SANDIA REPORT
SAND2009-4926
Unlimited Release
Printed April 2010

COYOTE - A Finite Element Computer Program for Nonlinear Heat Conduction Problems
Part I - Theoretical Background

David K. Gartling, Roy E. Hogan, and Micheal W. Glass

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Depeartment of Energy's National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

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

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2009-4926
Unlimited Release
March 2010

COYOTE - A FINITE ELEMENT COMPUTER PROGRAM FOR NONLINEAR HEAT CONDUCTION PROBLEMS

PART I - THEORETICAL BACKGROUND

Version 5.00
Released May 1, 2009
Printed March 19, 2010

David K. Gartling, Roy E. Hogan, and Micheal W. Glass
dkgartl@sandia.gov, rehogan@sandia.gov, and mwglass@sandia.gov
Engineering Sciences Center
Sandia National Laboratories
P. O. Box 5800
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. 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 SAND2010-0714.

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 usage and longevity that it currently enjoys. In response to user needs, the original program under went several minor upgrades plus a major revision during the late 1980's. 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, dictated the need for completely new versions of the older codes. In rewriting the COYOTE program, the two and three-dimensional codes were combined into a single software package, COYOTE II, that was released in 1994. The present series of reports describe the latest version (Version 5.0) of the program package, which has reverted to its original name, COYOTE, to avoid confusion in future releases.

In an effort to make the program more flexible and more generally applicable, a number of new capabilities and features have been added to COYOTE to produce COYOTE, Version 5.0. The major extension of the code is the capability to optionally include one or two additional diffusion equations that may be coupled to the primary heat conduction equation. The variables in these added equations are available to the boundary conditions, source terms and material property functions in the conduction equation. Conversely, the temperature field is available to the auxiliary diffusion equations. As part of this expansion, a time harmonic option for heat transfer was also added. Two thermal diffusion equations are solved in this case for the real and imaginary parts of the temperature field. Some changes to the sequential solution algorithm for coupled conduction and radiation have been made to improve convergence of this type of method. A fully coupled conduction/radiation solution method has been generalized and reinstalled to allow operation in a parallel environment. The fully coupled algorithm was made possible by a complete changeover to the use of the Finite Element Interface (FEI) with the improved access to the solver libraries available in the Trilinos package. The code may now be compiled using either a single or double precision word length. Some minor changes in problem capability and control have also been added. Most notable among these changes are the allowance of a time evolving mass flow to a bulk node (due to chemical reaction) and user defined material parameters now being passed to user subroutines.

Contents

List of Figures

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 COMSOL [1], COSMOSWorks [2], ABAQUS [3] and TSS [4] 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. Because of its architecture, the code has also proved useful for research in numerical algorithms and development of thermal analysis capabilities. This general philosophy has been retained in the current version of the program, COYOTE, Version 5.0, though the capabilities of the code have been significantly expanded. A major change in the code is its availability on parallel computer architectures and the increase in problem complexity and size that this implies.

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 [5]. Potential users of COYOTE are encouraged to become familiar with the present report and the simple example analyses reported in [5] before using the program.

In the following chapter the initial-boundary value problems treated by COYOTE 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 proce-

dures 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 surfaces of conducting regions. The coupling of COYOTE to other mechanics codes, in a step to step manner, also allows phenomena other than conduction to be simulated. The current version of the code allows one or two additional, nonlinear diffusion equations to be defined and coupled to the thermal problem.

In the following section, the equation describing the basic heat conduction problem will be outlined along with the limiting assumptions used in developing COYOTE. A subsequent section 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. Other sections define the multiple auxiliary diffusion equations that may be added to the thermal problem. The theoretical development in each section will treat the general three-dimensional problem since the two-dimensional (plane or axisymmetric) case 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,

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

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 principal 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. When the conduction process is coupled to other physical phenomena (*e.g.*, mechanical deformation) then the material properties may be functions of all other relevant field variables.

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 section.

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, $\Omega(t)$. Because 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 and consideration of density changes in the material. In the present formulation, deformation data are assumed to be supplied from an external source; mass conservation, if required, is accounted for within the code. 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 section.

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

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

All of the conditions stated above for Equation (2.1) also pertain to Equation (2.2). Since a

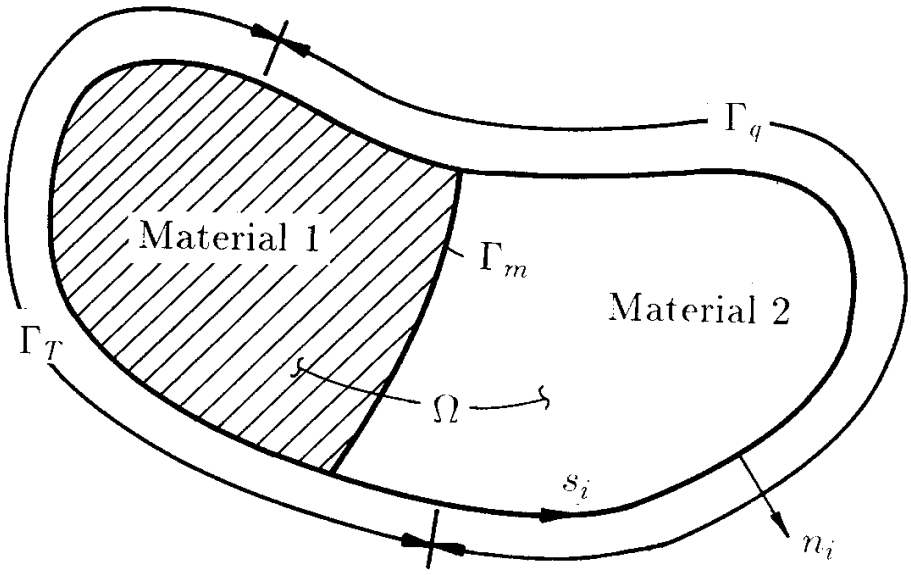

Figure 2.1: Schematic for boundary condition definitions.

momentum equation is not considered in the present formulation it is assumed that the velocity field is completely prescribed as a function of time and space; the nonconservative form of the energy equation also requires that the velocity field be divergence free. 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 permissable.

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

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

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

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

$$q_c = h_c(s_i, T, t)(T - T_c) \tag{2.5}$$

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

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 2.4.

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

$$T|_{\Gamma_m^+} = T|_{\Gamma_m^-} \tag{2.7}$$

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

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

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

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

$$T_f|_{\Gamma_m} = T_s|_{\Gamma_m} \tag{2.10}$$

$$k_f \frac{\partial T}{\partial n}\bigg|_{\Gamma_m} - k_s \frac{\partial T}{\partial n}\bigg|_{\Gamma_m} = \rho L \frac{\partial \Gamma_m}{\partial t} \tag{2.11}$$

where L is the latent heat and $\Gamma_m(t)$ is the unknown spatial position of the phase boundary. The subscripts 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

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

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

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

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 because 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 [9-12] and in Section 5.7. Other methods for including phase change effects, such as the heat source method or chemical kinetics approach [13], are possible and could be included in COYOTE with minor code modifications.

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 Bulk Nodes

The environment external to the continuum region modeled by COYOTE, influences the thermal diffusion process through the flux boundary conditions given in (2.5) and (2.6) and in particular through the specification of the reference temperatures, T_c and T_r. In some cases, a zero-dimensional model for the external region is useful in accounting for changes in the convective reference temperature, T_c. COYOTE treats such a region in terms of a bulk node. Note that the radiation boundary condition cannot usually be included with the bulk node because of its dependence on geometry.

A bulk node is characterized by a single temperature, $T_b(t)$, and pressure, $P_b(t)$ and is defined as a general control volume, CV, bounded by a control surface, CS. Mass and energy may flow across the control surface and the control volume may be time-dependent. Processes occurring within the bulk node are assumed to be in quasi-equilibrium and be uniformly distributed within the CV. At present, chemical reactions within the bulk node volume are not considered. The statement of mass conservation for the CV is

$$\frac{dM}{dt} = \frac{d}{dt}\int_{CV} \rho \, dV = \sum_{CS} \delta\dot{m}_{in} - \sum_{CS} \delta\dot{m}_{out} = f_M(t) \tag{2.14}$$

where M is the total mass and the summations are over the segments of the control surface with an incremental mass flux, $\delta\dot{m}$. The energy conservation for the CV is

$$\frac{dE}{dt} = \frac{d}{dt}\int_{CV} \rho \, \mathcal{E} \, dV = \sum_{CS} (h_0 \, \delta\dot{m} + \delta q + \delta\mathcal{P})_{in} - \sum_{CS} (h_0 \, \delta\dot{m} + \delta q + \delta\mathcal{P})_{out} \tag{2.15}$$

where E is the total energy, $\mathcal{E}$ is the specific energy, $h_0 \, \delta\dot{m}$ is the mass transfer energy rate and h_0 is the total enthalpy per unit mass, q is the thermal energy rate and $\mathcal{P}$ is the mechanical energy rate. For the bulk node of interest here, the kinetic and potential energy changes in the mass transfer rate can be neglected. Also, shaft work and shear forces are neglected in the

mechanical energy rate and conduction is neglected in the thermal rate. The bulk node energy equation then is

$$\frac{dU}{dt} = P_b\dot{V} + \sum_{CS}(h_0\,\delta\dot{m} + \delta q)_{in} - \sum_{CS}(h_0\,\delta\dot{m} + \delta q)_{out} = f_U(t) \tag{2.16}$$

where P_b is the uniform pressure for the bulk node and $\dot{V}$ is the time rate of change of the bulk node volume. Also, for an ideal substance, the internal energy is $U = MC_vT_b$ which allows the recovery of the bulk node temperature from the energy equation. For use in COYOTE, the summation over the control surface is replaced by a summation over the element surfaces bounding the bulk node. The thermal energy rates in (2.16) are replaced by Equation (2.5) as appropriate, where the reference temperature is now T_b. The mass flow terms in (2.14) and (2.16) may be imposed on the bulk node to model vents or orfices associated with the geometry or may be connected to the chemical decomposition of the materials surrounding the bulk node. The decomposition reaction may add mass and energy to the bulk node during the reaction and/or contribute to the bulk node through the removal of elements (element death). Element removal also leads to changes in the volume of the bulk node as does the Lagrangian motion of material surrounding the bulk node. The bulk node material is modeled as either an ideal gas or a constant pressure liquid. The mass and energy equations for the bulk node are decoupled from the finite element equations and are integrated in time with any of several time integration methods. For closed cavities, a simple algebraic equation is used to update the time-dependent volume, since the volume change is either prescribed or is incremental. Open cavities must have a representative volume specified.

In time independent applications, the bulk node equations reduce to a surface energy balance for the control surface surrounding the bulk node. A simple summation (surface integration) of the control surface boundary condition given by (2.5) leads to a single energy equation for the bulk node temperature. This equation is solved decoupled from the finite element equations and thus requires an iterative solution procedure to equilibrate the bulk node temperature with the temperature dependent, surface boundary conditions. A bulk node may be bounded by a radiation enclosure, but its temperature is determined solely by satisfying Equation (2.5), since the enclosure msut be transparent to any radiation considered.

2.4 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 may be 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.

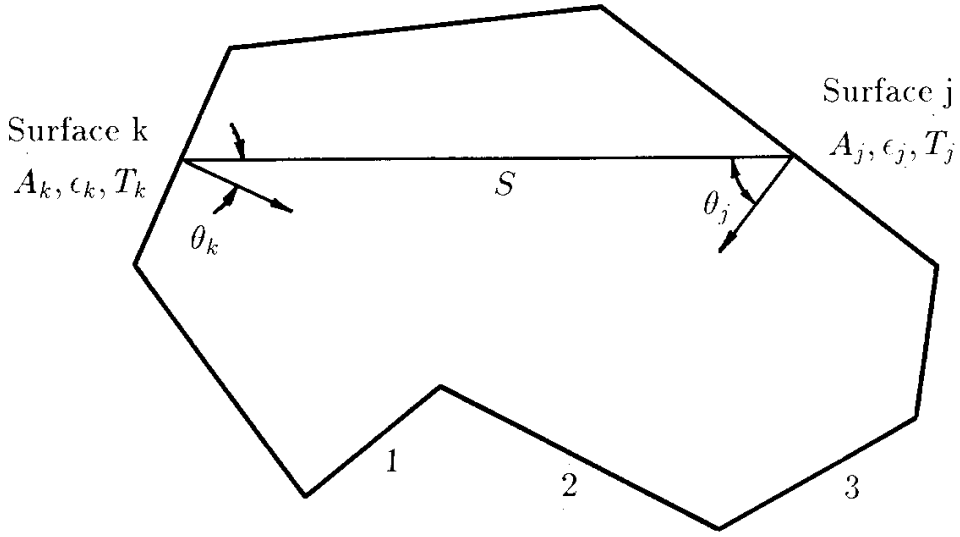

Figure 2.2: Nomenclature for enclosure radiation.

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 is assumed to 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-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, k, in the enclosure leads to the following system of equations

$$\sum_{j=1}^{N} \left[\frac{\delta_{kj}}{\epsilon_j} - F_{k-j} \left(\frac{1-\epsilon_j}{\epsilon_j} \right) \right] \frac{Q_j}{A_j} = \sum_{j=1}^{N} \left(\delta_{kj} - F_{k-j} \right) \sigma {T_j}^4. \tag{2.17}$$

Equation (2.17) relates the net radiation energy loss, Q_j, from each surface to the surface temperatures, 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 and an unobstructed view of each other, the view factors are defined by

$$F_{k-j} = \frac{1}{A_k} \int_{A_k} \int_{A_j} \frac{\cos\theta_k \cos\theta_j}{\pi S^2} \, dA_j \, dA_k \tag{2.18}$$

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 (2.18) 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}, including more general geometries with obstructions, will be outlined in a later chapter.

For purposes of computation it is convenient to rearrange (2.17) into the following series of equations

$$\sum_{j=1}^{N} \left[\delta_{kj} - (1-\epsilon_k)F_{k-j}\right] q_j^o = \epsilon_k \sigma T_k^4 \tag{2.19}$$

and

$$q_k = q_k^o - \sum_{j=1}^{N} F_{k-j} q_j^o. \tag{2.20}$$

Equations (2.19) and (2.20) 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 (2.19) can be solved for the outgoing radiative flux at each surface. Equation (2.20) 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 (2.19) and (2.20) in a finite element context will be discussed in Section 5.5.

2.5 Chemical Kinetics

The thermal diffusion problem outlined in Sections 2.1-2.4 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. COYOTE 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

$$\sum_{i=1}^{I} \nu'_{ij} \mathcal{M}_i \rightarrow \sum_{i=1}^{I} \nu''_{ij} \mathcal{M}_i \quad for \quad j = 1, 2, \ldots, J \tag{2.21}$$

where ν'_{ij}, ν''_{ij} are stoichiometric coefficients (usually integer values) and $\mathcal{M}_i$ is the chemical symbol for the ith 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 usually defined in the form:

$$r_j = k_j(T) \prod_{i=1}^{I} [N_i]^{\mu_{ij}} \quad for \quad j = 1, 2, \ldots, J \tag{2.22}$$

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

$$k_j(T) = T^{\beta_j} \, A_j \, exp(-E_j/RT) \tag{2.23}$$

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

$$\frac{d}{dt}[N_i] = \sum_{j=1}^{J} \nu_{ij} r_j \quad for \quad i = 1, 2, \ldots, I \tag{2.24}$$

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

$$Q_r = \sum_{j=1}^{J} q_j r_j \tag{2.25}$$

where q_j represents the known endothermic or exothermic energy release for reaction step j. Though the reaction rate expression in (2.22) is standard, it is not universal. COYOTE also permits reaction rates of an arbitrary form to be incorporated into the kinetics description through a user-defined subroutine. Also, chemical species may be defined that are algebraically related to other species and are not updated through the kinetics relations. These definitions are implemented through a user subroutine.

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

$$(\rho C)_{mix} = \sum_{i=1}^{I} [N_i](\rho C)_i \tag{2.26}$$

$$(k_{jk})_{mix} = \sum_{i=1}^{I} [N_i](k_{jk})_i \tag{2.27}$$

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

$$F_c = \frac{(1.0 - X_c)\sum_{i=1}^{I}[N_i](g)_i}{\sum_{i=1}^{I}[N_i]} \tag{2.28}$$

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 (2.24) 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.

2.6 Auxiliary Diffusion Equations

To generalize the application of the COYOTE code, one or two additional diffusion equations can be defined and coupled to the thermal problem. For the material region Ω, the additional time dependent diffusion processes are described by

$$\rho\, C^k \frac{\partial \phi^k}{\partial t} = \frac{\partial}{\partial x_i}\left(D_{ij}^k \frac{\partial \phi^k}{\partial x_j} \right) + Q^k \tag{2.29}$$

where the superscript k varies from A to B and denotes the particular diffusion process. The parameter C^k is an effective capacitance (that may or may not be required), D_{ij}^k is an anisotropic conductivity or diffusivity tensor and Q^k is the volume source. Note that these properties may be functions of spatial location, time and the variable ϕ^k. The dependent variable ϕ^k is defined by the particular application but is generally a function of position and time.

As a diffusion process, the partial differential equations in (2.29) require the same types of boundary conditions as found in the conduction equation. Analogous to (2.3) and (2.4) the appropriate boundary conditions are

$$\phi^k = f^{\phi^k}(s_i, t) \qquad on \quad \Gamma_{\phi^k} \tag{2.30}$$

$$\left(D_{ij}^k \frac{\partial \phi^k}{\partial x_j} \right) n_i + q_c^{\phi^k} = f^{q^{\phi^k}}(s_i, t) \qquad on \quad \Gamma_q^{\phi^k}. \tag{2.31}$$

In Equations (2.30) and (2.31) the functions f^{ϕ^k} and $f^{q^{\phi^k}}$ are specified values of the known dependent variables and variable fluxes. The convective component of the flux is provided by

$$q_c^{\phi^k} = h_c^{\phi^k}(s_i, \phi^k, t)(\phi^k - \phi_c^k) \tag{2.32}$$

where $h_c^{\phi^k}$ is a convection coefficient and ϕ_c^k is the reference value of the variable ϕ^k. Note that there is no radiation component for the general flux. If ϕ^k is used as a temperature variable, a radiation component for the flux would be defined. Conditions for interfaces and contact may also be defined for the auxiliary diffusion processes and are directly analogous to the conduction definitions.

