SANDIA REPORT SAND84—2057 • Unlimited Release • UC—70
Printed August 1985

Nevada Nuclear Waste Storage Investigations Project

NORIA—A Finite Element Computer Program for Analyzing Water, Vapor, Air, and Energy Transport in Porous Media

N. E. Bixler

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

"Prepared by Nevada Nuclear Waste Storage Investigations (NNWSI) Project participants as part of the Civilian Radioactive Waste Management Program (CRWM). The NNWSI Project is managed by the Waste Management Project Office (WMPO) of the U. S. Department of Energy, Nevada Operations Office (DOE/NV). NNWSI Project work is sponsored by the Office of Geologic Repositories (OGR) of the DOE Office of Civilian Radioactive Waste Management (OCRWM)."

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A06
Microfiche copy: A01

Distribution
UC-70

SAND84-2057
Unlimited Release
Printed August, 1985

NORIA—A Finite Element Computer Program for Analyzing Water, Vapor, Air, and Energy Transport in Porous Media

N. E. Bixler
Fluid and Thermal Sciences Department
Sandia National Laboratories
Albuquerque, NM 87185

Abstract

NORIA is a finite element computer program that solves four nonlinear, parabolic, partial differential equations simultaneously. The four equations describe the transport of water, water vapor, air, and energy through partially saturated porous media. The numerical procedure uses the standard Galerkin finite element method to handle spatial discretization of two-dimensional domains with either planar symmetry or axisymmetry. Time integration is performed by a third-order predictor-corrector scheme that uses error estimates to automatically adjust time-step size so as to maintain uniform local time truncation error throughout the calculation. Thus, the user is not required to select time-step size except at the first time step. Nearly all material properties, such as permeability, can either be set to constant values or can be defined as functions of the dependent and independent variables by user-supplied subroutines. The gas phase is taken to be ideal. This report is intended primarily as a user's manual but also includes discussions of the theory of two-phase transport in porous media and the numerical procedure used in NORIA.

Preface

A family of finite element computer programs has been developed at Sandia National Laboratories (SNL) that includes COYOTE (Gartling, 1982), NACHOS (Gartling, 1978), MARIAH (Gartling and Hickox, 1980), SAGUARO (Eaton et al., 1983), and most recently, NORIA. Each of these programs has several features in common, such as mesh generator, input deck structure, and graphics package. In fact, these programs have been kept as similar as possible. Much of the credit for these programs goes to D. K. Gartling, who assembled the basic building blocks from which all of these programs have been built.

NORIA strongly resembles its predecessors; however, several features have been built into NORIA that depart from those in the older finite element programs developed at SNL. Some of these features were developed to meet specific needs and others were developed to provide convenience and flexibility. The most notable inovation is in the time integrator, which uses a predictor-corrector scheme coupled with a Newton iteration procedure. The advantages are threefold: (1) time truncation error is third order in time-step size rather than second order as in most of the other codes developed at SNL; (2) time-step size is automatically adjusted at each time plane to maintain uniform time truncation error even when natural time scales vary severely during a calculation (variable time stepping is also used in SAGUARO, although the method is quite different than the one used here); and (3) the user is only required to specify time-step size at the initial time plane rather than at each time plane throughout the calculation. Thus, in addition to improved accuracy, the new time integrator offers convenience to the user because he need not make *a priori* estimates of natural time scales. In fact, an adaptive time-stepping scheme is a requirement in NORIA because large and unpredictable variations in natural time scale sometimes occur in nonisothermal, two-phase porous-flow problems. Because the user has little control over time-step size, the specification of the time planes at which output should be printed has changed significantly; time planes are specified by time rather than by number.

In writing NORIA, algorithmic changes from earlier finite element programs developed at SNL have been made only where required or highly advantageous, and the input deck structure has been changed as little as possible. As a result, users familiar with other finite element codes developed at SNL should have little difficulty in mastering the use of NORIA.

The work described in this report was performed for SNL as part of the Nevada Nuclear Waste Storage Investigations (NNWSI) Project. SNL is one of the principal organizations participating in the project, which is managed by the U. S. Department of Energy's Nevada Operations Office. The project is part of DOE's program to safely dispose of the high-level radioactive waste from nuclear power plants.

The DOE has determined that the safest and most feasible method currently known for the disposal of such wastes is to emplace them in mined geologic repositories. The NNWSI Project is conducting detailed studies of an area at Yucca Mountain in southern Nevada to determine the feasibility of developing a repository.

Before a license to operate the repository can be obtained, performance assessment studies must be performed. As part of this assessment, water, vapor, energy, and radionuclide transport in tuff must be analyzed. Predictions of behavior in the near field require a computer code that simulates water movement through unsaturated geologic media in the presence of a transient heat flux that is sufficient to cause vapor generation. Tractable forms of the governing equations and the equations-of-state have been formulated. NORIA has been developed to solve these equations. This report describes the development and application of the finite element code, NORIA, for use in analyzing two-dimensional transport of water, water vapor, air, and energy through partially saturated porous media.

A special note of thanks is owed to R. R. Eaton for helping in familiarizing me with SAGUARO, which was the point of departure for NORIA. Also, thanks to R. E. Benner for guidance in vectorization and optimization on the Cray 1S and to P. L. Hopkins for programming assistance on several of the plotting routines.

Contents

Table

Figures

1 Introduction

The calculation of transient, nonisothermal, two-phase flow in porous media is of interest in volcanology, geothermal energy technology, radioactive waste disposal, tertiary petroleum recovery, packed-bed reactor technology, and drying of porous products such as paper and foodstuffs. The mathematical equations that describe nonisothermal, two-phase flows in porous media are generally highly nonlinear and are thus not amenable to analytic solution except, perhaps, in the very simplest cases. Moreover, analytic solutions cannot be obtained when problem domains are irregular. The finite element method is a natural choice for solving problems with irregular boundaries, especially when Neumann (flux-type) boundary conditions are specified.

The solution procedure in NORIA is transient; no provision has been made for obtaining steady solutions because such a steady solution procedure would probably not be convergent for the vast majority of analyses for which NORIA would be used. However, it is always possible to construct a transient analysis that will approach the desired steady state after some elapsed time.

NORIA is intended to solve nonisothermal problems in which large gradients are expected in the gas pressure. When little or no pressure gradient is expected in the gas phase, SAGUARO (Eaton *et al.*, 1984) would be able to solve the problem much more efficiently than NORIA. If no gas phase is present at all, *i.e.*, the porous medium is fully saturated with liquid, MARIAH (Gartling and Hickox, 1980) would be the logical choice.

To the best of the author's knowledge, only three other codes exist that can solve the same types of problems that are solved by NORIA. PETROS developed by Hadley (1985) at Sandia National Laboratories (SNL), is a one-dimensional finite difference program that solves essentially the same set of equations as those solved by NORIA. TOUGH, developed by Pruess and Wang (1983) at Lawrence Berkeley Laboratory, is an integrated finite difference program that solves a simpler set of equations than those solved by NORIA. The integrated finite difference scheme used in TOUGH allows one-, two-, and three-dimensional calculations to be performed with relatively high efficiency. However, the mathematical model used in TOUGH does not include Knudsen diffusion or nonlinear binary diffusion. Finally, WAFE, developed by B. J. Travis of Los Alamos National Laboratory, is a finite difference program. No documentation is currently available.

Except for NORIA, each of the computer programs mentioned in the preceding paragraph is based on finite differences. To a large extent, the creation of NORIA was motivated by the need for a vapor transport program that could be used in conjunction with existing finite element programs to perform radionuclide transport (Martinez, 1985) and thermostress analyses. The compatibility of finite element and finite differ-

ence programs is marginal because dependent variables are assigned values only at node points in standard finite differences, whereas dependent variables are known everywhere within a domain in finite elements. The difference in representations requires either that the node points coincide or that some *ad hoc* method be devised to interpolate solutions between finite-difference node points in order to make the two methods compatible. Furthermore, differences in data structure require that a translator be built in order to allow finite difference and finite element programs to communicate with each other. These problems are best avoided by consistent use of either the finite element or the finite difference method for all coupled calculations.

The following sections provide theoretical background and information on how to use NORIA. Section 2 descibes the set of equations that govern the transport of liquid water, water vapor, air, and energy through a porous medium. Section 3 discusses the spatial discretization of the dependent variables in NORIA, the Galerkin finite element formulation of the governing equations, and the enforcement of boundary conditions. Details of the time integration procedure are set out in Section 4. Sections 5–7 are directed primarily to the use of NORIA: Section 5 gives an overview of the functions that are performed by NORIA; Section 6 describes the input needed to run NORIA; and Section 7 illustrates the information in Section 6 by a sample problem. Finally, Section 8 is a glossary of terms that may not be familiar to users who are not trained in finite element analysis and numerical methods.

2 Theory and Mathematical Model

2.1 Background

The theory of two-phase transport through porous media dates back to the late 1960s and early 1970s. Whitaker (1967 and 1973) first derived the basic governing equations based on averaging theory. Slattery (1970) was also a pioneer in the application of averaging theory to porous flows. More recent and more complete attempts to model such flows include the works of Zanotti and Carbonell (1984) and Hadley (1982). The equations in the following subsection are the same as those in Hadley (1985). Further elaboration of the derivation of these equations is given by Hadley, Wilson, and Nunziato (1985), who have compared the older and simpler averaging theory with a newer and more sophisticated approach known as mixture theory. The two theories lead to the same result and so put the mathematical model used here on a firm foundation. Furthermore, Hadley (1984) has shown that this model describes well the drying experiment performed by Ceaglske and Hougen (1937) in which an initially wet layer of sand was dried in a desiccator.

The mathematical model described in the following two subsections is relatively general. Even so, the following assumptions have been made:

- The two phases consist of a single component in liquid and vapor phases and a second component that is an inert gas (which does not dissolve in the liquid). Here, the liquid phase is assumed to be water and the inert gas is assumed to be air, but other constituents can be modeled equally well by NORIA.
- Both air and vapor are taken to be ideal. Thus, the partial pressure of each component is described by the ideal gas law, and the partial pressures are additive.
- The three phases are taken to be in local thermal equilibrium. Thus, the temperatures of rock matrix, liquid water, and gas are all equal locally.
- All viscous flow (flow at high enough densities so that molecular effects are unimportant) is laminar and obeys Richard's equation (Freeze and Cherry, 1979), which is a form of Darcy's law for unsaturated media.
- The liquid phase behaves as a Boussinesq fluid. In other words, the density is independent of pressure and varies only slightly in proportion to the difference between local temperature and a reference temperature.
- The porosity and density of the porous matrix are constant over each material. Up to ten materials are allowed.

Aside from these restrictions, all properties are general. Each property in each material can either be specified as a constant or can be defined in a user subroutine to depend on any of the dependent or independent variables.

2.2 Conservation Equations

The basic equations that govern two-phase flow in porous media are simply statements of conservation of mass and energy. Statements of conservation of mass for the liquid, vapor, and air components are respectively

$$\rho_l \frac{\partial \Theta}{\partial t} = -\nabla \cdot \mathbf{j}_l - F_v \tag{1}$$

$$\frac{\partial (\Phi - \Theta)\rho_v}{\partial t} = -\nabla \cdot \mathbf{j}_v + F_v \tag{2}$$

$$\frac{\partial (\Phi - \Theta)\rho_a}{\partial t} = -\nabla \cdot \mathbf{j}_a \tag{3}$$

where

ρ = density or partial density,

Θ = moisture content, *i.e.*, the volume fraction of rock that is filled by water,

Φ = porosity of the rock matrix,

t = time,

$\mathbf{j}$ = mass flux vector,

F_v = rate of vaporization per unit volume,

l = liquid water,

v = water vapor, and

a = air.

The terms on the left represent accumulation of mass; the first terms on the right represent net influx of mass; the second terms on the right are rates of generation or depletion of mass that result from evaporation or condensation.

As mentioned above, the ideal gas law is used to determine the densities in the gas phase:

$$\rho_c = \frac{p_c}{R_c T}$$

where

c = vapor or air,

p_c = partial pressure of component c,

R_c = ideal gas constant divided by the molecular mass of component c, and

T = absolute temperature.

The density of the gas phase, ρ_g, is the sum of the two component densities.

Conservation of energy leads to the following relationship:

$$[\rho C]_{ave}\frac{\partial T}{\partial t} = -(C_l\mathbf{j}_l + C_v\mathbf{j}_v + C_a\mathbf{j}_a)\cdot\nabla T + \nabla\cdot(\mathbf{\Lambda}\cdot\nabla T) - F_v L + Q \qquad (4)$$

where

C = heat capacity per unit mass,

$\mathbf{\Lambda}$ = thermal conductivity tensor,

L = latent heat of vaporization per unit mass, and

Q = heat input per unit volume.

The term on the left represents accumulation of sensible heat in the rock, water, and gas phases. The first term on the right represents the convection of sensible heat with the liquid and gas phases. The second term on the right represents the conduction of sensible heat. The third term on the right represents the the energy change resulting

from a liquid/vapor phase change. The last term is energy input by a volumetric heat source. The term in brackets is the average heat capacity per unit volume of the material as a whole, *i. e.*, rock matrix plus water plus gas, as defined by the following equation:

$$[\rho C]_{ave} = (1 - \Phi)\rho_s C_s + \Theta\rho_l C_l + (\Phi - \Theta)(\rho_v C_v + \rho_a C_a) \tag{5}$$

where

s = solid phase, *i.e.*, the rock grain.

For example, ρ_s is rock grain density.

In order to complete the governing Equations (1)–(4), relationships must be chosen to specify how the fluxes depend on the unknowns. Similarly, the rate of vaporization per unit volume, F_v, must be related to the unknowns. These relationships, which are constitutive in nature and thus depend on physical mechanisms, are discussed in the following two subsections.

2.3 Transport Equations

The following transport mechanisms are accounted for in the mathematical model used in NORIA:

- pressure-driven Darcy flow of liquid, vapor, and air;
- natural convection caused by temperature gradients (Boussinesq) in the liquid;
- natural convection in the gas phase caused by temperature and pressure gradients;
- Knudsen diffusion of gas molecules through pores caused by gradients in the density of a species;
- binary diffusion of vapor through air and *vice versa*, which results from gradients in the relative concentrations of the two species; and
- thermo-diffusion of each gas species through the other caused by temperature gradients.

Each of these mechanisms also contributes to convection of sensible heat. In addition, change of phase can occur with the associated latent energy effects (see Subsection 2.4).

The transport mechanisms included here are nearly the same as those used in PETROS (Hadley, 1984); the only difference is that Hadley considers buoyancy forces in the gas phase to be negligible. The expression for liquid flux is

$$\mathbf{j}_l = -\frac{\rho_l \mathbf{K}_l}{\mu_l} \cdot \nabla(P - \rho_l g \beta z \Delta T) \tag{6}$$

where

ρ_l = liquid density at a reference temperature,

$\mathbf{K}_l$ = permeability of the porous matrix to flow in the liquid phase,

μ_l = dynamic viscosity of the liquid,

P = effective pressure in the liquid phase, *i.e.*, $P = p_l + \rho_l g z$,

p_l = absolute pressure in the liquid phase,

g = the acceleration caused by gravity,

β = the coefficient of volumetric expansion for water,

z = the vertical coordinate in a Cartesian or polar cylindrical coordinate system, *i.e.*, the z-axis is aligned with the direction of gravity and is opposite in sense, and

ΔT = the difference between local temperature and the reference temperature, T_{ref}.

The single term on the right side of the equation represents Darcy flow resulting from effective pressure and density gradients.

The expressions for vapor and air fluxes are

$$\begin{aligned}
\mathbf{j}_v = \; & -\tfrac{\rho_v \mathbf{K}_g}{\mu_g} \cdot \nabla(p_g + \rho_g g z) \\
& - (\Phi - \Theta)\rho_v \tfrac{D_{Kv} D_{Ka}}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}} \nabla(p_g + \rho_g g z) \\
& - (\Phi - \Theta)\rho_v \tfrac{(p_g/p_v) D_{Kv} D_D}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}} \nabla(p_v + \rho_v g z) \\
& + (\Phi - \Theta)\rho_g \tfrac{R_a D_{Kv} D_T}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}} \nabla T
\end{aligned} \tag{8}$$

$$
\begin{aligned}
\mathbf{j}_a = & -\frac{\rho_a \mathbf{K}_g}{\mu_g} \cdot \nabla(p_g + \rho_g g z) \\
& - (\Phi - \Theta)\rho_a \frac{D_{Ka} D_{Kv}}{p_g D_B + p_v D_{Ka} + p_a D_{Kv}} \nabla(p_g + \rho_g g z) \\
& - (\Phi - \Theta)\rho_a \frac{(p_g/p_a) D_{Ka} D_B}{p_g D_B + p_v D_{Ka} + p_a D_{Kv}} \nabla(p_a + \rho_a g z) \\
& - (\Phi - \Theta)\rho_g \frac{R_v D_{Ka} D_T}{p_g D_B + p_v D_{Ka} + p_a D_{Kv}} \nabla T
\end{aligned}
\tag{9}
$$

where

$\mathbf{K}_g$ = permeability of the porous matrix to flow in the gaseous phase,

μ_g = dynamic viscosity of the gas phase, which is taken to be a partial-density weighted average of the component viscosities,

p_g = total gas pressure,

D_{Kv} = Knudsen diffusion coefficient for vapor,

D_{Ka} = Knudsen diffusion coefficient for air,

D_B = binary diffusion coefficient for a vapor/air mixture, and

D_T = thermo-diffusion coefficient for a vapor/air mixture.

The first terms on the right sides of these equations represent Darcy flow resulting from pressure and density gradients; the second terms represent Knudsen diffusion; the third terms represent binary diffusion; and the last terms represent thermo-diffusion.

Equations (6)–(8) determine the rates at which liquid water, water vapor, and air are convected. Together with Equations (1)–(4), they determine how a two-phase porous-flow system evolves with time. However, a relationship is still needed to determine the rate at which evaporation or condensation occurs.

2.4 Nonequilibrium Vaporization Model

Vaporization rate models fall into two classes: equilibrium and nonequilibrium. In the equilibrium model, it is assumed that vapor and liquid are always in local thermodynamic equilibrium, *i.e.*, vapor is always at its equilibrium pressure at the local temperature, provided that both phases are present.

$$p_v = p_v^*(T) \tag{9}$$

where

$p_v^*(T)$ = equilibrium vapor pressure of water as a function of temperature.

This model is appealing because one would expect local phase equilibrium to prevail unless local moisture content is nearly zero, in which case the vapor pressure should be determined by the conservation and transport equations, Equations (2) and (7). However, numerical implementation of vapor/liquid equilibrium has a basic shortcoming: when solving problems in which a subdomain of the problem domain is initially dry or becomes dry at some point during the computation, Equations (2) and (7) must be satisfied in a dry subdomain while the phase equilibrium condition, Equation (9), prevails in the complementary subdomain. Furthermore, an *ad hoc* cutoff in moisture content must be chosen to delineate dry from partially saturated subdomains. TOUGH uses the equilibrium vapor-pressure model as described above. Pruess and Wang (1983) find that this model results in rather severe temporal oscillations in local vapor pressures when a dry-out front forms and travels through a problem domain. More recently, Pruess implemented a vapor equilibrium model with vapor pressure lowering (see Udell, 1983) in TOUGH and found that no temporal oscillations in vapor pressure occurred because no dry-out zone was formed (Pruess, 1984). The alternative to what Pruess has done is to discard the equilibrium vapor-pressure model in favor of a nonequilibrium model.

The advantage of a nonequilibrium vapor-pressure model is that it accounts for the transition from partial saturation to zero saturation in a natural way. Furthermore, a single equation for vapor pressure suffices throughout the problem domain regardless of the existence of dry-out zones. Any nonequilibrium model is somewhat arbitrary in nature but perhaps no more so than an equilibrium model in which a moisture content cutoff must be specified to delineate partially saturated from dry subdomains.

NORIA is set up so that the user can either specify a model for the nonequilibrium vaporization rate or use a default model. In the default model, the vaporization rate is proportional to (1) the difference between local equilibrium vapor pressure and local partial vapor pressure and (2) the difference between local moisture content and residual moisture content.

$$F_v = c(\Theta - \Theta_r)(p_v^* - p_v) \tag{10}$$

where

Θ_r = residual moisture content and

c = a constant of proportionality.

By setting c to be large enough, the partial pressure of the vapor can be made to follow very nearly the local equilibrium vapor pressure. However, when moisture content approaches the residual value, Θ_r, the rate of vaporization approaches zero in a smooth fashion. The choice of c implies a time scale that controls the rate of vaporization. Ideally, c should be chosen so that the time scale for vaporization is somewhat smaller than the other time scales of a problem. An equation for estimating c can be derived by combining Equations (2) and (10) and by setting the vapor flux to zero.

$$t_e = \frac{\Phi}{c \cdot R_v \cdot T \cdot \Theta_r}$$

where

t_e = time scale for evaporation or condensation.

In practice, it may be necessary to vary c to ensure that the value is large enough to enforce local vapor/liquid equilibrium everywhere in the problem domain except where moisture content is close to its residual value.

3 Galerkin Finite Element Formulation

3.1 Weak Form of the Partial Differential Equations

The first step in applying the finite element method to a problem is to put the governing differential equations in weak form. This step has three parts: (1) products of the governing differential equations, Equations (1)–(4), are formed with a set of weighting functions that are defined in the following two subsections; (2) the products are integrated over the problem domain; and (3) Green's theorem is used to reduce all second-order derivatives to first order. Applying the first two steps results in the following expressions.

$$\int_\Omega \rho_l \frac{\partial \Theta}{\partial t} \psi_i dS = -\int_\Omega \nabla \cdot \mathbf{j}_l \psi_i dS - \int_\Omega F_v \psi_i dS \tag{11}$$

$$\int_\Omega \frac{\partial (\Phi - \Theta)\rho_v}{\partial t} \psi_i dS = -\int_\Omega \nabla \cdot \mathbf{j}_v \psi_i dS + \int_\Omega F_v \psi_i dS \tag{12}$$

$$\int_\Omega \frac{\partial (\Phi - \Theta)\rho_a}{\partial t} \psi_i dS = -\int_\Omega \nabla \cdot \mathbf{j}_a \psi_i dS \tag{13}$$

$$\begin{aligned} \int_\Omega [\rho C]_{ave} \frac{\partial T}{\partial t} \psi_i dS = & -\int_\Omega (C_l \mathbf{j}_l + C_v \mathbf{j}_v + C_a \mathbf{j}_a) \cdot \nabla T \psi_i dS \\ & + \int_\Omega \nabla \cdot (\Lambda \cdot \nabla T) \psi_i dS - \int_\Omega F_v L \psi_i dS + \int_\Omega Q \Psi_i dS \end{aligned} \tag{15}$$

where

ψ_i = the i-th weighting function,

Ω = the area of the problem domain, and

dS = the differential of area.

Applying Green's theorem to Equations (11)–(14) leads to the following results.

$$\int_{\Omega} \rho_l \frac{\partial \Theta}{\partial t} \psi_i dS - \int_{\Omega} \mathbf{j}_l \cdot \nabla \psi_i dS + \int_{\Omega} F_v \psi_i dS + \int_{\partial\Omega} \mathbf{j}_l \cdot \mathbf{n} \psi_i ds = 0 \tag{15}$$

$$\int_{\Omega} \frac{\partial (\Phi - \Theta)\rho_v}{\partial t} \psi_i dS - \int_{\Omega} \mathbf{j}_v \cdot \nabla \psi_i dS - \int_{\Omega} F_v \psi_i dS + \int_{\partial\Omega} \mathbf{j}_v \cdot \mathbf{n} \psi_i ds = 0 \tag{16}$$

$$\int_{\Omega} \frac{\partial (\Phi - \Theta)\rho_a}{\partial t} \psi_i dS - \int_{\Omega} \mathbf{j}_a \cdot \nabla \psi_i dS + \int_{\partial\Omega} \mathbf{j}_a \cdot \mathbf{n} \psi_i ds = 0 \tag{17}$$

$$\begin{aligned} \int_{\Omega} [\rho C]_{ave} \frac{\partial T}{\partial t} \psi_i dS \quad & + \int_{\Omega} (C_l \mathbf{j}_l + C_v \mathbf{j}_v + C_a \mathbf{j}_a) \cdot \nabla T \psi_i dS \\ & + \int_{\Omega} (\Lambda \cdot \nabla T) \cdot \nabla \psi_i dS + \int_{\Omega} F_v L \psi_i dS \\ & - \int_{\Omega} Q \psi_i dS - \int_{\partial\Omega} (\Lambda \cdot \nabla T) \cdot \mathbf{n} \psi_i ds = 0 \end{aligned} \tag{19}$$

where

$\mathbf{n}$ = the outward-pointing unit normal vector along the perimeter of the problem domain and

ds = the differential of length along the perimeter of the domain.

Equations (15)–(18) are the weak forms of Equations (1)–(4) and are the equations that are solved by NORIA. Equations (7)–(9) are used to evaluate $\mathbf{j}_l$, $\mathbf{j}_v$, and $\mathbf{j}_a$. An expression for the rate of vaporization, such as the one in Equation (10), is used to evaluate F_v in Equations (15)–(18).

3.2 Finite Element Basis Functions

The heart of the finite element method is the representation of dependent variables by a set of basis functions, each of which spans only a small subdomain of the problem domain. Definition of the set of basis functions is accomplished by subdividing the problem domain into elements and defining a set of local basis functions on each element. The problem domain is subdivided so that it is the direct sum of the elements, as shown in Figure 1, except, perhaps, at curvilinear domain boundaries where element sides might only approximate the true boundary.

