

SAND94-1173
Unlimited Release
Printed October 1994

Distribution
Category **UC-905**

**COYOTE -
A FINITE ELEMENT COMPUTER PROGRAM
FOR NONLINEAR HEAT CONDUCTION
PROBLEMS
PART I - THEORETICAL BACKGROUND**

Version 2.5

David K. Gartling and Roy E. Hogan
dkgartl@sandia.gov and rehogan@sandia.gov
Engineering Sciences Center
Sandia National Laboratories
Albuquerque, New Mexico 87185

ABSTRACT

The theoretical and numerical background for the finite element computer program, COYOTE, is presented in detail. COYOTE is designed for the multi-dimensional analysis of nonlinear heat conduction problems and other types of diffusion problems. A general description of the boundary value problems treated by the program is presented. The finite element formulation and the associated numerical methods used in COYOTE are also outlined. Instructions for use of the code are documented in SAND94-1179; examples of problems analyzed with the code are provided in SAND94-1180.

Preface

At the time of release of the first version of COYOTE in mid-1978, it was not anticipated that the code would receive the heavy usage that it currently enjoys. In response to user needs, the original program has undergone several minor upgrades plus a major revision in the past several years. In addition, a preliminary three-dimensional version of COYOTE was developed though it was not formally documented. Continued requests for additional capabilities combined with the significant changes in computer hardware and improved numerical algorithms have dictated the need for a completely new version of the older codes. In rewriting the COYOTE program, the two and three-dimensional codes have been combined into a single software package. The present series of reports describe this latest version of the program package, COYOTE.

In an effort to make the programs more flexible and more generally applicable, a number of new capabilities and features have been added to COYOTE. The element library has been expanded to include linear and quadratic versions of all solid elements; specialty elements, such as bars and shells, have also been included. In order to improve the performance of the solution algorithms, the direct matrix solution methods have been replaced by iterative methods of the conjugate gradient type. Nonlinear steady-state solutions are obtained by a standard Picard method augmented with a relaxation scheme. Transient analyses are performed with a first-order, backward Euler method, a second-order, trapezoid rule or a first-order explicit procedure. All the integration methods can be run with a fixed time step or a dynamic time step selection procedure. The capability to perform surface-to-surface radiation in conjunction with the heat conduction problem has also been added to the code. A significant effort has been made to provide a rapid view factor capability for large problems; this capability can also be accessed in a stand-alone mode through the companion code, CHAPARRAL. Also, material motion, in either an Eulerian or Lagrangian frame, can be accommodated through user input or through coupling with a solid mechanics code; material addition and deletion can be simulated, if required. A general contact algorithm has been installed for use with both static and dynamic problems. COYOTE has been extended to allow chemically reacting materials to be considered, through the use of a stiff-solver package. Minor improvements in the allowed material models and boundary condition types and dependencies have also been incorporated in COYOTE. Input to the code has been redesigned to make more use of keywords and simplify data preparation. The code is written in standard FORTRAN 77 to increase its portability; machine dependent utilities are isolated in a portable library. Finally, new pre- and post-processing file formats have been developed to permit stand-alone mesh generators and graphics programs to be easily interfaced with the analysis package.

The significant contributions of several individuals to this code development effort must be acknowledged. The formulation and solution strategy for the chemical kinetics options in COYOTE are due to M. R. Baer (1512). The enclosure radiation package was developed by M. W. Glass (1511). Finally, the development and testing of the iterative matrix solver was undertaken by P. R. Schunk (1511) and J. R. Shadid (1421).

Contents

Preface	iii
1 Introduction	1
2 Formulation of the Basic Equations	3
2.1 Heat Conduction Equation	3
2.2 Boundary and Interface Conditions	5
2.3 Enclosure Radiation	8
2.4 Chemical Kinetics	10
3 Finite Element Equations	13
3.1 Heat Conduction Equation	13
3.2 Convection Equation	16
4 Elements and Element Matrix Construction	19
4.1 Triangular Elements (2D)	19
4.2 Quadrilateral Elements (2D)	21
4.3 Hexahedral Elements (3D)	22
4.4 Prism Elements (3D)	25
4.5 Tetrahedral Element (3D)	26
4.6 Bar Element (3D and 2D)	28
4.7 Shell Element (3D)	28
4.8 Spatial Derivatives and Integrals	32
4.9 Matrix Evaluation	34
4.10 Element Boundary Conditions and Source Terms	36
4.10.1 Volumetric Sources	36

4.10.2	Surface Fluxes	36
4.10.3	Internal Surface Fluxes	39
4.10.4	Specified Temperature Boundary Conditions	41
4.11	Matrix Equation	41
5	Solution Procedures	43
5.1	Steady-State Algorithms	44
5.1.1	Successive Substitution Method	44
5.1.2	Continuation Method	45
5.1.3	Convergence Criteria	45
5.2	Transient Algorithms	46
5.2.1	Forward/Backward Euler Integration	47
5.2.2	Adams-Bashforth/Trapezoid Rule Integration	47
5.2.3	Implicit Integration Procedures	48
5.2.4	Time Step Control	49
5.2.5	Initialization	50
5.2.6	Forward Euler Integration	50
5.2.7	Matrix Diagonalization	51
5.2.8	Stability and Time Step Control	52
5.3	Matrix Solution Procedures	53
5.4	Radiation View Factor Algorithms	54
5.5	Radiation Solution Algorithms	55
5.6	Chemical Reaction Solution Algorithm	57
5.7	Phase Change Algorithms	58
5.8	Contact Algorithm	61
6	Pre- and Post-Processing	63
6.1	Mesh Generation	63
6.2	Flux Computation	63
6.3	Heat Flow Function	65
6.4	Gas Fraction	68
6.5	Graphical Output	68
7	References	69

List of Figures

2.1	Schematic for boundary condition definitions.	5
2.2	Nomenclature for enclosure radiation.	9
3.1	Finite element discretization of a region.	14
4.1	Two-dimensional triangular elements.	20
4.2	Two-dimensional quadrilateral elements.	22
4.3	Three-dimensional brick elements.	23
4.4	Three-dimensional prism elements.	25
4.5	Three-dimensional tetrahedral elements.	27
4.6	Three-dimensional bar elements.	29
4.7	Three-dimensional shell elements.	30
4.8	Nomenclature for element surface computations.	37
4.9	Nomenclature for contact resistance formulation.	40
5.1	Definition of material properties for phase change computation.	59
6.1	Definition of element boundary for heat function computation.	67

Chapter 1

Introduction

The need for the engineering analysis of systems in which the transport of thermal energy occurs primarily through a conduction process is a common situation. For all but the simplest geometries and boundary conditions, analytic solutions to heat conduction problems are unavailable, thus forcing the analyst to call upon some type of approximate numerical procedure. A wide variety of numerical packages currently exist for such applications, ranging in sophistication from the large, general purpose, commercial codes, such as SINDA [1], P/THERMAL [2], and ABAQUS [3] to codes written by individuals for specific problem applications.

The original purpose for developing the finite element code described here, COYOTE, was to bridge the gap between the complex commercial codes and the more simplistic, individual application programs. COYOTE was designed to treat most of the standard conduction problems of interest with a user-oriented input structure and format that was easily learned and remembered. This general philosophy has been retained in the current version of the program, COYOTE, though the capabilities of the code have been significantly expanded.

The present document describes the theoretical and numerical background for the COYOTE program. This volume is intended as a background document for the user's manual found in [4]. Potential users of COYOTE are encouraged to become familiar with the present report and the example analyses report [5] before using the program.

In the following chapter the initial-boundary value problems treated by COYOTE II are described. Chapter 3 presents a brief description of the finite element method (FEM) and its application to the current problem. Chapters 4 and 5 outline the computational techniques that are involved in forming the individual element equations and the equation

solution procedures needed for the diffusion problem. Chapter 6 outlines the auxiliary calculation procedures found in the code.

Chapter 2

Formulation of the Basic Equations

COYOTE was primarily developed for the solution of multi-dimensional, nonlinear heat conduction problems. However, exploiting the analogy between the general heat conduction equation and other diffusion equations encountered in engineering and physics [6,7], COYOTE can also be used for other applications. In conjunction with the thermal diffusion problem, COYOTE was also structured to include solid phase chemical reactions and radiation heat transfer between conducting surfaces.

In the following chapter, the equation describing the basic heat conduction problem will be outlined along with the limiting assumptions used in developing COYOTE. A subsequent chapter will discuss all relevant boundary conditions for the heat transfer problem including enclosure radiation. The general formulation for problems involving chemical kinetics is also outlined. The theoretical development in each chapter will treat the general three-dimensional problem since the two-dimensional (plane or axisymmetric) case usually follows in a straightforward manner.

2.1 Heat Conduction Equation

The appropriate mathematical description of the heat conduction process in a stationary material region, Ω , is given by,

$$\rho C \frac{\partial T}{\partial t} = \frac{\partial}{\partial x_i} \left(k_{ij} \frac{\partial T}{\partial x_j} \right) + Q \quad (2.1)$$

where ρ is the material density, C the specific heat, k_{ij} the thermal conductivity tensor, Q the volumetric heat source, t the time, x_i the spatial coordinates and T the temperature.

Equation (2.1) is written for a fixed, Cartesian reference frame with the i, j indices running between 1 and 3, and the usual summation conventions in effect.

For the present work, each material is allowed to be heterogeneous with the conductivity tensor being at most, orthotropic (*i.e.*, k_{ij} may have three distinct components, k_{11}, k_{22} and k_{33} when written in terms of the principle material axes [6]). In the general case, the material properties may be functions of time, spatial location, chemical composition, and/or temperature. The volumetric heat source may also depend on time, spatial location and/or temperature; endothermic or exothermic energy release due to a chemical reaction is also included in the variation of Q .

Equation (2.1) describes the thermal conduction process within a single material. Conduction heat transfer between materials and convective and radiative energy exchange with the surrounding environment depends on a set of interface and boundary conditions. These aspects of the boundary value problem will be considered in the next chapter.

The partial differential equation given in (2.1) is in fact more generally applicable than indicated above. If a material coordinate (Lagrangian) description is adopted for the region, Ω , in place of the fixed frame Eulerian description, then equation (2.1) is also valid for a translating, rotating and/or deforming region, $\Omega(t)$. Since no equations of motion are included in the present formulation, it is assumed that the kinematics for the material region are completely specified. For rigid body motions such a prescription is relatively straightforward; new material coordinates are directly defined by a translation and rotation of the region. Material deformation is generally more complex and requires the solution of a solid mechanics problem. In the present formulation this data is assumed to be supplied from an external source. Because the allowance of this type of material motion adds little to the complexity of the boundary value problem, a Lagrangian description of the conduction problem will be permitted as an optional form in the present development. One complication that does arise in conjunction with solid body motion and deformation is the occurrence of contact. Due to the fact that contact influences the specification of boundary conditions, this problem will be addressed in a subsequent chapter.

Motion of a material under a fixed Eulerian coordinate description is also possible, though the energy equation must be modified for this condition. If the material velocity field is specified by the vector U with Cartesian components $u_j(x_i, t)$, then the energy equation in (2.1) is altered to

$$\rho C \left(\frac{\partial T}{\partial t} + u_j \frac{\partial T}{\partial x_j} \right) = \frac{\partial}{\partial x_i} \left(k_{ij} \frac{\partial T}{\partial x_j} \right) + Q. \quad (2.2)$$

All of the conditions stated above for equation (2.1) also pertain to equation (2.2). Since a momentum equation is not considered in the present formulation it is assumed that the

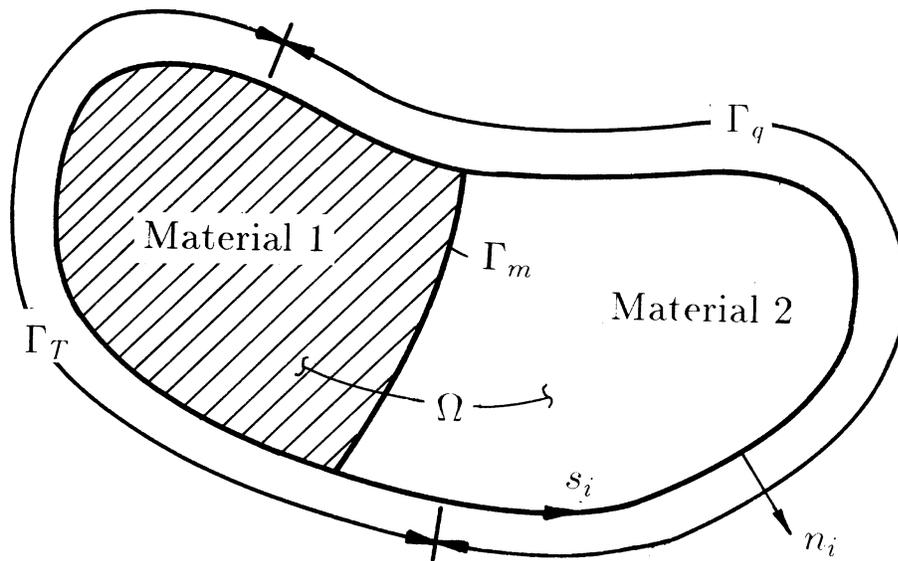


Figure 2.1: Schematic for boundary condition definitions.

velocity field is completely prescribed as a function of time and space. The additional advective term present in (2.2) adds minimal complexity to the formulation and will be allowed as an alternate energy equation when this type of material motion is prescribed. Note that equations (2.1) and (2.2) may occur in different regions of the same problem since they are both referenced to the same coordinate system. Mixtures of Eulerian and Lagrangian descriptions are also permissible.

2.2 Boundary and Interface Conditions

Boundary and interface conditions for the diffusion problem given by (2.1) or (2.2), are most easily described by reference to Figure 2.1. The region Ω is generally composed of a number of different materials, two of which are illustrated in Figure 2.1. The material interface is denoted by Γ_m ; the external boundary of the region Ω is defined by Γ . A two-dimensional representation of the region is used for simplicity.

The heat conduction problem requires that either the temperature or the heat flux be specified at all points of the boundary, Γ . In equation form, these conditions are given

by

$$T = f^T(s_i, t) \quad \text{on } \Gamma_T \quad (2.3)$$

$$\left(k_{ij} \frac{\partial T}{\partial x_j} \right) n_i + q_c + q_r = f^q(s_i, t) \quad \text{on } \Gamma_q. \quad (2.4)$$

In equations (2.3) and (2.4) the f^T and f^q functions are specified values of the known boundary temperature and heat flux. Also, n_i is the outward unit normal to the boundary Γ_q , s_i are coordinates defined on the boundary and $\Gamma = \Gamma_T \cup \Gamma_q$. The functions f^T and f^q are generally simple expressions for most boundaries of practical interest. The quantities q_c and q_r refer to the convective and radiative components of the boundary heat flux and are given by

$$q_c = h_c(s_i, T, t)(T - T_c) \quad (2.5)$$

$$q_r = \mathcal{F}(\epsilon)\sigma\epsilon(T^4 - T_r^4) \quad (2.6)$$

where h_c is the convective heat transfer coefficient, \mathcal{F} is the radiation form factor, σ is the Stefan-Boltzmann constant, and T_c and T_r are equilibrium temperatures for which no convection or radiation occurs. The form factor, \mathcal{F} , is related to the surface emissivity of the boundary, ϵ , and the position of the boundary relative to surrounding surfaces (see *e.g.*, [8]). This particular form of the radiation condition is useful for approximating the effects of simple, black-body radiation to a known temperature environment. More complex environments require the solution of the radiation transfer problem between the surrounding surfaces and the conducting body or between neighboring surfaces within the conducting region. This aspect of the problem is considered in Section 2.3.

Along the material interface Γ_m , the usual assumption is that the temperature and heat flux are continuous functions. That is,

$$T|_{\Gamma_m^+} = T|_{\Gamma_m^-} \quad (2.7)$$

$$\left(k_{ij} \frac{\partial T}{\partial x_j} \right) n_i \Big|_{\Gamma_m^+} = \left(k_{ij} \frac{\partial T}{\partial x_j} \right) n_i \Big|_{\Gamma_m^-} \quad (2.8)$$

where the superscript $+$, $-$ notation indicates properties or variables evaluated on either side of Γ_m . The above assumption is altered when contact resistance is a factor or when the interface is a phase boundary.

The problem of contact resistance between two stationary materials may be represented by either of two methods. One approach to modeling this effect assumes that the contact region is composed of a fictitious material, of small thickness, whose properties produce the appropriate resistance to heat flow across the interface. Typically, this gap material will have a negligible heat capacity and a nonlinear conductivity. In this case, the conditions in (2.7) and (2.8) are appropriate for all of the interfaces between the gap and solid materials. A slightly more mathematical representation of contact resistance

provides that the heat flux across the interface is described by an internal boundary condition of the form

$$q_g = h_g(s_i, \hat{T}, t)(T_m - T_s) \quad (2.9)$$

where h_g is an effective heat transfer coefficient for the gap region and \hat{T} is an average temperature between the surface temperatures, T_m and T_s . The subscripts m and s designate the “master” and “slave” sides of the contact surface, a distinction that is important in the numerical implementation of equation (2.9). The above flux condition is a generalization of the external boundary conditions presented in (2.5) and (2.6). In addition to representing contact resistance, this particular form of heat transfer between regions can also be used to simplify finite element mesh construction as shown in Section 4.10.3. Note that both of the above techniques for representing contact resistance between fixed surfaces may be used with the numerical methods considered here. When material motion or deformation is considered, the options for thermal boundary conditions along contacting surfaces are limited to the specification shown in (2.9) and the surface to surface radiation conditions described in a later chapter.

The conditions present at a phase boundary are somewhat more complex and require some additional equations. The difficulties at a phase boundary stem mainly from the fact that the location of the boundary Γ_m is not known *a priori*. Thus, the location of the moving interface becomes a required part of the solution. For the present application only melt/solid phase transitions will be considered. Also, it will be assumed that density changes upon change of phase may be neglected. With these assumptions, conditions at the interface are given by

$$T_f|_{\Gamma_m} = T_s|_{\Gamma_m} \quad (2.10)$$

$$k_f \frac{\partial T}{\partial n} \Big|_{\Gamma_m} - k_s \frac{\partial T}{\partial n} \Big|_{\Gamma_m} = \rho L \frac{\partial \Gamma_m}{\partial t} \quad (2.11)$$

where L is the latent heat and $\Gamma_m(t)$ is the unknown spatial position of the phase boundary. The subscript f and s denote the fluid and solid phases; the conductivities are shown as being isotropic though the solid phase tensor could be anisotropic. Basically, the melt/solid interface is taken to be a continuous temperature boundary with a discontinuous heat flux. This interface condition is not convenient for computational work when considering fixed grid methods. Following the work of Bonacini, *et al.* [9] and others, [10,11,12] the jump condition in (2.11) can be written in an alternate form using the so-called “enthalpy method.”

By observing that the latent heat, L , corresponds to the isothermal change in the enthalpy, H , for a material at the transition temperature, T_t , the following relation can be introduced

$$H(T) = \int_{T_{ref}}^T C(T) dT + L\eta(T - T_t) \quad (2.12)$$

with

$$\eta(\Delta) = \begin{cases} 1 & \text{if } \Delta \geq 0 \\ 0 & \text{if } \Delta < 0 \end{cases}$$

where η is the Heaviside function with argument Δ . The equivalent specific heat, C^* , is then introduced by

$$C^*(T) = \frac{dH}{dT} = C(T) + L\delta(T - T_t) \quad (2.13)$$

where δ is the Dirac delta function. Through the use of (2.13) latent heat effects may be included via the specific heat function and the jump in the heat flux (equation (2.11)) eliminated from the problem formulation. This particular approach to the problem has a theoretically sound basis as outlined in [9]. Moreover, it is a computationally effective modification since a two region problem with a jump condition has been converted to a single region problem with rapidly varying properties. For use in a finite element model, equation (2.13) requires additional modification. The details of this procedure are considered in references [5,10] and in Section 5.7.

Equations (2.1) through (2.13) provide a complete description of the boundary value problem for the temperature, T . When considering a time-dependent problem a suitable set of initial conditions describing the initial spatial distribution of T is also required.

2.3 Enclosure Radiation

Radiant energy exchange between neighboring surfaces of a region or between a region and its surroundings can produce large effects in the overall heat conduction problem. Though the radiation effects generally enter the conduction problem only through the boundary conditions, the coupling is especially strong due to the nonlinear dependence of the radiation on the surface temperature. COYOTE allows a restricted class of radiation problems to be solved in conjunction with the basic conduction problem.

Enclosure or surface-to-surface radiation in COYOTE is limited to diffuse gray surfaces. This assumption implies that all energy that is emitted or reflected from a surface is diffuse. Further, surface emissivity, ϵ , absorbtivity, α , and reflectivity, ρ , are independent of wavelength and direction so that $\epsilon(T) = \alpha(T) = 1 - \rho(T)$. Each individual area or surface that is considered in the radiation process must be at a uniform temperature; emitted and reflected energy are uniform over each such surface. Note that the definition of a surface is arbitrary and can be based on geometry alone or be defined to specifically satisfy the uniform temperature criteria.