The properties and boundary conditions for (2.29) were initially defined to be functions of spatial location, time and the dependent variable ϕ^k. To permit coupled diffusion processes, the material properties, sources and boundary conditions are also allowed to be functions of the temperature. Conversely, the thermal properties, sources and boundary conditions are allowed to be functions of ϕ^k. For complete generality, functional dependencies on variable derivatives should be considered. COYOTE does not generally allow this option, though its implementation is relatively straightforward.

When the motion of the material is described in a fixed Eulerian coordinate system, the auxiliary diffusion equations, if needed, must be modified to include the advective transport term. Following the form of the energy equation in (2.2) the advection-diffusion equations for the ϕ^k variables are

$$\rho\, C^k \left(\frac{\partial \phi^k}{\partial t} + u_j \frac{\partial \phi^k}{\partial x_j} \right) = \frac{\partial}{\partial x_i} \left(D_{ij}^k \frac{\partial \phi^k}{\partial x_j} \right) + Q^k. \tag{2.33}$$

The material velocity components u_j must again be divergence free. All of the boundary conditions noted above for the auxiliary diffusion equations are applicable to the advection-diffusion form of the equation.

2.7 Periodic Heat Conduction

The general heat conduction Equation (2.1) is applicable to any type of time varying diffusion problem. A special form of the conduction problem occurs when the boundary conditions and/or source term is periodic in time. Let the temperature and source be represented as periodic functions given by

$$T(x_i, t) = T_0(x_i) e^{i\omega t} = (T_R + iT_I) e^{i\omega t} \tag{2.34}$$

$$Q(x_i, t) = Q_0(x_i) e^{i\omega t} = (Q_R + iQ_I) e^{i\omega t} \tag{2.35}$$

where

$$e^{i\omega t} = \cos(\omega t) + i \sin(\omega t) \tag{2.36}$$

and the subscript 0 refers to the amplitude function (which is complex) and the subscripts R and I indicate real and imaginary components. The amplitudes are $T_0 = T_R + iT_I$ and $Q_0 = Q_R + iQ_I$. The circular frequency is $\omega = 2\pi f$ and the modulus and phase angle are

$$|T| = \sqrt{T_R^2 + T_I^2} \tag{2.37}$$

$$\beta = \tan^{-1}\left(\frac{T_I}{T_R}\right) \tag{2.38}$$

With these definitions the temperature can be also rewritten as

$$T(x_i, t) = |T(x_i)| \cos(\omega t + \beta) \tag{2.39}$$

Substituting (2.34) and (2.35) into the conduction equation leads to a time independent diffusion equation

$$i\omega\rho\, C T_0 = \frac{\partial}{\partial x_i}\left(k_{ij}\frac{\partial T_0}{\partial x_j}\right) + Q_0 \tag{2.40}$$

where the common exponential factor has been dropped from the equation. The conductivity and capacitance are assumed to be independent of temperature in this formulation. By considering the real and imaginary components separately, the conduction problem expands to a set of coupled (real) equations for the components with the form

$$-\omega\rho\, C T_I = \frac{\partial}{\partial x_i}\left(k_{ij}\frac{\partial T_R}{\partial x_j}\right) + Q_R \tag{2.41}$$

$$\omega\rho\, C T_R = \frac{\partial}{\partial x_i}\left(k_{ij}\frac{\partial T_I}{\partial x_j}\right) + Q_I \tag{2.42}$$

The boundary conditions must also be defined in terms of the periodic functions. For a specified temperature, the individual real and imaginary components may be defined

$$T_R = f_R^T(s_i) \qquad T_I = f_I^T(s_i) \qquad on \quad \Gamma_T \tag{2.43}$$

where the boundary condition phase and modulus are defined by (2.38) and (2.37). The heat flux boundary condition must also be periodic which when written by component is

$$\left(k_{ij}\frac{\partial T_R}{\partial x_j}\right) n_i + h_c(s_i)(T_R - T_{c_R}) = f_R^q(s_i) \qquad on \quad \Gamma_q. \tag{2.44}$$

$$\left(k_{ij}\frac{\partial T_I}{\partial x_j}\right) n_i + h_c(s_i)(T_I - T_{c_I}) = f_I^q(s_i) \qquad . \tag{2.45}$$

The reference temperature for convection has been assumed to be periodic with the same frequency as the temperature. Note that the radiation component has been neglected. Generally the heat transfer coefficients must be independent of temperature, though some types of averaging over the period may allow the inclusion of some nonlinearity.

The above equations define a periodic form of the boundary value problem for conduction. This continuum problem can be cast in a finite element form to allow the computation of the spatially varying temperature field; the time variation may be reconstructed from the periodic relation given in (2.34) or (2.39).

2.8 Front Tracking

In concert with the modeling of reactive materials, it may be necessary to follow the evolution of a material interface, such as a reaction front. For an Eulerian representation of such a problem, several methods exist to represent the interface and track its time-dependent motion. COYOTE currently employs a level set algorithm [14] for this problem that is based on the solution of the scalar advection equation

$$\frac{\partial f}{\partial t} + u_j \frac{\partial f}{\partial x_j} = 0. \tag{2.46}$$

In (2.46) the variable f represents the level set function which is usually defined as the signed distance function; the interface Γ is the zero level set of f. That is,

$$f(x_i, t) = \begin{cases} < 0 & \text{if } x_i \in \text{unreacted material} \\ 0 & \text{if } x_i \in \Gamma \\ > 0 & \text{if } x_i \in \text{reacted material} \end{cases}$$

The velocity of the interface, u_j, must be specified and is always oriented normal to the interface. The velocity would typically be a function of the field variables such as temperature and/or species. It is generally assumed that the interface represents a steep or discontinuous transition in material properties and behavior and Equation (2.46) describes the nondissipative motion of the front.

Since (2.46) is a hyperbolic equation, initial conditions must be specified for f and this would normally be a uniform field with $f < 0$. The initial appearance or specification of a front (*e.g.*,$f = 0$) would typically be generated at one or more locations within the domain as the reactive process reached a critical value in temperature or species. With a front defined and a velocity field specified, Equation (2.46) is then integrated in time to follow the motion of the interface. The numerical methods used to solve (2.46) in conjunction with the conduction problem are 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

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

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. Details of this process may be found in

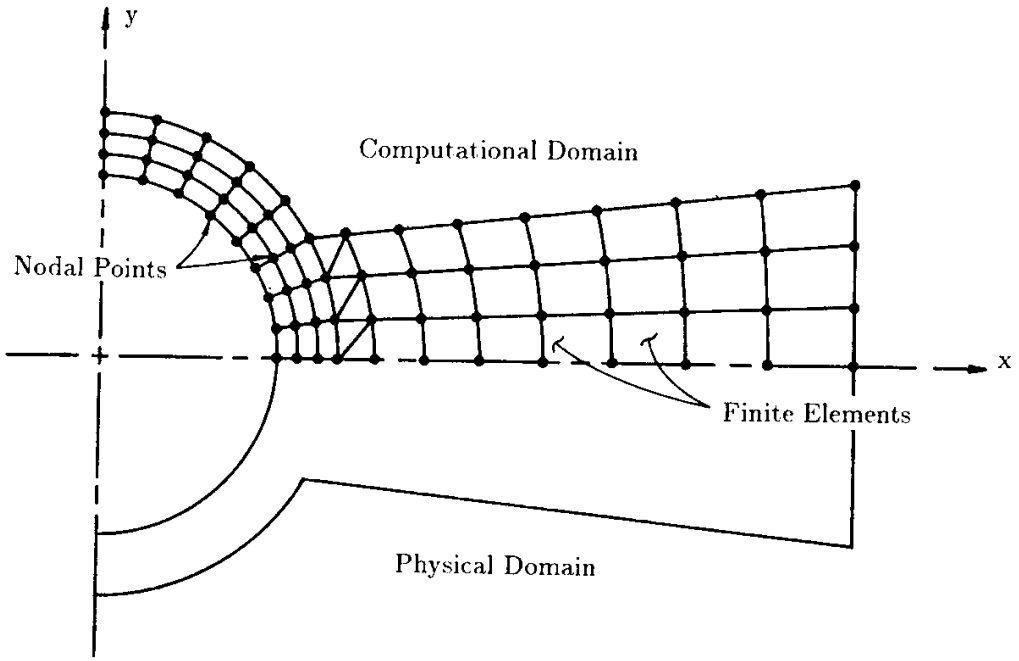

Figure 3.1: Finite element discretization of a region.

[13,15,16]. The global temperature field has a form similar to (3.1) and is expressed as

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

or in matrix notation,

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

where $\mathbf{\Theta}$ 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

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

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,

$$\langle \mathbf{\Theta}, f_T \rangle = \langle \mathbf{\Theta}, R_T \rangle = 0 \tag{3.5}$$

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

$$\langle a, b \rangle = \int_\Omega a \cdot b \, d\Omega \tag{3.6}$$

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

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

As is standard practice [13,15], 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.

$$\int_\Omega \rho C \mathbf{\Theta}\mathbf{\Theta}^T \frac{\partial \mathbf{T}}{\partial t}\, d\Omega + \int_\Omega \frac{\partial \mathbf{\Theta}}{\partial x_i}\left(k_{ij} \frac{\partial \mathbf{\Theta}^T}{\partial x_j}\mathbf{T}\right)\, d\Omega =$$

$$\int_\Omega \mathbf{\Theta} Q\, d\Omega + \int_\Gamma \mathbf{\Theta}\left(k_{ij} \frac{\partial \mathbf{\Theta}^T}{\partial x_j}\mathbf{T}\right) n_i\, d\Gamma. \tag{3.8}$$

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

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

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

$$\mathbf{M}(\mathbf{T})\dot{\mathbf{T}} + \mathbf{K}(\mathbf{T})\mathbf{T} = \mathbf{F_Q}(\mathbf{T}) + \mathbf{F}(\mathbf{T}) \tag{3.10}$$

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

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

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 [13,15]. 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 [13,15,16], 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

$$\mathbf{M} = \sum_{\mathbf{e}} \mathbf{M_e} \quad ; \quad \mathbf{K} = \sum_{\mathbf{e}} \mathbf{K_e} \quad ; \quad \mathbf{F} = \sum_{\mathbf{e}} \mathbf{F_e} \tag{3.12}$$

The sums in (3.12) are sometimes represented in terms of an assembly operator that includes the notion of a connectivity. In either case, the assemlby or sum is over all the elements in the domain and the element matrices are defined by

$$\begin{aligned} \mathbf{M_e} &= \int_{\Omega e} \rho C \mathbf{\Theta_e} \mathbf{\Theta_e}^T d\Omega \\ \mathbf{K_e} &= \int_{\Omega e} \frac{\partial \mathbf{\Theta_e}}{\partial x_i} k_{ij} \frac{\partial \mathbf{\Theta_e}^T}{\partial x_j} d\Omega \\ \mathbf{F_{Qe}} &= \int_{\Omega e} \mathbf{\Theta_e} Q d\Omega \\ \mathbf{F_e} &= \int_{\Gamma e} \mathbf{\Theta_e} (f^q - q_c - q_r) d\Gamma \end{aligned} \tag{3.13}$$

Once the form of the element interpolation function, $\mathbf{\Theta_e}$, 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 section 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)

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

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

$$u_j(x_i, t) = \mathbf{\Phi}^T(x_i)\mathbf{u_j}(t). \tag{3.14}$$

For generality, the interpolation function $\mathbf{\Phi}$ 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

$$f_T(\mathbf{\Theta}, \mathbf{\Phi}, \mathbf{u_j}, \mathbf{T}) = R_T. \tag{3.15}$$

Applying the Galerkin method with weight function $\mathbf{\Theta}$ produces

$$\int_\Omega \rho C \mathbf{\Theta}\mathbf{\Theta}^T \frac{\partial \mathbf{T}}{\partial t}\, d\Omega + \int_\Omega \rho C \mathbf{\Theta}\mathbf{\Phi}^T \mathbf{u_j} \frac{\partial \mathbf{\Theta}^T}{\partial x_j}\mathbf{T}\, d\Omega + \int_\Omega \frac{\partial \mathbf{\Theta}}{\partial x_i}\left(k_{ij}\frac{\partial \mathbf{\Theta}^T}{\partial x_j}\mathbf{T}\right) d\Omega =$$

$$\int_\Omega \mathbf{\Theta} Q\, d\Omega + \int_\Gamma \mathbf{\Theta}\left(f^q - q_c - q_r\right)\, d\Gamma \tag{3.16}$$

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

$$\mathbf{M}(\mathbf{T})\dot{\mathbf{T}} + \mathbf{C}(\mathbf{u_j}, \mathbf{T})\mathbf{T} + \mathbf{K}(\mathbf{T})\mathbf{T} = \mathbf{F_Q}(\mathbf{T}) + \mathbf{F}(\mathbf{T}) \tag{3.17}$$

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

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

while the element level matrix is

$$\mathbf{C_e} = \int_{\Omega_e} \rho C \mathbf{\Theta_e}\mathbf{\Phi_e}^T \mathbf{u_j} \frac{\partial \mathbf{\Theta_e}^T}{\partial x_j}\, d\Omega. \tag{3.19}$$

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 divergence free 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.

3.3 Auxiliary Diffusion Equations

The finite element form of the auxiliary diffusion equations is derived with the same procedure as used for the conduction equation. The dependent variables are represented by

$$\phi^k(x_i, t) = \mathbf{\Upsilon}^T(x_i)\boldsymbol{\phi}^{\mathbf{k}}(t)$$

where $\mathbf{\Upsilon}$ is a vector of basis functions and $\boldsymbol{\phi}^{\mathbf{k}}$ is the vector of nodal point values of the unknowns. Using the Galerkin form of the method of weighted residuals leads to

$$\int_\Omega \rho C^k \mathbf{\Upsilon}\mathbf{\Upsilon}^T \frac{\partial \boldsymbol{\phi}^{\mathbf{k}}}{\partial t}\, d\Omega + \int_\Omega \frac{\partial \mathbf{\Upsilon}}{\partial x_i}\left(D_{ij}^k \frac{\partial \mathbf{\Upsilon}^T}{\partial x_j}\boldsymbol{\phi}^{\mathbf{k}}\right) d\Omega = \int_\Omega \mathbf{\Upsilon} Q^k\, d\Omega + \int_\Gamma \mathbf{\Upsilon}\left(f^{q^{\phi^k}} - q_c^{\phi^k}\right) d\Gamma \tag{3.20}$$

where the diffusion term has been transformed by integration-by-parts and the divergence theorem; the boundary term is rewritten in terms of the specified natural boundary conditions. This equation is recognized as a matrix equation of the following form

$$\mathbf{M^k}(\boldsymbol{\phi^k})\dot{\boldsymbol{\phi}}^{\mathbf{k}} + \mathbf{K^k}(\boldsymbol{\phi^k})\boldsymbol{\phi^k} = \mathbf{F^k_Q}(\boldsymbol{\phi^k}) + \mathbf{F^k}(\boldsymbol{\phi^k}) \tag{3.21}$$

where the correspondence with the energy equations is obvious. In practice the shape functions $\boldsymbol{\Upsilon}$ are taken to be the same as the temperature interpolation functions, *i.e.* $\boldsymbol{\Upsilon} = \boldsymbol{\Theta}$.

The advection-diffusion form of the auxiliary equations is derived with the same procedure and results in a matrix equation that is of the same form as (3.17 or

$$\mathbf{M^k}(\boldsymbol{\phi^k})\dot{\boldsymbol{\phi}}^{\mathbf{k}} + \mathbf{C^k}(\mathbf{u_j}, \boldsymbol{\phi^k})\boldsymbol{\phi^k} + \mathbf{K^k}(\boldsymbol{\phi^k})\boldsymbol{\phi^k} = \mathbf{F^k_Q}(\boldsymbol{\phi^k}) + \mathbf{F^k}(\boldsymbol{\phi^k}) \tag{3.22}$$

Note that in writing (3.21) and (3.22) all of the possible functional dependencies have not been shown. In particular, the matrices and vectors (properties and boundary conditions) may be functions of the temperature when coupled to the energy equation.

3.4 Periodic Heat Conduction

The special case of periodic heat conduction is described by the complex partial differential equation given in (2.40). The finite element form of this equation is most conveniently derived using the component equations listed in (2.41) and (2.42) and the same weighted residual process outlined above. The real and imaginary components of the temperature are defined by the shape functions

$$T_R(x_i) = \boldsymbol{\Theta}^T(x_i)\mathbf{T_R} \;\; ; \;\; T_I(x_i) = \boldsymbol{\Theta}^T(x_i)\mathbf{T_I}$$

Using a Galerkin method with the component equations produces matrix equations of the form

$$-\omega\mathbf{M}\mathbf{T_I} + \mathbf{K}\mathbf{T_R} = \mathbf{F_{Q_R}} + \mathbf{F_R} \tag{3.23}$$

$$\omega\mathbf{M}\mathbf{T_R} + \mathbf{K}\mathbf{T_I} = \mathbf{F_{Q_I}} + \mathbf{F_I} \tag{3.24}$$

where the individual matrices and vectors are linear versions of the standard conduction versions

$$\begin{aligned}
\mathbf{M} &= \int_\Omega \rho C \boldsymbol{\Theta}\boldsymbol{\Theta}^T \, d\Omega \\
\mathbf{K} &= \int_\Omega k_{ij} \frac{\partial \boldsymbol{\Theta}}{\partial x_i} \frac{\partial \boldsymbol{\Theta}^T}{\partial x_j} \, d\Omega \\
\mathbf{F_{Q_R}} &= \int_\Omega \boldsymbol{\Theta} Q_R \, d\Omega \\
\mathbf{F_{Q_I}} &= \int_\Omega \boldsymbol{\Theta} Q_I \, d\Omega \\
\mathbf{F_R} &= \int_\Gamma \boldsymbol{\Theta} \left(f^{q_R} - h_c(\boldsymbol{\Theta}^T\mathbf{T_R} - T_{c_R}) \right) d\Gamma
\end{aligned} \tag{3.25}$$

$$\mathbf{F_I} = \int_\Gamma \mathbf{\Theta} \left(f^{q_I} - h_c(\mathbf{\Theta}^T \mathbf{T_I} - T_{c_I} \right) d\Gamma$$

The component Equations (3.23) and (3.24) represent a coupled system of equations for the real and imaginary components of the temperature in a time periodic problem. Material properties and boundary condition parameters may be functions of spatial location but may not depend on temperature.