A large number of element types have been used in finite element analysis. Four types are available in NORIA: (1) the eight-node isoparametric quadrilateral; (2) the eight-node subparametric quadrilateral; (3) the six-node isoparametric triangle; and (4) the six-node subparametric triangle. The number of nodes in an element corresponds to the number of basis functions used to represent a dependent variable within that element. Isoparametric and subparametric refer to the way an element is mapped into a standard element.

Finite element basis functions are generally defined on standard elements. In the natural coordinate system, denoted by (ξ, η), the standard eight-node quadrilateral element is a square with edges that fall on the lines $\xi = -1, +1$ and $\eta = -1, +1$ and with nodes that lie at corners and midway along sides; the standard six-node triangular element is an isosceles triangle with edges that fall on the lines $\xi = 0, \eta = 0$, and $\xi + \eta = 1$ and with nodes that lie at corners and midway along sides. The basis functions for the eight-node quadrilateral are

$$
\begin{aligned}
\phi_1 &= -\tfrac{1}{4}(1-\xi)(1-\eta)(1+\xi+\eta) \\
\phi_2 &= -\tfrac{1}{4}(1+\xi)(1-\eta)(1-\xi+\eta) \\
\phi_3 &= -\tfrac{1}{4}(1+\xi)(1+\eta)(1-\xi-\eta) \\
\phi_4 &= -\tfrac{1}{4}(1-\xi)(1+\eta)(1+\xi-\eta) \\
\phi_5 &= +\tfrac{1}{2}(1-\xi^2)(1-\eta) \\
\phi_6 &= +\tfrac{1}{2}(1+\xi)(1-\eta^2) \\
\phi_7 &= +\tfrac{1}{2}(1-\xi^2)(1+\eta) \\
\phi_8 &= +\tfrac{1}{2}(1-\xi)(1-\eta^2).
\end{aligned}
\tag{20}
$$

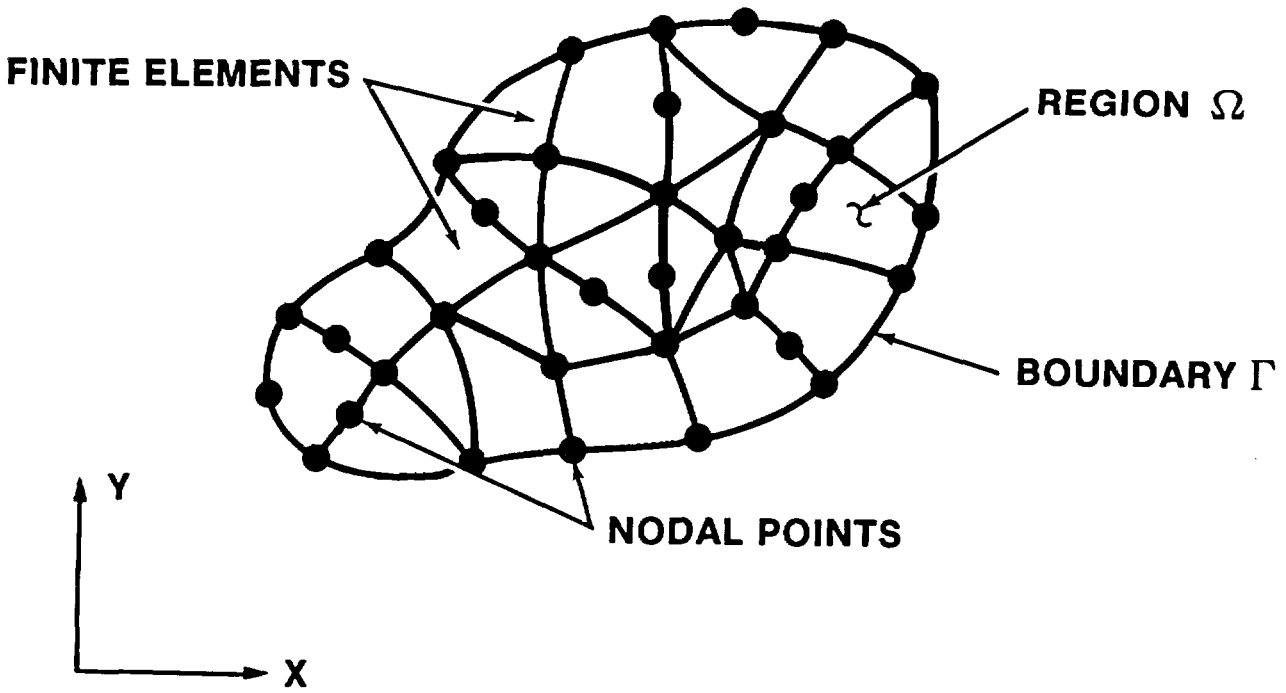


Figure 1: Finite Element Discretization of a Region

The basis functions for the six-node triangle are

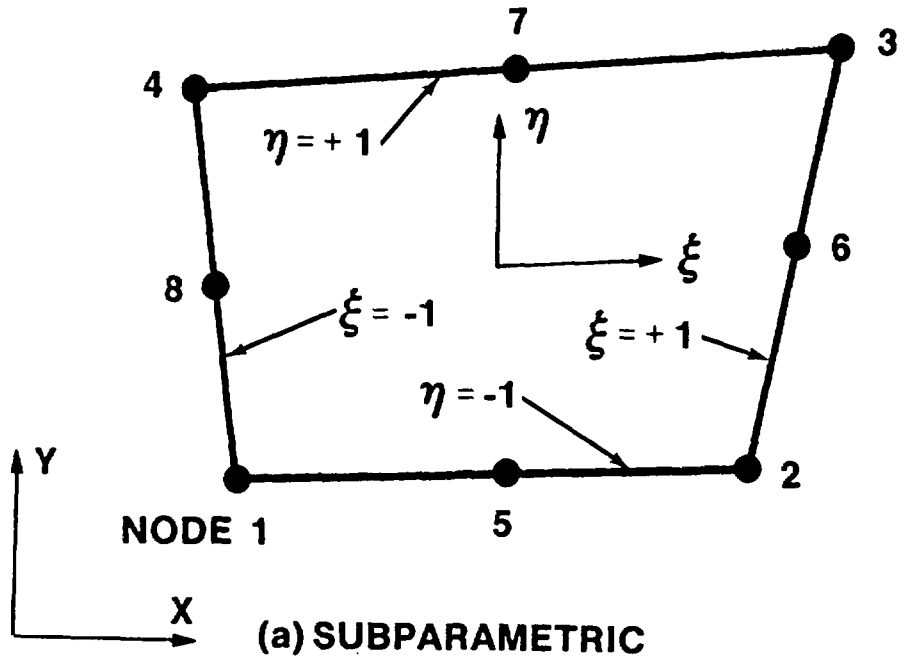
$$\begin{aligned}
\phi_1 &= (1-\xi-\eta)(1-2\xi-2\eta) \\
\phi_2 &= \xi(2\xi-1) \\
\phi_3 &= \eta(2\eta-1) \\
\phi_4 &= 4(1-\xi-\eta)\xi \\
\phi_5 &= 4\xi\eta \\
\phi_6 &= 4(1-\xi-\eta)\eta.
\end{aligned} \tag{21}$$

Here the subscripts refer to local node number, as shown in Figures 2 and 3.

The simplest conceivable problem domain is a square in the global coordinates, here either (x, z) or (r, z). If this domain were subdivided into some number of uniform square elements, the mapping of an element from the original problem domain to the standard domain would be accomplished by a translation and scaling. This is the simplest case of a subparametric mapping. Subparametric indicates that the mapping from the global domain to the standard domain is of lower order than the order of the basis functions themselves. For the eight-node quadrilateral element, the basis functions are of order two, *i.e.*, the basis functions are quadratic in the coordinates (ξ, η). Thus, any first-order, that is linear, transformation is allowed in a subparametric mapping. On the other hand, an isoparametric mapping is of the same order as that of the basis functions themselves. Thus, any second-order, that is quadratic, transformation is allowed in an isoparametric mapping. In practice, this means that subparametric elements have straight sides and isoparametric elements have parabolic sides. The same concepts are valid for triangles.

The finite element basis functions defined in Equations (19) and (20) have several properties that make them especially convenient:

- Each basis function is zero at all nodes except one, the node to which it corresponds, and has a value of unity there.
- Each basis function is continuous over the entire problem domain; therefore, any variable expanded in the set of finite element basis functions is continuous over the problem domain.
- Each basis function is non-zero over at most a few elements, typically four quadrilateral elements or six triangular elements.



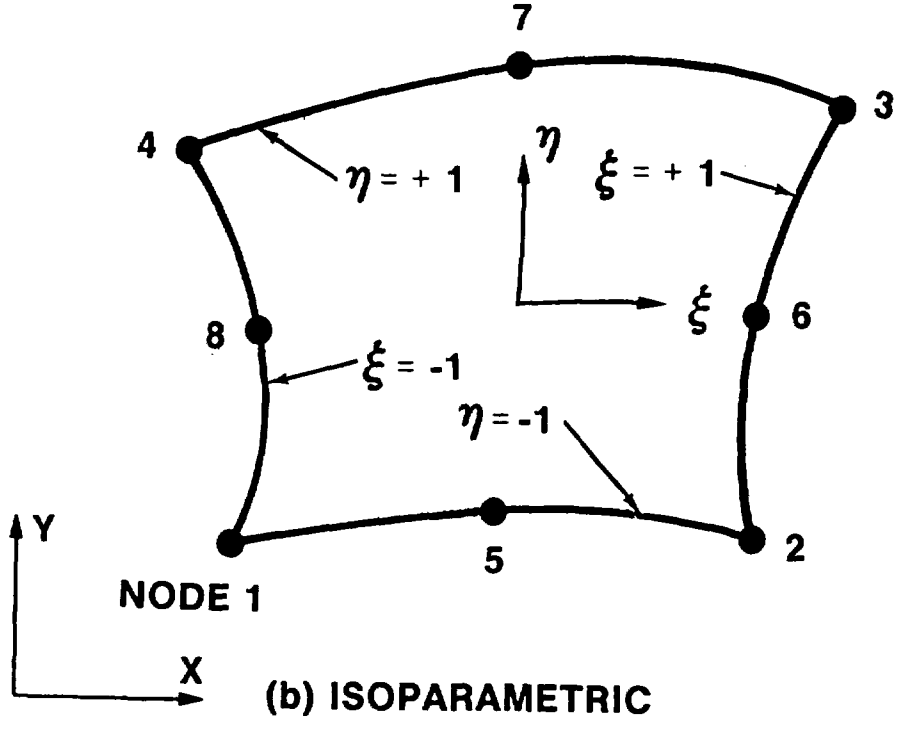
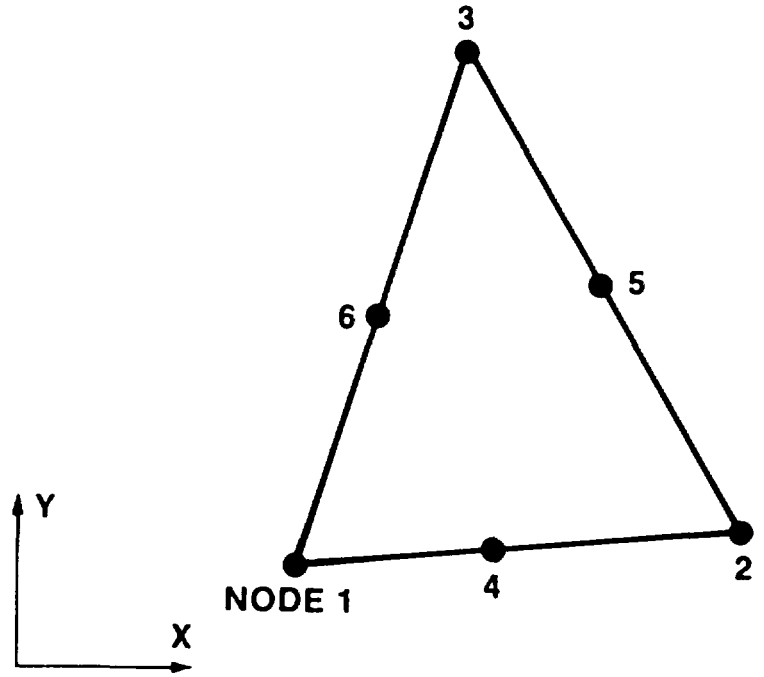
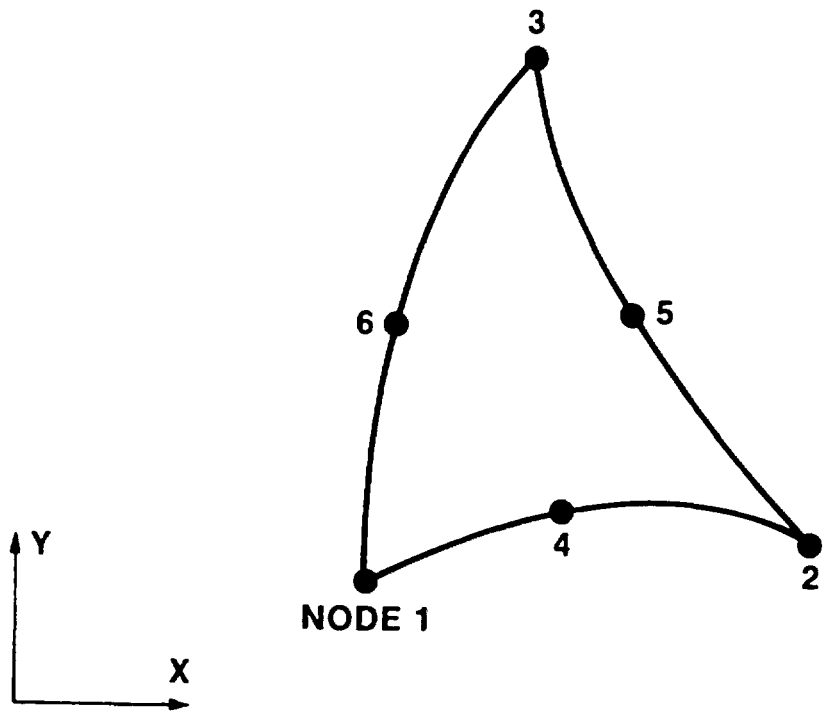


Figure 2: Quadrilateral Elements



(a) SUBPARAMETRIC



(b) ISOPARAMETRIC

Figure 3: Triangular Elements

The first property simplifies interpretation of results; the value of a dependent variable at a node point is simply the coefficient of the basis function corresponding to that node. The second property is mathematically essential in order to ensure the integrability of Equations (15)–(18). The third property means that resulting matrix equations will be sparse. Thus, special matrix-solvers can be used to reduce computational effort for direct elimination. The matrix-solver used here is a frontal method developed by Irons (1970) and generalized to handle asymmetric matrices by Gartling (1978).

3.3 Galerkin's Method

The preceding subsection defines the basis functions used to interpolate the dependent variables. Galerkin's method is to choose the same set of basis functions defined in Equations (19) and (20) as the weighting functions, the ψ_is, used in Equations (15)–(18). The set of all the ψ_is, called the global set, is the union of the sets of local basis functions for each of the elements making up a finite element mesh. Thus, the global set of ψ_is is identical to the global set of basis functions, the ϕ_is.

The expansions of the dependent variables, P, p_v, p_a, and T, in the global set of basis functions take the following forms:

$$P = \sum_{j=1}^{N} P_j \phi_j \tag{21}$$

$$p_v = \sum_{j=1}^{N} p_{v_j} \phi_j \tag{22}$$

$$p_a = \sum_{j=1}^{N} p_{a_j} \phi_j \tag{23}$$

$$T = \sum_{j=1}^{N} T_j \phi_j \tag{24}$$

where

N = the number of basis functions in the global set, which is the same as the number of node points in the finite element mesh,

P_j = a coefficient in the expansion of the effective liquid pressure, P,

p_{v_j} = a coefficient in the expansion of partial vapor pressure,

p_{a_j} = a coefficient in the expansion of partial air pressure, and

T_j = a coefficient in the expansion of temperature.

To simplify the expansions of the complicated nonlinear terms in Equations (15)–(18), groups of properties and unknowns are expanded in a single expansion in the ϕ_j. For example, the Darcy flow term in the vapor equation, Equation (16), is expanded as follows:

$$\int_\Omega \left[\frac{\rho_v \mathbf{K}_g}{\mu_g} \cdot \nabla(p_g + \rho_g g z)\right] \cdot \nabla\psi_i dS = \int_\Omega \left[\sum_{j=1}^N \left(\frac{\rho_v \mathbf{K}_g}{\mu_g}\right)_j \phi_j \cdot \nabla\left(\sum_{k=1}^N (p_g + \rho_g g z)_k \phi_k\right)\right] \cdot \nabla\phi_i dS. \tag{26}$$

The terms in parentheses followed by a subscript are evaluated at each of the node points of an element and expanded over that element using the same basis functions as those used in Equations (21)–(24). The advantage of the expansion shown in Equation (25) is that everything other than the basis functions can be taken outside of the integral so that element integrations can be performed once and for all.

$$\int_\Omega \left[\frac{\rho_v \mathbf{K}_g}{\mu_g} \cdot \nabla(p_g + \rho_g g z)\right] \cdot \nabla\psi_i dS = \sum_{j=1}^N \sum_{k=1}^N \left[\left(\frac{\rho_v \mathbf{K}_g}{\mu_g}\right)_j (p_g + \rho_g g z)_k\right] : \int_\Omega \phi_j \nabla\phi_k \nabla\phi_i dS \tag{27}$$

Each of the other expressions in Equations (15)–(18) is treated similarly. The final form of Equations (15)–(18) solved by NORIA is as shown below. The convention that repeated indices are summed is adopted here. Summed indices range from 1 to N as in Equations (15)–(18).

$$\left(\frac{\partial\Theta}{\partial t}\right)_j M_{ji} + (P - \rho_l g \beta z \Delta T)_k : \mathbf{M}_{jki} + (F_v)_j M_{ji} + \mathbf{j}_l \cdot \mathbf{n} M_i = 0$$

$$\left(\frac{\partial(\Phi-\Theta)\rho_v}{\partial t}\right)_j M_{ji} + \left(\frac{\rho_v \mathbf{K}_g}{\mu_g}\right)_j (p_g+\rho_g g z)_k : \mathbf{M}_{jki}$$
$$+ \left((\Phi-\Theta)\rho_v \frac{D_{Kv}D_{Ka}}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}}\right)_j (p_g+\rho_g g z)_k M_{jki}$$
$$+ \left((\Phi-\Theta)\rho_v \frac{(p_g/p_v)D_{Kv}D_D}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}}\right)_j (p_v+\rho_v g z)_k M_{jki}$$
$$- \left((\Phi-\Theta)\rho_g \frac{R_a D_{Kv}D_T}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}}\right)_j (T)_k M_{jki}$$
$$- (F_v)_j M_{ji} + \mathbf{j}_v \cdot \mathbf{n} M_i = 0$$

$$\left(\frac{\partial(\Phi-\Theta)\rho_a}{\partial t}\right)_j M_{ji} + \left(\frac{\rho_a \mathbf{K}_g}{\mu_g}\right)_j (p_g+\rho_g g z)_k : \mathbf{M}_{jki}$$
$$+ \left((\Phi-\Theta)\rho_a \frac{D_{Kv}D_{Ka}}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}}\right)_j (p_g+\rho_g g z)_k M_{jki}$$
$$+ \left((\Phi-\Theta)\rho_a \frac{(p_g/p_a)D_{Ka}D_D}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}}\right)_j (p_a+\rho_a g z)_k M_{jki}$$
$$+ \left((\Phi-\Theta)\rho_g \frac{R_v D_{Ka}D_T}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}}\right)_j (T)_k M_{jki}$$
$$+ \mathbf{j}_a \cdot \mathbf{n} M_i = 0$$

$$\left([\rho C]_{ave}\frac{\partial T}{\partial t}\right)_j M_{ji} - \left(C_l \frac{\mathbf{K}_l}{\mu_l}\right)_j (P - \rho_l g \beta z \Delta T)_k (T)_l : \mathbf{M}_{jkli}$$
$$- \left(C_v \frac{\rho_v \mathbf{K}_g}{\mu_g} + C_a \frac{\rho_a \mathbf{K}_g}{\mu_g}\right)_j (p_g+\rho_g g z)_k (T)_l : \mathbf{M}_{jkli}$$
$$- \left((\Phi-\Theta)C_v\rho_v \frac{D_{Kv}D_{Ka}}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}}\right)_j (p_g+\rho_g g z)_k (T)_l M_{jkli}$$
$$- \left((\Phi-\Theta)C_a\rho_a \frac{D_{Kv}D_{Ka}}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}}\right)_j (p_g+\rho_g g z)_k (T)_l M_{jkli}$$
$$- \left((\Phi-\Theta)C_v\rho_v \frac{(p_g/p_v)D_{Kv}D_D}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}}\right)_j (p_v+\rho_v g z)_k (T)_l M_{jkli}$$
$$- \left((\Phi-\Theta)C_a\rho_a \frac{(p_g/p_a)D_{Ka}D_D}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}}\right)_j (p_a+\rho_a g z)_k (T)_l M_{jkli}$$
$$+ \left((\Phi-\Theta)\rho_g \frac{C_v R_a D_{Kv}D_T - C_a R_v D_{Ka}D_T}{p_g D_D + p_v D_{Ka} + p_a D_{Kv}}\right)_j (T)_k (T)_l M_{jkli}$$
$$+ (\Lambda)_j (T)_k : \mathbf{M}_{jki} + (F_v L)_j M_{ji} - (Q)_j M_{ji} + \mathbf{q} \cdot \mathbf{n} M_i = 0$$

where **q** is the heat flux along boundaries and the subscripted matrix quantities are defined below.

$$M_i = \int_{\partial\Omega} \phi_i ds$$

$$M_{ji} = \int_{\Omega} \phi_j \phi_i dS$$

$$M_{jki} = \int_{\Omega} \phi_j \nabla\phi_k \cdot \nabla\phi_i dS$$

$$\mathbf{M}_{jki} = \int_{\Omega} \phi_j \nabla\phi_k \nabla\phi_i dS$$

$$M_{jkli} = \int_{\Omega} \phi_j \nabla\phi_k \cdot \nabla\phi_l \phi_i dS$$

$$\mathbf{M}_{jkli} = \int_{\Omega} \phi_j \nabla\phi_k \nabla\phi_l \phi_i dS$$

3.4 Boundary Conditions

Finite element boundary conditions are divided into two categories: essential and natural. An essential boundary condition is a Dirichlet boundary condition, *i.e.*, a boundary condition of the first kind; a natural boundary condition is any other type of boundary condition, *i. e.*, a Neumann (flux) or a Robin (mixed) boundary condition. A Robin boundary condition is one in which both a variable and its derivative appear. Treatment of essential boundary conditions is entirely different than that of natural boundary conditions in the finite element method. Essential boundary conditions are imposed exactly by forcing coefficients in the basis function expansions to take on the required values. For example, if the temperature along a boundary were to be set equal to $300K$, the coefficients of the basis functions used to expand temperature would be set to 300 for each of the nodes on that boundary. On the other hand, natural boundary conditions are never imposed exactly in the finite element method. Instead, prescribed flux values are inserted in the appropriate boundary integral in Equations (15)–(18). For example, if a boundary were prescribed to be adiabatic (no heat flux through the boundary) then the boundary integral term in Equation (18) would be set to zero there. No-flux boundary conditions are the default boundary conditions in NORIA. In

the absence of any boundary condition specified by the user, no mass fluxes and no conductive heat flux at all boundaries are the boundary conditions used in NORIA. Treatment of natural boundary conditions is one of the features that distinguishes finite element methods from finite difference techniques.

4 Time Integration Scheme

4.1 Strategy

The time integration scheme in NORIA was chosen because it is quite robust, automatically adjusts time-step size to accommodate variations in natural time scale, and is quite accurate (local time truncation error is third order). It uses a predictor-corrector scheme coupled with a Newton iteration procedure to march ahead in time (Gresho *et al.*, 1979). The Newton iteration scheme is essential in NORIA, given the highly nonlinear nature of the governing partial differential equations that it must solve. Furthermore, automatic time-step size adjustment is a requirement for solving two-phase porous-flow problems because natural time scales commonly vary by several orders of magnitude. The Jacobian matrix for the Newton procedure is constructed numerically by differencing each of the unknowns. Each step of the time integration procedure is described below. The overall scheme used here is very similar to one devised by Gresho *et al.* (1979). No steady state solution procedure is available in NORIA. Steady solutions can generally be obtained by constructing a transient calculation that approaches a steady state after some elapsed time. The problem with general purpose steady-state solvers is that such solvers are not likely to be convergent for the majority of problems that would be analyzed using NORIA.

4.2 Adams-Bashforth Predictor

The Adams-Bashforth predictor is described by Shampine and Gordon (1975) in their treatise on computer methods for initial value problems. The Adams-Bashforth predictor requires the dependent variables at the preceding time plane and their rates of change at the two preceding time planes:

$$(y_l^p)_{n+1} = (y_l)_n + \frac{\Delta t_n}{2}\left[\left(2 + \frac{\Delta t_n}{\Delta t_{n-1}}\right)(\dot{y}_l)_n - \frac{\Delta t_n}{\Delta t_{n-1}}(\dot{y}_l)_{n-1}\right] \tag{27}$$

where

$y_l =$ one of the coefficients in the expansions of the dependent variables, *i.e.*, $\{y_l\} = \{P_j, p_{vj}, p_{aj}, T_j\}$,

$j = 1, ..., N$,

$l = 1, ..., 4N$,

$\dot{y}_l =$ the rate of change of y_l,

$\Delta t_n = t_{n+1} - t_n$, and

$p =$ a predicted quantity.

