATED PROPERTIES
```

VARIABLES

ANI
ORDER OR GAUSS-LEGENDRE QUADRATURE FOR AREA INTEGRATION

AXEI
XI AND ETA COORD. USED IN AREA NUM. INTEGRATION

BC
MODIFIED BOUNDARY CONDITIONS OF REGIONS

CEQ
CAPACITANCE MATRIX FOR FLOW EQUATIONS

CI
CONSTANTS FOR FUNCTION INTGRT

CIL
CONVERGENCE ITERATION LIMIT

CMAT
HEAT CAPACITANCE MATRIX

CPULIM
LIMIT FOR CPU TIME

CSL
CONSTANTS FOR LINEAR SHAPE FUNCTIONS

CSQLG
CONSTANTS FOR QUADRATIC LAGRANGIAN SHAPE FUNCTIONS

DXT
DER. SHAPE FNCS. WRT X FOR ELMNT. TYPE (NESTED ARRAY)

DYT
DER. SHAPE FNCS. WRT Y FOR ELMNT. TYPE (NESTED ARRAY)

EC
ENERGY CALCULATION CONTROL (0-NO THERMAL CALCS.)

EF
INTERNAL ENERGY

EFINIT
INITIAL INTERNAL ENERGY

EFSTAR
INTERNAL ENERGY OF LAST CONVERGENCE ITERATION

ELN
NODE NUMBERS FOR EACH QUADRATIC ELEMENT

EREQ
RESULTANT VECTOR FOR ENERGY EQUATIONS

ERR
ALLOWABLE FIELD CONVERGENCE ERROR

FELT
FIELD ELEMENT TYPES (1-LINEAR, 2-QUAD)

FT
FIELD TYPE (1-INTERNAL ENERGY, ETC)

FC
FLUID CALCULATION CONTROL (0-NO FLOW, 1-FLOW CALCULATED)

FREQ
RESULTANT VECTOR FOR FLOW EQUATIONS

GF
GEOMETRIC FACTOR FOR X-Y OR R-Z COORDINATE SYSTEMS

GRZ
GRAVITATIONAL CONSTANTS IN X AND Y DIRECTIONS

ICTL
ITERATION CONTROL (0-SUCCESSIVE SUB. 1-NEWTON-RAPHSON)

```
KBAR
     MODIFIED STIFFNESS MATRIX FOR TRANSIENT FLOW EQUATIONS
KEQ
     STIFFNESS MATRIX FOR FLOW EQUATIONS
KT
     THERMAL CONDUCTIVITY
LNODE
     NODE NUMBERS FOR EACH LINEAR ELEMENT
LSS
     SHAPE FUNCTIONS FOR SIDE INTEGRATION
ND
     NUMBER OF DIVISIONS PER SIDE PER REGION
NE
     NUMBER OF ELEMENTS PER REGION
NST
     SHAPE FUNCTIONS FOR ELEMENT TYPES (NESTED ARRAY)
PF
     INTERPOLATED PRESSURE FIELD OF FLUID
PFEQ
     MATRIX POS. AND PRSCRB. VALUES OF FLUID
PFINIT
     INITIAL PRESSURE FIELD
PFSTAR
     PRESSURE OF LAST CONVERGENCE ITERATION
EPHI
     ENERGY FIELD VARIABLE SOLUTION
FPHI
     FLOW FIELD VARIABLE SOLUTION
PRNODE
     PRESCRIBED PRESSURE NODE INFORMATION
QAIF
     QUAD. AREA INTEGRATION FACTOR
QSIF
     SIDE INTEGRATION FACTOR
QSS
     SHAPE FUNCTIONS FOR SIDE INTEGRATION
RBAR
     MODIFIED RESULTANT VECTOR FOR TRANSIENT FLOW EQUATIONS
REQ
     RESULTANT VECTOR FOR FLOW EQUATIONS
RHO
     DENSITY OF MATERIAL
RELAST
     LAST TIME ITERATION RESULTANT VECTOR FOR ENERGY
RFLAST
     LAST TIME ITERATION RESULTANT VECTOR FOR FLOW
RN
     REGION NUMBER
RSELMT
     REGION SIDE ELEMENTS
RSLN
     REGION SIDE LINEAR NODES
RSN
     REGION SIDE NODES
```

```
RZC
     X AND Y COORDINATES OF CONTAINER
RZE
     X AND Y COORDINATES OF ELEMENTS
RZN
     X AND Y COORDINATES OF NODES
RZR
     X AND Y COORDINATES OF NODES OF REGIONS
SAREA
     SIDE AREAS OF RING ELEMENTS
SSC
     STEADY STATE CONTROL (1-EXIT EARLY IF REACH STEADY STATE)
SF
     STATE (OF THE MATERIAL) FIELD
SFN
     SAVE FIELDS EVERY SFN TIME STEPS
SNI
     ORDER OR GAUSS-LEGENDRE QUADRATURE FOR SIDE INTEGRATION
STATMSG
     STATUS MESSAGES
SXEI
     XI AND ETA COORD. USED IN SIDE NUM. INTEGRATION
TCTL
     TIME START, END AND INCREMENT CONTROL
TF
     TEMPERATURE FIELD
THETA
     TRANSIENT ALGORITHM CONTROL PARAMETER
TIME
     RECORD OF TIMES FOR EACH TIME STEP
TIMELIM
     TIME LIMIT ON RUN
TINIT
     INITIAL TEMPERATURE
TSC
     TIME STEP CONTROL (0-CONSTANT, 1-VARIABLE)
UF
     U VELOCITY FIELD
UFINIT
     INITIAL U VELOCITY FIELD
UFSTAR
     U VELOCITY FIELD OF LAST CONVERGENCE ITERATION
VF
     V VELOCITY FIELD
VFINIT
     INITIAL V VELOCITY FIELD
VFSTAR
     V VELOCITY FIELD OF LAST CONVERGENCE ITERATION
VIS
     FLUID VISCOSITY
VOL
     VOLUME OF THE RING ELEMENTS
XEN
     XI AND ETA COORD. OF NODES WRT REGION COORDINATES
```

FLOWCHART OF WORKSPACE

```
LX
    :CLS
    :STRIPE
PHASTRAN
    :FRONTEND
        :CLEANUP
        :INPUT
            :INGEOM
                :INREG
            :INBDY
            :INPROG
        :GRIDGEN
            :GRIDΔRSN
            :GRIDΔRSE
            :GRIDΔRSLN
            :GRIDΔELN
            :GRIDΔLNODE
        :COORXE
        :MAP
        :INITIAL
            :CHKINPUT
                :CHKINPRES
            :AINTCON
                :LSHAPE
                :QLGSHP
                :JACCHK
            :SINTCON
                :QLGSHP
                :LSHAPE
                :SIFAC
            :AREA
            :INITTEMP
            :INITVEL
            :INITPRES
                :INITMP
                    :INTQR
                        :INPLYGN
                        :INTQRΔREG
                            :REMDUPEL
                            :QUADRGXY
                        :INTQRΔUNN
                    :UPPROP
                        :INTQR
                            :INPLYGN
                            :INTQRΔREG
                                :REMDUPEL
                                :QUADRGXY
                            :INTQRΔUNN
                        :UPPROPΔTEMP
                :STATPRES
            :INITMPROP
            :RPRES
```

```
                    :DFN
                         :QLGSHP
                    :DFN
                         :QLGSHP
               :RCOND
               :GRAVVEC
                    :RHOZERO
                         :INTQR
                              :INPLYGN
                              :INTQRΔREG
                                   :REMDUPEL
                                   :QUADRGXY
                              :INTQRΔUNN
               :SETLAST
               :SETSTAR
:TIMESTEP
     :CPUCHK
          :CPUTIME
     :TIMECHK
     :AGAIN
          :SETSTAR
          :ENERGY
               :EGYEVL
                    :EGYMAT
                         :FLDMATΔCM
                         :FLDMATΔRC
                              :BDY
                                   :BDY2
                                        :BDY2ΔSUB
                              :RCOND
                         :FLDMATΔKAE
                         :FLDMATΔRP
                              :RPRES
                                   :DFN
                                        :QLGSHP
                                   :DFN
                                        :QLGSHP
                    :FMFEQΔ1
               :EGYBAR
                    :LUMP
                         :DIAG
               :PRSIDE
                    :PRSIDEΔFLD
                         :PRSIDEΔTEMP
                              :INTQR
                                   :INPLYGN
                                   :INTQRΔREG
                                        :REMDUPEL
                                        :QUADRGXY
                                   :INTQRΔUNN
                         :PRSIDEΔADJ
                    :RDUPL
               :PRSCRB
                    :RDUPL
               :PRSVLC
```

```
      :FLDSLV
            :FLDSLV∆STAR
      :EGYASN
:FLOW
      :FLWEVL
            :FLWMAT
                  :FLDMAT∆MM
                  :FLDMAT∆KA
                  :FLDMAT∆KNN
                  :FLDMAT∆LN
                  :FLDMAT∆LCN
                  :FLDMAT∆R
                        :GRAVVEC
                              :RHOZERO
                                    :INTQR
                                          :INPLYGN
                                          :INTQR∆REG
                                                :REMDUPEL
                                                :QUADRGXY
                                          :INTQR∆UNN
                        :BDY
                              :BDY2
                                    :BDY2∆SUB
                        :BDY
                              :BDY2
                                    :BDY2∆SUB
            :FMFEQ∆2
      :FLWBAR
            :LUMP
                  :DIAG
      :PRSIDE
            :PRSIDE∆FLD
                  :PRSIDE∆TEMP
                        :INTQR
                              :INPLYGN
                              :INTQR∆REG
                                    :REMDUPEL
                                    :QUADRGXY
                              :INTQR∆UNN
                  :PRSIDE∆ADJ
            :RDUPL
      :PRSPRES
            :PRSP1
            :PRSP2
      :PRSCRB
            :RDUPL
      :PRSVLC
      :FLDSLV
            :FLDSLV∆STAR
      :FLWASN
:UPPROP
      :INTQR
            :INPLYGN
            :INTQR∆REG
                  :REMDUPEL
```

```
                    :QUADRGXY
               :INTQRΔUNN
          :UPPROPΔTEMP
     :CHKCNV
          :CHKCNVΔS1
          :CHKCNVΔS1
          :CHKCNVΔS1
          :CHKCNVΔS1
     :SETSOLID
:SETSOLID
:ADJSTEP
     :CHKEF
     :CHKTF
     :INITIAL
          :CHKINPUT
               :CHKINPRES
          :AINTCON
               :LSHAPE
               :QLGSHP
               :JACCHK
          :SINTCON
               :QLGSHP
               :LSHAPE
               :SIFAC
          :AREA
          :INITTEMP
          :INITVEL
          :INITPRES
               :INITMP
                    :INTQR
                         :INPLYGN
                         :INTQRΔREG
                              :REMDUPEL
                              :QUADRGXY
                         :INTQRΔUNN
                    :UPPROP
                         :INTQR
                              :INPLYGN
                              :INTQRΔREG
                                   :REMDUPEL
                                   :QUADRGXY
                              :INTQRΔUNN
                         :UPPROPΔTEMP
               :STATPRES
          :INITMPROP
          :RPRES
               :DFN
                    :QLGSHP
               :DFN
                    :QLGSHP
          :RCOND
          :GRAVVEC
               :RHOZERO
                    :INTQR
                         :INPLYGN
```

```
                                    :INTQRΔREG
                                         :REMDUPEL
                                         :QUADRCXY


                                    :INTQRΔUNN
                         :SETLAST
                         :SETSTAR
               :SETLAST
               :SAVEFIELDS
               :CHKSS
          :RESULTS
```

APPENDIX B

COMPUTER PROGRAM LISTING AND RESULTS

This appendix contains the APL source program listing used in this research. The input data for Case 4 described in Chapter 5 is contained in the functions and global variables listed in this appendix. To run the program after loading the APL workspace enter the following:

PHASTRAN #

where # is the the number of divisions of a side of the solution domain. A value of seven results in 49 equal elements for this case. The results of such a run are given in the global variable RESULTS at the end of this appendix.

```
SYSTEM VARIABLE SETTINGS
⎕CT: 1E¯13
⎕FC: .,*0
⎕HT:
⎕IO: 1
⎕LC:
⎕LX:
⎕PP: 5
⎕PR:
⎕PW: 79
⎕RL: 16807
⎕TZ: 0
⎕WA: 31226812
⎕NLT:
```

```
ADDROWS
[0]   T←TABLE ADDROWS ROW;L
[1]   ⍝ ADDS ROW(S) TO A TABLE FILLING WITH BLANKS OR 0'S
[2]   ⍝
[3]    TABLE←TOMATRIX TABLE ⍝                MAKE SURE TABLE IS A MATRIX
[4]    ROW←TOMATRIX ROW ⍝                    CONVERT ROW TO A MATRIX
[5]    L←(¯1↑⍴TABLE)⌈¯1↑⍴ROW ⍝               FIND LARGEST OF COLUMNS
[6]    TABLE←((1↑⍴TABLE),L)↑TABLE ⍝          RESHAPE TABLE
[7]    T←TABLE,[1]((1↑⍴ROW),L)↑ROW ⍝         ADD ROW(S)
```

```
ADJSTEP
[0]    CIC ADJSTEP NIT;MSG
[1]    ⍝ ADJUSTS TIME STEP BASED ON ABILITY TO CONVERGE AND FIELD VALUES
[2]    ⍝
[3]     →(TSC=0)/0 ⍝                          EXIT IF TIME STEP CAN'T CHANGE
[4]     →(NIT>CIL)/DEC ⍝                      CHECK FOR TOO MANY ITERATIONS
[5]     →(~CHKEF)/DEC ⍝                       IF ENERGY OUT OF RANGE, DECREASE
[6]     →(~CHKTF)/DEC ⍝                       IF TEMP. OUT OF RANGE, DECREASE
[7]     →0 ⍝                                  EXIT, TIME STEP IS OK
[8]    DEC:TIME←TCTL[1] ⍝                     SET BACK TIME TO BEGINNING
[9]     INITIAL ⍝                             REINITIALIZE PROBLEM
[10]    NTS←1 ⍝                               DECREASE TIME STEP COUNTER
[11]    CIC←10⍴0 ⍝                            REINITIALIZE ITER. COUNTER
[12]    TCTL[3]←TCTL[3]÷2 ⍝                   DIVIDE TIME STEP BY 2
[13]   END:MSG←'TIME STEP CHANGED TO' ⍝       MESSAGE TO USER
[14]    STATUS MSG TCTL[3] ⍝                  DISPLAY AND RECORD MESSAGE
[15]    →0 ⍝                                  EXIT
[16]   STOP:STATUS 'NOT CONVERGED' ⍝          MESSAGE TO USER
[17]    → ⍝                                   END EXECUTION
```

```
AGAIN
[0]     NIT←AGAIN ITER
[1]    ⍝ INTERATIVE SOLUTION (FOR NON-LINEARITIES) AT EACH TIME STEP
[2]    ⍝
[3]     NIT←ITER←ITER+1 ⍝              ITERATION COUNTERS
[4]     →(ITER>CIL)/0 ⍝                CHECK IF TOO MANY ITERATIONS
[5]     SETSTAR ⍝                      SET LAST CONVERGENCE ITERATION VARIABLES
[6]     ENERGY ⍝                       FORM AND SOLVE ENERGY EQUATION
[7]     FLOW ⍝                         FORM AND SOLVE THE FLUID EQUATIONS
[8]     UPPROP ⍝                       UPDATE PROPERTIES BASED ON PF AND EF
[9]     →(CHKCNV NIT)/0 ⍝              CHECK CONVERGENCE OF TF, UF, VF AND PF
[10]    SETSOLID ⍝                     SET VELOCITIES OF SOLID NODES TO ZERO
[11]    NIT←AGAIN ITER ⍝               RECURSIVE CALL


AINTCON
[0]     AINTCON N;B1;B2;D;N1;ETA;QJAC;NSHP;LAS;QAS
[1]    ⍝ INTEGRATION CONSTANTS FOR USE WITH AINTGRT
[2]    ⍝
[3]     N1←1+ANI←N ⍝                                    ORDER OF GAUSS-LEGENDRE
[4]     ETA←(N1,N1)⍴N1↑(CI[1;;])[ANI;] ⍝                ETA COORDS.
[5]     AXEI←(2,(N1×N1))⍴(,⍉ETA),(,ETA) ⍝               XI AND ETA COORDS.
[6]     LAS←LSHAPE AXEI ⍝                               LINEAR SHAPE FUNCTION
[7]     QAS←QLGSHP AXEI ⍝                               QUAD. LAGRANGIAN SHAPE FNS.
[8]     QJAC←RZE JACOB QAS ⍝                            JACOBIAN FOR QUAD. ELEMENTS
[9]     QAIF←MDET QJAC ⍝                                QUAD. AREA INTEGRATE FACTOR
[10]    JACCHK QAIF ⍝                                   CHECK JACOBIAN
[11]    B1←RZE[;;1 3 5 7]BF LAS ⍝                       LINEAR GRADIANT MATRIX
[12]    B2←RZE BF QAS ⍝                                 QUAD. GRADIANT MATRIX
[13]    NSHP←4 2 1 3⍉(1,NE,⍴NSHP)⍴NSHP←LAS[1;;]⍝        SHAPE FNS. ELMNT. TYPE 1
[14]    NST←⊂NSHP ⍝                                     PUT IN GLOBAL VARIABLE
[15]    NSHP←4 2 1 3⍉(1,NE,⍴NSHP)⍴NSHP←QAS[1;;]⍝        SHAPE FNS. ELMNT. TYPE 2
[16]    NST←NST,⊂NSHP ⍝                                 ADD TO GLOBAL VARIABLE
[17]    D←4 1 2 3⍉(1,⍴D)⍴D←B1[;;1;] ⍝                   DERIV. WRT. X ELMNT. TYPE 1
[18]    DXT←⊂D ⍝                                        PUT IN GLOBAL VARIABLE
[19]    D←4 1 2 3⍉(1,⍴D)⍴D←B1[;;2;] ⍝                   DERIV. WRT. Y ELMNT. TYPE 1
[20]    DYT←⊂D ⍝                                        ADD TO GLOBAL VARIABLE
[21]    D←4 1 2 3⍉(1,⍴D)⍴D←B2[;;1;] ⍝                   DERIV. WRT. X ELMNT. TYPE 2
[22]    DXT←DXT,⊂D ⍝                                    ADD TO GLOBAL VARIABLE
[23]    D←4 1 2 3⍉(1,⍴D)⍴D←B2[;;2;] ⍝                   DERIV. WRT. Y ELMNT. TYPE 2
[24]    DYT←DYT,⊂D ⍝                                    ADD TO GLOBAL VARIABLE
```

```
AINTGRT
[0]     I←AINTGRT A;WE;WEX;N1
[1]    ⍝ INTEGRATES FUNCTION OVER AREA OF ELEMENT IN XI-ETA COOR. SYSTEM
[2]    ⍝
[3]     N1←ANI+1 ⍝                                   1+ORDER OF INTEGRATION
[4]     →(1∈⍴,A)/SCALAR ⍝                            CHECK IF A IS SCALAR
[5]     →(∧/(⍴QAIF)=2+⍴A)/NXT ⍝                      CHECK IF ⍴A IS LIKE ⍴QAIF
[6]     STATUS ERRMSG[1] ⍝                           MESSAGE TO USER
[7]     →0 ⍝                                         EXIT
[8]    NXT:A←A×(2⊃⍴⍴A)⍴(2⊃⍴A)⍴QAIF ⍝                 MULT. BY INTGRT. FACTOR
[9]     →CALC ⍝                                      JUMP TO FINISH INTEGRATION
[10]   SCALAR:A←A×QAIF ⍝                            MULT. BY INTGRT. FACTOR
[11]   CALC:WE←(N1,N1)⍴N1↑(CI[2;;])[ANI;] ⍝         WEIGHTING FACTORS
[12]    WEX←(⍴A)⍴⍉((×/(1↓⍴A)),(1↓⍴A))⍴(,WE)×.,⍉WE ⍝ RESHAPE WEIGHT FACTORS
[13]    I←+/[1]WEX×A ⍝                               NUMERICAL INTEGRATION


AREA
[0]    AREA;RN
[1]   ⍝ CALCULATES AREAS AND VOLUMES OF ELEMENTS
[2]   ⍝
[3]    SAREA←GF×SINTGRT 1 ⍝                 SIDE AREAS OF ELEMENTS
[4]    VOL←AINTGRT 1 ⍝                      VOLUME OF ELEMENTS


