SANDIA REPORT
SAND2020-4001 - Unlimited Release

Printed April 9, 2020

Sandia National Laboratories

SIERRA Multimechanics Module: Aria Thermal Theory Manual – Version 4.56

SIERRA Thermal/Fluid Development Team

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

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

ABSTRACT

Aria is a Galerkin finite element based program for solving coupled-physics problems described by systems of PDEs and is capable of solving nonlinear, implicit, transient and direct-to-steady state problems in two and three dimensions on parallel architectures. The suite of physics currently supported by Aria includes thermal energy transport, species transport, and electrostatics as well as generalized scalar, vector and tensor transport equations. Additionally, Aria includes support for manufacturing process flows via the incompressible Navier-Stokes equations specialized to a low Reynolds number ($Re < 1$) regime. Enhanced modeling support of manufacturing processing is made possible through use of either arbitrary Lagrangian-Eulerian (ALE) and level set based free and moving boundary tracking in conjunction with quasi-static nonlinear elastic solid mechanics for mesh control. Coupled physics problems are solved in several ways including fully-coupled Newton's method with analytic or numerical sensitivities, fully-coupled Newton-Krylov methods and a loosely-coupled nonlinear iteration about subsets of the system that are solved using combinations of the aforementioned methods. Error estimation, uniform and dynamic h-adaptivity and dynamic load balancing are some of Aria's more advanced capabilities.

ACKNOWLEDGMENTS

Aria represents a new generation code implementation of heat transfer algorithms. The mathematical models described herein would never have been implemented without the efforts of the Calore development team: Bob Cochran, Mike Glass, Randy Lober, Chris Newman, Steve Bova and Tolu Okusanya. Bruce Bainbridge, Ben Blackwell, Merlin Decker, Kevin Dowding and Vicente Romero have made significant contributions to the development and testing of Calore. Many of the algorithms implemented in Calore were first implemented in COYOTE, and therefore we are indebted to David Gartling, Roy Hogan, and others who contributed to its development. In particular, the algorithms associated with the chemistry were developed by Mel Baer. The basic capabilities of Calore were implemented in Aria by Pat Notz and Samuel Subia. Further implementation of new capabilities by the Sierra T/F code is a continuing effort.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This document describes the theoretical foundation of thermal analysis in Sierra Mechanics. The SIERRA Multimechanics Module: Aria, henceforth referred to as Aria for brevity, was developed at Sandia National Laboratories under the ASC program, and approximates linear and nonlinear continuum models of heat transfer. Aria uses the SIERRA Framework [8], which provides data management services commonly required by computational mechanics software, and facilitates the development of coupled, multi-mechanics applications for massively parallel computers. The mathematical models in Aria are based heavily on those of COYOTE, a well-established thermal analysis program that was also developed at Sandia [10, 11] and its ASC code predecessor, Calore [3]. Aria, Calore and COYOTE share a significant body of numerical methods, which are described in detail by Reddy and Gartling [23]. Throughout this document, the terms software and implementation are synonymous with the Aria thermal-fluid analysis computer program.

Whether one uses Aria to perform heat transfer analysis, or in developing a new capability for the Aria application, this document provides the information needed understand the existing numerical algorithm implementations. Justification for the fundamental assumptions of heat transfer, nor derivation of the energy conservation equations are included in this document. For a more thorough theoretical background, one is referred to one of the many available textbooks, e.g. [21, 17]. Another reference, which is freely available in downloadable electronic form, is Lienhard and Lienhard [18].

2. GOVERNING EQUATIONS

Aria, Sierra TF, was primarily developed to perform steady and unsteady thermal analysis of systems that consist of multiple solid materials, though it provides a limited capability for simulating fluid materials. Particular emphasis is given to systems that are associated with nuclear weapons components in normal and abnormal thermal environments, as well as certain manufacturing processes thereof. The main governing equations are the energy conservation equation in unsteady or steady form. A fundamental assumption of the mathematical models described herein is that the relevant length scales are large with respect to the molecular mean free path, so that the laws of continuum mechanics may be applied.

The governing equations and associated mathematical models may evolve with each release of Aria. It is not our intent to publish corresponding versions of this document and maintain a one-to-one correlation with each released version of Aria. Instead, we regard this document as a reference for all versions of Aria, and will note in the text if the implementation of a particular model equation is either obsolete or forthcoming at the time of writing. For a description of the features available in a particular version of Aria, the reader should consult the Aria Users' Reference Manual and Release Notes.

In this chapter, we begin with a brief discussion of a representative heat transfer problem, and present the conservation equation in differential form. These equations include terms for modeling volumetric heat sources and other, more specialized phenomena. Next, we discuss the set of initial and boundary conditions that are available in Aria.

To fix the notation, consider Figure 2.0-1, which is a schematic representation of a typical heat transfer problem. The entire domain is represented by Ω, which, for example, lies in three-dimensional coordinate space, with spatial coordinates $\mathbf{x}$. In this particular case, Ω consists of two separate subdomains, $\Omega = \Omega_1 \cup \Omega_2$. These subdomains may consist of different materials. The entire boundary of Ω is indicated by $\partial\Omega$, subject to one or more boundary conditions on subsections we denote with a subscript on Γ. For example, let Γ_q be that portion of $\partial\Omega$ along which a specified heat flux normal to the boundary is applied; similarly, let Γ_T be subject to an applied temperature; let the surface Γ_a be adiabatic (no heat flux); let Γ_r, be subject to an applied radiation heat flux; and let Γ_h be subject to a convective heat flux, which is modeled by Newton's law of cooling. Note that the boundary conditions are of two types: either the flux or the temperature is specified. Finally, the interface between Ω_1 and Ω_2 is denoted $\partial\Omega_{1-2}$. The interface conditions applied along a boundary such as $\partial\Omega_{1-2}$ are that both the temperature and the normal component of the heat flux are continuous. Given appropriate initial conditions, the problem is to determine the time-evolution of the temperature field.

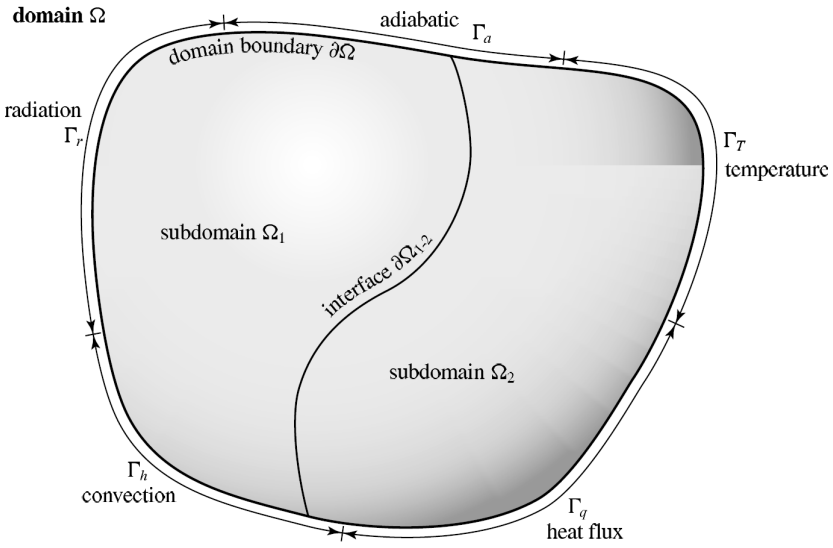

Figure 2.0-1.. A schematic diagram of the mathematical thermal model, showing the domain Ω; the subdomains Ω_i and their interfaces $\partial\Omega_{i-j}$; and the boundary conditions on the surface Γ.

2.1. CONSERVATION OF ENERGY

The conservation of energy within a solid material may be expressed as

$$\rho C_P \frac{\partial T}{\partial t} + \nabla \cdot \mathbf{q} = \dot{q}, \tag{2.1}$$

where T is the temperature, t is time, ρ is density, C_P is constant pressure specific heat, $\mathbf{q}$ is the heat flux vector, and $\dot{q}$ the volumetric heating. Note that very complex functional forms of the volumetric heating term are possible so that in general $\dot{q} = \dot{q}(\mathbf{x}, t, T)$. For example, as discussed in Section 3.1, it is possible to introduce non-diffusive chemical reactions so that $\dot{q}$ depends on additional chemical species variables, which are not represented in (2.1). In this case, additional conservation equations for the time evolution of the species must be solved. Fourier's Law of Heat Conduction expresses the heat flux as a function of temperature gradient, namely

$$\mathbf{q} = -\mathbf{K}\nabla T, \tag{2.2}$$

where $\mathbf{K} = k_{ij}$ denotes a thermal conductivity tensor. The corresponding conductivity matrix must be positive definite (in both the stationary and transient problems). For the case of an isotropic thermal conductivity (principal axes aligned with coordinate directions), the matrix is diagonal, with the element of row i and column j given by

$$k_{ij} = \begin{cases} k & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases} \tag{2.3}$$

and the heat flux model simplifies to

$$\mathbf{q} = -k\nabla T \ . \tag{2.4}$$

Upon substitution of equation (2.2) into (2.1), we obtain

$$\rho C_P \frac{\partial T}{\partial t} - \nabla \cdot (\mathbf{K} \nabla T) = \dot{q} \tag{2.5}$$

For fluid materials, energy of fluid motion characterized by the velocity vector $\mathbf{v}$ and pressure P gives rise to changes in temperature owing to deformation of the flow. Accounting for this energy leads to two different forms of the energy equation, one expressed in terms of C_v, the constant volume specific heat

$$\rho C_v \left(\frac{\partial T}{\partial t} + \mathbf{v} \cdot \nabla T \right) + T \left(\frac{\partial P}{\partial T} \right) \Bigg|_v \nabla \cdot \mathbf{v} - \nabla \cdot (\mathbf{K} \nabla T) = \dot{q}, \tag{2.6}$$

and another in terms of the constant pressure specific heat C_p

$$\rho C_P \left(\frac{\partial T}{\partial t} + \mathbf{v} \cdot \nabla T \right) - \rho T \left(\frac{\partial v}{\partial T} \right) \Bigg|_P \left(\frac{\partial P}{\partial t} + \mathbf{v} \cdot \nabla P \right) - \nabla \cdot (\mathbf{K} \nabla T) = \dot{q}, \tag{2.7}$$

where evaluations of the partial derivatives $\partial P / \partial T$ and $\partial v / \partial T$ require an equation of state for the fluid.

For constant density fluids the second term of equation(2.6) is zero and likewise for the second term of equation(2.7) by virtue of $\partial v / \partial T$. Furthermore for nearly incompressible flows and nearly constant pressure flows the second term of both equation(2.6) and (2.7) are zero. Hence in many cases one considers the energy transported by mechanism of convection by adding an additional term to equation (2.1) to obtain

$$\rho C_P \left(\frac{\partial T}{\partial t} + \mathbf{v} \cdot \nabla T \right) - \nabla \cdot (\mathbf{K} \nabla T) = \dot{q} \, . \tag{2.8}$$

We remark that if the ratio of convective forces to diffusive forces is large, then (2.8) becomes difficult to solve numerically. In many cases, specialized techniques for convection-dominated flows must be used [16, 23].

Typically, $\mathbf{v}$ is an unknown which is obtained from solving the mass and momentum conservation equations of fluid dynamics. These equations are outside the scope of the mathematical models described herein but are discussed in the Aria user manual. As long as the velocity field is divergence free, it may be regarded as input data and used in equation (2.8).

2.1.1. Statement of the Transient Problem

We are now in a position to state mathematically the initial–boundary value problem described at the beginning of this chapter. Let the domain Ω consist of N non-overlapping subdomains, each of which may be either solid or fluid. Let $\mathcal{S}$ be the set of subdomains in solid phase, and $\mathcal{F}$ the set of subdomains in fluid phase. Then the statement of the boundary value problem becomes, find the temperature

$T = T(\mathbf{x}, t)$, which satisfies

$$\rho C_P \frac{\partial T}{\partial t} - \nabla \cdot (\mathbf{K} \nabla T) = \dot{q} \qquad \forall\, \mathbf{x} \in \{\Omega_i \,|\, \Omega_i \in \mathcal{S}\}; t > t_0 \tag{2.9a}$$

$$\rho C_P \left(\frac{\partial T}{\partial t} + \mathbf{v} \cdot \nabla T \right) - \nabla \cdot (\mathbf{K} \nabla T) = \dot{q} \qquad \forall\, \mathbf{x} \in \{\Omega_j \,|\, \Omega_j \in \mathcal{F}\}; t > t_0 \tag{2.9b}$$

$$T(\mathbf{x}, t_0) = T_0(\mathbf{x}) \qquad \forall\, \mathbf{x} \in \Omega \tag{2.9c}$$

$$T(\mathbf{x}, t) = T_b(\mathbf{x}, t) \qquad \forall\, \mathbf{x} \in \Gamma_T \tag{2.9d}$$

$$q_n = f_b(\mathbf{x}, t, T) \qquad \forall\, \mathbf{x} \in \partial\Omega \setminus \Gamma_T \tag{2.9e}$$

$$[\![q_n]\!] = 0 \qquad \forall\, \mathbf{x} \in \{\partial\Omega_{i-k} \,|\, \Omega_i, \Omega_k \in \mathcal{S}\} \tag{2.9f}$$

Here q_n is the component of flux normal to a surface, where $q_n = \mathbf{q} \cdot \hat{\mathbf{n}}$, and $\hat{\mathbf{n}}$ is the outward unit normal vector. The notation $\partial\Omega \setminus \Gamma_T$ indicates the complement of Γ_T in $\partial\Omega$,

$$\partial\Omega \setminus \Gamma_T = \{\mathbf{x} \in \partial\Omega \mid \mathbf{x} \notin \Gamma_T\},$$

or the boundary of Ω excluding the surface Γ_T. In order for the problem to be well posed, it is not possible to specify both the flux and the temperature at the same location. Note that the initial condition T_0, the temperature boundary condition T_b, and the flux boundary condition f_b are usually specified in a piecewise manner over the subdomains and their boundaries. Various forms of the specified flux function $f_b(\mathbf{x}, t, T)$ are possible and will be described later in this chapter. For example, an adiabatic condition is specified if $f_b(\mathbf{x}, t, T) = 0$. The notation $[\![q_n]\!]$ represents the jump in the normal flux across a surface between two subdomains.