With the above assumptions the radiation problem can be approached using the net-radiation method as described in [8]. For purposes of discussion consider the two-

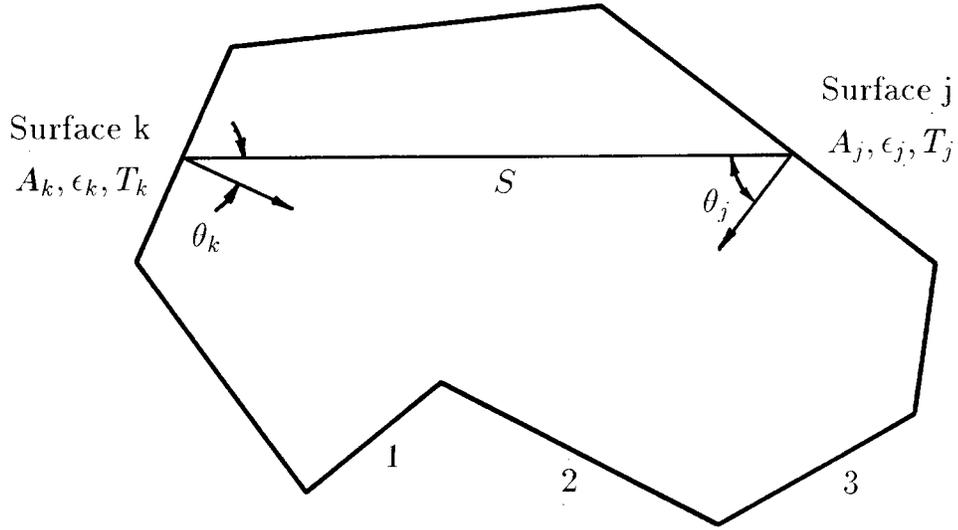


Figure 2.2: Nomenclature for enclosure radiation.

dimensional enclosure made up of N distinct surfaces as shown in Figure 2.2. Associated with each surface is a uniform temperature T_j ,

an area A_j and a surface emissivity ϵ_j . An energy balance for each surface in the enclosure leads to the following system of equations

$$\sum_{j=1}^N \left[\frac{\delta_{kj}}{\epsilon_j} - F_{k-j} \left(\frac{1 - \epsilon_j}{\epsilon_j} \right) \right] \frac{Q_j}{A_j} = \sum_{j=1}^N (\delta_{kj} - F_{k-j}) \sigma T_j^4. \quad (2.14)$$

Equation (2.14) relates the net energy loss, Q_j , from each surface to the temperature of each surface, where δ_{kj} is the unit tensor, σ is the Stefan-Boltzmann constant and F_{k-j} are radiation view (configuration) factors. The view factor is defined as the fraction of energy leaving a surface that arrives at a second surface. For surfaces with finite areas the view factors are defined by

$$F_{k-j} = \frac{1}{A_k} \int_{A_k} \int_{A_j} \frac{\cos \theta_k \cos \theta_j}{\pi S^2} dA_j dA_k \quad (2.15)$$

where S is the distance from a point on surface A_j to a point on surface A_k . The angles θ_j and θ_k are measured between the line S and the normals to the surface as shown in Figure 2.2 (see also [8]). It is clear from (2.15) that the view factors are purely geometric quantities that can in principle be evaluated for any given distribution of surfaces. Methods for evaluating F_{k-j} will be outlined in a later chapter.

For purposes of computation it is convenient to rearrange (2.14) into the following series of equations

$$\sum_{j=1}^N [\delta_{kj} - (1 - \epsilon_k)F_{k-j}] q_j^o = \epsilon_k \sigma T_k^4 \quad (2.16)$$

and

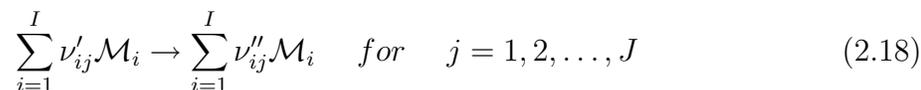
$$q_k = q_k^o - \sum_{j=1}^N F_{k-j} q_j^o. \quad (2.17)$$

Equations (2.16) and (2.17) are expressed in terms of the outgoing radiative flux for each surface, q_j^o , and the net flux from each surface $q_k = Q_k/A_k$. For known surface temperatures T_k in the enclosure, equation (2.16) can be solved for the outgoing radiative flux at each surface. Equation (2.17) then allows the net flux at each surface to be evaluated and applied to the conduction problem as a known flux boundary condition. The actual method of solution using (2.16) and (2.17) in a finite element context will be discussed in Section 5.5.

2.4 Chemical Kinetics

The thermal diffusion problem outlined in Sections 2.1-2.3 is modified significantly when one or more materials in the region Ω are allowed to undergo a chemical reaction. Each reactive material must be considered a mixture of I species with thermophysical properties now being a function of chemical composition. In addition, the J chemical reactions associated with a reactive material will normally produce a significant change in internal energy that subsequently provides a source term to the thermal diffusion problem. COY-OTE has been designed to handle a fairly general class of reaction-diffusion problems.

To describe a chemically reacting material, the stoichiometry, reaction kinetics and material property behavior must be specified. Consider a material involving I species with J reactions. The description of the allowed reactions (stoichiometry) is given by



where ν'_{ij}, ν''_{ij} are stoichiometric coefficients (usually integer values) and \mathcal{M}_i is the chemical symbol for the i th species. Generally, these expressions are given as reversible reactions; however, they are treated here as irreversible and the reversed reactions are specified as additional reaction steps. To accommodate expressions for global reactions, the stoichiometric coefficients are allowed to be non-integer.

For each step of the reaction, a reaction rate r_j , is defined in the form:

$$r_j = k_j(T) \prod_{i=1}^I [N_i]^{\mu_{ij}} \quad \text{for } j = 1, 2, \dots, J \quad (2.19)$$

where $[N_i]$ is the concentration variable for species i (or mole fraction), and μ_{ij} are the concentration exponents (usually $\mu_{ij} = \nu'_{ij}$ in kinetic theory, but here they are treated independently). Typically, the expressions for the kinetic coefficients $k_j(T)$, are given in an Arrhenius form

$$k_j(T) = T^{\beta_j} A_j \exp(-E_j/RT) \quad (2.20)$$

where β_j is the coefficient for a steric factor, A_j is the pre-exponential factor, E_j is the activation energy and the universal gas constant is R . It is convenient to define $\nu_{ij} = (\nu''_{ij} - \nu'_{ij})$ and thus the rate of change of the species (neglecting diffusion) are given as

$$\frac{d}{dt}[N_i] = \sum_{j=1}^J \nu_{ij} r_j \quad \text{for } i = 1, 2, \dots, I \quad (2.21)$$

The chemical reaction process is coupled directly to the thermal diffusion problem by the volumetric source term

$$Q_r = \sum_{j=1}^J q_j r_j \quad (2.22)$$

where q_j represents the known endothermic or exothermic energy release for reaction step j .

The material properties for the mixture are usually represented as mole fraction weighted averages of the I constituents. That is

$$(\rho C)_{mix} = \sum_{i=1}^I [N_i] (\rho C)_i \quad (2.23)$$

$$(k_{jk})_{mix} = \sum_{i=1}^I [N_i] (k_{jk})_i \quad (2.24)$$

where the constituent properties could still be functions of temperature. Another useful parameter for the mixture is the reacted gas fraction which is defined as the fraction of reacting material that exists in gas phase and is represented by

$$F_c = \frac{(1.0 - X_c) \sum_{i=1}^I [N_i](g)_i}{\sum_{i=1}^I [N_i]} \quad (2.25)$$

where $(g)_i$ is unity for gas phase species or zero for condensed phase species and X_c is the condensed fraction for the reactive material.

The species equations in (2.21) must be solved for each reactive material in conjunction with the thermal diffusion problem. This is a particularly difficult problem due to the disparity in time scales among the reaction equations and especially between the chemical processes and the thermal diffusion. In COYOTE, reactive materials are included via an operator splitting method and the use of stiff, ordinary differential equation solvers for the species equations. This methodology is outlined in Chapter 5.

Chapter 3

Finite Element Equations

The spatial discretization of the boundary value problem outlined in Chapter 2 by use of the finite element method may be approached by either of two procedures. Historically, the first and most popular approach consists of rewriting the boundary value problem in a variational form for use with the finite element approximation. An equivalent method uses the Galerkin form of the method of weighted residuals to create an integral form of the basic conservation law. This latter method is employed here.

3.1 Heat Conduction Equation

Let the region of interest, Ω , be divided into a number of simply shaped regions called finite elements, as shown in Figure 3.1. Within each element, a set of nodal points are established at which the dependent variable (*i.e.*, T) is evaluated. The variation of the temperature field within each element is approximated by an expansion of the form

$$T(x_i, t) = \sum_{n=1}^{N_e} \Theta_e^n(x_i) T_e^n(t) \quad (3.1)$$

where Θ_e represents the N_e interpolation functions and T_e are the N_e nodal point temperatures in the element. The ability to define simple, local approximations to the dependent variable is a primary feature of the finite element method. However, in order to develop a Galerkin, weighted residual formulation which is valid over the entire (global) domain, Ω , the local temperature variation in (3.1) must be extended to represent the temperature over the assemblage of elements. Standard compatibility properties of the piecewise element approximations given in (3.1) and the use of incidence relations (connectivity) for the assemblage of elements, allows a global temperature representation to be constructed.

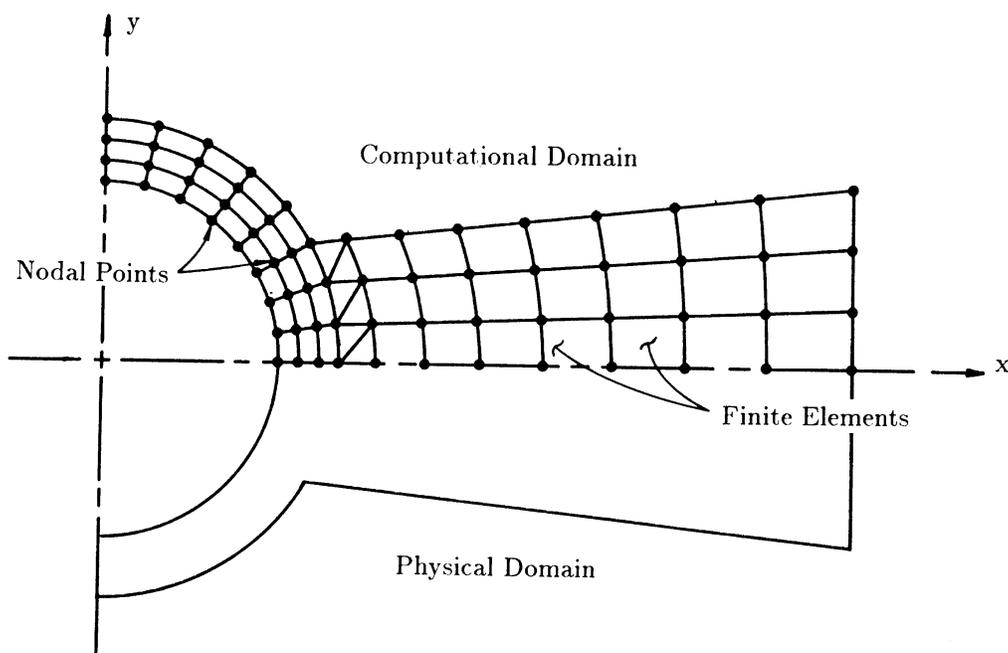


Figure 3.1: Finite element discretization of a region.

Details of this process may be found in [13,14]. The global temperature field has a form similar to (3.1) and is expressed as

$$T(x_i, t) = \sum_{n=1}^N \Theta^n(x_i) T^n(t) \quad (3.2)$$

or in matrix notation,

$$T(x_i, t) = \Theta^T(x_i) \mathbf{T}(t) \quad (3.3)$$

where Θ is now a vector of basis or interpolation functions defined on Ω , \mathbf{T} is a vector of nodal point unknowns, superscript T denotes a vector transpose, and N is the number of nodal points in the domain. Substitution of equation (3.3) into the partial differential equation (2.1) yields a set of residual equations, due to the approximate nature of equation (3.3). In functional form then

$$f_T(\Theta, \mathbf{T}) = R_T. \quad (3.4)$$

The Galerkin method guarantees the orthogonality of the residual vectors to the space spanned by the interpolation functions. This orthogonality is expressed by the inner product,

$$\langle \Theta, f_T \rangle = \langle \Theta, R_T \rangle = 0 \quad (3.5)$$

where $\langle a, b \rangle$ denotes the inner product defined by

$$\langle a, b \rangle = \int_{\Omega} a \cdot b \, d\Omega \quad (3.6)$$

Carrying out the above operations explicitly for the heat conduction equation (2.1) yields the following,

$$\int_{\Omega} \rho C \Theta \Theta^T \frac{\partial \mathbf{T}}{\partial t} \, d\Omega - \int_{\Omega} \Theta \frac{\partial}{\partial x_i} \left(k_{ij} \frac{\partial \Theta^T}{\partial x_j} \mathbf{T} \right) \, d\Omega - \int_{\Omega} \Theta Q \, d\Omega = 0. \quad (3.7)$$

As is standard practice [13], the second-order diffusion term in (3.7) may be rewritten using the divergence theorem to produce a first-order term plus a boundary integral.

$$\begin{aligned} \int_{\Omega} \rho C \Theta \Theta^T \frac{\partial \mathbf{T}}{\partial t} \, d\Omega + \int_{\Omega} \frac{\partial \Theta}{\partial x_i} \left(k_{ij} \frac{\partial \Theta^T}{\partial x_j} \mathbf{T} \right) \, d\Omega = \\ \int_{\Omega} \Theta Q \, d\Omega + \int_{\Gamma} \Theta \left(k_{ij} \frac{\partial \Theta^T}{\partial x_j} \mathbf{T} \right) n_i \, d\Gamma. \end{aligned} \quad (3.8)$$

Recognizing the boundary integral in (3.8) as part of the boundary condition specification in (2.4) allows this term to be reconfigured as

$$\int_{\Omega} \rho C \Theta \Theta^T \frac{\partial \mathbf{T}}{\partial t} \, d\Omega + \int_{\Omega} \frac{\partial \Theta}{\partial x_i} \left(k_{ij} \frac{\partial \Theta^T}{\partial x_j} \mathbf{T} \right) \, d\Omega = \int_{\Omega} \Theta Q \, d\Omega + \int_{\Gamma} \Theta (f^q - q_c - q_r) \, d\Gamma. \quad (3.9)$$

Noting that the nodal point unknowns are independent of the spatial integration and may be moved outside the integrals allows (3.9) to be written in the following matrix form

$$\mathbf{M}(\mathbf{T}) \dot{\mathbf{T}} + \mathbf{K}(\mathbf{T}) \mathbf{T} = \mathbf{F}_Q(\mathbf{T}) + \mathbf{F}(\mathbf{T}) \quad (3.10)$$

where the superposed dot indicates a time derivative and the possible dependencies on the dependent variable have been indicated. The individual matrices and vectors are defined by

$$\begin{aligned} \mathbf{M}(\mathbf{T}) &= \int_{\Omega} \rho C \Theta \Theta^T \, d\Omega \\ \mathbf{K}(\mathbf{T}) &= \int_{\Omega} \frac{\partial \Theta}{\partial x_i} \left(k_{ij} \frac{\partial \Theta^T}{\partial x_j} \right) \, d\Omega \\ \mathbf{F}_Q(\mathbf{T}) &= \int_{\Omega} \Theta Q \, d\Omega \\ \mathbf{F}(\mathbf{T}) &= \int_{\Gamma} \Theta (f^q - q_c - q_r) \, d\Gamma \end{aligned} \quad (3.11)$$

In deriving the matrix equation (3.10) from the Galerkin (weighted residual) method, it was important to work with the globally defined temperature approximation or basis

functions to avoid mathematical difficulties [13]. However, for purposes of implementation, it is more convenient to return to the local, element-level description of the equations. The process of constructing (assembling) the global matrices \mathbf{M} , \mathbf{K} , and \mathbf{F} from element level contributions is generally termed the direct stiffness method and is based primarily on the decomposition of the integrals defined for (3.11). Omitting the technical details [13,14], the global integrals over Ω can be written as the sum of the integrals over individual elements, Ω_e , which along with the appropriate incidence (or connectivity) relations between elements allows the following fundamental property to be defined

$$\mathbf{M} = \sum_{\mathbf{e}} \mathbf{M}_{\mathbf{e}} \quad ; \quad \mathbf{K} = \sum_{\mathbf{e}} \mathbf{K}_{\mathbf{e}} \quad ; \quad \mathbf{F} = \sum_{\mathbf{e}} \mathbf{F}_{\mathbf{e}} \quad (3.12)$$

In (3.12) the sum is over all the elements in the domain and the element matrices are defined by

$$\begin{aligned} \mathbf{M}_{\mathbf{e}} &= \int_{\Omega_e} \rho C \boldsymbol{\Theta}_{\mathbf{e}} \boldsymbol{\Theta}_{\mathbf{e}}^T d\Omega \\ \mathbf{K}_{\mathbf{e}} &= \int_{\Omega_e} \frac{\partial \boldsymbol{\Theta}_{\mathbf{e}}}{\partial x_i} k_{ij} \frac{\partial \boldsymbol{\Theta}_{\mathbf{e}}^T}{\partial x_j} d\Omega \\ \mathbf{F}_{\mathbf{Qe}} &= \int_{\Omega_e} \boldsymbol{\Theta}_{\mathbf{e}} Q d\Omega \\ \mathbf{F}_{\mathbf{e}} &= \int_{\Gamma_e} \boldsymbol{\Theta}_{\mathbf{e}} (f^q - q_c - q_r) d\Gamma \end{aligned} \quad (3.13)$$

Once the form of the element interpolation function, $\boldsymbol{\Theta}_{\mathbf{e}}$, is known and the element geometry is specified, the integrals in (3.13) can be evaluated. The global matrix problem is then constructed through use of (3.12).

3.2 Convection Equation

The derivation in the previous chapter considered the boundary value problem for thermal diffusion as described by equation (2.1). For an Eulerian coordinate frame with a specified material velocity, the advection-diffusion equation (2.2) is the appropriate continuum description for energy transport. The finite element form of this equation is derived by the same procedure as outlined above.

The temperature field is again represented by an expansion of the form given in (3.3)

$$T(x_i, t) = \boldsymbol{\Theta}^T(x_i) \mathbf{T}(t)$$

and the known velocity field is represented by a similar interpolation given by

$$u_j(x_i, t) = \boldsymbol{\Phi}^T(x_i) \mathbf{u}_j(t). \quad (3.14)$$

For generality, the interpolation function Φ in (3.14) is shown to be different from the interpolation for the temperature, though in practice these are usually the same function. Substituting (3.3) and (3.14) into (2.2) produces a residual equation of the form

$$f_T(\Theta, \Phi, \mathbf{u}_j, \mathbf{T}) = R_T. \quad (3.15)$$

Applying the Galerkin method with weight function Θ produces

$$\begin{aligned} \int_{\Omega} \rho C \Theta \Theta^T \frac{\partial \mathbf{T}}{\partial t} d\Omega + \int_{\Omega} \rho C \Theta \Phi^T \mathbf{u}_j \frac{\partial \Theta^T}{\partial x_j} \mathbf{T} d\Omega + \int_{\Omega} \frac{\partial \Theta}{\partial x_i} \left(k_{ij} \frac{\partial \Theta^T}{\partial x_j} \mathbf{T} \right) d\Omega = \\ \int_{\Omega} \Theta Q d\Omega + \int_{\Gamma} \Theta (f^q - q_c - q_r) d\Gamma \end{aligned} \quad (3.16)$$

where the second-order diffusion terms have been integrated by parts. The boundary conditions, being the same for this equation as equation (2.1), have been used to redefine the boundary integral. The matrix form of this equation is

$$\mathbf{M}(\mathbf{T}) \dot{\mathbf{T}} + \mathbf{C}(\mathbf{u}_j, \mathbf{T}) \mathbf{T} + \mathbf{K}(\mathbf{T}) \mathbf{T} = \mathbf{F}_Q(\mathbf{T}) + \mathbf{F}(\mathbf{T}) \quad (3.17)$$

which is directly analogous to (3.10). The global advection matrix is defined by

$$\mathbf{C}(\mathbf{u}_j, \mathbf{T}) = \int_{\Omega} \rho C \Theta \Phi^T \mathbf{u}_j \frac{\partial \Theta^T}{\partial x_j} d\Omega \quad (3.18)$$

while the element level matrix is

$$\mathbf{C}_e = \int_{\Omega_e} \rho C \Theta_e \Phi_e^T \mathbf{u}_j \frac{\partial \Theta_e^T}{\partial x_j} d\Omega. \quad (3.19)$$

As before, assembly of the global system follows the format defined in (3.12). The only difference between the convective-diffusive and diffusive forms of the energy equation is the occurrence of the advective matrix defined by (3.18). This term requires that a velocity field be defined as a function of space and time over the appropriate domain, Ω . Note that the convective term is unsymmetric and the global equation system is therefore unsymmetric when this formulation is employed. Also, this form of the energy equation is only available with the continuum elements in the element library; the convection formulation cannot be employed with bar or shell elements.

Chapter 4

Elements and Element Matrix Construction

The formulation of the equations for an individual element, as indicated by equations (3.10), (3.12), and (3.13), requires the specification of the shape function vectors for the approximation of the temperature. The form of the shape functions depend on the particular element being used; COYOTE employs two basic elements for two-dimensional analyses and three element types in the three-dimensional case. Special geometric elements, such as bars and shells, are also available. The interpolation functions for each of these elements are described below. For each element type both linear and quadratic interpolation is available; the higher-order functions are generally of the “serendipity” type [14] and avoid the use of nodes located in the interior of the element. Other element types, such as the higher-order Lagrange elements, could be added to COYOTE with minor code modifications. When the convective form of the energy equation is employed, the velocity interpolation is always constrained to be the same as the temperature interpolation for the element, *i.e.* $\Phi = \Theta$.