BDY
[0]     R←BDY FT;B;M;NC;N;D;S
[1]    ⍝ EVALUATES R VECTOR FROM BOUNDARY AND INTERNAL CONDITIONS
[2]    ⍝
[3]     B←FT FSTCM BC ⍝                             BC'S FOR THIS FIELD TYPE
[4]     NC←B[;1] ⍝                                  LIST OF BC TYPES
[5]     R←(4,(SNI+1),NE)⍴0 ⍝                        INITIALIZE R VECTOR TO ZERO
[6]    ⍝
[7]    LOOP:→(0=⍴NC)/NXT1 ⍝                        LOOP ON BOUNDARY CONDITIONS
[8]     N←1↑NC ⍝                                    TAKE FIRST BC TYPE
[9]     NC←(N≠NC)/NC ⍝                              REMOVE THIS TYPE FROM LIST
[10]    →(N=1)/LOOP ⍝                               IGNORE PRESCRIBED BC'S
[11]    B←N FSTCM B ⍝                               BC'S FOR THIS BC TYPE
[12]    D←RN FSTCM B ⍝                              BC'S FOR THIS REGION NUMBER
[13]    →(0=×/⍴D)/NXT1 ⍝                            EXIT IF NO SUCH BC'S
[14]    →(N=2)/B2 ⍝                                 BOUNDARY CONDITION TYPE 2
[15]    M←'BOUNDARY CONDITION NOT DEFINED (BDY)'⍝   WARNING TO USER
[16]    STATUS M ⍝                                  DISPLAY AND RECORD MESSAGE
[17]    →LOOP ⍝                                     CONTINUE LOOPING
[18]   B2:R←D BDY2 R ⍝                              PRESCRIBED FLUX BC
[19]    →LOOP ⍝                                     END OF LOOP
[20]   ⍝
[21]   NXT1:→(1 2=FELT[FT])/LINEAR,QUAD ⍝           CHECK IF LINEAR ELEMENTS
[22]    STOP ⍝                                      WRONG ELEMENT TYPE
[23]   QUAD:S←QSS[1;;] ⍝                           QUADRATIC SHAPE FUNCTIONS
[24]    →NXT2 ⍝                                     FINISH CALC. OF R VECTOR
[25]   LINEAR:S←LSS[1;;] ⍝                         LINEAR SHAPE FUNCTIONS
[26]   NXT2:R←4 2 3 1⍉((¯1↑⍴S),⍴R)⍴R ⍝             RESHAPE R
[27]    R←3 1 2 4⍉((⍴R)⍴S)×R ⍝                      AND MULTIPLY BY SHAPE FNS.
```

```
BDY2
[0]   G←D BDY2 R;D1
[1]   ⍝ MODIFIES ELEMENT R MATRIX FOR BOUNDARY COND. TYPE 2 (PRESCRIBED FLUX)
[2]   ⍝    D[;1] IS THE SIDE NUMBERS
[3]   ⍝    D[;2] IS THE PRESCRIBED FLUX VALUES
[4]   ⍝
[5]    →(0=1↑⍴D)/0 ⍝                          EXIT IF NO BOUNDARY CONDITIONS
[6]    G←R ⍝                                  INITIALIZE RETURN VARIABLE
[7]    D1←⊂[2]D ⍝                             NESTED ARRAY OF BOUNDARY COND.
[8]    BDY2ΔSUB⍤¯1⊢D1 ⍝                       HANDLE EACH BOUNDARY CONDITION
[9]    G←R ⍝                                  RETURN R VECTOR FROM BDYΔSUB


BDY2ΔSUB
[0]   BDY2ΔSUB D1;EL;S;PV
[1]   ⍝ SUB-FUNCTION OF BDY2 TO PRESCRIBE EACH (ELEMENT, SIDE COMBINATION)
[2]   ⍝    D1 IS A NESTED ARRAY OF PRESCRIBED SIDE BOUNDARY CONDITIONS
[3]   ⍝    R IS THE R MATRIX FROM BDY2 WHICH IS MODIFIED BY THIS FUNCTION
[4]   ⍝
[5]    S←D1[1] ⍝                              SIDE AFFECTED
[6]    EL←RSELMT[S;] ⍝                        ELEMENTS AFFECTED
[7]    PV←((⍴R)[2],⍴EL)⍴⍤¯1⍤¯1⊢D1[2] ⍝        PRSCRBD. VALUES AT INTG. PTS, EL
[8]    R[S;;EL]←PV ⍝                          MODIFY THE R MATRIX


BF
[0]   B←RZ BF S;C;RS
[1]   ⍝ CALCULATES THE FIELD VARIABLE GRADIANT INTERPOLATION MATRICIES
[2]    RS←1↑⍴S[2;;]
[3]    B←2 1 3 4⍉(NE,⍴C)⍴C←S[2;;],[1.5]S[3;;]
[4]    B←(INV RZ JACOB S)INPROD B


CHKCNV
[0]    CNV←CHKCNV NIT
[1]    ⍝ CHECKS CONVERGENCE OF EF, UF, VF AND PF WITH LAST ITERATION VALUES
[2]    ⍝  CNV RETURNS 1 IF ALL CONVERGED   0 IF NOT ALL CONVERGED
[3]    ⍝
[4]     CNV←ERR CHKCNVΔS1 'EF' ⍝                CHECK EF
[5]     →(FC=0)/END ⍝                          JUMP IF NO FLOW CALCS.
[6]     CNV←CNV∧ERR CHKCNVΔS1 'UF' ⍝           CHECK UF
[7]     CNV←CNV∧ERR CHKCNVΔS1 'VF' ⍝           CHECK VF
[8]     CNV←CNV∧ERR CHKCNVΔS1 'PF' ⍝           CHECK PF
[9]    END:→(CNV=0)/LIMIT ⍝                    CHECK FOR CONVERGENCE
[10]    STATUS 'CONVERGED WITH' NIT 'ITERATIONS' ⍝  MESSAGE TO USER
[11]    →0 ⍝                                   EXIT
[12]   LIMIT:→(NIT<CIL)/0 ⍝                    EXIT FOR MORE ITERATIONS
[13]    STATUS 'NOT CONVERGED IN' NIT 'ITERATIONS'⍝  MESSAGE TO USER
[14]   ⍝
```

```
CHKCNVΔS1
[0]     CNV←ERR CHKCNVΔS1 FLDNM;AFC;FLD;FLDSTAR;LS
[1]    ⍝ CHECKS FIELD VALUES FOR CONVERGENCE
[2]    ⍝
[3]     CNV←0 ⍝                                    DEFAULT IS NOT CONVERGED
[4]     FLD←⍎FLDNM ⍝                               FIELD VALUES
[5]     FLDSTAR←⍎FLDNM,'STAR' ⍝                    FIELD VALUES OF LAST ITERATION
[6]     LS←(|FLD)<ERR×⌈/|FLD ⍝                     LOCATIONS WITH SMALL VALUES
[7]     LS←LS∨(|FLD)<1E¯10 ⍝                       OR SMALL ABSOLUTE VALUES
[8]     LS←LS∨(|FLDSTAR)<ERR×⌈/|FLDSTAR ⍝          OR RELATIVE SMALL STAR VALUES
[9]     LS←LS∨0=FLDSTAR ⍝                          OR ZERO FLDSTAR VALUES
[10]    →(∧/LS)/CANT ⍝                             CHECK IF NO VALUES WILL LEFT
[11]    FLD←(LS←~LS)/FLD ⍝                         IGNORE SMALL FIELD VALUES
[12]    FLDSTAR←LS/FLDSTAR ⍝                       AND THOSE SMALL STAR VALUES
[13]    AFC←|(FLD-FLDSTAR)÷FLDSTAR ⍝               ABSOLUTE FRACTIONAL CHANGE
[14]    CNV←×/ERR>AFC ⍝                            RETURN A 1 IF ALL CONVERGED
[15]    →(CNV)/0 ⍝                                 EXIT IF ALL CONVERGED
[16]    →(ITER<CIL)/0 ⍝                            EXIT IF NOT AT ITERATION LIMIT
[17]    STATUS 'MAX ERROR ',FLDNM,': ',5⍕⌈/AFC⍝    DISPLAY MAXIMUM ERROR
[18]    →0 ⍝                                       EXIT
[19]   CANT: ⍝                                     CAN'T DETERMINE CONVERGENCE
[20]    STATUS 'UNKNOWN CONVERGENCE OF' FLDNM⍝     WARN USER
[21]    CNV←ITER≠1 ⍝                               OK IF NOT FIRST ITERATION


CHKEF
[0]    REC←CHKEF;RGD;RGE
[1]   ⍝ CHECKS ENERGY FIELD VALUES TO SEE IF WITHIN PROPERTY DATA RANGE
[2]   ⍝
[3]    RGD←MINMAX,PROPDATA[2;;] ⍝                  RANGE OF DATA VALUES
[4]    RGE←MINMAX EF ⍝                             RANGE OF ENERGY FIELD VALUES
[5]    REC←∧/RGE<.≤RGD ⍝                           CHECK IF WITHIN RANGE


CHKINPRES
[0]     CHKINPRES;A
[1]    ⍝ CHECK PRESRCRIBED PRESSURE NODE "PRNODE" SPECIFIED IN INPUT
[2]    ⍝
[3]     →(2≠⎕NC 'PRNODE')/0 ⍝                      CHECK IF CONFLICT WITH MATL. PROP.
[4]     A←MINMAX,(PROPDATAΔSEL⍳6)[1;;] ⍝           MIN. AND MAX. PRESSURE
[5]     →(A[2]<PRNODE[2])/ERR1 ⍝                   CHECK IF TOO HIGH
[6]     →(A[1]>PRNODE[2])/ERR2 ⍝                   CHECK IF TOO LOW
[7]     →0
[8]    ERR1:'CHECK INPUT:  TOO HIGH PRESSURE SPECIFIED FOR PROPERTY DATA'
[9]     STOP
[10]   ERR2:'CHECK INPUT:  TOO LOW PRESSURE SPECIFIED FOR PROPERTY DATA'
[11]    STOP
```

```
CHKINPUT
[0]    CHKINPUT;A;M
[1]   ⍝ CHECK INPUT PARAMETERS
[2]   ⍝
[3]    CHKINPRES ⍝                                    CHECK PRESSURE SPECIFICATIONS
[4]    →(~(5<A)∨0.2>A←+⌿(⌈/RZR)-L/RZR)/0 ⍝            CHECK ELEMENT ASPECT RATIOS
[5]    'WARNING:  ASPECT RATIO OF ELEMENTS GREATER THAN 5'


CHKSS
[0]    OK←CHKSS CIC
[1]   ⍝ CHECK IF STEADY STATE HAS BEEN REACHED
[2]   ⍝
[3]    OK←0 ⍝                                         INITIALIZE RETURN VARIABLE
[4]    →(SSC=0)/0 ⍝                                   STEADY STATE CONTROL PARAM.
[5]    OK←∧/1=CIC ⍝                                   CHECK LAST ITERATION COUNTS
[6]    →(~OK)/0 ⍝                                     EXIT IF ITERATIONS > 1
[7]    STATUS 'STEADY STATE SOLUTION REACHED' ⍝       MESSAGE TO USER


CHKTF
[0]     RTC←CHKTF;RGB;RGT;TMP;TOL
[1]    ⍝ CHECKS TEMPERATURE FIELD VALUES IF WITHIN PRESCRIBED VALUES
[2]    ⍝
[3]     TOL←.1 ⍝                                      TOLERANCE RANGE
[4]     RTC←1 ⍝                                       ASSUME OK
[5]     TMP←1 FSTCM BC ⍝                              ENERGY BOUNDARY CONDITIONS
[6]     TMP←1 FSTCM TMP ⍝                             ONLY PRESCRIBED BCS.
[7]     TMP←RN FSTCM TMP ⍝                            ONLY THIS REGION
[8]     TMP←0 1↓TMP ⍝                                 ANY SIDE
[9]     →(2>1↑⍴TMP)/0 ⍝                               EXIT IF NOT ENOUGH INFO.
[10]    RGB←MINMAX⍤⍤⍤TMP[;1] ⍝                        PRSCRB. TEMP. BCS.
[11]    RGB←(MEAN RGB)+¯1 1×(1+TOL)×0.5×-/⌽RGB ⍝      ADD TOLERANCE TO RANGE
[12]    RGT←MINMAX TF ⍝                               RANGE OF TEMPERATURE VALUES
[13]    RTC←∧/RGT<.≤RGB ⍝                             CHECK IF WITHIN RANGE


CLEANUP
[0]     CLEANUP;NAMES;ALTALP;C
[1]    ⍝ ERASES VARIABLES NAMED IN GLV EXCEPT THOSE WITH ALT. CHARACTER NAMES
[2]    ⍝
[3]     →(0=⎕NC 'GLV')/0 ⍝                            EXIT IF GLV NOT AVAILABLE
[4]     NAMES←((1↑⍴GLV),10)↑GLV ⍝                     GET VARIABLE NAMES FROM GLV
[5]     ALTALP←'ABCDEFGHIJKLMNOPQRSTUVWXYZ' ⍝         ALTERNATE ALPHABET
[6]     NAMES←(~∨/∨/NAMES∘.=ALTALP)⌿NAMES ⍝           DO NOT ERASE ALT. CHAR. NAMES
[7]     C←⎕EX NAMES ⍝                                 ERASE THE REST
[8]     →(1=×/C)/0 ⍝                                  CHECK IF ALL WERE ERASED
[9]     'VARIABLES NOT ERASED IN CLEANUP' ⍝           MESSAGE TO USER
[10]    NAMES[((~C)/(⍳⍴C)×C=0);] ⍝                    DISPLAY THOSE NOT ERASED
```

```
CLS
[0]    CLS;RC;CTLS;DATS
[1]   ⍝ CLEARS 3270 SCREEN
[2]    RC←120 ⎕SVO 2 4⍴'CTLSDATS'
[3]    →(∨/2≠RC)/NOSHARE
[4]    RC←1 0 1 0 ⎕SVC 'CTLS'
[5]    CTLS←'PAGE +1'
[6]    RC←CTLS
[7]    →(0∧.=RC)/0
[8]    'RETURN CODE OF ',(⍕RC),' FROM AP 120'
[9]    DATS
[10]   RC←⎕SVR 2 4⍴'CTLSDATS'
[11]   →0
[12]  NOSHARE:'OFFER TO AP 120 NOT ACCEPTED'


COLM
[0]    R←I COLM X
[1]   ⍝ RETURNS VALUES FOR "I" INDEX OF LAST DIMENSION OF "X"
[2]   ⍝
[3]    R←(I=⍳¯1+⍴X)/X ⍝                    SELECT DATA
[4]    →(1<⍴⍴,I)/0 ⍝                       IF MULTIPLE COLUMNS, FINISHED
[5]    R←(¯1+⍴R)⍴R ⍝                       ELSE RESHAPE, ELIM. LAST DIMENSION


COORXE
[0]    COORXE;A
[1]   ⍝ GENERATES XI-ETA COORDINATES OF THE NODES
[2]   ⍝
[3]    A←¯1+(¯1+⍳1+2×ND)×2÷2×ND ⍝          NORMALIZED NODES LOCATIONS
[4]    A←(2⍴⍴A)⍴A ⍝                        XI COORDINATES
[5]    A←A,[0.5]⌽⊖A ⍝                      ADD ETA COORDINATES
[6]    XEN←(2,0.5××/⍴A)⍴A ⍝                COORDS. WRT REGION COORDS.


CPUCHK
[0]    CK←CPUCHK
[1]   ⍝ RETURNS A 0 IF CPU TIME LIMIT IS EXCEEDED
[2]   ⍝
[3]    CK←~CPUTIME≥CPULIM ⍝                CHECK CPU TIME
[4]    →(CK=1)/0 ⍝                         EXIT IF OK
[5]    STATUS 'CPU LIMIT EXCEEDED' ⍝       MESSAGE TO USER


CPUTIME
[0]    CPU←CPUTIME
[1]   ⍝ RETURNS CPU SECONDS USED SINCE "TSTART" WAS ISSUED
[2]    CPU←0.001×⎕AI[2]-T1[2]
```

```
DFN
[0]   DF←DFN F;A;B;S
[1]   ⍝ CALCULATES DERIVATIVES OF FIELD VARIABLE AT THE ELEMENT NODES
[2]   ⍝
[3]   A←¯1 ¯1 0 ¯1 1 ¯1 1 0 1 1 0 1 ¯1 1 ¯1 0 0 0 ⍝  NODE XI-ETA COORDINATES
[4]   S←QLGSHP⍵9 2⍴A ⍝                              SHAPE FUNCTIONS AT NODES
[5]   B←RZE BF S ⍝                                  GRAD. INTERP. MATRIX
[6]   DF←3 2 1⍉+/B×OPAX(2 4)(F[ELN]) ⍝              DERIVATIVES AT NODES


DIAG
[0]   A←DIAG X
[1]   ⍝ FORMS DIAGONAL MATRIX FROM A VECTOR X
[2]   A←0 ¯1↓(-⍳⍴X)⌽((2⍴⍴X)⍴0),X


DIST
[0]   R←A DIST B
[1]   ⍝ CALCULATES THE DISTANCE BETWEEN CARTESIAN POINTS A AND POINTS B
[2]   ⍝    LAST DIMENSION OF A AND B IS 2 COLUMNS OF X AND Y'S
[3]   ⍝
[4]   R←(+/(B-A)*2)*0.5 ⍝                SQUARE ROOT OF SUM OF DIFF. SQUARED


DX
[0]   R←DX T
[1]   ⍝ RETURNS DERIVATIVES OF THE SHAPE FUNCTIONS WRT. X
[2]   ⍝   T IS THE FIELD TYPE (E.G. 1-ENERGY)
[3]   ⍝
[4]   R←⊃DXT[FELT[T]] ⍝                  SELECT FROM GLOBAL NESTED ARRAY


DY
[0]   R←DY T
[1]   ⍝ RETURNS DERIVATIVES OF THE SHAPE FUNCTIONS WRT. Y
[2]   ⍝   T IS THE FIELD TYPE (E.G. 1-ENERGY)
[3]   ⍝
[4]   R←⊃DYT[FELT[T]] ⍝                  SELECT FROM GLOBAL NESTED ARRAY


EGYASN
[0]   LPV EGYASN F
[1]   ⍝ ASSIGN ENERGY FIELD VALUES
[2]   ⍝
[3]   F←LPV\F ⍝                          EXPAND FOR PRESCRIBED VALUES
[4]   F[(PFEQ[1;])]←PFEQ[2;] ⍝           REINSERT PRESCRIBED VALUES
[5]   EF←FAN F ⍝                         ENERGY VALUES AT ALL NODES
```

```
EGYBAR
[0]    EGYBAR;A;B;CDT
[1]   ⍝ FORMS ENERGY EQUATIONS FOR SOLVING TRANSIENT RESPONSE
[2]   ⍝
[3]    CEQ←LUMP CEQ ⍝                          USE LUMPED CAPACITANCE
[4]    CDT←CEQ÷TCTL[3] ⍝                       CAPACITANCE ÷ ΔTIME STEP
[5]    KBAR←(THETA×KEQ)+CDT ⍝                  NEW STIFFNESS MATRIX
[6]    A←(CDT+KEQ×THETA-1)+.×EPHI ⍝            INTERMEDIATE CALCULATION
[7]    B←,(RELAST×1-THETA)+THETA×EREQ ⍝        INTERMEDIATE CALCULATION
[8]    RBAR←A+B ⍝                              NEW RESULTANT VECTOR


EGYEVL
[0]    EGYEVL;KAE;CM;RC;RP
[1]   ⍝ EVALUATES ENERGY MATRICES
[2]   ⍝
[3]    PFEQ←3 0⍴0 ⍝                        INITIALIZE PRESCRIBE VALUES MATRIX
[4]    EGYMAT ⍝                            CONSTRUCT FLUID SUB-MATRICES
[5]    FMFEQΔ1 ⍝                           FORM FIELD EQUATIONS


EGYMAT
[0]    EGYMAT
[1]   ⍝ CONSTRUCT THE ENERGY ELEMENT MATRICES
[2]   ⍝
[3]    FLDMATΔCM ⍝                             FORM CM MATRIX
[4]    FLDMATΔRC ⍝                             FORM RC VECTOR
[5]    →(FC=0)/0 ⍝                             EXIT IF NO FLOW CALCS.
[6]    FLDMATΔKAE ⍝                            FORM KAE MATRIX
[7]    FLDMATΔRP ⍝                             FORM RP VECTOR