2.1.2. Statement of the Stationary Problem

For stationary, or steady-state problems, (2.1) may be simplified since the time derivative vanishes, by definition. Accordingly, for solid subdomains we obtain

$$-\nabla \cdot (\mathbf{K} \nabla T) = \dot{q}, \tag{2.10}$$

and for fluid subdomains, (2.8) reduces to

$$\rho C_P \mathbf{v} \cdot \nabla T - \nabla \cdot (\mathbf{K} \nabla T) = \dot{q} \tag{2.11}$$

Hence, we are led to the following boundary value problem statement: Given a domain Ω, which consists of solid and fluid subdomains as described in Section 2.1.1, find the temperature $T = T(\mathbf{x})$, which satisfies

$$-\nabla \cdot (\mathbf{K} \nabla T) = \dot{q} \qquad \forall\, \mathbf{x} \in \{\Omega_i \,|\, \Omega_i \in \mathcal{S}\} \tag{2.12a}$$

$$\rho C_P \mathbf{v} \cdot \nabla T - \nabla \cdot (\mathbf{K} \nabla T) = \dot{q} \qquad \forall\, \mathbf{x} \in \{\Omega_i \,|\, \Omega_i \in \mathcal{F}\} \tag{2.12b}$$

$$T(\mathbf{x}) = T_b(\mathbf{x}) \qquad \forall\, \mathbf{x} \in \Gamma_T \tag{2.12c}$$

$$q_n = F_b(\mathbf{x}, T) \qquad \forall\, \mathbf{x} \in \partial\Omega \setminus \Gamma_T \tag{2.12d}$$

$$[\![q_n]\!] = 0 \qquad \forall\, \mathbf{x}\{\partial\Omega_{i-k} \,|\, \Omega_i, \Omega_k \in \mathcal{S}\} \tag{2.12e}$$

We assume that all the symbols in the equations have the meaning known in calculus. A summary of notation and symbols is given in Table 9.0-1 and 9.0-2 on page 63. We assume, also, that the various fields and their derivatives are well behaved. Later, in Section 5.2, we provide some discussion on the requirements on the values and derivatives of both material properties and boundary conditions such that a temperature solution exists and is unique.

2.2. MATERIAL PROPERTIES

To avoid issues in applying these equations to domains with multiple materials, i.e., discontinuities in conductivity, finite element boundaries must always be aligned with material boundaries [2].

In the software implementation, the density, specific heat, and thermal conductivity are allowed to vary. Note that in equation (2.5) the product ρC_P may change with time, but its sign must not be allowed to change. For all material properties, predefined Aria modules handle constants, piecewise linear time-dependent and temperature-dependent functions, and a user subroutine interface allows completely arbitrary variation. Particularly in the case of a user subroutine, it is left to the user to meet smoothness and admissibility requirements.

2.2.1. Anisotropic Thermal Conductivity

In the previous section on governing equations the thermal conductivity was treated as being an isotropic tensor where the tensor principal directions were aligned with the coordinate (xy) axes. However, in a more general setting one will often know the principal values of conductivity in a material orientation $(x'y')$ as shown in Figure 2.2-1 rather than the coordinate axes. In this case provisions are made to transform the thermal conductivity into a coordinate frame consistent with the formulation.

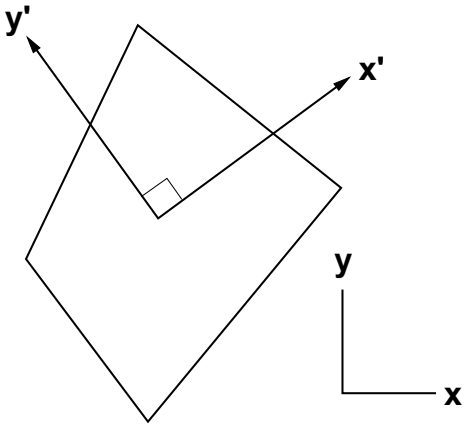

Figure 2.2-1.. Material thermal conductivity principle directions relative to computational axes directions.

Since the material orientation can vary spatially the transformation capability is supported for all elements of the meshed discretization thus the orientation and principal directions are more conveniently supplied from file. Alternatively, the transformation can be performed within a user subroutine. This capability is supported for both two and three dimensions.

3. VOLUMETRIC HEATING

In this chapter, we discuss the volumetric heating term, $\dot{q}$, which appears in the problem statements given in equations (2.9) and (2.12). This term may vary in space and time, and is defined piecewise over each subdomain Ω_i. Physically, $\dot{q}$ represents a local heat source or heat sink, expressed in the units of power per unit volume.

In Section 3.1 we discuss endothermic and exothermic chemical reactions, in Section 3.2 we discuss the concentrated point source, and in Section 3.3 we discuss the radiative source term.

3.1. CHEMICAL HEATING

Materials undergoing non–diffusive endothermic or exothermic chemical reactions can be modeled in Aria. The effect of such reactions is incorporated into the energy conservation equation as a volumetric heating term, which is calculated from the reaction rates. In this section, we give a brief overview of the equations implemented in Aria[1].

The volumetric heating due to the reactions may be written as

$$\dot{q}(\mathbf{x}, t, T) = \sum_{j=1}^{N_r} \mathfrak{R}_j r_j, \tag{3.1}$$

where $\mathfrak{R}_j$ is the known endothermic or exothermic energy release for step j of the reaction, r_j is the calculated reaction rate for the same step, and the summation over the index j is performed for the N_r reaction steps.

Step j of a multi–step chemical reaction can be expressed using the stoichiometric equation

$$\sum_{i=1}^{N_s} \nu'_{ij} M_i \rightarrow \sum_{i=1}^{N_s} \nu''_{ij} M_i, \tag{3.2}$$

where M_i is the chemical symbol for species i, ν'_{ij} is the stoichiometric coefficient of the reactant species i in reaction step j, ν''_{ij} is the stoichiometric coefficient of the product species i in reaction step j, and N_s represents the number of chemical species. When a species does not occur as a reactant or product in equation (3.2) for a particular reaction step, the corresponding stoichiometric coefficient is set to zero.

[1] Heat transfer in the presence of chemical reactions is detailed in Glassman's Combustion [13].

The net rate of change in the concentration of the species is determined from the following ordinary differential equation:

$$\frac{dN_i}{dt} = \sum_{j=1}^{N_r} (\nu''_{ij} - \nu'_{ij}) r_j, \tag{3.3}$$

where N_i is the concentration (or mole fraction) for species i. In equations (3.1) and (3.3), the reaction rates are calculated according to the law of mass action

$$r_j = k_j \prod_{i=1}^{N_s} N_i^{\mu_{ij}} \quad \text{for} \quad j = 1, 2, \ldots, N_r, \tag{3.4}$$

where k_j is the kinetic coefficient for reaction step j, and μ_{ij} is the given concentration exponent for reaction step j and species i. The kinetic coefficient, k_j, for each reaction step is usually determined from the Arrhenius equation, which may be written as

$$k_j = A_j \exp\left(\frac{-E_j}{RT}\right) T^{\beta_j}, \tag{3.5}$$

where A_j is the pre-exponential factor, E_j is the activation energy, R is the appropriate universal gas constant, T is the temperature and β_j is the steric coefficient.

Upon substitution of (3.5) and (3.4) into (3.3), we obtain a nonlinear system of ordinary differential equations for the evolution of the chemical species concentrations, namely

$$\frac{dN_i}{dt} = \sum_{j=1}^{N_r} \left[(\nu''_{ij} - \nu'_{ij}) A_j \exp\left(\frac{-E_j}{RT}\right) T^{\beta_j} \prod_{k=1}^{N_s} N_k^{\mu_{kj}} \right] \tag{3.6}$$

Equation (3.6), with appropriate initial conditions for the species concentrations, must be solved simultaneously with equation (2.9) to obtain the time evolution of the temperature field and species concentrations. This is a difficult coupled system of nonlinear equations to solve, because the time scales associated with the chemical reactions can be very different from the time scale associated with conduction. Furthermore, the time scale associated with one chemical reaction step may be very different from the next. An operator splitting strategy is used to decouple the concentration equations from the energy equation at every time step. More detail on this subject is presented in chapter 8.

Finally, the thermal properties of the chemical material are regarded as weighted averages of the N_s values associated with each species component. Another weighted average that is sometimes useful when interpreting results is the reacted gas fraction, f_{RG}, which is defined as the fraction of reacting material that exists in gas phase,

$$f_{RG} = \frac{(1 - X_c) \sum_{i=1}^{N_s} N_i g_i}{\sum_{i=1}^{N_s} N_i} \tag{3.7}$$

where X_c represents the condensed fraction for the reactive material, and the parameter g_i is defined as

$$g_i = \begin{cases} 1 & \text{if } N_i \text{ is a gas phase species} \\ 0 & \text{if } N_i \text{ is not a gas phase species} \end{cases} \tag{3.8}$$

3.2. POINT SOURCES

It is possible to specify a heating source or sink that is concentrated at a single point. In this case

$$\dot{q} = \dot{q}_0(t)\delta(\mathbf{x} - \mathbf{x}_0), \tag{3.9}$$

where $\dot{q}_0(t)$ is the magnitude of the point source and has units of power, $\mathbf{x}_0$ is the coordinates of the point at which the point source is located, and $\delta(\mathbf{x} - \mathbf{x}_0)$ is the Dirac delta function. Note that δ has units of the reciprocal of the volume. A point source of the form (3.9) has no real physical basis, since the power is concentrated at an infinitesimal point. Nevertheless, it is sometimes a useful model.

3.3. RADIATIVE SOURCES

Thermal energy may be transferred through radiation as well as conduction. Aria treats radiative transfer through the Simplified Spherical Harmonics (SPn) approximation [22], an asymptotic correction to the diffusion limit of the linear Boltzmann equation.

$$-\nabla \cdot \left(\frac{\mu_n^2}{\sigma_T} \nabla I_n \right) + \sigma_T I_n = \sigma_S 4\pi \sum_{m=1}^{(N+1)/2} w_m I_m + \sigma_A I_b \tag{3.10}$$

with the Mark boundary condition given by

$$-\frac{\mu_n}{\sigma_T}\nabla I_n \cdot \vec{n} = \frac{\epsilon}{2-\epsilon}\left(I_n - I_b\right) + \frac{1-\epsilon}{2-\epsilon}\left[\frac{\sum\left(I_k - \frac{\mu_k}{\sigma_T}\nabla I_k \cdot \vec{n}\right)\mu_k w_k}{\sum \mu_k w_k} - I_n + \frac{\mu_n}{\sigma_T}\nabla I_n \cdot \vec{n}\right] \tag{3.11}$$

where μ_n and w_n are the nodes and weights of the Gauss-Legendre quadrature rule respectively. Only $\mu_n > 0$ are included due to symmetry. I_b is the blackbody intensity given by $I_b = \frac{\sigma T^4}{\pi}$. σ_S, σ_A, and σ_T are the isotropic scattering, absorption, and extinction coefficients which are related by $\sigma_T = \sigma_A + \sigma_S$. The volumetric source is

$$\dot{q} = 4\pi\sigma_A \left(\sum_{m=1}^{(N+1)/2} w_m I_m - \frac{\sigma T^4}{\pi} \right) \tag{3.12}$$

4. INTERFACE AND BOUNDARY CONDITIONS

The initial–boundary value problem given in (2.9) and the boundary value problem given in (2.12) include specification of the temperature and flux on the external and internal surfaces. In this chapter we discuss the conditions that may be applied at these surfaces.

4.1. INTERFACE CONDITIONS

Heat conduction in a multi-material body, as shown in Figure 2.0-1, is a common occurrence in thermal models. At the interface between two subdomains Ω_i and Ω_j, we assume a zero jump in the heat flux in the direction normal to the interface,

$$[\![q_n]\!] = 0, \tag{4.1}$$

and one of two conditions involving the temperature.

4.1.1. Perfect Contact, or Tied Contact

The first and most common condition is to enforce continuity of temperature at the interface,

$$T\,|_{\partial\Omega_i} - T|_{\partial\Omega_j} = 0, \tag{4.2}$$

where the notation $u\,|_{\partial\Omega_i}$ indicates that the temperature is to be evaluated on the surface $\partial\Omega_i$, which is associated with the subdomain Ω_i.

4.1.2. Contact Resistance

The second condition allows for a model of imperfect contact between two solid surfaces, which can take account of surface roughness. Through such an interface, heat transfer follows different paths: effective conduction through solid-to-solid contact, poor conduction through gas-filled interstices, and inefficient thermal radiation across gaps.

We treat this contact by setting the "gap" flux across the interface proportional to the temperature drop,

$$q_n|_{\partial\Omega_i} - q_n|_{\partial\Omega_j} = h_c \left(T\,|_{\partial\Omega_i} - T|_{\partial\Omega_j} \right), \tag{4.3}$$

where the constant h_c is the contact conductance, which is similar to a heat transfer coefficient, e.g., it would have the units $\mathrm{W/m^2K}$. The value of contact conductance is dependent upon the following[1]:

- temperatures of the two materials at the contact surface;
- the materials in contact;
- surface finish and cleanliness;
- pressure at which the surfaces are forced together;
- the substance, or lack of it, in the interstitial spaces.