Estimates for the rates of change of dependent variables are made by the following formulas.

$$(\dot{y}_l)_n = \frac{2}{\Delta t_{n-1}}[(y_l)_n - (y_l)_{n-1}] - (\dot{y}_l)_{n-1} \tag{28}$$

$$(\dot{y}_l)_{n-1} = \frac{\Delta t_{n-2}}{\Delta t_{n-1} + \Delta t_{n-2}}\left(\frac{(y_l)_n - (y_l)_{n-1}}{\Delta t_{n-1}}\right) + \frac{\Delta t_{n-1}}{\Delta t_{n-1} + \Delta t_{n-2}}\left(\frac{(y_l)_{n-1} - (y_l)_{n-2}}{\Delta t_{n-2}}\right) \tag{29}$$

Equation (28) is the common expression for rate of change using the trapezoid rule. However, it was found that using Equation (28) at time plane n and at time plane $n-1$ produces temporal oscillations that become quite noticeable as steady state is approached. By using Equation (29) to estimate the rate of change of the dependent variables at time plane $n-1$, the oscillations are eliminated.

The Adams-Bashforth predictor cannot be used on the first two time steps because the dependent variables are not assigned values prior to time plane zero, which is the initial condition. Thus a start-up procedure is required. The procedure used in NORIA is described in Subsection 4.6.

4.3 Trapezoid-Rule Corrector

A trapezoid-rule (Crank-Nicolson) corrector step is taken after each predictor step to improve the predicted solution. In addition, using both steps allows local time truncation error to be estimated so that time-step size can be adjusted. Automatic time-step size selection is described in Subsection 4.5.

Application of the trapezoid-rule step involves three parts.

- Time derivatives in Equations (15)–(18) are replaced by differences. For example,

$$\left(\frac{\partial \Theta}{\partial t}\right)_{n+1/2} \approx \frac{\Theta_{n+1} - \Theta_n}{\Delta t_n} \tag{30}$$

  where the subscript $n+1/2$ indicates that the time derivative is evaluated midway between time planes n and $n+1$.

- Dependent variables and functions of dependent variables in Equations (15)–(18) are evaluated midway between time planes. For example,

$$(T)_{n+1/2} \approx \frac{T_{n+1} + T_n}{2}, \tag{31}$$

$$(\mathrm{j}_v)_{n+1/2} \approx \mathrm{j}_v(P_{n+1/2}, p_{vn+1/2}, p_{an+1/2}, T_{n+1/2}). \tag{32}$$

- Equations (15)–(18) are solved for each of the coefficients, P_j, p_{vj}, p_{aj}, and T_j, used in the expansion of the dependent variables at the new time plane.

The predictor step provides an estimate of the dependent variables at the $n+1$ time plane for use in Equations (30) and (31). If more than one Newton iteration is required at a given time plane, the previous Newton iterate replaces the predicted solution at the new time plane. Once the trapezoid-rule corrector step is complete, the size of the next time step is calculated, as described below, and another time step is initiated.

4.4 Newton Iteration Procedure

The equations generated in the trapezoid-rule step described above are usually highly nonlinear. As a result, an iterative method is needed to solve them. Newton's method is used in NORIA because it has good convergence properties and exhibits a good domain of attraction. Newton's method requires the construction of a Jacobian matrix, which is accomplished numerically in NORIA by the following steps: (1) calculating residuals, *i.e.*, the right sides of Equations (15)–(18); (2) incrementing each of the finite element coefficients one at a time and recalculating the residuals; and (3) using the differences of residuals to estimate the rates of change with respect to the finite element coefficients. The computational work involved in constructing the Jacobian matrix is reduced by taking into account that the residuals in an element depend only on the nodal coefficients of that element. The resulting equations are then solved using a frontal method developed by Irons (1970) and generalized by Gartling (1978) to handle asymmetric matrices.

Several parameters control the operation of the Newton iteration process. The first parameter controls the size of the increments to the dependent variables used in calculating the Jacobian matrix. Normally, each of the dependent variables is incremented by 10^{-6} of the maximum absolute value in the field; for example, temperature is incremented by 10^{-6} of the maximum value of T in the field. Three other parameters are used to control the number of Newton steps taken at a time plane. Normally, only one

Newton step is required; however, if the difference between the predicted and corrected solutions is too great, more than one Newton step may be taken. This difference is defined by a weighted *rms* norm:

$$E = \sqrt{\sum_{j=1}^{N} \left[\left(\frac{P_j - P_j^p}{P_{max}} \right)^2 + \left(\frac{p_{vj} - p_{vj}^p}{p_{vmax}} \right)^2 + \left(\frac{p_{aj} - p_{vj}^p}{p_{vmax}} \right)^2 + \left(\frac{T_j - T_j^p}{T_{max}} \right)^2 \right]}. \qquad (33)$$

Here,

$max =$ the maximum absolute value of the variable at node points and

$p =$ the predicted solution or, if one or more Newton steps have already been taken, the previous iterate.

Normally, Newton iteration is continued until $E \leq 0.02$; however, no more than 3 Newton steps are allowed at a given time plane. Also, if $E \geq 0.05$, the time-step size is reduced, as described in the following subsection, and the time step is repeated. One additional parameter that controls the choice of time-step size is described in the next subsection. The values of these parameters are not easily, and ordinarily need not be, modified by the user. However, they may be modified when a problem so requires.

4.5 Automatic Time-Step Selection

Time-step size is varied at each time plane to maintain local time truncation error as uniform as possible. Taylor expansions are used to estimate local time truncation error for the Adams-Bashforth predictor and the trapezoid-rule corrector. The error estimate for the predictor is

$$\| y_{l\,n+1}^p - y_l(t_{n+1}) \| = \frac{1}{12} \left(2 + 3 \frac{\Delta t_{n-1}}{\Delta t_n} \right) \Delta t_n^3 \| \frac{\partial^3 y_l(t_{n+1})}{\partial t^3} \| + O(\Delta t^4). \qquad (34)$$

Here, $y_l(t_{n+1})$ refers to the exact solution of the finite element Equations (15)–(18) at time t_{n+1}. The double vertical bars indicate the *rms* norm defined in Equation (33). The error estimate for the corrector is

$$\| y_{l\,n+1} - y_l(t_{n+1}) \| = \frac{1}{24} \Delta t_n^3 \| \frac{\partial^3 y_l(t_{n+1})}{\partial t^3} \| + O(\Delta t^4). \qquad (35)$$

By combining Equations (34) and (35) to eliminate the norm of the third time derivative of y_l and by using the triangle inequality to eliminate the exact solution $y_l(t_{n+1})$, the following inequality results:

$$d_{n+1} \equiv \|y_{l\,n+1} - y_l(t_{n+1})\| \leq \frac{\|y_{l\,n+1} - y^p_{l\,n+1}\|}{3 + 6\Delta t_{n-1}/\Delta t_n} + O(\Delta t_n^4). \tag{36}$$

Here, d_{n+1} is the error in the corrected solution at time plane $n+1$. Finally, Equation (35) implies that

$$\frac{d_{n+2}}{d_{n+1}} = \left(\frac{\Delta t_{n+1}}{\Delta t_n}\right)^3 + O(\Delta t_n), \tag{37}$$

which can be solved for Δt_{n+1} to get

$$\Delta t_{n+1} = \Delta t_n \left(\frac{\epsilon}{d_{n+1}}\right)^{1/3}. \tag{38}$$

The upper bound on d_{n+1} from Equation (36) is used in Equation (38) to establish a lower bound on Δt_{n+1}. Here, d_{n+2} has been set to ϵ, which is the desired value of the local time truncation error at the next time step. The default value of ϵ in NORIA is 0.001. Equation (38) is used to calculate time-step size for each time step beginning with $n = 4$. However, when the Δt_{n+1} is less than 85% of Δt_n, the solution at time plane $n+1$ is recalculated and Δt_n is replaced by the smaller value.

4.6 Start-Up Procedure

The Adams-Bashforth predictor cannot be used on the first two time steps because rates of change of the dependent variables are not known prior to the initial condition. As a result, a start-up procedure is required to initiate the time integrator. The one used here is to take two backward-difference steps before initiating the two-step time integration procedure. The reason backward difference is chosen instead of trapezoid-rule is that backward-difference steps help damp out discontinuities that may be present in the initial data. Luskin and Rannacher (1982) have shown that a few backward-difference steps taken at the beginning of a time integration procedure can actually enhance the accuracy of the overall results by damping singularities that might otherwise cause oscillations.

Automatic time-step size selection begins after the first predictor-corrector step, which is the third time step. The only exception is that if the user requests that

output be printed at a particular time, time-step size may be limited so that results can be computed at that exact value of time. User-requested output is described further in Section 6.

5 Program Description

5.1 Organization

The program organization in NORIA reflects the steps taken in setting up, solving, and evaluating the finite element analysis of water, vapor, air, and energy transport through a porous medium. NORIA includes its own mesh generator and plotting packages. NORIA contains about 100 routines that can be categorized according to five primary groups of tasks, which are described in detail in Subsections 5.2–5.6.

Mesh Generation. Assign nodal point locations. Assign nodal points to finite element and organize data for equation formulation. The finite element library in NORIA consists of four element types.

Boundary and Initial Conditions. Assign boundary conditions to appropriate nodal points and select initial conditions.

Solution Procedure. Form element coefficient matrices. Use material property data to assemble local residuals and matrices. And assemble local residuals and matrices into global ones. Solve transient equations for dependent variables.

Calculation of Derived Quatities. Postprocess solution field data to calculate heat fluxes or water, vapor, and air velocities.

Plotting. Plot mesh, solution field, and postprocessed data. Plotting options include grid points, elements, and solution field contours, profiles, and histories.

NORIA is designed for use on Cray Research Corporation's S1 computer. The variables are dimensioned to accommodate up to 500 elements; however, NORIA can be redimensioned to handle larger meshes.

5.2 Mesh Generation

Generation of node points in NORIA is independent of element specification. As a result, a user may generate more nodes than are actually used in a problem so that he may experiment with node point location before selecting a mesh. This option is especially useful when a large or complicated domain is to be subdivided into elements.

For purposes of mesh generation, a problem domain is considered to be made up of regions determined by the user. Region boundaries are approximated by curves that can consist of linear, quadratic, or cubic mappings of a line into an (x,z) or (r,z) coordinate system, *i.e.*,

$$x = P_1(\xi), \quad z = P_2(\xi)$$

or

$$r = P_1(\xi), \quad z = P_2(\xi).$$

Here, P_1 and P_2 are polynomials that may be linear, quadratic, or cubic, and ξ is a parametric coordinate. This approach allows relatively complicated boundary shapes to be modeled easily and accurately.

The user locates node points along the boundaries of a region by specifying the number of nodes along each region boundary and by specifying a geometrical factor that determines relative spacing. Node points within a region are generated automatically once the boundary nodes have been identified by an (I, J) numbering system. The user may specify node points along curves or at individual points. Node points may also be located by means of a subroutine supplied by the user.

The final step in mesh generation is to assign node points to elements. In the case of isoparametric elements, the element shape is determined by all of the nodes assigned to that element; in the case of subparametric elements, the element shape is determined by corner nodes. The midside nodes in subparametric elements are located midway on the line segment that connects adjacent corner nodes. In general, element boundaries that border a region do not coincide with the region boundary itself. However, isoparametric elements better approximate a curved boundary than do subparametric elements.

Two distinct errors are introduced in NORIA when approximating curved boundaries for a problem domain. The first error is in the step of approximating problem domain boundaries by region boundaries. The second error is in the step of approximating region boundaries by element boundaries. However, these errors are generally minor and disappear altogether when problem domains have straight boundaries. Furthermore, the first error can be reduced by breaking up an irregular problem domain into more and more regions and can be completely eliminated by assigning boundary node locations individually, either by means of the point option mentioned above or the user-supplied subroutine. The second error is inherent in the finite element method, but, as shown by Strang and Fix (1973), with certain mild restrictions on the shapes of elements, this error is of no higher order than the error made by representing the solution in terms of the finite element basis functions.

5.2.1 Element Library

Four basic element types are available in NORIA.

Isoparametric Eight-Node Quadrilateral. The isoparametric eight-node quadrilateral (QUAD8/8) can have curved sides, as shown in Figure 2, and uses the basis functions defined in Equation (19) to interpolate dependent variables and to map from the standard domain to a real domain.

Subparametric Eight-Node Quadrilateral. The subparametric eight-node quadrilateral (QUAD8/4) has straight sides and uses the basis functions defined in Equation (19) to interpolate dependent variables and bilinear basis functions, σ_j, to map from the standard domain to a real domain. Bilinear basis functions are defined as

$$\begin{aligned} \sigma_1 &= \tfrac{1}{4}(1-\xi)(1-\eta) \\ \sigma_2 &= \tfrac{1}{4}(1+\xi)(1-\eta) \\ \sigma_3 &= \tfrac{1}{4}(1+\xi)(1+\eta) \\ \sigma_4 &= \tfrac{1}{4}(1-\xi)(1+\eta). \end{aligned} \tag{40}$$

Isoparametric Six-Node Triangle. The isoparametric six-node triangle (TRI6/6) can have curved sides, as shown in Figure 3, and uses the basis functions defined in Equation (20) to interpolate dependent variables and to map from the standard domain to a real domain.

Subparametric Six-Node Triangle. The subparametric six-node triangle (TRI6/3) has straight sides and uses the basis functions defined in Equation (20) to interpolate dependent variables and linear basis functions, σ_j, to map from the standard domain to the real domain. Linear basis functions are defined as

$$\begin{aligned} \sigma_1 &= (1-\xi-\eta) \\ \sigma_2 &= \xi \\ \sigma_3 &= \eta. \end{aligned} \tag{41}$$

Isoparametric and subparametric transformations of an element are respectively defined as

$$x = \sum_{j=1}^{M} x_j \phi_j; \quad y = \sum_{j=1}^{M} y_j \phi_j; \tag{41}$$

$$x = \sum_{j=1}^{M} x_j \sigma_j; \quad y = \sum_{j=1}^{M} y_j \sigma_j. \tag{42}$$

Here, (x_j, y_j) are the nodal coordinates of an element in the real domain and M is the number of basis functions used in the expansion over an element.

Evaluation of the integrals in Equations (15)–(18) is carried out over standard elements by using the transformations defined in Equations (41) and (42). This evaluation requires that the Jacobian of the mapping be calculated. Integration is performed by quadrature: nine-point Gaussian quadrature is used on the quadrilateral elements and a seven-point quadrature developed by Hammer *et al.* (1956) is used on triangular elements.

5.3 Boundary and Initial Conditions

Boundary conditions are specified on an element basis. Effective liquid pressure, partial vapor pressure, partial air pressure, or temperature can be specified for individual nodal points or element sides. Use of the latter is restricted to cases where there is a uniform value along a side. Mass or conductive heat fluxes are specified as constants along element sides. Zero-mass-flux and zero-conductive-heat-flux conditions are the default boundary conditions in NORIA, *i.e.*, no boundary condition need be specified for an impermeable adiabatic surface. Moreover, any of the above types of boundary conditions may be specified as functions of any of the dependent or independent variables by means of user-supplied subroutines. A volumetric heat source may also be defined for any material, either as a constant over that material or as a function of any of the dependent or independent variables by means of a user-supplied subroutine.

Two options exist in NORIA for specification of initial conditions. The dependent variables may be specified in the input deck to be constants over each material. Arbitrary initial conditions may be input by a user-supplied tape file written in a format specified in Section 6.

5.4 Solution Procedure

The solution procedure used in NORIA is described in Section 4. The trapezoid-rule corrector is implemented on an element-by-element basis. The solution process is performed by Irons' frontal procedure (Irons, 1970), which has five distinct steps: (1) the elimination process is set up for the frontal solution procedure; (2) a residual vector and an element coefficient matrix are assembled for an element; (3) the local residual vector and the element coefficient matrix are respectively added to a global residual vector and coefficient matrix; (4) equations that have been fully summed are forward-eliminated; steps (2) through (4) are repeated until all elements have been treated; and (5) back substitution is carried out to produce the solution vector at the

new time plane. This process is repeated for each time plane and may be repeated several times for a single time plane, as described in Section 4.

5.5 Calculation of Derived Quantities

Quite frequently the user wishes to examine or plot quantities derived from the basic field variables. Options in NORIA include calculation of heat fluxes and determination of water, vapor, and air velocities. Derived quantities must be calculated before creating plots.

5.6 Plotting

NORIA contains a plotting package that can generate plots of nodal point locations, finite element mesh, outlines of materials, contours, histories, and profiles. Any of the dependent variables or moisture content can be contoured. History and profile plots can be drawn for the dependent variables, moisture content, velocities, or heat fluxes. In addition to these plotting options, NORIA results can be plotted by TRINITY, a versatile pre- and postprocessing package written by Gartling (1985).

6 Input Guide

6.1 Introduction

The structure of an input deck for NORIA corresponds to the steps required to formulate and solve a finite element problem. Through a series of command and data cards, the program is directed to such functions as generation of nodal points, construction of elements, construction of element-coefficient matrices, solution of the finite element equations, and postprocessing of solution field data. The actual sequence of commands to the program is quite flexible, although there are some obvious limitations to the order in which the operations can be carried out. In the following subsections, the command and data cards required by NORIA are described in roughly the same order that they would normally appear in an input deck.

The conventions listed below are used in the description of input cards.

- Upper-case words imply an alphanumeric input value, *e. g.*, FORMKF.
- Lower-case words imply a numerical value of the specified variable, *e. g.*, $xmax$.
- All numerical variables are input in a free-field format and successive variables are separated by commas. All input data are limited to eight characters under this format, because **numerical data that exceeds eight characters are truncated after the eighth character!**
- [ ] indicates optional parameters that may be omitted by using successive commas in a variable list. If the omitted parameter is not followed by another parameter, no commas need be used.
- $\prec \succ$ indicates the default value for an optional parameter.
- An asterisk (*) may be used to continue a variable list on a second data card. When using this option, the comma that follows the last variable and precedes the asterisk is omitted.
- A dollar sign ($) may be used to end a data card so that the remaining space on the card may be used for comments.
- The contents of each input card are indicated by underlining.
- All quantities associated with a coordinate direction are expressed in terms of the planar (x, z) coordinate system. The corresponding quantities for axisymmetric problems are input by replacing the horizontal coordinate, x, with the radial coordinate, r.

The command cards are described in the following order:

- Header card
- SETUP command card
- FORMKF command card
- OUTPUT command card
- UNZIPP command card
- HEATFLUX command card
- VELOCITY command card
- PLOT command card
- RESTART command card
- Program termination card

In the following subsections, the individual descriptions of the command cards are discussed, followed by descriptions of the input deck structure, user subroutines, initial conditions, error messages, and computer requirements.

6.2 Header Card

The header card must be the first card in a deck. If two or more problems are run in sequence, the header card follows the END, PROBLEM card of the previous problem. A $ must appear in the first column; the remaining 79 columns are available for a problem title. The header card has the following format.

$ PROBLEM TITLE

6.3 SETUP Command Card

The first task in formulating a finite element analysis using NORIA is to specify the material properties and to define the finite element mesh and boundary conditions. These functions are accomplished by means of the SETUP command and three sets of associated data cards.

The SETUP command card has the following format:

SETUP, [*iprint*], [*maxi*], [*order*], [*gridplot*]

where

iprint $\prec 2 \succ$ determines the amount of printout produced during the setup operation. Output increases with the value of *iprint* and ranges from no printout for *iprint* = 1 to full printout for *iprint* = 4.

maxi $\prec 18 \succ$ is the maximum number of I rows of node points to be generated. *Maxi* need only be specified if there are more than 18 I-rows or more than 277 J-rows. The limit on the maximum I- and J-rows is $I \cdot J \leq 5000$.

order $\prec \succ$ determines the numbering of the elements. For the default (order left blank), the elements are numbered by increasing (I, J) values [*e. g.*, $(1,1)$, $(2,1)$, $(3,1)$,..., $(1,2)$, $(2,2)$,...]. For *order* = PRESCRIBED, the elements are numbered according to their order in the input list. The elements should be ordered so that the front width of the problem is minimized (Irons, 1970).

grid plot $\prec \succ$ determines whether a grid point plot tape should be written. If a plot of the grid points is to be made in a subsequent call to the plot routine, then *grid plot* = PLOT; otherwise *grid plot* is left blank.

Following the SETUP command card, three sets of data cards are required. These data sets specify material properties, nodal point locations, and finite element mesh and boundary conditions. Each of the data sets is terminated by an END card. The third END card terminates the SETUP command and readies NORIA for the next command.

6.3.1 Material Data Cards

Material data cards are of two types—one for fluid properties and one for solid properties. Data cards for fluid properties have the following form:

[*material name*], *number*, ρ_l, μ_f, c_f, λ_f, R_c, D_{cK}, D_{va}, D_T, g, L, β_l, T_{ref}

where

material name is an optional alphanumeric material name for user reference.

number is an internal reference number for the material. Numbers 1 through 3 are reserved for water, vapor, and air, respectively.

ρ_l is the density of water at reference temperature, T_{ref}. This parameter is ignored in the property cards for vapor and air because gas densities are calculated by the ideal gas law.

μ_f is the dynamic viscosity of a pure fluid component.

c_f is the heat capacity of a pure fluid component.

λ_f is the thermal conductivity of a pure fluid component.

R_c is the ideal gas constant divided by the molecular mass of component c, either water vapor or air. This parameter is ignored on the property card for water.

D_{cK} is the Knudsen diffusion coefficient for pure component, c. This parameter is ignored on the property card for water.

D_{va} is the binary diffusion coefficient for a mixture of vapor and air. This parameter need be included only on the property card for vapor.

D_T is the thermo-diffusion coefficient for a vapor and air mixture. This parameter need be included only on the property card for vapor.

g is the acceleration caused by gravity. This parameter need be included only on the property card for water.

L is the latent heat of vaporization for water. This parameter need be included only on the property card for water.

β_l is the thermal expansion coefficient for water. This parameter need be included only on the property card for water.

T_{ref} is the reference temperature that corresponds to the density of water given above.

Most of the above parameters can either be set to a constant value or to VARIABLE. Subroutines must be supplied by the user to specify variable properties as functions of dependent and independent parameters, as described in Subsection 6.13. The following fluid parameters must be set to constant values: ρ_l, R_c, g, and T_{ref}. Default values for all parameters omitted on fluid property cards are 0.

Material properties for solid matrix materials are specified according to the following format:

$$\underline{[material\ name],\ number,\ \rho_s,\ c_s,\ \lambda_{11},\ \lambda_{22},\ \alpha,\ \Phi,\ K_{l11},\ K_{l22},\ K_{g11},\ K_{g22},\ S,}$$
$$\underline{T_0,\ P_0,\ p_{v0},\ p_{a0}}$$

where

material name is an optional alphanumeric material name for user reference.

number is an internal reference number for the material. Solid materials may use any or all of the numbers 4 through 13.

ρ_s is the grain density of solid material.

c_s is the heat capacity of a pure solid material, *i.e.*, the heat capacity of a grain of solid.

λ_{11} is the thermal conductivity of pure solid material in the direction of the first principal axis.

$\lambda_{22} \prec \lambda_{11} \succ$ is the thermal conductivity of pure solid material in the direction of the second principal axis.

$\alpha \prec 0° \succ$ is the angle measured from the x-axis to the first principal axis in the clockwise direction.

Φ is the porosity of a solid matrix. A material is treated as impermeable when Φ is set to zero.

K_{l11} is the matrix permeability to liquid in the direction of the first principal axis.

$K_{l22} \prec K_{l11} \succ$ is the matrix permeability to liquid in the direction of the second principal axis.

K_{g11} is the matrix permeability to gas in the direction of the first principal axis.

$K_{g22} \prec K_{g11} \succ$ is the matrix permeability to gas in the direction of the second principal axis.

$S \prec 0 \succ$ is the volumetric heat output for a material. This parameter can either be set to zero (by inserting a 0 or by including two consecutive commas) when there is no volumetric heating within a material; or it can be set to a constant or to VARIABLE.

T_0 is the initial temperature in a material. Temperature must be measured in absolute units.

P_0 is the initial effective liquid pressure in a material.

p_{v0} is the initial partial vapor pressure in a material.

p_{a0} is the initial partial air pressure in a material.

Again, most of the parameters can either be set to a constant value or to VARIABLE. Variable properties are determined in user-supplied subroutines described in Subsection 6.13. The following solid parameters must be set to constant values: ρ_s, α, Φ, T_0, P_0, p_{v0}, and p_{a0}. Default values are zero unless otherwise specified. An END command must follow the material property portion of the SETUP command.

END

Porous materials modeled by NORIA can have either isotropic or orthotropic permeability and thermal conductivity. For isotropic materials (materials in which permeability and conductivity are not functions of direction), permeability and conductivity are specified by setting $K_{11} = K$ and $\lambda_{11} = \lambda$. K_{22}, α, and λ_{22} are left blank (*i.e.*, by inserting two consecutive commas). For orthotropic materials (materials in which permeability and conductivity are functions of direction), permeability and conductivity are specified by their values along principal axes: K_{11} and λ_{11} are the respective values of permeability and thermal conductivity in the direction of the first principal axis; K_{22} and λ_{22} are the respective values of permeability and thermal conductivity in the direction of the second principal axis; and α is the angle measured in the clockwise direction from the x-axis to the first principal axis. The angle α need be specified only when the principal axes are not aligned with the coordinate axes.

NORIA does not contain any dimensional constants or properties; therefore, the user is free to choose any consistent set of units that he wishes. Furthermore, it is possible to cast the governing equations, (1)–(4) and (6)–(8), in dimensionless form. If a dimensionless form is used, some of the parameters in the material property cards would be replaced by dimensionless groups and some would be replaced by unity. This aproach might be useful if large-scale parametric studies were to be performed.