ENERGY
[0]     ENERGY;LPV;FT;RESULT
[1]    ⍝ FORMS AND SOLVES THE TRANSIENT ENERGY EQUATION
[2]    ⍝
[3]     →(EC=0)/0 ⍝                        EXIT IF NO THERMAL ENERGY CALCS.
[4]     FT←1 ⍝                             FIELD TYPE IS ENERGY
[5]     EGYEVL ⍝                           FORM THE ENERGY EQUATIONS
[6]     EGYBAR ⍝                           MODIFY EQUATIONS FOR TRANSIENT FORM.
[7]     PRSIDE FT ⍝                        PRSCRIBE REGION SIDE BC'S
[8]     PRSCRB ⍝                           MODIFY FLUID ELEMENT EQS.
[9]     LPV←PRSVLC FT ⍝                    LOCATIONS OF PRESCRIBED VALUES
[10]    RESULT←FLDSLV LPV ⍝                SOLVE THE FLUID EQUATIONS
[11]    LPV EGYASN RESULT ⍝                ASSIGN THE FIELD VALUES
```

```
FAN
[0]    R←FAN F;MAX
[1]    ⍝ RETURNS FIELD VARIABLE AT ALL NODES
[2]    ⍝    F IS FIELD VARIABLE
[3]    ⍝    R IS MAXIMUM OF NNG FOR ALL FIELD TYPES
[4]    ⍝
[5]     R←F ⍝                              INITIALIZE RETURN VARIABLE
[6]     MAX←⌈/NNG⍳⍴FELT ⍝                  MAXIMUM NODES PER ELEMENT
[7]     →(MAX=⍴F)/0 ⍝                      EXIT IF ALREADY SAME AS LARGEST ELEMENT
[8]     R←LINFV F ⍝                        CONVERT LINEAR TO QUADRATIC VALUES


FLDMAT∆CM
[0]    FLDMAT∆CM;A
[1]    ⍝ FORMS CM MATRIX FOR FLDMAT
[2]    ⍝
[3]     A←1 FVEB RHO
[4]     A←1 2 4 3⍉(NS 1)×OPAX(2 3)A
[5]     A←(NS 1)INPROD A
[6]     CM←NODEM AINTGRT A


FLDMAT∆KA
[0]    FLDMAT∆KA;A
[1]    ⍝ FORMS KA MATRIX FOR FLDMAT
[2]    ⍝
[3]     A←(NS 2)×OPAX(2 3)((2 FVEB RHO)×2 FVEB UF)
[4]     KA←NODEM AINTGRT A INPROD 1 2 4 3⍉DX 2
[5]     A←(NS 2)×OPAX(2 3)((2 FVEB RHO)×2 FVEB VF)
[6]     KA←KA+NODEM AINTGRT A INPROD 1 2 4 3⍉DY 2


FLDMAT∆KAE
[0]    FLDMAT∆KAE;A
[1]    ⍝ FORMS KAE MATRIX FOR FLDMAT
[2]    ⍝
[3]     A←(NS 1)×OPAX(2 3)((1 FVEB RHO)×1 FVEB UF)
[4]     KAE←NODEM AINTGRT A INPROD 1 2 4 3⍉DX 1
[5]     A←(NS 1)×OPAX(2 3)((1 FVEB RHO)×1 FVEB VF)
[6]     KAE←KAE+NODEM AINTGRT A INPROD 1 2 4 3⍉DY 1
```

```
FLDMATΔKNN
[0]    FLDMATΔKNN;A
[1]   ⍝ FORMS K11 K12 K21 K22 MATRICIES FOR FLDMAT
[2]   ⍝
[3]    A←1 2 4 3⍉(DX 2)×OPAX(2 3)(2 FVEB VIS)
[4]    K11←NODEM AINTGRT(DX 2)INPROD A
[5]    K12←NODEM AINTGRT(DY 2)INPROD A
[6]    A←1 2 4 3⍉(DY 2)×OPAX(2 3)(2 FVEB VIS)
[7]    K21←NODEM AINTGRT(DX 2)INPROD A
[8]    K22←NODEM AINTGRT(DY 2)INPROD A


FLDMATΔLCN
[0]    FLDMATΔLCN;A
[1]   ⍝ FORM LC1 AND LC2 MATRICIES FOR FLDMAT
[2]   ⍝
[3]    A←(NS 4)INPROD 1 2 4 3⍉DX 2
[4]    LC1←NODEM AINTGRT A×OPAX(2 4)(2 FVEB RHO)
[5]    A←(NS 4)INPROD 1 2 4 3⍉DY 3
[6]    LC2←NODEM AINTGRT A×OPAX(2 4)(3 FVEB RHO)


FLDMATΔLN
[0]    FLDMATΔLN
[1]   ⍝ FORMS L1 AND L2 MATRICIES FOR FLDMAT
[2]   ⍝
[3]    L1←-NODEM AINTGRT(DX 2)INPROD 1 2 4 3⍉NS 4
[4]    L2←-NODEM AINTGRT(DY 3)INPROD 1 2 4 3⍉NS 4


FLDMATΔMM
[0]    FLDMATΔMM;A
[1]   ⍝ FORMS MASS MATRIX FOR FLDMAT
[2]   ⍝
[3]    A←2 FVEB RHO ⍝                                  RHO AT QUAD. ELMNT NODES
[4]    A←(NS 2)INPROD 1 2 4 3⍉(NS 2)×OPAX(2 3)A ⍝      INTER. CALC.
[5]    MM←NODEM AINTGRT A ⍝                            MASS (INERTIA) MATRIX


FLDMATΔR
[0]    FLDMATΔR;GV
[1]   ⍝ FORM R VECTORS FOR FLDMAT
[2]   ⍝
[3]    GV←GRAVVEC
[4]    RU←GV[1;;]+NODEV SINTGRT BDY 2
[5]    RV←GV[2;;]+NODEV SINTGRT BDY 3
```

```
FLDMATΔRC
[0]    FLDMATΔRC
[1]  ⍝ FORMS RC VECTOR FOR FLDMAT
[2]  ⍝
[3]    RC←NODEV SINTGRT BDY 1
[4]    RC←RC+RCOND


FLDMATΔRP
[0]    FLDMATΔRP
[1]  ⍝ FORMS AND ASSIGNS RP MATRICIES FOR FLDMAT
[2]  ⍝
[3]    RP←RPRES


FLDSLV
[0]    F←FLDSLV LPV;X
[1]   ⍝ SOLVES EQUATIONS AND ASSIGNS FIELD VALUES
[2]   ⍝
[3]    →(ICTL=0 1)/SS,NR ⍝                  CHECK TYPE OF ITERATION METHOD
[4]   SS: ⍝                                 SUCCESSIVE SUBSTITUTION
[5]    F←,RBAR⌹KBAR ⍝                       SOLVE SIMULTANEOUS EQUATIONS
[6]    →0 ⍝                                 EXIT
[7]   NR: ⍝                                 NEWTON-RAPHSON
[8]    X←LPV/FLDSLVΔSTAR ⍝                  FIELDS FROM LAST ITERATION
[9]    F←(KBAR+.×X)-RBAR ⍝                  NEW FORCING FUNCTION
[10]   F←X+(-F)⌹KBAR ⍝                      SOLVE SIMULTANEOUS EQUATIONS


FLDSLVΔSTAR
[0]    F←FLDSLVΔSTAR
[1]  ⍝ RETURNS FIELDS FROM LAST ITERATION
[2]  ⍝
[3]    →(~FC=0)/N1 ⍝                        CHECK IF FLOW CALCS. WERE SOLVED
[4]    F←EF ⍝                               FOR CASE OF NO FLOW CALCULATIONS
[5]    →0 ⍝                                 EXIT
[6]  N1:EF,UF,VF,PF ⍝                       FOR CASE WITH FLOW CALCULATIONS
```

```
FLOW
[0]    FLOW;LPV;FT;RESULT
[1]   ⍝ FORM AND SOLVE THE FLUID FLOW EQUATIONS
[2]   ⍝
[3]    →(FC=0)/0 ⍝                          EXIT IF NO FLOW CALCS.
[4]    FT←2 3 4 ⍝                           FIELD TYPES U, V, AND P
[5]    FLWEVL ⍝                             FORM THE FLUID EQUATIONS
[6]    FLWBAR ⍝                             MODIFY EQUATIONS FOR TRANSIENT FORM.
[7]    PRSIDE FT ⍝                          PRSCRIBE REGION SIDE BC'S
[8]    PRSPRES ⍝                            PRESCRIBE PRESSURE NODE
[9]    PRSCRB ⍝                             MODIFY FLUID ELEMENT EQS.
[10]   LPV←PRSVLC FT ⍝                      LOCATIONS OF PRESCRIBED VALUES
[11]   RESULT←FLDSLV LPV ⍝                  SOLVE THE FLUID EQUATIONS
[12]   LPV FLWASN RESULT ⍝                  ASSIGN THE FIELD VALUES


FLWASN
[0]    LPV FLWASN F
[1]   ⍝ ASSIGN FLUID FLOW FIELD VALUES
[2]   ⍝
[3]    F←LPV\F ⍝                                  EXPAND FOR PRESCRIBED VALUES
[4]    F[(PFEQ[1;])]←PFEQ[2;] ⍝                   REINSERT PRESCRIBED VALUES
[5]    UF←(A←NNG 2)←F ⍝                           EXTRACT U VELOCITY VALUES
[6]    F←A←F ⍝                                    DROP THOSE VALUES
[7]    VF←(A←NNG 3)←F ⍝                           EXTRACT V VELQCITY VALUES
[8]    F←A←F ⍝                                    DROP THOSE VALUES
[9]    PF←(NNG 4)←F ⍝                             EXTRACT PRESSURE VALUES
[10]   (UF VF PF)←FAN⍤1 UF VF PF ⍝                EXPAND TO ALL NODES


FLWBAR
[0]    FLWBAR;A;B;CDT
[1]   ⍝ FORMS FLOW EQUATIONS FOR SOLVING TRANSIENT RESPONSE
[2]   ⍝
[3]    CEQ←LUMP CEQ ⍝                             USE LUMPED CAPACITANCE
[4]    CDT←CEQ÷TCTL[3] ⍝                          CAPACITANCE ÷ ΔTIME STEP
[5]    KBAR←(THETA×KEQ)+CDT ⍝                     NEW STIFFNESS MATRIX
[6]    A←(CDT+KEQ×THETA-1)+.×FPHI ⍝               INTERMEDIATE CALCULATION
[7]    B←,(RFLAST×1-THETA)+THETA×FREQ ⍝           INTERMEDIATE CALCULATION
[8]    RBAR←A+B ⍝                                 NEW RESULTANT VECTOR


FLWEVL
[0]    FLWEVL;MM;KA;KC;K11;K12;K21;K22;L1;L2;RC;RU;RV;LC1;LC2;RP
[1]   ⍝ EVALUATES FLUID FLOW ELEMENT MATRICES
[2]   ⍝
[3]    PFEQ←3 0⍴0 ⍝                               INITIALIZE PRESCRIBE VALUES MATRIX
[4]    FLWMAT ⍝                                   CONSTRUCT FLUID SUB-MATRICES
[5]    FMFEQ⍤2 ⍝                                  FORM FIELD EQUATIONS
```

```
FLWMAT
[0]    FLWMAT
[1]  ⍝ CONSTRUCT THE FLUID FLOW MATRICES
[2]  ⍝
[3]    FLDMAT∆MM ⍝                    FORM MASS MATRIX
[4]    FLDMAT∆KA ⍝                    FORM KA MATRIX
[5]    FLDMAT∆KNN ⍝                   FORM K11 K12 K21 K22 MATRICIES
[6]    FLDMAT∆LN ⍝                    FORM L1 L2 MATRICIES
[7]    FLDMAT∆LCN ⍝                   FORM LC1 LC2 MATRICIES
[8]    FLDMAT∆R ⍝                     FORM RU RV VECTORS
```

```
FMFEQ∆1
[0]    FMFEQ∆1
[1]  ⍝ FORM ENERGY FIELD EQUATIONS
[2]  ⍝
[3]    CEQ←CM ⍝                       CAPACITANCE MATRIX
[4]    EREQ←RC ⍝                      ENERGY RESULTANT VECTOR
[5]    →(FC=1)/FLOW ⍝                 CHECK FOR FLUID FLOW CALCS.
[6]    KEQ←(⍴CEQ)⍴0 ⍝                 NO FLOW STIFFNESS MATRIX
[7]    →0 ⍝                           EXIT
[8]  FLOW:KEQ←KAE ⍝                   WITH FLOW STIFFNESS MATRIX
[9]    EREQ←EREQ+RP ⍝                 WITH FLOW RESULTANT MATRIX
```

```
FMFEQ∆2
[0]    FMFEQ∆2;Z1;Z2;Z3
[1]  ⍝ FORM FIELD EQUATIONS
[2]  ⍝
[3]    Z1←(NNG 2 4)⍴0 ⍝                ZERO MATRIX
[4]    Z2←(NNG 2 2)⍴0 ⍝                ZERO MATRIX
[5]    Z3←(NNG 4 4)⍴0 ⍝                ZERO MATRIX
[6]    KEQ←(KA+K22+(4+3)×K11),K12,L1 ⍝ X MOMENTUM (STIFFNESS)
[7]    KEQ←KEQ,[1]K21,(KA+K11+(4+3)×K22),L2 ⍝ Y MOMENTUM (STIFFNESS)
[8]    KEQ←KEQ,[1]LC1,LC2,Z3 ⍝         CONTINUITY (STIFFNESS)
[9]    FREQ←RU,[1]RV,[1]((NNG 4),1)⍴0 ⍝ FLOW RESULTANT VECTOR
[10]   CEQ←MM,Z2,Z1 ⍝                 X MOMENTUM (INERTIAL)
[11]   CEQ←CEQ,[1]Z2,MM,Z1 ⍝          Y MOMENTUM (INERTIAL)
[12]   CEQ←CEQ,[1](⊖Z1),(⊖Z1),Z3 ⍝    CONTINUITY (INERTIAL)
```

```
FRONTEND
[0]     FRONTEND A
[1]    ⍝ PERFORMS UPFRONT ONCE ONLY FUNCTIONS
[2]    ⍝
[3]     TSTART ⍝                    START CLOCK FOR TIME CHECKING
[4]     CLEANUP ⍝                   ERASE OLD VARIABLES
[5]     STATMSG←0 0⍴' ' ⍝           INITIALIZE STATUS MESSAGE
[6]     ND←A ⍝                      NUMBER OF DIVISIONS PER SIDE OF REGION
[7]     INPUT ⍝                     SPECIFY INPUT PARAMETERS
[8]     GRIDGEN ⍝                   GENERATE ELEMENT NODE NUMBERING
[9]     COORXE ⍝                    GENERATE XI-ETA COORDINATES OF THE NODES
[10]    MAP ⍝                       MAP XI-ETA NODE COORDINATES INTO X-Y SYSTEM
[11]    INITIAL ⍝                   INITIALIZE VARIABLES


FSTCM
[0]     R←X FSTCM M
[1]    ⍝ RETURNS SUB-MATRIX WHERE MEMBERS OF X MATCH FIRST COLUMN OF M
[2]    ⍝
[3]     R←(M[;1]∈X)⌿0 1⍉M


FST1
[0]     R←FST1 A
[1]    ⍝ PUTS 1 IN COLUMN OF EACH ROW OF A WHERE FIRST POSITIVE CHANGE OCCURS
[2]     R←<\A


FVEB
[0]     R←FT FVEB X;ET
[1]    ⍝ RETURNS THE FIELD VARIABLE ON AN ELEMENT NODE BASIS
[2]    ⍝    X IS FIELD VAR. ON A GLOBAL NODE BASIS FOR HIGHEST ORDER ELEMENT
[3]    ⍝    FT IS THE TYPE OF FIELD (E.G. 1-ENERGY)
[4]    ⍝
[5]     ET←FELT[FT] ⍝                  CONVERT FIELD TYPE TO ELEMENT TYPE
[6]     →(ET=1 2)/TYPE1,TYPE2 ⍝        CHECK TYPE OF ELEMENT
[7]    TYPE1:R←X[ELN[;1 3 5 7]] ⍝      FIELD VARIABLES ON LINEAR ELEMENT BASIS
[8]     →0 ⍝                           EXIT
[9]    TYPE2:R←X[ELN] ⍝                FIELD VARIABLES ON QUAD. ELEMENT BASIS
```

```
FVCB
[0]    R←FT FVCB X;A;ET
[1]    ⍝ RETURNS THE FIELD VARIABLE ON AN GLOBAL NODE BASIS
[2]    ⍝    X IS FIELD VAR. ON A GLOBAL NODE BASIS FOR HIGHEST ORDER ELEMENT
[3]    ⍝    FT IS THE TYPE OF FIELD (E.G. 1-ENERGY)
[4]    ⍝
[5]     ET←FELT[FT] ⍝                          CHANGE FIELD TYPE TO ELEMENT TYPE
[6]     →(ET=1 2)/T1,T2 ⍝                      CHECK TYPE OF ELEMENT
[7]    T1: ⍝                                   CONVERT TO LINEAR GLOBAL BASIS
[8]     X←(2⍴A←1+2×ND)⍴X ⍝                    MAKE FIELD VARIABLE A MATRIX
[9]     X←(A←A⍴1 0)/X ⍝                       REMOVE LINEAR NODE COLUMNS
[10]    R←,A≠X ⍝                              REMOVE LINEAR NODE ROWS
[11]    →0 ⍝                                  EXIT
[12]   T2:R←X ⍝                              LEAVE ON QUADRATIC GLOBAL BASIS


FVIP
[0]    R←FVIP F
[1]    ⍝ CALC FIELD VARIABLES AT INTEGRATION POINTS
[2]    ⍝
[3]     R←+/(NS 2)[;;;1]×OPAX(2 3)F[ELN]


GRAVVEC
[0]    GV←GRAVVEC;R;RZ
[1]    ⍝ RETURNS GRAVITATIONAL BODY FORCE VECTOR
[2]    ⍝
[3]     R←RHO ⍝                               DENSITY
[4]     RZ←RHOZERO ⍝                          REFERENCE DENSITY
[5]     R[IEA1 SF∈1 2]←RZ ⍝                   SET SOLID AND 2⍝ NODES TO RHOZERO
[6]     R←R×1-R÷RZ ⍝                          NORMALIZED DENSITY
[7]     GV←(2 FVEB R)×OPAX(2 3)(NS 2) ⍝       MULTIPLY BY THE SHAPE FUNCTIONS
[8]     GV←-NODEV AINTGRT GV ⍝                INTEGRATE ON GLOBAL BASIS
[9]     GV←(GRZ[1]×GV),[0.5]GRZ[2]×GV ⍝       MULTIPLY BY GRAVITY


GRIDΔELN
[0]    ELN←GRIDΔELN CNQ;A
[1]    ⍝ GENERATES QUADRATIC NODE NUMBERS (ELEMENT BASIS) FOR GRIDGEN
[2]    ⍝
[3]     A←1,(¯1+3×ND)⍴1 1 0 ⍝                 EXPANSION VECTOR
[4]     ELN←A\CNQ ⍝                           EXPAND GLOBAL NODE NUMBER ON COLUMNS
[5]     ELN←A⍀ELN ⍝                           EXPAND GLOBAL NODE NUMBERS ON ROWS
[6]     ELN[;A]←ELN[;¯1+A←IEA1~A] ⍝           DUPLICATE COLUMNS
[7]     ELN[A;]←ELN[¯1+A;] ⍝                  DUPLICATE ROWS
[8]     ELN←1 3 2 4⍝(4⍴ND,3)⍴ELN ⍝            RESHAPE NODE NUMBERS
[9]     ELN←,[1 2],[3 4]ELN ⍝                 AND SEPARATE FOR EACH ELEMENT
[10]    ELN←ELN[;7 8 9 6 3 2 1 4 5] ⍝         REORDER FOR STANDARD ELEMENT
```

```
GRIDΔLNODE
[0]    LNODE←GRIDΔLNODE GNL;A;N
[1]   ⍝ GENERATES LINEAR NODE NUMBERS (ELEMENT BASIS) FOR GRIDGEN
[2]   ⍝
[3]    A←2+¯1⊖,⍉(2,N)⍴⍳N←ND+1
[4]    A←(((N,N)⍴⍳N*2)[;A])[A;]
[5]    A←((ND×2),ND,2)⍴((2×NE),2)⍴A
[6]    A←((ND*2),4)⍴1 3 2⍉(ND,4,ND)⍴1 3 2⍉A
[7]    LNODE←A[;3 4 2 1]


GRIDΔRSE
[0]    RSE←GRIDΔRSE;A;EN
[1]   ⍝ GENERATES REGION SIDE ELEMENT NUMBERS
[2]   ⍝
[3]    EN←(2⍴ND)⍴⍳ND*2 ⍝                    ELEMENT NUMBERS
[4]    A←EN[1,1+⍴EN;] ⍝                     ELEMENTS ON SIDES 3 AND 1
[5]    A←A,[1]⍉EN[;1,¯1+⍴EN] ⍝              ADD ELEMENTS ON SIDES 4 AND 2
[6]    RSE←A[2 4 1 3;] ⍝                    ELEMENT NUMBERS--EACH SIDE OF REGION