4.1 Triangular Elements (2D)

The triangular elements used in two-dimensional applications of COYOTE consist of a straight-sided, three-node element and a six-node element as shown in Figure 4.1. The linear interpolation function for the three-node element is given by

$$\Theta_1 = \left\{ \begin{array}{c} L_1 \\ L_2 \\ L_3 \end{array} \right\} \quad (4.1)$$

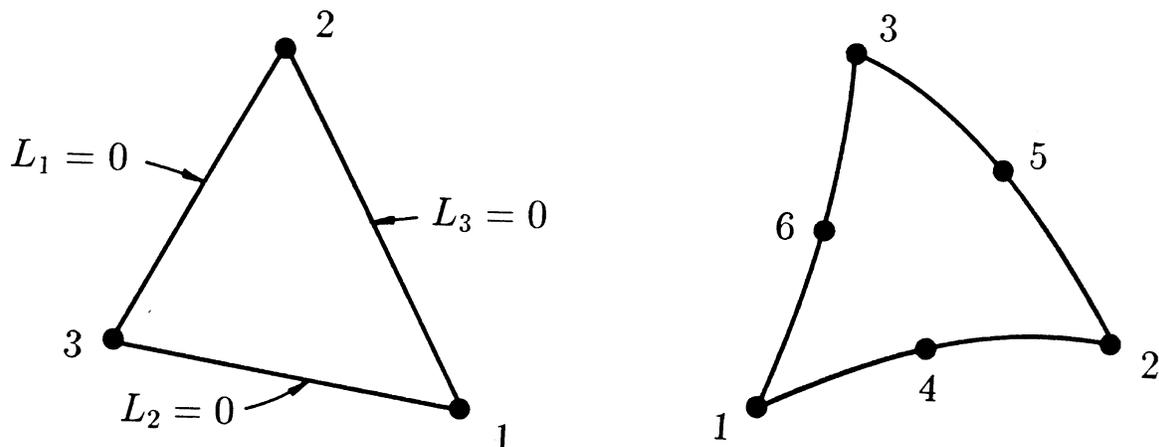


Figure 4.1: Two-dimensional triangular elements.

and the corresponding quadratic function for the six-node element is

$$\Theta_{\mathbf{q}} = \left\{ \begin{array}{l} L_1(2L_1 - 1) \\ L_2(2L_2 - 1) \\ L_3(2L_3 - 1) \\ 4L_1L_2 \\ 4L_2L_3 \\ 4L_3L_1 \end{array} \right\}. \quad (4.2)$$

The ordering of the functions in (4.1) and (4.2) corresponds to the ordering of the nodes shown in Figure 4.1. The shape functions are expressed in terms of the area or natural coordinates, L_i , for a triangle [13,14] which range from 0 to 1, and are related by the auxiliary condition $L_1 + L_2 + L_3 = 1$ (*i.e.*, there are only two independent area coordinates).

When the element interpolation functions are written in terms of the area coordinates, the relationship between the physical coordinates x, y (or r, z in the axisymmetric case) and the element coordinates is obtained from the parametric mapping concept originally developed by Ergatoudis, *et al.* [15]. That is, the coordinate transformation is given by

$$x = \mathbf{\Upsilon}^T \mathbf{x} \quad ; \quad y = \mathbf{\Upsilon}^T \mathbf{y} \quad (4.3)$$

where $\mathbf{\Upsilon}$ is a vector of interpolation functions on the triangle and the \mathbf{x}, \mathbf{y} are vectors of coordinates describing the geometry of the element (generally, nodal point coordinates).

The transformation given in (4.3) is quite general and allows for the description of curved-sided elements. In the present case, if $\Upsilon = \Theta_1$, a linear interpolation of the element boundary is possible. When $\Upsilon = \Theta_q$, a quadratic interpolation of the element geometry is allowed. Note that when the functions defining the element geometry are the same as those defining the dependent variable the element is termed isoparametric; a geometric description which is lower order than the dependent variable is defined as subparametric. COYOTE directly supports only isoparametric elements; straight-sided, higher order elements can be utilized by appropriately locating mid-edge nodes.

4.2 Quadrilateral Elements (2D)

Two types of quadrilateral elements are used in COYOTE – a four-node and an eight-node element. For the linear, four node element the interpolation functions are given by

$$\Theta_1 = \begin{Bmatrix} 1/4(1-s)(1-t) \\ 1/4(1+s)(1-t) \\ 1/4(1+s)(1+t) \\ 1/4(1-s)(1+t) \end{Bmatrix}. \quad (4.4)$$

The ordering of the functions in (4.4) corresponds to the nodal point ordering shown in Figure 4.2a. The interpolation functions are written in terms of the normalized or natural coordinates for the element, s, t , which vary from -1 to $+1$ as shown in the figure.

The eight-node element uses the biquadratic, “serendipity” functions [14] given by

$$\Theta_q = \begin{Bmatrix} 1/4(1-s)(1-t)(-s-t-1) \\ 1/4(1+s)(1-t)(s-t-1) \\ 1/4(1+s)(1+t)(s+t-1) \\ 1/4(1-s)(1+t)(-s+t-1) \\ 1/2(1-s^2)(1-t) \\ 1/2(1+s)(1-t^2) \\ 1/2(1-s^2)(1+t) \\ 1/2(1-s)(1-t^2) \end{Bmatrix}. \quad (4.5)$$

The parametric mapping concept described for the triangular element is also available for use with the quadrilaterals. Therefore, to relate the global coordinates x, y (or r, z) to the local s, t system, let

$$x = \Upsilon^T \mathbf{x} \quad ; \quad y = \Upsilon^T \mathbf{y}. \quad (4.6)$$

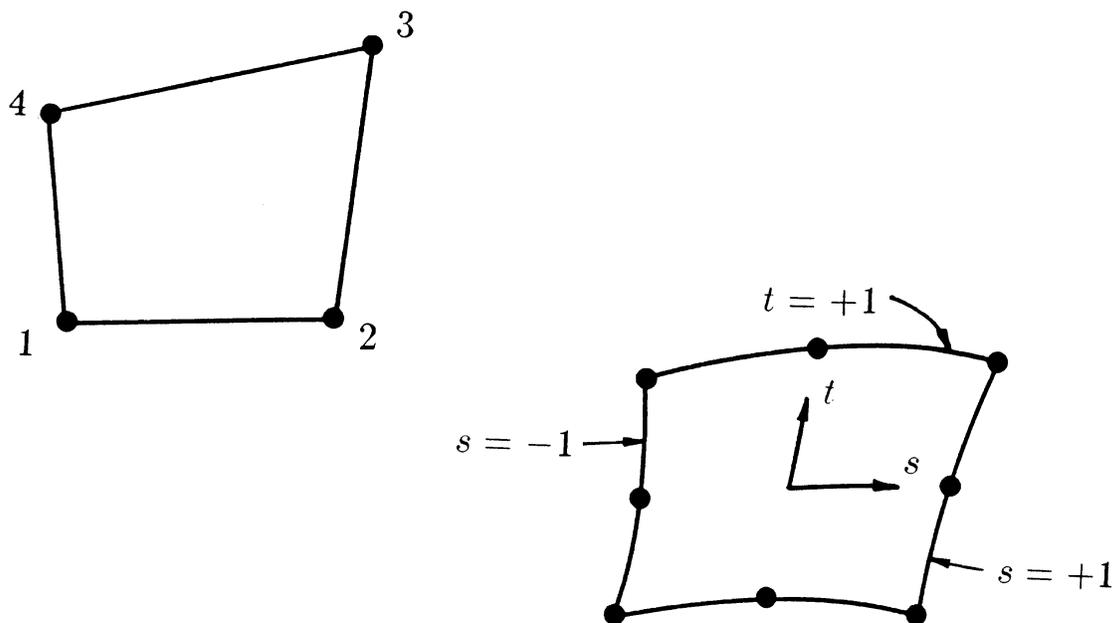


Figure 4.2: Two-dimensional quadrilateral elements.

where Υ may be either a linear or quadratic (“serendipity”) interpolation function. Again, COYOTE only supports the formulation of isoparametric quadrilaterals, though subparametric elements can be employed through the proper location of mid-edge nodes.

4.3 Hexahedral Elements (3D)

The hexahedral or brick elements available in COYOTE for three-dimensional analyses consist of a straight-edged, linear, eight-node element and a curved-sided, quadratic, twenty-node element as shown in Figure 4.3. The linear element has shape functions given by

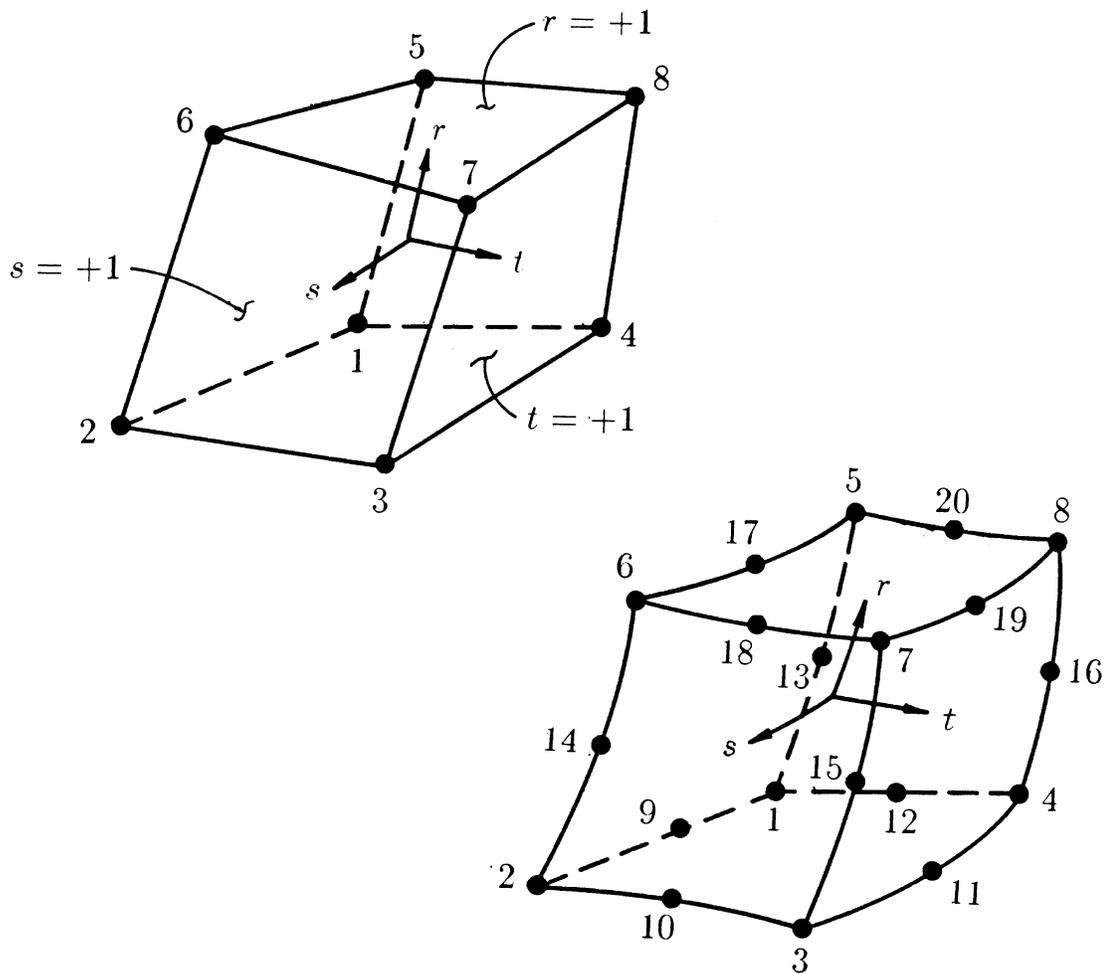


Figure 4.3: Three-dimensional brick elements.

$$\Theta_1 = \left\{ \begin{array}{l} 1/8(1-s)(1-t)(1-r) \\ 1/8(1+s)(1-t)(1-r) \\ 1/8(1+s)(1+t)(1-r) \\ 1/8(1-s)(1+t)(1-r) \\ 1/8(1-s)(1-t)(1+r) \\ 1/8(1+s)(1-t)(1+r) \\ 1/8(1+s)(1+t)(1+r) \\ 1/8(1-s)(1+t)(1+r) \end{array} \right\}. \quad (4.7)$$

The quadratic shape functions for the twenty-node element are given by

$$\Theta_q = \left\{ \begin{array}{l} 1/8(1-s)(1-t)(1-r)(-s-t-r-2) \\ 1/8(1+s)(1-t)(1-r)(s-t-r-2) \\ 1/8(1+s)(1+t)(1-r)(s+t-r-2) \\ 1/8(1-s)(1+t)(1-r)(-s+t-r-2) \\ 1/8(1-s)(1-t)(1+r)(-s-t+r-2) \\ 1/8(1+s)(1-t)(1+r)(s-t+r-2) \\ 1/8(1+s)(1+t)(1+r)(s+t+r-2) \\ 1/8(1-s)(1+t)(1+r)(-s+t+r-2) \\ 1/4(1-s^2)(1-t)(1-r) \\ 1/4(1+s)(1-t^2)(1-r) \\ 1/4(1-s^2)(1+t)(1-r) \\ 1/4(1-s)(1-t^2)(1-r) \\ 1/4(1-s)(1-t)(1-r^2) \\ 1/4(1+s)(1-t)(1-r^2) \\ 1/4(1+s)(1+t)(1-r^2) \\ 1/4(1-s)(1+t)(1-r^2) \\ 1/4(1-s^2)(1-t)(1+r) \\ 1/4(1+s)(1-t^2)(1+r) \\ 1/4(1-s^2)(1+t)(1+r) \\ 1/4(1-s)(1-t^2)(1+r) \end{array} \right\}. \quad (4.8)$$

The functions in (4.7) and (4.8) are ordered according to the nodal point ordering shown in Figure 4.3 and are written in terms of the local, normalized coordinates, s, t, r , which range from -1 to $+1$. COYOTE allows only the isoparametric form of each three-dimensional element where

$$x = \Upsilon^T \mathbf{x} \quad ; \quad y = \Upsilon^T \mathbf{y} \quad ; \quad z = \Upsilon^T \mathbf{z} \quad (4.9)$$

and Υ takes on the appropriate linear or quadratic form.

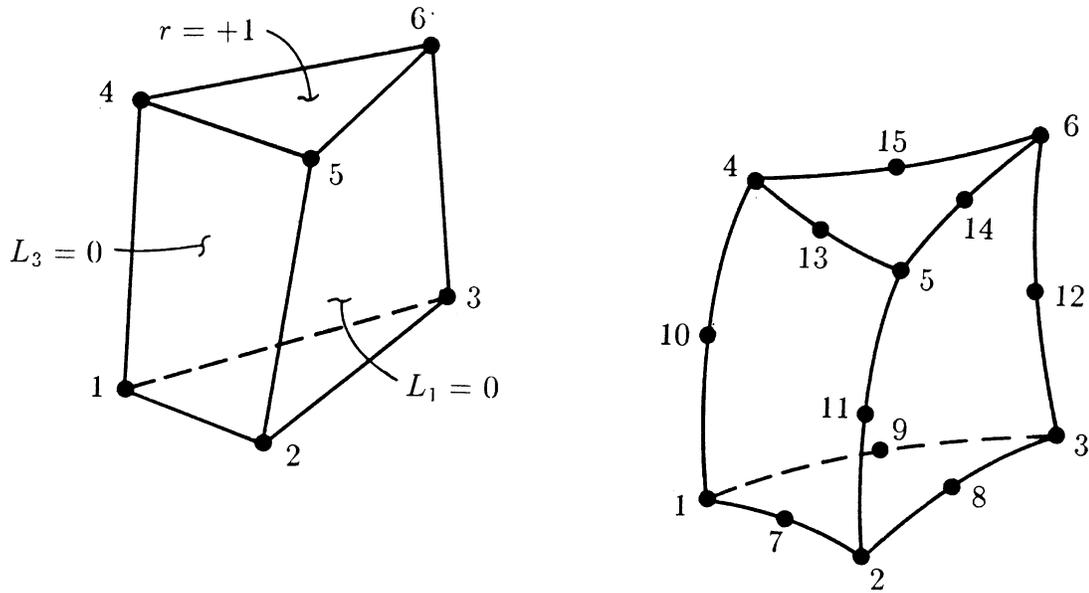


Figure 4.4: Three-dimensional prism elements.

4.4 Prism Elements (3D)

COYOTE employs a linear and quadratic version of a triangular prism or wedge element. The linear, straight-sided, six-node prism and curved-sided, fifteen-node quadratic element are shown in Figure 4.4. The shape functions for these elements are given by

$$\Theta_1 = \left\{ \begin{array}{l} 1/2L_1(1-r) \\ 1/2L_2(1-r) \\ 1/2L_3(1-r) \\ 1/2L_1(1+r) \\ 1/2L_2(1+r) \\ 1/2L_3(1+r) \end{array} \right\} \quad (4.10)$$

and

$$\Theta_{\mathbf{q}} = \left\{ \begin{array}{l} 1/2L_1[(2L_1 - 1)(1 - r) - (1 - r^2)] \\ 1/2L_2[(2L_2 - 1)(1 - r) - (1 - r^2)] \\ 1/2L_3[(2L_3 - 1)(1 - r) - (1 - r^2)] \\ 1/2L_1[(2L_1 - 1)(1 + r) - (1 - r^2)] \\ 1/2L_2[(2L_2 - 1)(1 + r) - (1 - r^2)] \\ 1/2L_3[(2L_3 - 1)(1 + r) - (1 - r^2)] \\ 2L_1L_2(1 - r) \\ 2L_2L_3(1 - r) \\ 2L_3L_1(1 - r) \\ L_1(1 - r^2) \\ L_2(1 - r^2) \\ L_3(1 - r^2) \\ 2L_1L_2(1 + r) \\ 2L_2L_3(1 + r) \\ 2L_3L_1(1 + r) \end{array} \right\}. \quad (4.11)$$

The functions in (4.10) and (4.11) use area coordinates, L_i , for describing the triangular cross-chapter and a normalized coordinate, r , for the axial coordinate. Note that $L_1 + L_2 + L_3 = 1$; the L_i vary from 0 to 1 and r varies from -1 to $+1$. Only the isoparametric forms of this element are employed in COYOTE.

4.5 Tetrahedral Element (3D)

The three-dimensional tetrahedron used in COYOTE may be either a four-node or ten-node isoparametric element as shown in Figure 4.5. The linear element is defined by the functions

$$\Theta_1 = \left\{ \begin{array}{l} L_1 \\ L_2 \\ L_3 \\ L_4 \end{array} \right\} \quad (4.12)$$

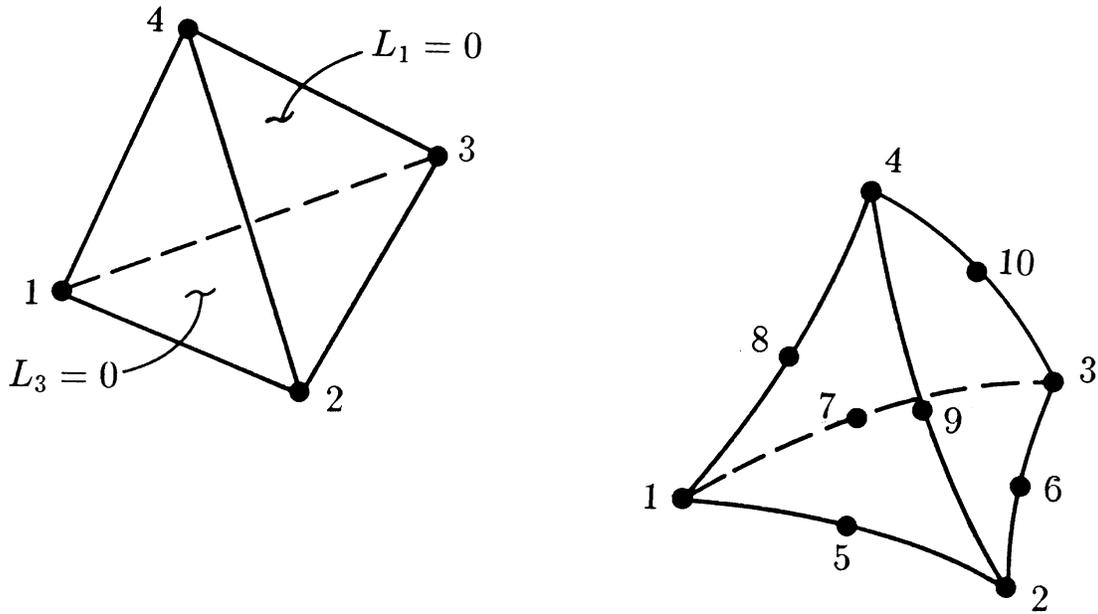


Figure 4.5: Three-dimensional tetrahedral elements.

while the quadratic element has shape functions of the form

$$\Theta_{\mathbf{q}} = \left\{ \begin{array}{l} L_1(2L_1 - 1) \\ L_2(2L_2 - 1) \\ L_3(2L_3 - 1) \\ L_4(2L_4 - 1) \\ 4L_1L_2 \\ 4L_2L_3 \\ 4L_3L_1 \\ 4L_1L_4 \\ 4L_2L_4 \\ 4L_3L_4 \end{array} \right\}. \quad (4.13)$$

The functions in (4.12) and (4.13) are ordered as shown in the figure and are written in terms of the volume coordinates [14] for the element, where $L_1 + L_2 + L_3 + L_4 = 1$. Again, only the isoparametric forms of this element are considered.

4.6 Bar Element (3D and 2D)

The three-dimensional bar elements available in COYOTE may be either a two-node or three-node, isoparametric element as shown in Figure 4.6. This element has a variable cross-chapteral area with conduction only allowed along the axis of the element. Note that the shape of the cross-chapter need not be explicitly defined here, though for purposes of boundary condition application, a circular cross-chapter is assumed. The shape function for the two-node element is defined by

$$\Theta_1 = \left\{ \begin{array}{l} 1/2(1-s) \\ 1/2(1+s) \end{array} \right\}. \quad (4.14)$$

and the three-node element is described by

$$\Theta_q = \left\{ \begin{array}{l} 1/2(s-1)s \\ 1/2(s+1)s \\ (1-s^2) \end{array} \right\}. \quad (4.15)$$

The functions in (4.14) and (4.15) are ordered as shown in the figure and are written in terms of the normalized coordinate s that varies from -1 to $+1$. The parametric mapping given in (4.9) relates the global coordinates x, y, z for the element to the local coordinate, s ; the mapping function Υ is defined by (4.14) and (4.15) for the two- and three-node elements, respectively.