3.5 Front Tracking Equation

The tracking of a material interface is provided by developing a discretized form of the level set equation given in (2.46). The finite element form of this equation is derived in the same manner as the convection equation with a similar result.

The front tracking or level set variable f is represented by an expansion of the form

$$f(x_i, t) = \mathbf{\Psi}^T(x_i)\mathbf{f}(t) \tag{3.26}$$

and the known, divergence free velocity field is represented by a similar interpolation given again by (3.14)

$$u_j(x_i, t) = \mathbf{\Phi}^T(x_i)\mathbf{u_j}(t).$$

Using these definitions in (2.46) and the standard weighted residual form, a matrix equation for the front tracking variable can be written as

$$\hat{\mathbf{M}}\dot{\mathbf{f}} + \hat{\mathbf{C}}(\mathbf{u_j})\mathbf{f} = \mathbf{0} \tag{3.27}$$

where the mass and advection matrices are defined by

$$\hat{\mathbf{M}} = \int_\Omega \mathbf{\Psi}\mathbf{\Psi}^T \, d\Omega \tag{3.28}$$

and

$$\hat{\mathbf{C}}(\mathbf{u_j}) = \int_\Omega \mathbf{\Psi}\mathbf{\Phi}^T \mathbf{u_j} \frac{\partial \mathbf{\Psi}^T}{\partial x_j} \, d\Omega \tag{3.29}$$

As with the convection equation this system is unsymmetric and requires the specification of a velocity field. Also, the front tracking equation is only available with the continuum elements in the element library. The shape functions denoted by $\mathbf{\Psi}$ are usually the same functions as $\mathbf{\Phi}$ and $\mathbf{\Theta}$.

Chapter 4

Elements and Element Matrix Construction

The formulation of the equations for an individual element, as indicated by Equations (3.10), (3.12), (3.13) and (3.19), 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 [13,16] 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.* $\mathbf{\Phi} = \mathbf{\Theta}$.

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

$$\mathbf{\Theta_l} = \left\{ \begin{array}{c} L_1 \\ L_2 \\ L_3 \end{array} \right\} \tag{4.1}$$

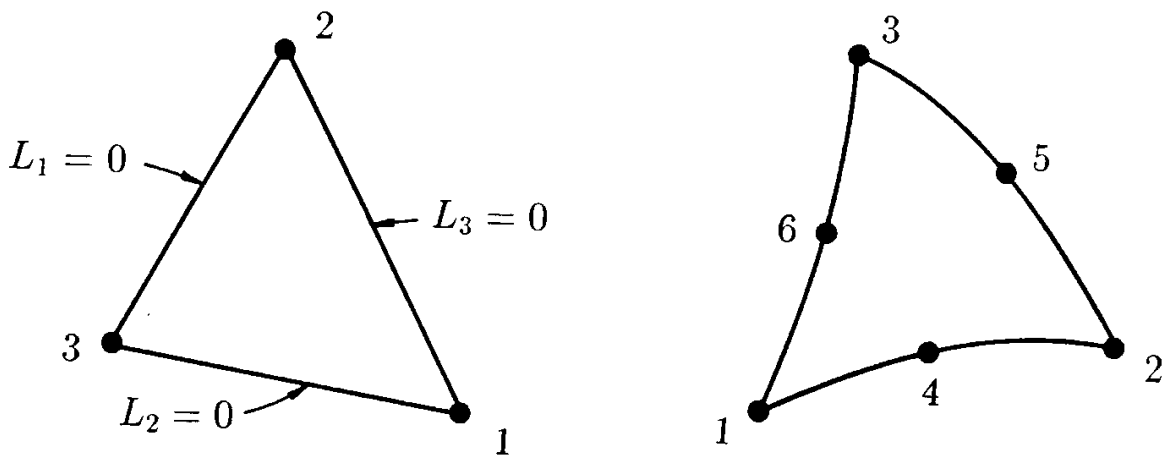

Figure 4.1: Two-dimensional triangular elements.

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

$$\mathbf{\Theta_q} = \left\{ \begin{array}{c} L_1(2L_1 - 1) \\ L_2(2L_2 - 1) \\ L_3(2L_3 - 1) \\ 4L_1L_2 \\ 4L_2L_3 \\ 4L_3L_1 \end{array} \right\}. \tag{4.2}$$

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 [13,16] 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.* [17]. That is, the coordinate transformation is given by

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

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 $\mathbf{\Upsilon} = \mathbf{\Theta_l}$, a linear interpolation of the element boundary is

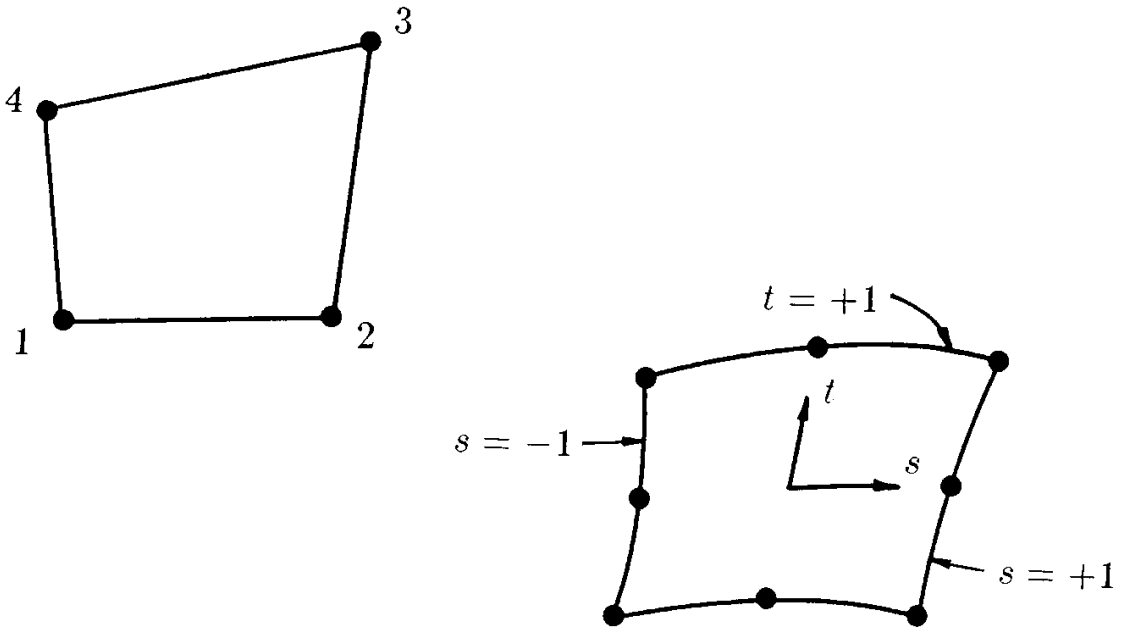

Figure 4.2: Two-dimensional quadrilateral elements.

possible. When $\boldsymbol{\Upsilon} = \boldsymbol{\Theta}_{\mathbf{q}}$, 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

$$\boldsymbol{\Theta}_{\mathbf{l}} = \left\{ \begin{array}{l} 1/4(1-s)(1-t) \\ 1/4(1+s)(1-t) \\ 1/4(1+s)(1+t) \\ 1/4(1-s)(1+t) \end{array} \right\}. \tag{4.4}$$

The ordering of the functions in (4.4) corresponds to the nodal point ordering shown in Fig-

ure 4.2. 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 [13,16] given by

$$\mathbf{\Theta_q} = \left\{ \begin{array}{c} 1/4(1-s)(1-t)(-s-t-1) \\ 1/4(1+s)(1-t)(s-t-1) \\ 1/4(1+s)(1+t)(s+t-1) \\ 1/4(1-s)(1+t)(-s+t-1) \\ 1/2(1-s^2)(1-t) \\ 1/2(1+s)(1-t^2) \\ 1/2(1-s^2)(1+t) \\ 1/2(1-s)(1-t^2) \end{array} \right\}. \tag{4.5}$$

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

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

where $\mathbf{\Upsilon}$ 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

$$\mathbf{\Theta_l} = \left\{ \begin{array}{c} 1/8(1-s)(1-t)(1-r) \\ 1/8(1+s)(1-t)(1-r) \\ 1/8(1+s)(1+t)(1-r) \\ 1/8(1-s)(1+t)(1-r) \\ 1/8(1-s)(1-t)(1+r) \\ 1/8(1+s)(1-t)(1+r) \\ 1/8(1+s)(1+t)(1+r) \\ 1/8(1-s)(1+t)(1+r) \end{array} \right\}. \tag{4.7}$$

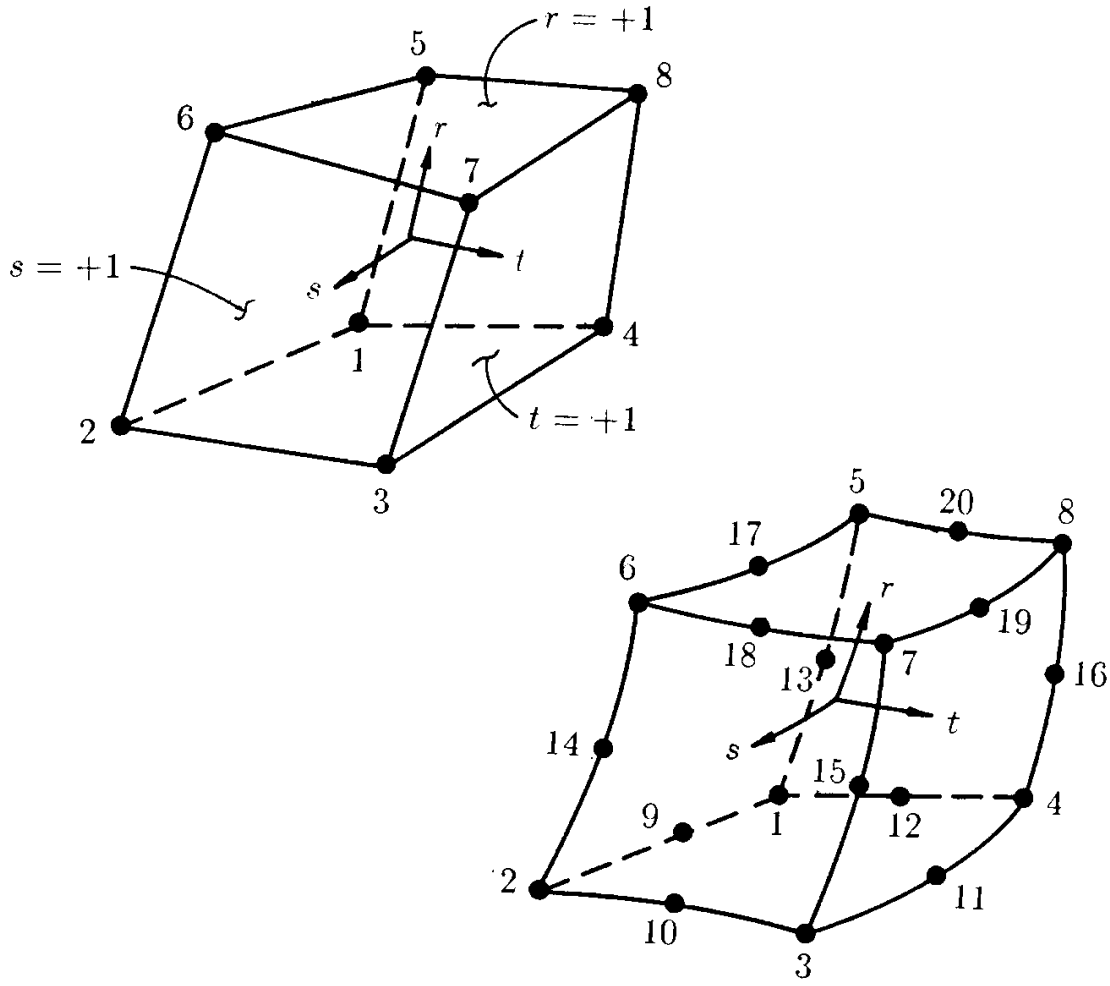

Figure 4.3: Three-dimensional brick elements.

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

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

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

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

and $\mathbf{\Upsilon}$ takes on the appropriate linear or quadratic form.

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

$$\mathbf{\Theta_l} = \left\{ \begin{array}{c} 1/2L_1(1-r) \\ 1/2L_2(1-r) \\ 1/2L_3(1-r) \\ 1/2L_1(1+r) \\ 1/2L_2(1+r) \\ 1/2L_3(1+r) \end{array} \right\} \tag{4.10}$$

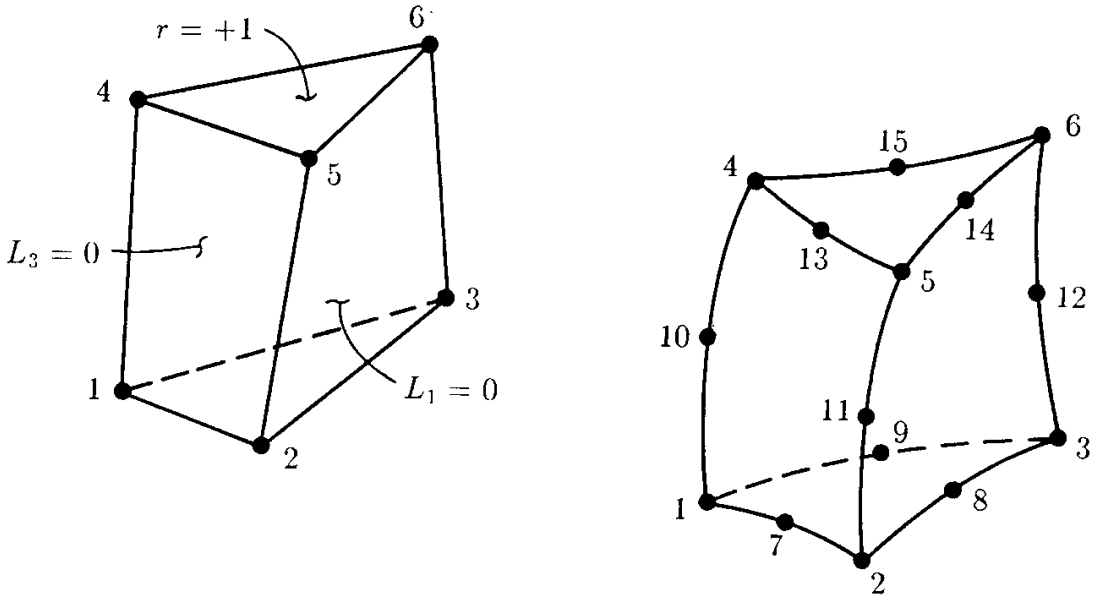

Figure 4.4: Three-dimensional prism elements.

and

$$\boldsymbol{\Theta}_{\mathbf{q}} = \left\{ \begin{array}{c} 1/2L_1[(2L_1-1)(1-r)-(1-r^2)] \\ 1/2L_2[(2L_2-1)(1-r)-(1-r^2)] \\ 1/2L_3[(2L_3-1)(1-r)-(1-r^2)] \\ 1/2L_1[(2L_1-1)(1+r)-(1-r^2)] \\ 1/2L_2[(2L_2-1)(1+r)-(1-r^2)] \\ 1/2L_3[(2L_3-1)(1+r)-(1-r^2)] \\ 2L_1L_2(1-r) \\ 2L_2L_3(1-r) \\ 2L_3L_1(1-r) \\ L_1(1-r^2) \\ L_2(1-r^2) \\ L_3(1-r^2) \\ 2L_1L_2(1+r) \\ 2L_2L_3(1+r) \\ 2L_3L_1(1+r) \end{array} \right\}. \tag{4.11}$$

The functions in (4.10) and (4.11) use area coordinates, L_i, for describing the triangular cross-section 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.

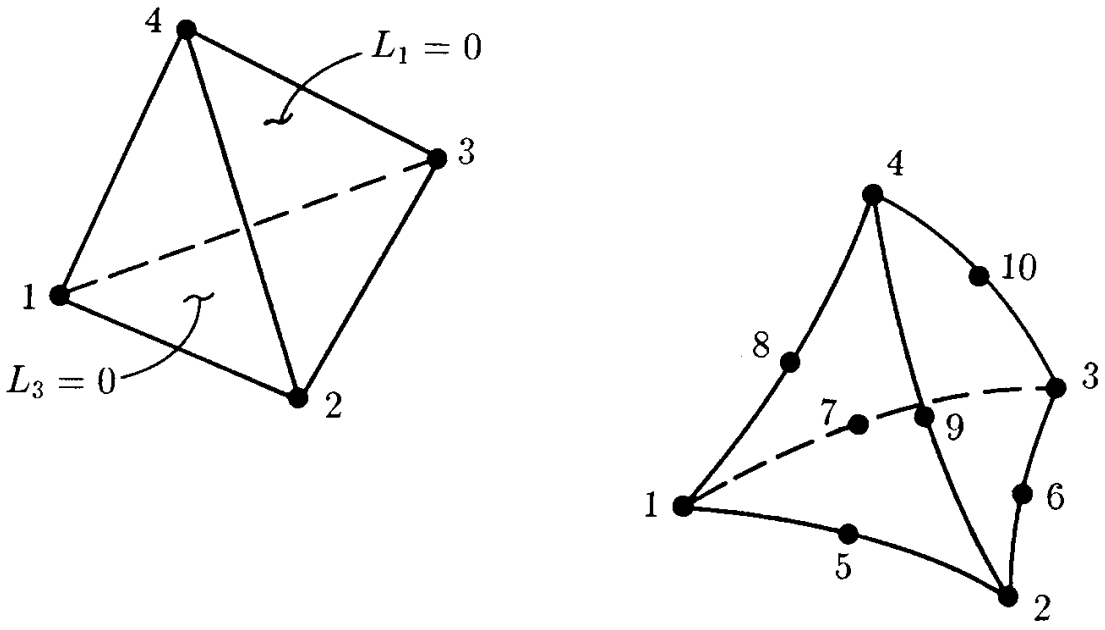

Figure 4.5: Three-dimensional tetrahedral elements.