4.1.3. Surface Radiation

In the case where radiation heat transfer is present in one subdomain Ω_i but not in an adjacent subdomain Ω_j the radiative heat flux must be accounted for at the interface between Ω_i and Ω_j. In this case, the total heat flux must be conserved.

$$q_n|_{\partial\Omega_i} - q_n|_{\partial\Omega_j} = q_{rad}, \tag{4.4}$$

For the SPn approximation, the radiative heat flux is given by

$$q_{rad} = -4\pi \sum_{m=1}^{(N+1)/2} w_m \frac{\mu_m^2}{\sigma_T} \nabla I_m \tag{4.5}$$

4.2. BOUNDARY CONDITIONS

As mentioned in Section 2.1.1, either the temperature or the flux may be specified on a given boundary surface: it is mathematically incorrect to specify the temperature and the flux at the same location. For the temperature solution to be well-behaved, the boundary data should be smooth. This means, for example, that the boundaries themselves should not have re-entrant corners or discontinuities in the curvature. Also the applied temperatures and fluxes should be continuous. However, when performing thermal analysis on systems of practical interest, such restrictions are often difficult to meet. Generally speaking, the effects of such discontinuities are usually manifested as local singularities, or localized oscillations in the temperature field. In practical terms, this may mean sub-optimal convergence of the associated numerical method.

[1] More on the theory of contact conductance, and the values of conductances for typical surface finishes and moderate contact pressures can be found in [15, 18, 19].

4.2.1. Specified Temperature

The specified temperature boundary condition is probably the most straightforward boundary condition to apply. Since, by definition, it is not a function of the unknown temperature field, its complexity is limited to being a known function of space and time. The specified temperature must be continuous on the surface of each subdomain, $\partial\Omega_i$.

4.2.2. Specified Heat Flux

The heat flux boundary condition may be written as

$$q_n = -\mathbf{k}\nabla T \cdot \hat{\mathbf{n}} = q_b\left(\mathbf{x}, t, T\right), \tag{4.6}$$

where $q_b(\mathbf{x}, t)$ is a given function. Note that an adiabatic boundary condition defines a zero heat flux boundary condition:

$$q_n = 0 \tag{4.7}$$

4.2.3. Convection Heat Flux

Heat exchange that occurs across surfaces that are exposed to a fluid environment may be modeled using Newton's Law of Cooling,

$$q_n = h\left(T - T_r\right), \tag{4.8}$$

where h indicates the convection coefficient, and T_r represents the reference temperature. In the finite element literature, this boundary condition is often referred to as mixed, since neither the temperature nor the flux is known, but their combination is a known function.

Generally speaking, h and T_r are not independent. The convection coefficient, which is usually a function of position, time, surface temperature, reference temperature, and possibly other parameters, is often evaluated using a correlation (e.g. see [21]). The reference temperature, which is sometimes associated with the free-stream fluid temperature, or the boundary layer temperature, etc., may be a function of time or other variables. Usually it is a known quantity, because the fluid with which it is associated is modeled as an infinite reservoir. However, if the size of this reservoir is finite, then its temperature can be affected by the energy transfer across the surface in question. This situation is discussed in Section 4.2.6.

4.2.4. Surface Radiation

A surface may exchange energy with its surroundings through thermal radiation. Any incident surface radiation will be either transmitted, reflected or absorbed. Letting τ, ρ and α represent the fractions of the incident flux in each category then

$$1 = \tau + \rho + \alpha \tag{4.9}$$

and for no transmission

$$1 = \rho + \alpha \qquad or \qquad \rho = 1 - \alpha\ . \tag{4.10}$$

Using the Kirchhoff identity

$$\alpha = \epsilon \tag{4.11}$$

then the reflectance is

$$\rho = 1 - \epsilon \tag{4.12}$$

where ϵ is the emissivity.

In order to understand the radiative energy balance at a surface one considers the rate at which energy streams away from the surface, the radiosity, defined as

$$J = E_b + \rho G \tag{4.13}$$

where E_b is the blackbody emissive power and G is the irradiation. Substituting for the reflectance (4.12) then

$$J = \epsilon E_b + (1 - \epsilon)G \quad .$$

The surface flux q is the difference between the energy that radiates away and the incident energy

$$q = J - G \tag{4.14}$$

and substitution for the radiosity we find that

$$q = \epsilon(\sigma T^4 - G) \quad . \tag{4.15}$$

When G is derived from an external temperature interaction this boundary condition is often called far-field radiation, since it usually models the radiative transfer of energy between a surface and the external environment. However, the boundary condition is found to have more general utility when one considers its role in modeling radiative transfer between two surfaces, 1 & 2, as shown in Figure 4.2-1, where A_1 is analogous to $\partial\Omega_i$.

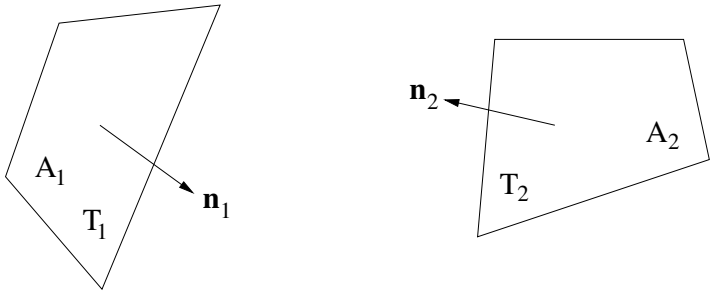

Figure 4.2-1.. Surface Radiative Exchange

For the case in which the temperature T_2 is known and independent of temperature T_1 then using the emissive power σT^4 the normal flux per unit area across A_1 may be written as

$$q = \sigma\epsilon F \left(T_1^4 - T_2^4\right), \tag{4.16}$$

where σ denotes the Stefan-Boltzmann constant, ϵ is the emissivity of the surface A_1 and F is the form factor. We remark that (4.16) is a nonlinear boundary condition, since the unknown temperature, T_1 is

raised to the fourth power and furthermore the emissivity may be a function of temperature. It is important to note that the form factor F may differ from the more familiar *view factor* F_{12} encountered in enclosure radiation problems. The question often asked is how does one determine the appropriate value of form factor F.

The form factor F can be best described using a network analogy of radiative transfer between two surfaces as shown below. Using the emitted energy E and radiosity J the network the heat flux can be

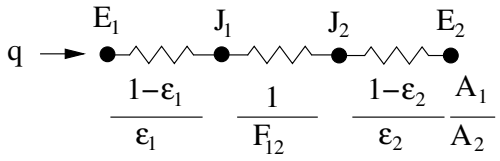

Figure 4.2-2.. Radiative Transfer Circuit Model

written in terms of a thermal resistance R as

$$q = \sigma \frac{T_1^4 - T_2^4}{R} \tag{4.17}$$

and from the network model

$$R = \frac{1-\epsilon_1}{\epsilon_1} + \frac{1}{F_{12}} + \frac{1-\epsilon_2}{\epsilon_2}\frac{A_1}{A_2} = \left(\frac{1}{\epsilon_1} - 1\right) + \frac{1}{F_{12}} + \left(\frac{1}{\epsilon_2} - 1\right)\frac{A_1}{A_2} \quad . \tag{4.18}$$

For $A_2 \gg A_1$ the third term of R can be neglected and $F_{12} = 1$. Comparing expressions (4.16) and (4.17) then

$$R = \frac{1}{\epsilon_1} \quad \text{and} \quad F = 1 \quad . \tag{4.19}$$

so estimation of F_{12} is not required.

For $\epsilon_2 = 1$ (black receiving surface) but A_2 not $\gg A_1$ and once again the third term of R can be neglected so that

$$R = \frac{1-\epsilon_1}{\epsilon_1} + \frac{1}{F_{12}} = \frac{(1-\epsilon_1)F_{12} + \epsilon_1}{\epsilon_1 F_{12}} \quad \text{and} \quad F = \frac{F_{12}}{(1-\epsilon_1)F_{12} + \epsilon_1} \quad . \tag{4.20}$$

If both surfaces are black $\epsilon_1 = \epsilon_2 = 1$ then from the previous expression (4.20) we find that $F = F_{12}$.

During a simulation the surface heat flux is integrated over the spatial discretization of surface A_1. Here we note that defining the flux on a per unit area basis enables us to apply the radiative flux (4.16) consistently with F evaluated for the entire surface A_1 even when the surface is discretized.

4.2.5. Enclosure Radiation

When energy radiates from one portion of a surface to another, and the intermediate medium is transparent(i.e., it does not absorb any energy), then enclosure radiation may be used to model the heat

flux on the surface. Using the net radiation method [25], the normal flux at a particular location on the surface may be written as the difference between the emitted radiative heat flux leaving the surface, and the absorbed incident radiative flux due to the rest of the enclosure, namely

$$q_n = \sigma \epsilon T^4 - \alpha G, \tag{4.21}$$

where α denotes the absorptivity of the surface, and G represents the surface irradiation. Under the additional assumption that the emissivity, absorptivity, and reflectivity are independent of direction and wavelength, we may write

$$\epsilon = \alpha = 1 - \rho, \tag{4.22}$$

where we have used the conventional symbol ρ for reflectivity. In this section, ρ always refers to reflectivity and not density.

Without loss of generality, we can regard the enclosure, $\Gamma_{\mathcal{E}}$, as a union of E surfaces,

$$\Gamma_{\mathcal{E}} = \Gamma_1 \cup \Gamma_2, \ldots \Gamma_{E-1} \cup \Gamma_E$$

This situation is illustrated in Figure 4.2-3, where the the radiosity for surface i in the enclosure is defined to be

$$J_i = \sigma \epsilon_i T_i^4 + \rho_i G_i, \tag{4.23}$$

where u_i is the spatially constant temperature on Γ_i. The surface irradiation for surface i is determined by the radiosity of all the other surfaces in the enclosure through the relation

$$G_i = \sum_{j=1}^{E} F_{ij} J_j, \tag{4.24}$$

where F_{ij} denotes the geometric viewfactor of surface i with respect to surface j. The viewfactor may be considered the fraction of energy that leaves surface i and arrives at surface j.

Upon substitution of equations (4.24) and (4.22) into (4.23), the radiosity may be written as

$$J_i - (1 - \epsilon_i) \sum_{j=1}^{N} F_{ij} J_j = \sigma \epsilon_i T_i^4, \tag{4.25}$$

Finally, the first J_i term in (4.25) may be moved inside the summation to yield

$$\sum_{j=1}^{N} \left[\delta_{ij} - (1 - \epsilon_i) F_{ij} \right] J_j = \sigma \epsilon_i T_i^4, \tag{4.26}$$

where

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{4.27}$$

(4.26) is a nonlinear system of equations for the radiosities that must be solved simultaneously with either (2.9) or (2.12). Finally, we may rewrite (4.21) to express the normal flux boundary condition on surface i as

$$q_n = \sigma \epsilon T^4 - \epsilon G_i, \tag{4.28}$$

where G_i is given by (4.24).

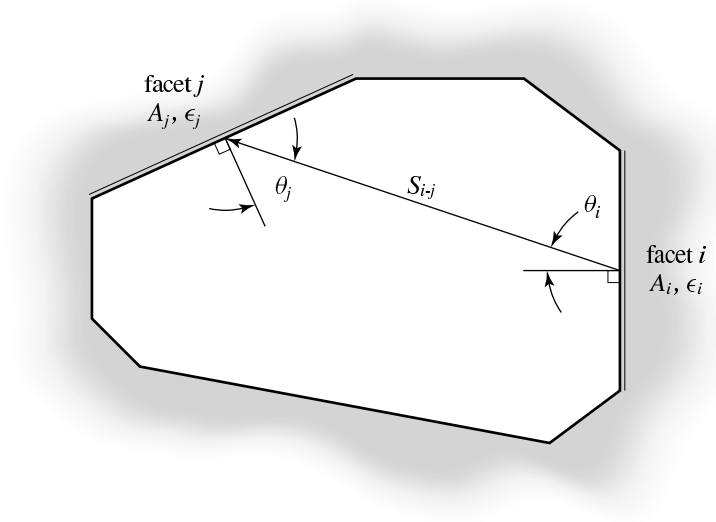

Figure 4.2-3.. Two arbitrary facets radiating energy to one another in a radiation enclosure. The energy exchanged depends on: the shape, orientation, distance, area A_i, A_j, temperatures T_i, T_j, and radiative properties of the facets ϵ_i, ϵ_j.

4.2.6. Bulk Fluid

The bulk fluid model is used in conjunction with the convective flux boundary condition given by (4.8). The idea is that the fluid reservoir to which the surface is attached is not infinite, so that the surface flux affects the reservoir temperature. In this case, the flux normal to the surface of the finite element mesh is given by

$$q_n = h\left(T_b - T\right), \tag{4.29}$$

where T_b represents the temperature of the bulk fluid, and T is again the local temperature of the surface. Note that this flux is equal in magnitude and opposite in sign to that in (4.8), since it is written with respect to the bulk fluid volume, instead of the finite element mesh. The bulk fluid temperature is an average temperature throughout the reservoir, which is obtained by solving the integral conservation equation

$$\frac{d}{dt}(V\rho cT_b) = -\int_{\partial V} q_n \, dA, \tag{4.30}$$

where ρ and c denote the density and the specific heat of the bulk fluid, respectively, and V indicates the bulk fluid volume. (4.30) states that the time rate of change of the bulk fluid energy is equal to the sum of the energy crossing the boundary. Note that we have implicitly made the flowing assumptions: there is no work due to pressure; there are no internal energy sources; there is no mass flow across ∂V. Upon substitution of (4.29), we obtain

$$\frac{d}{dt}(V\rho cT_b) + \int_{\partial V} h\left(T_b - T\right) \, d\Gamma = 0 \tag{4.31}$$

We remark that, although T_b is constant in space, T is allowed to vary across the interface separating the bulk fluid from the domain Ω. In summary, the flux boundary condition is defined by (4.29), which, since it involves T_b, must be solved simultaneously with (4.31).