6.3.2 Nodal-Point Data

After material properties have been specified, nodal points for the finite element mesh are generated. Generation of nodal points and generation of elements are distinct

operations in NORIA. Nodal-point locations are calculated using quadrilateral regions. The boundaries of the regions are determined by polynomial mappings, as described in Subsection 5.2. The user defines each region by specifying a set of points that lie on the boundary of the region. Nodal points are identified by an (I, J) numbering system. The location of nodal points within a region is entirely controlled by the number of points along each of the boundaries of the region and by geometric parameters that control relative nodal spacing along each of the boundaries. Three data cards are needed to define the nodal points in a region.

$$\underline{i_{min},\ j_{min},\ i_{max},\ j_{max},\ [g_1],\ [g_2],\ [g_3],\ [g_4],\ [\text{POLAR}],\ [x_0],\ [y_0]}$$

$$\underline{x_1,\ x_2,\ x_3,\ x_4,\ [x_5],\ [x_6],\ldots,[x_{12}]}$$

$$\underline{y_1,\ y_2,\ y_3,\ y_4,\ [y_5],\ [y_6],\ldots,[y_{12}]}$$

where

i_{min}, j_{min} are the minimum values of I and J for the region being generated (Figure 4a).

i_{max}, j_{max} are the maximum values of I and J for the region being generated (Figure 4a). The difference between the maximum and minimum values determines the number of node points in the region.

$g_i \prec 1 \succ$ specifies a geometric factor for nodal spacing along the ith side of the region (Figure 4a). The geometric factors are defined by

$$g_1 \equiv \frac{\Delta i_1}{\Delta i_2} \tag{43}$$

$$g_2 \equiv \frac{\Delta j_1}{\Delta j_2} \tag{44}$$

and are illustrated in Figure 4a. The definitions for g_3 and g_4 are the same as those for g_1 and g_2, respectively. The default values of unity give equal nodal spacing along a side. Geometric factors that are larger or smaller than unity give geometric nodal spacings (*i.e.*, the distances between consecutive nodes form a geometric progression with geometric factor g_i).

POLAR specifies the use of a polar rather than a Cartesian coordinate system. The default is a Cartesian coordinate system if this parameter is omitted.

$(x_0 \prec 0 \succ, y_0 \prec 0 \succ)$ designates the origin of a polar coordinate system.

(x_1, y_1)–(x_{12}, y_{12}) define the coordinates of the four corner and optional side points for each region. If a region is bounded by straight lines, only the coordinates of the four corners need be specified (*i.e.*, x_1–x_4 and y_1–y_4). If any of a region's sides are curved, then the appropriate side nodes must be specified as shown in Figure 4b. A quadratic mapping is used to define a boundary if one side point is specified; a cubic mapping is used to define a boundary if two side points are specified. Points should be nearly equally spaced along each side of a region.

There are no limits on the number of regions that may be used to define a set of nodal points. The only restriction is that $I_{max} \cdot J_{max} \leq 5000$.

Nodal points may be generated for triangular regions by assigning the same coordinates to two adjacent corner points. However, when this method is used, the location of interior node points is sometimes unpredictable. The user should verify the quality of such a mesh by creating a node-point plot as described below.

It is sometimes convenient to be able to position an individual nodal point or a string of nodes. The following data cards allow the user to implement these options.

POINT, i, j, x_1, y_1, [POLAR], [x_0], [y_0]

where

(i, j) is the (I, J) name for the point.

(x_1, y_1) is the coordinate pair that defines the location of the point.

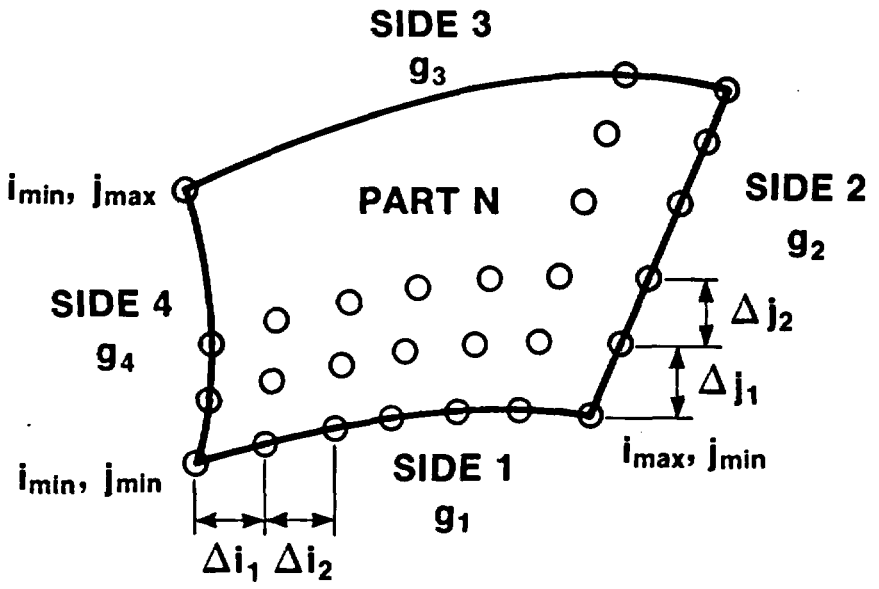
POLAR specifies the use of a polar rather than a Cartesian coordinate system.

$(x_0 \prec 0 \succ, y_0 \prec 0 \succ)$ designates the origin of a polar coordinate system.

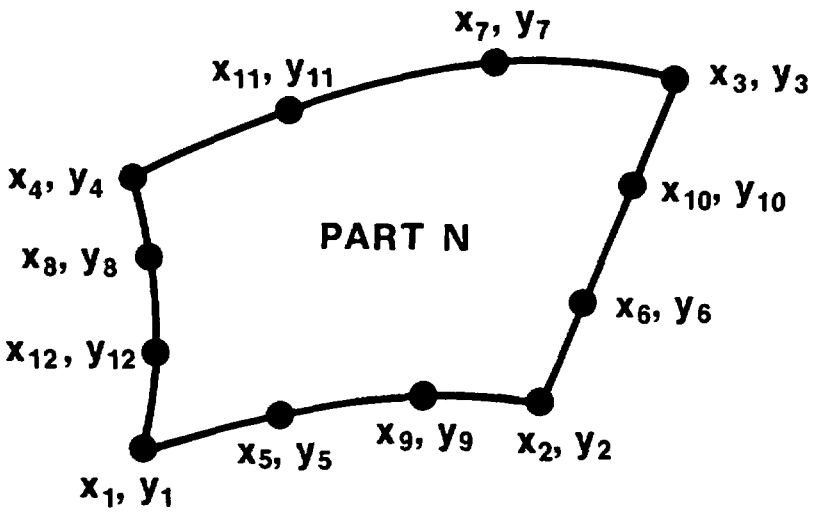
ARC, i_{min}, j_{min}, i_{max}, j_{max}, [g_1], [POLAR], [x_0], [y_0]

x_1, x_2, [x_3], [x_4]

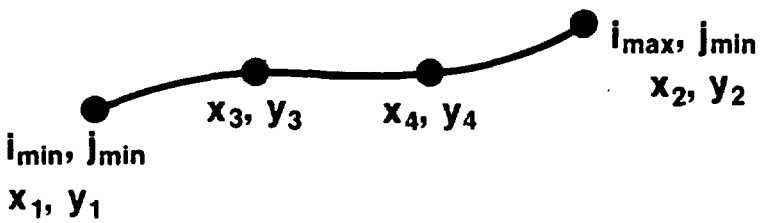
y_1, y_2, [y_3], [y_4]



(a) I, J NOMENCLATURE



(b) COORDINATE NOMENCLATURE



(c) ARC NOMENCLATURE

Figure 4: Nomenclature for Generation of Nodal Points

where

i_{min}, j_{min} are the minimum values of I and J for the arc, as shown in Figure 4c.

i_{max}, j_{max} are the maximum values of I and J for the arc, as shown in Figure 4c. Because a one-dimensional array of node points is generated by this command, either i_{min} must equal i_{max} or j_{min} must equal j_{max}. The difference between the maximum and minimum values determines the number of nodal points generated along the arc.

$g_1 \prec 1 \succ$ specifies the geometric factor for nodal spacing along the arc. The gradient is defined either by Equation (43) or Equation (44), depending on whether I or J is being incremented along the arc.

POLAR specifies the use of a polar rather than a Cartesian coordinate system.

$(x_0 \prec 0 \succ, y_0 \prec 0 \succ)$ designates the origin of a polar coordinate system.

(x_1, y_1)-(x_4, y_4) define the coordinates of the ends of the arc and optional intermediate points. If the arc is a straight line, only the first two sets of coordinates need be specified; if the arc is curved, either one or two intermediate points should be specified, as shown in Figure 4c.

There are no limits to the number of POINT and ARC data cards that may be used in generating a mesh. Both types of commands may appear anywhere within the nodal data portion of the SETUP command.

One further option exists for defining nodal point locations. Nodal coordinates can be defined by the user in a subroutine called EXTDEF. This subroutine provides a great deal of flexibility and is described in Subsection 6.13. Subroutine EXTDEF is accessed by placing the following card anywhere within the nodal data portion of the SETUP command.

EXTDEF

An END command must follow the nodal data.

END

6.3.3 Element and Boundary Data

Following the generation of nodal points, NORIA is ready to accept element and boundary condition data. Because the nodal points are generated independently of the elements, selection of nodes to construct an element is very flexible. The process of constructing an element consists of identifying a group of nodal points to serve as corner and midside nodes of the element. This concept is apparent from the form of an element data card.

$$\underline{\textit{element type}, \textit{mat}, i_1, j_2, [i_2], [j_2], \ldots, [i_n], [j_n]}$$

where

element type is an alphanumeric name for the type of element. The possible element types are described below.

mat is the matrix material number for the element. This number should correspond to one of the material numbers that appears on a matrix material property card.

(i_1, j_1), $[(i_2, j_2)$,...] are the (I, J) values for the node points in the element. The nodes are listed counterclockwise around an element, starting with any corner, as shown in Figure 5. In most situations, the list of (I, J) values may be significantly condensed. When (I, J) is specified only for the first node, the following values for the remaining nodes are assumed:

$$i_4 = i_8 = i_1$$

$$i_5 = i_7 = i_1 + 1$$

$$i_2 = i_3 = i_6 = i_1 + 2$$

$$j_2 = j_5 = j_1$$

$$j_6 = j_8 = j_1 + 1$$

$$j_3 = j_4 = j_7 = j_1 + 2.$$

When (I, J) values are specified for corner nodes only, the (I, J) values for each midside node are taken to be the average of the values at the adjacent corners.

An element's type is specified by the *element type* parameter. The following element types may be used in NORIA.

- QUAD8/4 is a subparametric quadrilateral with straight sides oriented arbitrarily.
- QUAD8/8 is a general isoparametric quadrilateral that uses a quadratic mapping to determine the shape of element sides.
- TRI6/3 is a subparametric triangle with arbitrarily oriented straight sides.
- TRI6/6 is a general isoparametric triangle that uses a quadratic mapping to determine the shape of element sides.

In the generation of subparametric elements, physical coordinates (x, y) of midside nodes need not lie precisely on the element side because these coordinates are not used in element construction. The basis functions defined in Equations (19) and (20) are used to interpolate unknowns over quadrilateral and triangular elements, respectively.

Two points about the (I, J) identification of an element are noteworthy. Each element is identified internally by the (I, J) pair of the first node named on the element data card, *i.e.*, (i_1, j_1). Because any corner node may be named first on an element data card, the internal identification may not be unique. The user must avoid assignment of a duplicate (I, J) identifier to any element that has an imposed boundary condition. This problem can always be avoided by selecting an appropriate nodal ordering on the relevant element data cards. The second point is that element connectivity is determined by nodal (I, J) values. The problem is that in some situations it is convenient to assign more than one (I, J) identifier to a single node point. The user must make sure that elements having a common node point use the same (I, J) identifier for that node. Otherwise, the elements will not be properly connected.

Boundary conditions are specified by element and may appear anywhere after the element to which the boundary condition applies has been defined and before the end of the element and boundary condition portion of the SETUP command. Essential boundary conditions can be specified as being constant along an element side or by node; natural boundary conditions can be specified only as being constant along element sides. The format for prescribing boundary conditions is

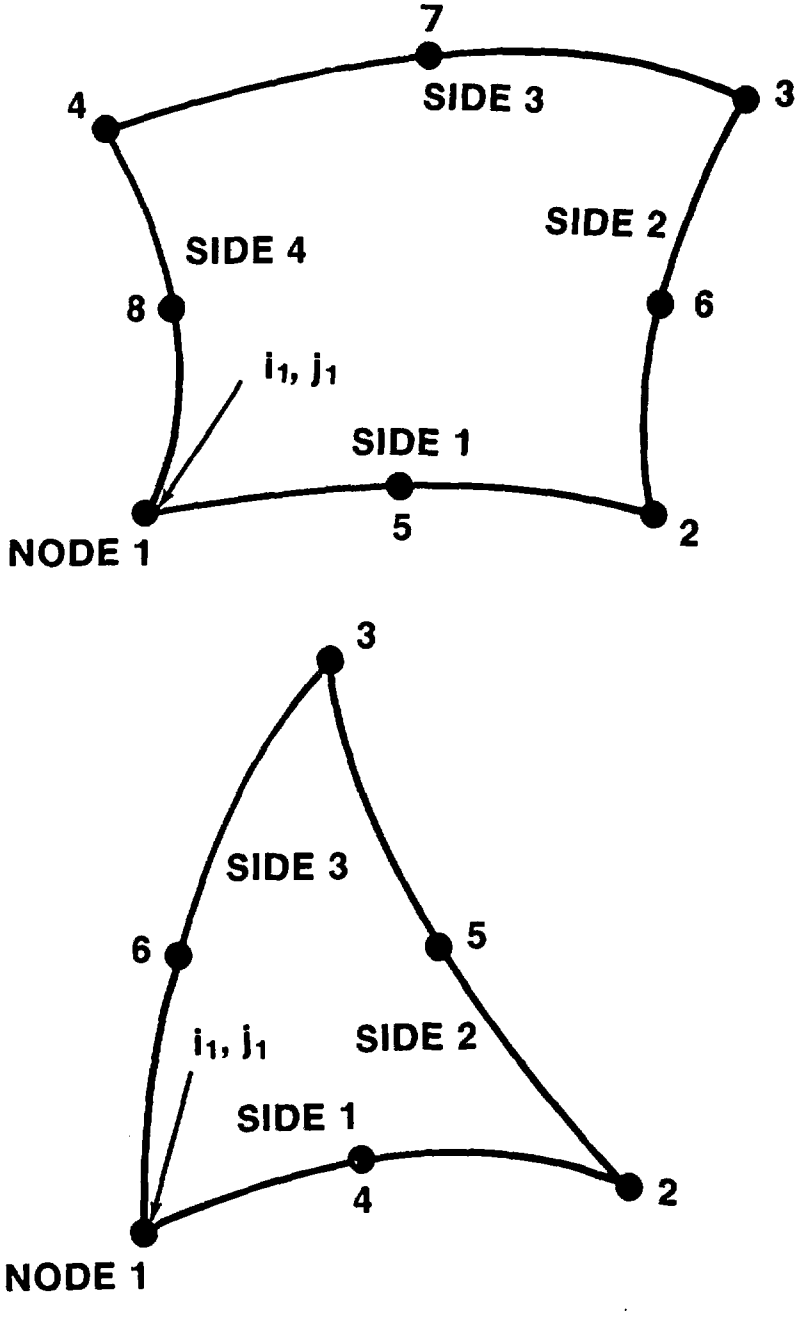


Figure 5: Numbering of Element Nodes and Sides

BC, *b. c. type*, i_1, j_1, *side/node*, *value/curve no.*

where

b. c. type is an alphanumeric name for the type of boundary condition. The possible types are described below.

(i_1, j_1) is the (I, J) identification of the element to which the boundary condition applies, *i.e.*, the first (I, J) pair named on the element data card.

side/node identifies the side or node of the element to which the boundary condition is to be applied. The numbering of nodes and sides begins with the identifying node, *i.e.*, the first node named on the element data card, and proceeds counterclockwise, as shown in Figure 5.

value/curve no. is the numerical value of the applied boundary condition or the number of the CURVE subroutine from which the value of the boundary condition will be determined. CURVE subroutines are described further in Subsection 6.11.

The choices of *b. c. type* that may be used in NORIA are

- PEFF specifies a constant value for effective liquid pressure at a node.
- PEFFSIDE specifies a constant value for effective liquid pressure along an element side.
- PEFFVARY specifies that effective liquid pressure along an element side is variable. A CURVE subroutine must be supplied by the user to determine how effective liquid pressure depends on other dependent and independent variables.
- PV specifies a constant value for partial vapor pressure at a node.
- PVSIDE specifies a constant value for partial vapor pressure along an element side.
- PVVARY specifies that partial vapor pressure along an element side is variable. A CURVE subroutine must be supplied by the user to determine how partial vapor pressure depends on other dependent and independent variables.
- PA specifies a constant value for partial air pressure at a node.
- PASIDE specifies a constant value for partial air pressure along an element side.

- PAVARY specifies that partial air pressure along an element side is variable. A CURVE subroutine must be supplied by the user to determine how partial air pressure depends on other dependent and independent variables.
- T specifies a constant value for temperature at a node.
- TSIDE specifies a constant value for temperature along an element side.
- TVARY specifies that temperature along an element side is variable. A CURVE subroutine must be supplied by the user to determine how temperature depends on other dependent and independent variables.
- ULSIDE specifies a constant value of liquid velocity in the direction of the outward-pointing normal along an element side.
- ULVARY specifies that liquid velocity in the direction of the outward-pointing normal is constant along an element side but depends on other dependent and independent variables there. A CURVE subroutine must be supplied by the user to implement this boundary condition.
- UVSIDE specifies a constant value of vapor velocity in the direction of the outward-pointing normal along an element side.
- UVVARY specifies that vapor velocity in the direction of the outward-pointing normal is constant along an element side but depends on other dependent and independent variables there. A CURVE subroutine must be supplied by the user to implement this boundary condition.
- UASIDE specifies a constant value of air velocity in the direction of the outward-pointing normal along an element side.
- UAVARY specifies that air velocity in the direction of the outward-pointing normal is constant along an element side but depends on other dependent and independent variables there. A CURVE subroutine must be supplied by the user to implement this boundary condition.
- QSIDE specifies a constant value of conductive heat flux in the direction of the outward pointing normal along an element side.
- QVARY specifies that conductive heat flux in the direction of the outward-pointing normal is constant along an element side but depends on other dependent and independent variables there. A CURVE subroutine must be supplied by the user to implement this boundary condition.
- Impermeable surface requires no boundary condition to be specified.
- Adiabatic surface requires no boundary condition to be specified.

Each of the above boundary conditions can be used with any of the above element types. NORIA will permit a total of six CURVE subroutines to be used. The format for these subroutines is described in Subsection 6.11.

Convective and radiative boundary conditions are special cases of the general non-constant heat-flux boundary condition, QVARY, which can be easily incorporated in NORIA analyses.

In order to facilitate the specification of elements and boundary conditions, a looping feature is available in NORIA. This feature allows the definition of ILOOPs and JLOOPs (which are similar to FORTRAN DO-loops) for incrementing data in the I and J directions. Nesting of the loops may be in either order, but no more than one ILOOP or JLOOP may be in effect at once. All I or J values are given the same increment within the loop. The looping commands have the following form:

ILOOP, *npass*, *inc.*

JLOOP, *npass*, *inc.*

where

npass specifies the number of passes to be made through the loop.

inc. specifies the increment to be added to the I or J values found within the loop. The *inc.* parameter may be negative.

Element data cards, boundary condition data cards, or a combination of the two, follow the the looping commands. I-looping and J-looping are terminated respectively by the following commands.

IEND

JEND

Values of I and J on the first time through the loop are those given on the data cards that lie within the loop. These looping commands may appear at any point within the element and boundary condition portion of the SETUP command.

6.4 FORMKF Command Card

After the completion of the SETUP command, the next task in the finite element formulation of a problem is to perform the element integrations necessary to solve the finite element Equations (15)–(18). This task is triggered by the FORMKF command:

FORMKF, [*symmetry*]

where

symmetry is an alphanumeric name that indicates the type of coordinate system desired. A two-dimensional Cartesian coordinate system with coordinates x and z is used in performing the element integrals if *symmetry* is omitted; a two-dimensional polar coordinate system with coordinates r and z is used in performing the element integrals if *symmetry* is set to AXISYM. In other words, planar symmetry is the default option, and axisymmetry is specified by setting *symmetry* =AXISYM.

6.5 OUTPUT Command Card

It is sometimes convenient to limit the amount of output generated during the solution of a problem by NORIA. It may also be convenient to output values of the dependent variables at locations other than nodal points. Both of these features are available in NORIA by using the OUTPUT command card. Selective printing of dependent variables is enforced by an OUTPUT command card with the following format:

OUTPUT, *type*, n_1, n_2,..., n_{50}

where

type is an alphanumeric name that indicates how the subsequent data are to be interpreted. If *type* =SINGLE, the data that follow are taken to be element numbers. If *type* =STRING, the data that follow are interpreted as pairs of element numbers that define the lower and upper limits on a string of elements.

n_1, n_2,..., n_{50} is a list of element numbers that indicates which elements are to be included in the output list. A maximum of 50 individual element numbers or 25 element pairs may be specified on a single card.

There is no limit to the number of OUTPUT command cards that may be used in a NORIA input deck. OUTPUT cards of both *types* may be mixed together in any order. However, all OUTPUT command cards must precede the UNZIPP command card described in the following subsection. If no OUTPUT cards are included in the input deck, dependent variables are printed at every element in the finite element mesh.

Output at special points, *i.e.*, points that do not coincide with nodal points, may be requested by an OUTPUT card of the following format:

OUTPUT, POINTS, x_1, y_1, x_2, y_2,..., x_{25}, y_{25}

where

(x_1, y_1)–(x_{25}, y_{25}) is a list of (x, z) [or (r, z)] coordinates for the special points. A maximum of 25 points may be specified on a single command card.

NORIA allows up to 50 special points to be specified during an analysis. Results at special points are printed in the output and are stored for possible later use in history plots. OUTPUT command cards that designate special points may be mixed with cards that limit output.

6.6 UNZIPP Command Card

The UNZIPP command card instructs NORIA to begin assembling and integrating the finite element Equations (15)–(18). The UNZIPP command card has the following format:

UNZIPP, t_i, [t_f], Δt_i, [*no. steps*], [*init. cond.*], [*nprint*], [$tprint_1$],..., [$tprint_{45}$]

where

t_i is initial time. If t_i is set to TAPE, the initial time is read from a disk file called TAPE19.

$t_f \prec 10^{100} \succ$ is final time, *i.e.*, the time at which the time integration procedure will stop.

Δt_i is the initial time-step size. Δt_i also establishes the minimum time step allowed during the time integration procedure.

no. steps $\prec 1000 \succ$ is the maximum number of time steps to be taken.

init. cond. specifies the source of the initial conditions for the problem. If this parameter is omitted, initial conditions are taken from material property cards. If this parameter is set to TAPE, initial conditions are read from TAPE19, as explained below.

nprint is the number of time planes at which results are to be printed. Normally, results are printed at the final time plane so the user need not request output then. If *nprint* is omitted, results will be printed only at the last time plane of a calculation.

$tprint_1$–$tprint_{45}$ specify the time planes at which output is desired. A maximum of 45 time planes may be designated for output. The value of *nprint* should not be greater than the number of *tprints* supplied by the user. Specifying one or more *tprints* will alter the selection of time-step size so that results can be computed at the exact values of time requested by the user.

There is no need for multiple UNZIPP cards in NORIA because time-step size is adjusted automatically by the program and because specification of time planes at which results are output is quite flexible. The UNZIPP card must be followed by an END card:

END

Initial conditions may be specified on material property cards or read from TAPE19, as described above. TAPE19 can be created in one of two ways: it can be saved from a previous NORIA run using the RESTART option described below or it can be written by another program. The appropriate format for TAPE19 is described in Subsection 6.14.

6.7 HEATFLUX Command Card

NORIA includes the option to compute conductive heat fluxes on an element basis by using the following command card:

HEATFLUX, *time, location*

where

time is the time at which the heat-flux calculations are desired. Normally, the calculation is performed at the first time plane that follows the specified value of *time*. However, the user can cause the heat-flux calculation to be performed at the exact value of *time* specified by requesting that output be printed at *tprint* = *time*. If *time* is set to ALLTIMES, heat fluxes are computed at each element for all time planes.

location specifies where the heat-flux calculations are to be performed. For *location* = FULL, heat-flux calculations are made at each element in the mesh. A second option is to replace *location* by a list of up to 20 elements at which heat-flux calculations are to be performed. This parameter is omitted if *time* is set to ALLTIMES.

The numbering of local node points is shown in Figure 5. However, heat fluxes are not calculated at node points but at points halfway between node points in the mapped domain, *i.e.*, at the points $(\xi, \eta) = (\pm 1/2, \pm 1)$ or $(\pm 1, \pm 1/2)$. These points are numbered sequentially in the counterclockwise direction from the identifying node of the element, as shown in Figure 6. Output produced by the HEATFLUX command has three parts:

- Heat-flux components in the x- and z-directions (or r- and z-directions) are printed at each of the two points halfway between adjacent nodes on each side of an element.
- Heat fluxes normal to the element boundary are printed at these same points. Normal heat fluxes are calculated by taking the inner product of each heat-flux vector defined above with the outward-pointing normal to the element boundary there.
- Integrals of heat flux along each side of an element are printed. These quantities have the dimensions of power per unit depth of the domain. Heat-flux integrals are especially useful because they allow the user to estimate the rates at which energy is conducted across any of surface within or on the boundaries of a problem domain.

However, no output is generated when *time* is set to ALLTIMES. This option is used only to create an internal data file so that conductive heat-flux histories or profiles can be plotted (see HISTORY or PROFILE option in Subsection 6.9). There is no limit to the number of HEATFLUX commands that may appear in a NORIA input deck.