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

$$\mathbf{\Theta_l} = \left\{ \begin{array}{c} L_1 \\ L_2 \\ L_3 \\ L_4 \end{array} \right\} \tag{4.12}$$

while the quadratic element has shape functions of the form

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

The functions in (4.12) and (4.13) are ordered as shown in the figure and are written in terms of the volume coordinates [13,16] 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-sectional area with conduction only allowed along the axis of the element. Note that the shape of the cross-section need not be explicitly defined here, though for purposes of boundary condition application, a circular cross-section is assumed. The shape function for the two-node element is defined by

$$\mathbf{\Theta_l} = \left\{ \begin{array}{c} 1/2(1-s) \\ 1/2(1+s) \end{array} \right\}. \tag{4.14}$$

and the three-node element is described by

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

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 $\mathbf{\Upsilon}$ 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-sectional 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. The two-dimensional versions of these elements are equivalent to a two-dimensional 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

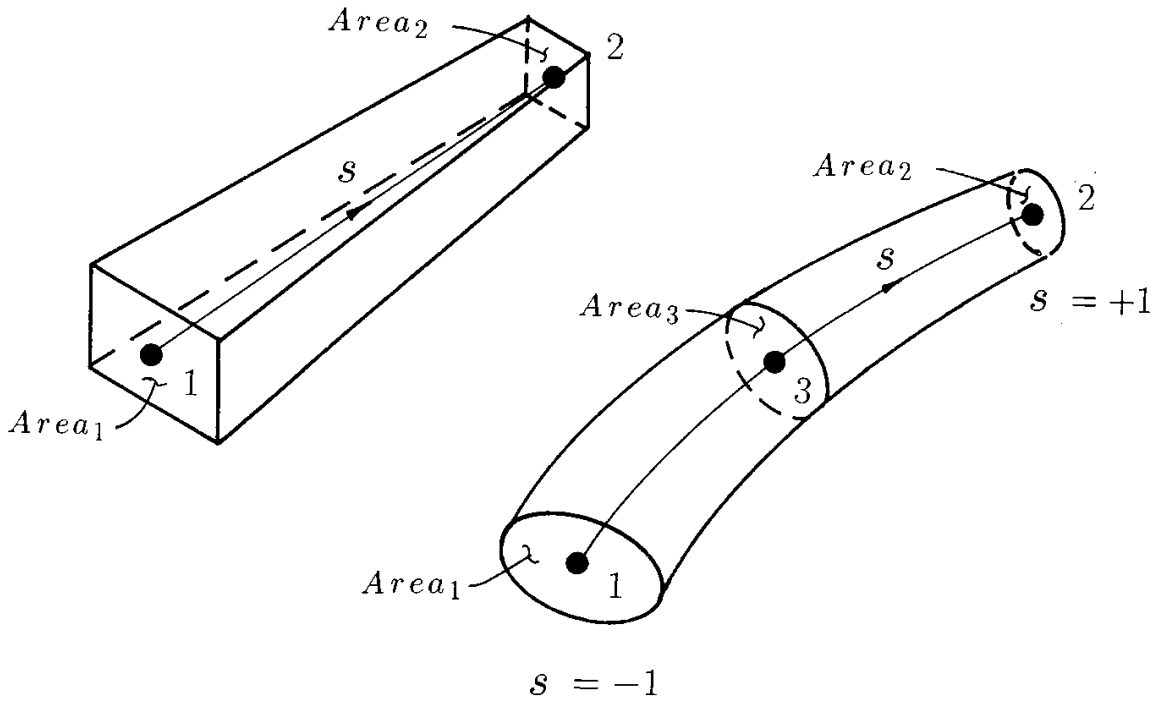

Figure 4.6: Three-dimensional bar elements.

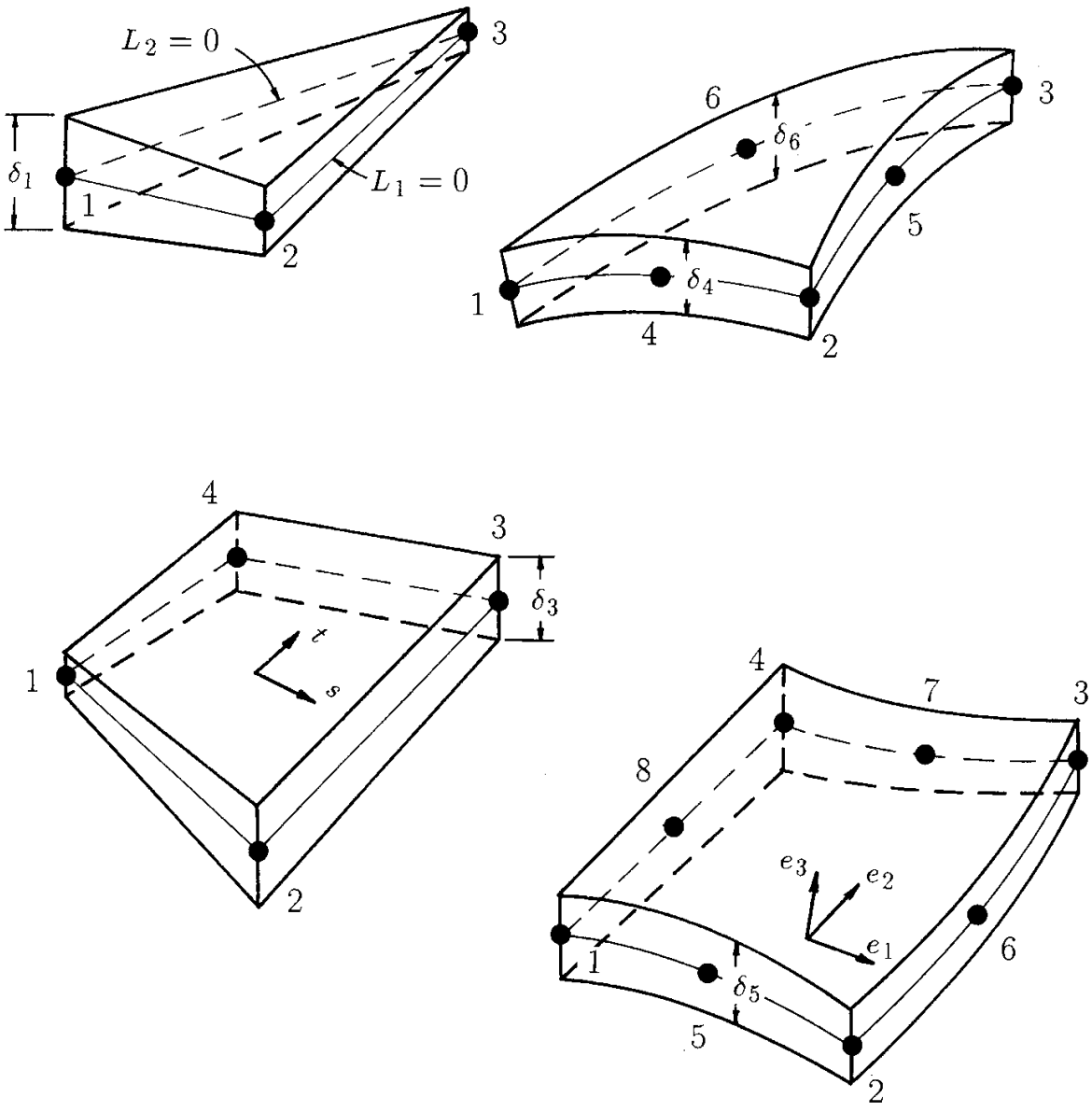

Figure 4.7: Three-dimensional shell elements.

shell thickness. These elements are shown in Figure 4.7. The temperature shape function for the three-node, triangular element is defined by

$$\mathbf{\Theta_l} = \left\{ \begin{array}{c} L_1 \\ L_2 \\ L_3 \end{array} \right\} \tag{4.16}$$

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

$$\mathbf{\Theta_q} = \left\{ \begin{array}{c} L_1(2L_1 - 1) \\ L_2(2L_2 - 1) \\ L_3(2L_3 - 1) \\ 4L_1L_2 \\ 4L_2L_3 \\ 4L_3L_1 \end{array} \right\} \tag{4.17}$$

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

$$\mathbf{\Theta_l} = \left\{ \begin{array}{c} 1/4(1-s)(1-t) \\ 1/4(1+s)(1-t) \\ 1/4(1+s)(1+t) \\ 1/4(1-s)(1+t) \end{array} \right\} \tag{4.18}$$

while the eight-node, "serendipity" shell is defined by

$$\mathbf{\Theta_q} = \left\{ \begin{array}{c} 1/4(1-s)(1-t)(-s-t-1) \\ 1/4(1+s)(1-t)(s-t-1) \\ 1/4(1+s)(1+t)(s+t-1) \\ 1/4(1-s)(1+t)(-s+t-1) \\ 1/2(1-s^2)(1-t) \\ 1/2(1+s)(1-t^2) \\ 1/2(1-s^2)(1+t) \\ 1/2(1-s)(1-t^2) \end{array} \right\} \tag{4.19}$$

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 geometric representation of the planar elements and the shell elements is quite different. The parametric mapping for any of the shell elements is accomplished with the following definitions

$$\begin{aligned} x &= \mathbf{\Upsilon}^T\mathbf{x} + r\mathbf{\Upsilon}^T\frac{\delta}{2}\mathbf{e_3}\cdot\mathbf{e_x} \\ y &= \mathbf{\Upsilon}^T\mathbf{y} + r\mathbf{\Upsilon}^T\frac{\delta}{2}\mathbf{e_3}\cdot\mathbf{e_y} \end{aligned} \tag{4.20}$$

$$z = \boldsymbol{\Upsilon}^T \mathbf{z} + r \boldsymbol{\Upsilon}^T \frac{\delta}{2} \mathbf{e_3} \cdot \mathbf{e_z}$$

where $\boldsymbol{\Upsilon}$ 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 section.

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 because the limits of integration are simple and independent of the global coordinates. The shape functions presented in the previous sections 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

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

where $\boldsymbol{\Lambda}$ represents any of the element interpolation functions (*e.g.*, $\boldsymbol{\Theta_l}, \boldsymbol{\Theta_q}$) 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 sections (*e.g.*, Equations (4.3), (4.9) or (4.20)) can be used to define the components of $\mathbf{J}$. That is,

$$\begin{aligned} J_{11} &= \frac{\partial \boldsymbol{\Upsilon}}{\partial s}^T \mathbf{x} \quad ; \quad J_{12} = \frac{\partial \boldsymbol{\Upsilon}}{\partial s}^T \mathbf{y} \quad ; \quad J_{13} = \frac{\partial \boldsymbol{\Upsilon}}{\partial s}^T \mathbf{z} \\ J_{21} &= \frac{\partial \boldsymbol{\Upsilon}}{\partial t}^T \mathbf{x} \quad ; \quad J_{22} = \frac{\partial \boldsymbol{\Upsilon}}{\partial t}^T \mathbf{y} \quad ; \quad J_{23} = \frac{\partial \boldsymbol{\Upsilon}}{\partial t}^T \mathbf{z} \\ J_{31} &= \frac{\partial \boldsymbol{\Upsilon}}{\partial r}^T \mathbf{x} \quad ; \quad J_{32} = \frac{\partial \boldsymbol{\Upsilon}}{\partial r}^T \mathbf{y} \quad ; \quad J_{33} = \frac{\partial \boldsymbol{\Upsilon}}{\partial r}^T \mathbf{z} \end{aligned} \tag{4.22}$$

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

$$\left\{ \begin{array}{c} \frac{\partial \mathbf{\Lambda}}{\partial x} \\ \\ \frac{\partial \mathbf{\Lambda}}{\partial y} \\ \\ \frac{\partial \mathbf{\Lambda}}{\partial z} \end{array} \right\} = \mathbf{J^{-1}} \left\{ \begin{array}{c} \frac{\partial \mathbf{\Lambda}}{\partial s} \\ \\ \frac{\partial \mathbf{\Lambda}}{\partial t} \\ \\ \frac{\partial \mathbf{\Lambda}}{\partial r} \end{array} \right\} = \frac{1}{|\mathbf{J}|} \left[\begin{array}{ccc} \mathcal{J}_{11} & \mathcal{J}_{12} & \mathcal{J}_{13} \\ \\ \mathcal{J}_{21} & \mathcal{J}_{22} & \mathcal{J}_{23} \\ \\ \mathcal{J}_{31} & \mathcal{J}_{32} & \mathcal{J}_{33} \end{array} \right] \left\{ \begin{array}{c} \frac{\partial \mathbf{\Lambda}}{\partial s} \\ \\ \frac{\partial \mathbf{\Lambda}}{\partial t} \\ \\ \frac{\partial \mathbf{\Lambda}}{\partial r} \end{array} \right\} \tag{4.23}$$

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

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

where

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

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

$$d\Omega = dx\, dy\, dz = |\mathbf{J}|\, ds\, dt\, dr \tag{4.26}$$

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

$$d\Omega = dx\, dy = |\mathbf{J}|\, ds\, dt \tag{4.27}$$

and

$$d\Omega = r\, d\Theta\, dr\, dz = 2\pi\, r\, |\mathbf{J}|\, ds\, dt \tag{4.28}$$

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-sectional areas and thicknesses. Thus for a bar the volume transforms according to

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

where again

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

and $A(s)$ is the cross-sectional 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 Chapter 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

$$\mathbf{K_{12}} = \mathbf{K_{xy}} = \int_{\Omega_e} k_{xy} \frac{\partial \boldsymbol{\Theta}}{\partial x} \frac{\partial \boldsymbol{\Theta}}{\partial y}^T d\Omega \tag{4.30}$$

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

$$\begin{aligned} \mathbf{K_{xy}} \;=\; & \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} k_{xy} \underbrace{\frac{1}{|\mathbf{J}|}\left[\mathcal{J}_{11}\frac{\partial \boldsymbol{\Theta}_\mathbf{q}}{\partial s} + \mathcal{J}_{12}\frac{\partial \boldsymbol{\Theta}_\mathbf{q}}{\partial t} + \mathcal{J}_{13}\frac{\partial \boldsymbol{\Theta}_\mathbf{q}}{\partial r} \right]}_{\frac{\partial \boldsymbol{\Theta}_\mathbf{q}}{\partial x}} \\ & \cdot \underbrace{\left[\mathcal{J}_{21}\frac{\partial \boldsymbol{\Theta}_\mathbf{q}}{\partial s}^T + \mathcal{J}_{22}\frac{\partial \boldsymbol{\Theta}_\mathbf{q}}{\partial t}^T + \mathcal{J}_{23}\frac{\partial \boldsymbol{\Theta}_\mathbf{q}}{\partial r}^T \right] \frac{1}{|\mathbf{J}|}}_{\frac{\partial \boldsymbol{\Theta}_\mathbf{q}}{\partial y}^T} |\mathbf{J}|\, dsdtdr \end{aligned} \tag{4.31}$$

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

The above integral is of the general form

$$I = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} f(s,t,r)\, dsdtdr \tag{4.32}$$

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

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

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 [18] 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 $\mathbf{K_{xy}}$. 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 Boundary Conditions and Source Terms

In this section 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,

$$\mathbf{F_Q} = \int_{\Omega_e} \mathbf{\Theta} Q \, d\Omega. \tag{4.34}$$

The source term is allowed to vary over the element in an arbitrary manner, which is indicated by $Q(s,t,r)$. This volume source may contain an externally defined contribution as well as an internally defined energy release from a chemically reactive material. 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

$$\mathbf{F_Q} = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{\Theta} Q(s,t,r) |\mathbf{J}| \, dsdtdr \tag{4.35}$$

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 thermal load 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.

A special form of volume source is the point source which can be written in a computational form as

$$\mathbf{F_Q} = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{\Theta} Q \delta(s_o, t_0, r_0) |\mathbf{J}| \, dsdtdr \tag{4.36}$$

where δ is the Dirac delta function evaluated at some point (s_o, t_0, r_0) within the element. Evaluating the integral in (4.36) produces the components of the heat source vector

$$\mathbf{F_Q} = \mathbf{\Theta}(s_o, t_0, r_0) \bar{Q} \tag{4.37}$$

where $\bar{Q}$ is the total energy per unit time for the point source and the shape function is evaluated at the source location. The point source thus produces a response at all of the nodes in the element even though its effect should be more localized. The inherent limitations and inaccuracies in this type of singular source must be considered before using the point source form of volume heating.