GRIDΔRSLN
[0]    RSLN←GRIDΔRSLN GNL;A
[1]   ⍝ GENERATES REGION SIDE LINEAR NODE NUMBERS FOR GRIDGEN
[2]   ⍝
[3]    A←2↑[1]¯1⊖GNL ⍝                      LINEAR NODE NUMBERS FOR SIDES 1 AND 3
[4]    A←A,[1]⍉2↑[2]¯1⌽GNL ⍝                ADD NODE NUMBERS FOR SIDES 2 AND 4
[5]    RSLN←A[1 3 2 4;] ⍝                   LINEAR NODE NUMBERS ON EACH SIDE


GRIDΔRSN
[0]    RSN←GRIDΔRSN GNQ;A
[1]   ⍝ GENERATES REGION SIDE QUADRATIC NODE NUMBERS FOR GRIDGEN
[2]   ⍝
[3]    A←2↑[1]¯1⊖GNQ ⍝                      NODE NUMBERS FOR SIDES 1 AND 3
[4]    A←A,[1]⍉2↑[2]¯1⌽GNQ ⍝                ADD NODE NUMBERS FOR SIDES 2 AND 4
[5]    RSN←A[1 3 2 4;] ⍝                    NODE NUMBERS ON EACH SIDE OF REGION


GRIDGEN
[0]    GRIDGEN;A;GNL;GNQ
[1]   ⍝ GENERATES ELEMENT NODE NUMBERING
[2]   ⍝
[3]    GNQ←(2⍴A)⍴⍳(A←1+2×ND)*2 ⍝            GLOBAL NODE NUMBERS--QUAD. ELEMENTS
[4]    GNL←(2⍴A)⍴⍳(A←1+ND)*2 ⍝              GLOBAL NODE NUMBERS--LINEAR ELEMENTS
[5]    RSN←GRIDΔRSN GNQ ⍝                   NODE NUMBERS ON EACH SIDE OF REGION
[6]    RSELMT←GRIDΔRSE ⍝                    ELEMENT NUMBERS--EACH SIDE OF REGION
[7]    RSLN←GRIDΔRSLN GNL ⍝                 LINEAR NODE NBRS.-EACH SIDE OF REGION
[8]    ELN←GRIDΔELN GNQ ⍝                   QUADRATIC NODE NUMBERS--ELEMENT BASIS
[9]    LNODE←GRIDΔLNODE GNL ⍝               LINEAR NODE NUMBERS--ELEMENT BASIS
```

```
HERON
[0]    A←HERON T;S
[1]   ⍝ CALCULATES AREAS OF TRIANGLES "T" USING HERON'S FORMULA
[2]   ⍝ LAST DIMENSION OF T CONTAINS 3 SIDE LENGTHS
[3]    S←0.5×+/T
[4]    A←(0⌈S×(S-1 ⌿ T)×(S-2 ⌿ T)×(S-3 ⌿ T))*0.5


IEA1
[0]    R←IEA1 A
[1]   ⍝ RETURNS INDICIES OF LOCATION OF ALL ONES IN VECTOR A
[2]    R←R/(⍳⍴A)×R←A=1


IFST1
[0]    R←IFST1 A
[1]   ⍝ RETURNS INDICIES OF FIRST POSITIVE CHANGE IN EACH ROW OF A
[2]    R←(,A)/,A×(⍴A)⍴⍳¯1↑⍴A←FST1 A,1


INBDY
[0]     INBDY
[1]    ⍝ SPECIFY REGION BOUNDARY CONDITIONS
[2]    ⍝ FIELD
[3]    ⍝    1  TEMPERATURE
[4]    ⍝    2  FLUID VELOCITY IN X DIRECTION
[5]    ⍝    3  FLUID VELOCITY IN Y DIRECTION
[6]    ⍝ TYPES
[7]    ⍝    1  PRESCRIBED VALUE
[8]    ⍝    2  PRESCRIBED FLUX(SIGN CONVENTION FLUX IN IS POSITIVE)
[9]     BC←5 0⍴0
[10]   ⍝  FIELD  TYPE  REGION  SIDE  PROPERTIES
[11]    BC←BC,1 1 1 1 ¯.5
[12]    BC←BC,1 1 1 3 1.5
[13]   ⍝
[14]    BC←BC,2 1 1 4 0
[15]    BC←BC,2 1 1 2 0
[16]    BC←BC,2 1 1 1 0
[17]    BC←BC,2 1 1 3 0
[18]   ⍝
[19]    BC←BC,3 1 1 4 0
[20]    BC←BC,3 1 1 2 0
[21]    BC←BC,3 1 1 1 0
[22]    BC←BC,3 1 1 3 0
[23]    BC←⍪BC
[24]   ⍝
[25]    PRNODE←2 0 ⍝                          PRESCRIBED PRESSURE NODE
```

```
INGEOM
[0]    INGEOM;A;RR;ZR
[1]   ⍝ SPECIFY GEOMETRY INPUT PARAMETERS
[2]   ⍝
[3]    INREC ⍝                          SPECIFY COORDINATES OF REGION
[4]    CF←1 ⍝                           CARTESIAN COORDINATES
[5]    GRZ←¯2E¯2 0 ⍝                    GRAVITATIONAL CONSTANTS


INITIAL
[0]    INITIAL;A
[1]   ⍝ INITIALIZE VARIABLES
[2]   ⍝
[3]    CHKINPUT ⍝                       CHECK INPUT PARAMETERS
[4]    RN←1 ⍝                           REGION NUMBER
[5]    AINTCON ANI←2 ⍝                  AREA INTEGRATION CONSTANTS
[6]    SINTCON SNI←2 ⍝                  SURFACE INTEGRATION CONSTANTS
[7]    AREA ⍝                           VOLUME AND SIDE AREAS OF ELEMENTS
[8]    TIME←TCTL[1] ⍝                   INITIALIZE TIME
[9]    INITTEMP ⍝                       INITIALIZE TEMPERATURE
[10]   INITVEL ⍝                        INITIALIZE VELOCITIES
[11]   INITPRES ⍝                       INITIALIZE PRESSURE
[12]   INITMPROP ⍝                      INITIALIZE MATERIAL PROPERTIES
[13]   EREQ←RPRES+RCOND ⍝               ENERGY RESULTANT VECTOR
[14]   A←,[1 2]GRAVVEC ⍝                GRAVITATIONAL BODY FORCE VECTORS
[15]   FREQ←A,[1]((NNG 4),1)⍴0 ⍝        FLOW RESULTANT VECTOR
[16]   SETLAST ⍝                        SET LAST TIME ITER. VARS.
[17]   SETSTAR ⍝                        SET LAST CONVERGENCE ITER. VARS.
[18]   SAVEFLDS←0⍴0 ⍝                   INITIALIZE STORAGE OF FIELD VARS.


INITMP
[0]    INITMP;A;DATA
[1]   ⍝ INITIALIZE MATERIAL PROPERTIES: CASE WHERE PRESSURE NODE PRESCRIBED
[2]   ⍝
[3]    TF←(NNG 2)⍴TINIT ⍝               INITIALIZE TEMPERATURES TO TINIT
[4]    DATA←(PROPDATA∆SEL 1 4 6)[1 4 2;;] ⍝  ASSUME INITIALLY 1⍵
[5]    A←(PF,[1.5]TF)INTQR DATA ⍝       INTERPOLATE
[6]    EF←A[1;1;] ⍝                     ENERGY FIELD
[7]    UPPROP ⍝                         UPDATE ALL PROPERTIES


INITMPROP
[0]    INITMPROP
[1]   ⍝ INITIALIZE MATERIAL PROPERTIES USING AVERAGE CONDITIONS
[2]   ⍝
[3]    →(2=⎕NC 'PRNODE')/NP ⍝           CHECK IF PRESSURE NODE PRESCRIBED
[4]    STOP ⍝                           STOP EXECUTION
[5]   NP:INITMP ⍝                       INITIALIZE PROPERTIES
```

```
INITPRES
[0]    INITPRES
[1]   ⍝ INITIALIZE PRESSURE FIELD
[2]   ⍝
[3]    →(0=⎕NC 'PFINIT')/NXT1 ⍝                    CHECK IF PFINIT EXISTS
[4]    PF←PFINIT ⍝                                 USE VALUES IN PFINIT
[5]    →0 ⍝                                        EXIT
[6]   NXT1:PF←(NNG 1)⍴PRNODE[1] ⍝                  FIRST USE REFERENCE PRESSURE
[7]    INITMP ⍝                                    CALC. DENSITY
[8]    PF←FAN STATPRES ⍝                           CALC. STATIC PRESSURES


INITTEMP
[0]    INITTEMP;A
[1]   ⍝ INITIALIZE TEMPERATURE
[2]   ⍝
[3]    TINIT←,⍉(2⍴1+A)⍴(A←2×ND)INTERVALS ¯.5 1.5 ⍝  INITIALIZE TEMPERATURE


INITVEL
[0]    INITVEL
[1]   ⍝ INITIALIZE FLUID VELOCITIES
[2]   ⍝
[3]    →(~∧/2=⎕NC 2 6⍴'UFINITVFINIT')/ZERO ⍝       CHECK FOR INITIAL VELOCITIES
[4]    UF←UFINIT ⍝                                 INITIAL U VELOCITY
[5]    VF←VFINIT ⍝                                 INITIAL V VELOCITY
[6]    →0 ⍝                                        EXIT
[7]   ZERO:UF←(NNG 2)⍴0 ⍝                          ASSUME ZERO U VELOCITY
[8]    VF←(NNG 2)⍴0 ⍝                              ASSUME ZERO V VELOCITY
```

```
INPLYGN
[0]    R←XY INPLYGN Q;AREAP;CNP;INP;MXY;NXY;NN;S;S1;S2
[1]    ⍝ FINDS IF POINTS XY ARE IN POLYGONS Q
[2]    ⍝     XY IS 2 COLUMN MATRIX AND Q ⍴ (NBR. POLYGONS) (NBR. NODES EA.) 2
[3]    ⍝     Q NODES ARE NUMBERED SEQUENTIALLY AROUND PERIMETER
[4]    ⍝     R RETURNS NBR. OF THE POLYGON CLOSEST TO EACH XY ⍴ (1↑⍴XY)
[5]    ⍝ APPROACH IS TO FIND AREAS OF POLYGONS BY DIVIDING INTO TRIANGLES AND
[6]    ⍝ COMPARING TO SUM OF AREAS OF TRIANGLES OF XY AND EACH POLYGON SIDE
[7]    ⍝
[8]     NN←(⍴Q)[2] ⍝                              NUMBER OF NODES PER POLYGON
[9]     INP←INP,[1.5]1⌽INP←⍳NN ⍝                  INDS. OF SUCCESSIVE NODE PAIRS
[10]    CNP←Q[;INP;] ⍝                            COORDS. OF SUCC. NODE PAIRS
[11]    MXY←MEAN 1 3 2⍉Q ⍝                        MEAN X Y COORDS. EACH PLYGN.
[12]    MXY←2 1 3⍉(NN,⍴MXY)⍴MXY ⍝                 RESHAPED FOR NBR. OF NODES
[13]    S←CNP[;;1;]DIST CNP[;;2;] ⍝               SIDE LENGTHS OF POLYGONS
[14]    S1←CNP[;;1;]DIST MXY ⍝                    LENGTHS OF 1 NODE TO MEAN PT.
[15]    S2←CNP[;;2;]DIST MXY ⍝                    LENGTHS OF 2 NODE TO MEAN PT.
[16]    AREAP←|+/HERON S,S1,[2.5]S2 ⍝              AREA OF EACH POLYGON
[17]    CNP←((1↑⍴XY),⍴CNP)⍴CNP ⍝                  RESHAPE NODE PAIRS FOR EACH XY
[18]    NXY←(1⌽⍳4)⍉(1⌽⍴CNP[;;;1;])⍴⍉XY ⍝          RESHAPE XY ALL PLYGNS., NODES
[19]    S←((1↑⍴XY),⍴S)⍴S ⍝                        RESHAPE PLYGN SIDE LENGTHS
[20]    S1←CNP[;;;1;]DIST NXY ⍝                   CALC LENGTHS OF 1 NODE TO XY
[21]    S2←CNP[;;;2;]DIST NXY ⍝                   CALC LENGTHS OF 2 NODE TO XY
[22]    R←⍉|+/HERON S,S1,[3.5]S2 ⍝                AREAS EACH PLYGN. WITH EA XY
[23]    R←((⍉R)-(1⌽⍴R)⍴AREAP)÷OPAX 2 AREAP ⍝      NORMALIZED DIFF. IN AREAS
[24]    R←IFST1(⌊/R)=OPAX 1 R ⍝                   CLOSEST PLYGN. FOR EACH XY
```

```
INPROD
[0]    D←A INPROD B;C;RA;N;CB;R;E
[1]    ⍝ CALCULATES INNER PRODUCT OF MULTIPLE MATRICIES
[2]    ⍝
[3]     →((⍴⍴A)≠(⍴⍴B))/CHK
[4]     →(~(×/(E←2↑¯2⌽⍴A)=(2↑¯2⌽⍴B)))/CHK
[5]     →((C←1↑¯1⌽⍴A)=(R←1↑¯2⌽⍴B))/NXT
[6]    CHK:'INCORRECT ⍴ IN INPROD'
[7]     STOP
[8]    NXT:RA←1↑¯2⌽⍴A
[9]     CB←1↑¯1⌽⍴B
[10]    D←(E,RA,CB)⍴+/[1](4 2 3 1⍉(CB,N,RA,C)⍴A)×3 2 1 4⍉(RA,(N←×/E),R,CB)⍴B
```

```
INPROG
[0]     INPROG
[1]    ⍝ SPECIFY PROGRAM OPERATION INPUT PARAMETERS
[2]    ⍝
[3]     CIL←6 ⍝                          CONVERGENCE ITERATION LIMIT
[4]     CPULIM←90×3600 ⍝                 LIMIT FOR CPU TIME
[5]     TIMELIM←72×3600 ⍝                TIME LIMIT
[6]     EC←1 ⍝                           ENERGY CALC. CONTROL (0-NO THERMAL CALC.)
[7]     ERR←0.05 ⍝                       ALLOWABLE FIELD CONVERGENCE ERROR
[8]     FC←1 ⍝                           FLUID FLOW CALC. CONTROL (0-NO FLOW CALC.)
[9]     FELT←2 2 2 1 ⍝                   ELEMENT TYPES FOR EACH FIELD
[10]    ICTL←0 ⍝                         ITERATION CONTROL (0-SUB 1-NEWTON-RAPHSON)
[11]    SFN←5 ⍝                          SAVE FIELDS AT EVERY SFN TIME STEPS
[12]    TCTL←0 200 2 ⍝                   TIME START, END AND INCREMENT CONTROL
[13]    THETA←0.5 ⍝                      TRANSIENT ALGORITHM CONTROL PARAMETER
[14]    TSC←0 ⍝                          TIME STEP CONTROL (0-CONSTANT, 1-VARIABLE)
[15]    SSC←1 ⍝                          STEADY STATE CONTROL


INPUT
[0]     INPUT
[1]    ⍝ SPECIFY AND PRINT OUT INPUT PARAMETERS
[2]    ⍝
[3]     INGEOM ⍝                                 GEOMETRY FACTORS
[4]     INBDY ⍝                                  BOUNDARY CONDITIONS
[5]     INPROG ⍝                                 PROGRAM OPERATION PARAMETERS
[6]     NE←ND×ND ⍝                               NUMBER OF ELEMENTS PER REGION


INREG
[0]     INREG;RR;ZR;A
[1]    ⍝ SPECIFY INITIAL R AND Z COORDINATES OF REGION
[2]    ⍝
[3]    ⍝'LARGE SQUARE BOX'
[4]     RR←.1×0 0.5 1 1 1 0.5 0 0 0.5
[5]     ZR←.1×0 0 0 0.5 1 1 1 0.5 0.5
[6]     RZR←RR,[0.5]ZR


INTERVALS
[0]     I←N INTERVALS P;B;E
[1]    ⍝ GENERATES N INTERVALS FROM P[1] TO P[2]
[2]     B←1↓P
[3]     E←1↓P
[4]     I←B+(((E-B)÷N)×¯1+⍳N+1)
```

```
INTQR
[0]    R←XY INTQR D;A;B;MAX;MIN;ND;NXY
[1]    ⍝ INTERPOLATES BY FINDING CLOSEST REGION AND USING DBL. QUAD. REGRESS.
[2]    ⍝  XY ARE X AND Y VALUES TO BE INTERPOLATED
[3]    ⍝  D ARE INTERP. DATA X, Y AND Z (MAY BE MULTIPLE Z) ⍴ ≥3 NR 8
[4]    ⍝    NR IS THE NUMBER OF QUADRATIC REGIONS
[5]    ⍝  R RETURNS INTERP. VALUES WITH 2 PARTIAL DERIV. ⍴(¯2+1↑⍴D) 3 (1↑⍴XY)
[6]    ⍝
[7]     →(0=1↑⍴XY)/NOXY ⍝                          EXIT IF NO POINTS
[8]    ⍝
[9]     (A MAX MIN)←NORMR⌹,[2 3]D ⍝                NORMALIZE REGION COORDS.
[10]    ND←(⍴D)⍴⌹A ⍝                              RESHAPE NORMALIZED COORDS.
[11]    NXY←(XY-OPAX 2(2↑MIN))÷OPAX 2(2↑MAX-MIN) ⍝ NORMALIZE XY WITH MAX MIN
[12]    A←3 1 2⌹A←ND[1 2;;] ⍝                      RESHAPE NORM. REGION X Y
[13]    A←NXY INPLYGN A ⍝                          FIND CLOSEST REGION
[14]    R←ND INTQRΔREG NXY,A ⍝                     DO REGRESSION ANALYSIS
[15]    R←INTQRΔUNN(R MAX MIN) ⍝                   UNNORMALIZED THE RESULT
[16]    →0 ⍝                                       EXIT
[17]   ⍝
[18]   NOXY:R←((¯2+1↑⍴D),3,0)⍴0 ⍝                  RETURN IF NO VALUES
```

```
INTQRΔREG
[0]    R←D INTQRΔREG XY;A;B;E;I;NDV;RI;UI
[1]    ⍝ CALLED BY INTQR, IT PERFORMS DOUBLE QUADRATIC REGRESSION BY REGION
[2]    ⍝  XY IS A 3 COLUMN MATRIX OF X AND Y VALUES AND REGION NUMBERS
[3]    ⍝  D IS REGION X Y Z DATA⍴ (≥3) (NUMBER OF REGIONS) 8
[4]    ⍝  R IS INTERP. VALUES AND 2 PART. DERIV. ⍴ (¯2+¯1+⍴⊃D[2]) 3 (1+⍴XY)
[5]    ⍝
[6]     RI←XY[;3] ⍝                          REGION INDICIES
[7]     XY←XY[;1 2] ⍝                        X AND Y DATA
[8]     E←3 1 2⌹D ⍝                          TRANSPOSE REGION DATA
[9]     UI←REMDUPEL RI ⍝                     UNIQUE REGION INDICIES
[10]    NDV←(¯2+¯1+⍴E) ⍝                     NUMBER OF DEPENDENT VARIABLES
[11]    R←(NDV,3,1↑⍴XY)⍴0 ⍝                  INITIALIZE RETURN VARIABLE
[12]    A←(NDV,⍴A)⍴A←((2↑⍴E),2)↑E ⍝          RESHAPE INTERP. DATA (INDEP. VARS.)
[13]    E←A,[4]2 3 1⌹((2↑⍴E),-NDV)↑E ⍝       ADD BACK AGAIN THE DEPEND. VARS.
[14]   ⍝                                     E HAS ⍴ NDV (NBR. REGIONS) 8 3
[15]   ⍝
[16]   LOOP:→(0=⍴UI)/0 ⍝                     LOOP ON EACH REGION OF INTEREST
[17]    A←⊂[2 3]E[;UI[1];;] ⍝                NESTED ARRAY OF REGION DATA
[18]    I←IEA1 UI[1]=RI ⍝                    INDICIES OF XY WITHIN THIS REGION
[19]    B←⊂[2 3]((⍴A),⍴B)⍴B←XY[I;] ⍝         NESTED ARRAY OF XY IN THIS REGION
[20]    A←⊃A QUADRGXY⍤B ⍝                    DBL. QUAD. REGRES. ON ⍤ DEP. VAR.
[21]    B←MINMAX⌹2 0↑D[;UI[1];] ⍝            MIN AND MAX VALS. FOR REGION ⍴ 2 NDV
[22]    A[;;1]←⌹(⌹A[;;1])LIMITTO B ⍝         LIMIT INTERP. Z VALS. TO REGION Z'S
[23]    R[;;I]←1 3 2⌹A ⍝                     ASSIGN RETURN VALUES
[24]    UI←1↓UI ⍝                            DROP ONE FROM LOOP VARIABLE
[25]    →LOOP ⍝                              END OF LOOP
```