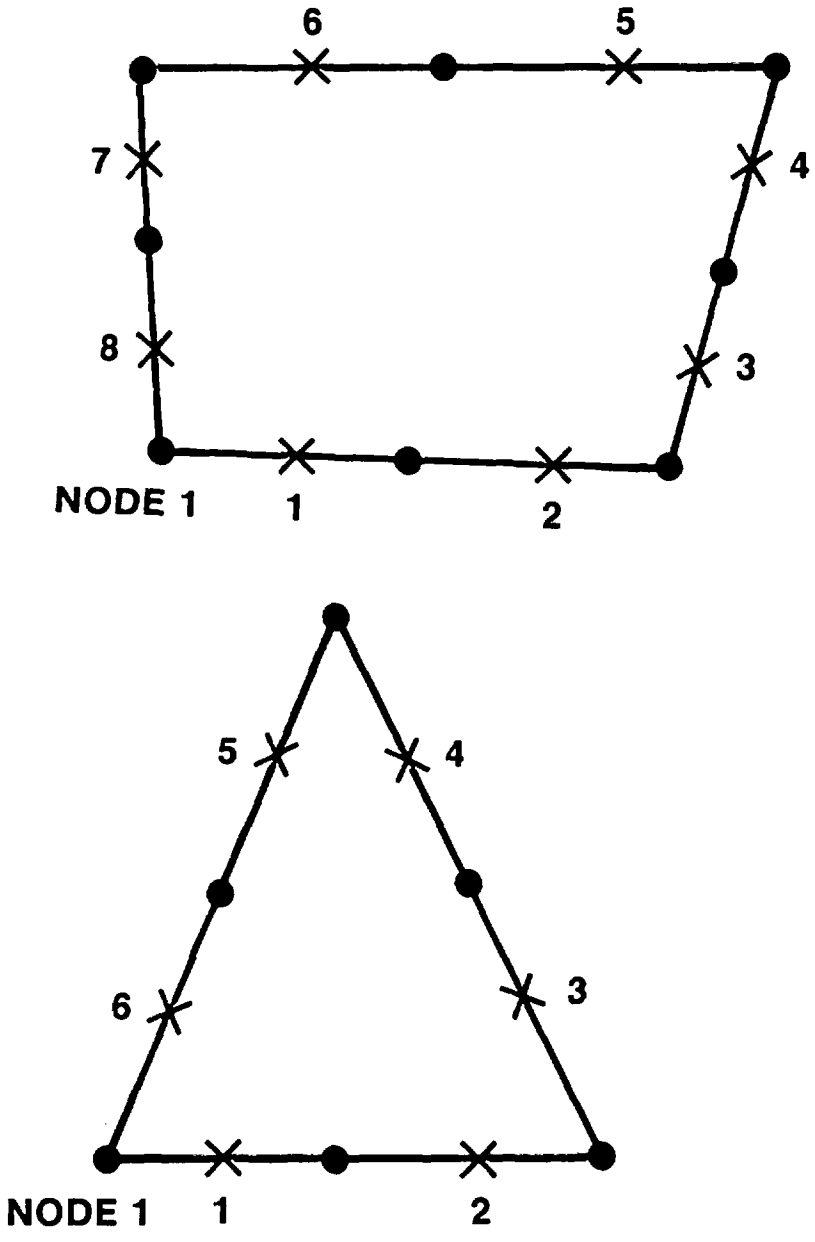


Figure 6: Numbering for Heat-Flux Calculations

6.8 VELOCITY Command Card

NORIA includes the option of computing velocities of liquid water, water vapor, and air on an element basis using the following command card:

VELOCITY, *time*, *location*

where

time is the time at which the velocity calculations are desired. Normally, the calculation is performed at the first time plane that follows the specified value of *time*. However, the user can cause the velocity calculations to be performed at the exact value of *time* specified by requesting that output be printed at *tprint* = *time*. If *time* is set to ALLTIMES, velocities are computed at each element for all time planes.

location specifies where the velocity calculations are to be performed. For *location* = FULL, velocity calculations are made for each element in the mesh. A second option is to replace *location* by a list of up to 20 elements at which velocity calculations are to be performed. This parameter is omitted if *time* is set to ALLTIMES.

Velocities are calculated at four Gauss points within an element and then extrapolated to the corner nodes using bilinear basis functions. Both velocity components (x and z or r and z) are printed for liquid water, water vapor, and air. However, no output is generated when *time* is set to ALLTIMES. This option is used only to create an internal data file so that velocity histories or profiles can be plotted (see HISTORY or PROFILE option in the following subsection). There is no limit to the number of VELOCITY commands that may appear in a NORIA input deck.

6.9 PLOT Command Card

NORIA contains a plotting package to facilitate setting up finite element meshes and interpreting data. Six types of plots are available in NORIA, and all can be obtained by using the following command card:

PLOT, *type*, x_{min}, y_{min}, x_{max}, y_{max}, [i_{min}, j_{min}, i_{max}, j_{max}],

[*xscale*], [*yscale*], [*number*], [*special pts.*]

where

type is an alphanumeric name that specifies the type of plot desired. The permissible parameter values are listed below.

(x_{min}, y_{min}) specifies the coordinate at the lower left corner of a rectangular window that defines a frame for the plot. For nodal-point, element, outline, and contour plots (defined below), the rectangular window determines which elements or node points are included in the plot. Elements that fall partly outside of the window are excluded from such plots. For history and profile plots, the window determines the abscissa and ordinate limits. If these parameters are omitted, NORIA automatically scales the axes to permit all of the data to be plotted.

(x_{max}, y_{max}) specify the coordinate at the upper right corner of a rectangular window that defines a frame for the plot.

(i_{min}, j_{min}) is an optional specification of the lower left corner of a rectangular window to be plotted. If the (I,J) limits are specified, the (x, y) limits set the border of the plot. The (I,J) limits are not used in history and profile plots.

(i_{max}, j_{max}) is the corresponding optional specification of the upper right corner of a rectangular window to be plotted.

xscale $\prec 1 \succ$ specifies the magnification for the x-coordinate of the plot. Plots are always the largest possible size consistent with the plotting device.

yscale $\prec 1 \succ$ specifies the magnification for the y-coordinate of the plot.

number specifies whether element numbers are to be displayed on a plot. This option is used only on element plots. If *number* = NUMBER, element numbers are plotted at element centroids; if *number* is omitted, element numbers are not plotted.

special pts. specifies whether the location of special output points (see OUTPUT command) are to be displayed on an element plot. Special points are plotted only if *special pts.* =SPECIAL.

The *type* parameter may be set to any of the following values:

- POINTS generates a plot of nodal points from a file created during the SETUP operation. In order to execute this option, *grid plot* must be set to PLOT on the SETUP command card.
- ELEMENT generates a plot of the elements.

- OUTLINE generates an outline plot of each material boundary.
- CONTOUR generates a contour plot of water pressure, vapor pressure, air pressure, moisture content, or temperature.
- HISTORY generates a history plot of any of the dependent variables, moisture content, heat flux, or velocity at specified locations.
- PROFILE generates a plot of any of the dependent variables, moisture content, heat flux, or velocity as a function of position.

A series of data cards is required following CONTOUR, HISTORY, and PROFILE plot commands.

6.9.1 Contour Data Cards

One or more data cards of the following form must be supplied after each PLOT, CONTOUR card:

$$\underline{contour,\ time,\ number,\ c_1,\ c_2,\ldots,\ c_{20}}$$

where

contour is an alphanumeric name that indicates the variable to be contoured. This parameter may be set to PLIQ, PVAP, PAIR, ISOTHERMS, or MOISTURE to create contour plots of effective liquid pressure, partial vapor pressure, partial air pressure, absolute temperature, and moisture content, respectively.

time is the time at which the contour plot is desired. Normally, the plot consists of results at the first time plane that follows the specified value of *time*. However, the user can cause the plotting to be performed at the exact value of *time* specified by requesting that output be printed at $tprint = time$.

number specifies the number of contours to be plotted. A maximum of 20 contour lines is allowed on each plot.

$c_1, c_2,\ldots, c_{20}$ are optional parameters that specify values of the contours to be plotted. If these parameters are omitted, the contours are evenly distributed over the interval between the maximum and minimum values of the variable being contoured.

If c_1–c_{20} are omitted, it is possible to generate a blank plot when contouring a window of the problem domain. A sequence of contour data cards of any type may follow a single PLOT, CONTOUR card. An END card must be used at the end of the sequence of contour data cards.

END

A looping feature is available to simplify specification of data for a series of contour plots. The looping command has the following form:

PLOTLOOP, *no. plots*, *time inc.*

where

no. plots specifies the number of plots to be generated.

time inc. specifies the time interval between plots. Contour plots are drawn at the first time planes that follow the specified values of time.

The PLOTLOOP command is followed by one contour data card and is terminated by the following command.

PLOTEND

There is no limit to the number of PLOTLOOP data sets that may follow a PLOT, CONTOUR command card. The sequence of contour plots is again terminated by an END card. PLOTLOOPs and individual contour data cards may be mixed together in any sequence under a PLOT, CONTOUR command card.

6.9.2 History Data Cards

One or more of the following data cards must follow a PLOT, HISTORY command card.

location, *no. points*, $time_1$, $time_2$, $element_1$, $node_1$,..., $element_{10}$, $node_{10}$

where

location is an alphanumeric name that indicates the variable to be plotted as a function of time. Possible choices for *location* are: (1) PELOCATION for plotting effective liquid pressure; (2) PGLOCATION for plotting total gas pressure; (3) PVLOCATION for plotting partial vapor pressure; (4) PALOCATION for plotting partial air pressure; (5) TLOCATION for plotting absolute temperature; (6) MLOCATION for plotting moisture content; (7) ULLOCATION for plotting horizontal (or radial) velocity in the liquid phase; (8) VLLOCATION for plotting vertical velocity in the liquid phase; (9) UVLOCATION for plotting horizontal (or radial) velocity of vapor; (10) VVLOCATION for plotting vertical velocity of vapor; (11) UALOCATION for plotting horizontal (or radial) velocity of air; (12) VALOCATION for plotting vertical velocity of air; and (13) QLOCATION for plotting the component of heat flux normal to an element side. Each of these quantities except heat flux may also be plotted at special points by replacing LOCATION by SPECIAL. (For example, TLOCATION would be replaced by TSPECIAL.)

no. points specifies the number of histories to be plotted. A maximum of 10 time histories per plot is allowed.

$time_1$, $time_2$ indicate the times at which a history plot is to begin and end, respectively. Data on a single history plot may represent a maximum of 400 time planes.

$element_i$, $node_i$ are element and local node numbers of the nodes at which histories are to be plotted. A maximum of 10 points may be included on a data card. If history curves are to be plotted at special points, an element and node number pair are replaced by a special point number. A maximum of 10 special point numbers is allowed on a data card. The local node number for pressures, temperatures, and heat flux may range from one to eight; the local node number for velocities may range from one to four.

There is no restriction on the number or types of *location* cards that may follow a PLOT, HISTORY command card. The sequence of *location* cards is terminated by an end card.

END

A HEATFLUX, ALLTIMES command must be executed before creating heat-flux history plots, and a VELOCITY, ALLTIMES command must be executed before creating velocity history plots.

6.9.3 Profile Data Cards

The data cards required to execute the PLOT, PROFILE option have the following form:

TIMEPLANE, *no. planes*, $time_1$, $time_2$,..., $time_{10}$

location, *type*, *no. pairs*, $element_1$, $side/node_1$,..., $element_{20}$, $side/node_{20}$

where

no. planes specifies the number of profiles (at different times) to be plotted. A maximum of 10 profiles per plot is allowed.

$time_1$,..., $time_{10}$ specifies the times at which a profile is to be plotted. The parameter *no. planes* should agree with the number of $time_i$s supplied.

location is an alphanumeric name that indicates the variable to be plotted as a function of position. The possible values of *location* are the same as those described under History Data Cards.

type is an alphanumeric name that indicates the way the curve along which a profile is plotted is to be specified. If *type* =SIDES, the curve is specified along element sides; if *type* =NODES, the curve is specified at individual nodes.

no. pairs specifies the number of ($element_i$, $side/node_i$) pairs needed to describe the curve along which a profile is to be plotted.

$element_i$ specifies an element that forms part of the curve along which a profile is to be plotted.

$side/node_i$ specifies a side or node that is part of the curve along which a profile is to be plotted.

There is no restriction on the number of TIMEPLANE/*location* data sets that may follow a PLOT, PROFILE command. However, TIMEPLANE and *location* data cards must occur in pairs, the TIMEPLANE card first. The data sequence for a PLOT, PROFILE command card is terminated by an END card.

END

When *type* is set to SIDES, spatial distance is measured along element sides; when *type* is set to NODES, spatial distance is measured along straight lines between consecutive node points. The input sequence determines the profile path in both options. The path is not required to be continuous: some elements along a curve may be omitted. Multiple *side/node* specifications for an individual element are permitted. When the SIDES option is used, the profile path is directed along element sides in the direction of increasing node number (disregarding midside nodes). If this path is inappropriate, the NODES option should be used.

6.10 RESTART Command Card

NORIA allows transient solution data and associated mesh data to be saved for further computations or postprocessing through the use of the RESTART command. The following command is used to save solution data.

RESTART, SAVE

In order to restart from previously saved solution data, the following command card is used:

RESTART, RESET, [*nsteps*]

where

nsteps $\prec 0 \succ$ indicates the time-plane number at which the solution file is to be positioned during the restart process. The solution file is rewound if *nsteps* is omitted. The solution file is positioned after the *nsteps* time plane; the initial condition is counted as the first time plane. Generally, if n time steps have already been taken in a transient analysis, the user should set $nsteps = n$ to continue the transient analysis.

The RESTART, SAVE command should occur after the UNZIPP command sequence and after all plotting has been completed. The RESTART, RESET command should immediately follow the header card. No PLOT command card should be inserted between the RESTART, RESET and the UNZIPP command cards.

A FORMKF command card must precede an UNZIPP command sequence when a transient solution is being continued via the restart option. In addition, *init. cond.* on

the UNZIPP command card should be set to TAPE. It is usually desirable to set t_i to TAPE as well so that initial time is automatically set to the last time computed during the previous run.

The RESTART commands direct NORIA to collect (or distribute) mesh and solution data from (or to) two files: TAPE13, which contains the mesh data, and TAPE19, which contains the solution data. These files must then be accessed (or saved) by the appropriate system control cards. Typically, these files would be saved as tape or disc files.

6.11 Program Termination Command Card

There are two ways to terminate a NORIA run. If two or more problems are to be run in sequence, the appropriate termination command is an END command.

END

The appropriate termination command for a single-problem run or the final problem in a multiple-problem run is a STOP command.

STOP

Another option that is available to NORIA users is to postprocess NORIA solution data with TRINITY (Gartling, 1985). TRINITY can produce most of the plots described in Subsection 6.9 and can also produce color-fringe plots. Coupling NORIA with TRINITY is accomplished by the following termination command.

STOP, POST

The POST option causes mesh and solution data to be gathered on TAPE12. TAPE12 must then be routed back to a VAX for postprocessing by TRINITY (Gartling, 1985).

6.12 Input Deck Sequence

The order of commands in a NORIA input deck is somewhat flexible. There are some obvious limitations to command sequence, however, because some computations are necessary prerequisites to others. The following comments provide some guidelines.

- The POINTS, ELEMENT, and OUTLINE plot options must follow the SETUP command sequence because the information used to generate these plots is created during the SETUP procedure.

- If SETUP, FORMKF, and UNZIPP command cards are included in an input deck, they should always occur in the order listed here. However, in many cases, it is desirable to insert other command cards between these cards.

- Plot commands should follow the UNZIPP command when a RESTART, RESET command is used.

- Other plot options may be included anywhere after the UNZIPP command, assuming that an UNZIPP command card occurs in the input deck.

- The RESTART, SAVE command must follow an UNZIPP command and should follow all PLOT commands. The execution of RESTART, SAVE before plotting could result in a conflict in file usage.

- If a transient calculation is being continued, the RESTART, RESET command should immediately follow the header card and should immediately precede both FORMKF and UNZIPP commands. However, the SETUP command sequence may be inserted between the RESTART and the FORMKF commands. If the SETUP command sequence is included, it be should identical to the one used in the previous transient analysis.

- If OUTPUT command cards are included in the input deck, they should always follow the SETUP command card sequence and precede the UNZIPP command card.

Aside from these restrictions, input deck structure is quite flexible.

6.13 User Subroutines

All problems run using NORIA require subroutines supplied by the user. These subroutines are required whenever any material properties depend on time, spatial coordinates, or the dependent variables and when boundary conditions are nonconstant. NORIA recognizes 28 user-defined subroutines. Only a subset of these is normally needed in a computation. Each of these subroutines is described below. Default subroutines all use MKS units and degrees Kelvin; however, NORIA can use any consistent set of units that the user desires.

6.13.1 Viscosities

Subroutine VISCL can be used to prescribe the functional dependence of liquid viscosity. Liquid viscosity is determined by calls to VISCL if μ_l is set to VARIABLE on the material data card for water. The default version of VISCL is

```
      SUBROUTINE VISCL(AMUL,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION AMUL(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES VISCOSITY IN
C     UNITS OF KG/M/S FROM TEMPERATURE IN DEGREES
C     KELVIN.
C
      DO 10 I=1,NN
      AMUL(I)=2.414E-3*10.0**(247.8/(T(I)-140.00))
   10 CONTINUE
      RETURN
      END
```

which corresponds to the following constitutive relationship (Gray, 1972):

$$\mathrm{AMUL} = 0.002414 \cdot 10^{247.8/(\mathrm{T}-140.00)}$$

where

AMUL is an output vector that contains the viscosity of the liquid phase at each node point of the current element.

T is an input vector that contains the absolute temperature at each node point of the current element.

TH is an input vector that contains the moisture content at each node point of the current element.

P is an input vector that contains the effective pressure in the liquid phase at each node point of the current element.

PV is an input vector that contains the partial pressure of vapor at each node point of the current element.

PA is an input vector that contains the partial pressure of air at each node point of the current element.

X is an input vector that contains the horizontal or radial coordinate of each node point of the current element.

Y is an input vector that contains the vertical coordinate of each node point of the current element.

TIME is an input variable that contains the current value of time.

PHI is an input variable that contains the porosity of the solid matrix for the current element.

RHOG is an input variable that contains the product of liquid density and gravitational acceleration, $\rho_l \cdot g$.

NELEM is an integer-input variable that contains the current element number.

MAT is an integer-input variable that contains the current material number, *number*.

NN is an integer-input variable that contains the number of nodes in the current element.

VISCL may be modified to account for a different functional dependence on temperature or on any of the other variables included as input parameters to the subroutine.

The subroutine that prescribes vapor viscosity is VISCV. Vapor viscosity is determined by calls to VISCV if μ_v is set to VARIABLE on the material data card for vapor. The default version of VISCV is

```
      SUBROUTINE VISCV(AMUV,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION AMUV(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES VISCOSITY IN
C     UNITS OF KG/M/S
C
      DO 10 I=1,NN
      AMUV(I)=8.04E-6+4.2E-8*(T(I)-273.15)
   10 CONTINUE
      RETURN
      END
```

which corresponds to the following constitutive relationship (Gray, 1972):

$$\text{AMUV} = 8.04 \cdot 10^{-6} + 4.2 \cdot 10^{-8}(\text{T} - 273.15)$$

where

AMUV is an output vector that contains the viscosity of vapor at each node point of the current element.

Other variables are the same as those in VISCL. VISCV may be modified to account for a different functional dependence on temperature or on any of the other variables included as input parameters to the subroutine.

The subroutine that prescribes air viscosity is VISCA. Air viscosity is determined by calls to VISCA if μ_a is set to VARIABLE on the material data card for air. The default version of VISCA is

```
      SUBROUTINE VISCA(AMUA,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION AMUA(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES VISCOSITY IN
C     UNITS OF KG/M/S
C
      DO 10 I=1,NN
      AMUA(I)=1.71E-5+4.386E-8*(T(I)-273.15)
     2       -1.056E-11*(T(I)-273.15)**2
   10 CONTINUE
      RETURN
      END
```

which corresponds to the following constitutive relationship (Watson, 1972):

$$\text{AMUA} = 1.71 \cdot 10^{-5} + 4.386 \cdot 10^{-8}(\text{T} - 273.15) - 1.056 \cdot 10^{-11}(\text{T} - 273.15)^2$$

where

AMUA is an output vector that contains the viscosity of air at each node point of the current element.

Other variables are the same as those in VISCL. VISCA may be modified to account for a different functional dependence on temperature or on any of the other variables included as input parameters to the subroutine.

6.13.2 Heat Capacities

Subroutine HCAPL can be used to prescribe the functional dependence of liquid heat capacity. Liquid heat capacity is determined by calls to HCAPL if c_l is set to VARIABLE on the material data card for water. The default version of HCAPL is

```
      SUBROUTINE HCAPL(HCL,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION HCL(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES HEAT CAPACITY IN
C     UNITS OF J/KG
C
      DO 10 I=1,NN
      HCL(I)=6122.-11.76*T(I)+0.01772*T(I)**2
   10 CONTINUE
      RETURN
      END
```

which corresponds to the following constitutive relationship (Vargaftik, 1975):

$$\mathrm{HCL} = 6122 - 11.76 \cdot \mathrm{T} + 0.01772 \cdot \mathrm{T}^2$$

where

HCL is an output vector that contains the heat capacity of the liquid phase at each node point of the current element.

Other variables are the same as those in the viscosity subroutines above. HCAPL may be modified to account for a different functional dependence on temperature or on any of the other variables included as input parameters to the subroutine.

The subroutine that prescribes the heat capacity of vapor is HCAPV. The heat capacity of vapor is determined by calls to HCAPV if c_v is set to VARIABLE on the material data card for vapor. The default version of HCAPV is

```
      SUBROUTINE HCAPV(HCV,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION HCV(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES HEAT CAPACITY IN
C     UNITS OF J/KG
C
      DO 10 I=1,NN
      HCV(I)=1860.0
   10 CONTINUE
      RETURN
      END
```

where

HCV is an output vector that contains the heat capacity of vapor at each node point of the current element.

Other variables are the same as those in the viscosity subroutines. HCAPV may be modified to account for a functional dependence on temperature or on any of the other variables included as input parameters to the subroutine.

The subroutine that prescribes air heat capacity is HCAPA. The heat capacity of air is determined by calls to HCAPA if c_a is set to VARIABLE on the material data card for air. The default version of HCAPA is

```
      SUBROUTINE HCAPA(HCA,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION HCA(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES HEAT CAPACITY IN
C     UNITS OF J/KG
C
      DO 10 I=1,NN
      HCA(I)=1000.0
   10 CONTINUE
      RETURN
      END
```

where

HCA is an output vector that contains the heat capacity of air at each node point of the current element.

Other variables are the same as those in the viscosity subroutines. HCAPA may be modified to account for a functional dependence on temperature or on any of the other variables included as input parameters to the subroutine.

The subroutine that prescribes heat capacity of a solid grain is HCAPS. The heat capacity of a solid grain is determined by calls to HCAPS if c_s is set to VARIABLE on the material data card for a solid matrix. The default version of HCAPS is

```
      SUBROUTINE HCAPS(HCS,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION HCS(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES HEAT CAPACITY IN
C     UNITS OF J/KG
C
      DO 10 I=1,NN
      HCS(I)=840.0
   10 CONTINUE
      RETURN
      END
```

where

HCS is an output vector that contains the heat capacity of a solid grain at each node point of the current element.

Other variables are the same as those in the viscosity subroutines. HCAPS may be modified to account for a functional dependence on temperature or on any of the other variables included as input parameters to the subroutine. The value of heat capacity of the solid material used in the default subroutine corresponds to a specific type of welded tuff. Other solid materials will likely have different properties.

6.13.3 Thermal Conductivities

Subroutine FLAMBL can be used to prescribe the functional dependence of the thermal conductivity of liquid. The thermal conductivity of liquid is determined by calls to SLAMBL if λ_l is set to VARIABLE on the material data card for water. The default version of FLAMBL is

```
      SUBROUTINE FLAMBL(FLAML,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION FLAML(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES THERMAL
C     CONDUCTIVITY IN UNITS OF W/M PER
C     DEGREE KELVIN
C
      DO 10 I=1,NN
      FLAML(I)=5.45
   10 CONTINUE
      RETURN
      END
```

where

FLAML is an output vector that contains the thermal conductivity of the liquid phase at each node point of the current element.

Other variables are the same as those in the subroutines above. FLAMBL may be modified to account for a functional dependence on temperature or on any of the other variables included as input parameters to the subroutine. The appropriate value for the thermal conductivity of water depends to some extent on the particular application. The value of 5.45 W/mK used above is based on an experimental measurement in which the rock matrix was welded tuff (Nimick, 1984).

The subroutine that defines the thermal conductivity of vapor is FLAMBV. The thermal conductivity of vapor is determined by calls to FLAMBV if λ_v is set to VARIABLE on the material data card for vapor. The default version of FLAMBV is

```
      SUBROUTINE FLAMBV(FLAMV,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION FLAMV(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
```

```
C
C     THIS SUBROUTINE CALCULATES THERMAL
C     CONDUCTIVITY IN UNITS OF W/M PER
C     DEGREE KELVIN
C
      DO 10 I=1,NN
      FLAMV(I)=0.0
   10 CONTINUE
      RETURN
      END
```

where

FLAMV is an output vector that contains the thermal conductivity of vapor at each node point of the current element.

Other variables are the same as those in the subroutines above. FLAMBV may be modified to account for a functional dependence on temperature or on any of the other variables included as input parameters to the subroutine.