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 element sides coinciding

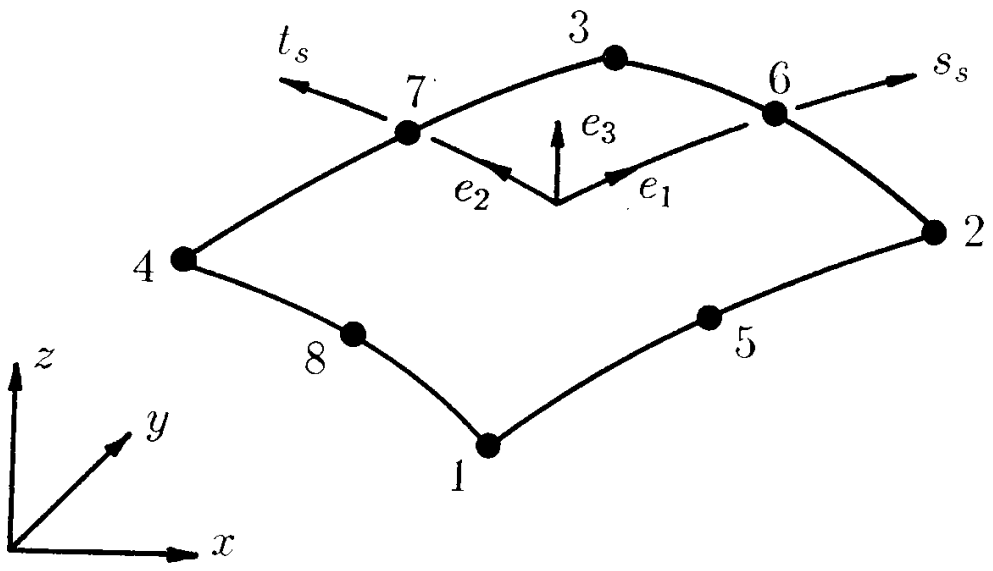

Figure 4.8: Nomenclature for element surface computations.

with the "exterior" boundaries of the problem domain. Due to material addition or deletion the "exterior" surfaces of an element may change with time. An algorithm exists to allow elements to inherit or loose flux boundary conditions as the element topology dictates. The surface flux vector is given by

$$\mathbf{F} = \int_{\Gamma_e} \mathbf{\Theta} q_i n_i \, d\Gamma = \int_{\Gamma_e} \mathbf{\Theta} (f^q - q_c - q_r) \, d\Gamma \tag{4.38}$$

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)- (2.6).

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

$$d\Gamma = |\mathbf{J_s}| \, ds_s dt_s \tag{4.39}$$

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 **e** vectors as

$$|\mathbf{J_s}| = |\mathbf{e_1} \times \mathbf{e_2}| = \left[(\mathbf{e_1} \cdot \mathbf{e_1})(\mathbf{e_2} \cdot \mathbf{e_2}) - (\mathbf{e_1} \cdot \mathbf{e_2})^2\right]^{1/2}. \tag{4.40}$$

The **e** vectors can be expressed in terms of the global coordinates by

$$\mathbf{e_1} = \left\{ \begin{array}{c} \frac{\partial x}{\partial s_s} \\ \\ \frac{\partial y}{\partial s_s} \\ \\ \frac{\partial z}{\partial s_s} \end{array} \right\} \quad ; \quad \mathbf{e_2} = \left\{ \begin{array}{c} \frac{\partial x}{\partial t_s} \\ \\ \frac{\partial y}{\partial t_s} \\ \\ \frac{\partial z}{\partial t_s} \end{array} \right\} \tag{4.41}$$

Using the parametric mapping concept allows

$$x = \hat{\mathbf{\Upsilon}}^T \mathbf{x} \quad ; \quad y = \hat{\mathbf{\Upsilon}}^T \mathbf{y} \quad ; \quad z = \hat{\mathbf{\Upsilon}}^T \mathbf{z} \tag{4.42}$$

where the $\hat{}$ notation indicates the restriction of the interpolation function to an element face (edge). The functions $\hat{\mathbf{\Upsilon}}$ may be either linear or quadratic depending on the type of mapping used to describe the element geometry. Using (4.42) in (4.41) then

$$\mathbf{e_1} = \left\{ \begin{array}{c} \frac{\partial \hat{\mathbf{\Upsilon}}^T}{\partial s_s}\mathbf{x} \\ \\ \frac{\partial \hat{\mathbf{\Upsilon}}^T}{\partial s_s}\mathbf{y} \\ \\ \frac{\partial \hat{\mathbf{\Upsilon}}^T}{\partial s_s}\mathbf{z} \end{array} \right\} \quad ; \quad \mathbf{e_2} = \left\{ \begin{array}{c} \frac{\partial \hat{\mathbf{\Upsilon}}^T}{\partial t_s}\mathbf{x} \\ \\ \frac{\partial \hat{\mathbf{\Upsilon}}^T}{\partial t_s}\mathbf{y} \\ \\ \frac{\partial \hat{\mathbf{\Upsilon}}^T}{\partial t_s}\mathbf{z} \end{array} \right\} \tag{4.43}$$

Equations (4.40) and (4.43) provide a means for computing $|\mathbf{J_s}|$, thus allowing the transformation in (4.39) to be employed. Note that in two dimensions the above relations simplify and the elemental length of an element edge is given by

$$d\Gamma = \left[\left(\frac{\partial \hat{\mathbf{\Upsilon}}^T}{\partial s}\mathbf{x} \right)^2 + \left(\frac{\partial \hat{\mathbf{\Upsilon}}^T}{\partial s}\mathbf{y} \right)^2 \right]^{\frac{1}{2}} ds = \Delta \, ds \tag{4.44}$$

where s is the coordinate along the edge of an element. Axisymetric geometries require integration over an element edge that is rotated through an angle of 2π and (4.44) should therefore be expressed as

$$d\Gamma = \left[\left(\frac{\partial \hat{\mathbf{\Upsilon}}^T}{\partial s}\mathbf{r} \right)^2 + \left(\frac{\partial \hat{\mathbf{\Upsilon}}^T}{\partial s}\mathbf{z} \right)^2 \right]^{\frac{1}{2}} r \, d\Theta \, ds = 2\pi\Delta \, r \, ds \tag{4.45}$$

where r is the radius for the element edge and would be interpolated by $r = \mathbf{\Upsilon}^T \mathbf{r}$. For most three-dimensional surfaces on shell or bar elements, the Jacobian in (4.44) is modified to include the z coordinate. The integration over the end of a bar is simplified because the cross-sectional area normal to the axis of the bar is specified.

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)-(2.6) the normal heat flux consists of three components

$$q_i n_i = f^q - q_c - q_r = f^q - h_c(T - T_c) - h_r(T - T_r) \tag{4.46}$$

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

$$\begin{aligned}\mathbf{F}(\mathbf{T}) = &\int_{-1}^{+1}\int_{-1}^{+1} \hat{\mathbf{\Theta}} f^q(s_s,t_s)|\mathbf{J_s}|\, ds_s dt_s \\ &- \int_{-1}^{+1}\int_{-1}^{+1} \hat{\mathbf{\Theta}} h_c(s_s,t_s)\hat{\mathbf{\Theta}}^T|\mathbf{J_s}|\, ds_s dt_s\, (\mathbf{T}-\mathbf{T_c}(s_s,t_s)) \\ &- \int_{-1}^{+1}\int_{-1}^{+1} \hat{\mathbf{\Theta}} h_r(s_s,t_s)\hat{\mathbf{\Theta}}^T|\mathbf{J_s}|\, ds_s dt_s\, (\mathbf{T}-\mathbf{T_r}(s_s,t_s)) \end{aligned} \tag{4.47}$$

or

$$\mathbf{F}(\mathbf{T}) = \mathbf{F_q} - \mathbf{H_c T} + \mathbf{H_c T_c} - \mathbf{H_r T} + \mathbf{H_r T_r}. \tag{4.48}$$

The integrals in (4.47) 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 ($\mathbf{H_c T}$ and $\mathbf{H_r T}$); 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 section 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

$$\mathbf{F_g}(\mathbf{T}) = -\int_{\Gamma_m} \mathbf{\Theta} q_g\, d\Gamma = -\int_{\Gamma_m} \mathbf{\Theta} h_g (T_m - T_s)\, d\Gamma \tag{4.49}$$

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, because 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

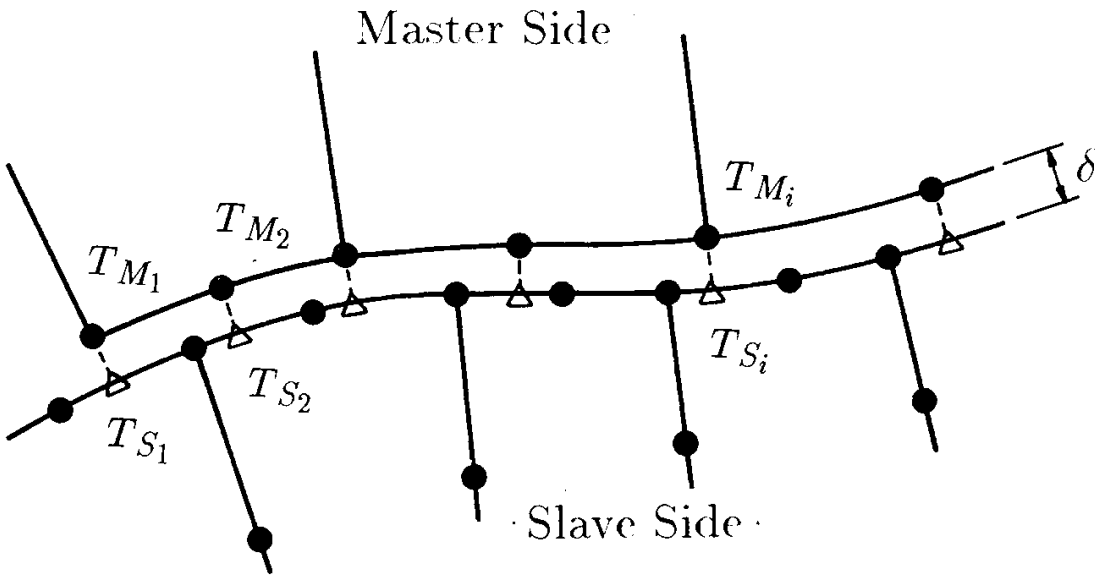

Figure 4.9: Nomenclature for contact resistance formulation.

on each side of the contact surface are not aligned. If a node on the master surface does not have on 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

$$\mathbf{F_g(T)} = -\int_{-1}^{+1}\int_{-1}^{+1} \hat{\mathbf{\Theta}} h_g(s_s, t_s) \hat{\mathbf{\Theta}}^{\mathbf{T}} |\mathbf{J_s}| ds_s\, dt_s (\mathbf{T_m} - \mathbf{T_s}) \tag{4.50}$$

or in matrix form

$$\mathbf{F_g(T)} = -\mathbf{H_g T_m} + \mathbf{H_g T_s} \tag{4.51}$$

In developing (4.51) 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 temperatures on the master surface; the matrix in the term $\mathbf{H_g T_m}$ 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 is usually solved via an iterative process. A non-iterative (implicit) formulation and solution method for contact can be developed by writing the slave temperature, T_s, in terms of

the nodal point temperatures of the slave element. In this case, the $\mathbf{H_g T_s}$ vector is altered by an interpolation matrix, $\mathbf{I_s}$, such that $\mathbf{H_g I_s T}$ is the appropriate term for the slave side where $\mathbf{T}$ are the nodal point temperatures on the slave surface. This term is combined with the diffusion matrix during the solution process. The only drawback to this implict form is the need to redo connectivities and matrix pointers when the contact surfaces are dynamic.

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. Section 5.9 outlines the algorithm used to determine the occurrence of contact and the spatial location of master nodes on the slave surface. This is the thermal equivalent of the slide line algorithms used in solid mechanics [19].

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 (4.49) forces the temperature distributions on each side of the contact surface to be essentially equal. Though this method can be made to work in practice, it is not optimal as large values of h_g can cause ill-conditioning of the matrix problem and difficulties in reaching convergence with an iterative solution method. The constraint boundary conditions mentioned in a subsequent section are a better alternative for connecting dissimilar mesh regions.

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.

4.10.5 Temperature Constraint Conditions

For some applications it is necessary to specify the functional relationship between the temperature at one node and temperature at one or more other nodes. The enforcing of temperature continuity between coincident surfaces with dissimilar meshes and the specification of spatially periodic temperature boundary conditions are two examples of this type of constraint. The multipoint constraint algorithm is structured like the contact algorithm with one surface labeled the master surface and the constrained temperature surface labeled the slave surface. The locations of the slave nodes on the master surface are found and recorded using the same search procedure as used in the contact algorithm. The field equation for each slave node is replaced with a constraint equation that relates the temperature at the slave node to some

function of the nodal temperatures on the master surface. In the case of temperature continuity, the slave node value is the interpolant of the master node values. For periodic conditions, the slave node temperature would typically be set to the master surface temperature plus some temperature increment representing the heat flux across the periodic geometry. Both of these types of constraints are available in COYOTE for static mesh configurations.

In matrix form a set of constraint equations for the temperature can be written as

$$\mathbf{CT} = \mathbf{IT_c} + \mathbf{C_k T_k} = \mathbf{F_c} \tag{4.52}$$

where $\mathbf{T_c}$ are constrained temperatures, $\mathbf{T_k}$ are "known" nodal point temperatures, $\mathbf{I}$ is the identity matrix and $\mathbf{C_k}$ is the constraint coefficient matrix. The vector $\mathbf{F_c}$ is zero for equality constraints and equal to the temperature offset for periodic boundary consitions. The summation form of the constraint equation shows that there are more columns than rows in the system, *i.e.*, the $\mathbf{C}$ matrix is not square. Either a Lagrange multiplier method or a penalty formulation can be used to incorporate the constraints into the basic finite element problem [20]. COYOTE uses a penalty method that takes the form

$$\mathbf{K_c T} = \mathbf{C^T} \alpha \mathbf{CT} = \mathbf{C^T} \alpha \mathbf{F_c} \tag{4.53}$$

where α is a matrix of penalty coefficients and $\mathbf{K_c}$ is the constraint matrix that is added to the global diffusion matrix. The penalty coefficients are problem dependent but should generally be sized at one to two orders of magnitude larger than a representative thermal conductivity.

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 (4.48) and (4.51), then (3.10) becomes

$$\begin{aligned} \mathbf{M(T)\dot{T}} + \mathbf{K(T)T} &= \mathbf{F_Q(T)} + \mathbf{F_q} - \mathbf{H_c T} + \mathbf{H_c T_c} \\ &\quad -\mathbf{H_r T} + \mathbf{H_r T_r} - \mathbf{H_g T} + \mathbf{H_g T_s} \end{aligned} \tag{4.54}$$

Rearranging (4.54) allows the final form of the discrete system to be written as

$$\overline{\mathbf{M}}(\mathbf{T})\dot{\mathbf{T}} + \overline{\mathbf{K}}(\mathbf{T})\mathbf{T} = \overline{\mathbf{F}}(\mathbf{T}) \tag{4.55}$$

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

$$\mathbf{F_c} = \mathbf{H_c T_c}$$

$$\mathbf{F_r} = \mathbf{H_r T_r}$$

$$\mathbf{F_g} = \mathbf{H_g T_s}$$

The equation in (4.55) 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 (4.55) is modified to include the unsymmetric, velocity dependent advection term.

A similar system can be written for the auxiliary variables $\boldsymbol{\phi}^{\mathbf{k}}$ when auxiliary diffusion or auxiliary advection-diffusion processes are required. Expressing (3.21) in a form analogous to (4.54) leads to

$$\mathbf{M}^{\mathbf{k}}(\boldsymbol{\phi}^{\mathbf{k}})\dot{\boldsymbol{\phi}}^{\mathbf{k}} + \mathbf{K}^{\mathbf{k}}(\boldsymbol{\phi}^{\mathbf{k}})\boldsymbol{\phi}^{\mathbf{k}} = \mathbf{F}^{\mathbf{k}}_{\mathbf{Q}}(\boldsymbol{\phi}^{\mathbf{k}}) + \mathbf{F}^{\mathbf{k}}_{\mathbf{q}}(\boldsymbol{\phi}^{\mathbf{k}}) - \mathbf{H}^{\mathbf{k}}_{\mathbf{c}}\boldsymbol{\phi}^{\mathbf{k}} + \mathbf{H}^{\mathbf{k}}_{\mathbf{c}}\boldsymbol{\phi}^{\mathbf{k}}_{\mathbf{c}} - \mathbf{H}^{\mathbf{k}}_{\mathbf{g}}\boldsymbol{\phi}^{\mathbf{k}} + \mathbf{H}^{\mathbf{k}}_{\mathbf{g}}\boldsymbol{\phi}^{\mathbf{k}}_{\mathbf{s}}$$

or more compactly

$$\overline{\mathbf{M}}^{\mathbf{k}}(\boldsymbol{\phi}^{\mathbf{k}})\dot{\boldsymbol{\phi}}^{\mathbf{k}} + \overline{\mathbf{K}}^{\mathbf{k}}(\boldsymbol{\phi}^{\mathbf{k}})\boldsymbol{\phi}^{\mathbf{k}} = \overline{\mathbf{F}}^{\mathbf{k}}(\boldsymbol{\phi}^{\mathbf{k}}) \tag{4.56}$$

with

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

$$\overline{\mathbf{K}}^{\mathbf{k}} = \mathbf{K}^{\mathbf{k}} + \mathbf{H}^{\mathbf{k}}_{\mathbf{c}} + \mathbf{H}^{\mathbf{k}}_{\mathbf{g}}$$

$$\overline{\mathbf{F}}^{\mathbf{k}} = \mathbf{F}^{\mathbf{k}}_{\mathbf{Q}} + \mathbf{F}^{\mathbf{k}}_{\mathbf{q}} + \mathbf{F}^{\mathbf{k}}_{\mathbf{c}} + \mathbf{F}^{\mathbf{k}}_{\mathbf{g}}$$

and

$$\mathbf{F}^{\mathbf{k}}_{\mathbf{c}} = \mathbf{H}^{\mathbf{k}}_{\mathbf{c}}\boldsymbol{\phi}^{\mathbf{k}}_{\mathbf{c}}$$

$$\mathbf{F}^{\mathbf{k}}_{\mathbf{g}} = \mathbf{H}^{\mathbf{k}}_{\mathbf{g}}\boldsymbol{\phi}^{\mathbf{k}}_{\mathbf{s}}$$

and the superscript $\mathbf{k}$ can be either A or B. The possible coupling of (4.56) to (4.55) through properties, boundary conditions and/or source terms is not shown explicitly but is allowed in the implementation. When convective terms are required, the $\overline{\mathbf{K}}^{\mathbf{k}}$ matrix is modified as in the conduction equation.