The bar elements for two-dimensional, planar problems are also defined by the shape functions in (4.14) and (4.15). In this case, the isoparametric mapping is carried out from the x, y coordinates to the local coordinate s . The variable, cross-chapteral area for the two-dimensional case reduces to a variable thickness with unit depth. The axisymmetric, two-dimensional bar is treated in a similar manner, though it is rotated through an angle of 2π about the z axis. In both two-dimensional cases the bar element should be thought of as a one-dimensional conduction element in the plane of the problem. These elements are also equivalent to the two-dimensional version of a shell element.

4.7 Shell Element (3D)

The three-dimensional shell elements defined in COYOTE are specialized elements that allow conduction in the plane of the element but no transport through the thickness. Shells with both triangular and quadrilateral planforms are available; all elements allow variations in the shell thickness. These elements are shown in Figure 4.7. The temperature shape function for the three-node, triangular element is defined by

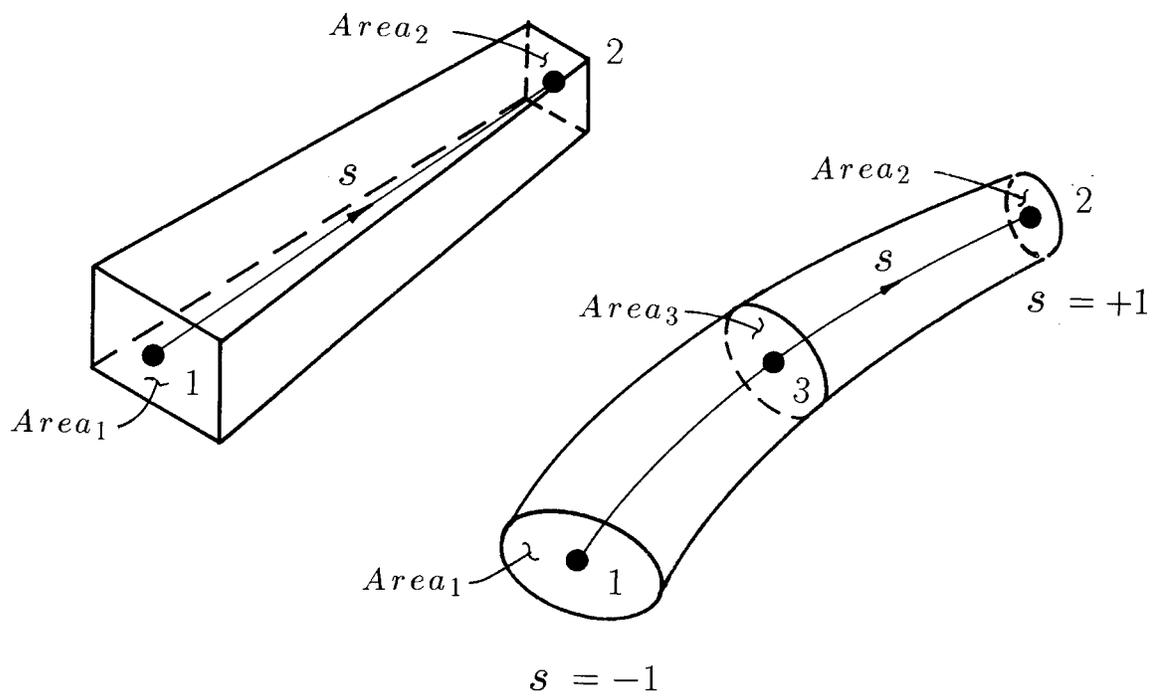


Figure 4.6: Three-dimensional bar elements.

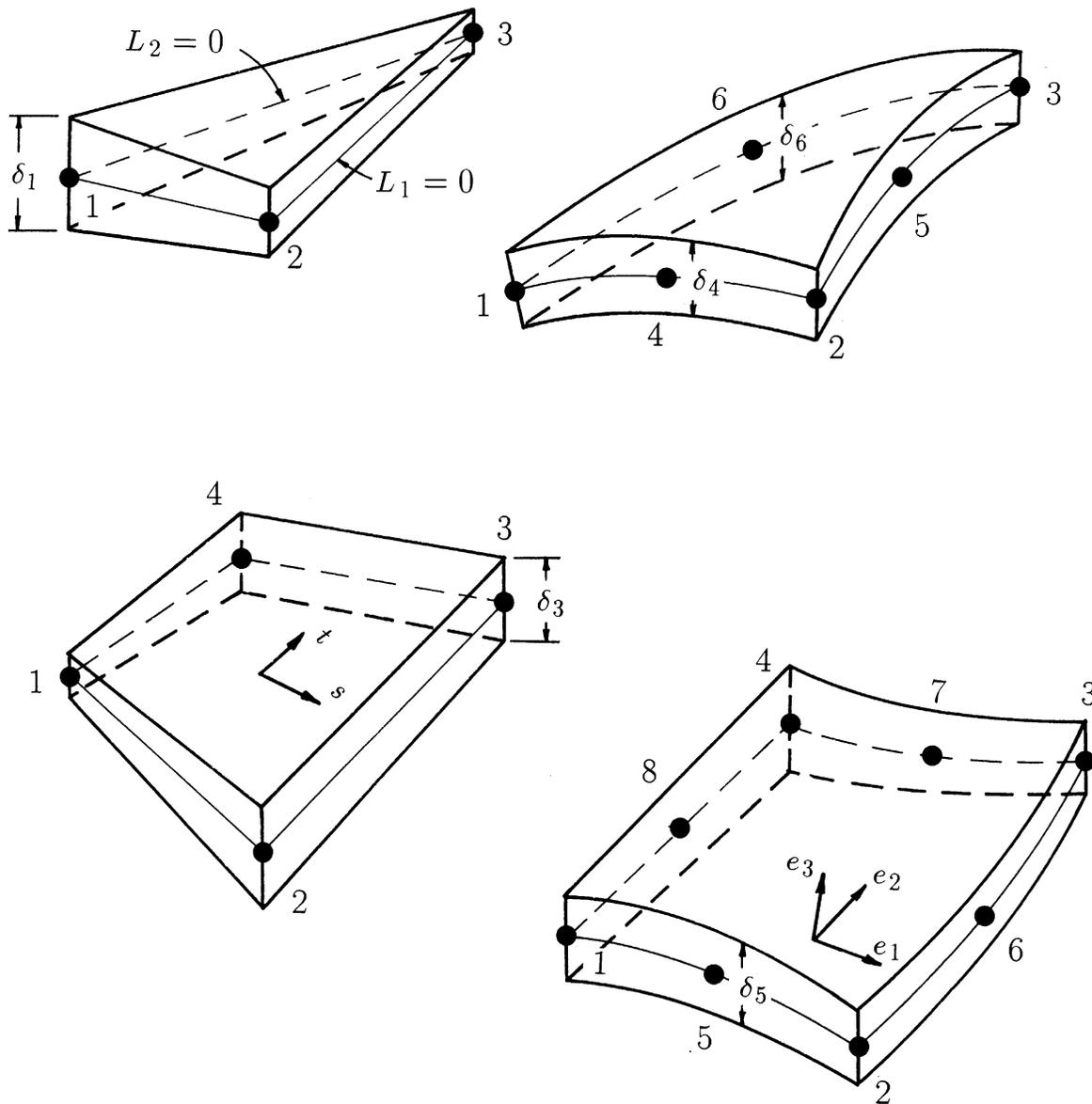


Figure 4.7: Three-dimensional shell elements.

$$\Theta_1 = \begin{Bmatrix} L_1 \\ L_2 \\ L_3 \end{Bmatrix} \quad (4.16)$$

and the six-node, triangular shell has the following shape functions

$$\Theta_q = \begin{Bmatrix} L_1(2L_1 - 1) \\ L_2(2L_2 - 1) \\ L_3(2L_3 - 1) \\ 4L_1L_2 \\ 4L_2L_3 \\ 4L_3L_1 \end{Bmatrix} \quad (4.17)$$

where the L_i are the standard, in-plane area coordinates that vary from 0 to +1. The four-node, quadrilateral shell has shape functions of the form

$$\Theta_1 = \begin{Bmatrix} 1/4(1-s)(1-t) \\ 1/4(1+s)(1-t) \\ 1/4(1+s)(1+t) \\ 1/4(1-s)(1+t) \end{Bmatrix} \quad (4.18)$$

while the eight-node, “serendipity” shell is defined by

$$\Theta_q = \begin{Bmatrix} 1/4(1-s)(1-t)(-s-t-1) \\ 1/4(1+s)(1-t)(s-t-1) \\ 1/4(1+s)(1+t)(s+t-1) \\ 1/4(1-s)(1+t)(-s+t-1) \\ 1/2(1-s^2)(1-t) \\ 1/2(1+s)(1-t^2) \\ 1/2(1-s^2)(1+t) \\ 1/2(1-s)(1-t^2) \end{Bmatrix} \quad (4.19)$$

and the normalized s, t coordinates vary from -1 to $+1$. The shape functions defined in (4.16)-(4.19) are recognized as being identical to the interpolation functions for the two-dimensional triangular and quadrilateral elements from Sections 4.1 and 4.2. Though the interpolation of temperature within the plane of the elements is similar, the geometrical representation of the planar elements and the shell elements are quite different. The parametric mapping for any of the shell elements is accomplished with the following definitions

$$\begin{aligned} x &= \Upsilon^T \mathbf{x} + r \Upsilon^T \frac{\delta}{2} \mathbf{e}_3 \cdot \mathbf{e}_x \\ y &= \Upsilon^T \mathbf{y} + r \Upsilon^T \frac{\delta}{2} \mathbf{e}_3 \cdot \mathbf{e}_y \end{aligned} \quad (4.20)$$

$$z = \mathbf{\Upsilon}^T \mathbf{z} + r \mathbf{\Upsilon}^T \frac{\delta}{2} \mathbf{e}_3 \cdot \mathbf{e}_z$$

where $\mathbf{\Upsilon}$ is the appropriate linear or quadratic interpolation within the plane (*e.g.*, equations (4.16)-(4.19)), $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are vectors of coordinates for the midplane nodes of the element, r is the normalized coordinate along the normal to the element midplane and δ is a vector of thickness values at the nodes. The vectors $\mathbf{e}_1, \mathbf{e}_2$ are defined as being tangent to the curvilinear coordinates s, t on the element midplane; \mathbf{e}_3 is normal to the element midplane and is defined by $\mathbf{e}_3 = \mathbf{e}_2 \times \mathbf{e}_1$. The unit vectors $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ define the orientation of the global coordinate system. Note that, in general, \mathbf{e}_3 varies over the planform of the element ($\mathbf{e}_3(s, t)$ or $\mathbf{e}_3(L_1, L_2)$) and this variation must be accounted for in the construction of the Jacobian entries for the element mapping procedure. All of these vectors are more completely defined in a subsequent chapter.

4.8 Spatial Derivatives and Integrals

The construction of the various finite element coefficient matrices in (3.13) and (3.19) requires the integration of combinations of the interpolation functions and their spatial derivatives over the volume (area) of the element. The integration process is most easily carried out in the normalized or natural coordinate system for each element since the limits of integration are simple and independent of the global coordinates. The shape functions presented in the previous chapters were expressed in the natural coordinate system for each element. There remains the task of expressing spatial derivatives of the shape functions in terms of the same normalized coordinates. The following relations, based on the chain rule and the parametric mapping ideas, are needed

$$\begin{pmatrix} \frac{\partial \Lambda}{\partial s} \\ \frac{\partial \Lambda}{\partial t} \\ \frac{\partial \Lambda}{\partial r} \end{pmatrix} = \begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} & \frac{\partial z}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t} \\ \frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} & \frac{\partial z}{\partial r} \end{bmatrix} \begin{pmatrix} \frac{\partial \Lambda}{\partial x} \\ \frac{\partial \Lambda}{\partial y} \\ \frac{\partial \Lambda}{\partial z} \end{pmatrix} = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} \begin{pmatrix} \frac{\partial \Lambda}{\partial x} \\ \frac{\partial \Lambda}{\partial y} \\ \frac{\partial \Lambda}{\partial z} \end{pmatrix} = \mathbf{J} \begin{pmatrix} \frac{\partial \Lambda}{\partial x} \\ \frac{\partial \Lambda}{\partial y} \\ \frac{\partial \Lambda}{\partial z} \end{pmatrix} \quad (4.21)$$

where Λ represents any of the element interpolation functions (*e.g.*, Θ_1, Θ_q) and \mathbf{J} is the Jacobian of the transformation from global coordinates x, y, z to the local element coordinates s, t, r . The parametric mapping scheme defined in the previous chapters (*e.g.*, equations (4.3), (4.9) or (4.20)) can be used to define the components of \mathbf{J} . That is,

$$\begin{aligned} J_{11} &= \frac{\partial \mathbf{\Upsilon}^T}{\partial s} \mathbf{x} & ; & & J_{12} &= \frac{\partial \mathbf{\Upsilon}^T}{\partial s} \mathbf{y} & ; & & J_{13} &= \frac{\partial \mathbf{\Upsilon}^T}{\partial s} \mathbf{z} \\ J_{21} &= \frac{\partial \mathbf{\Upsilon}^T}{\partial t} \mathbf{x} & ; & & J_{22} &= \frac{\partial \mathbf{\Upsilon}^T}{\partial t} \mathbf{y} & ; & & J_{23} &= \frac{\partial \mathbf{\Upsilon}^T}{\partial t} \mathbf{z} \end{aligned} \quad (4.22)$$

$$J_{31} = \frac{\partial \boldsymbol{\Upsilon}^T}{\partial r} \mathbf{x} \quad ; \quad J_{32} = \frac{\partial \boldsymbol{\Upsilon}^T}{\partial r} \mathbf{y} \quad ; \quad J_{33} = \frac{\partial \boldsymbol{\Upsilon}^T}{\partial r} \mathbf{z}$$

Inverting the transformation matrix in (4.21) provides the required definition of the spatial derivatives of the shape functions in terms of the local element coordinates

$$\begin{Bmatrix} \frac{\partial \Lambda}{\partial x} \\ \frac{\partial \Lambda}{\partial y} \\ \frac{\partial \Lambda}{\partial z} \end{Bmatrix} = \mathbf{J}^{-1} \begin{Bmatrix} \frac{\partial \Lambda}{\partial s} \\ \frac{\partial \Lambda}{\partial t} \\ \frac{\partial \Lambda}{\partial r} \end{Bmatrix} = \frac{1}{|\mathbf{J}|} \begin{bmatrix} \mathcal{J}_{11} & \mathcal{J}_{12} & \mathcal{J}_{13} \\ \mathcal{J}_{21} & \mathcal{J}_{22} & \mathcal{J}_{23} \\ \mathcal{J}_{31} & \mathcal{J}_{32} & \mathcal{J}_{33} \end{bmatrix} \begin{Bmatrix} \frac{\partial \Lambda}{\partial s} \\ \frac{\partial \Lambda}{\partial t} \\ \frac{\partial \Lambda}{\partial r} \end{Bmatrix} \quad (4.23)$$

where $|\mathbf{J}|$ indicates the determinant of the Jacobian matrix \mathbf{J} . The components \mathcal{J}_{ij} are complex functions of the components of \mathbf{J} that can be obtained by inverting the 3×3 Jacobian matrix. In practice, the Jacobian is usually inverted numerically. For the two-dimensional case the above equations are simplified substantially and permit analytic manipulation of the resulting 2×2 matrix. The Jacobian for the transformation of the bar element is given by

$$\frac{\partial \Lambda}{\partial r} = \mathbf{J}^{-1} \frac{\partial \Lambda}{\partial s} = \frac{1}{\Delta} \frac{\partial \Lambda}{\partial s} \quad (4.24)$$

where

$$\Delta = \left[\left(\frac{\partial \hat{\boldsymbol{\Upsilon}}^T}{\partial s} \mathbf{x} \right)^2 + \left(\frac{\partial \hat{\boldsymbol{\Upsilon}}^T}{\partial s} \mathbf{y} \right)^2 + \left(\frac{\partial \hat{\boldsymbol{\Upsilon}}^T}{\partial s} \mathbf{z} \right)^2 \right]^{\frac{1}{2}} \quad (4.25)$$

and the coordinate r is the physical coordinate along the axis of the bar. The mapping for the shell elements follows the definitions in (4.21)-(4.23) though some terms in the Jacobian are slightly more complex due to the presence of a normal vector in the basis function definition given by equation (4.20).

In performing integrations over the element volume it is also necessary to transform the integration variables and limits from the global coordinates to the local element coordinates. The differential elemental volume transforms according to

$$d\Omega = dx dy dz = |\mathbf{J}| ds dt dr \quad (4.26)$$

while for two-dimensional geometries the planar area is transformed by

$$d\Omega = dx dy = |\mathbf{J}| ds dt \quad (4.27)$$

and

$$d\Omega = r d\Theta dr dz = 2\pi r |\mathbf{J}| ds dt \quad (4.28)$$

for axisymmetric geometries, where the circumferential dependence has been explicitly evaluated to produce the 2π factor. In the axisymmetric case the radius r would be

interpolated by $r = \mathbf{\Upsilon}^T \mathbf{r}$. The integration limits for the integrals transform to the limits on the local coordinates s, t, r , *i.e.*, -1 to $+1$.

In the above equations the s, t, r variables for a brick element have been used for the purpose of explanation. Similar relations for a tetrahedral element can be derived by replacing s, t, r with L_1, L_2 and L_3 . The L_4 variable does not enter the formulas due to the relation $L_1 + L_2 + L_3 + L_4 = 1$. Hybrid coordinates, such as those used in the prism element, are treated in an analogous manner. The two-dimensional case also follows the above procedures.

For the special case that involves the bar elements, the differential volume is rewritten to allow for the explicit specification of cross-chapteral areas and thicknesses. Thus for a bar the volume transforms according to

$$d\Omega = |\mathbf{J}| ds = A(s)\Delta ds \quad (4.29)$$

where again

$$\Delta = \left[\left(\frac{\partial \hat{\mathbf{Y}}^T}{\partial s} \mathbf{x} \right)^2 + \left(\frac{\partial \hat{\mathbf{Y}}^T}{\partial s} \mathbf{y} \right)^2 + \left(\frac{\partial \hat{\mathbf{Y}}^T}{\partial s} \mathbf{z} \right)^2 \right]^{\frac{1}{2}}$$

and $A(s)$ is the cross-chapteral area of the bar as a function of normalized distance along the bar. For two-dimensional bars, $A(s) = \delta(s) \cdot 1$ where $\delta(s)$ is the bar thickness. In the case of a three-dimensional shell, the differential volume is expressed as shown in (4.26); the possibility of a variable thickness in the shell requires a full mapping for the shell volume.

4.9 Matrix Evaluation

With the previous definitions it is now possible to derive a computational form for the matrix coefficients involved in the finite element equations of Section 3. For purposes of discussion, only a representative term from the matrix system will be considered in detail; the evaluation of the remaining terms follows in a similar manner.

Consider a cross derivative component of the diffusion matrix given by equation (3.13) as

$$\mathbf{K}_{12} = \mathbf{K}_{\mathbf{xy}} = \int_{\Omega_e} k_{xy} \frac{\partial \Theta}{\partial x} \frac{\partial \Theta}{\partial y} d\Omega \quad (4.30)$$

which will be evaluated for a three-dimensional, twenty-node, brick element. From the previous definitions in (4.23) and (4.26), equation (4.30) can be written as

$$\begin{aligned}
\mathbf{K}_{\mathbf{xy}} = & \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} k_{xy} \frac{1}{|\mathbf{J}|} \underbrace{\left[\mathcal{J}_{11} \frac{\partial \Theta_{\mathbf{q}}}{\partial s} + \mathcal{J}_{12} \frac{\partial \Theta_{\mathbf{q}}}{\partial t} + \mathcal{J}_{13} \frac{\partial \Theta_{\mathbf{q}}}{\partial r} \right]}_{\frac{\partial \Theta_{\mathbf{q}}}{\partial x}} \\
& \cdot \underbrace{\left[\mathcal{J}_{21} \frac{\partial \Theta_{\mathbf{q}}^T}{\partial s} + \mathcal{J}_{22} \frac{\partial \Theta_{\mathbf{q}}^T}{\partial t} + \mathcal{J}_{23} \frac{\partial \Theta_{\mathbf{q}}^T}{\partial r} \right]}_{\frac{\partial \Theta_{\mathbf{q}}^T}{\partial y}} \frac{1}{|\mathbf{J}|} ds dt dr \quad (4.31)
\end{aligned}$$

where the $\Theta_{\mathbf{q}}$ functions are given in (4.8). For an isoparametric (curve-sided) element the components of \mathcal{J}_{ij} would be evaluated using $\mathbf{\Upsilon} = \Theta_{\mathbf{q}}$ from equation (4.8).