The subroutine that prescribes the thermal conductivity of air is FLAMBA. The thermal conductivity of air is determined by calls to FLAMBA if λ_a is set to VARIABLE on the material data card for air. The default version of FLAMBA is

```
      SUBROUTINE FLAMBA(FLAMV,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION FLAMA(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES THERMAL
C     CONDUCTIVITY IN UNITS OF W/M PER
C     DEGREE KELVIN
C
      DO 10 I=1,NN
      FLAMA(I)=0.0
   10 CONTINUE
      RETURN
      END
```

where

FLAMA is an output vector that contains the thermal conductivity of air at each node point of the current element.

Other variables are the same as those in the subroutines above. FLAMBA may be modified to account for a functional dependence on temperature or on any of the other variables included as input parameters to the subroutine.

The subroutine that prescribes thermal conductivity of a solid grain is SLAMBDA. The thermal conductivity of a solid grain is determined by calls to SLAMBDA if λ_s is set to VARIABLE on the material data card for a solid matrix. The default version of SLAMBDA is

```
      SUBROUTINE SLAMBDA(SLAM11,SLAM22,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION SLAM11(8),SLAM22(8),T(8),TH(8),P(8),PV(8),PA(8),
     1  X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES THERMAL
C     CONDUCTIVITY IN UNITS OF W/M PER
C     DEGREE KELVIN
C
      DO 10 I=1,NN
      SLAM11(I)=1.74
      SLAM22(I)=1.74
   10 CONTINUE
      RETURN
      END
```

where

SLAM11 is an output vector that contains the value of thermal conductivity of a solid grain in the direction of the first principal axis at each node point of the current element.

SLAM22 is an output vector that contains the value of thermal conductivity of a solid grain in the direction of the second principal axis at each node point of the current element.

Other variables are the same as those in the subroutines above. SLAMBDA may be modified to account for a functional dependence on temperature or on any of the other variables included as input parameters to the subroutine. The value of 1.74 W/mK used above is for a particular sample of welded tuff.

6.13.4 Liquid Coefficient of Volumetric Expansion

The subroutine ISOVOL can be used to prescribe the functional dependence of the coefficient of volumetric expansion for the liquid. ISOVOL is called if β_l is set to VARIABLE on the material data card for water. The default version of ISOVOL is

```
      SUBROUTINE ISOVOL(BETA,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION BETA(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES VOLUMETRIC
C     EXPANSION COEFFICIENT IN UNITS OF PER
C     DEGREE KELVIN
C
      DO 10 I=1,NN
      BETA(I)=0.0
   10 CONTINUE
      RETURN
      END
```

where

BETA is an output vector that contains the coefficient of volumetric expansion for the liquid phase at each node point of the current element.

Other variables are the same as those in the subroutines above. ISOVOL may be modified to account for a functional dependence on temperature or on any of the other variables included as input parameters to the subroutine.

6.13.5 Moisture Content

Subroutine FLUIDC must be used to prescribe the functional dependence of moisture content. FLUIDC is always called regardless of the parameters specified on any of the material property cards. The default version of FLUIDC is

```
      SUBROUTINE FLUIDC(TH,T,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION TH(8),T(8),P(8),PV(8),PA(8),X(8),Y(8)
      DATA RSAT,ALPHA,BETA /0.0801,0.00567,1.798/
C
C     THIS SUBROUTINE CALCULATES MOISTURE
C     CONTENT FROM PRESSURES IN PASCALS
C

C     CALCULATE MOISTURE CONTENT
      ALAMBDA=1.0-1.0/BETA
      DO 10 I=1,NN
      PC=PV(I)+PA(I)-P(I)+RHOG*Y(I)
      PC=PC*((647.3-T(I))/354.3)**(-0.0809)
      HEAD=PC/RHOG
      AH=ALFA*HEAD
      TH(I)=PHI*((1.0-RSAT)*(1.0+AH**BETA)
     2  **(-ALAMBDA)+RSAT)
   10 CONTINUE
      RETURN
      END
```

which corresponds to the following constitutive relationship (Klavetter, 1984):

$$\mathrm{SEF} = (1 - \mathrm{RSAT})\left[1 + \left(\frac{\mathrm{ALPHA} \cdot \mathrm{PC}}{\mathrm{RHOG}}\right)^{\mathrm{BETA}}\right]^{-\mathrm{ALAMBDA}} + \mathrm{RSAT}$$

where

TH is an output vector that contains the moisture content for the liquid phase at each node point of the current element.

RSAT is the residual saturation, *i.e.*, the minimum saturation of the porous matrix.

ALPHA, BETA are parameters used to prescribe how moisture content depends on capillary pressure.

ALAMBDA is defined by ALAMBDA= 1 − 1/BETA.

PC is the capillary pressure and is defined by

$$PC = (-P + PV + PA + RHOG \cdot Y)\left(\frac{647.3 - T)}{354.3}\right)^{-0.0809}$$

SEF is the effective saturation of liquid water and is defined by SEF = [(TH/PHI) − RSAT]/[1−RSAT].

Other variables are the same as those in the subroutines above. FLUIDC may be modified to account for a different functional dependence on capillary pressure and temperature or on any of the variables included as input parameters to the subroutine.

6.13.6 Permeabilities

Subroutine PERM can be used to prescribe the functional dependence of the liquid- and gas-phase permeabilities. PERM is called if K_{l11} is set to VARIABLE on a material data card for a solid matrix. The default version of PERM is

```
      SUBROUTINE PERM(AKL11,AKL22,AKG11,AKG22,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION AKL11(8),AKL22(8),AKG11(8),AKG22(8),T(8),
     1  TH(8),P(8),PV(8),PA(8),X(8),Y(8)
      DATA ALFA,BETA,PERMST /0.00567,1.798,1.9E-18/
C
C     THIS SUBROUTINE CALCULATES PERMEABILITY
C     IN UNITS OF M**2
C

C     ROCK PERMEABILITY TO LIQUID AND GAS
      ALAMBDA=1.0-1.0/BETA
      DO 10 I=1,NN
      PC=PV(I)+PA(I)-P(I)+RHOG*Y(I)
      PC=PC*((647.3-T(I))/354.3)**0.0809
      HEAD=PC/RHOG
      AH=ALFA*HEAD
      AKL11(I)=PERMST*(1.0-AH**(BETA-1.0)
     2  *(1.0+AH**BETA)**(-ALAMBDA))**2
     3  /(1.0+AH**BETA)**(ALAMBDA/2.0)
      AKL22(I)=AKL11(I)
      AKG11(I)=PERMST-AKL11(I)
```

```
      AKG22(I)=AKG11(I)
   10 CONTINUE
      RETURN
      END
```

which corresponds to the following constitutive relationships (Peters *et al.*, 1984):

$$AKL11 = PERMST\frac{\left[1 - AH^{BETA-1}\left(1 + AH^{BETA}\right)^{-ALAMBDA}\right]^2}{\left(1 + AH^{BETA}\right)^{ALAMBDA/2}}$$

$$AH = ALFA \cdot PC/RHOG$$

$$AKL22 = AKL11$$

$$AKG11 = AKG22 = PERMST - AKL11$$

where

AKL11 is an output vector that contains the value of matrix permeability to the liquid in the direction of the first principal axis at each node point of the current element.

AKL22 is an output vector that contains the value of matrix permeability to the liquid in the direction of the second principal axis at each node point of the current element.

AKG11 is an output vector that contains the value of matrix permeability to the gas in the direction of the first principal axis at each node point of the current element.

AKG22 is an output vector that contains the value of matrix permeability to the gas in the direction of the second principal axis at each node point of the current element.

PERMST defines the absolute permeability of the solid matrix to flow of either phase.

Other variables are the same as those in the subroutines above. PERM may be modified to account for a different functional dependence on moisture content or on any of the variables included as input parameters to the subroutine.

6.13.7 Diffusion Coefficients

Subroutine KNUDIFV can be used to prescribe the functional dependence of the Knudsen diffusion coefficient for vapor. KNUDIFV is called if D_{Kv} is set to VARIABLE on the material data card for vapor. The default version of KNUDIFV is

```
      SUBROUTINE KNUDIFV(DKV,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION DKV(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
      DATA PORRAD,RV,PI,TORT /4.00E-8,462.0,3.142,5.0/
C
C     THIS SUBROUTINE CALCULATES THE KNUDSEN
C     DIFFUSION COEFFICIENT FOR WATER VAPOR
C     IN UNITS OF M**2/SEC GIVEN TEMPERATURE
C     IN DEGREES KELVIN
C
      DO 10 I=1,NN
      DKV(I)=(2.0/3.0)*PORRAD*SQRT(8.0*RV*T(I)/PI)/TORT
   10 CONTINUE
      RETURN
      END
```

which corresponds to the following constitutive relationship (Hadley, 1982):

$$\mathrm{DKV} = \frac{2 \cdot \mathrm{PORRAD}}{3 \cdot \mathrm{TORT}} \cdot \sqrt{\frac{8 \cdot \mathrm{RV} \cdot \mathrm{T}}{\pi}}$$

where

DKV is an output vector that contains the Knudsen diffusion coefficient for vapor at each node point of the current element.

PORRAD is a parameter that specifies the average pore radius of a matrix material.

RV is the ideal gas constant divided by the molecular mass of vapor.

TORT is the tortuosity of the solid matrix.

Other variables are the same as those in the subroutines above. KNUDIFV may be modified to account for a different functional dependence on any of the variables included as input parameters to the subroutine.

The subroutine that prescribes the functional dependence of the Knudsen diffusion coefficient for air is called KNUDIFA. KNUDIFA is called if D_{Ka} is set to VARIABLE on the material data card for air. The default version of KNUDIFA is

```
      SUBROUTINE KNUDIFA(DKA,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION DKA(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
      DATA PORRAD,RA,PI,TORT /4.00E-8,462.0,3.142,5.0/
C
C     THIS SUBROUTINE CALCULATES THE KNUDSEN
C     DIFFUSION COEFFICIENT FOR AIR
C     IN UNITS OF M**2/SEC GIVEN TEMPERATURE
C     IN DEGREES KELVIN
C
      DO 10 I=1,NN
      DKA(I)=(2.0/3.0)*PORRAD*SQRT(8.0*RA*T(I)/PI)/TORT
   10 CONTINUE
      RETURN
      END
```

which corresponds to the following constitutive relationship (Hadley, 1982):

$$\mathrm{DKA} = \frac{2 \cdot \mathrm{PORRAD}}{3 \cdot \mathrm{TORT}} \cdot \sqrt{\frac{8 \cdot \mathrm{RA} \cdot \mathrm{T}}{\pi}}$$

where

DKA is an output vector that contains the Knudsen diffusion coefficient for air at each node point of the current element.

RA is the ideal gas constant divided by the molecular mass of air.

Other variables are the same as those in KNUDIFV. KNUDIFA may be modified to account for a different functional dependence on any of the variables included as input parameters to the subroutine.

The subroutine BINDIF can be used to prescribe the functional dependence of the binary diffusion coefficient. BINDIF is called when D_B is set to VARIABLE on the material data card for vapor. The default version of BINDIF is

```
      SUBROUTINE BINDIF(DB,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION DB(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
      DATA TORT /5.0/
C
C     THIS SUBROUTINE CALCULATES THE BINARY
C     DIFFUSION COEFFICIENT FOR WATER VAPOR
C     AND AIR IN UNITS OF M**2/SEC GIVEN
C     TEMPERATURE IN DEGREES KELVIN AND
C     PRESSURES IN PASCALS
C
      DO 10 I=1,NN
      DB(I)=4.40E-6*T(I)**2.334/(PV(I)+PA(I))/TORT
   10 CONTINUE
      RETURN
      END
```

which corresponds to the following constitutive relationship (Bird *et al.*, 1960):

$$\mathrm{DB} = \frac{4.40 \cdot 10^{-6} \mathrm{T}^{2.334}}{(\mathrm{PV} + \mathrm{PA}) \cdot \mathrm{TORT}}$$

where

DB is an output vector that contains the binary diffusion coefficient at each node point of the current element.

Other variables are the same as those in KNUDIFV. BINDIF may be modified to account for a different functional dependence on any of the variables included as input parameters to the subroutine.

The subroutine THERDIF can be used to prescribe the functional dependence of the thermal diffusion coefficient. THERDIF is called if D_T is set to VARIABLE on the material data card for vapor. The default version of THERDIF is

```
      SUBROUTINE THERDIF(DT,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION DT(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES THE THERMO-
```

```
C     DIFFUSION COEFFICIENT FOR WATER VAPOR
C     AND AIR IN UNITS OF M**2/SEC
C
      DO 10 I=1,NN
      DT(I)=0.0
   10 CONTINUE
      RETURN
      END
```

where

DT is an output vector that contains the thermo-diffusion coefficient at each node point of the current element.

Other variables are the same as those in KNUDIFV. THERDIF may be modified to account for a functional dependence on any of the variables included as input parameters to the subroutine.

6.13.8 Latent Heat of Vaporization

Subroutine LATHEAT can be used to prescribe the functional dependence of the latent heat of vaporization. LATHEAT is called if L is set to VARIABLE on the material data card for water. The default version of LATHEAT is

```
      SUBROUTINE LATHEAT(AL,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION AL(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES LATENT
C     HEAT OF VAPORIZATION FOR WATER
C     IN UNITS OF J/KG GIVEN
C     TEMPERATURE IN DEGREES KELVIN
C
      DO 10 I=1,NN
      AL(I)=2.746E6-4.301E3*(T(I)-273.0)
   10 CONTINUE
      RETURN
      END
```

which corresponds to the following constitutive relationship (a linear fit to steam table values):

$$AL = 2.746 \cdot 10^6 - 4.301 \cdot 10^3 \cdot (T - 273)$$

where

AL is an output vector that contains the value of the latent heat of vaporization at each node point of the current element.

Other variables are the same as those in the subroutines above. LATHEAT may be modified to account for a different functional dependence on any of the variables included as input parameters to the subroutine.

6.13.9 Rate of Evaporation

The subroutine EVAP is always called to specify the local rate of vaporization. The default version of EVAP is

```
      SUBROUTINE EVAP(VAPR,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION VAPR(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
      DATA RSAT /0.0669/
C
C     THIS SUBROUTINE CALCULATES RATE OF
C     EVAPORATION IN UNITS OF KG/M**3/S
C     GIVEN TEMPERATURE IN DEGREES KELVIN
C     AND PRESSURES IN PASCALS
C
      DO 10 I=1,NN
      SEF=(TH(I)/PHI-RSAT)/(1.0-RSAT)
      VAPR(I)=SEF*(PVAP(T(I))-PV(I))*1.0E-5
   10 CONTINUE
      RETURN
      END
```

which corresponds to the following constitutive relationship:

$$\text{VAPR} = 10^{-5} \cdot \text{SEF} \cdot (\text{PVAP(T)} - \text{PV})$$

where

VAPR is an output vector that contains the rate of vaporization at each node point of the current element.

PVAP is the equilibrium vapor pressure at local conditions.

Other variables are the same as those in the subroutines above. EVAP may be modified to account for a different functional dependence on any of the variables included as input parameters to the subroutine. The equilibrium vapor pressure, PVAP, is determined by a functional subroutine. The default version of the functional subroutine PVAP is

```
      FUNCTION PVAP(T)
      DIMENSION PAR(36)
      DATA PAR/0.023,0.042,0.073,0.122,0.197,0.308,0.467,0.692,1.00,
     1         1.41,1.96,2.66,3.56,4.7,6.1,7.82,9.89,12.38,15.34,18.82,
     2         22.88,27.59,33.02,39.22,46.3,54.29,63.29,73.42,84.78,
     3         97.4,111.43,126.99,144.2,163.16,184.07,207.49/
C
C     THIS SUBROUTINE CALCULATES EQUILIBRIUM
C     VAPOR PRESSURE IN PASCALS GIVEN
C     TEMPERATURE IN DEGREES KELVIN
C
      TIN=T-293.
      IF (TIN.LT.0.0) TIN=0.0
      IF (TIN.GT.350.0) TIN=350.0
      CALL INTERPQ(PAR,36,10.0,TIN,PVAP,SLOPE)
      PVAP=PVAP*1.0E5
      RETURN
      END
```

PVAP determines equilibrium vapor pressure as a function of temperature from steam table values by using a quadratic interpolation routine, INTERPQ. PVAP can easily be augmented to account for vapor pressure lowering or other physical mechanisms that might be of interest.

6.13.10 Heat Source

The subroutine SOURCE can be used to prescribe the functional dependence of heat generation. SOURCE is called if S is set to VARIABLE on the material data card for a solid material. The default version of SOURCE is

```
      SUBROUTINE SOURCE(S,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION S(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES VOLUMETRIC
C     HEAT GENERATION IN UNITS OF W/M**3
C
      DO 10 I=1,NN
      S(I)=0.0
   10 CONTINUE
      RETURN
      END
```

where

S is an output vector that contains the rate of heat generation at each node point of the current element.

Other variables are the same as those in the subroutines above. SOURCE may be modified to account for a functional dependence on any of the variables included as input parameters to the subroutine.

6.13.11 Boundary Conditions

When boundary condition options PEFFVARY, PVVARY, PAVARY, TVARY, ULVARY, UVVARY, UAVARY, or QVARY are used, the user must supply one or more CURVEn subroutines to define how the boundary condition depends on the independent or dependent variables. NORIA will accept up to 6 CURVEn subroutines, (n can be 1 through 6). The value of n is prescribed by the *value/curve no.* parameter on the BC card. CURVEn subroutines have the following form:

```
      SUBROUTINE CURVEn(VALUE,T,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
```

```
      DIMENSION T(8),P(8),PV(8),PA(8),X(8),Y(8)
      DIMENSION ISIDE1(3),ISIDE2(3),ISIDE3(3),ISIDE4(3)
      DATA (ISIDE1(I),I=1,3),(ISIDE2(I),I=1,3),(ISIDE3(I),I=1,3),
     1     (ISIDE4(I),I=1,3) /1,5,2,2,6,3,3,7,4,4,8,1/
C
C     NATURAL BOUNDARY CONDITIONS:
C     THIS SUBROUTINE CALCULATES VELOCITY
C     OF LIQUID WATER, WATER VAPOR, OR AIR
C     IN UNITS OF M/S OR HEAT FLUX IN UNITS
C     OF W/M**2.  FLUXES ARE POSITIVE IN
C     THE DIRECTION OF THE OUTWARD POINTING
C     NORMAL TO THE ELEMENT SIDE.  FOR
C     EXAMPLE
C
      VALUE=0.0
C
C     ESSENTIAL BOUNDARY CONDITIONS:  THIS
C     SUBROUTINE PLACES VALUES DIRECTLY
C     IN THE VECTORS CONTAINING THE
C     DEPENDENT VARIABLES.  FOR EXAMPLE
C
      DO 10 I=1,3
      T(ISIDE1(I))=F(TIME)
   10 CONTINUE
      RETURN
      END
```

where

VALUE is the current value assigned for a flux-type boundary condition.

ISIDEn is a vector that defines the node points that lie on side n of a quadrilateral element. These vectors are useful for assigning essential boundary conditions.

When essential boundary conditions are being assigned, *i.e.*, using PEFFVARY, PVVARY, PAVARY, or TVARY, the boundary condition values are inserted directly in the appropriate vector of dependent variables as illustrated above. In the illustration, F(TIME) represents any function of time, but it could equally well be a function of any or all of the independent or dependent variables.

6.13.12 Mesh Generation

When an EXTDEF card is inserted in the nodal-point generation portion of the SETUP sequence of data cards, NORIA calls the user-defined subroutine EXTDEF. This subroutine provides a great deal of flexibility in defining node point locations. EXTDEF has the following form:

```
      SUBROUTINE EXTDEF(X,Y,MAXI)
      DIMENSION X(MAXI,1),Y(MAXI,1)
C
C     THIS SUBROUTINE CONTAINS FORTRAN
C     CODE TO DEFINE NODAL COORDINATES,
C     X AND Y, FOR EACH REQUIRED NODE POINT
C
      IMAX=11
      JMAX=21
      DO 10 I=1,IMAX
      DO 10 J=1,JMAX
      X(I,J)=F1(I,J)
      Y(I,J)=F2(I,J)
   10 CONTINUE
      RETURN
      END
```

where

X is an array of nodal x-coordinates. The two subscripts are the (I,J) label for a given node point.

Y is an array of nodal y-coordinates. The two subscripts are the (I,J) label for a given node point.

MAXI is an integer that specifies the largest I value that can be used in the mesh. The value of MAXI is specified on the SETUP command card.

IMAX in this illustration is the number of I-rows in the problem domain.

JMAX in this illustration is the number of J-rows in the problem domain.

F1, F2 are arbitrary functions that determine the x- and y-coordinates for each (I,J) pair.

6.14 Initial Conditions

Initial conditions for which each of the dependent variables is constant over a given material may be specified on the solid material property cards. Nonuniform initial conditions may be input to NORIA from an external storage device such as a disc pack or magnetic tape. The statement in NORIA that reads initial data is

```
      READ (19) TIME,TMAX,TMIN,PMAX,PMIN,PVMAX,PVMIN,
     1  PAMAX,PAMIN,NUMEL,((US(I,J),I=1,32),J=1,NUMEL)
```

where

TIME is the value of time corresponding to the initial condition.

TMAX, TMIN are the respective maximum and minimum values of temperature in the field.

PMAX, PMIN are the respective maximum and minimum values of effective liquid pressure in the field.

PVMAX, PVMIN are the respective maximum and minimum values of vapor pressure in the field.

PAMAX, PAMIN are the respective maximum and minimum values of air pressure in the field.

NUMEL is the total number of elements in the domain.

US is a matrix that contains all of the coefficients in the finite element expansions of the dependent variables. The first subscript indicates element number; the second subscript indicates degree of freedom. The first eight degrees of freedom are the finite element coefficients for effective liquid pressure numbered by node. Similarly, the second eight are for vapor pressure, the third eight are for air pressure, and the fourth eight are for temperature, all numbered by node. The above READ statement also applies for triangular elements (which contain 24 degrees of freedom per element); the extra degrees of freedom are ignored by NORIA.

Files that contain initial data to be read by NORIA must be given the local file name FT19.

6.15 Error Messages

NORIA contains a host of error checks that test for bad input data, improper dimensions, *etc.* When an error is encountered, an error message is printed and execution is usually terminated with a STOP 1. Error messages in NORIA are listed below.

- BCTIME-TIME CURVE NUMBER TOO LARGE—A time curve number greater than six was encountered.
- CONTOUR-UNRECOGNIZED COMMAND—Error in contour specification. The user should check spelling on contour type, looping command, and termination.
- DRIVER-UNRECOGNIZED COMMAND—A command was either misspelled or was expected and not found.
- FFLD-END OF DATA—An end of file was encountered on the input file. Check termination card.
- FFLDSB-INPUT VARIABLE TOO LONG—An input variable with more than ten characters was encountered.
- ELDATA-BC APPLIED TO AN UNDEFINED ELEMENT—A boundary condition was applied to an element not yet defined.
- ELDATA-BC ON IMPROPER SIDE OF ELEMENT—A boundary condition was specified for an improper element side.
- ELDATA-EXCESSIVE BOUNDARY CONDITIONS ON ELEMENT—More than eight boundary conditions have been applied to an element.
- ELDATA-LOOP PREVIOUSLY DEFINED—Error in specification of looping. Check for third loop within two existing loops.
- ELDATA-MAXIMUM NUMBER OF ELEMENTS EXCEEDED—Reduce number of elements used or redimension code.
- ELDATA-UNRECOGNIZED BOUNDARY CONDITION—Boundary condition type is in error. Check spelling.
- ELDATA-UNRECOGNIZED COMMAND—Erroneous element, boundary condition, or looping specification. Check spelling.
- FORMKFP-BAD ELEMENT JACOBIAN—A negative element area was found. Check element coordinates and connectivity.
- FORMKFP-TOO MANY TIME-VARYING BCS—More than six time-varying boundary conditions were specified.

- MATREAD-TOO MANY MATERIALS SPECIFIED—More than 10 solid materials were specified.
- NMESH-IJ MAX OR MIN EXCEEDS SPECIFIED VALUE—A grid point was found with an i_{max}, j_{max}, i_{min}, or j_{min} that exceeded the specified value on the SETUP card.
- PLOTZ-UNRECOGNIZED COMMAND—A plot command was used with an incomplete or misspelled parameter list.
- PRESOLN-INSUFFICIENT STORAGE—Insufficient storage for element connectivity. Reduce problem size or redimension code.
- PRESOLN-INSUFFICIENT STORAGE FOR FRONT WIDTH—Reduce problem size or redimension code.
- PRINTER-UNRECOGNIZED COMMAND—An OUTPUT command was used with an incomplete or misspelled parameter list.
- QUAD-ZERO JACOBIAN—A quadrilateral element with a zero area was found. Check element coordinates.
- RESTART-UNRECOGNIZED COMMAND—A restart command card was used with an incomplete or misspelled parameter list.
- SPLOT-TIMEPLANE DATA CARD NOT FOUND—A timeplane data card for profile plot is missing.
- SPLOT-UNRECOGNIZED COMMAND—There is an error in profile plot specification. Check spelling and termination.
- TIMEPLT-UNRECOGNIZED COMMAND—There is an error in history plot specification. Check spelling and termination.
- TRI-ZERO JACOBIAN—A triangular element with a zero area was found. Check element coordinates.
- UNZIPP-COMMAND CARD MISSING—A termination card is missing.
- UNZIPP-MAXIMUM STORAGE EXCEEDED—Maximum active storage exceeded. Reduce problem size or redimension code.
- UNZIPP-TIME STEP SIZE TOO SMALL—Time integration procedure attempted to use a time step smaller than the user-specified initial time step.
- UNZIPP-ZERO PIVOT—A zero was found on the diagonal of the equation being eliminated. Equation system is singular or element connectivity is in error.