When auxiliary variables are considered, the matrix systems in (4.55) and (4.56) are combined to form an overall system

$$\hat{\mathbf{M}}\dot{\mathbf{X}} + \hat{\mathbf{K}}\mathbf{X} = \hat{\mathbf{F}} \tag{4.57}$$

with

$$\hat{\mathbf{M}} = \begin{bmatrix} \overline{\mathbf{M}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \overline{\mathbf{M}}^{\mathbf{A}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \overline{\mathbf{M}}^{\mathbf{B}} \end{bmatrix} \quad ; \quad \hat{\mathbf{K}} = \begin{bmatrix} \overline{\mathbf{K}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \overline{\mathbf{K}}^{\mathbf{A}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \overline{\mathbf{K}}^{\mathbf{B}} \end{bmatrix} \tag{4.58}$$

$$\hat{\mathbf{F}} = \begin{Bmatrix} \overline{\mathbf{F}} \\ \overline{\mathbf{F}}^{\mathbf{A}} \\ \overline{\mathbf{F}}^{\mathbf{B}} \end{Bmatrix} \quad ; \quad \mathbf{X} = \begin{Bmatrix} \mathbf{T} \\ \boldsymbol{\phi}^{\mathbf{A}} \\ \boldsymbol{\phi}^{\mathbf{B}} \end{Bmatrix} \tag{4.59}$$

The system is shown for the most general case where two auxiliary variables are defined; if only one auxiliary variable is needed, the $\boldsymbol{\phi}^{\mathbf{B}}$ equation is eliminated.

The periodic heat conduction problem produced a set of coupled equations for the real and imaginary components of the temperature. As an assembled matrix problem, the equations in (3.23) and (3.24) are written as

$$\tilde{\mathbf{M}}\mathbf{Y} + \tilde{\mathbf{K}}\mathbf{Y} = \check{\mathbf{K}}\mathbf{Y} = \check{\mathbf{F}} \tag{4.60}$$

with

$$\tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{0} & -\omega\overline{\mathbf{M}} \\ \omega\overline{\mathbf{M}} & \mathbf{0} \end{bmatrix} \quad ; \quad \tilde{\mathbf{K}} = \begin{bmatrix} \overline{\mathbf{K}} & \mathbf{0} \\ \mathbf{0} & \overline{\mathbf{K}} \end{bmatrix} \tag{4.61}$$

$$\check{\mathbf{K}} = \begin{bmatrix} \overline{\mathbf{K}} & -\omega\overline{\mathbf{M}} \\ \omega\overline{\mathbf{M}} & \overline{\mathbf{K}} \end{bmatrix} \tag{4.62}$$

$$\check{\mathbf{F}} = \begin{Bmatrix} \mathbf{F_R} \\ \mathbf{F_I} \end{Bmatrix} \quad ; \quad \mathbf{Y} = \begin{Bmatrix} \mathbf{T_R} \\ \mathbf{T_I} \end{Bmatrix} \tag{4.63}$$

This defines a linear, time independent matrix problem for a periodic simulation. Note that the matrix problem is unsymmetric.

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

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

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 global 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 or a time periodic form alters these characteristics slightly by making the system unsymmetric; the nonlinearity and matrix structure remain the same. Note that with a simple redefinition of the matrices and vectors, the equation in (5.1) may represent the conduction problem, a coupled conduction, auxiliary variable problem or a time periodic problem.

In the following sections 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. Explict consideration of the coupled

auxiliary variable problem will not be considered. 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

$$\overline{\mathbf{K}}(\mathbf{T})\mathbf{T} = \overline{\mathbf{F}}(\mathbf{T}) \tag{5.2}$$

which is recognized as a system of nonlinear, algebraic equations. Consider first the case where $\overline{\mathbf{K}}$ and $\overline{\mathbf{F}}$ are not functions of $\overline{\mathbf{T}}$. 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.

Note that the time periodic problem leads to a matrix problem similar to the linear form of (5.2) but for the real and imaginary components of the temperature. This case can thus be solved directly, without iteration.

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

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

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

$$\overline{\mathbf{K}}(\mathbf{T}^n)\mathbf{T}^\star = \overline{\mathbf{F}}(\mathbf{T}^n) \tag{5.4}$$

and

$$\mathbf{T}^{n+1} = \alpha\mathbf{T}^n + (1-\alpha)\mathbf{T}^\star \qquad 0 \le \alpha < 1.$$

The above methods are adequate for the large majority of thermal diffusion problems because 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

$$\overline{\mathbf{K}}(\mathbf{T}, \lambda)\mathbf{T} = \overline{\mathbf{F}}(\mathbf{T}, \lambda) \tag{5.5}$$

which suggests the zeroth order continuation method

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

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

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

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

$$d_{n+1} \leq \epsilon^T \tag{5.8}$$

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 stablity 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 weaknessess, 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.* [21] 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

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

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

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

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

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

or in a form more suitable for computation

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

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

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

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

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

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

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

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

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

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 usually carried to absolute convergence. Rather, a one-step correction is usually employed as advocated in [21]; this is the default procedure in COYOTE. The one-step procedure is quite efficient and can be very accurate provided the time step is suitably controlled. Multiple iterations within a time step may be employed as an option, but would not normally be required except in strongly nonlinear problems involving enclosure radiation and/or chemical kinetics.

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 [21]. 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 [21] as

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

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

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

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 [21], 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. Conversely, the initial time step should not be so small as to produce spatial oscillations due to unresolved temperature gradients. See Appendix C in the User's Manual [5] for a method to select an initial time step.

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 (5.9) and is written here as

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

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

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

As written in (5.18), this algorithm does not requires 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 [16]. 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, $\mathbf{M_D}$, is formed by the components given by

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

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 [22] produce a time step restriction of the following form

$$\lambda_{max}^G \Delta t \leq 2 \tag{5.20}$$

where λ_{max}^G is the largest eigenvalue for the (global) 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 [23,24], 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 [25] is employed. The largest element eigenvalue can be bounded by the sum of the largest eigenvalues at each integration point. That is

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

where N_{ip} is the number of integration points in the element. As shown in [25], the nonzero eigenvalues at an integration point can be found from the following matrix

$$\mathbf{S} = \frac{w}{m}\mathbf{k}\mathbf{B}\mathbf{B}^{T} \tag{5.22}$$

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

$$\mathbf{B} = \left\{ \begin{array}{c} \frac{\partial \mathbf{\Theta}}{\partial x} \\ \\ \frac{\partial \mathbf{\Theta}}{\partial y} \\ \\ \frac{\partial \mathbf{\Theta}}{\partial z} \end{array} \right\}. \tag{5.23}$$

The vector $\mathbf{\Theta}$ is the element interpolation function for the temperature. The maximum eigenvalue of $\mathbf{S}$ is readily computed for each integration point as 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 [23,25]. 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

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{5.24}$$

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 was initially embedded in a PCG library package that was developed by Schunk and Shadid [26]; this package has been superceded by the Aztec library [27] which has been designed for either serial or parallel computer architectures. Current versions of the Aztec solvers are accessed through the more general solver package Trilinos [28,29]. The COYOTE interface to the solver libraries is through the Finite Element Interface (FEI) [30] which standardizes the interaction and formats for matrix solution. Access to other solver libraries is also simplified with use of the FEI. 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 [29,30] that only records the nonzero entries of the $\mathbf{A}$ matrix. Complete details on the iterative solvers available in COYOTE can be found in [27-29].

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 2.4, 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 or changing element topology, must have the view factors updated periodically. COYOTE employs a number of number of different

methods for the actual view factor computation, all of which are embedded in the companion computer code, CHAPARRAL [31]. 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 [31].

The basic view factor definition is given by (2.18) as

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

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). Numerous methods have been developed for evaluating F_{k-j} and many of these are catalogued in [8]. An adaptive procedure has been utilized in CHAPARRAL where the algorithm selected for each pair-wise view factor computation is chosen based on a geometric and computational cost basis. The possible procedures include: a) an analytic method, b) Hottel string method, c) Gauss quadrature, d) Monte Carlo integration and e) Quasi-Monte Carlo integration. The algorithm selection procedure and algorithms used for determining third surface shadowing are detailed in the CHAPARRAL manual [31].

A significant difficulty with the procedures cited above is their relatively poor efficiency for three-dimensional problems with very large numbers of surfaces. CHAPARRAL has an alternate procedure available for these situations which is based on a hemicube algorithm [31,32]. This method peforms 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 [31]. A comparison of all of the above view factor methods has been reported in [32].

5.5 Radiation Solution Algorithms

The enclosure radiation problem was outlined in Section 2.4 and resulted in an equation of the following form

$$\sum_{j=1}^{N} \left[\delta_{kj} - (1-\epsilon_j)\, F_{k-j}\right] q_j = \sum_{j=1}^{N} \left(\delta_{kj} - F_{k-j}\right) \epsilon_j \sigma {T_j}^4. \tag{5.26}$$

which may also be rewritten as the set of two equations

$$\sum_{j=1}^{N} \left[\delta_{kj} - (1-\epsilon_k) F_{k-j}\right] q_j^o = \epsilon_k \, \sigma \, T_k^4 \tag{5.27}$$

$$q_k = q_k^o - \sum_{j=1}^{N} F_{k-j} \, q_j^o. \tag{5.28}$$

where q_j are the uniform surface fluxes and T_k are the uniform surface temperatures; q_j^o are the uniform outgoing surface fluxes.

The discrete form of the radiation problem in (5.26) can be expressed in matrix form as

$$(\mathbf{I} - \mathcal{F}\boldsymbol{\rho})\bar{\mathbf{q}} = (\mathbf{I} - \mathcal{F})\boldsymbol{\epsilon}\sigma\bar{\mathbf{T}}^4 = (\mathbf{I} - \mathcal{F})\boldsymbol{\epsilon}\mathbf{E_b} \tag{5.29}$$

or in a compact form

$$\mathbf{A}(\bar{\mathbf{T}})\bar{\mathbf{q}} = \mathbf{D}(\bar{\mathbf{T}})\bar{\mathbf{T}} \tag{5.30}$$

where $\mathbf{I}$ is the identity matrix, $\mathcal{F}$ is the matrix of view factors, $\boldsymbol{\rho} = \mathbf{I} - \boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon}$ are diagonal matrices of reflectances and emittances, respectively, and $\mathbf{E_b}$ is a vector representing the black-body emissive power. Also, $\mathbf{A}$ is the radiative flux coefficient matrix, with a temperature dependence due to surface property variations, and $\mathbf{D}$ is the surface temperature coefficient matrix with a cubic dependence on surface temperature. The vector $\bar{\mathbf{q}}$ represents the uniform net radiative flux on each surface and $\bar{\mathbf{T}}$ is the vector of uniform surface temperatures. The computational form of (5.27) and (5.28) may be expressed in matrix form as

$$(\mathbf{I} - \mathcal{F}\boldsymbol{\rho})\bar{\mathbf{q}}^{\mathbf{o}} = \boldsymbol{\epsilon}\mathbf{E_b} \tag{5.31}$$

$$\bar{\mathbf{q}} = (\mathbf{I} - \mathcal{F})\bar{\mathbf{q}}^{\mathbf{o}} \tag{5.32}$$

In all of these formulations, the assumptions of the radiative model require the values of the surface temperatures $\bar{\mathbf{T}}$ to be constant over each surface. In COYOTE it is assumed that each surface in the radiation problem corresponds to a face or edge of a finite element. The required uniform surface temperatures needed for use in the radiation equations are obtained by combining (averaging) the nodal point temperatures on the appropriate element face or edge.

5.5.1 Solution Strategies (Segregated)

COYOTE contains a series of solution strategies that may be invoked for transient and steady solutions of coupled conduction and enclosure radiation problems. The methods are outlined below in order of increasing complexity.

When the surface temperatures for all surfaces are known, Equation (5.30) forms a set of linear algebraic equations for the unknown net surface fluxes, $\bar{\mathbf{q}}$. That is, Equation (5.30) can be written as

$$\mathbf{A}(\bar{\mathbf{T}}^n)\bar{\mathbf{q}}^{n+1} = \mathbf{D}(\bar{\mathbf{T}}^n)\bar{\mathbf{T}}^n \tag{5.33}$$

The combination of (5.31) and (5.32) could also be used for generating the solution for $\bar{\mathbf{q}}$. Note that the matrix $\mathbf{A}$ is a full matrix due to the surface to surface coupling represented by the view factors $\mathcal{F}$. This characteristic, along with the possible temperature dependencies, suggests the use of an iterative solution method for (5.33) rather than a direct matrix factorization.

COYOTE employs a Gauss-Seidel or generalized minimum residual (GMRES) method to solve (5.33) for the net surface fluxes, $\bar{\mathbf{q}}$; these solution algorithms are located in the CHAPARRAL code and are explained in more detail in [31]. 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 segregated explicit Picard solution procedure outlined above is fairly reliable and can be very efficient for time-dependent problems when a one-step corrector method is appropriate. Some improvement in the coupling may be realized by using various semi-implicit methods that rearranges the radiative flux boundary condition to be more dependent on the conduction temperature field. Rewritting Equation (5.32) in terms of the black-body emissive power and irradition at the new time level or iteration

$$\bar{\mathbf{q}}^{n+1} = \boldsymbol{\epsilon}\mathbf{E_b}^{n+1} - \boldsymbol{\epsilon}\mathbf{G}^{n+1} = \boldsymbol{\epsilon}\,\sigma\,(\bar{\mathbf{T}}^{n+1})^4 - \boldsymbol{\epsilon}\mathbf{G}^{n+1} \tag{5.34}$$

Assume that the irradiation does not vary significantly from the previous iteration and can be approximated by $\mathbf{G}^n$. Also, linearize the emissive power such that (5.34) becomes

$$\bar{\mathbf{q}}^{n+1} \approx \boldsymbol{\epsilon}\sigma(\bar{\mathbf{T}}^n)^3\bar{\mathbf{T}}^{n+1} - \boldsymbol{\epsilon}\mathbf{G}^n = \bar{h}_r(\bar{\mathbf{T}}^n)\bar{\mathbf{T}}n + 1 - \boldsymbol{\epsilon}\mathbf{G}^{n+1} \tag{5.35}$$

This form of the radiative flux is analogous to other heat transfer types of boundary conditions and is treated computationally in a similar manner; the first term on the right-hand-side of (5.35) is added to the diffusion matrix in the conduction equation and the remaining terms is part of the force vector. This type of semi-implicit formulation is available in COYOTE and has proven to be of benefit in some time-dependent problems. Another type of semi-implict treatment, based on Newton's method, is also available in the code and has proven to be even more effective than the Picard type method outlined here. Details of this approach are given in [33].

When time-independent problems are considered, the decoupled nature of the process leads to significant convergence problems. Convergence problems also occur when a transient problem approaches a steady state or time-independent interval. 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 significant 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 partial resolution of this dilema may be found in the use of a transient or false transient technique, as described above, with the additional constraint that the solution be tightly converged within each time step. Multiple iterations within the time step force the temperature and radiative flux solutions to reach equilibrium with the further benefit that the adaptive time step selection algorithm is able to maintain a reasonable time step. However, for radiation dominated problems even this approach will eventually degrade. Convergence within the time step becomes increasingly difficult to achieve; a time step reduction may temporarily help the solution process though this remedy runs counter to the physical situation where the time step should increase as a steady

state is approached. The proper resolution of this problem is found in a more fully coupled solution technique that simultaneously solves for the temperature and surface flux.

5.5.2 Solution Strategies (Coupled)

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

$$\overline{\mathbf{K}}(\mathbf{T})\mathbf{T} + \overline{\mathbf{B}}\mathbf{q} = \overline{\mathbf{F}}(\mathbf{T}) \tag{5.36}$$

and

$$\mathbf{A}(\mathbf{T})\mathbf{q} = \mathbf{F}_\sigma(\mathbf{T}) = \mathbf{D}(\mathbf{T})\mathbf{T} \tag{5.37}$$

In the conduction Equation (5.36) the boundary conditions involving the radiative surface fluxes have been removed from the $\overline{\mathbf{F}}$ 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 (5.37) for $\mathbf{q}$ (inverting the $\mathbf{A}$ matrix) and substituting the result into (5.36), 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. Rewritting (5.36) and (5.37) as

$$\left\{ \begin{array}{c} \mathbf{R_T} \\ \\ \mathbf{R_q} \end{array} \right\} = \left\{ \begin{array}{c} \overline{\mathbf{K}}(\mathbf{T})\mathbf{T} + \overline{\mathbf{B}}\mathbf{q} - \overline{\mathbf{F}}(\mathbf{T}) \\ \\ -\mathbf{D}(\mathbf{T})\mathbf{T} + \mathbf{A}(\mathbf{T})\mathbf{q} \end{array} \right\} \tag{5.38}$$

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

$$[\mathbf{J}^n] \left\{ \begin{array}{c} \mathbf{\Delta T}^{n+1} \\ \\ \mathbf{\Delta q}^{n+1} \end{array} \right\} = \left[\begin{array}{cc} \frac{\partial \mathbf{R_T}}{\partial \mathbf{T}} & \frac{\partial \mathbf{R_T}}{\partial \mathbf{q}} \\ \\ \frac{\partial \mathbf{R_q}}{\partial \mathbf{T}} & \frac{\partial \mathbf{R_q}}{\partial \mathbf{q}} \end{array} \right] \left\{ \begin{array}{c} \mathbf{\Delta T}^{n+1} \\ \\ \mathbf{\Delta q}^{n+1} \end{array} \right\} = - \left\{ \begin{array}{c} \mathbf{R_T}^n \\ \\ \mathbf{R_q}^n \end{array} \right\} \tag{5.39}$$

The iterative solution procedure specified in (5.39) has very good convergence properties and is available as an option in COYOTE. 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, because these are linearized and evaluated at the previously computed temperature. Though the method in (5.39) 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 [33].