The above integral is of the general form

$$I = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} f(s, t, r) ds dt dr \quad (4.32)$$

where $f(s, t, r)$ is a rational function of the normalized coordinates. All of the element matrices are of this form and can be conveniently evaluated using a numerical quadrature procedure. That is, the integral in (4.32) can be evaluated by the formula

$$I = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n W_i W_j W_k f(s_i, t_j, r_k) \quad (4.33)$$

where W_i are weighting coefficients, s_i, t_j, r_k are quadrature points in the integration interval and n is the number of quadrature points in the formula. For linear and quadratic brick elements, COYOTE generally employs a product Gauss quadrature rule as shown in (4.33) with $n = 2$ and $n = 3$, respectively; the two-dimensional quadrilaterals employ a similar scheme with a double sum used in (4.33) and $n = 2$ for a linear element and $n = 3$ for the quadratic elements. Other elements in the library are also evaluated using quadrature formulas, though the forms of (4.32) and (4.33) are slightly different in these cases. For elements using volume or area coordinates, the limits on the definite integral in (4.32) run from 0 to 1. Also, these elements typically do not use a product rule but rather a single summation over the total number of quadrature points. In COYOTE, the tetrahedral elements are evaluated with a four point or a five point formula and the triangular elements with a three point or a seven point rule. The prism uses a three point or a seven point integration rule in the triangular plane and a 2 or 3 point Gauss formula along the normalized axis. Integration rules for the bar and shell elements follow these same procedures. Nonproduct Gauss rules [16] are also available for use with the hexahedral elements and may offer some economic benefits over the product Gauss rules.

Application of the quadrature formula in (4.33) to the integral in equation (4.31) produces the element coefficient matrix $\mathbf{K}_{\mathbf{xy}}$. Note that variable coefficients, such as the

thermal conductivity, must be evaluated at the integration points if they vary over the element. Constant coefficients are of course removed from the integral and play no role in the quadrature procedure.

4.10 Element Boundary Conditions and Source Terms

In this chapter the construction of boundary conditions and volumetric source terms for the element matrix equations is considered. Though the required flux vectors are numerically evaluated in the same manner as the coefficient matrices, a number of additional assumptions and details are necessary that require further comment.

4.10.1 Volumetric Sources

The flux vectors for the conduction equation consist of two parts: a part due to volumetric sources and a part due to surface fluxes. Consider first the volumetric term,

$$\mathbf{F}_Q = \int_{\Omega_e} \Theta Q d\Omega. \quad (4.34)$$

The source term is allowed to vary over the element in an arbitrary manner, which is indicated by $Q(s, t, r)$. As given previously in equations (4.26)-(4.28), the elemental volume can also be written in terms of the normalized coordinates. Thus, in a computational form (4.34) becomes

$$\mathbf{F}_Q = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} \Theta Q(s, t, r) |\mathbf{J}| ds dt dr \quad (4.35)$$

for a three-dimensional brick element; similar forms are derivable for the other elements in two and three dimensions. The integral in (4.35) can be evaluated with a standard numerical quadrature rule to produce the force vector \mathbf{F}_Q . To accomplish the numerical integration, the volume source must be evaluated at the quadrature points. If the volume source depends on other variables, such as the temperature, temperature rate, spatial location, *etc.*, these quantities can be provided at the quadrature points through use of the element basis functions.

4.10.2 Surface Fluxes

The remaining flux vectors in the conduction equation arise from surface fluxes distributed along element boundaries. These terms need only be considered for those el-

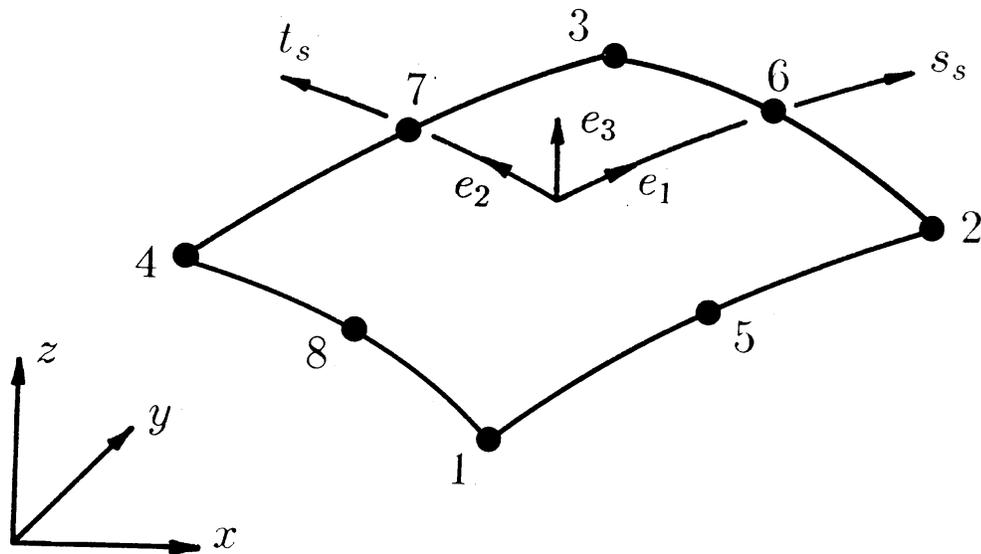


Figure 4.8: Nomenclature for element surface computations.

element sides coinciding with the “exterior” boundaries of the problem domain; contributions from interior element boundaries are generally cancelled by adjoining elements. The surface flux vector is given by

$$\mathbf{F} = \int_{\Gamma_e} \Theta q_i n_i d\Gamma = \int_{\Gamma_e} \Theta (f^q - q_c - q_r) d\Gamma \quad (4.36)$$

where Γ_e is the surface of the element, $q_i n_i$ is the heat flux normal to the surface and the remaining terms are defined in (2.4)-(??).

The computation of the indicated surface integrals are most easily carried out in the normalized or natural coordinate system for the face (edge) of an element. This requires that the elemental surface area (edge length) $d\Gamma$, be related to the local surface coordinates. Consider the typical quadrilateral element face shown in Figure 4.8 where the vectors \mathbf{e}_1 and \mathbf{e}_2

are defined as being tangent to the curvilinear coordinates, s_s and t_s . The \mathbf{e} vectors are not necessarily unit vectors; the s_s and t_s coordinates are assumed to be natural coordinates for the element face. The elemental area $d\Gamma$ in terms of the global coordinates x, y, z is related to an elemental area in surface coordinates by

$$d\Gamma = |\mathbf{J}_s| ds_s dt_s \quad (4.37)$$

where \mathbf{J}_s is the Jacobian of the coordinate transformation and $|\cdot|$ indicates the determinant. The determinant of the Jacobian can be written in terms of the \mathbf{e} vectors as

$$|\mathbf{J}_s| = |\mathbf{e}_1 \times \mathbf{e}_2| = [(\mathbf{e}_1 \cdot \mathbf{e}_1)(\mathbf{e}_2 \cdot \mathbf{e}_2) - (\mathbf{e}_1 \cdot \mathbf{e}_2)^2]^{1/2}. \quad (4.38)$$

The \mathbf{e} vectors can be expressed in terms of the global coordinates by

$$\mathbf{e}_1 = \begin{Bmatrix} \frac{\partial x}{\partial s_s} \\ \frac{\partial y}{\partial s_s} \\ \frac{\partial z}{\partial s_s} \end{Bmatrix} \quad ; \quad \mathbf{e}_2 = \begin{Bmatrix} \frac{\partial x}{\partial t_s} \\ \frac{\partial y}{\partial t_s} \\ \frac{\partial z}{\partial t_s} \end{Bmatrix} \quad (4.39)$$

Using the parametric mapping concept allows

$$x = \hat{\mathbf{Y}}^T \mathbf{x} \quad ; \quad y = \hat{\mathbf{Y}}^T \mathbf{y} \quad ; \quad z = \hat{\mathbf{Y}}^T \mathbf{z} \quad (4.40)$$

where the $\hat{\cdot}$ notation indicates the restriction of the interpolation function to an element face (edge). The functions $\hat{\mathbf{Y}}$ may be either linear or quadratic depending on the type of mapping used to describe the element geometry. Using (4.40) in (4.39) then

$$\mathbf{e}_1 = \begin{Bmatrix} \frac{\partial \hat{\mathbf{Y}}^T \mathbf{x}}{\partial s_s} \\ \frac{\partial \hat{\mathbf{Y}}^T \mathbf{y}}{\partial s_s} \\ \frac{\partial \hat{\mathbf{Y}}^T \mathbf{z}}{\partial s_s} \end{Bmatrix} \quad ; \quad \mathbf{e}_2 = \begin{Bmatrix} \frac{\partial \hat{\mathbf{Y}}^T \mathbf{x}}{\partial t_s} \\ \frac{\partial \hat{\mathbf{Y}}^T \mathbf{y}}{\partial t_s} \\ \frac{\partial \hat{\mathbf{Y}}^T \mathbf{z}}{\partial t_s} \end{Bmatrix} \quad (4.41)$$

Equations (4.38) and (4.41) provide a means for computing $|\mathbf{J}_s|$, thus allowing the transformation in (4.37) to be employed. Note that in two dimensions the above relations simplify and the elemental length of an element edge is given by

$$d\Gamma = \left[\left(\frac{\partial \hat{\mathbf{Y}}^T \mathbf{x}}{\partial s} \right)^2 + \left(\frac{\partial \hat{\mathbf{Y}}^T \mathbf{y}}{\partial s} \right)^2 \right]^{\frac{1}{2}} ds = \Delta ds \quad (4.42)$$

where s is the coordinate along the edge of an element. Axisymmetric geometries require integration over an element edge that is rotated through an angle of 2π and (4.42) should therefore be expressed as

$$d\Gamma = \left[\left(\frac{\partial \hat{\mathbf{Y}}^T \mathbf{r}}{\partial s} \right)^2 + \left(\frac{\partial \hat{\mathbf{Y}}^T \mathbf{z}}{\partial s} \right)^2 \right]^{\frac{1}{2}} r d\Theta ds = 2\pi \Delta r ds \quad (4.43)$$

where r is the radius for the element edge and would be interpolated by $r = \mathbf{Y}^T \mathbf{r}$. For three-dimensional edges or surfaces on shell or bar elements, the Jacobian in (4.42) is modified to include the z coordinate.

To complete the specification of the integrand in (4.38) the variation of $q_i n_i$ with s_s and t_s is required. From the boundary condition definitions in equations (2.4)-(??) the normal heat flux consists of three components

$$q_i n_i = f^q - q_c - q_r = f^q - h_c(T - T_c) - h_r(T - T_r) \quad (4.44)$$

For calculation of the boundary fluxes it is assumed that the applied flux, f^q , convective and radiative coefficients, h_c , h_r and reference temperatures T_c , T_r are known functions of the surface (edge) coordinates, s_s , t_s . Then using the standard surface (edge) interpolation for the temperature, the heat flux vector can be written for the three-dimensional case as

$$\begin{aligned} \mathbf{F}(\mathbf{T}) &= \int_{-1}^{+1} \int_{-1}^{+1} \hat{\Theta} f^q(s_s, t_s) |\mathbf{J}_s| ds_s dt_s \\ &- \int_{-1}^{+1} \int_{-1}^{+1} \hat{\Theta} h_c(s_s, t_s) \hat{\Theta}^T |\mathbf{J}_s| ds_s dt_s (\mathbf{T} - \mathbf{T}_c(s_s, t_s)) \\ &- \int_{-1}^{+1} \int_{-1}^{+1} \hat{\Theta} h_r(s_s, t_s) \hat{\Theta}^T |\mathbf{J}_s| ds_s dt_s (\mathbf{T} - \mathbf{T}_r(s_s, t_s)) \end{aligned} \quad (4.45)$$

or

$$\mathbf{F}(\mathbf{T}) = \mathbf{F}_q - \mathbf{H}_c \mathbf{T} + \mathbf{H}_c \mathbf{T}_c - \mathbf{H}_r \mathbf{T} + \mathbf{H}_r \mathbf{T}_r. \quad (4.46)$$

The integrals in (4.45) are evaluated using a numerical quadrature procedure over the element surface; in two dimensions only integrals along an element edge need to be considered. Variable coefficients, such as f^q , h_c , h_r , *etc.*, must be evaluated at the quadrature points and may depend on interpolated variables such as temperature, spatial location, *etc.* Note that some of the terms in \mathbf{F} contain unknown element temperatures ($\mathbf{H}_c \mathbf{T}$ and $\mathbf{H}_r \mathbf{T}$); for solution purposes these terms are moved from the flux vector to the left-hand-side of the matrix equation in (3.10) and added to the diffusion matrix \mathbf{K} .

4.10.3 Internal Surface Fluxes

The flux vectors considered in the previous chapter were derived from boundary conditions that are applied to the external boundaries of the heat conduction problem. As shown by equation (2.9), it is sometimes appropriate to consider “internal” flux conditions associated with surface contact at a material interface. The computational form for this internal boundary condition is derived in the same manner as presented above. Also, these terms need only be constructed for element surfaces that are part of the contact surface.

The internal or gap surface flux vector for the master surface is given by

$$\mathbf{F}_g(\mathbf{T}) = - \int_{\Gamma_m} \Theta q_g d\Gamma = - \int_{\Gamma_m} \Theta h_g (T_m - T_s) d\Gamma \quad (4.47)$$

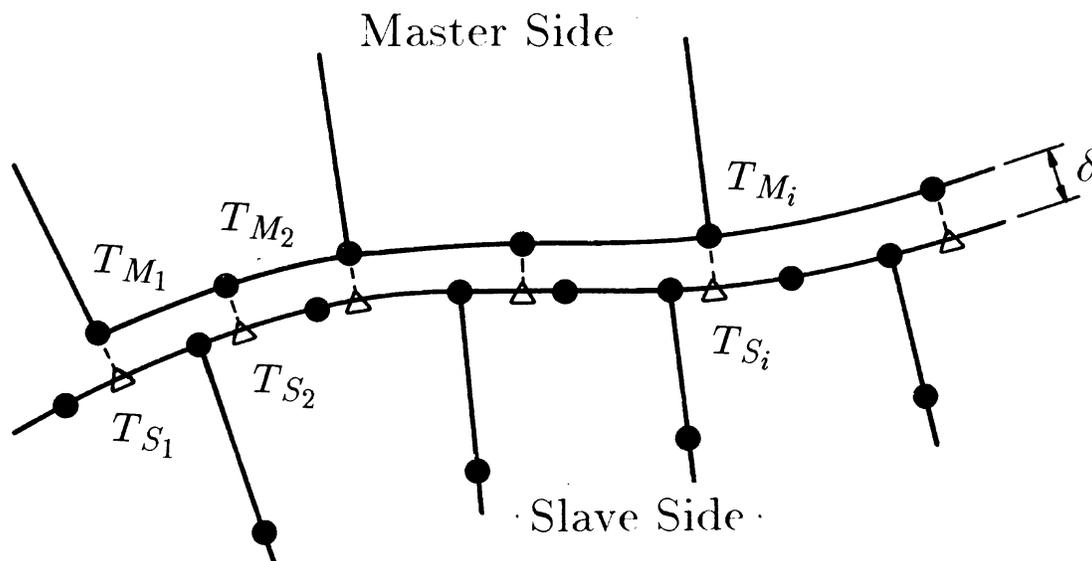


Figure 4.9: Nomenclature for contact resistance formulation.

where Γ_m is the contact area for the master surface, h_g is an effective heat transfer coefficient and T_m and T_s are temperatures on each side of the contact surface. The numerical implementation of this condition requires that “master” and “slave” sides of the contact surface be defined. Also, since unknown temperatures occur on both sides of the gap, each contact surface must be processed in turn as a “master” surface; the opposite or “slave” surface provides an estimate of the reference temperature for heat transfer across the gap. For generality, the situation shown in the two-dimensional sketch of Figure 4.9 is considered, where the nodes and elements on each side of the contact surface are not aligned. If a node on the master surface does not have an image on the

slave surface, then h_g is set to zero for that location and the contact heat flux for that node is not evaluated.

In a computational form the flux vector for the gap can be written as

$$\mathbf{F}_g(\mathbf{T}) = - \int_{-1}^{+1} \int_{-1}^{+1} \hat{\mathbf{\Theta}} h_g(s_s, t_s) \hat{\mathbf{\Theta}}^T |\mathbf{J}_s| ds_s dt_s (\mathbf{T}_m - \mathbf{T}_s) \quad (4.48)$$

or in matrix form

$$\mathbf{F}_g(\mathbf{T}) = -\mathbf{H}_g \mathbf{T}_m + \mathbf{H}_g \mathbf{T}_s \quad (4.49)$$

In developing (4.49) the contact coefficient h_g was assumed to vary in a known manner over the contact surface. The vector \mathbf{T}_m corresponds to unknown nodal point temper-

atures on the master surface; the term $\mathbf{H}_g \mathbf{T}_m = \mathbf{H}_g \mathbf{T}$ is combined with the diffusion matrix \mathbf{K} during the solution process. The temperatures in the vector \mathbf{T}_s are not generally nodal point temperatures but rather interpolated temperatures on the slave surface, adjacent to the master surface nodes. The temperature vector \mathbf{T}_s is obtained by (basis function) interpolation on the slave surface; it is assumed that the slave surface temperatures can be interpolated by the same functions as used to describe temperature variations on the master surface. Since the temperature fields along the contact surface are generally unknown, this type of interface condition must be solved via an iterative process.

The above formulation was carried out for the general case where the master and slave nodes were not aligned. This does not preclude the use of a standard mesh where there is a direct, one-to-one correspondence between nodes along the contact surface. In addition to providing a generalized contact resistance model, the above formulation provides a simple method for connecting regions with different mesh spacings. For “large” values of h_g , equation (4.47) forces the temperature distributions on each side of the contact surface to be essentially equal. This is the thermal equivalent of the slide line algorithms used in solid mechanics [17]. Section 5.8 outlines the algorithm used to determine the occurrence of contact and the spatial location of master nodes on the slave surface.

4.10.4 Specified Temperature Boundary Conditions

In addition to the (“natural”) boundary conditions specified by the boundary integrals presented above, “essential” boundary conditions specifying particular values of the temperature must also be considered. Application of a specified temperature boundary condition results in the field equation for that particular nodal point temperature being replaced by a constraint equation that enforces the proper boundary value. COYOTE uses a penalty method [18] to implement this type of constraint condition.

4.11 Matrix Equation

As a result of the manipulations outlined in the previous chapters, the element matrices and boundary conditions can be constructed for a variety of element types. An assembly process, such as indicated by equation (3.12) leads to the general matrix equation as shown in (3.10) or (3.13). The equation in (3.10) can be made more specific by considering the individual terms from various types of boundary conditions. Using (4.46) and (4.49), then (3.10) becomes

$$\mathbf{M}(\mathbf{T})\dot{\mathbf{T}} + \mathbf{K}(\mathbf{T})\mathbf{T} = \mathbf{F}_Q(\mathbf{T}) + \mathbf{F}_q - \mathbf{H}_c \mathbf{T} + \mathbf{H}_c \mathbf{T}_c$$

$$- \mathbf{H}_r \mathbf{T} + \mathbf{H}_r \mathbf{T}_r - \mathbf{H}_g \mathbf{T} + \mathbf{H}_g \mathbf{T}_s \quad (4.50)$$

Rearranging (4.50) allows the final form of the discrete system to be written as

$$\overline{\mathbf{M}}(\mathbf{T}) \dot{\mathbf{T}} + \overline{\mathbf{K}}(\mathbf{T}) \mathbf{T} = \overline{\mathbf{F}}(\mathbf{T}) \quad (4.51)$$

with

$$\overline{\mathbf{M}} = \mathbf{M}$$

$$\overline{\mathbf{K}} = \mathbf{K} + \mathbf{H}_c + \mathbf{H}_r + \mathbf{H}_g$$

$$\overline{\mathbf{F}} = \mathbf{F}_Q + \mathbf{F}_q + \mathbf{F}_c + \mathbf{F}_r + \mathbf{F}_g$$

and

$$\mathbf{F}_c = \mathbf{H}_c \mathbf{T}_c$$

$$\mathbf{F}_r = \mathbf{H}_r \mathbf{T}_r$$

$$\mathbf{F}_g = \mathbf{H}_g \mathbf{T}_s$$

The equation in (4.51) represents the finite element analogue of the heat conduction problem that must be solved for the domain of interest. When convective terms are required, the $\overline{\mathbf{K}}$ matrix in (4.51) is modified to include the unsymmetric, velocity dependent advection term.

Chapter 5

Solution Procedures

The major computational effort in any finite element procedure occurs in the solution of the assembled matrix equations that describe the discretized problem. This is especially true in the case of highly nonlinear equations or problems with coupled physical phenomena, both of which can be found in the present case. In addition to computational efficiency these characteristics also introduce questions regarding the ability to achieve a solution, *i.e.*, convergence for a given set of data. The choice of a solution algorithm is therefore a critical element in the overall utility, robustness and efficiency of a computer code such as COYOTE.

As described previously, the basic matrix problem of concern can be written as

$$\overline{\mathbf{M}}(\mathbf{T})\dot{\mathbf{T}} + \overline{\mathbf{K}}(\mathbf{T})\mathbf{T} = \overline{\mathbf{F}}(\mathbf{T}). \quad (5.1)$$

where $\overline{\mathbf{M}}$ represents the capacitance matrix, $\overline{\mathbf{K}}$ contains the diffusion terms and $\overline{\mathbf{F}}$ provides the boundary and volumetric forcing functions. In the most general case, each term in (5.1) may depend explicitly on the temperature due to variable coefficients or thermal properties. A dependence on other variables, such as time or spatial location, is also possible, though this does not affect the nonlinearity of the system. In all cases the matrices are large, sparse, and symmetric in their structure; a proper ordering of the equations will produce a banded matrix system. The inclusion of the advective term alters these characteristics slightly by making the system unsymmetric; the nonlinearity and matrix structure remain the same.

In the following chapters the solution algorithms for treating the steady and time-dependent forms of (5.1) are outlined. The solution procedures are defined in general terms and are meant to be applied to all of the classes and subclasses of diffusion and advection-diffusion problems defined by equation (5.1). No further explicit consideration

of the advection-diffusion case will be given as it is a standard subset of the general problem. Most of the algorithms described lead to linear, algebraic equations that must be solved at each iteration or time step of the solution process. The matrix solution methods used in COYOTE are also discussed in this chapter.