6.16 Computer Requirements and Control Cards

Because NORIA is a large code (approximately 10000 source statements), it must be run on a relatively large computer. The current version of NORIA is designed primarily for the CRAY 1S, but it should be possible to adapt the code to run on other large scientific computers or smaller computers with virtual memory. The following discussion of control cards is directed at users who will run NORIA on SNLA's CRAY 1S.

The central processor time needed to run NORIA is somewhat difficult to predict because, depending on the difficulty of the problem, more than one Newton iteration may be required per time step, and time steps are repeated when errors are too large. Therefore, it is generally advisable to set the time limit for a run to be a factor of one and one-half to two higher than would be expected for a specified number of time steps, especially when embarking on a new problem. As a rule of thumb, NORIA requires about 10 seconds for compilation and start-up and about 0.075 seconds per element per time step for solution of the differential equations. More time should be allowed for large problems, *i.e.*, problems with 200 or more elements. Finally, an additional 10 or 20 seconds should be added to the estimated run time if significant postprocessing is to be performed, such as computing heat fluxes, computing velocities, or plotting.

NORIA is maintained as an UPDATE file in SNLA's permanent file system and may be accessed by the following control cards:

FILE, OLDPL, RT=2.

PFGET, OLDPL, NORIA1A, AU=NEBIXLE.

The attached file must be processed with UPDATE before compilation. Control cards to accomplish the updating and compilation are

```
NORIA, Txxxx, STSCZ.          NAME        BOX NO.
USER,USERNAME,{NOS PASSWORD}.
CHARGE,xxxxxxx.
SLTLIB.
UID.                          NAME        BOX NO.
CUPDATE, P=NORIA1A, UN=NEBIXLE, F.
CFT, I=COMPILE, L=0.
*.  INCLUDE THE FOLLOWING CARD FOR PLOTTING.
RSCORLB.
ACCESS, DN=$SCILIB, ID=U111BF3A.
*.  INCLUDE THE FOLLOWING TWO CARDS ONLY
*.  WHEN RESTARTING FROM A PREVIOUS RUN.
```

```
ACCESS, DN=TP13, PDN=GEOM, UQ.
ACCESS, DN=TP19, PDN=SOLN, UQ.
*.  INCLUDE THE FOLLOWING CARD FOR PLOTTING.
ASSIGN, DN=POPIN, A=FT10.
ASSIGN, DN=TP13, A=FT13.
ASSIGN, DN=POSTFIL, A=FT14.
ASSIGN, DN=TP19, A=FT19.
LDR, LIB=$SCILIB:RSCORS.
*.  INCLUDE THE FOLLOWING TWO CARDS ONLY
*.  WHEN PLOTTING.  THE x IN Rx INDICATES
*.  THE NUMBER OF THE RJE STATION TO WHICH
*.  THE PLOTS WILL BE ROUTED.
POP, POPIN, POPOUT, HC1.
XCOMQ, POPOUT, HC1, CS=Rx.
*.  THE FOLLOWING THREE CARDS CREATE A
*.  MICROFICHE COPY OF ALL OUTPUT DATA.
REWIND, DN=$OUT.
COPYD, I=$OUT, O=FOUT.
XFICHE, FOUT, T='problem description'.
*.  INCLUDE THE FOLLOWING CARD ONLY WHEN
*.  POSTPROCESSING WITH TRINITY.  THE x IN
*.  Vx INDICATES THE NODE NUMBER OF THE
*.  VAX TO WHICH THE FILE WILL BE ROUTED.
DISPOSE, DN=POSTFIL, DC=ST, DF=TR, TEXT=^
  'CTASK.'^
  'ROUTE, POSTFIL, DC=P8, TID=Vx.'.
*.  INCLUDE THE FOLLOWING TWO CARDS ONLY
*.  ON THE INITIAL RUN OF A SEQUENCE OF
*.  RESTART RUNS.  THE UQ PARAMETER ON
*.  THE ACCESS CARDS ABOVE WILL CAUSE
*.  ALL MODIFICATIONS TO TP13 AND TP19
*.  TO BE MADE TO THE PERMANENT FILES.
*.  ADDITIONAL USES OF THE SAVE COMMANDS
*.  WILL CREATE REDUNDANT FILES.
SAVE, DN=TP13, PDN=GEOM, RT=365.
SAVE, DN=TP19, PDN=SOLN, RT=365.
EXIT.
7-8-9
UPDATE DECK (CAN BE OMITTED IF NO UPDATES
  ARE REQUIRED.  HOWEVER, BOTH 7-8-9 CARDS
  ARE ALWAYS REQUIRED.)
7-8-9
NORIA DATA DECK
```

6-7-8-9

The above deck contains nearly all of the options ordinarily needed by the user. In many cases, only a subset of the above control commands are needed. The simplest way to temporarily delete control cards from the set is to precede any unwanted cards with an asterisk followed by a period (*.). This method converts the card to a comment card so that it will be ignored by the Cray operating system.

In many cases, the use of the restart option results in the creation of very large files. There is a danger that these files will be deleted from Cray disk space after a short period of retention time because policies at the Central Computing Facility (CCF) are intended to prevent the somewhat limited disk space from filling up. Therefore, it may be prudent to back up disk files, either by using MASS (Jones and Elsbernd, 1984) or by retaining files on magnetic tape.

7 Sample Calculation

A sample calculation is given in this section to demonstrate the command cards described in Section 6. This example illustrates some, but not all, of the salient features of NORIA. User manuals for SAGUARO (Eaton *et al.*, 1983) and MARIAH (Gartling and Hickox, 1980) may help familiarize the user with some of the features of NORIA not demonstrated here.

The sample calculation involves a small cylindrical heater emplaced coaxially at the center of a cylindrical block of volcanic tuff. The rock is initially partially saturated with water and is at room temperature ($25°C$). At some point in time (t=0) the heater is turned on so that it produces a constant power output of 50 W or 75 W. The external boundary of the block of tuff is maintained at room temperature and 1 atmosphere pressure. Also, it is assumed that neither liquid nor vapor passes through the external boundary. Liquid, vapor, air, and tuff properties are listed in Table 1. The control cards, user subroutines, and data cards used in this calculation are listed below.

The problem defined above has axial symmetry and, if gravity is neglected, the problem is also symmetric about the midplane perpendicular to the axis of symmetry. Thus, the problem domain considered here extends from $r = 0$ out to $r = R$ (the radius of the cylinder of rock), and from the midplane (denoted by $z = 0$) to the top of the cylinder of rock ($z = R$). Thus, the problem domain is R by R, and R is taken to be 0.3 m.

7.1 Results

Figures 7 and 8 respectively display the finite element mesh used in this calculation and an outline plot of the two materials. Results are shown in Figures 9 and 10. Figure 9 shows the steady temperature profiles along the midplane of the cylinder of rock (*i.e.*, along the lower boundary of the problem domain) for Cases 1 and 2. (Results shown here occur 6 wk after the heat source was turned on, at which time steady state has essentially been reached.) As expected, temperatures are highest near the heat source and tail off to room temperature at the outer edge of the block. Figure 10 shows the steady saturation profiles at the midplane of the cylinder of rock. Saturations are lowest near the heat source and rise to about the initial value of 80% at the outer edge of the cylinder.

Table 1: Properties Used in Sample Calculation

PROPERTY	WATER	VAPOR	AIR	TUFF
Density (kg/m^3)	1000			2770
Viscosity ($kg/m \cdot s$)	$A \cdot (T - 273)^B$	$0.2 \cdot 10^{-4}$	$0.2 \cdot 10^{-4}$	
A	0.01668			
B	-0.8987			
Heat Capacity (J/kg·K)	6076	0	0	713.8
Thermal Conductivity (W/m·K)	2.294	0	0	1.325
Average Pore Radius (m)				10^{-7}
Tortuosity				5
Porosity				0.17
Absolute Permeability (m^2)				$1.9 \cdot 10^{-17}$
Capillary Pressure (Pa)				
α				$5.19 \cdot 10^{-3}$
β				1.787
S_r				$6.69 \cdot 10^{-2}$

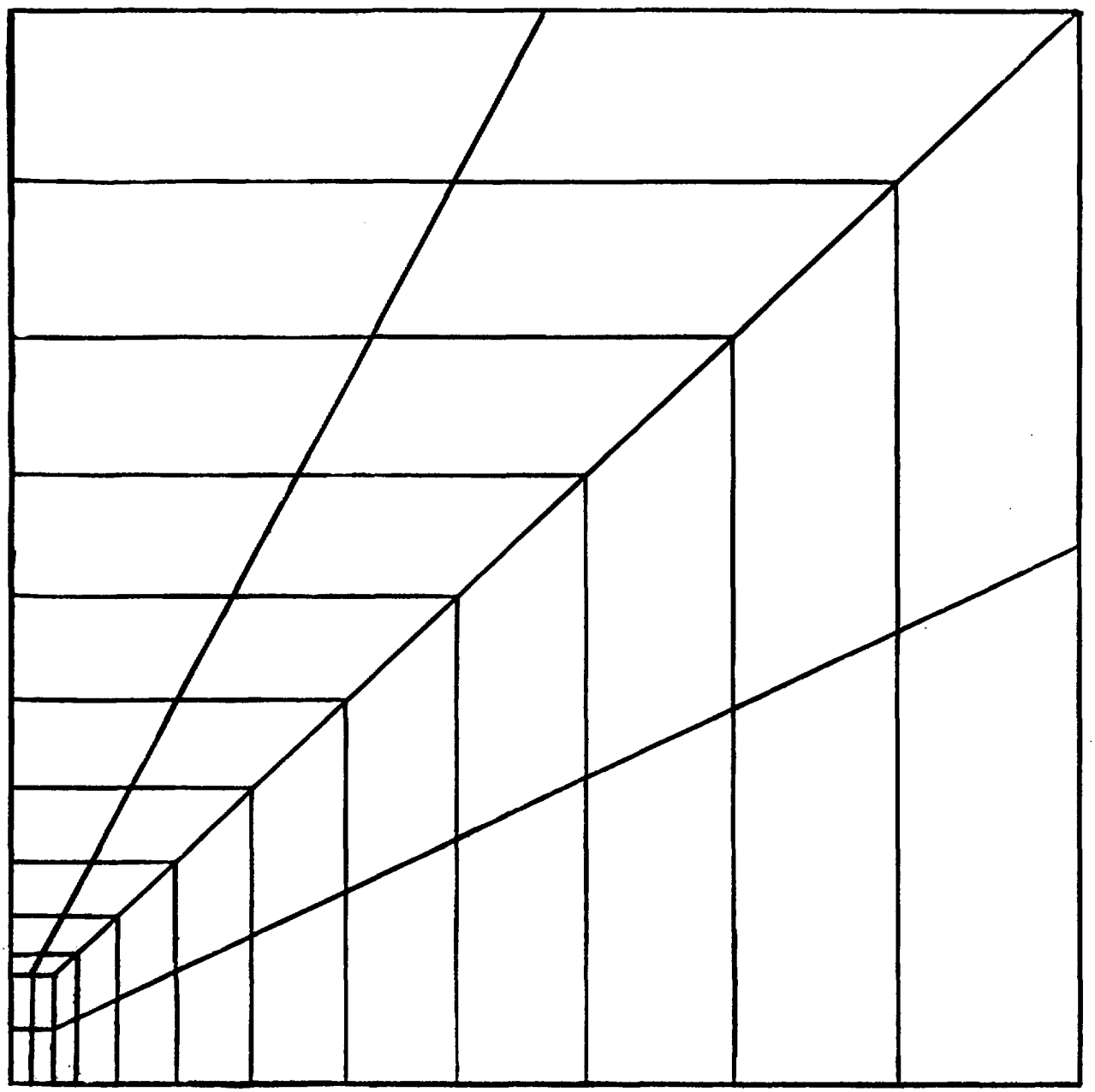


Figure 7: Finite Element Mesh for Sample Calculation

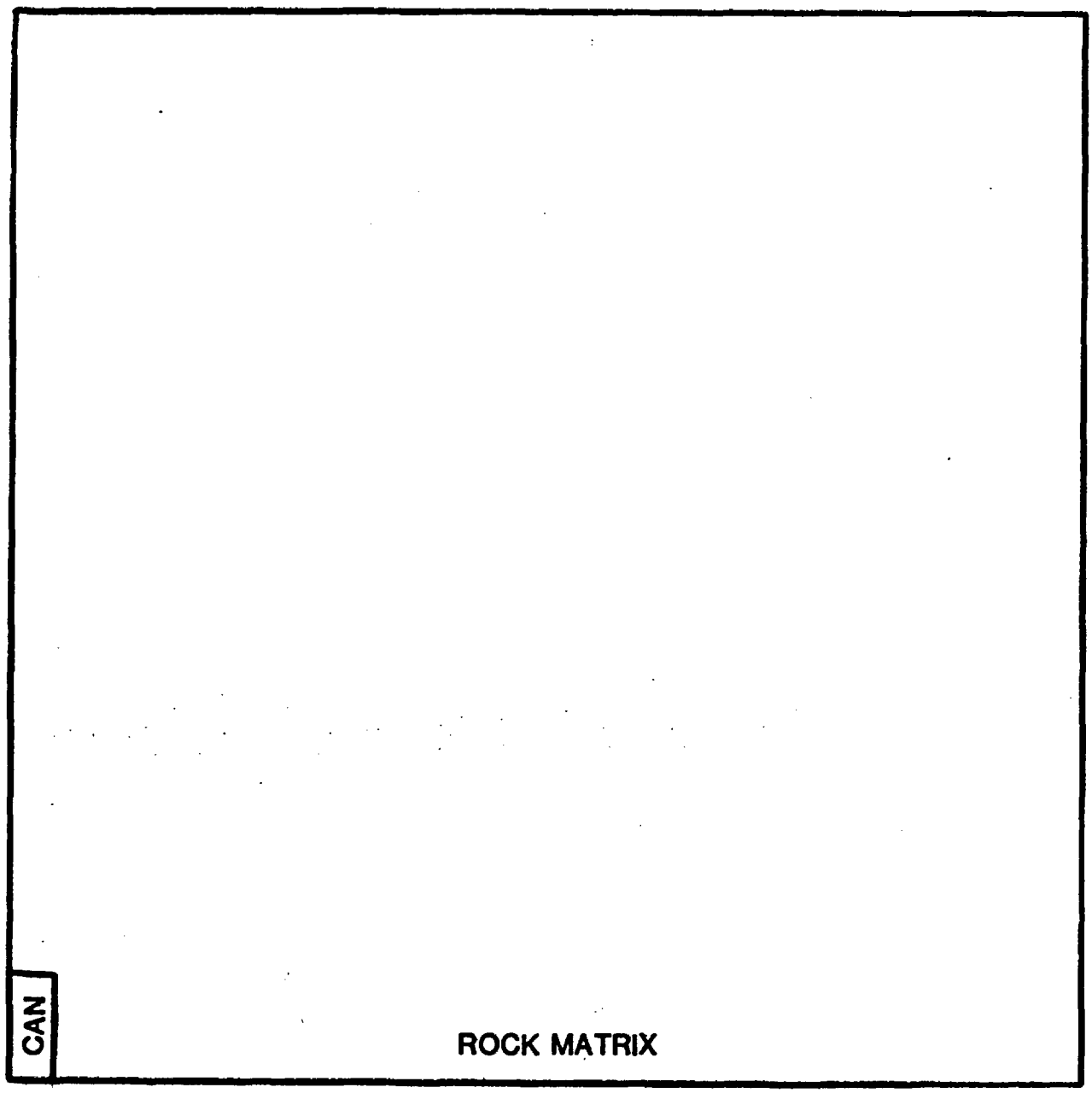


Figure 8: Material Outline Plot for Sample Calculation

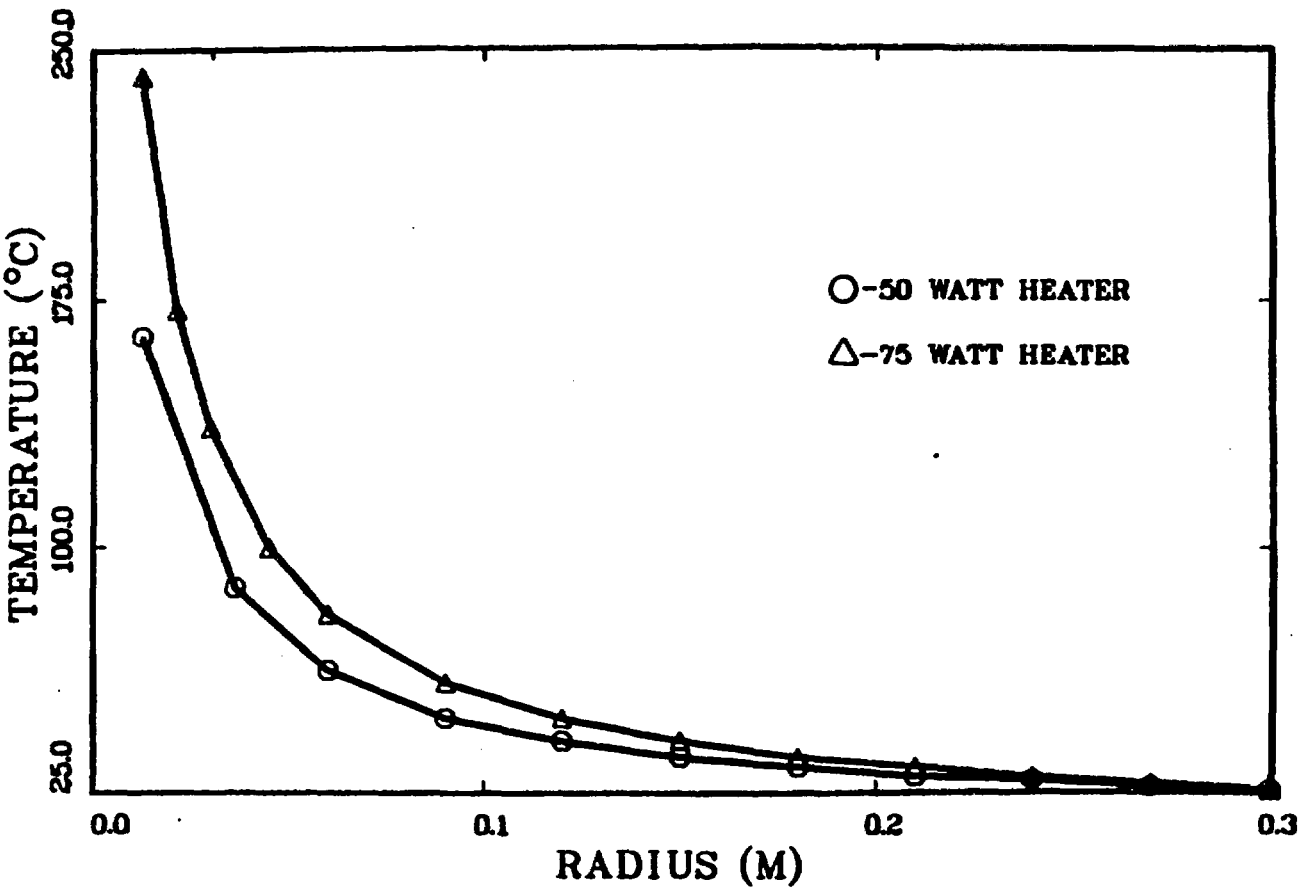


Figure 9: Steady Temperature Profiles at Rock Midplane

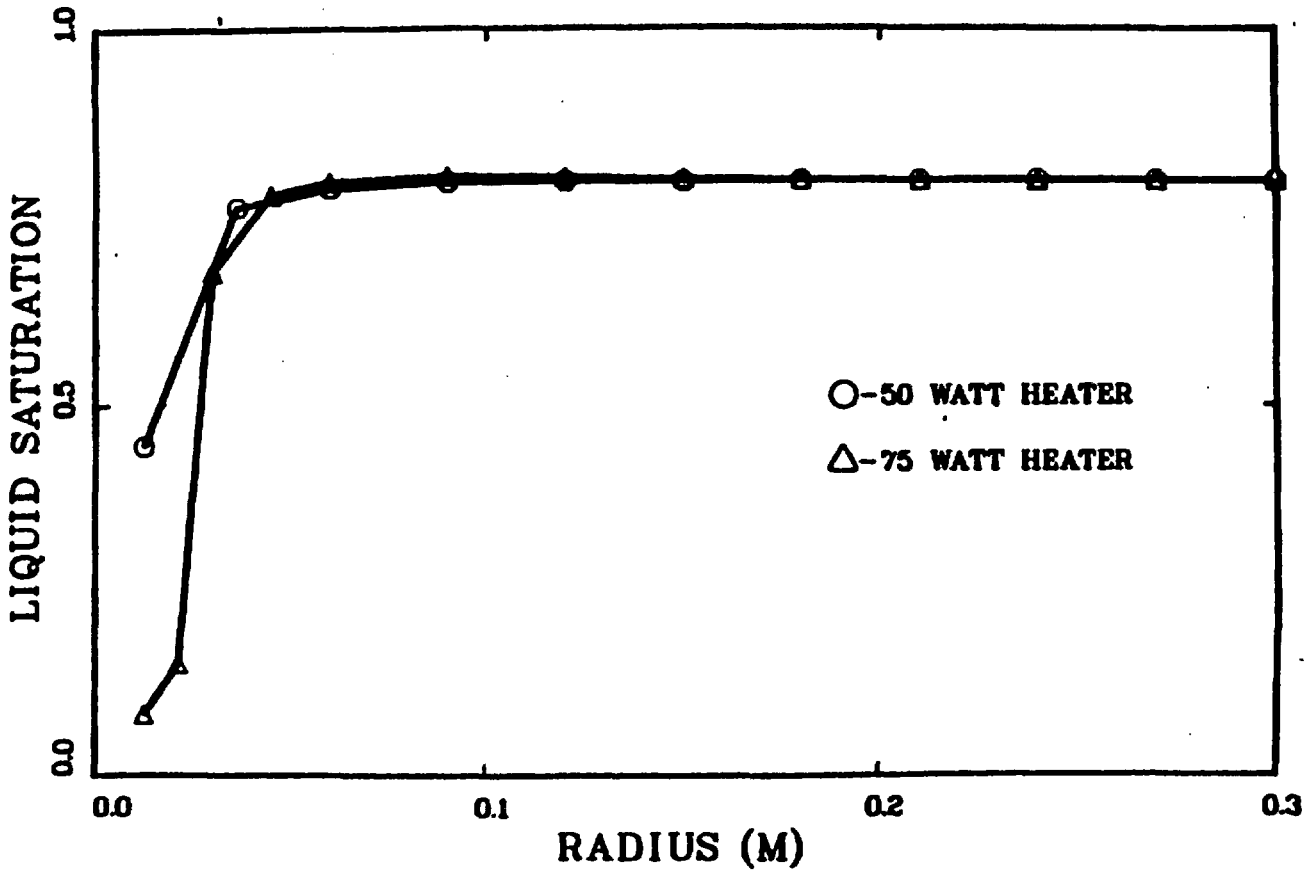


Figure 10: Steady Saturation Profiles at Rock Midplane

7.2 Control Cards

```
NORIA, T600, STSCZ.          NAME        BOX NO.
USER,USERNAME,{NOS PASSWORD}.
CHARGE,xxxxxxx.
SLTLIB.
UID.                         NAME        BOX NO.
CUPDATE, P=NORIA1A, UN=NEBIXLE, F.
CFT, I=COMPILE, L=0.
*.  INCLUDE THE FOLLOWING CARD FOR PLOTTING.
RSCORLB.
ACCESS, DN=$SCILIB, ID=U111BF3A.
*.  INCLUDE THE FOLLOWING CARD FOR PLOTTING.
ASSIGN, DN=POPIN, A=FT10.
ASSIGN, DN=TP13, A=FT13.
ASSIGN, DN=POSTFIL, A=FT14.
ASSIGN, DN=TP19, A=FT19.
LDR, LIB=$SCILIB:RSCORS.
*.  INCLUDE THE FOLLOWING TWO CARDS ONLY
*.  WHEN PLOTTING.  THE x IN Rx INDICATES
*.  THE NUMBER OF THE RJE STATION TO WHICH
*.  THE PLOTS WILL BE ROUTED.
POP, POPIN, POPOUT, HC1.
XCOMQ, POPOUT, HC1, CS=Rx.
*.  THE FOLLOWING THREE CARDS CREATE A
*.  MICROFICHE COPY OF ALL OUTPUT DATA.
REWIND, DN=$OUT.
COPYD, I=$OUT, O=FOUT.
XFICHE, FOUT, T='problem description'.
EXIT.
```