It should be noted that the fully coupled algorithm may also be applied to the transient problem. After a time integration method is specified, Equation (5.36) is augmented with a

capacitance matrix which follows through to the Jacobian definition in (5.39). Though the system in (5.36) and (5.37) is time singular in the radiosities, no significant difficulties should be encountered in the solution of this system except those noted above. This method is available in COYOTE as an option.

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 2.5. 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 [34].

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 [35] and the second generation package, CHEMEQ2 [36]. The techniques used in CHEMEQ and CHEMEQ2 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 reactive elements is used to regulate the choice of the thermal diffusion time step. The choice of the thermal diffusion time step is computed from

$$\Delta t_{n+1} = min\{\Delta t_{diff}, X_{chem} \times \Delta t_{chem}\} \tag{5.40}$$

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 (5.40) 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 (5.40). 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

$$C^*(T) = C(T) + L\delta^*(T - T_t, \Delta T) \tag{5.41}$$

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 (5.41) is thus an approximation for the behavior of pure materials that change phase at a specified temperature, T_t, but more accurately approximates nonpure substances that have truely 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 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

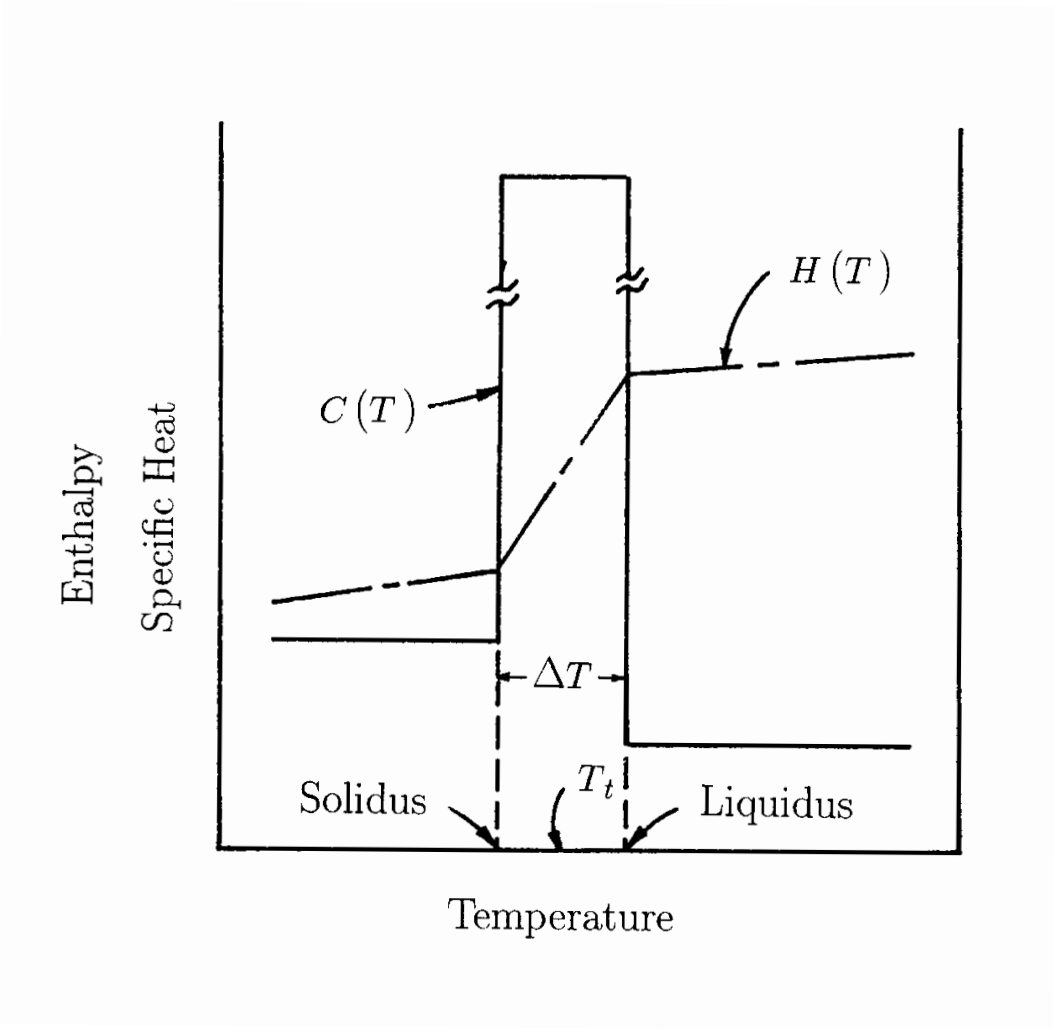

Figure 5.1: Definition of material properties for phase change computation.

on the local slope of the enthalpy function. In one case, the following definition is used

$$C_p = \frac{dH}{dT} = \frac{dH/dt}{dT/dt} \tag{5.42}$$

which can be rewritten in a computational form as

$$C_p = \frac{H(T^{n+1}) - H(T^n)}{T^{n+1} - T^n} \tag{5.43}$$

where the superscript denotes the time step number. Equation (5.43) 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 (5.43) 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 (5.42). 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

$$C_p = \left[\frac{\nabla H \cdot \nabla H}{\nabla T \cdot \nabla T}\right]^{1/2} \tag{5.44}$$

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 (5.41) 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

$$Q_{lh} = -\rho L \delta^*(T - T_t, \Delta T)\frac{\partial T}{\partial t} \tag{5.45}$$

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 Bulk Node Algorithms

The ordinary differential equations that describe the mass and energy of a bulk node, Equations (2.14) and (2.16), are integrated separately from the finite element equations. The time integration methods available for the bulk node equations are the same as the methods used for the conduction equation. That is, the implicit backward Euler and trapezoid rule algorithms are available as is the explicit, forward Euler method. A predictor/corrector implementation

for the implicit methods is also provided. Using the backward Euler method as an example, the bulk node mass and energy equations would be evaluated by

$$M^{n+1} = M^n + \Delta t_n f_M(t^n) \tag{5.46}$$

$$U^{n+1} = U^n + \Delta t_n f_U(t^n) \tag{5.47}$$

where Δt_n is the time step and the f functions are the right-hand-sides of (2.14) and (2.16), respectively. The bulk node volume is not integrated in time but simply updated according to the specified volume change that may include element removal/addition or movement of the bulk node containment. For problems where the bulk node volume is entirely described by element surfaces, an internal volume computation is provided. This volume is computed by summing the volumes of all tetrahedons formed from the triangular facets of an each bounding element face and a reference point. Unbounded bulk nodes must have a volume specified.

The backward Euler integration formulas in (5.46) and (5.47) are the recommended algorithms for the bulk node equations; the use of a compatible predictor equation is not recommended. Typically, the right-hand-side data for these equations is rough, especially in situations where element death contributes to the bulk node equations. Higher-order integration, with a predictor equation, is not warranted in these cases and the backward Euler method provides the best accuracy and stability.

Though the bulk node is described in terms of the internal energy, the temperature is the variable of interest. The bulk node temperature is recovered from the latest value of the internal energy by solving the following equation for T^{n+1}.

$$\left(\int_{T_0}^{T^{n+1}} C_v(T)\, dT + u_0 \right) M^{n+1} = U^{n+1} \tag{5.48}$$

In Equation (5.48), C_v is the specific heat at constant volume for the bulk node material and u_0, T_0 are the reference internal energy per unit mass and the reference temperature for the internal energy, respectively. The integral in (5.48) is computed with an adaptive quadrature (trapezoid rule) scheme that continually halves the integration intervals until the integral value is converged. An outside iterative loop alters the upper limit on the integral until the T^{n+1} value results in a match with the total internal energy. Once the bulk node temperature is converged the pressure for the bulk node is computed from the equation of state, which is typically a perfect gas,

$$P^{n+1} = \frac{M^{n+1}}{V^{n+1}} \frac{R}{\mathcal{M}} T^{n+1} \tag{5.49}$$

where $\mathcal{M}$ is the molecular weight.

5.9 Contact and Multipoint Constraint Algorithms

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. The multipoint constraint (MPC) equations for temperature that were outlined in Section 4.10.5 also depend on a master and slave surface paradigm. However, before these formulations can be utilized, the geometric properties associated with master and slave surfaces must be determined. Contact detection involves identifying the time at which contact (or separation) occurs and the location (coordinates) of the master nodes on the slave surface. As used in COYOTE, the multipoint constraints are static and therefore master node locations on a slave surface need to be found only once. COYOTE employs a contact/MPC detection algorithm that was initially developed for use in solid mechanics finite element codes [37]. 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 because 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 search process for finding the location of a node on an element surface involves the assembly of a node list and a list of potential contacting elements. The data in these lists are reorganized by a recursive bisection method to associate a node with a reduced list of elements that are spatially nearby. The node is then tested against each element in the group until its proper partner is found; the local coordinates for the node within the element are determined. With this data in hand, either the contact boundary condition or the MPC construction can be completed.

For both contact and MPC implementations, the possible mismatch of master and slave element faces presents some difficulty for the consistent application of flux boundary conditions to the exposed (non-contacting) surfaces [13]. Partially uncovered element surfaces may be treated exactly if the surface imprints are computed and an outline of the exposed region is generated. Boundary conditions are then evaluated using a boundary quadrature. COYOTE uses a more approximate method that avoids the surface intersection computation. Since the surface boundary conditions are evaluated numerically at surface integration points, COYOTE simply checks the covered/uncovered status of each integration point on a surface. Those quadrature points

that are uncovered participate in the boundary condition evaluation; contributions from covered points are neglected. The algorithm for evaluating quadrature point coverage is the same as the original slave node/master element search algorithm. This method for partially covered element surfaces is efficient and effective for reasonable surface mesh densities. Accuracy is obviously degraded when boundary conditions are applied to large, partially exposed elements. Also, partially covered surfaces are wholly included in radiation enclosure computations though the application of the radiative flux is still governed by the number of uncovered quadrature points.

5.10 Front Tracking Algorithm

The solution of the level set or front tracking equation given in (3.27) is accomplished with the same integration algorithms as employed for the time-dependent conduction equation. The solution of the front tracking equation is always decoupled from the conduction solution and is updated after the conduction step. If the forward Euler method is selected then (3.27) can be rewritten as

$$\mathbf{f}^{n+1} = \mathbf{f}^n + \Delta t_n \, \hat{\mathbf{M}}^{-1} \left[-\hat{\mathbf{C}} \mathbf{f}^n \right]. \tag{5.50}$$

The efficient use of (5.50) requires that the mass matrix $\hat{\mathbf{M}}$ be easily inverted. This scheme is conditionally stable and must obey a time step restriction similar to (5.20). This algorithm would normally be subcycled with respect to the conduction solution, especially if the conduction solution is generated with an implicit integration method.

An implicit method such as the trapezoid rule can also be used with (3.27). The trapezoid rule form is given by

$$\left[\frac{2}{\Delta t_n} \hat{\mathbf{M}} + \hat{\mathbf{C}} \right] \mathbf{f}^{n+1} = \frac{2}{\Delta t_n} \hat{\mathbf{M}} \mathbf{f}^n + \hat{\mathbf{M}} \dot{\mathbf{f}}^n. \tag{5.51}$$

This scheme is unconditionally stable but would normally be run subcycled to the conduction equation to maintain accuracy. The form given in (5.51) requires a matrix solution at each time step. The iterative matrix solvers used in the conduction problem are also used with this equation system.

5.11 Parallel Solution Methods

The parallel version of COYOTE is structured for a domain decomposition approach to a finite element problem. Under this paradigm, the overall problem geometry (elements) is divided into N groups of elements for execution on N processors of the parallel platform. A copy of COYOTE is running on each processor and each processor is primarily responsible for the element construction and solution of the subproblem assigned to the processor. Communication of data between processors is handled through a message passing utility.

The above description is superficial but conveys the essence of the algorithm. A few further details are of interest but not essential to the understanding of the algorithms in COYOTE. The initial problem description in terms of the mesh configuration and input data are decomposed and load-balanced using external utilities such as CHACO [38] and the NEMESIS library [39]. The result of the decomposition is a set of N files describing the subproblem for each processor. On each processor the finite element equations for the elements assigned to the processor are constructed and boundary conditions applied. The partially assembled matrix is passed to the solver library, through the FEI [30], which computes a solution using an iterative method. Communication between processors during the matrix solve is accomplished using a message passing library, MPI [40] and is contained within the solver library. Upon completion of the matrix solution, each processor performs any post-processing operations for its assigned group of elements and then proceeds to the next time step or iteration. Global (mesh-independent) data that must be communicated between processors in COYOTE, such as time step and norm data, is handled through wrapper utilities that access the MPI routines.

For problems involving enclosure radiation, COYOTE calls the CHAPARRAL library [31] to compute view factors and solve the radiosity matrix problem. The surface description for each enclosure is passed to CHAPARRAL which performs its own load-balance prior to computing view factors. The parallel solution of the radiosity problem is handled by another call to the solver library. The solution of the chemical kinetics problem is another subprocess that should be load-balanced and solved in parallel. Unlike the previous parallel tasks that are decomposed based on geometry, the decomposition of the chemistry problem should consider the current reactivity of each element, *i.e.*, the decomposition is task-based. This option has not been implemented in the present version of COYOTE and the chemistry solution is produced using the overall finite element decomposition, which is not optimal. Finally, search procedures, such as those needed for contact detection and the multipoint constraints, are particularly difficult in parallel, since the data is usually nonlocal and off the processor. The algorithm used here is based on the parallel, solid mechanics contact scheme described in [37,41].

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 quantitites. 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 [42]. A complete description of the mesh generation interface to COYOTE is available in the user's manual [5]. For parallel applications the EXODUS II input file must be split into a parallel file structure using the NEMESIS libraries [39] and assigned to the individual processors.

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

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

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

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

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

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

$$\left\{ \begin{array}{c} \frac{\partial \mathbf{\Theta}}{\partial x} \\ \\ \frac{\partial \mathbf{\Theta}}{\partial y} \\ \\ \frac{\partial \mathbf{\Theta}}{\partial z} \end{array} \right\} = \mathbf{J^{-1}} \left\{ \begin{array}{c} \frac{\partial \mathbf{\Theta}}{\partial s} \\ \\ \frac{\partial \mathbf{\Theta}}{\partial t} \\ \\ \frac{\partial \mathbf{\Theta}}{\partial r} \end{array} \right\} = \frac{1}{|\mathbf{J}|} \left[\begin{array}{ccc} \mathcal{J}_{11} & \mathcal{J}_{12} & \mathcal{J}_{13} \\ \\ \mathcal{J}_{21} & \mathcal{J}_{22} & \mathcal{J}_{23} \\ \\ \mathcal{J}_{31} & \mathcal{J}_{32} & \mathcal{J}_{33} \end{array} \right] \left\{ \begin{array}{c} \frac{\partial \mathbf{\Theta}}{\partial s} \\ \\ \frac{\partial \mathbf{\Theta}}{\partial t} \\ \\ \frac{\partial \mathbf{\Theta}}{\partial r} \end{array} \right\}$$

Using these definitions the flux components become

$$\begin{aligned} q_x = &-\frac{k_{xx}}{|\mathbf{J}|}\left(\mathcal{J}_{11}\frac{\partial \mathbf{\Theta}^T}{\partial s}\mathbf{T} + \mathcal{J}_{12}\frac{\partial \mathbf{\Theta}^T}{\partial t}\mathbf{T} + \mathcal{J}_{13}\frac{\partial \mathbf{\Theta}^T}{\partial r}\mathbf{T}\right) \\ &-\frac{k_{xy}}{|\mathbf{J}|}\left(\mathcal{J}_{21}\frac{\partial \mathbf{\Theta}^T}{\partial s}\mathbf{T} + \mathcal{J}_{22}\frac{\partial \mathbf{\Theta}^T}{\partial t}\mathbf{T} + \mathcal{J}_{23}\frac{\partial \mathbf{\Theta}^T}{\partial r}\mathbf{T}\right) \\ &-\frac{k_{xz}}{|\mathbf{J}|}\left(\mathcal{J}_{31}\frac{\partial \mathbf{\Theta}^T}{\partial s}\mathbf{T} + \mathcal{J}_{32}\frac{\partial \mathbf{\Theta}^T}{\partial t}\mathbf{T} + \mathcal{J}_{33}\frac{\partial \mathbf{\Theta}^T}{\partial r}\mathbf{T}\right) \end{aligned}$$

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

$$\begin{aligned} q_z = &-\frac{k_{zx}}{|\mathbf{J}|}\left(\mathcal{J}_{11}\frac{\partial \mathbf{\Theta}^T}{\partial s}\mathbf{T} + \mathcal{J}_{12}\frac{\partial \mathbf{\Theta}^T}{\partial t}\mathbf{T} + \mathcal{J}_{13}\frac{\partial \mathbf{\Theta}^T}{\partial r}\mathbf{T}\right) \\ &-\frac{k_{zy}}{|\mathbf{J}|}\left(\mathcal{J}_{21}\frac{\partial \mathbf{\Theta}^T}{\partial s}\mathbf{T} + \mathcal{J}_{22}\frac{\partial \mathbf{\Theta}^T}{\partial t}\mathbf{T} + \mathcal{J}_{23}\frac{\partial \mathbf{\Theta}^T}{\partial r}\mathbf{T}\right) \\ &-\frac{k_{zz}}{|\mathbf{J}|}\left(\mathcal{J}_{31}\frac{\partial \mathbf{\Theta}^T}{\partial s}\mathbf{T} + \mathcal{J}_{32}\frac{\partial \mathbf{\Theta}^T}{\partial t}\mathbf{T} + \mathcal{J}_{33}\frac{\partial \mathbf{\Theta}^T}{\partial r}\mathbf{T}\right) \end{aligned}$$

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

$$q_n = \mathbf{q} \cdot \mathbf{n}$$

$$\mathbf{q} = q_x\mathbf{e_x} + q_y\mathbf{e_y} + q_z\mathbf{e_z} \tag{6.3}$$
$$\mathbf{n} = n_x\mathbf{e_x} + n_y\mathbf{e_y} + n_z\mathbf{e_z}$$

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

$$\mathbf{e_1} = \left\{ \begin{array}{c} \frac{\partial x}{\partial s_s} \\ \\ \frac{\partial y}{\partial s_s} \\ \\ \frac{\partial z}{\partial s_s} \end{array} \right\} \quad ; \quad \mathbf{e_2} = \left\{ \begin{array}{c} \frac{\partial x}{\partial t_s} \\ \\ \frac{\partial y}{\partial t_s} \\ \\ \frac{\partial z}{\partial t_s} \end{array} \right\}$$

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 (4.40) 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 [43]. 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. Flux components computed directly from the boundary condition specification are also available from COYOTE. These values are not post-processed quantities but are computed during the element matrix generation phase and reported during the time stepping or iteration process.