5.1 Steady-State Algorithms

The time-independent form of (5.1) is

$$\bar{\mathbf{K}}(\mathbf{T})\mathbf{T} = \bar{\mathbf{F}}(\mathbf{T}) \quad (5.2)$$

which is recognized as a system of nonlinear, algebraic equations. Consider first the case where $\bar{\mathbf{K}}$ and $\bar{\mathbf{F}}$ are not functions of $\bar{\mathbf{T}}$. In this situation (5.2) reduces to a linear matrix equation which can be solved directly, without iteration. When (5.2) retains its nonlinear form, an iterative technique is required. COYOTE currently employs a single type of iterative method, though it may be combined with other techniques to expand the possibilities for achieving a steady state solution.

5.1.1 Successive Substitution Method

A particularly simple iterative method with a large radius of convergence is the successive substitution (Picard, functional iteration) method described by

$$\bar{\mathbf{K}}(\mathbf{T}^n)\mathbf{T}^{n+1} = \bar{\mathbf{F}}(\mathbf{T}^n) \quad (5.3)$$

where the superscript indicates the iteration level. For the mildly nonlinear behavior typically found in heat conduction problems, the rate of convergence of (5.3) is generally good, despite being a first-order method. An improvement in convergence rate can sometimes be realized by use of a relaxation formula where

$$\bar{\mathbf{K}}(\mathbf{T}^n)\mathbf{T}^* = \bar{\mathbf{F}}(\mathbf{T}^n) \quad (5.4)$$

and

$$\mathbf{T}^{n+1} = \alpha\mathbf{T}^n + (1 - \alpha)\mathbf{T}^* \quad 0 \leq \alpha < 1.$$

The above methods are adequate for the large majority of thermal diffusion problems since nonlinearities associated with material properties and boundary conditions are usually quite mild. In the few instances where nonlinear behavior causes the above schemes to converge slowly or diverge, alternate methods could be developed. One popular choice

is the use of second-order methods, such as Newton's method. In the present application Newton's method has not been implemented; the construction of a Jacobian matrix for nonanalytic specifications of material properties and boundary conditions is not believed to be cost effective. However, for more complex nonlinearities, such as enclosure radiation, Newton's method is viewed as an essential technique. Section 5.5 describes the use of Newton's method for the enclosure radiation problem.

5.1.2 Continuation Method

Failure to achieve a converged solution using (5.3) or (5.4) can often be ascribed to the use of a poor initial guess (\mathbf{T}^0) for the iterative algorithm. There are two general approaches to the problem of generating good initial estimates for a solution vector and both involve some type of "tracking" of the solution. The first procedure is simply the method of false transients in which the solution is followed through use of a time parameter. The transient algorithms described in a later chapter are candidates for this approach.

A second method consists of incrementally approaching the final solution through a series of intermediate solutions. These intermediate solutions may be of physical interest or may simply be a means to obtain the required solution. The formal algorithms used to implement this procedure are termed continuation methods and can be used with either of the iterative methods in (5.3) and (5.4).

Assume that the solution for (5.2) depends continuously on some real parameter, λ . For heat conduction problems, λ could be the magnitude of a volumetric source or the magnitude of a boundary condition. Then (5.2) can be written in general as

$$\bar{\mathbf{K}}(\mathbf{T}, \lambda)\mathbf{T} = \bar{\mathbf{F}}(\mathbf{T}, \lambda) \quad (5.5)$$

which suggests the zeroth order continuation method

$$\bar{\mathbf{K}}(\mathbf{T}_\lambda^n, \lambda^m)\mathbf{T}_\lambda^{n+1} = \bar{\mathbf{F}}(\mathbf{T}_\lambda^n, \lambda^m) \quad (5.6)$$

where (5.6) is solved for a series of problems with increasing values of the continuation parameter $\lambda^m = \lambda^{m-1} + \Delta\lambda$. The converged solution, T_λ , at one value of λ is used as the starting solution at the next higher value of λ ; the iterative method in (5.3) or (5.4) is used at each value of λ to achieve a converged solution. This technique is available in COYOTE and can be used effectively with some very nonlinear problems.

5.1.3 Convergence Criteria

The use of an iterative solution method necessitates the definition of a convergence and stopping criteria to terminate the iteration process. The usual measure of convergence

is a norm on the change in the solution vector between successive iterations. COYOTE employs the discrete RMS norm defined by

$$d_{n+1} = \left[\frac{1}{N_{nodes} \cdot T_{max}^2} \sum_{i=1}^{N_{nodes}} (T_{i(n+1)} - T_{i(n)})^2 \right]^{\frac{1}{2}} \quad (5.7)$$

In the definition in (5.7) N_{nodes} is the total number of nodal points and T_{max} is an appropriate temperature scale for the problem; T_{max} may be specified or computed from the temperature solution vector.

The criteria for terminating the iteration process is based on a user supplied tolerance. The iterative algorithm is terminated when the following inequality is satisfied

$$d_{n+1} \leq \epsilon^T \quad (5.8)$$

where ϵ^T is set by the user and has a typical value of 0.0001. The iterative process may also be terminated after a fixed number of iterations. This option acts as a backup criteria to prevent very slowly convergent or divergent problems from wasting computation time.

5.2 Transient Algorithms

Equation (5.1) represents a discrete space, continuous time approximation to the original system of partial differential equations. A direct time integration procedure replaces the continuous time derivative with an approximation for the history of the dependent variables over a small portion of the problem time scale. The result is an incremental procedure that advances the solution by discrete steps in time. In constructing such a procedure, questions of numerical stability and accuracy must be considered.

A large body of literature is available on possible time integration schemes for equations of the diffusion type. Both implicit and explicit methods, as well as mode superposition, have been used successfully. Each of these approaches have their own strengths and weaknesses, many of which are problem dependent. In order to provide solution capabilities for as wide a range of problems as possible, three different integration schemes are used in COYOTE. Two types of implicit methods are available, both of which make use of a predictor/corrector strategy to improve efficiency and accuracy. These procedures were originally developed by Gresho, *et al.* [19] and are used in COYOTE with only minor modifications. The third integration scheme is an explicit procedure that can be used effectively with the lower order elements available in the code. All of the integration methods may be used with either a fixed time step or an adaptive time step selection algorithm.

5.2.1 Forward/Backward Euler Integration

The first-order implicit integration method used in COYOTE employs a forward Euler scheme as a predictor with the backward Euler method functioning as the corrector step. Omitting the details of the derivation, the application of the explicit, forward Euler formula to equation (5.1) produces

$$\overline{\mathbf{M}}\mathbf{T}_p^{n+1} = \overline{\mathbf{M}}\mathbf{T}^n + \Delta t_n \left[\overline{\mathbf{F}}(\mathbf{T}^n) - \overline{\mathbf{K}}(\mathbf{T}^n)\mathbf{T}^n \right]. \quad (5.9)$$

This can be written in a form that is more suitable for computation by replacing the bracketed term with a rearranged form of (5.1) to produce

$$\mathbf{T}_p^{n+1} = \mathbf{T}^n + \Delta t_n \dot{\mathbf{T}}^n. \quad (5.10)$$

In equations (5.9) and (5.10) the superscript indicates the timeplane, the subscript p denotes a predicted value and $\Delta t_n = t_{n+1} - t_n$. By using the form shown in (5.10) a matrix inversion of $\overline{\mathbf{M}}$ is avoided; the “acceleration” vector $\dot{\mathbf{T}}^n$ is computed from a form of the corrector formula as shown below.

The corrector step of the first-order scheme is provided by the backward Euler (or fully implicit) method. When applied to equation (5.1) this implicit method yields

$$\overline{\mathbf{M}}\mathbf{T}^{n+1} = \overline{\mathbf{M}}\mathbf{T}^n + \Delta t_n \left[\overline{\mathbf{F}}(\mathbf{T}^{n+1}) - \overline{\mathbf{K}}(\mathbf{T}^{n+1})\mathbf{T}^{n+1} \right] \quad (5.11)$$

or in a form more suitable for computation

$$\left[\frac{1}{\Delta t_n} \overline{\mathbf{M}} + \overline{\mathbf{K}}(\mathbf{T}^{n+1}) \right] \mathbf{T}^{n+1} = \frac{1}{\Delta t_n} \overline{\mathbf{M}}\mathbf{T}^n + \overline{\mathbf{F}}(\mathbf{T}^{n+1}). \quad (5.12)$$

The implicit nature of this method is evident from the form of (5.12), since it is in effect, a nonlinear, algebraic system for the variables \mathbf{T} at timeplane $n + 1$.

The solution to (5.12) at timeplane $n + 1$ can be achieved by an iteration procedure such as Picard’s method. The rate of convergence of Picard’s method is greatly increased if the initial solution estimate is “close” to the true solution. The solution predicted from (5.9) provides this initial guess for the iterative procedure in a cost-effective manner.

5.2.2 Adams-Bashforth/Trapezoid Rule Integration

An implicit integration method that is second-order accurate in time can be developed along the same lines as described above. A second-order equivalent to the forward Euler method is the variable step, Adams-Bashforth predictor given by

$$\mathbf{T}_p^{n+1} = \mathbf{T}^n + \frac{\Delta t_n}{2} \left[\left(2 + \frac{\Delta t_n}{\Delta t_{n-1}} \right) \dot{\mathbf{T}}^n - \left(\frac{\Delta t_n}{\Delta t_{n-1}} \right) \dot{\mathbf{T}}^{n-1} \right] \quad (5.13)$$

where $\Delta t_n = t_{n+1} - t_n$ and $\Delta t_{n-1} = t_n - t_{n-1}$. This formula can be used to predict the solution vector given two “acceleration” vectors from previous timeplanes; no matrix solution is required.

A compatible corrector formula for use with (5.13) is available in the form of the trapezoid rule. When applied to equation (5.1) the trapezoid rule produces

$$\left[\frac{2}{\Delta t_n} \overline{\mathbf{M}} + \overline{\mathbf{K}}(\mathbf{T}^{n+1}) \right] \mathbf{T}^{n+1} = \frac{2}{\Delta t_n} \overline{\mathbf{M}} \mathbf{T}^n + \overline{\mathbf{M}} \dot{\mathbf{T}}^n + \overline{\mathbf{F}}(\mathbf{T}^{n+1}). \quad (5.14)$$

Equation (5.14) is observed to be a nonlinear, algebraic system for the vector \mathbf{T}^{n+1} and can again be solved using an iterative procedure such as Picard’s method.

5.2.3 Implicit Integration Procedures

The integration formulas outlined above form the basis for the implicit solution of time-dependent problems in COYOTE. The similarity of the first- and second-order methods makes it possible to include both procedures in a single, overall algorithm. The major steps in the time integration procedure are outlined here.

At the beginning of each time step it is assumed that all of the required solution and “acceleration” vectors are known and the time increment for the next step has been selected. To advance the solution from time t_n to time t_{n+1} then requires the following steps:

- 1) A tentative solution vector, \mathbf{T}_p^{n+1} , is computed using the predictor equations (5.10) or (5.13).
- 2) The corrector equations (5.12) or (5.14) are solved for the “true” solution, \mathbf{T}^{n+1} . This involves the iterative solution of (5.12) or (5.14) via Picard’s method. The predicted values \mathbf{T}_p^{n+1} are used to initialize the equation for the iteration procedure.
- 3) The “acceleration” vectors are updated using the new solution \mathbf{T}^{n+1} and the “inverted” forms of the corrector formulas. For the first-order method the acceleration is computed from the backward Euler definition

$$\dot{\mathbf{T}}^{n+1} = \frac{1}{\Delta t_n} (\mathbf{T}^{n+1} - \mathbf{T}^n)$$

while the second-order accelerations are derived from the trapezoid rule

$$\dot{\mathbf{T}}^{n+1} = \frac{2}{\Delta t_n} (\mathbf{T}^{n+1} - \mathbf{T}^n) - \dot{\mathbf{T}}^n.$$

- 4) A new integration time step is computed. The time step selection process is based on an analysis of the time truncation errors in the predictor and corrector formulas as described in Section 5.2.4. If a constant time step is being used, this step is omitted.
- 5) Return to step 1 for next time increment.

In actual implementation the Picard iteration process in step 2 is not carried to absolute convergence. Rather, a one-step correction is employed as advocated in [19]. This procedure is quite efficient and can be very accurate provided the time step is suitably controlled.

5.2.4 Time Step Control

Both of the implicit time integration procedures available in COYOTE can be used with a fixed, user specified time step or a time step that changes only at certain points during the integration interval. However, the *a priori* selection and modification of a reasonable integration time step can be a difficult task, especially for a complex problem. One of the benefits of using the predictor/corrector algorithms described here is that it provides a rational basis for dynamically selecting the time step.

The detailed derivation of the time step selection formula is omitted here. The reader interested in further details is referred to [19]. The general ideas for the time step selection process come from the well-established procedures for solving ordinary differential equations. By comparing the time truncation errors for two time integration methods of comparable order, a formula can be developed to predict the next time step based on a user specified error tolerance. In the present case, the time truncation errors for the explicit predictor and implicit corrector steps are analyzed and provide the required formulas.

The time step estimation formula is given by [19] as

$$\Delta t_{n+1} = \Delta t_n \left(b \cdot \frac{\epsilon^t}{d_{n+1}} \right)^m \quad (5.15)$$

where $m = 1/2$, $b = 2$ for the first-order method and $m = 1/3$, $b = 3(1 + \Delta t_{n-1}/\Delta t_n)$ for the second-order scheme. The user specified error tolerance for the integration process is ϵ^t , which has a typical value of 0.0001. The quantity d_{n+1} is an appropriate norm on the integration error, which is defined as the difference between the predicted solution and

the corrected value. In COYOTE the following RMS norm is used

$$d_{n+1} = \left[\frac{1}{N_{nodes} \cdot T_{max}^2} \sum_{i=1}^{N_{nodes}} \left(T_i^{(n+1)} - T_{i_p}^{(n+1)} \right)^2 \right]^{\frac{1}{2}} \quad (5.16)$$

where N_{nodes} is the number of nodes in the mesh and T_{max} is an appropriate maximum temperature for the problem that may be either specified or computed from the solution vector.

Unlike the procedure described in [19], COYOTE always uses the newly computed time step derived from (5.15). If $\Delta t_{n+1} \leq 0.5\Delta t_n$ a warning message is given to indicate a large reduction in the time step has occurred. However, the previous time step is not rejected nor recomputed.

5.2.5 Initialization

The predictor equations (5.10) and (5.13) require that one or more acceleration vectors be available at each timeplane in order to estimate a new solution vector. At the beginning of a transient solution these vectors are not generally available and thus a special starting procedure must be used. The approach taken in COYOTE is to use the dissipative, backward Euler method for the first few steps and then switch to either of the standard predictor/corrector methods. This procedure has the advantage that any nonphysical features of the numerical model are quickly damped by the backward Euler scheme.

For the first time step, the implicit, backward Euler scheme is used alone; the second step uses a forward Euler predictor and backward Euler corrector. Both of these steps use a fixed, user supplied time step. At the third step, the usual predictor/corrector integration procedure begins and automatic time step selection is started, if this option has been requested. The initial time step supplied by the user to start the problem should be very conservative to prevent large time step reductions when the automatic selection procedure takes control.

5.2.6 Forward Euler Integration

The explicit integration method used in COYOTE is based on the first-order, forward Euler method. This algorithm was previously defined in (5.10) and is written here as

$$\overline{\mathbf{M}}\mathbf{T}^{n+1} = \overline{\mathbf{M}}\mathbf{T}^n + \Delta t_n \left[\overline{\mathbf{F}}(\mathbf{T}^n) - \overline{\mathbf{K}}(\mathbf{T}^n)\mathbf{T}^n \right]. \quad (5.17)$$

Inverting the capacitance matrix $\overline{\mathbf{M}}$ allows (5.14) to be written in a computationally effective form as

$$\mathbf{T}^{n+1} = \mathbf{T}^n + \Delta t_n \overline{\mathbf{M}}^{-1} \left[\overline{\mathbf{F}}(\mathbf{T}^n) - \overline{\mathbf{K}}(\mathbf{T}^n) \mathbf{T}^n \right] = \mathbf{T}^n + \Delta t_n \overline{\mathbf{M}}^{-1} \overline{\mathbf{F}}_{\text{eff}}. \quad (5.18)$$

As written in (5.18), this algorithm does not require the solution of a matrix system but simply the construction of an effective flux vector from known data and a matrix-vector product.

The practical utility of (5.18) relies on two aspects of the algorithm. First, the inverse of the effective capacitance matrix must be easily obtainable, *i.e.*, it must be computationally inexpensive. Also, the explicit nature of the method means that the stability of the algorithm must be considered when choosing an integration time step.

5.2.7 Matrix Diagonalization

A particular feature of the finite element method when used for time-dependent problems is the inherent coupling that occurs between nodal point time derivatives. By inspecting the form of the element level capacitance matrix, as shown in (3.13), it is clear that \mathbf{M} is not a simple diagonal matrix but has a banded structure. This structure carries over to the global matrix, $\overline{\mathbf{M}}$.

The inverse to $\overline{\mathbf{M}}$ could be directly computed for use in the explicit algorithm in (5.18). Unfortunately, the inverse of a banded matrix is a full matrix; the generally large size of $\overline{\mathbf{M}}$ for two and three-dimensional problems precludes the use of such an approach. For $\overline{\mathbf{M}}^{-1}$ to be computed efficiently, $\overline{\mathbf{M}}$ must have a diagonal form. Several strategies for diagonalizing $\overline{\mathbf{M}}$ have been proposed in the literature [14]. The approach taken in COYOTE is to use the row-sum technique to approximate \mathbf{M} at the element level. That is, the diagonal form of the element matrix, $\mathbf{M}_{\mathbf{D}}$, is formed by the components given by

$$m_{ii}^D = \sum_{j=1}^N m_{ij} \quad ; \quad m_{ij}^D = 0 \quad i \neq j \quad (5.19)$$

where N is the number of degrees of freedom in the element and m_{ij} are components of the original capacitance matrix; the global matrix will also have the same diagonal form. In the algorithm given in (5.18) the term $\overline{\mathbf{M}}^{-1}$ is replaced with $\overline{\mathbf{M}}_{\mathbf{D}}^{-1}$, which is easily computed from $1/m_{ii}^D$.

Diagonalization (or “lumping”) techniques, such as the row-sum method, have been widely used and investigated. It is known that the temporal response of the discretized equations is altered by such techniques, though good results can still be obtained with

careful use. A major limitation to diagonalization (and row-sum in particular) is its restriction to low-order (linear) finite element approximations. Higher order basis functions generally produce poor results when used in a diagonalized form. This result implies that the explicit integration procedures in COYOTE should only be used with the linear elements available in the code.

5.2.8 Stability and Time Step Control

Explicit integration methods are conditionally stable and thus require limits on the size of the integration time step. Conventional stability analyses of the forward Euler scheme for a diffusion equation [20] produce a time step restriction of the following form

$$\lambda_{max}^G \Delta t \leq 2 \quad (5.20)$$

where λ_{max}^G is the largest eigenvalue for the matrix system $\overline{\mathbf{M}}_{\mathbf{D}}^{-1} \overline{\mathbf{K}}$ from equation (5.18). The largest eigenvalue for the system can be bounded by the largest element eigenvalue, which in the present case is proportional to $1/h^2$ with h being a representative element dimension. From these results it is clear that the time step restriction for the explicit method is quite severe especially on highly refined meshes.

An effective control of the time step for the explicit method relies directly on the ability to evaluate the largest element eigenvalue in the system. Exact element eigenvalues could be computed by a number of methods but this type of accuracy and computational expense is not warranted for most applications. Instead, simple eigenvalue estimates that are rapidly computed from a formula are preferred. Such formulas have been derived for bar, quadrilateral and hexahedral elements with a linear temperature approximation [21,22], though these results are restricted to cases where one-point quadrature is used to evaluate the finite element integrals. For elements with multiple integration points, such as used in COYOTE, the theory and estimation procedure due to Lin [23] is required. The largest element eigenvalue can be bounded by the sum of the largest eigenvalues at each integration point. That is

$$\lambda_{max}^G \leq \lambda_{max}^E \leq \sum_{i=1}^{N_{ip}} \lambda_{max}^i \quad (5.21)$$

where N_{ip} is the number of integration points in the element. As shown in [23], the nonzero eigenvalues at an integration point can be found from the following matrix

$$\mathbf{S} = \frac{w}{m} \mathbf{k} \mathbf{B} \mathbf{B}^T \quad (5.22)$$

where w is the quadrature weight factor, m is the lumped capacitance at the integration point, \mathbf{k} is the conductivity matrix and \mathbf{B} is the temperature gradient operator defined

by

$$\mathbf{B} = \begin{Bmatrix} \frac{\partial \Theta}{\partial x} \\ \frac{\partial \Theta}{\partial y} \\ \frac{\partial \Theta}{\partial z} \end{Bmatrix}. \quad (5.23)$$

The vector Θ is the element interpolation function for the temperature. The maximum eigenvalue of \mathbf{S} is readily computed for each integration point since the operators in (5.22) are all available from the construction of the element matrices.

Further details of the derivation of (5.21) and (5.22) can be found in [21,23]. COYOTE uses (5.22) to estimate λ_{max}^i and the bound in (5.21) to estimate the maximum system eigenvalue; the system eigenvalue and equation (5.20) allow computation of a usable time step. The maximum stable time step computed from (5.20) can be also be scaled by the user to ensure a conservative time integration strategy.

5.3 Matrix Solution Procedures

When most of the algorithms of the previous chapters are applied at a given iteration or time step, the result is a matrix equation of the form

$$\mathbf{Ax} = \mathbf{b} \quad (5.24)$$

In the problems considered here the matrix \mathbf{A} is large, sparse, banded and symmetric; an unsymmetric system may occur in some applications. A solution to (5.24) can be achieved by either an iterative or direct method. Historically, direct methods, such as the frontal method or other forms of Gauss elimination, have been the solution methods of choice for most finite element applications. However, the computer memory and CPU inefficiency of direct methods with respect to large, three-dimensional problems, has produced renewed interest in iterative methods for (5.24).