For the stationary problem given by (2.12), (4.31) reduces to

$$\int_{\partial V} h\,(T_b - T)\; d\Gamma = 0. \tag{4.32}$$

5. DISCRETIZATION

In this chapter, we discretize the governing equations and their boundary conditions in space and time[1]. There are several approaches that can be used to accomplish this. Aria currently first semi-discretizes in space using the Galerkin method, and then discretizes in time using finite differences. Alternative approaches include first semi-discretizing in time using finite differences, then discretizing in space. Another possibility would be to discretize in space and time simultaneously using a space-time finite element formulation.

In Section 5.1, we use the method of weighted residuals[2] to obtain a weak statement for the stationary problem given by (2.12); we then use this weak statement to form the associated Galerkin approximations in Section 5.3. In Section 5.4, we form a similar weak statement for the transient problem given by (2.9); in Section 5.5, we discretize the resulting weak statement in space using Galerkin's method and use finite differences to discretize in time. We continue the analysis in chapter 6, where we introduce the finite element approximations.

Although the stationary and transient problem statements include separate partial differential equations for the fluid and solid subdomains, in this chapter, we consider only the energy equation for fluid subdomains since the equation for solid subdomains may be obtained by setting $T_i = 0$. The development in this chapter closely follows that of Becker, et al. [2]. The interested reader should consult this reference for a more detailed presentation.

5.1. WEAK STATEMENT OF THE STATIONARY PROBLEM

We begin with (2.11), which we rewrite as

$$\rho C_P \mathbf{v} \cdot \nabla T - \nabla \cdot (\mathbf{k} \nabla T) - \dot{q} = 0 \tag{5.1}$$

We emphasize that (5.1) is part of the problem statement given by (2.12), and is therefore defined over each subdomain Ω_ι. In fact, because of the possibility of discontinuities in the material properties at the interface between two subdomains, the second derivatives of u may not exist on such interfaces. This means that it is not possible to simply integrate (5.1) over the entire domain Ω. In fact, the lack of well–defined second derivatives is precisely the reason why this equation was written in each subdomain in the problem statement given by (2.12) in the first place. Accordingly, we begin by multiplying (5.1) by

[1] The development in this chapter closely follows that in Finite Elements: An Introduction, by Becker, et al. [2].

[2] This classical method is treated in depth in The Method of Weighted Residuals and Variational Principles, by Finlayson [9]

an arbitrary, admissible test function, w, and integrating the result over a single subdomain Ω_ι. This integral may be written as

$$\int_{\Omega_\iota} w \left[\rho C_P \mathbf{v} \cdot \nabla T - \nabla \cdot (\mathbf{k} \nabla T) - \dot{q}\right] \, dV = 0 \tag{5.2}$$

Next, in order to integrate the second term in (5.2) by parts, we introduce the following identity:

$$\int_{\Omega_\iota} \nabla \cdot (w \mathbf{k} \nabla T) \, d\Omega = \int_{\Omega_\iota} \nabla w \cdot \mathbf{k} \nabla T \, d\Omega + \int_{\Omega_\iota} w \nabla \cdot (\mathbf{k} \nabla T) \, d\Omega \tag{5.3}$$

Upon substitution of (5.3) into (5.2), we obtain

$$\int_{\Omega_\iota} (w \rho C_P \mathbf{v} \cdot \nabla T + \nabla w \cdot \mathbf{k} \nabla T - w \dot{q}) \, d\Omega - \int_{\Omega_\iota} \nabla \cdot (w \mathbf{k} \nabla T) \, d\Omega = 0 \tag{5.4}$$

Next, we introduce the Gauss Divergence Theorem, which converts a volume integral to a surface integral, and may be written as

$$\int_{\Omega_\iota} \nabla \cdot (w \mathbf{k} \nabla T) \, d\Omega = \int_{\partial\Omega_\iota} w \mathbf{k} \nabla T \cdot \hat{\mathbf{n}} \, d\Gamma \tag{5.5}$$

Upon substitution of (5.5) into (5.4) and rearranging terms, we obtain

$$\int_{\Omega_\iota} (w \rho C_P \mathbf{v} \cdot \nabla T + \nabla w \cdot \mathbf{k} \nabla T - w \dot{q}) \, d\Omega = - \int_{\partial\Omega_\iota} w q_n \, d\Gamma, \tag{5.6}$$

where we have substituted $q_n = -\mathbf{k} \nabla T \cdot \hat{\mathbf{n}}$.

The next step in obtaining the weak statement is to sum the contributions from (5.6) over each subdomain Ω_ι to obtain the integral over the entire domain, Ω. This is now valid, because no second derivatives appear in (5.6). Hence, we may write

$$\sum_{\iota=1}^{N} \int_{\Omega_\iota} (w \rho C_P \mathbf{v} \cdot \nabla T + \nabla w \cdot \mathbf{k} \nabla T - w \dot{q}) \, d\Omega = - \sum_{\iota=1}^{N} \int_{\partial\Omega_\iota} w q_n \, d\Gamma \tag{5.7}$$

The sum of the volume integrals on the left hand side of (5.7) may be readily combined into a single integral, namely

$$\sum_{\iota=1}^{N} \int_{\Omega_\iota} (w \rho C_P \mathbf{v} \cdot \nabla T + \nabla w \cdot \mathbf{k} \nabla T - w \dot{q}) \, d\Omega = \int_{\Omega} (w \rho C_P \mathbf{v} \cdot \nabla T + \nabla w \cdot \mathbf{k} \nabla T - w \dot{q}) \, d\Omega \tag{5.8}$$

The sum of the surface integrals on the right hand side of (5.7) requires more care. Consider that the surface integral associated with each subdomain Ω_ι consists of two parts. The first part is over the portion of $\partial\Omega_\iota$ which intersects the exterior surface $\partial\Omega$, which we denote $\partial\Omega_\iota \cap \partial\Omega$. The second part is

what is left over, and consists of internal surfaces that divide one subdomain from another, which we denote $\partial\Omega_\iota \setminus \partial\Omega$. The sum of surface integrals on the right hand side of (5.7) is therefore

$$\sum_{\iota=1}^{N} \int_{\partial\Omega_\iota} wq_n \, d\Gamma = \sum_{\iota=1}^{N} \int_{\partial\Omega_\iota \cap \partial\Omega} wq_n \, d\Gamma + \sum_{\iota=1}^{N} \int_{\partial\Omega_\iota \setminus \partial\Omega} wq_n \, d\Gamma \tag{5.9}$$

Now, the sum of the integrals over the exterior surfaces $\partial\Omega_\iota \cap \partial\Omega$ is simply the integral over the entire boundary surface $\partial\Omega$. The second sum in (5.9) reduces to the sum of the integrals of the flux jumps $[\![q_n]\!]$ over each interior interface[3]. Since the problem statement given in (2.12) specifies that these jumps are zero, (5.9) reduces to

$$\sum_{\iota=1}^{N} \int_{\partial\Omega_\iota} wq_n \, d\Gamma = \int_{\partial\Omega} wq_n \, d\Gamma \tag{5.10}$$

Upon substitution of equations (5.10) and (5.8) into (5.7), we have

$$\int_\Omega \left(w\rho C_P \mathbf{v} \cdot \nabla T + \nabla w \cdot \mathbf{k} \nabla T \right) \, d\Omega = -\int_{\partial\Omega} wq_n \, d\Gamma + \int_\Omega w\dot{q} \, d\Omega \tag{5.11}$$

Finally, we describe the class of admissible test functions w, which appear in (5.11). Clearly, the values of w and its first derivatives must exist so that the integrals in (5.11) are well–defined. Furthermore, the value of w should vanish on Γ_T, which we previously defined in Section 2.1.1 to be that portion of $\partial\Omega$ on which the temperature is specified.

Now, we can completely replace the set of differential equations and interface conditions given in (2.12) with the following alternative problem: Find the function $T(\mathbf{x})$, such that $T = T_b$ on Γ_T, $w = 0$ on Γ_T, and

$$\int_\Omega \left(w\rho C_P \mathbf{v} \cdot \nabla T + \nabla w \cdot \mathbf{k} \nabla T \right) \, d\Omega = -\int_{\partial\Omega \setminus \Gamma_T} wq_n \, d\Gamma + \int_\Omega w\dot{q} \, d\Omega \tag{5.12}$$

for all admissible w.

(5.12) is often called a weak statement of the problem given by (2.12) because the second derivatives of the temperature do not appear. More specifically, the original differential equation (5.1) requires that the solution $u(\mathbf{x})$ have second derivatives that exist in each subdomain Ω_ι, whereas (5.12) only requires that the first derivatives of $T(\mathbf{x})$ be integrable over the entire domain Ω. We remark that a solution to the original problem (2.12) is also a solution to the weak statement (5.12). However, the converse is not necessarily true: a solution to the weak statement is only a solution to the original problem if it is sufficiently smooth.

5.2. WELL-POSED PROBLEMS

In order that a unique solution exists and behaves itself, for both the stationary and transient problems, there are many requirements on the input data that must be satisfied. By input data we mean the supplied domain geometry, initial conditions, boundary conditions, and material coefficients. A fully

[3] The details on how the sum of the weighted fluxes on the subdomain boundaries reduces to the jumps is in Becker, et al. [2].

detailed enumeration of the requirements on the input data is beyond the scope of this manual, and some requirements fall in the realm of research. We have, however, hinted at some of the known requirements at various points in the text, and below in this Section we mention a couple other issues. This is an important topic, and we hope in a future edition to be able to cover it more fully.

As an aside, consider the meaning of the term well-posed problem. We say that a problem is well-posed if there exists a solution, it is unique, and it depends continuously on the data—otherwise the problem is said to be ill-posed. A well-posed problem is not always more physically realistic than an ill-posed one, and many times a well-posed problem may be unrealistic. As for the term stability, it is most often used to mean that the problem is "continuous with respect to the data". That is, if we change the problem slightly, the solution changes only slightly.

Regarding specific requirements on the heat source $\dot{q}$—for the stationary problem, in order that the solution exist, a compatibility condition between $\dot{q}$ and the applied boundary data must be satisfied. The compatibility condition is a statement of the global conservation principle for Ω. Its exact form depends upon the individual terms which appear in the differential conservation equations in (2.12), as well as the applied boundary data[4]. For the transient problem, any function $\dot{q}$ that is sufficiently smooth over each subdomain Ω_i is admissible, as long as it is finite and integrable in space and time.

When using a set of boundary conditions that do not specify temperature at any point, the solution (temperature field) is only defined up to an arbitrary constant, i.e., the solution is not unique. Aria may or may not be able to automatically check for this requirement.

Regarding the smoothness of input functions, we must assume that the necessary derivatives of input quantities exist in order that the solution, $T(\mathbf{x}, t)$, exists. In other words, there are restrictions that must be imposed on the smoothness of the various input data, including material properties and boundary conditions. The smoothness restrictions are stronger in the strong form of the equations (in chapter 2), and correspondingly weaker in the weak form of the equations discussed in this chapter. It is the weak form of the equations to which the Aria program actually tries to provide an approximate solution. No matter what the smoothness restrictions, Aria cannot reliably enforce them—especially inside user subroutines.

To summarize, it is primarily up to the user to provide valid admissible input data. Exercise caution to ensure a proper problem formulation.

5.3. GALERKIN APPROXIMATION FOR THE STATIONARY PROBLEM

Solutions to the weak statement given by (5.12) lie in a certain infinite-dimensional space of functions that have derivatives; we denote this space H. Galerkin's method seeks approximate solutions to the weak statement represented by a linear combination of a finite set of basis functions $\{\psi_1, \psi_2, \ldots, \psi_N\}$

[4] Again, more details on the compatibility condition may be found in Becker, et al. [2].

that defines a finite-dimensional subspace $H^h \subset H$. We then seek a function $T_h \in H^h$ of the form

$$T_h = \sum_{i=1}^{N} \alpha_i \psi_i, \tag{5.13}$$

which satisfies the weak form in H^h and where the α_i are unknown constants. If the ψ_i are Lagrange basis functions, then the constants $\alpha_i = T_i$: the constants correspond to the evaluation of the approximation at a node.

Upon substitution of (5.13) into (5.12), and allowing the test function w to be each element of H^h, we arrive at the discrete form of our weak statement: Find the function $T_h(\mathbf{x}) \in H^h$, such that $T_h = T_b$ on Γ_T and

$$\sum_{j=1}^{N} \left[\int_\Omega \left(\psi_i \rho C_P \mathbf{u} \cdot \nabla \psi_j + \nabla \psi_i \cdot \mathbf{k} \nabla \psi_j \right) d\Omega \right] \alpha_j = - \int_{\partial\Omega \setminus \Gamma_T} \psi_i q_n \, d\Gamma + \int_\Omega \psi_i \dot{q} d\Omega \tag{5.14}$$

for all $\psi_i, i = 1, \ldots, N$.

Here, (5.14) represents N fully discrete equations for the unknown constants T_j. For known flux functions $q_n \neq q_n(u)$, we may write this system of equations as the matrix system

$$\sum_{j=1}^{N} \left(U_{ij} + K_{ij} \right) \alpha_j = f_i, \quad i = 1, \ldots, N \tag{5.15}$$

where

$$U_{ij} = \int_\Omega \psi_i \rho C_P \mathbf{u} \cdot \nabla \psi_j \, d\Omega \tag{5.16a}$$

$$K_{ij} = \int_\Omega \nabla \psi_i \cdot \mathbf{k} \nabla \psi_j \, d\Omega \tag{5.16b}$$

$$f_i = - \int_{\partial\Omega \setminus \Gamma_T} \psi_i q_n \, d\Gamma + \int_\Omega \psi_i \dot{q} \, d\Omega \tag{5.16c}$$

The matrix U_{ij} represents the convection matrix, K_{ij} the diffusion matrix, and F_i is the forcing vector. Note that if $\mathbf{v} \neq 0$, then the matrix system in (5.15) is non-symmetric.