```
INTQRΔUNN
[0]     U←INTQRΔUNN A;R;MMM
[1]    ⍝ UNNORMALIZE A MATRIX "A" (NESTED ARRAY) WRT. RANGE
[2]    ⍝  A[1] NORMALIZED MATRIX BETWEEN 0 AND 1 EACH COLUMN
[3]    ⍝  A[2] MAXIMUM VALUES FOR EACH COLUMN
[4]    ⍝  A[3] MINIMUM VALUES FOR EACH COLUMN
[5]    ⍝  R RETURNS INTERP. VALUES WITH 2 PART. DERIV. ρ(DEP. VARS.) 3 (1+ρXY)
[6]    ⍝
[7]     (R MAX MIN)←A ⍝                                    MATRIX, MAX. AND MINS.
[8]     MMM←MAX-MIN ⍝                                      MAX MINUS MIN VALUES
[9]     U←(2+MIN)+OPAX(1)((2+MMM)×OPAX(1)R[;,1;]) ⍝        NORM. DEPENDENT VARS.
[10]    U←U,[2](2+MMM)×OPAX(1)(R[;,2;]+OPAX(2)MMM[1]) ⍝    DERIV. WRT. X
[11]    U←U,[2](2+MMM)×OPAX(1)(R[;,3;]+OPAX(2)MMM[2]) ⍝    DERIV. WRT. Y


INV
[0]     I←INV A;R
[1]    ⍝ CALCULATES INVERSE OF MULTIPLE 2×2 MATRICESρ N 2 2
[2]    ⍝
[3]     R←ρA
[4]     A←(((1+ρ(,A))↑4),4)ρ,A
[5]     I←(ρA)ρ1 0 0 1
[6]     A[;4]←A[;4]-A[;3]×A[;2]↑A[;1]
[7]     I[;3]←¯1×I[;1]×A[;3]↑A[;1]
[8]     I[;1]←(I[;1]-I[;3]×A[;2]↑A[;4])↑A[;1]
[9]     I[;2]←(I[;2]-I[;4]×A[;2]↑A[;4])↑A[;1]
[10]    I[;3]←I[;3]↑A[;4]
[11]    I[;4]←I[;4]↑A[;4]
[12]    I←RρI


JACCHK
[0]    JACCHK A
[1]   ⍝ CHECK IF DETERMINANT OF JACOBIAN HAS A SIGN REVERSAL
[2]    →(0≥(⌊/,A)×⌈/,A)/WARN
[3]    →0
[4]   WARN:'WARNING!!! DETERMINANT OF JACOBIAN HAS A SIGN REVERSAL'
```

```
JACOB
[0]    JAC←RZ JACOB S;RS;DX;DE;RN;ZN;A
[1]    ⍝ CALC. THE JACOBIAN MATRICES FOR ALL ELEMENTS
[2]    ⍝   S ARE THE SHAPE FUNCTIONS
[3]    ⍝   RZ ARE THE COORDINATES OF THE ELEMENTS ⍴2 NE (  NODES PER ELEMENT)
[4]    ⍝
[5]     RS←(⍴S)[2]
[6]     DX←2 1 3⍉(NE,¯2+⍴S)⍴S[2;;]
[7]     DE←2 1 3⍉(NE,¯2+⍴S)⍴S[3;;]
[8]     RN←(RS,¯2+⍴RZ)⍴RZ[1;;]
[9]     ZN←(RS,¯2+⍴RZ)⍴RZ[2;;]
[10]    A←((RS×NE),1)⍴+/DX×RN
[11]    A←A,((RS×NE),1)⍴+/DX×ZN
[12]    A←A,((RS×NE),1)⍴+/DE×RN
[13]    A←A,((RS×NE),1)⍴+/DE×ZN
[14]    JAC←(RS,NE,2,2)⍴A


LIMITTO
[0]    R←Y LIMITTO X
[1]    ⍝ LIMIT Y TO RANGE BETWEEN DEFINED BY X
[2]    ⍝  THE FIRST DIMENSION OF X MUST BE ⍴2 THE MIN AND MAX VALUES
[3]    ⍝  R RETURNS MODIFIED Y  ⍴ (⍴Y)
[4]    ⍝
[5]     X←MINMAX X ⍝                              MAKE SURE MIN VALUES ARE FIRST
[6]     Y←Y⌈OPAX(⍴⍴Y)(,(1,1+⍴X)⍴X) ⍝              SET MIN VALUES
[7]     R←Y⌊OPAX(⍴⍴Y)(,(¯1,1+⍴X)⍴X) ⍝             SET MAX VALUES


LINFV
[0]    R←LINFV F;SHP;A
[1]    ⍝ CONVERTS LINEAR FIELD VARIABLE TO VALUES AT ALL NODES
[2]    ⍝   F IS LINEAR FIELD VARIABLE ⍴ (ND+1)*2
[3]    ⍝   R IS NODAL VALUES (GLOBAL BASIS)
[4]    ⍝
[5]     R←(NNG 2)⍴0 ⍝                             INITIALIZE RETURN VARIABLE
[6]     A←2 5⍴0 1 0 ¯1 0 ¯1 0 1 0 0 ⍝             MID NODES XI ETA COORDS.
[7]     A←(LSHAPE A)[1;;] ⍝                       SHAPE FUNCTIONS FOR MID NODES
[8]     SHP←((NE),⍴A)⍴A ⍝                         RESHAPE FOR ALL ELEMENTS
[9]     A←+/SHP×2 3 1⍉(1⌽⍴SHP)⍴⍉F←F[LNODE] ⍝      MID NODE VALUES
[10]    A←F,A ⍝ ⍝                                 COMBINE WITH LIN. NODE VALUES
[11]    A←A[;1 5 2 6 3 7 4 8 9] ⍝                 REORDER ELEMENT BASIS
[12]    R[,ELN]←,A ⍝                              RETURN VARIABLE
```

```
LSHAPE
[0]    SHP←LSHAPE XE;C;ETA;XI
[1]   ⍝ CALCULATES LINEAR SHAPE FUNCTIONS AND THEIR DERIVATIVES
[2]   ⍝
[3]    XI←XE[1;] ⍝                                  XI COORDINATES
[4]    ETA←XE[2;] ⍝                                 ETA COORDINATES
[5]    C←1,ETA,XI,[1.5]ETA×XI ⍝                     POLYNOMIAL COEFFICIENTS
[6]    C←1 3 2 4⍴⍉(3,4,⍴C)⍴C ⍝                      RESHAPE TO MATCH CSL
[7]    SHP←+/C×2 1 3 4⍴((1↑⍴XI),⍴CSL)⍴CSL ⍝         CALCULATE SHAPE FUNCTIONS


LUMP
[0]    R←LUMP C
[1]   ⍝ LUMPS THE CAPACITANCE MATRIX BY ROWWISE SUMMATION
[2]   ⍝
[3]    C←+⌿C ⍝                                  ROWWISE SUMMATION
[4]    R←DIAG C ⍝                               USE SUM TERMS FOR DIAGONALS


LX
[0]     LX;A
[1]    ⍝ LATENT EXPRESSION
[2]    ⍝
[3]     STACK←'CMS(192' ⍝                       INITIALIZE STACK VARIABLE
[4]     A←100 ⎕SVO ' ' ⍝                        SHARE SYSTEM VARIABLE
[5]     A←101 ⎕SVO '      ' ⍝                   SHARE STACK VARIABLE
[6]     STACK←')EDITOR 2' ⍝                     USE EDITOR 2
[7]     CLS ⍝                                   CLEAR THE SCREEN
[8]     STRIPE ⍝                                DISPLAY A STRIPE ON SCREEN
[9]     SKIP 1 ⍝                                SKIP A LINE
[10]    'PHASTRAN' ⍝                            HEADER
[11]    SKIP 1 ⍝                                SKIP A LINE
[12]    WSID←'PHASTRAN' ⍝                       WORKSPACE NAME


MAP
[0]     MAP;C;ETA;X;SHP;XI;Y
[1]    ⍝ MAP XI-ETA COORDINATES OF NODES INTO X-Y SYSTEM
[2]    ⍝
[3]     XI←XEN[1;]⍝                                    XI COORDINATES
[4]     ETA←XEN[2;] ⍝                                  ETA COORDINATES
[5]     C←(ETA×XI*2),(XI×ETA*2),[1.5](XI*2)×ETA*2 ⍝    BUILD MATRIX COLUMNS OF
[6]     C←1,XI,ETA,(XI×ETA),(XI*2),(ETA*2),C ⍝         .XI-ETA POLYNOMIAL
[7]     SHP←+/C×OPAX(1 3)(((1↑⍴C),⍴CSQLG)⍴CSQLG) ⍝     SHAPE FUNCTIONS
[8]     X←+/SHP×OPAX(2)RZR[1;] ⍝                       X COORDINATES
[9]     Y←+/SHP×OPAX(2)RZR[2;] ⍝                       Y COORDINATES
[10]    RZN←X,[0.5]Y ⍝                                 COORDS. OF GLOBAL NODES
[11]    RZE←X[ELN],[0.5]Y[ELN] ⍝                       COORDS. OF ELEMENT NODES
```

```
MDET
[0]    A←MDET B
[1]  ⍝ CALCULATES DETERMINANT OF MULTIPLE 2×2 MATRICES⍴ N 2 2
[2]  ⍝
[3]    A←(((1↑⍴A)↑4),4)⍴A←,B
[4]    A←(A[;1]×A[;4])-A[;2]×A[;3]
[5]    A←(2↑((⍴⍴B)-2)↓⍴B)⍴A


MEAN
[0]    M←MEAN A
[1]  ⍝ FINDS MEAN OVER LAST INDEX OF A VECTOR
[2]    M←(+⌿1↑⍴A)×+/A


MINMAX
[0]    R←MINMAX X
[1]  ⍝ RETURNS THE MINIMUM AND MAXIMUM VALUES OF X
[2]    R←(⌊⌿X),[0.5]⌈⌿X


NNC
[0]    N←NNC FT;NLN;NQN
[1]  ⍝ RETURNS THE TOTAL NUMBER OF NODES (GLOBAL) FOR EACH FIELD TYPE
[2]  ⍝
[3]    NLN←(ND+1)*2 ⍝                          NUMBER OF LINEAR NODES
[4]    NQN←(1+2×ND)*2 ⍝                        NUMBER OF QUADRATIC NODES
[5]    N←(NLN,NQN)[FELT[FT]] ⍝                 RETURN VARIABLE
```

```
NODEM
[0]     B←NODEM A;B1;B2;C;D;R;RNODE;CNODE
[1]    ⍝ ASSEMBLE NODEL MATRICIES FROM ELEMENT MATRICIES
[2]    ⍝
[3]     R←1↑1↓⍴A ⍝                                  NUMBER OF ROWS
[4]     C←¯1↑1↓⍴A ⍝                                 NUMBER OF COLUMNS
[5]     →(R=4)/RL ⍝                                 CHECK IF LINEAR ROWS
[6]     RNODE←ELN ⍝                                 USE QUADRATIC NUMBERING
[7]     B1←NNG 2 ⍝                                  NUMBER OF QUADRATIC NODES
[8]     →CTYPE ⍝                                    JUMP AND CHECK COLUMN TYPE
[9]    RL:RNODE←LNODE ⍝                             USE LINEAR NUMBERING
[10]    B1←(ND+1)*2 ⍝                               NUMBER OF LINEAR NODES
[11]   CTYPE:→(C=4)/CL ⍝                            CHECK IF QUADRATIC COLUMNS
[12]    CNODE←ELN ⍝                                 USE QUADRATIC NUMBERING
[13]    B2←NNG 2 ⍝                                  NUMBER OF QUADRATIC NODES
[14]    →NXT ⍝                                      JUMP
[15]   CL:CNODE←LNODE ⍝                             USE LINEAR NUMBERING
[16]    B2←(ND+1)*2 ⍝                               NUMBER OF LINEAR NODES
[17]   NXT:D←2 3 1⍴⍨(R,C,NE)⍴⍉CNODE ⍝               RESHAPE COLUMN NODE NUMBERS
[18]    D←,D+3 2 1⍴⍨(C,R,NE)⍴⍉(RNODE-1)×B2 ⍝        COMBINE WITH ROW NUMBERS
[19]    D←D REDUCE,A ⍝                              SUM WHERE MULTIPLE INDICIES
[20]    B←(×/B1,B2)⍴0 ⍝                             INITIALIZE RETURN VARIABLE
[21]    B[D[2;]]←,D[1;] ⍝                           REORDER RETURN VARIABLE
[22]    B←(B1,B2)⍴B ⍝                               RESHAPE RETURN VARIABLE
```

```
NODEV
[0]     B←NODEV A;B1;D;R;RNODE
[1]    ⍝ ASSEMBLE NODEL VECTOR FROM ELEMENT VECTORS
[2]    ⍝
[3]     R←1↑1↓⍴A ⍝                                  NUMBER OF ROWS
[4]     →(R=4)/RL ⍝                                 CHECK IF LINEAR
[5]     RNODE←ELN ⍝                                 QUADRATIC NUMBERING
[6]     B1←NNG 2 ⍝                                  NUMBER OF QUADRATIC NODES
[7]     →NXT ⍝                                      JUMP
[8]    RL:RNODE←LNODE ⍝                             LINEAR NODE NUMBERING
[9]     B1←(ND+1)*2 ⍝                               NUMBER OF LINEAR NODES
[10]   NXT:D←,RNODE ⍝                               STRING OUT NUMBERS
[11]    D←D REDUCE,A ⍝                              SUM WHERE MULTIPLE INDICIES
[12]    B←B1⍴0 ⍝                                    INITIALIZE RETURN VARIABLE
[13]    B[D[2;]]←D[1;] ⍝                            REORDER RETURN VARIABLE
[14]    B←(B1,1)⍴B ⍝                                RESHAPE RETURN VARIABLE
```

```
NORMR
[0]    R←NORMR F;MIN;MAX
[1]   ⍝ NORMALIZE MATRIX "F" WRT. RANGE (RESULT IS A NESTED ARRAY)
[2]   ⍝     R[1] NORMALIZED MATRIX BETWEEN 0 AND 1
[3]   ⍝     R[2] MAXIMUM VALUES FOR EACH COLUMN
[4]   ⍝     R[3] MINIMUM VALUES FOR EACH COLUMN
[5]   ⍝
[6]    MAX←⌈⌿F ⍝                                    MAXIMUM OF EACH COLUMN
[7]    MIN←⌊⌿F ⍝                                    MINIMUM OF EACH COLUMN
[8]    R←(F-(⍴F)⍴MIN)÷(⍴F)⍴MAX-MIN ⍝               NORMALIZE FROM 0-1
[9]    R←R MAX MIN ⍝                                RETURN NEW ARRAY, WITH MAX. AND MIN
```

```
NS
[0]   R←NS T
[1]   ⍝ RETURNS SHAPE FUNCTIONS FOR ELEMENT TYPES
[2]   ⍝    T IS THE FIELD TYPE (E.G. 1-ENERGY)
[3]   ⍝
[4]   R←⊃NST[FELT[T]] ⍝                    SELECT FROM GLOBAL NESTED ARRAY


OPAX
[0]    R←X(FN OPAX)B;AS;MIL;RT;Y
[1]    ⍝ APPLIES DIATIC PRIMATIVE OPERATOR ALONG AXES
[2]    ⍝
[3]     (AS Y)←B ⍝                        RIGHT ARGUMENT CONTAINS AXES AND Y
[4]     →((⍴⍴X)=⍴⍴Y)/APP ⍝                IF RANKS ARE SAME APPLY THE OPERATOR
[5]     →((⍴⍴Y)>⍴⍴X)/YL ⍝                 FIND LARGEST RANK
[6]     MIL←~(⍳⍴⍴X)∈AS ⍝                  MISSING INDEX LOCATIONS
[7]     →(~∧/(⍴Y)=(~MIL)/⍴X)/ERR1 ⍝       CHECK FOR RANK ERROR
[8]     RT←(MIL/⍳⍴⍴X),(~MIL)/⍳⍴⍴X ⍝       RESHAPE AND TRANSPOSE INFO.
[9]     Y←RT⍉((⍴X)[RT])⍴Y ⍝               RESHAPE Y
[10]    →APP ⍝                            JUMP TO APPLY THE OPERATOR
[11]   YL:MIL←~(⍳⍴⍴Y)∈AS ⍝                MISSING INDEX LOCATIONS
[12]    →(~∧/(⍴X)=(~MIL)/⍴Y)/ERR1 ⍝       CHECK FOR RANK ERROR
[13]    RT←(MIL/⍳⍴⍴Y),(~MIL)/⍳⍴⍴Y ⍝       RESHAPE AND TRANSPOSE INFO.
[14]    X←RT⍉((⍴Y)[RT])⍴X ⍝               RESHAPE X
[15]   APP:R←X FN Y ⍝                     PERFORM THE OPERATION
[16]    →0 ⍝                              EXIT
[17]   ERR1:⎕ES 'RANK ERROR'


PHASTRAN
[0]    PHASTRAN A
[1]   ⍝ MAIN CONTROL FUNCTION FOR PHASE CHANGE ANALYSIS MODEL
[2]   ⍝
[3]    FRONTEND A ⍝                          UPFRONT, ONCE ONLY FUNCTIONS
[4]    TIMESTEP ⍝ ⍝                          TIME INCREMENT FUNCTION
[5]    RESULTS←RESULTS ⍝                     FORMAT FINAL RESULTS
[6]    STATUS TSTOP ⍝                        DISPLAY COMPUTER USAGE


PROPDATA∆SEL
[0]    R←PROPDATA∆SEL N
[1]   ⍝ RETURNS SUBSET OF PROPDATA BASED ON STATE CONDITIONS AND PROPERTIES
[2]   ⍝    N IS NUMERIC VECTOR CORRESPONDING TO SF NOTATION
[3]   ⍝
[4]    R←PROPDATA ⍝                          ALL PROPERTY DATA
[5]    R←(∨/R[3;;]∈N)/[2]R ⍝                 ONLY REQUESTED STATES
```

```
PRSCRB
[0]     PRSCRB;A;BNP;KB
[1]   ⍝ PRESCRIBES FINITE ELEMENT EQUATIONS IN THE GLOBAL VARS. KBAR AND RBAR
[2]   ⍝  PFEQ[1;] POSITIONS IN KBAR (1 THRU 1↑⍴KBAR)
[3]   ⍝  PFEQ[2;] CORRESPONDING PRESCRIBED VALUES
[4]   ⍝  PFEQ[3;] (0 FOR SOLID NODES, 1 FOR BOUNDARY CONDITIONS)
[5]   ⍝
[6]     PFEQ←PFEQ[;⍋PFEQ[1;]] ⍝                    PUT IN ASCENDING ORDER
[7]     PFEQ←(RDUPL PFEQ[1;])/PFEQ ⍝               REMOVE DUPLS. (LEAVE 1ST)
[8]     A←⍳1↑⍴KBAR ⍝                               ALL POSITIONS IN KBAR
[9]     PFEQ←(PFEQ[1;]∈A)/PFEQ ⍝                   REMOVE ANY OUT OF RANGE
[10]    BNP←~A∈PFEQ[1;] ⍝                          BOOL. POS. NOT PRSCRBD.
[11]    A←-(~BNP)/KBAR ⍝                           COLUMNS OF K'S PRESCRIBED
[12]    KB←((⍴A)⍴PFEQ[2;])×A ⍝                     K'S × PRESCRIBED VALUES
[13]    KB←+/PFEQ[3;]×[2]KB ⍝                      ZERO SOLID NODES
[14]    RBAR←(BNP/RBAR)+BNP/KB ⍝                   NEW R VECTOR
[15]    KBAR←BNP/[1]BNP/KBAR ⍝                     NEW K MATRIX