The solution methods used in COYOTE are based on a preconditioned conjugate gradient (PCG) algorithm. The matrix solution technique is embedded in a PCG library package that was developed by Schunk and Shadid [24]. For application to the heat conduction problem, any of three different preconditions can be invoked: Jacobi, Polynomial and the Incomplete Choleski method. The unsymmetric convection problem requires an iterative method such as the generalized minimum residual method (GMRES). The fully coupled conduction-radiation problem (see Section 5.5) also requires an unsymmetric method such as GMRES. These unsymmetric methods may be used with

any of several preconditioners; all of the available preconditioners are derived from the assembled, global matrix, \mathbf{A} . To accommodate the storage requirements of \mathbf{A} , a standard sparse matrix format is used [25] that only records the nonzero entries of the \mathbf{A} matrix. Complete details on the iterative solvers available in COYOTE can be found in [24].

5.4 Radiation View Factor Algorithms

When enclosure radiation is coupled to the conduction problem, a number of auxiliary computations must be included in the solution process. As defined in Section 2.3, the net radiation method requires the evaluation of view factors for all radiating surfaces in the enclosure. For geometries that are stationary, this computation need only be performed once while radiation problems that include regions with specified motions must have the view factors updated periodically. COYOTE employs a number of different methods for the actual view factor computation, all of which are embedded in the companion computer code, CHAPARRAL [26]. CHAPARRAL was designed to take advantage of some well established view factor techniques as well as implement some newer, more efficient procedures. Full details of the algorithms in CHAPARRAL and use of the code libraries are available in [26].

The basic view factor definition is given by (15) as

$$F_{k-j} = \frac{1}{A_k} \int_{A_k} \int_{A_j} \frac{\cos \theta_k \cos \theta_j}{\pi S^2} dA_j dA_k \quad (5.25)$$

which is recognized as a relation that depends only on the geometry of the enclosure surfaces; the possibility of third surface shadowing must also be considered in the evaluation of (5.25). Three standard methods have been developed for evaluating F_{k-j} and these are generally known as a) the double area summation method [8], b) the contour or line integration method [8], and c) the semi-analytic or Mitalas and Stephenson method [27]. Each of these techniques is optimal for specific orientations of the view factor surfaces. The computer code FACET [27] employs all of these methods for view factor computation in conjunction with a selection criteria for switching between techniques. The FACET code has been made part of CHAPARRAL and can be accessed as an option for two- and three-dimensional geometries.

A significant difficulty with the procedures cited above is their poor efficiency for three-dimensional problems with large numbers of surfaces. CHAPARRAL has an alternate procedure available for these situations which is based on a hemicube algorithm [26,28]. This method performs very well, with good accuracy, on large-scale problems; the hemicube method is not available for two-dimensional applications. Details of the

technique and its implementation can be found in [26]. A comparison of all of the above view factor methods has been reported in [28].

5.5 Radiation Solution Algorithms

The enclosure radiation problem was outlined in Section 2.3 and resulted in two equations of the following form

$$\sum_{j=1}^N [\delta_{kj} - (1 - \epsilon_k)F_{k-j}] q_j^o = \epsilon_k \sigma T_k^4 \quad (5.26)$$

$$q_k = q_k^o - \sum_{j=1}^N F_{k-j} q_j^o. \quad (5.27)$$

When the surface temperatures for all surfaces are known, equation (5.26) forms a set of linear algebraic equations for the unknown, outgoing surface fluxes, q_j^o . That is, equation (5.26) can be written as

$$\mathbf{A}(\mathbf{T})\mathbf{q} = \mathbf{F}_\sigma(\mathbf{T}) \quad (5.28)$$

where \mathbf{A} is a function of \mathbf{T} due to the possible dependence of surface emissivities on temperature. The matrix \mathbf{A} is a full matrix due to the surface to surface coupling represented by the view factors F_{k-j} . This characteristic, along with the possible temperature dependencies, suggests the use of an iterative solution method for (5.28) rather than a direct matrix factorization.

COYOTE employs a Gauss-Seidel or progressive refinement method to solve (5.28) for the components of \mathbf{q} ; these solution algorithms are located in the CHAPARRAL code and are explained in more detail in [26]. When the \mathbf{q} values are available, equation (5.27) is then used to compute the effective flux to the surface. The surface fluxes provide boundary conditions to the finite element model for the conduction process. When new surface temperatures are computed, due to either a new time step or iteration cycle, the above process is repeated to obtain a new surface flux condition. The surface temperatures used in the above computation must be uniform over each surface in order to satisfy the conditions of the radiation model. In COYOTE it is assumed that each surface in the radiation problem corresponds to a face or edge of a finite element. The uniform surface temperature needed for use in (5.26) is obtained by combining (averaging) the nodal point temperatures on the appropriate element face or edge.

The decoupled, cyclic solution procedure outlined above is very reliable and efficient for time-dependent problems since the change in surface temperature over a time step is generally small and convergence is rapid. However, when time-independent problems are considered, the decoupled nature of the process leads to significant convergence problems.

The basic difficulty is the dependence of the radiative flux on the fourth power of the surface temperature; modest changes in temperature between iterations can lead to very large changes in surface flux which produce even larger variations in surface temperature. This nonlinear feedback can be controlled to a very limited extent by relaxation techniques such as found in equation (5.4). The proper resolution of this problem is found in one of two approaches - a (false) transient technique as described above that always stays close to the true solution or a more fully coupled solution technique that simultaneously solves for the temperature and surface flux.

For time independent problems the finite element form of the conduction and radiation equations can be expressed as

$$\bar{\mathbf{K}}(\mathbf{T})\mathbf{T} + \bar{\mathbf{B}}\mathbf{q} = \bar{\mathbf{F}}(\mathbf{T}) \quad (5.29)$$

and

$$\mathbf{A}(\mathbf{T})\mathbf{q} = \mathbf{F}_\sigma(\mathbf{T}) = \mathbf{D}(\mathbf{T})\mathbf{T} \quad (5.30)$$

In the conduction equation (5.29) the boundary conditions involving the radiative surface fluxes have been removed from the $\bar{\mathbf{F}}$ vector and written explicitly on the left-hand-side of the equation. Also, the right-hand-side of the net radiation equation has been linearized and made explicit in the surface temperature. The solution of this equation set for \mathbf{T} and \mathbf{q} could, in theory, be accomplished by solving (5.30) for \mathbf{q} (inverting the \mathbf{A} matrix) and substituting the result into (5.29), which would form a very nonlinear equation for the temperature. The fact that \mathbf{A} is usually a large (full) matrix precludes the use of this approach and forces consideration of simultaneous solution methods, such as Newton's method. Rewriting (5.29) and (5.30) as

$$\begin{Bmatrix} \mathbf{R}_\mathbf{T} \\ \mathbf{R}_\mathbf{q} \end{Bmatrix} = \begin{Bmatrix} \bar{\mathbf{K}}(\mathbf{T})\mathbf{T} + \bar{\mathbf{B}}\mathbf{q} \\ -\mathbf{D}(\mathbf{T})\mathbf{T} + \mathbf{A}(\mathbf{T})\mathbf{q} \end{Bmatrix} = \begin{Bmatrix} \bar{\mathbf{F}}(\mathbf{T}) \\ \mathbf{0} \end{Bmatrix} \quad (5.31)$$

This nonlinear system can be solved via Newton's method with the definition

$$[\mathbf{J}^n] \begin{Bmatrix} \Delta\mathbf{T}^{n+1} \\ \Delta\mathbf{q}^{n+1} \end{Bmatrix} = \begin{bmatrix} \frac{\partial\mathbf{R}_\mathbf{T}}{\partial\mathbf{T}} & \frac{\partial\mathbf{R}_\mathbf{T}}{\partial\mathbf{q}} \\ \frac{\partial\mathbf{R}_\mathbf{q}}{\partial\mathbf{T}} & \frac{\partial\mathbf{R}_\mathbf{q}}{\partial\mathbf{q}} \end{bmatrix} \begin{Bmatrix} \Delta\mathbf{T}^{n+1} \\ \Delta\mathbf{q}^{n+1} \end{Bmatrix} = - \begin{Bmatrix} \mathbf{R}_\mathbf{T}^n \\ \mathbf{R}_\mathbf{q}^n \end{Bmatrix} \quad (5.32)$$

The iterative solution procedure specified in (5.32) has very good convergence properties and is available as an option in COYOTE II. Note that the Jacobian is constructed to properly handle the nonlinearities that occur in the coupling between conduction and enclosure radiation, *i.e.*, the third power dependence on temperature found in the \mathbf{D} matrix. The Jacobian does not account for other temperature dependencies, such as material property or boundary condition variations, since these are linearized and

evaluated at the previously computed temperature. Though the method in (5.32) is very reliable, there is a significant penalty in computational cost. The matrix problem is increased in size by the total number of enclosure surfaces, which for complex, three-dimensional geometries may be very large. Also, the matrix problem is now unsymmetric and requires an iterative solution method such as GMRES. Further details of this solution option and its efficiency can be found in [29].

5.6 Chemical Reaction Solution Algorithm

The presence of reactive materials in the conduction problem requires that a number of nonlinear conservation equations be solved for the chemical species in conjunction with the temperature field. The general formulation for the chemistry problem was outlined in Section 2.4. The mathematical nature (stiffness) of the kinetic equations dictate that for computational efficiency, the chemistry and thermal diffusion equations be solved independently. In COYOTE the solution process is formally based on an operator splitting technique [30].

In the present application, operator splitting is particularly effective due to the form of the kinetic equations. Since diffusion of the species is neglected, the kinetic equations have no spatial gradients and reduce to ordinary differential equations that can be defined locally on each finite element. In essence, the chemical species can be viewed as state variables for each element and can be solved on an element-by-element basis. In COYOTE, all species equations are defined at the integration points for each element. During a time step, the chemistry solution is advanced first using a fixed (frozen) temperature field; the temperature field is subsequently advanced over the same time interval using the recently evaluated (frozen) chemistry result. If a predictor/corrector time integration method is employed, the frozen temperature field used for the chemistry solution is the temperature produced from the predictor step. When a predictor equation is not employed for time integration, the last available temperature field is used for the chemistry solution.

The inherent stiffness of the kinetic equations requires that special integration methods be used to advance the chemistry solution in time. COYOTE makes use of the stiff ordinary differential equation (ODE) routines developed by T. R. Young in the package, CHEMEQ [31]. The techniques used in CHEMEQ were developed specifically for chemical reaction systems and are based on a combination of classical predictor/corrector methods and asymptotic methods for the stiff components of the system. The rate equations for each reactive element are solved using their own integration time step over the global time step of interest. The most restrictive chemistry time step for all of the reac-

tive elements is used to regulate the choice of the thermal diffusion time step. The choice of the thermal diffusion time step is computed from

$$\Delta t_{n+1} = \min\{\Delta t_{diff}, X_{chem} \times \Delta t_{chem}\} \quad (5.33)$$

where Δt_{diff} is the estimated time step for the heat conduction equation (*e.g.*, equation (5.15) or (5.20)), and Δt_{chem} is the minimum time step estimated for the chemistry solution. The parameter X_{chem} is a user-defined scale factor that typically has a value between 10 and 100. When reactive processes are unimportant, the adaptive time integration in CHEMEQ will produce a chemistry time step that is relatively large and equation (5.33) will allow the conduction solution to dictate the problem time scale. As the reactive process accelerates, the chemistry time step will decrease significantly and ultimately control the time step formula in equation (5.33). The transition point for control of the global time step is dictated by the user through the X_{chem} parameter.

5.7 Phase Change Algorithms

The standard enthalpy method for including latent heat effects in a change of phase problem was outlined in Section 2.2. The solution procedure consists of replacing the specific heat for the material with an effective specific heat function that includes the temperature-dependent latent heat release. In a form that is amenable to computation, the effective specific heat is given by

$$C^*(T) = C(T) + L\delta^*(T - T_t, \Delta T) \quad (5.34)$$

where δ^* is the delta form function; δ^* has a large but finite value in the interval centered about T_t and is zero outside the interval. This equation is the computational analogue to equation (2.13) and is illustrated in Figure 5.1. The interval ΔT is often referred to as the “mushy” zone and corresponds to the difference between the liquidus, T_l , and solidus, T_s , temperatures for the material. Note that (5.34) is thus an approximation for the behavior of pure materials that change phase at a specified temperature, T_t , but is accurate for nonpure substances that have truly distinct liquidus and solidus temperatures.

The effective capacitance model described above is available in COYOTE and represents the usual method for this type of simulation. However, some caution must be exercised when generating time dependent solutions with this model. Since the transition temperature interval, ΔT , is often small compared to the overall temperature variation in the conduction problem, there are some severe practical limitations on the time integration procedure. In general, the time-stepping algorithm must be controlled such that every node that “changes phase” is forced to attain a temperature value in the interval bracketed by ΔT . If a nodal point does not “land” in the ΔT range but, simply

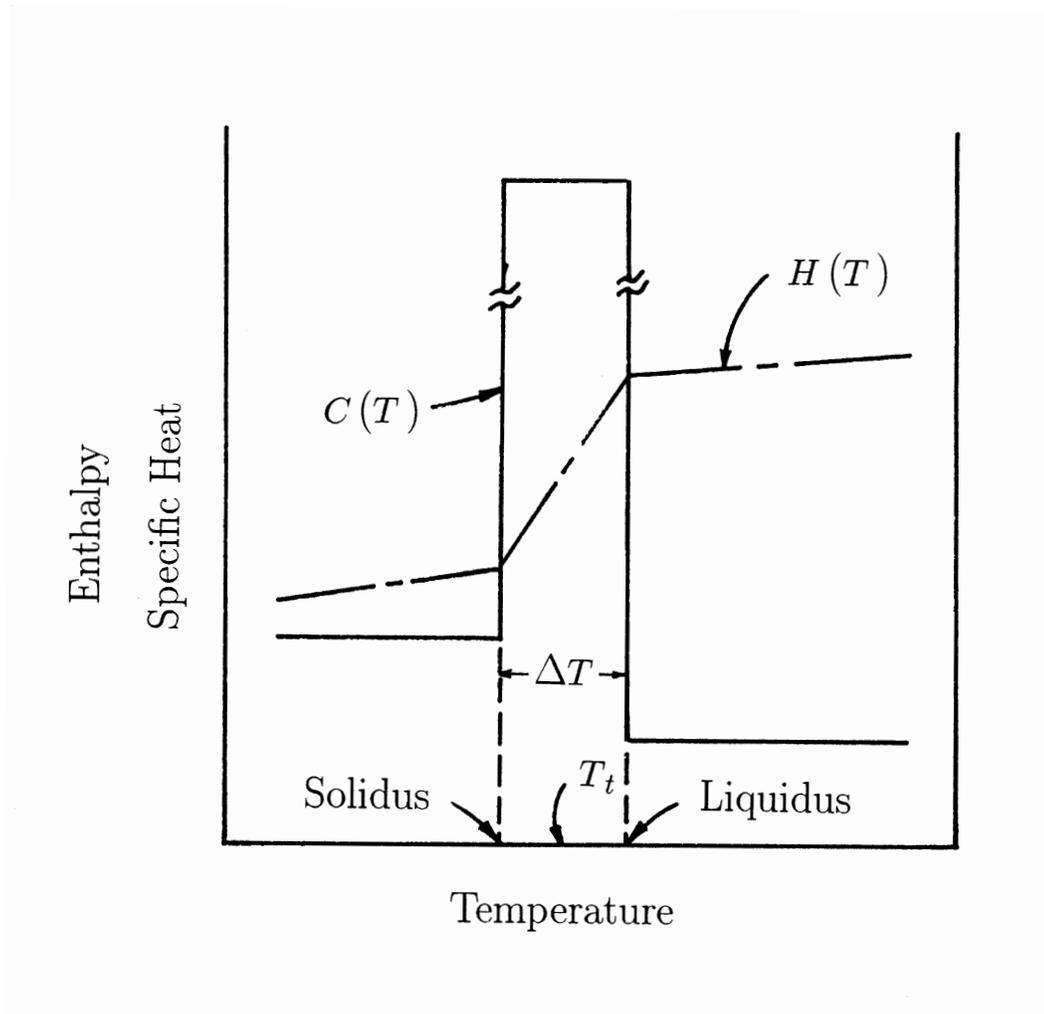


Figure 5.1: Definition of material properties for phase change computation.

steps over this temperature interval, the latent heat effect is lost for that node and an incorrect temperature response and energy balance will result. Methods for dealing with this difficulty include broadening the ΔT range and placing a limit on the maximum temperature change that can occur during a time step. Integration time step control, based on limiting the temperature change, is available in COYOTE as an option.

Alternatives to the methods based on specific heat make use of the enthalpy, H , versus temperature curve for the phase change material and compute an effective specific heat based on the local slope of the enthalpy function. In one case, the following definition is used

$$C_p = \frac{dH}{dT} = \frac{dH/dt}{dT/dt} \quad (5.35)$$

which can be rewritten in a computational form as

$$C_p = \frac{H(T^{n+1}) - H(T^n)}{T^{n+1} - T^n} \quad (5.36)$$

where the superscript denotes the time step number. Equation (5.36) can be evaluated at each element integration point to produce the effective specific heat needed for the construction of the element integrals. For situations where the denominator in (5.36) is zero, an artificial temperature difference is created to allow the derivative to be evaluated. Note that this approach has the same type of time step restrictions as the previous, capacitance-based method. A second method that employs the enthalpy function is defined by using spatial gradients in place of the time derivatives in (5.35). In this case the enthalpy is first computed at the nodes of the element (knowing T) and the effective specific heat at the integration points is then recovered from

$$C_p = \left[\frac{\nabla H \cdot \nabla H}{\nabla T \cdot \nabla T} \right]^{1/2} \quad (5.37)$$

where ∇H and ∇T are evaluated via the element shape functions. This technique will maintain its accuracy as long as the phase boundary passes through each element and does not skip over an element. Both of these enthalpy-based methods are available in COYOTE.

A final method for simulating latent heat release involves the construction of a temperature dependent, volumetric heat source. From equation (5.34) the term involving the latent heat can be transferred to the right-hand-side of the energy equation (2.1) to produce a volumetric source term of the form

$$Q_{th} = -\rho L \delta^*(T - T_t, \Delta T) \frac{\partial T}{\partial t} \quad (5.38)$$

The definition of δ^* indicates that the volume source is only active during the phase change and has a magnitude proportional to the latent heat release. The presence of

the time derivative in the source definition complicates the solution process and would generally lead to the source term be lagged in time. This type of phase change model could be used in COYOTE through proper definition of the source term, but is not recommended.

5.8 Contact Algorithm

The heat transfer aspects of contact between two material regions were considered in Section 4.10.3 where a surface flux vector was developed based on the identification of a master and slave surface. However, before this formulation can be utilized, the geometric properties associated with the contact conditions must be determined. Contact detection involves identifying the time at which contact (or separation) occurs and the location (coordinates) of the slave nodes on the master surface. COYOTE II employs a contact detection algorithm that was initially developed for use in solid mechanics finite element codes [32]. The use of these specific techniques allows coupled, thermal-stress problems with contact surfaces to be simulated with a completely consistent approach.

Two specific types of contact are considered in COYOTE and these differ only in the method of defining potential contacting surfaces. For problems in which a contact history is known, COYOTE allows contacting surface pairs to be specifically identified. In this case, the search for slave node locations on a master surface is limited exclusively to the paired surface. This option is most effective for static contact and predefined sliding or normal contact. A more general option in COYOTE allows multiple surfaces and/or blocks of elements to be defined such that arbitrary combinations of surface contacts may occur. This situation requires a more global search for slave node locations since contacting surface pairs are not predefined. The general nature of this specification allows the kinematics of the various material regions to dictate the occurrence of contact. Also, problems involving self contact (*e.g.*, buckling or folding) may be considered as well as simulations with surfaces that evolve in time (*e.g.*, tearing, material addition and deletion). Note that since COYOTE considers only the energy equation and has no facilities for momentum or force computations, the kinematics specified for a problem must be consistent with any contact processes that occur. In particular, situations involving penetration and deformation of contacting regions are expected to be resolved by a solid mechanics code before being passed to COYOTE.

The global search procedures used to detect contact and locate the coordinates for a slave node on a master surface are detailed in [32]. This reference also describes the facility to continually redefine the exterior surface of a finite element region for problems

involving material addition and deletion. The subroutines developed for the algorithms defined in [32] have been directly incorporated into COYOTE.

Chapter 6

Pre- and Post-Processing

The COYOTE program was designed to be a self-contained analysis package with the necessary options to set up a problem, solve for the required dependent variables and analyze the resultant solution in terms of derived quantities. The present chapter documents some of the numerical procedures used in the pre-solution and data analysis chapters of the program.

6.1 Mesh Generation

COYOTE contains no mesh generation capability and relies completely on external mesh generation software for a geometric description of the problem. The code reads mesh generation data from a standard format file called EXODUS II [33]. A complete description of the mesh generation interface to COYOTE is available in the user's manual [4].