The finite element method provides a convenient, systematic way to construct the basis functions ψ_i. We address this important issue in chapter 6, in which we also discuss the case where q_n is a function of the temperature, wherein the essential difference is that the associated boundary integral has contributions to the matrix entries K_{ij} as well as the forcing function f_i.

5.4. WEAK STATEMENT OF THE TRANSIENT PROBLEM

The development of the weak statement for the time-dependent case closely follows that for the stationary case, which we developed in Section 5.1. We begin with the time-dependent energy conservation equation, which we rewrite here as

$$\rho C_P \left(\frac{\partial T}{\partial t} + \mathbf{v} \cdot \nabla T \right) - \nabla \cdot (\mathbf{k} \nabla T) - \dot{q} = 0 \tag{5.17}$$

The key difference between (5.17) and (5.1) is the introduction of the new independent variable, t, and the associated time derivative term. All of the discussion in Section 5.1 regarding subdomain interfaces, flux jumps, and solution smoothness is relevant for the present case. If the method of weighted residuals is applied to (5.17), for the problem statement given as (2.9), then the following weak statement is obtained: Find the function $T(\mathbf{x}, t)$, such that $T(\mathbf{x}, t_0) = T_0$, $T(\mathbf{x}, t) = T_b$ on Γ_T, and $w = 0$ on Γ_T, which satisfies

$$\int_\Omega \left(w \rho C_P \frac{\partial T}{\partial t} + w \rho C_P \mathbf{v} \cdot \nabla T + \nabla w \cdot \mathbf{k} \nabla T \right) d\Omega = - \int_{\partial\Omega \setminus \Gamma_T} w q_n \, d\Gamma + \int_\Omega w \dot{q} \, d\Omega \tag{5.18}$$

for all admissible w.

5.5. GALERKIN APPROXIMATION FOR TRANSIENT PROBLEM

As in Section 5.3, let the infinite-dimensional space of solution functions be denoted H. We again introduce a finite set of basis functions $\{\psi_1, \psi_2, \ldots, \psi_N\}$, which defines a finite-dimensional subspace $H^h \subset H$. Given a value of t, we then seek a function $T_h(\mathbf{x}, t) \in H^h$ of the form

$$T_h(\mathbf{x}, t) = \sum_{i=1}^{N} \alpha_i(t) \psi_i(\mathbf{x}) \tag{5.19}$$

which satisfies the weak form in H^h, and where the $\alpha_i(t)$ are unknown, time-dependent coefficients. If the ψ_i are Lagrange basis functions, then the parameters $\alpha_i(t) = T_i(t)$: the interpolation parameters correspond to the evaluation of the interpolation function at a given node. We assume that t is bounded,

$$t_0 \leq t \leq t_1 < \infty \tag{5.20}$$

Upon substitution of (5.19) into (5.18), and setting $w = \psi_i$, we arrive at the semi-discrete form of our weak statement: Find the function $T_h(\mathbf{x}, t) \in H^h$, where $t_0 \leq t \leq t_1$, such that $T(\mathbf{x}, t_0) = T_0$, $T_h(\mathbf{x}, t) = T_b$ on Γ_T, and

$$\begin{aligned}
&\sum_{j=1}^{N} \left[\int_\Omega \rho C_P \psi_i \psi_j d\Omega \right] \dot{\alpha}_j(t) \\
&\qquad + \sum_{j=1}^{N} \left[\int_\Omega \left(\psi_i \rho C_P \mathbf{u} \cdot \nabla \psi_j + \nabla \psi_i \cdot \mathbf{k} \nabla \psi_j \right) d\Omega \right] \alpha_j(t) \\
&\qquad = - \int_{\partial\Omega \setminus \Gamma_T} \psi_i q_n \, d\Gamma + \int_\Omega \psi_i \dot{q} \, d\Omega
\end{aligned} \tag{5.21}$$

for each $\psi_i, i = 1, \ldots, N$, and where $\dot{\alpha}(t)$ indicates the time derivative of $\alpha(t)$.

Here, (5.21), represents N ordinary differential equations for the N unknown functions $\alpha_j(t)$. For known flux functions $q_n \neq q_n(u)$, we may write this system of equations as the matrix system

$$\sum_{j=1}^{N} M_{ij} \dot{\alpha}_j(t) + \sum_{j=1}^{N} \left(U_{ij} + K_{ij} \right) \alpha_j(t) = f_i, \quad i = 1, \ldots, N \tag{5.22}$$

where

$$M_{ij} = \int_\Omega \rho C_P \psi_i \psi_j d\Omega \tag{5.23a}$$

$$U_{ij} = \int_\Omega \psi_i \rho C_P \mathbf{v} \cdot \nabla \psi_j \, d\Omega \tag{5.23b}$$

$$K_{ij} = \int_\Omega \nabla \psi_i \cdot \mathbf{k} \nabla \psi_j \, d\Omega \tag{5.23c}$$

$$f_i = - \int_{\partial\Omega \setminus \Gamma_T} \psi_i q_n \, d\Gamma + \int_\Omega \psi_i \dot{q} \, d\Omega \tag{5.23d}$$

The matrix M_{ij} is often referred to as the mass or capacitance matrix. Note that if $\mathbf{v} \neq 0$, then the matrix system in (5.22) is non-symmetric. Also, if any of the parameters in (5.23a-b), such as the material properties, applied fluxes, or volumetric heating are time dependent, then the corresponding matrices or vectors will also be time dependent. The system of ordinary differential equations given in (5.22) must be numerically integrated in time. We discuss this issue in Section 5.6.

5.6. TIME DISCRETIZATION

There are many ways to integrate the system of ordinary differential equations given in (5.22). The method employed by Aria uses is usually referred to as the variable-implicit method or theta-method. Aria has the ability to adaptively step the ODE to reduce local error with respect to time, but we emphasize that this says nothing about the global error in the ODE, i.e., the tolerance on the local error is not a tolerance on the total accuracy of the ODE solution. More significantly, we should point out

that a small error in the temporal discretization does not necessarily imply that there is a small error in the spatial discretization.

We begin the description of Aria's time integration strategy by rewriting (5.22) as

$$\mathbf{M}(t)\dot{\mathbf{T}}(t) + \hat{\mathbf{K}}(t)\mathbf{T}(t) - \mathbf{f}(t) = 0, \tag{5.24}$$

where $\dot{\mathbf{T}}(t)$ and $\mathbf{T}(t)$ are the vectors of entries $\dot{\alpha}_j(t)$ and $\alpha_j(t)$, respectively, and where $\mathbf{M}$ is the matrix with entries given by (5.23a), $\hat{\mathbf{K}}$ is the matrix with entries $\hat{K}_{ij} = U_{ij} + K_{ij}$, and $\mathbf{f}$ is the vector with entries given by (5.23d).

Numerical time integration of a partial differential equation requires both a discretized form of the governing equation and a time integrator. The discrete form of (5.24) for time level n can be written as

$$\mathbf{M}^n\dot{\mathbf{T}}^n + \hat{\mathbf{K}}^n\mathbf{T}^n - \mathbf{f}^n = 0. \tag{5.25}$$

A time integrator requires that the discrete approximation of $\mathbf{T}(t)$ be expressed as a function of calculated data and one choice of this approximation is

$$\mathbf{T}^{n+1} = f(\mathbf{T}^{n+1}, t^{n+1}, \mathbf{T}^n, t^n, \mathbf{T}^{n-1}, t^{n-1}, \ldots) \tag{5.26}$$

where f is often a function of computed derivatives of $\mathbf{T}$.

One form of (5.26) used in Aria is an implicit method which employs

$$\mathbf{T}^{n+1} = \mathbf{T}^n + \Delta t(1-\theta)\dot{\mathbf{T}}^n + \theta\Delta t\dot{\mathbf{T}}^{n+1} \tag{5.27}$$

where Δt is the time step size and θ denotes the evaluation level within the time step, $\theta = 0 \Rightarrow t(n)$ and $\theta = 1 \Rightarrow t(n+1)$.

The degree of implicitness of a given time discretization of (5.24) is determined by the time level at which $\hat{\mathbf{K}}(t)\mathbf{T}(t)$ is evaluated—if at the old time level, then the scheme is said to be explicit, if at the new time level, then fully implicit, and if at some average of the two, then semi-implicit. Combining (5.27) with (5.25) at time levels n and $n+1$ one obtains a variable implicit method

$$\left(\frac{1}{\Delta t^n}\mathbf{M} + \theta\hat{\mathbf{K}}^{n+1}\right)\mathbf{T}^{n+1} = \theta\mathbf{f}^{n+1} + (1-\theta)\mathbf{M}\dot{\mathbf{T}}^n + \frac{1}{\Delta t^n}\mathbf{M}\mathbf{T}^n \tag{5.28}$$

The case of $\theta = 1$ corresponds to Euler implicit, $\theta = 1/2$ corresponds to the trapezoid rule, and $\theta = 2/3$ corresponds to Galerkin implicit. In general, this method is accurate only to first order in time, but the special case of $\theta = 1/2$ can be shown to be to be second-order accurate. It is important to note that if $\hat{\mathbf{K}}$ or $\mathbf{f}$ depend upon $\mathbf{T}$, then (5.28) is nonlinear.

The time derivative $\dot{\mathbf{T}}^n$ can be calculated using (5.27) as

$$\dot{\mathbf{T}}^n = \frac{\mathbf{T}^n - \mathbf{T}^{n-1}}{\theta\Delta t^{n-1}} - \frac{1-\theta}{\theta}\dot{\mathbf{T}}^{n-1}\ . \tag{5.29}$$

Note that for $\theta = 1$, $\dot{\mathbf{T}}^n$ is not needed for evaluation of (5.28). If $\theta \neq 1$, then the recursive nature of (5.29) introduces some difficulty to the solution scheme since $\dot{\mathbf{T}}^{n-1}$ is not initially known. To circumvent this potential issue, Aria uses the fully-implicit case ($\theta = 1$) and (5.28) becomes

$$\left(\frac{1}{\Delta t^n}\mathbf{M} + \hat{\mathbf{K}}^{n+1}\right)\mathbf{T}^{n+1} = \mathbf{f}^{n+1} + \frac{1}{\Delta t^n}\mathbf{M}\mathbf{T}^n \tag{5.30}$$

or as one might expect

$$\frac{1}{\Delta t^n}\mathbf{M}\dot{\mathbf{T}}^{n+1} + \hat{\mathbf{K}}^{n+1}\mathbf{T}^{n+1} - \mathbf{f}^{n+1} \ . \tag{5.31}$$

Thus in the fully-implicit formulation one solves the discretized set of equation using information from the current time plane with a selected representation of the time derivative.

In the simplest case one begins with an initial guess for $\mathbf{T}^n$ and uses the first-order accurate time derivative known as BDF1 (Backward Finite Difference 1).

$$\dot{\mathbf{T}}^{n+1} = \frac{\mathbf{T}^{n+1} - \mathbf{T}^n}{\Delta t^n} \tag{5.32}$$

Another choice for the time derivative $\dot{\mathbf{T}}^n$ is the second-order approximation known as BDF2 (Backward Finite Difference 2)

$$\dot{\mathbf{T}}^{n+1} = \frac{3\mathbf{T}^{n+1} - 4\mathbf{T}^n + \mathbf{T}^{n-1}}{2\Delta t} \ . \tag{5.33}$$

Obviously utilization of the above requires that the first two solution steps of (5.30) be obtained using the first-order derivative (5.32). Theoretical time step bounds are available for the BDF2 time integrator and the scheme is often used in time accurate simulations.

While it is possible to use a fixed, constant Δt while iterating (5.30), it is often preferable to allow the timestep size to vary. Aria uses an automatic timestep size selection algorithm that allows Δt to vary while satisfying several constraints[5]. The approach first calculates a candidate timestep size, using local time truncation error estimates, then adjusts this value using several heuristic rules.

To calculate the candidate timestep size, assume that two time integrators of comparable accuracy are available. If we compare local truncation error estimates for each scheme, and if we have a target value of a truncation error norm in mind, then we can estimate Δt. The implicit scheme given by (5.28) is one such integrator, but a second is needed. Since this second scheme is only required for the time step selection algorithm, computational speed is important, and therefore we consider only explicit schemes. If θ is chosen so that the implicit scheme is first-order accurate in time, then we use forward Euler differencing, which may be written as

$$\mathbf{T}_p^{n+1} = \mathbf{T}^n + \Delta t^n \dot{\mathbf{T}}^n \tag{5.34}$$

[5] This algorithm is based on the approached described by Gresho, et al. [14], and is similar to that which is implemented in COYOTE [10, 11].

where $\mathbf{T}_p$ indicates the temperature that is predicted as a result of applying (5.34), and the acceleration vector $\dot{\mathbf{T}}^n$ is calculated using (5.29). For $\theta = 1/2$, a second-order accurate explicit scheme is required, and the Adams–Bashforth difference formula is used, namely

$$\mathbf{T}_p^{n+1} = \mathbf{T}^n + \frac{\Delta t^n}{2}\left[\left(2 + \frac{\Delta t^n}{\Delta t^{n-1}}\right)\dot{\mathbf{T}}^n - \frac{\Delta t^n}{\Delta t^{n-1}}\dot{\mathbf{T}}^{n-1}\right] \tag{5.35}$$

where the acceleration vectors are calculated using $\theta = 1/2$ in (5.29).