PRSIDE
[0]     PRSIDE FT;B;C;D1;D2
[1]   ⍝ MODIFIES PFEQ WHICH DEFINES PRESCRIBED BOUNDARY CONDTIONS
[2]   ⍝
[3]     B←BC[;3 2 1 4 5] ⍝                         REORDER BC (REG. TYPE FT SIDE VAL.)
[4]     B←RN FSTCM B ⍝                             ONLY THIS REGION
[5]     B←1 FSTCM B ⍝                              ONLY PRESCRIBED BC'S
[6]     ((⍴FT)⍴⊂B)PRSIDE∆FLD⍤⍤FT ⍝                 PRESCRIBE BC'S FOR EACH FIELD TYPE
[7]   END:D1←PFEQ[1;] ⍝                            MODIFY PFEQ TO ELIMINATE
[8]     D1←⌽(C←RDUPL D1)/D1←⌽D1 ⍝                  -DUPLICATE NODE BC'S
[9]     D2←⌽C/⌽PFEQ[2 3;] ⍝                        -LEAVING ONLY THE FIRST
[10]    PFEQ←D1,[1]D2 ⍝                            RESET GLOBAL VARIABLE PFEQ


PRSIDE∆ADJ
[0]     AN←ON PRSIDE∆ADJ TYPE;PFTS
[1]   ⍝ ADJUSTS ORIGINAL NODE POSITIONS IN A MATRIX FOR FIELD TYPE
[2]   ⍝
[3]     PFTS←(-1++/∧\TYPE≠⌽FT)↓FT ⍝                PREVIOUS FIELD TYPES
[4]     AN←ON++/NNC PFTS ⍝                         ADJUSTED NUMBERS (POSITIONS)


PRSIDE∆FLD
[0]     B PRSIDE∆FLD TYPE;N;V
[1]   ⍝ PRESCRIBES BOUNDARY CONDITIONS FOR A FIELD
[2]   ⍝
[3]     B←TYPE FSTCM B ⍝                           BOUNDARY CONDITIONS FOR TYPE
[4]     →(0=⍴B)/0 ⍝                                EXIT IF NO BC'S FOR THIS TYPE
[5]     N←TYPE SIDENODES B[;1] ⍝                   FIND NODE NUMBERS
[6]     V←,⍉(⌽⍴N)⍴⍤⍤B[;2] ⍝                        FIND ASSOCIATED VALUES
[7]     (N V)←((,N)V)PRSIDE∆TEMP TYPE ⍝            HANDLE TEMP. BC'S DIFFERENTLY
[8]     N←(,N)PRSIDE∆ADJ TYPE ⍝                    ADJUST MATRIX POSITIONING
[9]     PFEQ←PFEQ,N,[1]V,[0.5]1 ⍝                  ADD PREVIOUS PRESCRIBED VALUES
```

```
PRSIDEΔTEMP
[0]    D←NV PRSIDEΔTEMP TYPE;I;C;I1;I2;N;V
[1]   ⍝ CONVERTS TEMPERATURE BOUNDARY CONDITIONS TO ENERGY BC'S
[2]   ⍝
[3]    D←NV ⍝                                    INITIALIZE RETURN VARIABLE
[4]    →(TYPE≠1)/0 ⍝                             EXIT IF NOT TEMPERATURE BC
[5]    (N V)←NV ⍝                                NODES AND VALUES
[6]    D←N,[0.5](⍴N)⍴0 ⍝                        SETUP RETURN VARIABLE
[7]    I1←IEA1 I←(1 FVGB SF)[N]∈1 4 6 ⍝          IND. OF NODES IN SINGLE PHASE
[8]    I←(PROPDATAΔSEL 1 4 6)[4 1 2;;] ⍝         SINGLE-PHASE PROPERTY DATA
[9]    C←V[I1],[1.5](1 FVGB PF)[N[I1]] ⍝         USE TEMPERATURE AND PRESSURE
[10]   D[2;I1]←(C INTQR I)[1;1;] ⍝               INTERPOLATE SINGLE-PHASE POINTS
[11]   D←⊂[2]D ⍝                                 RETURN NESTED ARRAY
```

```
PRSPRES
[0]    PRSPRES
[1]   ⍝ PRESCRIBES PRESSURE NODEM
[2]   ⍝
[3]    →(2=⎕NC 'PRNODE')/NP ⍝                    CHECK IF PRESSURE NODE PRESCRIBED
[4]   ⍝
[5]    PRSP1 ⍝                                   PRESCRIBE PRES. (TOTAL MASS CONSTRAINED)
[6]    →0
[7]   NP:PRSP2 ⍝                                 PRESCRIBED PRESSURE NODE (OPEN SYSTEM)
```

```
PRSP1
[0]    PRSP1;A;B;NN
[1]   ⍝ PRESCRIBES PRESSURE NODEM: CASE WHERE TOTAL MASS CONSTRAINED
[2]   ⍝    USES PRESSURE VALUE FROM LAST ITERATION
[3]   ⍝
[4]    NN←NNC 2 ⍝                                NUMBER OF QUAD. NODES (GLOBAL BASIS)
[5]    A←NN⍴((1+2×ND)⍴1 0),(ND+1)⍴0 ⍝            BOOL. WHERE PRES. NODES OCCUR
[6]    B←∨⌿(2,NN)⍴(NN+⍳2×NN)∈PFEQ[1;] ⍝         BOOL. WHERE VELOCITIES PRESCRIBED
[7]    A←(A≠0)/A←(~A×B)×A\⍳(ND+1)*2 ⍝            PRES. NODES WHERE VEL. NOT PRSCRBED.
[8]    B←PF[1+A] ⍝                               USE 1ST UNPRSCRBED. PRES. NODE
[9]    PFEQ←PFEQ,3 1⍴((1+A)+3×NN),B,1 ⍝          PRESCRIBE THE PRESSURE NODE
```

```
PRSP2
[0]    PRSP2;N
[1]   ⍝ PRESCRIBES PRESSURE NODEM: CASE OF OPEN SYSTEM
[2]   ⍝
[3]    N←(+/NNC 2 3)+PRNODE[1] ⍝                 MATRIX LOCATION OF NODE NUMBER
[4]    PFEQ←PFEQ,3 1⍴N,PRNODE[2],1 ⍝             PRESCRIBE THE PRESSURE NODE
```

```
PRSVLC
[0]    CE←PRSVLC FT
[1]  ⍝ RETURNS LOCATIONS OF PRESCRIBED VALUES FOR THESE FIELD TYPES
[2]  ⍝
[3]    CE←(+/NNC FT)⍴1 ⍝                          VECTOR OF 1'S FOR ALL NODES
[4]    CE[(PFEQ[1;])]←0 ⍝                         INSERT 0'S WHERE PRESCRIBED


QLGSHP
[0]    SHP←QLGSHP XE;A;C;ETA;XI
[1]  ⍝ CALCS. QUADRATIC LAGRANGIAN SHAPE FUNCTIONS AND DERIVATIVES
[2]  ⍝
[3]    XI←XE[1;] ⍝                                XI COORDINATES
[4]    ETA←XE[2;] ⍝                               ETA COORDINATES
[5]    C←(ETA×XI*2),(XI×ETA*2),[1.5](XI×ETA)*2 ⍝  BUILD MATRIX OF POLYNOMIAL
[6]    C←1,XI,ETA,(XI×ETA),(XI*2),(ETA*2),C ⍝     .COEFFICIENTS
[7]    SHP←C+.×⌹CSQLG ⍝                           CALCULATE SHAPE FUNCTIONS
[8]    A←CSQLG×OPAX 2(0 1 0 1 2 0 2 1 2) ⍝        COEF. FOR DERIV. WRT XI
[9]    A←A[;2 5 4 7 1 8 3 9 6] ⍝                  REORDER
[10]   SHP←SHP,[0.5]C+.×⌹A ⍝                      DERIV. WRT XI
[11]   A←CSQLG×OPAX 2(0 0 1 1 0 2 1 2 2) ⍝        COEF. FOR DERIV. WRT ETA
[12]   A←A[;3 4 6 8 7 1 9 2 5] ⍝                  REORDER
[13]   SHP←SHP,[1]C+.×⌹A ⍝                        DERIV. WRT ETA


QUADΔPLY
[0]    P←QUADΔPLY XY;X;Y
[1]  ⍝ FORMS POLYNOMIAL FOR 2 DIMENSIONAL QUADRATIC
[2]  ⍝   XY IS A 2 COLUMN MATRIX OF X Y VALUES
[3]  ⍝
[4]    X←XY[;1] ⍝                                                  X VALUES
[5]    Y←XY[;2] ⍝                                                  Y VALUES
[6]    P←1,X,(X*2),Y,(Y*2),(X×Y),(Y×X*2),[1.5](X×Y*2) ⍝           FORM POLYNOMIAL


QUADRGXY
[0]    Z←D QUADRGXY XY;C;K;DX;DY;PXY
[1]  ⍝ PERFORMS QUADRATIC REGRESSION ANALYSIS IN 3 DIMENSIONS X Y Z
[2]  ⍝   XY IS A 2 COLUMN MATRIX OF X AND Y VALUES TO BE INTERPOLATED
[3]  ⍝   D IS A 3 COLUMN MATRIX OF X Y Z DATA USED IN THE INTERPOLATION
[4]  ⍝   Z RETURNS INTERP. VALUES AND THEIR DERIVATIVES ⍴ (1↑⍴XY) 3
[5]  ⍝
[6]    K←QUADΔPLY D[;1 2] ⍝                        POLYNOMIAL OF INTERP. DATA
[7]    K←D[;3]⌹K ⍝                                 REGRESSION ON DATA
[8]    PXY←QUADΔPLY XY ⍝                           POLY. OF XY'S TO BE INTERP.
[9]    Z←PXY+.×K ⍝                                 INTERPOLATED VALUES
[10]   C←1 2 0 1 1 2 0 0×K[2 3 1 6 8 7 1 1] ⍝      COEF. FOR DERIV. WRT. X
[11]   DX←PXY+.×C ⍝                                DERIV. OF Z WRT. X AT XY
[12]   Z←Z,[1.5]DX ⍝                               CATENATE TO RETURN VARIABLE
[13]   C←1 1 1 2 0 2 0 0×K[4 6 7 5 1 8 1 1] ⍝      COEF. FOR DERIV. WRT. Y
[14]   DY←PXY+.×C ⍝                                DERIV. OF Z WRT. Y AT XY
[15]   Z←Z,DY ⍝                                    CATENATE TO RETURN VARIABLE
```

```
RCOND
[0]    RC←RCOND
[1]   ⍝ RETURNS SUB-VECTOR FOR INTERNAL CONDUCTION
[2]   ⍝
[3]    RC←+/[3]((DX 1),DY 1)×OPAX(2 3)(1 FVEB TF) ⍝  DERIVATIVES OF TEMP.
[4]    RC←RC×OPAX(1 2)(FVIP KT) ⍝                    MULT. BY CONDUCTIVITY
[5]    RC←+/RC×OPAX(1 2 4)((DX 1),DY 1) ⍝            MULT. BY DERIV. SHP. FNS.
[6]    RC←((⍴RC),1)⍴RC ⍝                             RESHAPE TO COLUMN VECTOR
[7]    RC←-NODEV AINTGRT RC ⍝                        INTGRT. GLOBAL NODE BASIS


RDUPL
[0]    R←RDUPL X
[1]   ⍝ RETURNS A BOOLEAN FOR REDUCING DUPLICATE VALUES LEAVING ONLY THE 1ST
[2]   ⍝
[3]    R←1 1⍲<\X∘.=X


REDUCE
[0]     X←IND REDUCE VAL;C;D;C2;C3;I
[1]    ⍝ REDUCES (BY SUMMATION) A VECTOR WITH MULTIPLE INDICIES
[2]    ⍝
[3]     IND←IND[I←⍋IND] ⍝                    REORDER INDICIES SEQUENTIALLY
[4]     VAL←VAL[I] ⍝                         AND THE CORRESPONDING VALUES
[5]     C2←IND≠1⌽IND ⍝                       LOCATIONS OF DUPLICATES
[6]     →(1≠×/C2)/NXT ⍝                      JUMP IF DUPLICATE INDICIES EXIST
[7]     X←VAL,[0.5]IND ⍝                     RETURN VARIABLE
[8]     →0 ⍝                                 EXIT
[9]    NXT:
[10]    C3←C2+(~C2)×((⍴C2)-1)⌽~C2 ⍝          BOOLEAN WITH 0'S AT FIRST DUPL.
[11]    D←C3/IND×C3 ⍝                        REDUCED SET OF INDICIES
[12]    C←VAL+1⌽VAL×(¯1+⍴VAL)⌽C3≠1 ⍝         SUM VALUES
[13]    C←((¯1+⍴VAL)⌽C3)/C ⍝                 ELIMINATE THOSE VALUES
[14]    X←D REDUCE C ⍝                       RECURSION TO FURTHER REDUCE


REMDUPEL
[0]    R←REMDUPEL X
[1]   ⍝ REMOVES DUPLICATE ELEMENTS OF X
[2]   ⍝
[3]    R←((X⍳X)=⍳⍴X)/X


RESULTS
[0]    R←RESULTS;HDR
[1]   ⍝ DISPLAYS FIELD VARIABLES
[2]   ⍝
[3]    R←PF,EF,TF,SF,UF,[1.5]VF
[4]    HDR←'PRESSURE' 'ENERGY' 'TEMPERATURE' 'STATE' 'U-VELOCITY' 'V-VELOCITY'
[5]    R←⍪HDR,[1]R
```

```
RHOZERO
[0]    R←RHOZERO;DATA;IP
[1]    ⍝ CALCULATES THE REFERENCE DENSITY
[2]    ⍝
[3]    DATA←(PROPDATA∆SEL 4)[1 4 5;;] ⍝          USE LIQUID DATA
[4]    IP←1 2⍴(MEAN PF),MEAN TF ⍝                USE MEAN PRES. AND TEMP.
[5]    R←1↑,IP INTQR DATA ⍝                      INTERPOLATE ON DENSITY


RPRES
[0]    RP←RPRES
[1]    ⍝ FORMS PRESSURE RESULTANT VECTOR
[2]    ⍝
[3]    RP←PF[ELN]×(DFN UF)[1;;]+(DFN VF)[2;;]
[4]    RP←-NODEV AINTGRT(NS 1)×[2 3]RP


SAVEFIELDS
[0]    NTS SAVEFIELDS SFN
[1]    ⍝ SAVES FIELDS AT EVERY DFN TIME STEP, PUTS IN NESTED ARRAY SAVEFLDS
[2]    ⍝
[3]    →(0≠SFN|NTS)/0 ⍝                                          EXIT IF NOT
[4]    SAVEFLDS←SAVEFLDS,⊂TIME(PF,EF,TF,SF,UF,[1.5]VF) ⍝         SAVE FIELDS


SETLAST
[0]    SETLAST
[1]    ⍝ SETS VARIABLES FROM LAST TIME ITERATION
[2]    ⍝
[3]    RELAST←EREQ ⍝                          LAST ENERGY RESULTANT
[4]    RFLAST←FREQ ⍝                          LAST FLOW RESULTANT
[5]    EPHI←EF ⍝                              LAST ENERGY SOLUTION
[6]    FPHI←UF,VF,4 FVCB PF ⍝                 LAST FLOW SOLUTION


SETSOLID
[0]    SETSOLID;A;N;NSOL;P
[1]    ⍝ SETS VELOCITIES TO ZERO FOR NODES IN SOLID STATE
[2]    ⍝
[3]    NSOL←IEA1~SF∈4 ⍝                       FIND NON-LIQUID NODES
[4]    UF[NSOL]←0 ⍝                           SET U VELOCITY TO ZERO
[5]    VF[NSOL]←0 ⍝                           SET V VELOCITY TO ZERO
```

```
SETSTAR
[0]    SETSTAR
[1]   ⍝ SETS EF, UF, VF AND PF AT LAST CONVERGENCE ITERATION
[2]   ⍝
[3]    EFSTAR←EF ⍝                                  LAST ENERGY VALUES
[4]    →(FC=0)/0 ⍝                                  CHECK IF FLOW CALCS. WERE SOLVED
[5]    UFSTAR←UF ⍝                                  LAST U VELOCITIES
[6]    VFSTAR←VF ⍝                                  LAST V VELOCITIES
[7]    PFSTAR←PF ⍝                                  LAST PRESSURES


SIDENODES
[0]    R←FT SIDENODES S;ET
[1]   ⍝ RETURNS THE REGION SIDE NODES FOR A FIELD TYPE
[2]   ⍝    S IS THE SIDE NUMBERS
[3]   ⍝    FT IS THE TYPE OF FIELD (E.G. 1-ENERGY)
[4]   ⍝
[5]    ET←FELT[FT] ⍝                                CHANGE FIELD TYPE TO ELEMENT TYPE
[6]    →(ET=1 2)/T1,T2 ⍝                            CHECK TYPE OF ELEMENT
[7]    →0 ⍝                                         EXIT IF NOT VALID ELEMENT TYPE
[8]   T1:R←RSLN[S;] ⍝                               LINEAR SIDE NODES
[9]    →0 ⍝                                         EXIT
[10]  T2:R←RSN[S;] ⍝                               QUADRATIC SIDE NODES


SIFAC
[0]    SIF←SIFAC;DX;DE;XN;YN;A;B;C;D;S1;S2;S3;S4
[1]   ⍝ CALCULATES THE SIDE INTEGRATION FACTOR FOR USE WITH SINTGRT
[2]   ⍝
[3]    DX←2 1 3⍉(NE,⍴QSS[2;;])⍴QSS[2;;] ⍝           DERIV. SHP. FNS. WRT. XI
[4]    DE←2 1 3⍉(NE,⍴QSS[3;;])⍴QSS[3;;] ⍝           DERIV. SHP. FNS. WRT. ETA
[5]    XN←((⍴QSS)[2],¯2+⍴RZE)⍴RZE[1;;] ⍝             X COORDS. OF NODES
[6]    YN←((⍴QSS)[2],¯2+⍴RZE)⍴RZE[2;;] ⍝             Y COORDS. OF NODES
[7]    A←(4,((⍴QSS)[2]+4),NE)⍴+/DX×XN ⍝              DERIVATIVES × COORDINATES
[8]    B←(4,((⍴QSS)[2]+4),NE)⍴+/DX×YN ⍝              DERIVATIVES × COORDINATES
[9]    C←(4,((⍴QSS)[2]+4),NE)⍴+/DE×XN ⍝              DERIVATIVES × COORDINATES
[10]   D←(4,((⍴QSS)[2]+4),NE)⍴+/DE×YN ⍝              DERIVATIVES × COORDINATES
[11]   S1←,⍉((A[1;;]*2)+B[1;;]*2)*0.5 ⍝              SIDE 1
[12]   S2←,⍉((C[2;;]*2)+D[2;;]*2)*0.5 ⍝              SIDE 2
[13]   S3←,⍉((A[3;;]*2)+B[3;;]*2)*0.5 ⍝              SIDE 3
[14]   S4←,⍉((C[4;;]*2)+D[4;;]*2)*0.5 ⍝              SIDE 4
[15]   A←2 1 3⍉(NE,⍴QSS[1;;])⍴QSS[1;;] ⍝             INCLUDE THE SHAPE FUNCTION
[16]   A←(4,(SNI+1),NE)⍴+/A ⍝                        .AS PART OF THE ...
[17]   SIF←A×(⍴A)⍴⍉S1,S2,S3,[1.5]S4 ⍝               SIDE INTEGRATION FACTOR
```

```
SINTCON
[0]    SINTCON N;A;N1
[1]   ⍝ INTEGRATION CONSTANTS FOR USE WITH SINTGRT
[2]   ⍝
[3]    N1←1+SNI←N ⍝                                  ORDER OF GAUSS-LEGENDRE
[4]    A←N1↑(CI[1;;])[SNI;] ⍝                        INTEGRATION POINT CONSTANTS
[5]    SXEI←A,(N1⍴1),A,(N1⍴¯1) ⍝                     XI COORDINATES
[6]    SXEI←SXEI,[0.5](N1⍴¯1),A,(N1⍴1),A ⍝           ADD ETA COORDINATES
[7]    QSS←QLGSHP SXEI ⍝                             QUADRATIC SHAPE FUNCTIONS
[8]    LSS←LSHAPE SXEI ⍝                             LINEAR SHAPE FUNCTIONS
[9]    QSIF←SIFAC ⍝                                  SIDE INTEGRATION FACTOR