6.2 Flux Computation

The thermal fluxes associated with the conduction equation can be computed in COYOTE on an element-by-element basis. Fourier's law provides the definition of the conductive heat flux as

$$\begin{aligned}q_x &= -k_{xx} \frac{\partial T}{\partial x} - k_{xy} \frac{\partial T}{\partial y} - k_{xz} \frac{\partial T}{\partial z} \\q_y &= -k_{yx} \frac{\partial T}{\partial x} - k_{yy} \frac{\partial T}{\partial y} - k_{yz} \frac{\partial T}{\partial z}\end{aligned}\tag{6.1}$$

$$q_z = -k_{zx} \frac{\partial T}{\partial x} - k_{zy} \frac{\partial T}{\partial y} - k_{zz} \frac{\partial T}{\partial z}$$

The component fluxes in (6.1) are computed by using the standard finite element approximations for T ,

$$T(x_i, t) = \Theta^T(x_i) \mathbf{T}(t)$$

and the relations for the local temperature derivatives as derived in Section 4.8. That is,

$$\begin{pmatrix} \frac{\partial \Theta}{\partial x} \\ \frac{\partial \Theta}{\partial y} \\ \frac{\partial \Theta}{\partial z} \end{pmatrix} = \mathbf{J}^{-1} \begin{pmatrix} \frac{\partial \Theta}{\partial s} \\ \frac{\partial \Theta}{\partial t} \\ \frac{\partial \Theta}{\partial r} \end{pmatrix} = \frac{1}{|\mathbf{J}|} \begin{bmatrix} \mathcal{J}_{11} & \mathcal{J}_{12} & \mathcal{J}_{13} \\ \mathcal{J}_{21} & \mathcal{J}_{22} & \mathcal{J}_{23} \\ \mathcal{J}_{31} & \mathcal{J}_{32} & \mathcal{J}_{33} \end{bmatrix} \begin{pmatrix} \frac{\partial \Theta}{\partial s} \\ \frac{\partial \Theta}{\partial t} \\ \frac{\partial \Theta}{\partial r} \end{pmatrix}$$

Using these definitions the flux components become

$$\begin{aligned} q_x &= -\frac{k_{xx}}{|\mathbf{J}|} \left(\mathcal{J}_{11} \frac{\partial \Theta^T}{\partial s} \mathbf{T} + \mathcal{J}_{12} \frac{\partial \Theta^T}{\partial t} \mathbf{T} + \mathcal{J}_{13} \frac{\partial \Theta^T}{\partial r} \mathbf{T} \right) \\ &\quad -\frac{k_{xy}}{|\mathbf{J}|} \left(\mathcal{J}_{21} \frac{\partial \Theta^T}{\partial s} \mathbf{T} + \mathcal{J}_{22} \frac{\partial \Theta^T}{\partial t} \mathbf{T} + \mathcal{J}_{23} \frac{\partial \Theta^T}{\partial r} \mathbf{T} \right) \\ &\quad -\frac{k_{xz}}{|\mathbf{J}|} \left(\mathcal{J}_{31} \frac{\partial \Theta^T}{\partial s} \mathbf{T} + \mathcal{J}_{32} \frac{\partial \Theta^T}{\partial t} \mathbf{T} + \mathcal{J}_{33} \frac{\partial \Theta^T}{\partial r} \mathbf{T} \right) \\ q_y &= -\frac{k_{yx}}{|\mathbf{J}|} \left(\mathcal{J}_{11} \frac{\partial \Theta^T}{\partial s} \mathbf{T} + \mathcal{J}_{12} \frac{\partial \Theta^T}{\partial t} \mathbf{T} + \mathcal{J}_{13} \frac{\partial \Theta^T}{\partial r} \mathbf{T} \right) \\ &\quad -\frac{k_{yy}}{|\mathbf{J}|} \left(\mathcal{J}_{21} \frac{\partial \Theta^T}{\partial s} \mathbf{T} + \mathcal{J}_{22} \frac{\partial \Theta^T}{\partial t} \mathbf{T} + \mathcal{J}_{23} \frac{\partial \Theta^T}{\partial r} \mathbf{T} \right) \\ &\quad -\frac{k_{yz}}{|\mathbf{J}|} \left(\mathcal{J}_{31} \frac{\partial \Theta^T}{\partial s} \mathbf{T} + \mathcal{J}_{32} \frac{\partial \Theta^T}{\partial t} \mathbf{T} + \mathcal{J}_{33} \frac{\partial \Theta^T}{\partial r} \mathbf{T} \right) \\ q_z &= -\frac{k_{zx}}{|\mathbf{J}|} \left(\mathcal{J}_{11} \frac{\partial \Theta^T}{\partial s} \mathbf{T} + \mathcal{J}_{12} \frac{\partial \Theta^T}{\partial t} \mathbf{T} + \mathcal{J}_{13} \frac{\partial \Theta^T}{\partial r} \mathbf{T} \right) \\ &\quad -\frac{k_{zy}}{|\mathbf{J}|} \left(\mathcal{J}_{21} \frac{\partial \Theta^T}{\partial s} \mathbf{T} + \mathcal{J}_{22} \frac{\partial \Theta^T}{\partial t} \mathbf{T} + \mathcal{J}_{23} \frac{\partial \Theta^T}{\partial r} \mathbf{T} \right) \\ &\quad -\frac{k_{zz}}{|\mathbf{J}|} \left(\mathcal{J}_{31} \frac{\partial \Theta^T}{\partial s} \mathbf{T} + \mathcal{J}_{32} \frac{\partial \Theta^T}{\partial t} \mathbf{T} + \mathcal{J}_{33} \frac{\partial \Theta^T}{\partial r} \mathbf{T} \right) \end{aligned} \tag{6.2}$$

In addition to the local components of the flux vector, the heat flux normal to the surface (edge) is often of importance. By definition

$$\begin{aligned} q_n &= \mathbf{q} \cdot \mathbf{n} \\ \mathbf{q} &= q_x \mathbf{e}_x + q_y \mathbf{e}_y + q_z \mathbf{e}_z \\ \mathbf{n} &= n_x \mathbf{e}_x + n_y \mathbf{e}_y + n_z \mathbf{e}_z \end{aligned} \tag{6.3}$$

and thus

$$q_n = q_x n_x + q_y n_y + q_z n_z. \tag{6.4}$$

In order to employ (6.4), the components of the normal vector are required. These are obtained from the surface vectors \mathbf{e}_1 and \mathbf{e}_2 given by

$$\mathbf{e}_1 = \begin{Bmatrix} \frac{\partial x}{\partial s_s} \\ \frac{\partial y}{\partial s_s} \\ \frac{\partial z}{\partial s_s} \end{Bmatrix} ; \quad \mathbf{e}_2 = \begin{Bmatrix} \frac{\partial x}{\partial t_s} \\ \frac{\partial y}{\partial t_s} \\ \frac{\partial z}{\partial t_s} \end{Bmatrix}$$

which were previously defined in Section 4.10.2. These vectors are related to the unit normal \mathbf{n} by

$$\mathbf{n} = \frac{\mathbf{e}_1 \times \mathbf{e}_2}{|\mathbf{J}_s|}$$

where $|\mathbf{J}_s|$ is defined in (4.38) as $|\mathbf{e}_1 \times \mathbf{e}_2|$.

The definitions in (6.2) are sufficient to define the flux components at any point s_0, t_0, r_0 within an element. In COYOTE the flux components are evaluated in the interior of each element at selected integration points. For quadrilateral and hexahedral elements the selected interior points are typically the $2 \times 2 \times 2$ Gauss points as recommended by [34]. Other element types also have recommended interior points for accurate derivative computations. Note that fluxes computed from temperature gradients are discontinuous between elements. To produce a continuous flux distribution, the integration point flux values are linearly extrapolated to the nodes of each element and averaged between all connected elements. Flux components can also be combined with the definition in (6.4) to generate the normal flux on the element surface (edge); fluxes normal to the element surface (edge) may be integrated over the boundary to define the total energy transfer to or from the element.

6.3 Heat Flow Function

For two-dimensional problems, Kimura and Bejan [35] have proposed the use of a heat flow function to assist in the visualization of energy transport. The heat function is

directly analogous to the stream function for incompressible fluid flow and is constructed to satisfy the steady, source free form of the energy equation. In formal terms, the heat function \mathcal{H} is the remaining nonzero component of a vector potential that identically satisfies a form of equation (2.1). By definition

$$\begin{aligned} q_1 = q_x &= \rho C u_x T - k \frac{\partial T}{\partial x} = \frac{\partial \mathcal{H}}{\partial y} \\ q_2 = q_y &= \rho C u_y T - k \frac{\partial T}{\partial y} = -\frac{\partial \mathcal{H}}{\partial x} \end{aligned} \quad (6.5)$$

where the flux components have been defined as the total of the convective and diffusive fluxes. For simplicity the definitions in (6.5) have also assumed an isotropic conductivity though this is not a required restriction. In the usual applications considered here, the velocities in (6.5) will be zero and the heat function will reduce to a definition for a heat flux line, *i.e.* a line that is everywhere tangent to the local flux vector. The change in the heat function is an exact differential such that

$$\begin{aligned} \delta \mathcal{H} &= \int_A^B \mathbf{q} \cdot \mathbf{n} d\Gamma \\ \mathbf{q} &= q_x \mathbf{e}_x + q_y \mathbf{e}_y \\ \mathbf{n} &= n_x \mathbf{e}_x + n_y \mathbf{e}_y \end{aligned} \quad (6.6)$$

where \mathbf{n} is the normal to the integration path $d\Gamma$, \mathbf{q} is the total flux vector along the path and \mathbf{e}_i are unit vectors in the coordinate directions.

The calculation of the change in the heat function within a finite element can be carried out using (6.6) once a suitable integration path AB is identified. In COYOTE the integration path is taken along the two-dimensional element boundaries. Consider the typical element boundary shown in Figure 6.1 with the following definitions

$$\begin{aligned} q_x &= \hat{\Phi}^T \mathbf{q}_x & ; & & q_y &= \hat{\Phi}^T \mathbf{q}_y \\ x &= \hat{\Upsilon}^T \mathbf{x} & ; & & y &= \hat{\Upsilon}^T \mathbf{y} \end{aligned} \quad (6.7)$$

where $\hat{\Phi}$ and $\hat{\Upsilon}$ are interpolation (edge) functions and \mathbf{q}_x , \mathbf{q}_y , \mathbf{x} , \mathbf{y} are vectors of nodal point fluxes and coordinates. The normal vector is given by

$$\mathbf{n} = \frac{1}{\Delta} \frac{\partial y}{\partial s} \mathbf{e}_x - \frac{1}{\Delta} \frac{\partial x}{\partial s} \mathbf{e}_y \quad (6.8)$$

with $d\Gamma$ defined in the usual way by

$$d\Gamma = \left[\left(\frac{\partial x}{\partial s} \right)^2 + \left(\frac{\partial y}{\partial s} \right)^2 \right]^{\frac{1}{2}} ds$$

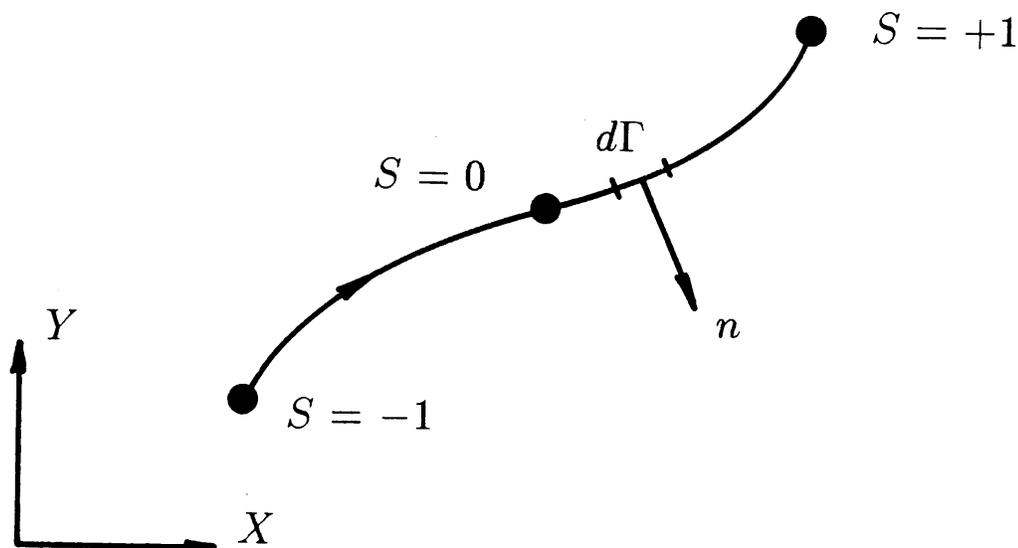


Figure 6.1: Definition of element boundary for heat function computation.

or using the definitions of (6.7)

$$d\Gamma = \left[\left(\frac{\partial \hat{\mathbf{Y}}^T}{\partial s} \mathbf{x} \right)^2 + \left(\frac{\partial \hat{\mathbf{Y}}^T}{\partial s} \mathbf{y} \right)^2 \right]^{\frac{1}{2}} ds = \Delta ds.$$

Combining these relations with the definition for $\delta\mathcal{H}$ produces

$$\delta\mathcal{H} = \int_{-1}^{+1} \left(\frac{\partial \hat{\mathbf{Y}}^T}{\partial s} \mathbf{y} \hat{\Phi}^T \mathbf{q}_x - \frac{\partial \hat{\mathbf{Y}}^T}{\partial s} \mathbf{x} \hat{\Phi}^T \mathbf{q}_y \right) ds. \quad (6.9)$$

The interpolation function definitions were described previously in Section 4.10.2; the function $\hat{\mathbf{Y}}$ can be either linear or quadratic depending on the shape of the element boundary. The change in the heat function along any element boundary can be computed from (6.9) once the element geometry, velocity and temperature fields are specified; the fluxes needed in (6.9) are derived from the definitions in (6.7) and the formulas outlined in the previous chapter. Computation of the heat function field for an entire finite element mesh is generated by applying (6.9) along successive element boundaries, starting at a node for which a base value of \mathcal{H} has been specified.

The calculation of the heat function for axisymmetric geometries follows a similar

procedure with the appropriate definition for \mathcal{H} being,

$$q_1 = q_r = \frac{1}{r} \frac{\partial \mathcal{H}}{\partial z} \quad ; \quad q_2 = q_z = -\frac{1}{r} \frac{\partial \mathcal{H}}{\partial r} \quad (6.10)$$

and

$$\mathbf{q} = q_r r \mathbf{e}_r + q_z r \mathbf{e}_z$$

$$\mathbf{n} = n_r \mathbf{e}_r + n_z \mathbf{e}_z.$$

6.4 Gas Fraction

For thermal problems with chemical reaction, the reacted gas fraction can be computed from (2.25). For a coupled thermo-mechanical analysis, the gas fraction may be an input variable to the constitutive model for the reacting material.

6.5 Graphical Output

COYOTE contains no graphics capability and relies completely on external visualization software. The code outputs solution data in a standard format file called EXODUS II [33] that can be accessed by any of several graphics packages, such as the BLOT code [36] or the AVS [37] software. Details of the output file are available in [4].

Chapter 7

References

1. J. P. Smith, "SINDA User's Manual," Report 14690-H001-R0-00, TRW Systems Group, Redondo Beach, CA (1971)
2. F. A. Rockenbach, "P/THERMAL, Version 1.0 Technical Reference Manual," Report P/N2192006, PDA Engineering, Santa Ana, CA (1986)
3. ABAQUS User's Manual, Version 4.8, Hibbitt, Karlsson and Sorenson, Inc., Providence, RI (1990)
4. D. K. Gartling and R. E. Hogan, "COYOTE - A Finite Element Computer Program for Nonlinear Heat Conduction Problems, Part II - User's Manual," SAND94-1179, Sandia National Laboratories, Albuquerque, NM (1994)
5. D. K. Gartling and M. B. Sirman, "COYOTE - A Finite Element Computer Program for Nonlinear Heat Conduction Problems, Part III - Example Problems," SAND94-1180, Sandia National Laboratories, Albuquerque, NM (1994)
6. H. S. Carslaw and J. C. Jaeger, "*Conduction of Heat in Solids*," Clarendon Press, Oxford, 2nd Edition (1959)
7. J. Crank, "*The Mathematics of Diffusion*," Clarendon Press, Oxford, 2nd Edition (1975)
8. R. Siegel and J. R. Howell, "*Thermal Radiation Heat Transfer*," McGraw Hill, NY, 2nd Edition (1981)
9. C. Bonacina, G. Comini, A. Fasano and M. Primicerio, "Numerical Solution of Phase-Change Problems," *Int. J. Heat Mass Transfer*, **16**, 1825-1832 (1973)
10. D. K. Gartling, "Finite Element Analysis of Convective Heat Transfer Problems with Change of Phase," *Computer Methods in Fluids*, K. Morgan, C. Taylor and C. A. Brebbia, Eds., Pentech Press, London, 257-284 (1980)

11. G. Comini, S. DelGuidice, R. Lewis and O. C. Zienkiewicz, "Finite Element Solution of Nonlinear Heat Conduction Problems with Special Reference to Phase Change," *Int. J. Num. Meth. Engng.*, **8**, 613-624 (1974)
12. K. Morgan, "A Numerical Analysis of Freezing and Melting With Convection," *Comp. Meth. Applied Mech. Engr.*, **28**, 275-284 (1981)
13. E. B. Becker, G. F. Carey and J. T. Oden, "*Finite Elements, An Introduction, Volume I*," Prentice-Hall, NJ (1981)
14. O. C. Zienkiewicz, "*The Finite Element Method*," McGraw-Hill, London, 3rd Edition (1977)
15. I. Ergatoudis, B. M. Irons and O. C. Zienkiewicz, "Curved, Isoparametric, 'Quadrilateral', Elements for Finite Element Analysis," *Int. J. Solids Structures*, **4**, 31-42 (1968)
16. B. M. Irons, "Quadrature Rules for Brick Based Finite Elements," *Int. J. Num. Meth. Engng.*, **3**, 293-294 (1971)
17. L. M. Taylor and D. P. Flanagan, "PRONTO 3D - A Three-Dimensional Transient Solid Dynamics Program," SAND87-1912, Sandia National Laboratories, Albuquerque, NM (1989)
18. R. D. Cook, D. S. Malkus and M. E. Plesha, "*Concepts and Applications of Finite Element Analysis*," John Wiley and Sons, NY (1989)
19. P. M. Gresho, R. L. Lee and R. L. Sani, "On the Time Dependent Solution of the Incompressible Navier-Stokes Equations in Two and Three Dimensions," *Recent Advances in Numerical Methods in Fluids, Volume 1*, Pineridge Press, Swansea, U. K., 27-81 (1980)
20. T. J. R. Hughes, "Analysis of Transient Algorithms with Particular Referenceto Stability Behavior," *Computational Methods for Transient Analysis*, T. Belytschko and T. J. R. Hughes, Eds., North-Holland, Amsterdam, 68-155 (1983)
21. T. Belytschko, "An Overview of Semidiscretization and Time Integration Procedures," *Computational Methods for Transient Analysis*, T. Belytschko and T. J. R. Hughes, Eds., North-Holland, Amsterdam, 1-65 (1983)
22. W. K. Liu and T. Belytschko, "Efficient Linear and Nonlinear Heat Conduction with a Quadrilateral Element," *Int. J. Num. Meth. Engng.*, **20**, 931-948 (1984)
23. J. I. Lin, "Bounds on Eigenvalues of Finite Element Systems," *Int. J. Num. Meth. Engng.*, **32**, 957-967 (1991)
24. P. R. Schunk and J. N. Shadid, "Iterative Solvers in Implicit Finite Element Codes," SAND92-1158, Sandia National Laboratories, Albuquerque, NM (1992)

-
25. Y. Saad, "SPARSKIT: A Basic Tool Kit for Sparse Matrix Computations," Tech. Report, Research Institute for Advanced Computer Science, NASA Ames, Moffitt Field, CA (1990)
 26. M. W. Glass, "Chaparral - A Library Package for Solving Large Enclosure Radiation Heat Transfer Problems," (in preparation), Sandia National Laboratories, Albuquerque, NM (1994)
 27. A. B. Shapiro, "FACET - A Radiation View Factor Computer Code for Axisymmetric, 2D Planar, and 3D Geometries with Shadowing," UCID-19887, Lawrence Livermore Laboratory, Livermore, CA (1983)
 28. A. F. Emery, O. Johansson, M. Lobo and A. Abrous, "A Comparative Study of Methods for Computing the Diffuse Radiation Viewfactors for Complex Structures," *J. Heat Trans.*, **113**, 413-422 (1991)
 29. R. E. Hogan and D. K. Gartling, "Finite Element Solution of Fully Coupled Heat Conduction and Enclosure Radiation Problems," *Proc. 8th Int. Conf. Num. Meth. Thermal Prob.*, Pineridge Press, Swansea, U. K., 1651-1662 (1993)
 30. I. S. Wichman, "On the Use of Operator-Splitting Methods for the Equations of Combustion," *Combust. Flame*, **83**, 240-252 (1991)
 31. T. R. Young, "CHEMEQ - A Subroutine for Solving Stiff Ordinary Differential Equations," NRL Memorandum Report 4091, Naval Research Laboratory, Washington, DC (1980)
 32. M. W. Heinstein, S. W. Attaway, J. W. Swegle and F. J. Mello, "A General-Purpose Contact Detection Algorithm for Nonlinear Structural Analysis Codes," SAND92-2141, Sandia National Laboratories, Albuquerque, NM (1993)
 33. L. A. Schoof and V. R. Yarberr, "EXODUS II - A Finite Element Data Model," SAND92-2137, Sandia National Laboratories, Albuquerque, NM (1994)
 34. E. Hinton and J. S. Campbell, "Local and Global Smoothing of Discontinuous Finite Element Functions Using a Least Squares Method," *Int. J. Num. Meth. Engng.*, **8**, 461-480 (1974)
 35. S. Kimura and A. Bejan, "The 'Heatline' Visualization of Convective Heat Transfer," *J. Heat Trans.*, **105**, 916-919 (1983)
 36. A. P. Gilkey and J. H. Glick, "BLOT - A Mesh and Curve Plot Program for the Output of a Finite Element Analysis," SAND88-1432, Sandia National Laboratories, Albuquerque, NM (1989)
 37. C. D. Upson, *et al.*, "The Application Visualization System: A Computational Environment for Scientific Visualization," *IEEE Computer Graphics and Applications*, **9** (1989)