For $\theta > 1/2$, the candidate time step size, $(\Delta t)_c$, is estimated according to

$$(\Delta t)_c^{n+1} = \Delta t^n \sqrt{\frac{\theta\varepsilon}{(\theta - 1/2)\|\mathbf{T}^{n+1} - \mathbf{T}_p^{n+1}\|}} \tag{5.36}$$

where ε is the given, allowable value of the truncation error norm[6]. A typical value of the truncation error norm would be 0.0001. Here, the norm $\|\cdot\|$ indicates the non-dimensional root mean square norm of a vector,

$$\|\mathbf{v}\| = \frac{1}{v_{\max}}\sqrt{\frac{1}{N}\sum_{i=1}^{N} v_i^2} \tag{5.37}$$

where

$$v_{\max} = \max_{1\le i\le N} |v_i| \tag{5.38}$$

Note that in (5.36), the timestep grows as the square root of ε. For $\theta = 1/2$, the candidate timestep size is calculated according to the second-order formula

$$(\Delta t)_c^{n+1} = \Delta t^n \sqrt[3]{3\left(1 + \frac{\Delta t^{n-1}}{\Delta t^n}\right)\frac{\varepsilon}{\|\mathbf{T}^{n+1} - \mathbf{T}_p^{n+1}\|}} \tag{5.39}$$

Note that in (5.39), the timestep grows with the cube root of ε. We emphasize that the initialization problem caused by the recursive nature of (5.35) is circumvented by using the initial Δt with $\theta = 1$ for the first two timesteps, then switching to $\theta = 1/2$ and using (5.39) to calculate Δt thereafter.

In practice, it is useful to limit the candidate timestep size calculated by the above procedure. Aria imposes three limits on the timestep. The first limit is enforced only when there is volumetric heating due to chemical reactions, as their effect is not directly included in either (5.36) or (5.39). Aria uses the CHEMEQ library [26] to integrate (8.3) over the time interval. CHEMEQ uses a stiff ordinary differential equation integrator, which adaptively subcycles species equations according to their stiffness. Usually, the chemical time scales are much shorter than that of conduction. Hence, if $(\delta t)_s^n$ represents the minimum timestep size used to integrate the chemical species at timestep n, then we limit $(\Delta t)_c^{n+1}$ according to

$$(\Delta t)_c^{n+1} = \min\left((\Delta t)_c^{n+1}, \chi(\delta t)_s^n\right) \tag{5.40}$$

where χ is a multiplication factor, typically on the order of 100.

[6] We do not derive the local truncation error estimates here, since the method of using Taylor series analysis for this purpose is well established, for example see [14].

The second limit on the timestep allows control of how fast the solution changes from one timestep to the next. Let $|\Delta u|_m$ denote the maximum change in temperature at a single mesh node from timestep n to $n+1$, and $|\Delta T|_a$ denote the allowable value of this difference. Then we enforce

$$\text{if} \quad \left(|\Delta T|_m \frac{(\Delta t)_c^{n+1}}{\Delta t^n} > |\Delta T|_a \right) \quad \text{then} \quad (\Delta t)_c^{n+1} = \frac{0.95 |\Delta T|_a \Delta t^n}{|\Delta T|_m} \tag{5.41}$$

(5.41) states that if the estimated maximum change in temperature exceeds the allowable change, then the candidate timestep size is set to 5% less than the value required to prevent this condition.

Finally, the user is allowed to specify minimum and maximum values of the timestep size, Δt_{min} and Δt_{max}, respectively. For the BDF2 method a theoretical limit of $\Delta t_{\text{max}} = (1+\sqrt{2})\Delta t^n$ is also imposed. Therefore, we also restrict the candidate timestep size so that

$$\Delta t_{\text{min}} \leq (\Delta t)_c^{n+1} \leq \Delta t_{\text{max}} \tag{5.42}$$

In summary, the candidate timestep size is first estimated using (5.36) for $\theta > 1/2$ and (5.39) for $\theta = 1/2$. Next, each of the applicable limits given in (5.40)- (5.42) are enforced to determine the timestep size for the next iteration of (5.28), Δt^{n+1}.

6. FINITE ELEMENT APPROXIMATIONS

In chapter 5, we discretized the governing equations in space using Galerkin's method. In this chapter, we discuss two important, unresolved issues. The first issue is the method for constructing the basis functions ψ_i, which appear in the weak statements given by Equations (5.14) and (5.21). The second issue is the calculation of the associated integrals over the domain and its boundary. The finite element method solves both of these issues, and is concisely described by Becker, et al [2]:

> The finite element method provides a general and systematic technique for constructing basis functions for Galerkin approximations of boundary value problems. The main idea is that the basis functions [...] can be defined piecewise over subregions of the domain called finite elements and that over any subdomain the [basis functions] can be chosen to be very simple functions such as polynomials of low degree.

If the finite elements are also chosen to have convenient shapes (like triangles and quadrilaterals) for calculating integrals, then the integration over the entire domain is also facilitated.

To fix ideas, consider Figure 6.0-1, which shows a one-dimensional domain partitioned into four finite elements. In Figure 6.0-1(a), the global, piecewise linear basis function for node 3, ψ_3, is shown as the standard "hat" function. This function has a value of unity at node 3 and zero in elements not connected to node 3. The finite element approach to constructing ψ_3 is shown in Figure 6.0-1(b). On each element, two linear functions ψ_i^e are defined, $e \in \{1, 2, 3, 4\}$, $i \in \{1, 2\}$. If we add ψ_2^2 from the left element to ψ_1^3 from the right element, then we see that the global basis function ψ_3 is constructed.

Now, consider the global stiffness matrix $\mathbf{K}$, with row i and column j defined by (5.23c). We now rewrite this integral as the sum over each finite element, namely

$$K_{ij} = \sum_{e=1}^{E} \int_{\Omega^e} \nabla\psi_i \cdot \mathbf{k}\nabla\psi_j \, d\Omega \tag{6.1}$$

In (5.23c), the global basis function ψ_i is nonzero only on those elements which support node i. Hence we can consider the contributions to the global stiffness matrix from element e, and define

$$K_{ij}^e = \int_{\Omega^e} \nabla\psi_i^e \cdot \mathbf{k}\nabla\psi_j^e \, d\Omega \tag{6.2}$$

There is a subtle difference in the interpretation of the indices in (6.1) as opposed to (6.2). In (6.1), the indices i and j are global, and span all nodes in the finite element mesh. In (6.2), the indices i and j are local, and only span the number of element shape functions. Aria calculates the element stiffness matrix contributions defined in (6.2) and assembles the results into the correct locations in the global stiffness matrix using a mapping from the local node numbers of a given element to the associated global node

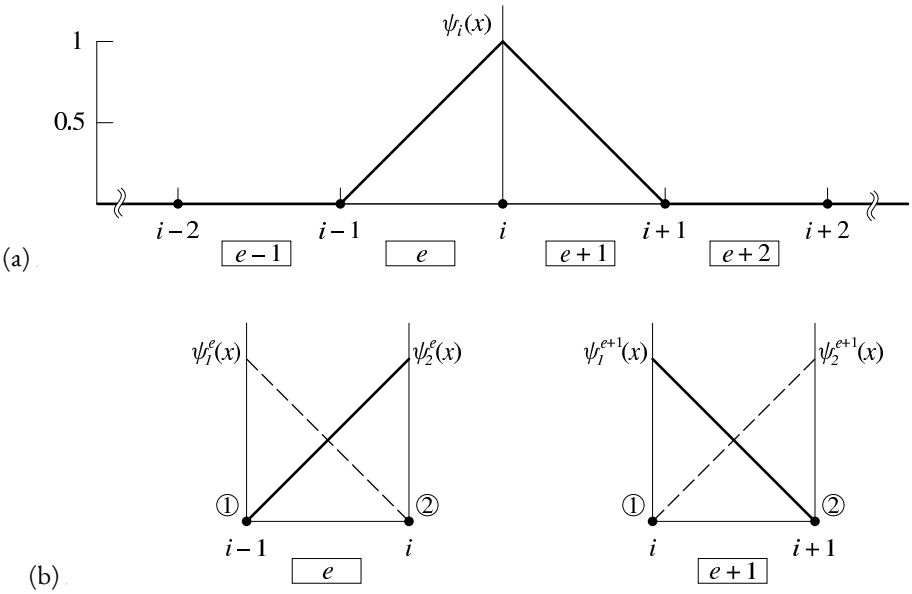

Figure 6.0-1.. A linear finite element global basis function at a node is formed from the nodal shape functions of two neighboring elements. (a) The global basis function for node i has support over the two elements e and $e+1$ that share the node, and takes on the value of zero everywhere else. (b) The associated element shape functions, ψ_2^e (on local node 2 of element e), and ψ_1^{e+1} (on local node 1 of element $e+1$), are combined to define the global basis function for node i.

numbers. In fact, each of the integrals in the weak statements derived in Chapter 5 involves a basis function and is calculated in the same way.

In Section 6.1, we discuss how the integrals are computed on a master element. Next, in Section 6.2, we discuss the master element interpolation functions for the elements used in Aria.

6.1. ELEMENT INTEGRATION

Element integrals of the form given by (6.2) are defined on the element coordinates in physical space. When calculating these quantities numerically, it is convenient to transform these integrals from physical space to a reference element, also called a master element, in computational space. This transformation is performed locally, for each finite element in the mesh. In this section we discuss the details of this process for two-dimensional physical space in Cartesian coordinates. The approach is completely general, and extensible to three-dimensional space and other coordinate systems. This situation is illustrated in Figure 6.1-1, showing such a transformation from the quadrilateral element, which spans the region $[-1,1] \times [-1,1]$, in master coordinates (ξ, η) to an arbitrary region in physical coordinates (x, y).

Returning to (6.2), given some function $f(x, y)$, we wish to calculate the integral

$$I = \int_{\Omega^e} f(x,y)\, dx\, dy \tag{6.3}$$

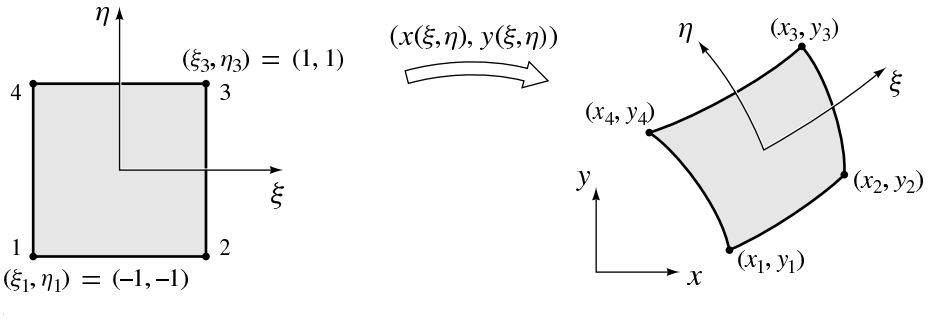

Figure 6.1-1.. The quadrilateral master element in (ξ, η) on the region $[-1, 1] \times [-1, 1]$ is mapped to its image in physical coordinates (x, y).

on the master element in (ξ, η) space. The change of variables theorem [20] from vector calculus states that (6.3) is equivalent to

$$I = \int_{\hat{\Omega}^e} \hat{f}(\xi, \eta) |\mathbf{J}| \, d\xi \, d\eta \tag{6.4}$$

where $\hat{\Omega}^e$ is the master element and $|\mathbf{J}|$ is the determinant of the non-singular Jacobian matrix of the transformation

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x(\xi, \eta) \\ y(\xi, \eta) \end{pmatrix} \tag{6.5}$$

Recall the Jacobian matrix of (6.5)

$$\mathbf{J} = \begin{pmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} \end{pmatrix} \tag{6.6}$$

and its determinant

$$|\mathbf{J}| = \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta} \frac{\partial y}{\partial \xi} \tag{6.7}$$

Using the local finite element shape functions to construct isoparametric transformations for (6.5), we have

$$x(\xi, \eta) = \sum_{i=1}^{n} x_i \hat{\psi}_i^e \tag{6.8}$$

$$y(\xi, \eta) = \sum_{i=1}^{n} y_i \hat{\psi}_i^e, \tag{6.9}$$

where n is the number of shape functions on element e, and the coordinates of each node of the element is given by the pair (x_i, y_i).

Since we seek a numerical solution, it is convenient to compute the integrals numerically; we use Gaussian quadrature rules of the form

$$I = \int_{\hat{\Omega}^e} \hat{f}(\xi, \eta) |\mathbf{J}| \, d\xi \, d\eta \approx \sum_{g=1}^{G} w_g \hat{f}(\xi_g, \eta_g) |\mathbf{J}|_{(\xi_g, \eta_g)}, \tag{6.10}$$

where the number of Gauss points is denoted G, (ξ_g, η_g) are the Gauss point coordinates (abscissas), $|\mathbf{J}|_{(\xi_g,\eta_g)}$ indicates that the Jacobian determinant is to be evaluated at the Gauss point coordinates, and w_g is the associated weight. For master element shapes in common use, such as triangles, hexahedra, quadrilaterals, etc., the quadrature points and weights are well-known and tabulated, e.g., see [2].