SINTGRT
[0]     I←SINTGRT A;N1;WE
[1]    ⍝ INTEGRATES FUNCTION OVER SIDES OF ELEMENT IN XI-ETA COOR. SYSTEM
[2]    ⍝
[3]     N1←SNI+1 ⍝                                   1+ORDER OF INTEGRATION
[4]     →(1∈⍴,A)/SCALAR ⍝                            CHECK IF A IS SCALAR
[5]     →(∧/(⍴QSIF)=3↑⍴A)/NXT ⍝                      CHECK IF ⍴A IS LIKE ⍴QAIF
[6]     STATUS ERRMSG[2] ⍝                           MESSAGE  TO USER
[7]     →0 ⍝                                         EXIT
[8]    NXT:A←A×(¯1⍳⍴⍴A)⍉(¯1⍳⍴A)⍴QSIF ⍝              MULTIPLY BY INTGRT. FACTOR
[9]     →END ⍝                                       JUMP TO FINISH INTEGRATION
[10]   SCALAR:A←A×QSIF ⍝                            MULTIPLY BY INTGRT. FACTOR
[11]   END:WE←(2⍳⍴⍴A)⍉(2⍳⍴A)⍴N1↑(CI[2;;])[SNI;] ⍝  WEIGHTING FACTORS
[12]    I←+/[1]+/[1]WE×A ⍝                           INTEGRATE ALL SIDES


SKIP
[0]    S←SKIP N
[1]   ⍝ PRODUCES "N" BLANK ROWS
[2]    S←(N,1)⍴' '


STATPRES
[0]    SPF←STATPRES;A;NXN;XYC
[1]   ⍝ CALCULATES THE STATIC PRESSURE DISTRIBUTION
[2]   ⍝
[3]    NXN←(NS 4)INPROD 1 2 4 3⍉NS 4 ⍝               PRODUCT OF SHAPE FUNCTIONS
[4]    XYC←RZE[;;1 3 5 7] ⍝                         X Y COORDS. OF CORNER NODES
[5]    A←+/[1]GRZ×[1]((⍴XYC)⍴4 FVEB RHO)×XYC ⍝       DENSITY×GRAVITY×LOCATION
[6]    A←NXN INPROD((3↑⍴NXN),1)⍴A ⍝                 MULTIPLY BY SHAPE FUNCTIONS
[7]    A←NODEV AINTGRT A ⍝                          INTEGRATE ON GLOBAL BASIS
[8]    SPF←,A⌹NODEM AINTGRT NXN ⍝                   SOLVE FOR PRESSURE
[9]    SPF←SPF+PRNODE[2]-SPF[1↑PRNODE] ⍝            ADD REFERENCE PRESSURE
```

```
STATUS
[0]    STATUS MSG
[1]   ⍝ MANAGES STATUS MESSAGES
[2]   ⍝
[3]    MSG←⍕MSG ⍝                          FORMAT MESSAGE
[4]    STATMSG←STATMSG ADDROWS MSG ⍝       RECORD IN STATMSG
[5]    MSG ⍝                               DISPLAY THE MESSAGE AT TERMINAL


STRIPE
[0]    S←STRIPE
[1]   ⍝ CREATES STRIPE LINE BORDER
[2]    S←79⍴⎕AV[145]


TIMECHK
[0]    CK←TIMECHK
[1]   ⍝ RETURNS A 0 IF TIME LIMIT IS EXCEEDED
[2]   ⍝
[3]    CK←~(0.001×⎕AI[3]-T1[3])≥TIMELIM ⍝          CHECK RUN TIME
[4]    →(CK=1)/0 ⍝                                 EXIT IF OK
[5]    STATUS 'RUN TIME LIMIT EXCEEDED' ⍝          MESSAGE TO USER
[6]   ⍝                                            TO SET SΔTIMECHK


TIMESTEP
[0]    TIMESTEP;CIC;NIT;NTS
[1]   ⍝ TIME STEPPING FUNCTION
[2]   ⍝
[3]    NTS←0 ⍝                          INITIALIZE TIME STEP COUNTER
[4]    CIC←10⍴0 ⍝                       INITIALIZE CONVERGENCE ITER. COUNTER
[5]   LOOP:TIME←TIME+TCTL[3] ⍝          LOOP ON TIME
[6]    NTS←NTS+1 ⍝                      INCREMENT TIME STEP COUNTER
[7]    STATUS 'TIME'(¯1+TIME) ⍝         DISPLAY TIME
[8]    →(TCTL[2]<¯1+TIME)/0 ⍝           EXIT IF AT TIME LIMIT
[9]    →(0=CPUCHK)/0 ⍝                  EXIT IF CPU TIME EXCEEDED
[10]   →(0=TIMECHK)/0 ⍝                 EXIT IF RUN TIME EXCEEDED
[11]   NIT←AGAIN 0 ⍝                    CONVERGE AND RETURN ITERATION NUMBER
[12]   SETSOLID ⍝                       SET SOLID NODE VELOCITIES TO ZERO
[13]   CIC←1↓CIC,NIT ⍝                  UPDATE CONVERGENCE ITERATION COUNTER
[14]   CIC ADJSTEP NIT ⍝                ADJUST TIME STEP BASED ON ITER. NBR.
[15]   SETLAST ⍝                        SET EF, UF, VF, PF FOR LAST TIME STEP
[16]   NTS SAVEFIELDS SFN ⍝             SAVE FIELDS AT CERTAIN TIME STEPS
[17]   →(CHKSS CIC)/0 ⍝                 EXIT IF STEADY STATE REACHED
[18]   →LOOP ⍝                          CONTINUE IN TIME
```

```
TOMATRIX
[0]    M←TOMATRIX V
[1]    ⍝ CHANGES A SCALAR, VECTOR OR ARRAY OF RANK 3 OR HIGHER TO A MATRIX
[2]    ⍝
[3]     M←V ⍝                                  INITIALIZE RETURN VARIABLE
[4]     →(2=⍴⍴V)/0 ⍝                           EXIT IF ALREADY MATRIX
[5]     →(2<⍴⍴V)/N1 ⍝                          CHECK IF RANK GREATER THAN 2
[6]     M←(1,⍴V)⍴V←,V ⍝                       CONVERT SCALAR OR VECTOR TO A MATRIX
[7]     →0 ⍝                                   EXIT
[8]    N1:M←((×/(¯1+⍴⍴V)↑⍴V),¯1↑⍴V)⍴V ⍝       CONVERT HIGHER RANK ARRAY TO A MATRIX


TSTART
[0]    TSTART
[1]    ⍝ STARTS CLOCK FOR TIME CHECKING
[2]    T1←⎕AI


TSTOP
[0]    R←TSTOP;T2
[1]    ⍝ DISPLAYS CPU, CONNECT TIMES SINCE TSTART WAS ISSUED
[2]    ⍝
[3]     T2←⎕AI ⍝                                                  CURRENT ACCOUT INFO.
[4]     SKIP 1 ⍝                                                  SKIP A LINE
[5]     R←(⍕0.001×T2[2]-T1[2]),'   SEC. CPU' ⍝                    DISPLAY CPU USAGE
[6]     R←R ADDROWS(⍕0.001×T2[3]-T1[3]),'   SEC. CT' ⍝            DISPLAY CONNECT TIME


UPPROP
[0]     UPPROP;DATA;IP;DP
[1]    ⍝ UPDATES PROPERTIES ON GLOBAL NODE BASIS
[2]    ⍝
[3]     DATA←PROPDATA⌷SEL;6 ⍝                    PROPERTY DATA
[4]     IP←PF,[1.5]EF ⍝                          INDEPENDENT PROPERTIES
[5]     DP←IP INTQR DATA ⍝                       PERFORM INTERPOLATION
[6]     SF←DP[1;1;] ⍝                            STATE
[7]     TF←DP[2;1;] ⍝                            TEMPERATURE
[8]     RHO←DP[3;1;] ⍝                           DENSITY
[9]     KT←DP[4;1;] ⍝                            CONDUCTIVITY
[10]    VIS←DP[5;1;] ⍝                           VISCOSITY
[11]    UPPROP⌷TEMP ⍝                            PRESCRIBED TEMPERATURES
```

```
UPPROPΔTEMP
[0]    UPPROPΔTEMP;B;N;T
[1]    ⍝ PLACES PRESCRIBED TEMPERATURES IN UPDATED PROPERTIES
[2]    ⍝
[3]    B←BC[;3 2 1 4 5] ⍝                    REORDER BC (REC. TYPE FT SIDE VAL.)
[4]    B←RN FSTCM B ⍝                        BC'S FOR THIS REGION
[5]    B←1 FSTCM B ⍝                         ONLY PRESCRIBED BC'S
[6]    B←1 FSTCM B ⍝                         ONLY TEMP. BC'S
[7]    →(0=1↑⍴B)/0 ⍝                         EXIT IF NO TEMP. BC'S
[8]    B←⍉⍤¯1⊢B ⍝                            PRESCRIBED TEMPERATURE CONDITIONS
[9]    N←1 SIDENODES B[;1] ⍝                 FIND NODE NUMBERS
[10]   T←⍉(⊂⍴N)⍴B[;2] ⍝                      FIND ASSOCIATED TEMP. VALUES
[11]   TF[,N]←,T ⍝                           REPLACE WITH PRESCRIBED TEMPERATURES
```

CI 2 6 7 *NUMERIC*

```
  →
++ ¯0.57735  0.57735  0         0         0         0         0
     0       ¯0.7746   0.7746    0         0         0         0
   ¯0.33998  0.33998  ¯0.86114   0.86114   0         0         0
     0       ¯0.53847  0.53847  ¯0.90618   0.90618   0         0
   ¯0.23862  0.23862  ¯0.66121   0.66121  ¯0.93247   0.93247   0
     0       ¯0.40585  0.40585  ¯0.74153   0.74153  ¯0.94911   0.94911

     1         1       0         0         0         0         0
     0.88889   0.55556 0.55556   0         0         0         0
     0.65215   0.65215 0.34785   0.34785   0         0         0
     0.56889   0.47863 0.47863   0.23693   0.23693   0         0
     0.46791   0.46791 0.36076   0.36076   0.17132   0.17132   0
     0.41796   0.38183 0.38183   0.27971   0.27971   0.12948   0.12948
   ~
```

CSL 3 4 4 *NUMERIC*

```
  →
++  0.25  ¯0.25  ¯0.25   0.25
    0.25  ¯0.25   0.25  ¯0.25
    0.25   0.25   0.25  ¯0.25
    0.25   0.25  ¯0.25  ¯0.25

   ¯0.25   0.25   0      0
    0.25  ¯0.25   0      0
    0.25   0.25   0      0
   ¯0.25  ¯0.25   0      0

   ¯0.25   0      0.25   0
   ¯0.25   0     ¯0.25   0
    0.25   0      0.25   0
    0.25   0     ¯0.25   0
   ~
```

CSQLQ 9 9 *NUMERIC*

```
0   0    0    0.25   0    0    ¯0.25  ¯0.25   0.25
0   0   ¯0.5  0      0    0.5   0.5    0      ¯0.5
0   0    0   ¯0.25   0    0    ¯0.25   0.25   0.25
0   0.5  0    0      0.5  0     0     ¯0.5    ¯0.5
0   0    0    0.25   0    0     0.25   0.25   0.25
0   0    0.5  0      0    0.5  ¯0.5    0      ¯0.5
0   0    0   ¯0.25   0    0     0.25  ¯0.25   0.25
0  ¯0.5  0    0      0.5  0     0      0.5    ¯0.5
1   0    0    0     ¯1   ¯1     0      0      1
```

```
PROPDATA      7 3 8   NUMERIC

  →
++1000      1000      1000       500         0         0         0       500
  1000      1000      1000       500         0         0         0       500
  1000      1000      1000       500         0         0         0       500

  2000      3000      4000      4000      4000      3000      2000      2000
  4000      4500      5000      5000      5000      4500      4000      4000
  5000      6000      7000      7000      7000      6000      5000      5000

     1         1         1         1         1         1         1         1
     2         2         2         2         2         2         2         2
     4         4         4         4         4         4         4         4

    ¯2        ¯1         0         0         0        ¯1        ¯2        ¯2
     0         0         0         0         0         0         0         0
     0         1         2         2         2         1         0         0

  1000      1000      1000      1000      1000      1000      1000      1000
  1000      1000      1000      1000      1000      1000      1000      1000
  1000       999       998       998       998       999      1000      1000

     1         1         1         1         1         1         1         1
     1         1         1         1         1         1         1         1
     1         1         1         1         1         1         1         1