When auxiliary variables are defined, the flux components for these variables may also be computed. As the process is completely analogous to the conduction process, no further detailed explanation is required.

6.3 Time Harmonic Functions

For time harmonic problems, the real and imaginary components of the temperature are the primary quantities produced by COYOTE. For purposes of analysis, other more useful forms of

the thermal response may be required. The real and imaginary temperatures may be combined to provide a modulus and phase angle at each nodal point which may be output to the post-processing file. These quantities are simply defined as

$$|\mathbf{T}| = \sqrt{\mathbf{T_R}^2 + \mathbf{T_I}^2} \tag{6.5}$$

$$\beta = \tan^{-1}\left(\frac{\mathbf{T_I}}{\mathbf{T_R}}\right) \tag{6.6}$$

The temperature at any point and time may be reconstructed from these fields and the use of the definition in (2.39).

The heat flux components for the periodic case may be computed directly from the definition in (6.1) for the real and imaginary components. That is

$$q_i^R = -k_{ij}\frac{\partial \mathbf{\Theta}^T}{\partial x_j}\mathbf{T_R} \tag{6.7}$$

$$q_i^I = -k_{ij}\frac{\partial \mathbf{\Theta}^T}{\partial x_j}\mathbf{T_I} \tag{6.8}$$

where the actual spatial derivatives are computed as in (6.2). For each component of the heat flux vector, a modulus and phase angle may be defined in a manner analogous to the temperature field.

$$|q_i| = \sqrt{(q_i^R)^2 + (q_i^I)^2} \tag{6.9}$$

$$\gamma = \tan^{-1}\left(\frac{q_i^I}{q_i^R}\right) \tag{6.10}$$

This data may be output to the post-processing file. Another useful form of the flux result would be the normal flux at a boundary. Using the definition in (6.4) real and imaginary compnents of the normal flux can be constructed; this can be followed by a construction of the modulus and phase angle for the flux normal to the boundary.

6.4 Heat Flow Function

For two-dimensional problems, Kimura and Bejan [44] 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

$$\begin{aligned} q_1 = q_x &= \rho C u_x T - k\frac{\partial T}{\partial x} = \frac{\partial \mathcal{H}}{\partial y} \\ q_2 = q_y &= \rho C u_y T - k\frac{\partial T}{\partial y} = -\frac{\partial \mathcal{H}}{\partial x} \end{aligned} \tag{6.11}$$

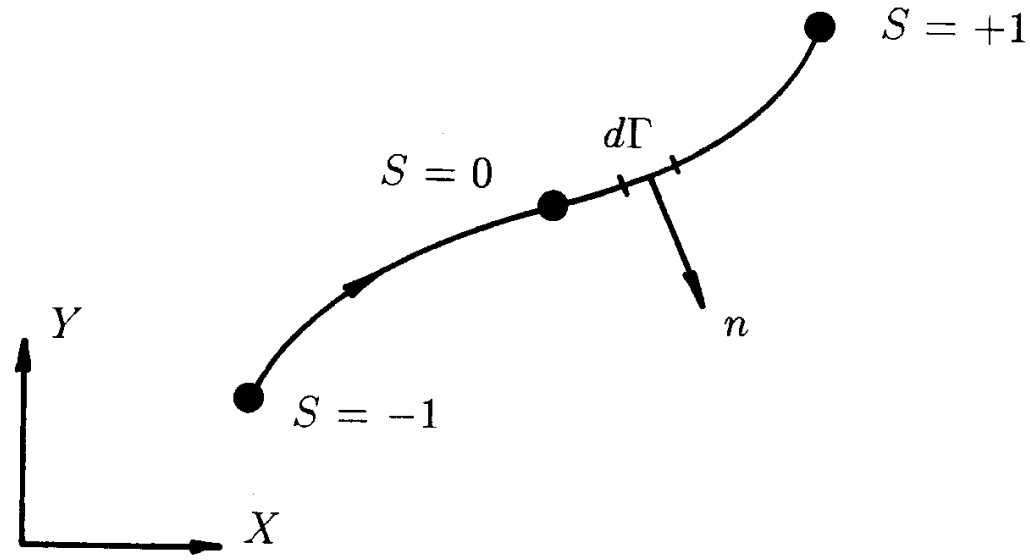

Figure 6.1: Definition of element boundary for heat function computation.

where the flux components have been defined as the total of the convective and diffusive fluxes. For simplicity the definitions in (6.11) have also assumed an isotropic conductivity though this is not a required restriction. In the usual applications considered here, the velocities in (6.11) 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

$$\delta\mathcal{H} = \int_A^B \mathbf{q}\cdot\mathbf{n}\, d\Gamma \tag{6.12}$$

$$\mathbf{q} = q_x\mathbf{e_x} + q_y\mathbf{e_y}$$

$$\mathbf{n} = n_x\mathbf{e_x} + n_y\mathbf{e_y}$$

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 (6.12) 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

$$q_x = \hat{\mathbf{\Phi}}^T\mathbf{q_x} \quad ; \quad q_y = \hat{\mathbf{\Phi}}^T\mathbf{q_y}$$

$$x = \hat{\mathbf{\Upsilon}}^T \mathbf{x} \quad ; \quad y = \hat{\mathbf{\Upsilon}}^T \mathbf{y} \tag{6.13}$$

where $\hat{\mathbf{\Phi}}$ and $\hat{\mathbf{\Upsilon}}$ 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

$$\mathbf{n} = \frac{1}{\Delta}\frac{\partial y}{\partial s}\mathbf{e_x} - \frac{1}{\Delta}\frac{\partial x}{\partial s}\mathbf{e_y} \tag{6.14}$$

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

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

or using the definitions of (6.13)

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

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

$$\delta\mathcal{H} = \int_{-1}^{+1} \left(\frac{\partial \hat{\mathbf{\Upsilon}}^T}{\partial s}\mathbf{y}\ \hat{\mathbf{\Phi}}^T\mathbf{q_x} - \frac{\partial \hat{\mathbf{\Upsilon}}^T}{\partial s}\mathbf{x}\ \hat{\mathbf{\Phi}}^T\mathbf{q_y}\right) ds. \tag{6.15}$$

The interpolation function definitions were described previously in Section 4.10.2; the function $\hat{\mathbf{\Upsilon}}$ 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 (6.15) once the element geometry, velocity and temperature fields are specified; the fluxes needed in (6.15) are derived from the definitions in (6.13) and the formulas outlined in the previous chapter. Computation of the heat function field for an entire finite element mesh is generated by applying (6.15) 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,

$$q_1 = q_r = \frac{1}{r}\frac{\partial \mathcal{H}}{\partial z} \quad ; \quad q_2 = q_z = -\frac{1}{r}\frac{\partial \mathcal{H}}{\partial r} \tag{6.16}$$

and

$$\mathbf{q} = q_r r\ \mathbf{e_r} + q_z r\ \mathbf{e_z}$$

$$\mathbf{n} = n_r \mathbf{e_r} + n_z \mathbf{e_z}.$$

6.5 Error Estimation

An error estimation procedure, based on the Zienkiewicz-Zhu theory [45], is available in COYOTE and utilizes the flux computation procedure outlined in a previous section. Following [45],

an appropriate norm for the energy Equation (2.1) is

$$\|T\|^2 = \int_\Omega (\nabla T \cdot \nabla T)\, d\Omega \tag{6.17}$$

and by definition an error in the computed temperature is defined as $\epsilon^T = T^e - T^h$ where the superscripts e and h refer to exact and computed, respectively. Using the definition for ϵ^T in (6.17), leads to a norm for the temperature error

$$\|\epsilon^T\|^2 = \int_\Omega \nabla(T^e - T^h) \cdot \nabla(T^e - T^h)\, d\Omega \tag{6.18}$$

A more convenient form for the error norm can be constructed if the original norm includes the thermal conductivity. In this case a similar procedure and Fourier's law produces an error norm on the flux

$$\|\epsilon^q\|^2 = \int_\Omega (\mathbf{q}^e - \mathbf{q}^h) \cdot (\mathbf{q}^e - \mathbf{q}^h)\, d\Omega \tag{6.19}$$

where $\mathbf{q}$ is a flux vector.

Since the exact (superscript e) values are not generally available, some method of approximating these terms is required. The Zienkiewicz-Zhu approach uses a local projection method to estimate these values at the nodes of an element. In particular, a local least-squares method is used for the patch of elements surrounding each node to recover a higher-order approximation to the flux components at the node. These values can then be compared with the flux values computed directly from the temperature solution; an element error indicator is then formed by integration over the element surface as indicated in Equation (6.19).

The recovered fluxes at the node are approximated by a polynomial expansion such as

$$q_i{}^e = \mathbf{\Lambda}^T \mathbf{a} = \{1\ x\ y\ xy\} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \tag{6.20}$$

where (6.20) has been written for an arbitrary component of the flux and the polynomial is for a typical two-dimensional case. The vector of coefficients $\mathbf{a}$ is computed from the least-squares problem

$$minimize\ I_i = \frac{1}{2V} \int_\Omega (q_i{}^e - q_i{}^h)^2\, d\Omega \tag{6.21}$$

for each coordinate direction i. The minimization problem in (6.21) leads to a matrix problem of the form

$$\left[\int_{\Omega_p} \mathbf{\Lambda}\mathbf{\Lambda}^T\, d\Omega \right] \mathbf{a} = \int_{\Omega_p} \mathbf{\Lambda} \sum_{gp} q_i{}^h\, d\Omega \tag{6.22}$$

for each flux component, where Ω_p is the area or volume of the patch of elements surrounding the node. The sum on the right-hand-side of (6.22) is over the number of Gauss (integration) points in the patch of elements surrounding the node. After solution of (6.22), the recovered (higher-order) fluxes at the nodes can be evaluated at the element integration points and compared to

the fluxes computed from the temperature interpolation functions. These differences are then integrated over the element to form the norm in (6.19) for each element. The element error values are normalized over the entire mesh and can be output to the post-processing file. Note that in the current version of COYOTE, the error estimation procedure is not operational in a parallel environment.

6.6 Species and Gas Fraction

For thermal problems with chemical reaction, chemical species are computed at the integration points of each element. For post-processing purposes, the integration point values for each of the species are averaged over the element and output to the post-processing file as element quantities. The reacted gas fraction can be computed directly from (2.28). This quantity is also described as constant over the element when output to the post-processing file. For a coupled thermo-mechanical analysis, the gas fraction may be transferred to the mechanical code for use in various constitutive models.

6.7 Element and Element Block Variables

Element quantities such as mass, volume and internal energy are computed from their basic definitions for every element in the mesh. These quantities may be written to the post-processing file as a user option. Similar quantities are available for each element block and may be referenced through the block id number. In addition, the maximum, minimum and average temperature for each element block is reported. The average block temperature is computed from the ratio of the block internal energy and the volume integral of the specific heat. Bulk node variables are recorded for post-processing and include the volume, temperature, pressure, mass and energy.

6.8 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 [42] that can be accessed by any of several graphics packages, such as the BLOT code [46], EnSight [47] or ParaView [48] software. Details of the output file are available in [5,42].

Chapter 7

References

1. COMSOL Multiphysics Users Guide, Version 3.5, COMSOL AB (2008), http://www.comsol.com

2. COSMOSWorks, Version 2.95, Solidworks Corp. (2008), http://www.cosmosm.com

3. ABAQUS User's Manual, Version 6.8, SIMULIA (2008), http://www.simulia.com

4. Thermal Synthesizer System Users Manual, Version 12.1, Spacedesign Corp. (2007), http://www.spacedesign.com

5. D. K. Gartling, R. E. Hogan and M. W. Glass, "COYOTE - A Finite Element Computer Program for Nonlinear Heat Conduction Problems, Part II - User's Manual," SAND2010-0714, Sandia National Laboratories, Albuquerque, NM (2010)

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. Del Guidice, 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. J. N. Reddy and D. K. Gartling, "*The Finite Element Method in Heat Transfer and Fluid Dynamics*," 2nd Edition, CRC Press, Boca Raton, FL (2001)

14. J. A. Sethian, "*Level Set Methods and Fast Marching Methods*, Cambridge University Press, Cambridge, UK (1999)

15. E. B. Becker, G. F. Carey and J. T. Oden, "*Finite Elements, An Introduction, Volume I*," Prentice-Hall, NJ (1981)

16. O. C. Zienkiewicz, "*The Finite Element Method*," McGraw-Hill, London, 3rd Edition (1977)

17. 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)

18. B. M. Irons, "Quadrature Rules for Brick Based Finite Elements," *Int. J. Num. Meth. Engng.*, **3**, 293-294 (1971)

19. L. M. Taylor and D. P. Flanagan, "PRONTO 3D - A Three-Dimensional Transient Solid Dynamics Program," SAND87-1912, Sandia National Laboratories, Albuquerque, NM (1989)

20. R. D. Cook, D. S. Malkus and M. E. Plesha, "*Concepts and Applications of Finite Element Analysis*," John Wiley and Sons, NY (1989)

21. 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)

22. 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)

23. 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)

24. 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)

25. J. I. Lin, "Bounds on Eigenvalues of Finite Element Systems," *Int. J. Num. Meth. Engng.*, **32**, 957-967 (1991)

26. P. R. Schunk and J. N. Shadid, "Iterative Solvers in Implicit Finite Element Codes," SAND92-1158, Sandia National Laboratories, Albuquerque, NM (1992)

27. R. S. Tuminaro, M. Heroux, S. A. Hutchinson, and J. N. Shadid, "AZTEC User's Guide, Version 2.1," SAND99-8801J, Sandia National Laboratories, Albuquerque, NM (1999)

28. M. Salaz, M. Heroux and D. Day, "Trilinos Tutorial," SAND2004-2189, Sandia National Laboratories, Albuquerque, NM (2007)

29. M. Heroux and J. Willenbring, "Trilinos Users Guide," SAND2003-2952, Sandia National Laboratories, Albuquerque, NM (2007)

30. R. L. Clay, K. D. Mish, I. J. Otero, L. M. Taylor and A. B. Williams, "An Annotated Reference Guide to the Finite-Element Interface (FEI) Specification, Version 1.0," SAND99-8229, Sandia National Laboratories, Albuquerque, NM (1999)

31. M. W. Glass, "CHAPARRAL V2.x - A Library for Solving Large Enclosure Radiation Heat Transfer Problems," SAND01-xxxx (in preparation), Sandia National Laboratories, Albuquerque, NM (2001)

32. 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)

33. R. E. Hogan and D. K. Gartling, "Solution Strategies for Coupled Conduction/Radiation Problems," *Comm. Numer. Meth. Engng.*, **24**, 523-542 (2008)

34. I. S. Wichman, "On the Use of Operator-Splitting Methods for the Equations of Combustion," *Combust. Flame*, **83**, 240-252 (1991)

35. T. R. Young, "CHEMEQ - A Subroutine for Solving Stiff Ordinary Differential Equations," NRL Memorandum Report 4091, Naval Research Laboratory, Washington, DC (1980)

36. D. R. Mott, E. S. Oran and B. van Leer, "A Quasi-Steady-State Solver for the Stiff Ordinary Differential Equations of Reaction Kinetics," *J. Comp. Phys.*, **164**, 407-428 (2000)

37. 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)

38. B. A. Hendrickson and R. Leland, "The Chaco User's Guide - Version 1.0," SAND93-2339, Sandia National Laboratories, Albuquerque, NM (1993)

39. G. L. Hennigan, M. St. John, and J. N. Shadid, "NEMESIS I: A Set of Functions for Describing Unstructured Finite Element Data on Parallel Computers," http://endo.sandia.gov/SEACAS/Documentation/Nemesis_Users_Guide.pdf, Sandia National Laboratories, Albuquerque, NM (1998)

40. W. Gropp, E. Lusk and A. Skjellum, "*Using MPI*," MIT Press, Cambridge, MA (1995)

41. S. Plimpton, S. Attaway, B. Hendrickson, J. Swegle, C. Vaughn and D. Gardner, "Transient Dynamics Simulations: Parallel Algorithms for Contact Detection and Smoothed Particle Hydrodynamics," *Proc. Supercomputing '96*, Pittsburgh, PA (1996)

42. L. A. Schoof and V. R. Yarberry, "EXODUS II - A Finite Element Data Model," SAND92-2137, Sandia National Laboratories, Albuquerque, NM (1994)

43. 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)

44. S. Kimura and A. Bejan, "The 'Heatline' Visualization of Convective Heat Transfer," *J. Heat Trans.*, **105**, 916-919 (1983)

45. O. C. Zienkiewicz and J. Z. Zhu, "A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis," *Int. J. Num. Meth. Engng.*, **24**, 337-357 (1987)

46. 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)

47. EnSight User Manual, Version 8.2, Computational Engineering International, (2006), http://www.ensight.com

48. ParaView Guide, Version 3.0, Kitware, Inc., (2008), http://www.kitware.com

Distribution:

1	MS0826	D. K. Gartling	1500
1	MS0836	R. E. Hogan	1514
1	MS0382	M. W. Glass	1541
1	MS0899	Technical Library	9536 (electronic copy)