Table 6.1-1.. Weights and coordinates for the Gauss quadrature rules on elements with linear shape functions.

element topology	degree G	weights w_g	coordinates ξ_g
1D Edge	2	$1, 1$	$-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}$
2D Triangle	3	$\frac{1}{6}, \frac{1}{6}, \frac{1}{6}$	$(\frac{1}{2}, \frac{1}{2}), (0, \frac{1}{2}), (\frac{1}{2}, 0)$
2D Quadrilateral	4	$\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}$	$(-\gamma, -\gamma), (\gamma, -\gamma),$ $(\gamma, \gamma), (-\gamma, \gamma)$ †
3D Tetrahedron	4	$\frac{1}{24}, \frac{1}{24}, \frac{1}{24}, \frac{1}{24}$	$(\alpha, \beta, \beta), (\beta, \alpha, \beta),$ $(\beta, \beta, \alpha), (\beta, \beta, \beta)$ ‡
3D Hexahedron	8	$\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}$	$(-\gamma, -\gamma, -\gamma), (\gamma, -\gamma, -\gamma),$ $(\gamma, \gamma, -\gamma), (-\gamma, \gamma, -\gamma),$ $(-\gamma, -\gamma, \gamma), (\gamma, -\gamma, \gamma),$ $(\gamma, \gamma, \gamma), (-\gamma, \gamma, \gamma)$ †

† $\gamma = \sqrt{3}/6$
‡ $\alpha = 0.58541020, \beta = 0.13819660$

For linear elements, Aria uses the rules shown in Table 6.1-1, which correspond to the master element topologies given in Section 6.2. Because the order of quadrature is fixed for a given element (and the associated polynomial degree of element shape functions), the user of Aria should not forget to account for possible errors in the quadrature. Error in quadrature could be significant when user subroutines for boundary conditions, source terms, etc., attempt to represent highly oscillatory, or unsmooth data.

6.2. LINEAR MASTER ELEMENTS

In this section, we present the shape functions $\hat{\psi}_i^e$ for the linear master elements currently supported in Aria. Presently, all elements use the Lagrange basis functions. For the sake of brevity, we drop the superscript e on the element shape function. Hence, the notation $\hat{\psi}_i$ is equivalent to $\hat{\psi}_i^e$.

6.2.1. 2D Triangular Element

The master element for the two-dimensional triangle with nodes numbered locally as shown. The Gauss quadrature rule given in Table 6.1-1 for the 2D Triangle is used to integrate this element. The

shape functions are

$$\begin{aligned} \hat{\psi}_1 &= 1-\xi-\eta \\ \hat{\psi}_2 &= \xi \\ \hat{\psi}_3 &= \eta \end{aligned} \tag{6.11}$$

6.2.2. 2D Quadrilateral Element

The master element for the two-dimensional quadrilateral is shown in Figure 6.1-1. The Gauss quadrature rule for the 2D Quadrilateral given in Table 6.1-1 is used to integrate this element. The shape functions are

$$\begin{aligned} \hat{\psi}_1 &= \tfrac{1}{4}(1-\xi)(1-\eta) \\ \hat{\psi}_2 &= \tfrac{1}{4}(1+\xi)(1-\eta) \\ \hat{\psi}_3 &= \tfrac{1}{4}(1+\xi)(1+\eta) \\ \hat{\psi}_4 &= \tfrac{1}{4}(1-\xi)(1+\eta) \end{aligned} \tag{6.12}$$

6.2.3. 2D Face Element

When applying flux boundary conditions, it is often necessary to integrate on the surfaces of the two-dimensional elements presented in Sections 6.2.1 and 6.2.2. In both cases, the surface is topologically a one-dimensional edge. The two shape functions that define this element are

$$\begin{aligned} \hat{\psi}_1 &= \tfrac{1}{2}(1-\xi) \\ \hat{\psi}_2 &= \tfrac{1}{2}(1+\xi) \end{aligned} \tag{6.13}$$

The Gauss quadrature rule for the 1D Edge topology that is given in Table 6.1-1 is used to integrate this element.

Note that the location along this edge is given by the single parametric coordinate, ξ, although there is a pair of physical Cartesian coordinates, (x, y) associated with each value of ξ. This fact makes the construction of the transformation given as (6.5) a little less obvious. In this case, the determinant of the Jacobian matrix is the ratio of the edge length in physical coordinates to the edge length in computational coordinates,

$$|\mathbf{J}| = \sqrt{\left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2}$$

Since we are using isoparametric mappings, it is easy to show that, using the linear shape functions given as (6.13), this determinant is given by

$$|\mathbf{J}| = \frac{1}{2}\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$$

6.2.4. 3D Tetrahedron Element

The topology of the four-node, linear tetrahedron element is shown in Figure 6.2-1. The Gauss quadrature rule for the 3D Tetrahedron given in Table 6.1-1 is used to integrate this element. The shape functions are

$$\begin{aligned} \hat{\psi}_1 &= 1-\xi-\eta-\zeta \\ \hat{\psi}_2 &= \xi \\ \hat{\psi}_3 &= \eta \\ \hat{\psi}_4 &= \zeta \end{aligned} \tag{6.14}$$

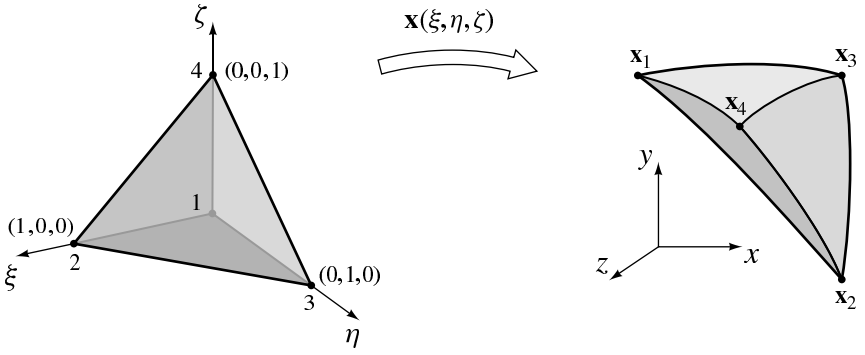

Figure 6.2-1.. The master tetrahedron element is mapped to its image in physical coordinates (x, y, z).

6.2.5. 3D Triangular Face Element

The tetrahedral element presented in Section 6.2.4 has three-node, triangular faces, which are topologically equivalent to the two-dimensional triangular element that was presented in Section 6.2.1. Accordingly, the shape shape functions are identical, and, for convenience, we repeat them here:

$$\begin{aligned} \hat{\psi}_1 &= 1-\xi-\eta \\ \hat{\psi}_2 &= \xi \\ \hat{\psi}_3 &= \eta \end{aligned} \tag{6.15}$$

The Gauss quadrature rule for the 2D Triangle that is given in Table 6.1-1 is used to integrate this element.

Since the physical coordinates are three-dimensional, but there are only two parameters on the surface, (ξ, η), the transformation from computational coordinates to physical coordinates requires a little elaboration. Vector calculus provides a formula for the integral of a scalar function over a parameterized surface [20]. We are interested in the surface integral

$$I = \int_{\Gamma^e} f(x, y, z)\, dx\, dy\, dz \tag{6.16}$$

where Γ^e is the three dimensional surface of our face element, and is given by the isoparametric mapping

$$\begin{pmatrix} x(\xi,\eta) \\ y(\xi,\eta) \\ z(\xi,\eta) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{3} x_i \hat{\psi}_i(\xi,\eta) \\ \sum_{i=1}^{3} y_i \hat{\psi}_i(\xi,\eta) \\ \sum_{i=1}^{3} z_i \hat{\psi}_i(\xi,\eta) \end{pmatrix} \tag{6.17}$$

It can be shown that (6.16) may be computed as

$$I = \int_{\hat{\Gamma}^e} \hat{f}(\xi,\eta)|\mathbf{J}|\, d\xi\, d\eta, \tag{6.18}$$

where $\hat{\Gamma}^e$ is the master element, $\hat{f}(\xi,\eta) = f(x(\xi,\eta), y(\xi,\eta), z(\xi,\eta))$, and the determinant of the Jacobian matrix may be written as

$$|\mathbf{J}| = \sqrt{\left|\frac{\partial(x,y)}{\partial(\xi,\eta)}\right|^2 + \left|\frac{\partial(y,z)}{\partial(\xi,\eta)}\right|^2 + \left|\frac{\partial(x,z)}{\partial(\xi,\eta)}\right|^2} \tag{6.19}$$

(6.19) involves the calculation of the determinants of three sub-matrices. For example,

$$\left|\frac{\partial(x,y)}{\partial(\xi,\eta)}\right| = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}$$

6.2.6. 3D Hexahedron Element

Figure 6.2-2 shows the topology of the eight-node, linear hexahedral master element. The Gauss quadrature rule for the 3D Hexahedron that is given in Table 6.1-1 is used to integrate this element.

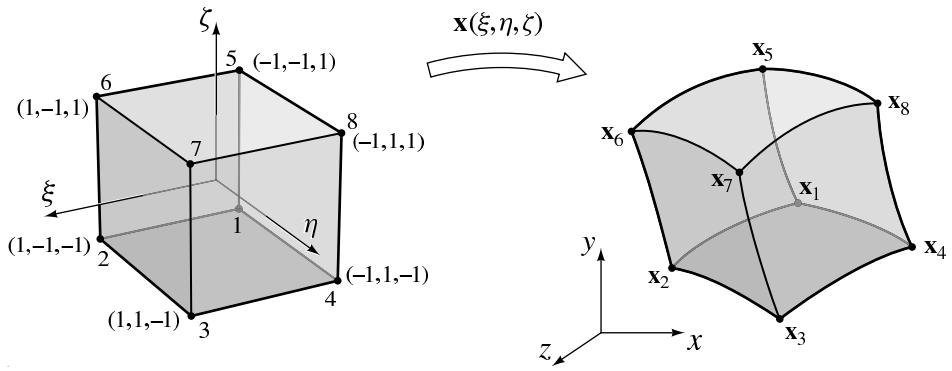

Figure 6.2-2.. The master hexahedron element in (ξ,η,ζ) on the region $[-1,1]\times[-1,1]\times[-1,1]$ is mapped to its image in physical coordinates (x,y,z).

The shape functions are

$$\hat{\psi}_1 = \frac{1}{8}(1-\xi)(1-\eta)(1-\zeta) \tag{6.20}$$

$$\hat{\psi}_2 = \frac{1}{8}(1+\xi)(1-\eta)(1-\zeta) \tag{6.21}$$

$$\hat{\psi}_3 = \frac{1}{8}(1+\xi)(1+\eta)(1-\zeta) \tag{6.22}$$

$$\hat{\psi}_4 = \frac{1}{8}(1-\xi)(1+\eta)(1-\zeta) \tag{6.23}$$

$$\hat{\psi}_5 = \frac{1}{8}(1-\xi)(1-\eta)(1+\zeta) \tag{6.24}$$

$$\hat{\psi}_6 = \frac{1}{8}(1+\xi)(1-\eta)(1+\zeta) \tag{6.25}$$

$$\hat{\psi}_7 = \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta) \tag{6.26}$$

$$\hat{\psi}_8 = \frac{1}{8}(1-\xi)(1+\eta)(1+\zeta) \tag{6.27}$$

6.2.7. 3D Quadrilateral Face Element

The hexahedron element described in Section 6.2.6 has quadrilateral faces. The shape functions that are used for interpolating on one of these faces are identical to the shape functions used for the two-dimensional quadrilateral element that was discussed in Section 6.2.2. We repeat them here for convenience:

$$\begin{aligned} \hat{\psi}_1 &= \tfrac{1}{4}(1-\xi)(1-\eta) \\ \hat{\psi}_2 &= \tfrac{1}{4}(1+\xi)(1-\eta) \\ \hat{\psi}_3 &= \tfrac{1}{4}(1+\xi)(1+\eta) \\ \hat{\psi}_4 &= \tfrac{1}{4}(1-\xi)(1+\eta) \end{aligned} \tag{6.28}$$

As discussed in Section 6.2.5, integrals on this surface master element may be computed according to (6.18), with the determinant given by (6.19). The Gauss quadrature rule for the 2D Quadrilateral that is given in Table 6.1-1 is used to integrate this element.

6.2.8. 3D Triangular Shell Element

Sometimes, a subdomain $\Omega_i \subset \Omega$ may be very thin, and may also possibly consist of a highly conductive material like aluminum. In this case, it is reasonable to neglect the thermal gradient through the thickness of Ω_i. Shell elements are specialized volume elements that are often used to model this situation. The topology of the linear triangular shell is illustrated in Figure 6.2-3, which has three nodes, and two faces. Topologically, the shell has no thickness that can be derived solely from its node coordinates; instead, the thickness is an attribute specified as input data. The shape functions are

identical to the 3D Triangular Face element given in Section 6.2.5, and we repeat them here for convenience:

$$\begin{aligned} \hat{\psi}_1 &= 1-\xi-\eta \\ \hat{\psi}_2 &= \xi \\ \hat{\psi}_3 &= \eta \end{aligned} \tag{6.29}$$

The transformation from computational coordinates to physical coordinates for the three-dimensional shell element depends on the thickness attribute. In this case, we wish to calculate the volume integral

$$I = \int_{\Omega^e} f(x,y,z)\, dx\, dy\, dz \tag{6.30}$$

The parametric mapping from computational coordinates to physical coordinates may be regarded as the surface mapping, plus a component that comes up out of the shell mid-plane. With reference to Figure 6.2-3, we may write this mapping as

$$\begin{pmatrix} x(\xi,\eta,\zeta) \\ y(\xi,\eta,\zeta) \\ z(\xi,\eta,\zeta) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{I} x_i\hat{\psi}_i(\xi,\eta) + \zeta\theta_x \sum_{i=1}^{I} \tau_i\hat{\psi}_i(\xi,\eta) \\ \sum_{i=1}^{I} y_i\hat{\psi}_i(\xi,\eta) + \zeta\theta_y \sum_{i=1}^{I} \tau_i\hat{\psi}_i(\xi,\eta) \\ \sum_{i=1}^{I} z_i\hat{\psi}_i(\xi,\eta) + \zeta\theta_z \sum_{i=1}^{I} \tau_i\hat{\psi}_i(\xi,\eta) \end{pmatrix} \tag{6.31}$$

where the shape functions $\hat{\psi}_i$ are given by (6.29), $I = 3$, τ_i denotes the thickness of the shell at node i, $\zeta \in [-1,1]$ is the parametric coordinate in the direction locally orthogonal to the shell mid-plane, and we have defined the inner products

$$\begin{aligned} \theta_x &= \hat{\mathbf{n}}_\zeta \cdot \hat{\mathbf{i}} \\ \theta_y &= \hat{\mathbf{n}}_\zeta \cdot \hat{\mathbf{j}} \\ \theta_z &= \hat{\mathbf{n}}_\zeta \cdot \hat{\mathbf{k}} \end{aligned} \tag{6.32}$$

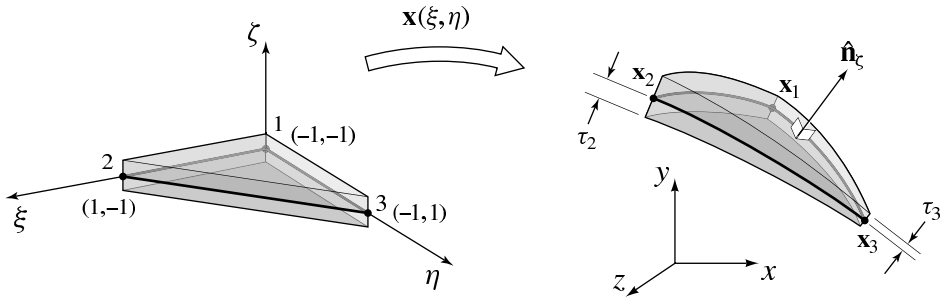

Figure 6.2-3.. The midplane of the master triangular shell element is mapped to its image in physical coordinates (x, y, z), and the variation in the shell thickness is represented with the element shape functions.