     0.01      0.01      0.01      0.01      0.01      0.01      0.01      0.01
     0.01      0.005     0.001     0.001     0.001     0.005     0.01      0.01
     0.001     0.001     0.001     0.001     0.001     0.001     0.001     0.001
  ~
```

RESULTS 226 61 CHARACTER

PRESSURE	ENERGY	TEMPERATURE	STATE	U-VELOCITY	V-VELOCITY
0.2854	6500	1.5	4	0.000E0	0
0.1427	6500	1.5	4	0.000E0	0
0	6500	1.5	4	0.000E0	0
-0.1427	6500	1.5	4	0.000E0	0
-0.2855	6500	1.5	4	0.000E0	0
-0.4282	6500	1.5	4	0.000E0	0
-0.571	6500	1.5	4	0.000E0	0
-0.7136	6500	1.5	4	0.000E0	0
-0.8563	6500	1.5	4	0.000E0	0
-0.9989	6500	1.5	4	0.000E0	0
-1.141	6500	1.5	4	0.000E0	0
-1.284	6500	1.5	4	0.000E0	0
-1.427	6500	1.5	4	0.000E0	0
-1.569	6500	1.5	4	0.000E0	0
-1.712	6500	1.5	4	0.000E0	0
0.2854	6012	1.012	4	7.690E-5	0.00008336
0.1427	5946	0.946	4	2.046E-4	0.00008165
-0.0000731	5908	0.9075	4	3.577E-4	0.00006989
-0.1428	5896	0.8958	4	4.575E-4	0.00005228
-0.2856	5907	0.9067	4	5.411E-4	0.00002314
-0.4284	5945	0.9449	4	5.644E-4	-0.000007881
-0.5711	6000	0.9997	4	5.391E-4	-0.00003793
-0.7138	6075	1.075	4	4.795E-4	-0.00005737
-0.8565	6152	1.152	4	3.738E-4	-0.00006606
-0.9991	6238	1.238	4	2.752E-4	-0.00006041
-1.142	6312	1.312	4	1.726E-4	-0.00004949
-1.284	6376	1.376	4	9.500E-5	-0.00003728
-1.427	6418	1.418	4	2.949E-5	-0.00002886
-1.57	6439	1.439	4	0.000E0	0
-1.712	6443	1.443	4	0.000E0	0
0.2854	5632	0.6318	4	2.318E-5	0.0001787
0.1426	5545	0.545	4	1.049E-4	0.0002365
-0.0001462	5481	0.4808	4	2.209E-4	0.0002257
-0.1429	5470	0.4701	4	3.298E-4	0.0001771
-0.2857	5502	0.5017	4	4.416E-4	0.000109
-0.4285	5561	0.5606	4	5.146E-4	0.00003637
-0.5712	5637	0.6374	4	5.562E-4	-0.00004134
-0.7139	5737	0.7369	4	5.541E-4	-0.0001082
-0.8566	5852	0.8516	4	4.907E-4	-0.000156
-0.9993	5991	0.9909	4	3.931E-4	-0.0001709
-1.142	6125	1.125	4	2.625E-4	-0.0001586
-1.285	6253	1.253	4	1.459E-4	-0.000129
-1.427	6341	1.341	4	4.358E-5	-0.00009754
-1.57	6387	1.387	4	0.000E0	0
-1.712	6393	1.393	4	0.000E0	0
0.2853	5412	0.4121	4	1.603E-5	0.0002286
0.1425	5349	0.3488	4	8.098E-5	0.0003182
-0.0002494	5319	0.3193	4	1.640E-4	0.00031
-0.1431	5339	0.3395	4	2.295E-4	0.0002478
-0.2859	5391	0.3905	4	3.002E-4	0.0001634
-0.4286	5454	0.454	4	3.464E-4	0.00007213
-0.5714	5528	0.5275	4	3.814E-4	-0.00003524
-0.7141	5612	0.612	4		

-0.8568	5718	0.7179	4	3.991E-4	-0.0001379
-0.9995	5855	0.8546	4	3.816E-4	-0.0002268
-1.142	6004	1.004	4	3.355E-4	-0.0002763
-1.285	6163	1.163	4	2.390E-4	-0.0002778
-1.427	6280	1.28	4	1.381E-4	-0.0002388
-1.57	6342	1.342	4	4.294E-5	-0.0001798
-1.713	6349	1.349	4	0.000E0	0
0.2853	5297	0.2969	4	0.000E0	0
0.1425	5270	0.2704	4	2.439E-5	0.0002732
-0.0003526	5284	0.2837	4	5.313E-5	0.0003844
-0.1432	5346	0.3456	4	9.395E-5	0.0003759
-0.286	5426	0.4259	4	1.278E-4	0.0003055
-0.4288	5501	0.5012	4	1.708E-4	0.0002102
-0.5716	5567	0.5665	4	1.968E-4	0.0001018
-0.7143	5628	0.6279	4	2.150E-4	-0.00001421
-0.857	5707	0.7074	4	2.245E-4	-0.0001309
-0.9997	5816	0.8159	4	2.223E-4	-0.0002462
-1.142	5952	0.9524	4	2.133E-4	-0.0003246
-1.285	6110	1.11	4	1.586E-4	-0.0003514
-1.428	6236	1.236	4	9.541E-5	-0.0003224
-1.57	6303	1.303	4	2.864E-5	-0.0002394
-1.713	6307	1.307	4	0.000E0	0
0.2852	5230	0.2296	4	0.000E0	0
0.1424	5217	0.2167	4	2.624E-6	0.0002847
-0.0004481	5254	0.2544	4	-9.626E-6	0.0003924
-0.1433	5346	0.346	4	-7.733E-7	0.0003962
-0.2861	5462	0.4616	4	7.066E-6	0.0003328
-0.4289	5563	0.563	4	3.202E-5	0.0002357
-0.5717	5641	0.641	4	4.466E-5	0.0001173
-0.7144	5704	0.7041	4	5.148E-5	-0.000001565
-0.8572	5772	0.7717	4	5.570E-5	-0.0001208
-0.9998	5859	0.8589	4	6.010E-5	-0.0002452
-1.143	5973	0.9731	4	7.170E-5	-0.0003405
-1.285	6107	1.107	4	5.406E-5	-0.0003918
-1.428	6217	1.217	4	3.411E-5	-0.0003808
-1.57	6270	1.27	4	7.590E-6	-0.0002802
-1.713	6266	1.266	4	0.000E0	0
0.2852	5185	0.1851	4	0.000E0	0
0.1423	5185	0.1854	4	-2.477E-5	0.0002507
-0.0005436	5240	0.2404	4	-8.511E-5	0.0003568
-0.1434	5355	0.3553	4	-1.117E-4	0.0003877
-0.2863	5503	0.5031	4	-1.290E-4	0.0003448
-0.4291	5629	0.629	4	-1.174E-4	0.0002526
-0.5719	5722	0.7219	4	-1.110E-4	0.0001315
-0.7146	5790	0.7901	4	-1.057E-4	0.000008866
-0.8573	5850	0.8496	4	-9.617E-5	-0.0001139
-1	5920	0.9204	4	-8.120E-5	-0.0002334
-1.143	6010	1.01	4	-5.397E-5	-0.0003323
-1.285	6113	1.113	4	-4.359E-5	-0.0003957
-1.428	6197	1.197	4	-2.668E-5	-0.0003941
-1.571	6229	1.229	4	-1.513E-5	-0.0002801
-1.713	6215	1.215	4	0.000E0	0
0.2851	5147	0.1467	4	0.000E0	0
0.1423	5149	0.1492	4	-5.409E-5	0.0001906
-0.0005823	5199	0.1994	4	-1.536E-4	0.0002931

-0.1435	5314	0.3136	4	-2.207E-4	0.0003536
-0.2865	5485	0.4848	4	-2.669E-4	0.0003357
-0.4292	5643	0.6433	4	-2.719E-4	0.0002525
-0.5719	5767	0.7674	4	-2.684E-4	0.0001368
-0.7147	5854	0.8544	4	-2.550E-4	0.00001995
-0.8574	5922	0.9217	4	-2.360E-4	-0.0000985
-1	5991	0.9906	4	-2.082E-4	-0.000203
-1.143	6063	1.063	4	-1.596E-4	-0.0002941
-1.285	6134	1.134	4	-1.189E-4	-0.0003584
-1.428	6181	1.181	4	-6.472E-5	-0.0003592
-1.571	6178	1.178	4	-1.935E-5	-0.0002477
-1.714	6140	1.14	4	0.000E0	0
0.285	5109	0.1093	4	0.000E0	0
0.1422	5114	0.1137	4	-5.810E-5	0.00009632
-0.0006209	5142	0.1416	4	-1.857E-4	0.000186
-0.1437	5229	0.2292	4	-2.911E-4	0.0002786
-0.2867	5395	0.3953	4	-3.771E-4	0.0002964
-0.4294	5587	0.5868	4	-4.205E-4	0.0002258
-0.572	5749	0.7486	4	-4.344E-4	0.0001229
-0.7148	5856	0.856	4	-4.174E-4	0.00002274
-0.8575	5934	0.9337	4	-3.825E-4	-0.00007911
-1	5999	0.9992	4	-3.329E-4	-0.0001666
-1.143	6053	1.053	4	-2.552E-4	-0.0002447
-1.286	6093	1.093	4	-1.817E-4	-0.0002964
-1.428	6104	1.104	4	-9.400E-5	-0.000294
-1.571	6070	1.07	4	-2.657E-5	-0.0002042
-1.714	6010	1.01	4	0.000E0	0
0.2853	5076	0.07633	4	0.000E0	0
0.1423	5075	0.07484	4	-2.655E-5	0.000007806
-0.0007695	5073	0.07344	4	-1.255E-4	0.00005711
-0.1436	5104	0.1041	4	-2.420E-4	0.000162
-0.2864	5208	0.2078	4	-3.636E-4	0.0002192
-0.4292	5377	0.3773	4	-4.634E-4	0.0001783
-0.5721	5574	0.5737	4	-5.458E-4	0.0001127
-0.7148	5706	0.7064	4	-5.654E-4	0.00005014
-0.8576	5815	0.8151	4	-5.412E-4	-0.00002281
-1	5887	0.8869	4	-4.817E-4	-0.00008481
-1.143	5935	0.9348	4	-3.750E-4	-0.0001423
-1.286	5951	0.9514	4	-2.675E-4	-0.0001789
-1.429	5930	0.9296	4	-1.383E-4	-0.0001855
-1.571	5866	0.8657	4	-3.904E-5	-0.0001434
-1.714	5789	0.7888	4	0.000E0	0
0.2856	5015	0.01501	4	0.000E0	0
0.1424	5050	0.0499	4	1.650E-5	-0.00003792
-0.0009181	4989	0	2	0.000E0	0
-0.1435	4962	0	2	0.000E0	0
-0.2861	5012	0.01226	4	-1.752E-4	0.00006113
-0.4291	5054	0.05399	4	-3.263E-4	0.00006718
-0.5721	5203	0.2031	4	-4.345E-4	0.00006556
-0.7149	5319	0.3186	4	-5.081E-4	0.00005376
-0.8577	5442	0.4417	4	-5.271E-4	0.00002952
-1	5511	0.5109	4	-5.059E-4	0.000001969
-1.143	5549	0.5494	4	-4.196E-4	-0.00002726
-1.286	5563	0.5625	4	-3.240E-4	-0.00005135
-1.429	5547	0.5468	4	-1.853E-4	-0.00006537

-1.572	5512	0.5123	4	-6.661E-5	-0.00006016
-1.714	5434	0.434	4	0.000E0	0
0.2853	3904	-0.09627	1	0.000E0	0
0.1423	3900	-0.1004	1	0.000E0	0
-0.0007391	3892	-0.1082	1	0.000E0	0
-0.1435	3884	-0.1159	1	0.000E0	0
-0.2863	3887	-0.1134	1	0.000E0	0
-0.4292	3911	-0.08889	1	0.000E0	0
-0.5721	3958	-0.04232	1	0.000E0	0
-0.7149	4014	0	2	0.000E0	0
-0.8577	4139	0	2	0.000E0	0
-1	4313	0	2	0.000E0	0
-1.143	4473	0	2	0.000E0	0
-1.286	4536	0	2	0.000E0	0
-1.429	4543	0	2	0.000E0	0
-1.572	4416	0	2	0.000E0	0
-1.714	4331	0	2	0.000E0	0
0.285	3766	-0.2341	1	0.000E0	0
0.1422	3768	-0.2324	1	0.000E0	0
-0.0005601	3762	-0.2379	1	0.000E0	0
-0.1435	3761	-0.2388	1	0.000E0	0
-0.2865	3762	-0.2377	1	0.000E0	0
-0.4293	3771	-0.2295	1	0.000E0	0
-0.5721	3790	-0.2105	1	0.000E0	0
-0.7149	3802	-0.1979	1	0.000E0	0
-0.8578	3807	-0.1934	1	0.000E0	0
-1.001	3802	-0.1975	1	0.000E0	0
-1.143	3800	-0.2005	1	0.000E0	0
-1.286	3799	-0.2013	1	0.000E0	0
-1.429	3799	-0.2006	1	0.000E0	0
-1.572	3803	-0.1972	1	0.000E0	0
-1.714	3801	-0.1993	1	0.000E0	0
0.2851	3634	-0.3658	1	0.000E0	0
0.1422	3634	-0.3663	1	0.000E0	0
-0.0006266	3633	-0.3673	1	0.000E0	0
-0.1435	3632	-0.368	1	0.000E0	0
-0.2864	3633	-0.3674	1	0.000E0	0
-0.4293	3636	-0.3642	1	0.000E0	0
-0.5721	3641	-0.3586	1	0.000E0	0
-0.7149	3647	-0.353	1	0.000E0	0
-0.8577	3650	-0.3495	1	0.000E0	0
-1	3651	-0.3486	1	0.000E0	0
-1.143	3651	-0.349	1	0.000E0	0
-1.286	3651	-0.349	1	0.000E0	0
-1.429	3652	-0.3485	1	0.000E0	0
-1.572	3652	-0.3479	1	0.000E0	0
-1.715	3652	-0.3476	1	0.000E0	0
0.2852	3500	-0.5	1	0.000E0	0
0.1422	3500	-0.5	1	0.000E0	0
-0.0006932	3500	-0.5	1	0.000E0	0
-0.1435	3500	-0.5	1	0.000E0	0
-0.2864	3500	-0.5	1	0.000E0	0
-0.4292	3500	-0.5	1	0.000E0	0
-0.5721	3500	-0.5	1	0.000E0	0
-0.7149	3500	-0.5	1	0.000E0	0

-0.8577	3500	-0.5	1	0.000E0	0
-1	3500	-0.5	1	0.000E0	0
-1.143	3500	-0.5	1	0.000E0	0
-1.286	3500	-0.5	1	0.000E0	0
-1.429	3500	-0.5	1	0.000E0	0
-1.572	3500	-0.5	1	0.000E0	0
-1.715	3500	-0.5	1	0.000E0	0

REFERENCES

REFERENCES

1. Albert, M. R., and O'Neill, K., "Moving Boundary-Moving Mesh Analysis of Phase Change Using Finite Elements with Transfinite Mappings," *Int. J. Num. Methods in Engin.*, **23**, 591, (1986).

2. Arpaci, S., and Larsen, P. S., *Convection Heat Transfer*, Prentice-Hall, Inc., N.J., (1984).

3. Baker, A. J., "A Finite Element Solution Algorithm for Viscous Incompressible Fluid Dynamics," *Int. J. Num. Meth. Engin.*, **6**, 89, (1973).

4. Baker, A. J., *Finite Element Computational Fluid Mechanics*, McGraw-Hill Book Co., N.Y., (1983).

5. Bamberger, M., Nadiv, S., and Barkay, G. B., "Mathematical Model for the Solidification of High-carbon Steel in Continuous Casting," *Iron and Steel International*, February, (1977).

6. Ben-Sabar, E., and Caswell, B., "A Stable Finite Element Simulation of Convective Transport," *Int. J. Numer. Methods Eng.*, **14**, 545, (1979).

7. Bercovier, M., and Pironneau, O., "Error Estimates for Finite Element Method Solution of the Stokes Problem in Primitive Variables," *Numer. Math.*, **33**, 211, (1979).

8. Bonacina, C., et al.,*Int. J. Heat and Mass Trans.*, **16**, 1825, (1973).

9. Budhia, H., and Kreith, F., "Heat transfer with Melting of Freezing in a Wedge," *Int. J. Heat and Mass Trans.*, **5**, 339 (1962).

10. Burns, R. K., "Preliminary Thermal Performance Analysis of the Solar Brayton Heat Receiver," *NASA Report TN D-6268*, (1971).

11. Carnahan, B., Luther, H. A., and Wilkes, J. O., *Applied Numerical Methods*, John Wiley and Sons, Inc., N.Y., (1969).

12. Chang, C. J., and Brown, R. A., "Finite Element Calculation of Buoyancy-Driven Convection Near a Melt/Solid Phase Boundary," *Proceedings, Second National Symposium on Numerical Methods in Heat Transfer, Maryland*, (1983).

13. Chen, C. J., Naseri-Neshat, H., and Ho, K. S., "Finite-Analytic Numerical Solution of Heat Transfer in Two-Dimensional Cavity Flow," *Numer. Heat Transfer*, **4**, 179, (1981).

14. Crowley, A. B., "Numerical Solution of Stefan Problems", *Mass Transfer*, **21**, 215, (1977).

15. De Vahl Davis, G., "Laminar Natural Convection in an Enclosed Rectangular Cavity," *Int. J. Heat Mass Transfer*, **11**, 1675, (1968).

16. Emery, A. F., Sugihara, K., and Jones, A. T., "A Comparison of Some of the Thermal Characteristics of Finite Element and Finite Difference Calculations of Transient Problems," *Num. Heat Trans.*, **2**, 97, (1979).

17. Ergatoudis, I., Irons, B. M., and Zienkewicz, O. C., "Curved, Isoparametric, 'Quadrilateral' Elements for Finite Element Analysis," *Int. J. Solids Structures*, **4**, 31, (1968).

18. Ettouney, H. M., and Brown, R. A., "Finite-Element Methods for Steady Solidification Problems," *Journal of Computational Physics*, **49**, 118, (1983).

19. Felippa, C. A. and Clough, R. W., "The Finite Element Method in Solid Mechanics," *SIAM-AMS Proceedings*, American Mathematical Society, Providence, R.I., **2**, 210, (1970).

20. Fletcher, C. A. J., "On an Alternating Direction Implicit Finite Element Method for Flow Problems," *Computer Methods in Applied Mechanics and Engineering*, **30**, 307, (1982).

21. Gallagher, R. H., Zienkiewicz, O. C., Oden, J. T., Morandi Cecchi, M., and Taylor, C., eds., *Finite Elements in Fluids*, Vol. 3, John Wiley and Sons, Inc., N.Y., (1978).

22. Galerkin, B. G., "Series Occurring in Some Problems of Elastic Stability of Rods and Plates," *Engineering Bulletin*, **19**, 897, (1915).

23. Gartling, D. K., and Nickel, R. E., "A Finite Element Convergence Study for Accelerating Flow Problems," *Int. J. Num. Methods in Engin.*, **11**, 1155, (1977).

24. Gartling, D. K., "Convective Heat Transfer Analysis by the Finite Element Method," *Comput. Methods Appl. Mech. Eng.*, **12**, 365, (1977).

25. Geradin, M., Idelsohn, S., and Hogge, M., "Computational Strategies for the Solution of Large Nonlinear Problems Via Quasi-Newton Methods," *Computers and Structures*, **13**, 73, (1980).

26. Gilman, L., and Rose, A. J., *APL, An Interactive Approach*, John Wiley and Sons, Inc., N.Y., (1984).

27 Graebel, W.P., "The Influence of Prandtl Number on Free Convection in a Rectangular Cavity," *Int. J. Heat Mass Transfer*, **24**, 125, (1981).

28. Gresho, P. M., Lee, R. L., and Sani, R. L., "Advection Dominated Flows with Emphasis on Consequences of Mass Lumpings," *Finite Elements in Fluids*, Vol. 3, Gallagher, R. H., Zienkiewicz, O. C., Oden, J. T., Cecchi, M. M., and Taylor, C., (eds.), John Wiley and Sons, New York, 335, (1978).

29. Hogge, M. A., "A Comparison of Two- and Three-Level Integration Schemes for Non-Linear Heat Conduction," *Numerical Methods in Heat Transfer*, Chapter 4, 75, Lewis, R. W., Morgan, K., and Zienkiewicz, O. C. (eds.), John Wiley and Sons, New York, (1981).

30. Hood, P., and Taylor, C., "A Numerical Solution of the Navier-Stokes Equations Using the Finite-Element Technique, *Comput. Fluids*, **1**, 73, (1973).

31. Hood, P., and Taylor, C., "Navier Stokes Equations Using Mixed Interpolation, *Finite Element Methods in Flow Problems*, Proceedings of the International Symposium on Finite Element Methods in Flow Problems, 121, Oden, J. T., Zienkiewicz, O. C., Gallagher, R. H., and Taylor, C., (eds.), held at University College of Swansea, Wales., January 1974, University of Alabama Press, University of Alabama, Huntsville, Ala., (1974).

32. Huebner, K. H., and Thornton, E. A., *The Finite Element Method for Engineers*, John Wiley and Sons, Inc., N.Y., (1982).

33. Iverson, K. E., *A Programming Language*, John Wiley and Sons, New York, (1962).

34. Klann, J. L., "Steady-State Analysis of a Brayton Space-Power System," *NASA Report TN D-5673*, February, (1970).

35. Kreith, F., *Principles of Heat Transfer*, Intext Press Inc., N.Y., (1973).

36. Labus, T. L., "Gas-Jet Impingement Normal to a Liquid Surface," *NASA Report TND-6368*, (1971).

37. Lazaridis, A., "A Numerical Solution of the Multidimensional Solidification (or Melting) Problem," *Int. J. Heat Mass Transfer*, **13**, 1459, (1970).

38. Lunardini, V. J., *Heat Transfer in Cold Climates*, Van Nostrand Reinhold Co., N.Y., (1981).

39. Marshall, R. S., Heinrich, J. C., and Zienkiewicz, O. C., "Natural convection in a Square Enclosure by a Finite-element, Penalty Function Method Using Primitive Fluid Variables," *Num. Heat Transfer*, **1**, 315, (1978).

40. Merte, H., Jr., "Profile and Departure Size of Condensation Drops on Vertical Surfaces," *Wärme- und Stoffübertragung* **17**, 171, (1983).

41. Murray, W. D., and Landis, F., "Numerical and Machine Solutions of Transient Heat Conduction Problems Involving Melting or Freezing," *J. Heat Transfer*, **81**, 106, (1959).

42. Oden, J. T.,"A General Theory of Finite Elements, II Applications," *Int. J. Num. Methods Engin.*, **1**, 247, (1969).

43. Oliveira, E. R. A., "Theoretical Foundations of the Finite Element Method," *Int. J. Solids Struct.*, **4**, 929, (1968).

44. Olson, M. D., "Formulation of a Variational Principle-Finite Element Method for Viscous Flows," Proc. *Variational Methods in Engineering*, Southampton University, 5.27, (1972).

45. Olson, M. D., and Tuann, S. Y., "Primitive Variables Versus Stream Function Finite Element Solutions of the Navier-Stokes Equations," *Finite Elements in Fluids*, Vol. 3, Gallagher, R. H., and Zienkiewicz, O. C., eds., John Wiley & Sons, Inc., N.Y., (1978).

46. Otis, D. R., "Solving the Melting Problem Using the Electric Analog to Heat Conduction," Heat Transfer and Fluid Mechanics Institute, Stanford University, Stanford, California, (1956).

47. Ozoe, H., Sayama, H., and Churchill, S. W., "Natural Convection in an Inclined Square Channel," *Int. J. Heat Mass Transfer*, **17**, 401, (1974).

48. Pearson, J. R. A., "On Convection Cells Induced by Surface Tension,", *J. Fluid Mech.*, **4**, 489, (1958).

49. Pepper, D. W. and Cooper, R. E., "Numerical Solution of Recirculating Flow by a Simple Finite Element Recursion Relation," *J. Comp. Fluids*, **8**, 213, (1980).

50. Poots, G., "An Approximate Treatment of a Heat Conduction Problem Involving a Two-Dimensional Solidification Front," *Int. J. Heat Mass Transfer*, **5**, 339, (1962).

51. Roux, B., Grondin, J. C., Bontoux, P., and Gilly, B., "On a High-Order Accurate Method for the Numerical Study of Natural Convection in a Vertical Cavity," *Num. Heat Transfer*, **1**, 331, (1978).

52. Schneider, G. E., "Computation of heat Transfer with Solid/Liquid Phase Change Including Free Convection," *J. Thermophysics*, **1**, 136, (1968).

53. Shamsundar, N., and Sparrow, E. M., "Analysis of Multidimensional Conduction Phase Change Via the Enthalpy Model," *Journal of Heat Transfer*, **97**, 333, (1975).

54. Siegel, R., and Savino, J. M., "An Analytical Solution for Solidification of a Moving Warm Liquid onto an Isothermal Cold Wall," *Int J. Heat Mass Transfer*, **12**, 803, (1968).

55. Siegel, R., "Effects of Reduced Gravity on Heat Transfer," *Advances in Heat Transfer*, Academic Press, Inc., N.Y., **4**, 143, (1967).

56. Springer, G. S., and Olson, D. R., "Axisymmetric Solidification and Melting of Materials,", *ASME Paper 63-WA-185*, (1963).

57. Taylor, C., and Ijam, A. Z., "A Finite Element Numerical Solution of Natural Convection in Enclosed Cavities," *Comput. Methods Appl. Mech. Eng.*, **19**, 429, (1979).

58. Stefan, J., "Polar Ice Progression," *Ann. Phys. Chem.*, **42**, 269, (1891).

59. Tien, L. C., "Freezing of a Convective Liquid in Crystal-Growth Tube", Ph.D. Thesis, The University of Michigan, Ann Arbor, Michigan, (1968).

60. Valle, A., "The Finite-Element Method and the Stefan

Problem", Ph.D. Thesis, The University of Michigan, Ann Arbor, Michigan, (1979).

61. Van Wylen, G. J., and Sonntag, R. E., *Fundamentals of Classical Thermodynamics*, John Wiley and Sons, Inc., N.Y., (1978).

62. White, R. E., *An Introduction to the Finite Element Method with Applications to Nonlinear Problems*, John Wiley and Sons, Inc., N.Y., (1985).

63. Wilkes, J. O., and Churchill, S. W., "The finite Difference Computation of Natural Convection in a rectangular Enclosure," *AIChE J.*, **12**, 161, (1966).

64. Yamada, Y., Ito, K., Yokouchi, Y., Tamano, T., and Ohtsubo, T., "Finite Element Analysis of Steady Fluid and Metal Flow," *Finite Elements in Fluids*, Chapter 4, 73, Gallagher, R. H., Oden, J. T., Taylor, C., and Zienkiewicz, O. C., (eds.), John Wiley and Sons, New York, (1975).

65. Zienkiewicz, O. C., and Phillips, D. V., "An Automatic Mesh Generation Scheme for Plane and Curved Surfaces by 'Isoparametric' Co-ordinates," *Int. J. Num. Methods Engin.*, **3**, 519, (1971).

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-103721		

4. Title and Subtitle

A Finite Element Model of Conduction, Convection, and Phase Change Near a Solid/Melt Interface

5. Report Date

January 1991

6. Performing Organization Code

7. Author(s)

Larry A. Viterna

8. Performing Organization Report No.

E-5952

10. Work Unit No.

474-12-10

9. Performing Organization Name and Address

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

11. Contract or Grant No.

13. Type of Report and Period Covered

Technical Memorandum

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Washington, D.C. 20546-0001

14. Sponsoring Agency Code

15. Supplementary Notes

Report was submitted as a dissertation in partial fulfillment of the requirements for the degree Doctor of Philosophy to the University of Michigan, Ann Arbor, Michigan 48109. Responsible person, Larry Viterna, (216) 433-5398.

16. Abstract

Detailed understanding of heat transfer and fluid flow is required for many aerospace thermal systems. These systems often include phase change and operate over a range of accelerations or effective gravitational fields. An approach to analyzing such systems is presented which requires the simultaneous solution of the conservation laws of energy, momentum, and mass, as well as an equation of state. The variable property form of the governing equations are developed in two-dimensional Cartesian coordinates for a Newtonian fluid. A numerical procedure for solving the governing equations is presented and implemented in a computer program. The Galerkin form of the finite element method is used to solve the spatial variation of the field variables, along with the implicit Crank-Nicolson time marching algorithm. Quadratic Lagrangian elements are used for the internal energy and the two components of velocity. Linear Lagrangian elements are used for the pressure. The location of the solid/liquid interface as well as the temperatures are determined from the calculated internal energy and pressure. This approach is quite general in that it can describe heat transfer without phase change, phase change with a sharp interface, and phase change without an interface. Analytical results from this model are compared to those of other researchers studying transient conduction, convection, and phase change and are found to be in good agreement. The numerical procedure presented requires significant computer resources, but this is not unusual when compared to similar studies by other researchers. Several methods are suggested to reduce the computational times.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Finite element; Phase change; Latent heat; Enthalpy; Computational fluids	Unclassified – Unlimited Subject Category 34

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	187	A09

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161