7.3 User Subroutines

```
      SUBROUTINE VISCL(AMUL,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION AMUL(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES VISCOSITY IN
C     UNITS OF KG/M/S FROM TEMPERATURE IN DEGREES
C     KELVIN.
```

```
C
      DO 10 I=1,NN
      AMUL(I)=0.01668*(T(I)-273.2015)**(-.8987)
   10 CONTINUE
      RETURN
      END

      SUBROUTINE FLUIDC(TH,T,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION TH(8),T(8),P(8),PV(8),PA(8),X(8),Y(8)
      DATA RSAT,ALPHA,BETA /0.0669,0.00519,1.787/
C
C     THIS SUBROUTINE CALCULATES MOISTURE
C     CONTENT FROM PRESSURES IN PASCALS
C
      ALAMBDA=1.0-1.0/BETA
      IF (MAT.NE.4) GO TO 30
      DO 20 J=1,NN
      PC=-P(J)+PV(J)+PA(J)
      SEF=(1.0+(ALPHA*PC/RHOG)**(BETA))**
     1  (-ALAMBDA)
      TH(J)=(SEF+RSAT-SEF*RSAT)*PHI
   20 CONTINUE
      RETURN
   30 CONTINUE
      DO 40 J=1,NN
      TH(J)=0.0
   40 CONTINUE
      RETURN
      END

      SUBROUTINE PERM(AKL11,AKL22,AKG11,AKG22,T,TH,
     1  P,PV,PA,X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION AKL11(8),AKL22(8),AKG11(8),AKG22(8),T(8),
     1  TH(8),P(8),PV(8),PA(8),X(8),Y(8)
      DATA RSAT,BETA,PERMST /0.0669,1.787,1.9E-18/
C
C     THIS SUBROUTINE CALCULATES PERMEABILITY
C     IN UNITS OF M**2
C
      IF (MAT.NE.4) GO TO 30
      ALAMBDA=1.0-1.0/BETA
      DO 10 J=1,NN
```

```
      SEF=(TH(J)/PHI-RSAT)/(1.0-RSAT)
      SEF=AMAX1(SEF,0.0001)
      SEF=AMIN1(SEF,0.9999)
      AKL11(J) = SQRT(SEF)*(1.0-(1.0-SEF**(1.0/ALAMBDA))
     1  **ALAMBDA)**2
   10 CONTINUE
      DO 20 J=1,NN
      AKG11(J) = 1.0-AKL11(J)
      AKL11(J) = AKL11(J)*PERMST
      AKG11(J) = AKG11(J)*PERMST
      AKL22(J) = AKL11(J)
      AKG22(J) = AKG11(J)
   20 CONTINUE
      RETURN
   30 CONTINUE
      DO 40 J=1,NN
      AKL11(J)=0.0
      AKL22(J)=0.0
      AKG11(J)=0.0
      AKG22(J)=0.0
   40 CONTINUE
      RETURN
      END

      SUBROUTINE KNUDIFV(DKV,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION DKV(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
      DATA PORRAD,RV,PI,TORT /1.0E-7,462.0,3.142,5.0/
C
C     THIS SUBROUTINE CALCULATES THE KNUDSEN
C     DIFFUSION COEFFICIENT FOR WATER VAPOR
C     IN UNITS OF M**2/S GIVEN TEMPERATURE
C     IN DEGREES KELVIN
C
      DO 10 I=1,NN
      DKV(I)=(2.0/3.0)*PORRAD*SQRT(8.0*RV*T(I)/PI)/TORT
   10 CONTINUE
      RETURN
      END

      SUBROUTINE KNUDIFA(DKA,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION DKA(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
```

```
      DATA PORRAD,RA,PI,TORT /1.0E-7,288.0,3.142,5.0/
C
C     THIS SUBROUTINE CALCULATES THE KNUDSEN
C     DIFFUSION COEFFICIENT FOR AIR
C     IN UNITS OF M**2/S GIVEN TEMPERATURE
C     IN DEGREES KELVIN
C
      DO 10 I=1,NN
      DKA(I)=(2.0/3.0)*PORRAD*SQRT(8.0*RA*T(I)/PI)/TORT
   10 CONTINUE
      RETURN
      END

      SUBROUTINE BINDIF(DB,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION DB(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
      DATA TORT /5.0/
C
C     THIS SUBROUTINE CALCULATES THE BINARY
C     DIFFUSION COEFFICIENT FOR WATER VAPOR
C     AND AIR IN UNITS OF M**2/S GIVEN
C     TEMPERATURE IN DEGREES KELVIN AND
C     PRESSURES IN PASCALS
C
      DO 10 I=1,NN
      DB(I)=2.254/(PV(I)+PA(I))*(T(I)/256.0)**1.81/TORT
   10 CONTINUE
      RETURN
      END

      SUBROUTINE LATHEAT(AL,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION AL(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES LATENT
C     HEAT OF VAPORIZATION FOR WATER
C     IN UNITS OF J/KG GIVEN
C     TEMPERATURE IN DEGREES KELVIN
C
      DO 10 I=1,NN
      AL(I)=2.746E6-4.301E3*(T(I)-273.0)
   10 CONTINUE
      RETURN
```

```
      END

      SUBROUTINE EVAP(VAPR,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION VAPR(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
      DATA RSAT /0.0669/
C
C     THIS SUBROUTINE CALCULATES RATE OF
C     EVAPORATION IN UNITS OF KG/M**3/S
C     GIVEN TEMPERATURE IN DEGREES KELVIN
C     AND PRESSURES IN PASCALS
C
      DO 10 I=1,NN
      SEF=(TH(I)/PHI-RSAT)/(1.0-RSAT)
      VAPR(I)=SEF*(PVAP(T(I))-PV(I))*1.0E-5
   10 CONTINUE
      RETURN
      END

      SUBROUTINE SOURCE(S,T,TH,P,PV,PA,
     1  X,Y,TIME,PHI,RHOG,NELEM,MAT,NN)
      DIMENSION S(8),T(8),TH(8),P(8),PV(8),PA(8),X(8),Y(8)
C
C     THIS SUBROUTINE CALCULATES VOLUMETRIC
C     HEAT GENERATION IN UNITS OF W/M**3
C
      DO 10 I=1,NN
C     THE FOLLOWING CARD WAS USED FOR THE 50
C     WATT HEATER CASE
      S(I)=1.6977E6
C     THE FOLLOWING CARD WAS USED FOR THE 75
C     WATT HEATER CASE
      S(I)=2.5465E6
   10 CONTINUE
      RETURN
      END
```

7.4 Data Cards

```
$  NORIA TEST RUN
SETUP,,25,PRESCRIBED
```

```
WATER,1,1000.0,VARIABLE,6076.0,2.294,,,,,0.0*
  VARIABLE,0.0,0.0
VAPOR,2,,0.200E-4,200.0,,,,VARIABLE,,462.0*
  0.228E-5,2.254,0.0,0.334E6
AIR,  3,,0.200E-4,100.0,,,,VARIABLE,,288.0,0.180E-5
ROCK,4,2770.0,713.8,,,,0.17,,,,,VARIABLE,NONE*
  299.0,0.136,0.3296E4,1.0E5
CAN ,5,2770.0,713.8,,,,0.00,,,,,VARIABLE,VARIABLE*
  299.0,0.136,0.3296E4,1.0E5
END
1,1,4,4
0.0,0.009375,0.009375,0.0
0.0,0.0,0.0225,0.0225
5,1,15,5
0.0125,0.05,0.05,0.0125
0.0,0.0,0.06522,0.03
5,5,15,9
0.0125,0.05,0.0,0.0
0.03,0.06522,0.06522,0.03
15,1,25,5,0.07,1.0,0.07,1.0
0.05,0.3,0.3,0.05
0.0,0.0,0.3,0.06522
15,5,25,9,0.07,1.0,0.07,1.0
0.05,0.3,0.0,0.0
0.06522,0.3,0.3,0.06522
END
QUAD8/4,5,1,1
QUAD8/4,5,3,1
QUAD8/4,5,3,3,5,3,5,5,5,7,4,3,5,4,5,6,3,4
QUAD8/4,5,1,3,3,3,5,7,5,9,2,3,3,4,5,8,1,4
ILOOP,10,2
JLOOP,4,2
QUAD8/4,4,5,1
JEND
IEND
JLOOP,4,2
BC,TSIDE,23,1,2,299.0
BC,PASIDE,23,1,2,1.0E5
JEND
END
FORMKF,AXISYM
UNZIPP,TAPE,3.6288E6,0.01,150,TAPE,11,8.640E4*
  1.728E5,2.592E5,3.456E5,4.320E5,5.184E5,6.048E5*
```

```
  1.2096E6,1.8144E6,2.4192E6,3.0240E6
END
PLOT,ELEMENTS,0.0,0.0,0.305,0.305,,,,,,,NUMBER
PLOT,OUTLINE,0.0,0.0,0.305,0.305
STOP,NOPRINT
```

8 Glossary

Absolute pressure—equivalent to thermodynamic pressure. A complete vacuum is the datum pressure level.

Basis function—one of a set of functions used to expand an arbitrary function. In the finite element method, the basis functions are defined on elements and dependent variables are expanded in them [see Equations (21)–(24) in Subsection 3.3].

Bilinear—refers to functions that are linear in each of two variables.

Binary diffusion—a term that describes the molecular diffusion of a species through another species. Here, binary diffusion occurs in gas composed of water vapor and air.

Effective pressure—an expression for pressure in the liquid phase defined following Equation (6) in Subsection 2.3.

Element coefficient matrix—a matrix that contains the contributions to the global coefficient matrix, that is the Jacobian matrix used in the Newton iteration procedure, from an individual element.

Element connectivity—a term used to describe the way in which the finite elements are interconnected. Two adjacent elements may be connected, *i.e.*, may allow mass and heat transfer through the common side, or may not be connected depending on the way nodal points are assigned to the elements.

Equilibrium vapor pressure—the pressure of vapor that would be in equilibrium with liquid at a specified temperature.

Gauss point—a quadrature point in a scheme devised by Gauss. (See "Quadrature".)

Global coefficient matrix—a term used here for the Jacobian matrix used in the Newton iteration procedure.

Knudsen diffusion—a term that describes the diffusion of molecules through a medium when the mean free path of a molecule is comparable or longer than a representative length scale for the medium. (Here, the representative length scale for the medium is an average pore radius.)

Natural coordinate system—a term describing the coordinate system in the mapped domain where a standard element lies (see "Standard element"). Here, the natural coordinate system is a rectangular Cartesian coordinate system and the mapped domain is defined by a subparametric or an isoparametric mapping of an element in the original problem domain.

Newton iteration procedure—a well-known iteration procedure for solving a set of nonlinear equations. The basis for the iteration procedure is a Taylor expansion that is truncated after the linear term. Applying Newton's iteration procedure to a set of nonlinear equations involves solving a system of linear equations at each iteration. A Newton step is the same as a Newton iteration.

Partial pressure—a thermodynamic term referring to pressure that a gas component would have if no other gas components were present. (This definition is valid for ideal gasses, which are assumed here.)

Predictor-corrector—a general category of numerical time integration schemes that involves two steps. The first step is to predict what the solution should be at the next time plane. The prediction is based on a knowledge of how the variables have behaved over past time steps. The corrector step uses the predicted values at the new time plane to improve upon the solution. The corrector step improves both the accuracy and stability of the overall time integration procedure.

Principal axes—the axes used to define an orthotropic property. The directions of the two principle axes are so chosen that the tensor used to define the orthotropic property is diagonal.

Quadrature—a general category of methods that serve to estimate the values of integrals.

Residual—an equation that measures the degree to which one of the governing equations is satisfied in some integral sense. Normally, residual equations are set equal to zero; however, in practice they can only be set equal to approximately zero if they contain nonlinear terms.

Residual moisture content—the moisture content below which moisture is not free to flow by pressure or density gradient.

Standard element—an element in the isoparametrically or subparametrically mapped domain. Standard elements have simple shapes, in NORIA either squares or isosceles triangles.

Weak form—a term that originated from the topic of variational calculus. There are several weak forms of a differential equation. One is the integral of the product of the differential equation with an arbitrary function, which is referred to as a weighting function. Other weak forms result from the application of Green's theorem to this integral equation.

Weighting function—an arbitrary function that appears in the weak form of a differential equation. (See "Weak form".)

9 References

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

Ceaglske, N.H. and Hougen, O. A. *Ind. Eng. Chem.* **29** 1937, pp. 805–813.

Eaton, R. R., Gartling, D. K., and Larson, D. E. SAGUARO—A Finite Element Computer Program for Partially Saturated Porous Flow Problems, *Sandia National Laboratories,* SAND82–2772, 1983.

Freeze, R. A. and Cherry, J. A. *Groundwater,* Prentice-Hall Inc. Englewood Cliffs, NJ, 1979.

Gartling, D.K. NACHOS—A Finite Element Computer Program for Incompressible Flow Problems, *Sandia National Laboratories,* SAND77–1333 and SAND77–1334, 1978.

Gartling, D. K. and Hickox, C. E. MARIAH—A Finite Element Computer Program for Incompressible Porous Flow Problems, *Sandia National Laboratories,* SAND79–1622 and SAND79–1623, 1980.

Gartling, D. K. COYOTE—A Finite Element Program for Nonlinear Heat Conduction Problems, *Sandia National Laboratories,* SAND77–1332 (revised), 1982.

Gartling, D. K. TRINITY—A Pre- and Postprocessing Program for Two-Dimensional Finite Element Data, *Sandia National Laboratories,* SAND84–2090, 1985.

Gray, D. E. *American Institute of Physics Handbook, Third Ed.,* McGraw-Hill, New York, NY, 1972.

Gresho, P. M., Lee, R. L., and Sani, R. L. On the Time Dependent Solution of the Incompressible Navier-Stokes Equations in Two- and Three-Dimensions, *Recent Advances in Numerical Methods in Fluids, Volume I,* Pineridge Press, Swansea, U. K., 1979.

Hadley, G. R. Theoretical Treatment of Evaporation Front Drying, *Int. J. Heat Mass Transfer* **25** 1982, pp. 1511–1522.

Hadley, G. R. Numerical Modeling of the Drying of Porous Materials, *Proceedings of the Fourth International Drying Symposium* **1** Tokyo, Japan 1984.

Hadley, G. R. PETROS—A Program for Calculating Transport of Heat, Water, Water Vapor, and Air Through a Porous Material, *Sandia National Laboratories,* SAND84–0878, 1985.

Hadley, G. R., Wilson, R. K., and Nunziato, J. W. Modeling Multiphase Mixtures: The Volume Averaging Method and the Continuum Theory of Mixtures, *to be submitted, 1985.*

Hammer, P. C., Marlowe, O. P., and Stroud, A. H. Numerical Integration over Simplexes and Cones, *Math. Tables Aids Comp.* **10** 1956, pp. 130–137.

Irons, B. M. A Frontal Solution Program for Finite Element Analysis, *Int. J. Num. Meth. Engng.* **2** 1970, pp. 5–32.

Klavetter, E. A. Analytical Expression for Surface Tension as a Function of Temperature, *Sandia National Laboratories,* memo to B. S. Langkopf, December 5, 1984.

Jones, R., and Elsbernd, A. Sandia Cray-1 Supplements, *Sandia National Laboratories,* 1984.

Luskin, M. and Rannacher, R. On the Smoothing Property of the Crank-Nicolson Scheme, *Applicable Analysis* **14** 1982, pp. 117–135.

Martinez, M. J. FEMTRAN—A Finite Element Computer Program for Simulating Radionuclide Transport Through Porous Media, *Sandia National Laboratories,* SAND84–0747, 1985.

Nimick, F. F., Bauer, S. J., and Tillerson, J. R. Recommended Matrix and Rock-Mass Bulk, Mechanical and Thermal Properties for Thermomechanical Stratigraphy of Yucca Mountain, *Sandia National Laboratories,* Keystone Document 6310–85–1, October 1984.

Peters, R. R. *et al.* Fracture and Matrix Hydrologic Characteristics of Tuffaceous Materials from Yucca Mountain, Nye County, Nevada, *Sandia National Laboratories,* SAND84–1471 (in preparation).

Pruess, K. and Wang, J. S. Y. TOUGH—A Numerical Model for Nonisothermal Unsaturated Flow to Study Waste Canister Heating Effects, *Materials Research Symposia Proc.* **26** Boston, MA 1983, pp. 1031–1038.

Pruess, K. Heat Driven Flow in Partially Saturated Fractured Porous Media, *Lawrence Berkeley Laboratory Annual Report* 1984.

Shampine, L. and Gordon, M. *Computer Solution of Ordinary Differential Equations: The Initial Value Problem,* W. Freeman and Company, San Francisco, CA, 1975.

Slattery, J. C. Two-Phase Flow Through Porous Media, *AIChE Journal,* **16** 1970, pp. 245–352.

Strang, G. and Fix, G. J. *An Analysis of the Finite Element Method,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973.

Udell, K. S. Heat Transfer in Porous Media Heated from Above with Evaporation, Condensation, and Capillary Effects, *J. Heat Trans.* **105** 1983, pp. 485–492.

Vargaftik, N. B. *Tables of Thermophysical Properties of Liquids and Gasses,* Hemisphere Pub. Corp., London, England, 1975.

Watson, J. T. R. *Viscosity of Gasses in Metric Units,* National Eng. Lab., Edinburgh, Scotland, 1972.

Whitaker, S. Diffusion and Dispersion in Porous Media, *AIChE Journal* **13** 1967, pp. 420–427.

Whitaker, S. The Transport Equations for Multi-Phase Systems, *Chem. Eng. Sci.* **28** 1973, pp. 139–147.

Zanotti, F. and Carbonell, R. G. Development of Transport Equations for Multiphase Systems—I and II, *Chem. Eng. Sci.* **39** 1984, pp. 263–297.

Distribution:

B. C. Rusche (RW-1)
Director
Office of Civilian Radioactive
Waste Management
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

J. W. Bennett (RW-22)
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

Ralph Stein (RW-23)
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

J. J. Fiore (RW-22)
Program Management Division
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

M. W. Frei (RW-23)
Engineering & Licensing Division
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

E. S. Burton (RW-25)
Siting Division
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

C. R. Cooley (RW-24)
Geosciences & Technology Division
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

T. P. Longo (RW-25)
Program Management Division
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

Cy Klingsberg (RW-24)
Geosciences and Technology Division
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

B. G. Gale (RW-25)
Siting Division
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

R. J. Blaney (RW-22)
Program Management Division
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

R. W. Gale (RW-40)
Office of Policy, Integration, and
Outreach
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

J. E. Shaheen (RW-44)
Outreach Programs
Office of Policy, Integration, and Outreach
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

J. O. Neff, Manager
Salt Repository Project Office
U.S. Department of Energy
505 King Avenue
Columbus, OH 43201

D. C. Newton (RW-23)
Engineering & Licensing Division
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington, D.C. 20585

O. L. Olson, Manager
Basalt Waste Isolation Project Office
U.S. Department of Energy
Richland Operations Office
Post Office Box 550
Richland, WA 99352

D. L. Vieth, Director (4)
Waste Management Project Office
U.S. Department of Energy
Post Office Box 14100
Las Vegas, NV 89114

D. F. Miller, Director
Office of Public Affairs
U.S. Department of Energy
Post Office Box 14100
Las Vegas, NV 89114

D. A. Nowack (12)
Office of Public Affairs
U.S. Department of Energy
Post Office Box 14100
Las Vegas, NV 89114

B. W. Church, Director
Health Physics Division
U.S. Department of Energy
Post Office Box 14100
Las Vegas, NV 89114

Chief, Repository Projects Branch
Division of Waste Management
U.S. Nuclear Regulatory Commission
Washington, D.C. 20555

Document Control Center
Division of Waste Management
U.S. Nuclear Regulatory Commission
Washington, D.C. 20555

S. A. Mann, Manager
Crystalline Rock Project Office
U.S. Department of Energy
9800 South Cass Avenue
Argonne, IL 60439

K. Street, Jr.
Lawrence Livermore National Laboratory
Post Office Box 808
Mail Stop L-209
Livermore, CA 94550

L. D. Ramspott (3)
Technical Project Officer for NNWSI
Lawrence Livermore National Laboratory
Post Office Box 808
Mail Stop L-204
Livermore, CA 94550

W. J. Purcell (RW-20)
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington DC 20585

D. T. Oakley (4)
Technical Project Officer for NNWSI
Los Alamos National Laboratory
Post Office Box 1663
Mail Stop F-671
Los Alamos, NM 87545

W. W. Dudley, Jr. (3)
Technical Project Officer for NNWSI
U.S. Geological Survey
Post Office Box 25046
418 Federal Center
Denver, CO 80225

NTS Section Leader
Repository Project Branch
Division of Waste Management
U.S. Nuclear Regulatory Commission
Washington, D.C. 20555

V. M. Glanzman
U.S. Geological Survey
Post Office Box 25046
913 Federal Center
Denver, CO 80225

P. T. Prestholt
NRC Site Representative
1050 East Flamingo Road
Suite 319
Las Vegas, NV 89109

M. E. Spaeth
Technical Project Officer for NNWSI
Science Applications
International Corporation
2769 South Highland Drive
Las Vegas, NV 89109

SAIC-T&MSS Library (2)
Science Applications
International Corporation
2950 South Highland Drive
Las Vegas, NV 89109

W. S. Twenhofel, Consultant
Science Applications
International Corporation
820 Estes Street
Lakewood, CO 80215

A. E. Gurrola
General Manager
Energy Support Division
Holmes & Narver, Inc.
Post Office Box 14340
Las Vegas, NV 89114

J. A. Cross, Manager
Las Vegas Branch
Fenix & Scisson, Inc.
Post Office Box 15408
Las Vegas, NV 89114

N. E. Carter
Battelle Columbus Laboratory
Office of Nuclear Waste Isolation
505 King Avenue
Columbus, OH 43201

John Fordham
Desert Research Institute
Water Resources Center
P.O. Box 60220
Reno, NV 89506

J. B. Wright
Technical Project Officer for NNWSI
Westinghouse Electric Corporation
Waste Technology Services Division
Nevada Operations
Post Office Box 708
Mail Stop 703
Mercury, NV 89023

ONWI Library
Battelle Columbus Laboratory
Office of Nuclear Waste Isolation
505 King Avenue
Columbus, OH 43201

M. W. Hewitt, Program Manager
Roy F. Weston, Inc.
2301 Research Blvd.,
Third Floor
Rockville, MD 20850

H. D. Cunningham
General Manager
Reynolds Electrical &
Engineering Co., Inc.
P.O. Box 14400
Mail Stop 555
Las Vegas, NV 89114

T. Hay, Executive Assistant
Office of the Governor
State of Nevada
Capitol Complex
Carson City, NV 89710

R. R. Loux, Jr., Director (3)
Nuclear Waste Project Office
State of Nevada
Capitol Complex
Carson City, NV 89710

C. H. Johnson, Technical
Program Manager
Nuclear Waste Project Office
State of Nevada
Capitol Complex
Carson City, NV 89710

Dr. Martin Mifflin
Desert Research Institute
Water Resources Center
Suite 1
2505 Chandler Avenue
Las Vegas, NV 89120

Department of Comprehensive
Planning
Clark County
225 Bridger Avenue, 7th Floor
Las Vegas, NV 89155

Lincoln County Commission
Lincoln County
Post Office Box 90
Pioche, NV 89043

Community Planning and
Development
City of North Las Vegas
Post Office Box 4086
North Las Vegas, NV 89030

City Manager
City of Henderson
Henderson, NV 89015

V. J. Cassella (RW-22)
Office of Geologic Repositories
U.S. Department of Energy
Forrestal Building
Washington, DC 20585

N. A. Norman
Project Manager
Bechtel National Inc.
P. O. Box 3965
San Francisco, CA 94119

Flo Butler
Los Alamos Technical Associates
1650 Trinity Drive
Los Alamos, New Mexico 87544

Planning Department
Nye County
Post Office Box 153
Tonopah, NV 89049

Economic Development
Department
City of Las Vegas
400 East Stewart Avenue
Las Vegas, NV 89101

Director of Community
Planning
City of Boulder City
Post Office Box 367
Boulder City, NV 89005

Commission of the
European Communities
200 Rue del la Loi
B-1049 Brussels
BELGIUM

Technical Information Center
Roy F. Weston, Inc.
2301 Research Blvd.,
Third Floor
Rockville, MD 20850

R. Harig
Parsons Brinkerhoff Quade &
Douglas, Inc.
1625 Van Ness Avenue
San Francisco, CA 94109-3678

Dr. Madan M. Singh, President
Engineers International, Inc.
98 East Naperville Road
Westmont, IL 60559-1595

Internal Distribution:

1510 J. W. Nunziato
1511 G. G. Weigand
1511 R. E. Benner
1511 N. E. Bixler (40)
1511 R. R. Eaton
1511 D. K. Gartling
1511 R. C. Givler
1511 P. Hopkins
1511 C. M. Korbin

1511 D. F. McTigue
1511 L. A. Mondy
1512 J. C. Cummings
1512 A. J. Russo
1512 J. E. Shepherd
1513 D. W. Larson
1513 M. E. Larsen
1513 J. A. Schutt
1513 C. E. Sisson
1513 R. K. Wilson
1520 D. J. McCloskey
1521 Z. E. Beisinger
1521 L. J. Branstetter
1521 R. D. Krieg
1521 C. M. Stone
1523 J. H. Biffle
1524 L. M. Taylor
1530 L. W. Davison
1531 S. L. Thompson
1540 W. C. Luth
1541 H. C. Hardee
1541 C. R. Carrigan
2644 D. M. Darsey
2644 C. J. Pavlakos
2646 M. R. Scott
6241 D. A. Glowka
6242 J. C. Dunn
6242 C. E. Hickox
6300 R. W. Lynch
6310 T. O. Hunter
6311 L. W. Scully
6311 L. Perrine (2)
6312 F. W. Bingham
6312 N. K. Hayden
6312 R. R. Peters
6312 R. W. Prindle
6313 E. A. Klavetter
6313 B. M. Schwartz
6313 R. M. Zimmerman
6314 J. R. Tillerson
6314 J. Fernandez
6315 S. Sinnak
6315 Y. T. Lin
6330 W. D. Weart
6332 WMT Library (20)
6310 NNWSICF
7531 B. F. Blackwell
8120 L. D. Bertholf
8124 R. J. Gallagher
8125 M. J. Fish
8233 V. K. Gabrielson
8242 W. E. Mason, Jr.
8331 R. J. Kee
8024 M. A. Pound
3141 C. M. Ostrander (5)
3151 W. L. Garner (3)
for DOE/TIC (Unlimited Release)
DOE/TIC (28)
(C. H. Dalin, 3154-3)

Org.	Bldg.	Name	Rec'd by	Org.	Bldg.	Name	Rec'd by

Sandia National Laboratories