In (6.32), $\hat{\mathbf{n}}_\zeta$ is the unit vector in the direction locally orthogonal to the element mid-plane, and $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$ are the unit vectors in the x, y and z coordinate directions, respectively.

The volume integral in (6.30) on the master element is

$$I = \int_{\hat{\Omega}^e} \hat{f}(\xi, \eta, \zeta)|\mathbf{J}| \, d\xi \, d\eta \, d\zeta \tag{6.33}$$

where $|\mathbf{J}|$ is the determinant of the Jacobian matrix obtained from the transformation given by (6.31).

Aria currently only supports constant thickness shells. In this case, I reduces to

$$I = \tau \int_{\hat{\Gamma}^e} \hat{f}(\xi, \eta, 0)|\mathbf{J}| \, d\xi \, d\eta \tag{6.34}$$

where τ is the constant shell thickness, $\hat{\Gamma}^e$ denotes the triangular surface of the shell mid-plane, $|\mathbf{J}|$ is the determinant of the transformation given by (6.17), and the Gauss quadrature rule for the 2D Triangle that is given in Table 6.1-1 is used to perform the numerical integration.

6.2.9. 3D Quadrilateral Shell Element

The three-dimensional, linear, quadrilateral shell element is, in principle, no different from the triangular shell element which we discussed in Section 6.2.8. As illustrated in Figure 6.2-4, the quadrilateral shell has four nodes, instead of three, with the basis functions given in (6.28). For convenience, we repeat these basis functions here:

$$\begin{aligned} \hat{\psi}_1 &= \tfrac{1}{4}(1-\xi)(1-\eta) \\ \hat{\psi}_2 &= \tfrac{1}{4}(1+\xi)(1-\eta) \\ \hat{\psi}_3 &= \tfrac{1}{4}(1+\xi)(1+\eta) \\ \hat{\psi}_4 &= \tfrac{1}{4}(1-\xi)(1+\eta) \end{aligned} \tag{6.35}$$

The transformation from computational coordinates to physical coordinates is again given by (6.31), but with $I = 4$, and the $\hat{\psi}_i$ are defined by (6.35).

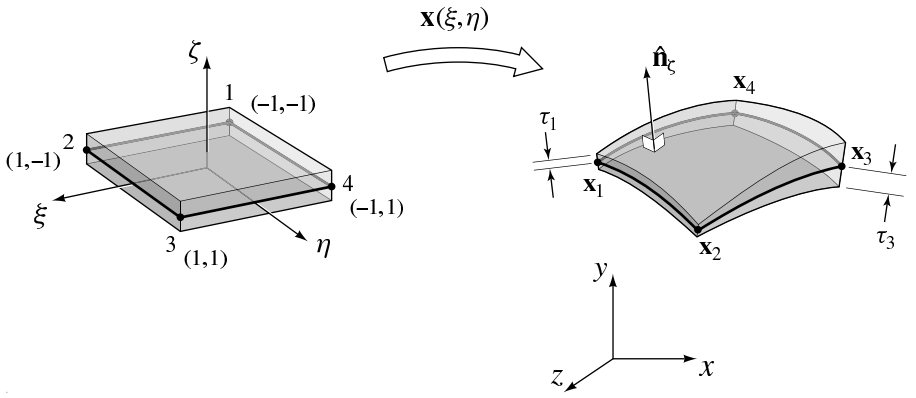

Figure 6.2-4.. The midplane of the master quadrilateral shell element is mapped to its image in physical coordinates (x, y, z), and the variation in the shell thickness is represented with the element shape functions.

Currently, Aria only supports constant thickness shells, and the simplifications discussed in Section 6.2.8 also apply in this case. The volume integral therefore reduces to the product of the shell thickness and the integral over the mid-plane surface, with the transformation from computational to physical coordinates identical to that which is described in Section 6.2.7.

7. ELEMENT CONTRIBUTIONS

In Chapters 3 and 4, we discussed the mathematical models for the physical boundary conditions and volumetric heating terms that are admissible for the stationary and transient problem statements, which were presented previously in Chapter 2. Subsequently, in Chapter 5, we developed the associated weak statements, which introduced integral terms to be calculated using finite elements. In this chapter, we examine each integral in the weak statements, and present the form of the associated element contributions to the global system of equations. These contributions are calculated locally on each element, and accumulated into their correct locations in the global system. We consider the system represented by (5.22), since it is the most general and contains all the terms of interest.

7.1. CAPACITANCE OPERATOR

The capacitance matrix is also called the mass matrix in the finite element literature. The contribution to (5.23a) for element Ω^e may be written as

$$M_{ij}^e = \int_{\Omega^e} \rho C_P \psi_i^e \psi_j^e \, d\Omega \tag{7.1}$$

It is trivial to show that $M_{ij}^e = M_{ji}^e$, and therefore the contributions to the global matrix are symmetric. The element matrix M_{ij}^e is a square, $N \times N$ matrix, where N is the number of shape functions, ψ_i^e. For example, for the eight-node linear hexahedral element, M_{ij}^e is of size 8×8.

(7.1) is often referred to as the consistent mass matrix. This nomenclature is to distinguish it from certain diagonal approximations, which are called lumped mass matrices. If an explicit time integrator, such as forward Euler time differencing, is used, the above consistent mass matrix leads to a coupled system of equations. Historically this disadvantage was circumvented by diagonalizing M_{ij}^e, or lumping the mass, thereby permitting numerical solutions to be obtained without solving a coupled system of equations. Since Aria uses implicit time differencing, the advantages of using a lumped mass matrix are less apparent. Time integrators using the lumped mass matrix are usually more diffusive than their consistent counterparts; sometimes, if a numerical solution is plagued by non-physical oscillations, this is an advantage, but usually it is better to eliminate the cause of the oscillations, rather than mask them with diffusion. Lumping the mass matrix does make the resulting fully discrete system of equations more diagonally dominant, but this advantage may be offset by a increase in magnitude and phase error, e.g., see [7]. Nevertheless, Aria provides both the consistent mass matrix and the lumped mass matrix approximation, the latter being obtained by using a nodal quadrature rule.

7.2. DIFFUSION OPERATOR

The diffusion element matrix was presented as (6.2), and we repeat it here for the sake of completeness:

$$K_{ij}^e = \int_{\Omega^e} \nabla\psi_i^e \cdot \mathbf{k}\nabla\psi_j^e \, d\Omega \tag{7.2}$$

This matrix is a square, $N \times N$ matrix, where N is the number of shape functions, ψ_i^e. It is easy to show that, as long as the thermal conductivity tensor is symmetric, then $K_{ij}^e = K_{ji}^e$. Crandall [6] has shown that the thermal conductivity tensor is always symmetric for finite, bounded conductors.

7.3. CONVECTION OPERATOR

The element matrix for the convection term may be written as

$$U_{ij}^e = \int_{\Omega^e} \rho C_P \psi_i^e \mathbf{v} \cdot \nabla\psi_j^e \, d\Omega \tag{7.3}$$

where the velocity field, $\mathbf{v}$, is known. This matrix is also of size $N \times N$, where N is the number of shape functions defined on the master element. However, note that this matrix is not symmetric, since

$$U_{ji}^e = \int_{\Omega^e} \rho C_P \psi_j^e \mathbf{v} \cdot \nabla\psi_i^e \, d\Omega \neq U_{ij}^e$$

7.4. SOURCE/SINK

A source/sink term contributes only to the forcing vector, and enters the problem as data. The associated element contribution is

$$F_i^e = \int_{\Omega^e} \dot{q}\psi_i^e \, d\Omega \tag{7.4}$$

This vector has a length equal to the number of finite element shape functions on the associated master element.

7.5. SPECIFIED HEAT FLUX

Let Γ_H denote the surface on which a specified heat flux boundary condition of the form given by (4.6). Then the contribution to the fully discrete system enters into the forcing vector,

$$F_i^e = -\int_{\Gamma_H^e} q_b \psi_i^e \, d\Gamma \tag{7.5}$$

where Γ_H^e is a finite element on the surface Γ_H. This vector has a length equal to the number of finite element shape functions on the associated face master element.

Note that for adiabatic boundary conditions, $q_b = 0$, and the above integral vanishes. Hence, in the finite element implementation, the adiabatic boundary condition enters naturally into the problem, and boundary surfaces that are left unspecified are treated as adiabatic.

7.6. CONVECTION HEAT FLUX

In the absence of the bulk fluid model, the convection heat flux given as (4.8) contributes to both the coefficient matrix and the forcing vector. Let Γ_c^e be an arbitrary finite element on the surface over which this boundary condition is to be applied. Recall that, in (5.21), the flux integral is on the right hand side and has a negative sign. Upon substitution of (4.8) into (5.21), we obtain

$$-\int_{\Gamma_c^e} \psi_i^e q_n \, d\Gamma = -\int_{\Gamma_c^e} h\psi_i^e \left(T - T_r\right) \, d\Gamma \tag{7.6}$$

If we substitute the finite element interpolant for T, the result may be written as

$$-\int_{\Gamma_c^e} \psi_i^e q_n \, d\Gamma = -\sum_{j=1}^{N} \left(\int_{\Gamma_c^e} h\psi_i^e \psi_j^e \, d\Gamma \right) T_j + \int_{\Gamma_c^e} \psi_i^e T_r \, d\Gamma \tag{7.7}$$

The first term on the right hand side of the above equation contributes to the global matrix; we may define the element contribution to row i and column j as

$$K_{ij}^e = \int_{\Gamma_c^e} h\psi_i \psi_j \, d\Gamma \tag{7.8}$$

Note that this matrix contribution is symmetric. The known reference temperature, T_r, enters as a contribution to the forcing function, namely

$$F_i^e = \int_{\Gamma_c^e} hT_r \psi_i^e \, d\Gamma \tag{7.9}$$

7.7. BULK FLUID

When the bulk fluid model is used in conjunction with a convective flux boundary condition, the reference temperature is no longer known data, and must be computed during the solution process. In this case, (5.22) is augmented by the discretized version of the bulk fluid element conservation statement. An additional row and column is added to the matrix, since an additional unknown and its associated equation are present.

From the standpoint of the temperature field solution, the convection heat flux boundary condition discussed in Section 7.6 is modified, since (7.7) becomes

$$\int_{\Gamma_c^e} \psi_i^e q_n d\Gamma = \sum_{j=1}^{N} \left(\int_{\Gamma_c^e} h\psi_i^e \psi_j^e d\Gamma \right) T_j - T_b \int_{\Gamma_c^e} h\psi_i^e \, d\Gamma \tag{7.10}$$

where we have moved T_b out of the integral, since it is constant on an element. Furthermore, since both terms on the right hand side of (7.10) are unknown, they both contribute to the global matrix. We may write this as the rectangular matrix contribution

$$\begin{bmatrix} K_{ij}^e & B_i^e \end{bmatrix} \begin{pmatrix} T_j \\ T_b \end{pmatrix} \tag{7.11}$$

where

$$B_i^e = -\int_{\Gamma_c^e} h\psi_i^e \, d\Gamma \tag{7.12}$$

and K_{ij}^e is given by (7.8).

The implication of (7.10) is that each node on the surface Γ_c is coupled to the bulk fluid temperature. The additional equation for the unknown T_b is provided by the bulk fluid conservation statement given by (4.31), which we repeat here:

$$\frac{d}{dt}(\rho C_P V T_b) + \int_{\Gamma_c} h\,(T_b - T)\; d\Gamma = 0 \tag{7.13}$$

Assuming a constant volume V, and substituting the finite element interpolant for T, the semidiscretized version of (7.13) is

$$\rho C_P V \frac{dT_b}{dt} + \sum_{e=1}^{E} \left[\int_{\Gamma_c^e} h \, d\Gamma \right] T_b - \sum_{e=1}^{E} \sum_{j=1}^{N} \left[\int_{\Gamma_c^e} h\psi_j^e \, d\Gamma \right] T_j = 0 \tag{7.14}$$

Consideration of the first term in (7.14) reveals that each off-diagonal entry in the new row in the mass matrix which appears in (5.22) is zero, and that the diagonal entry is exactly $\rho C_P V$. The second and third terms contribute to the new row in the stiffness matrix. The rectangular element contribution is

$$\begin{bmatrix} B_j^e & H^e \end{bmatrix} \begin{pmatrix} T_j \\ T_b \end{pmatrix} \tag{7.15}$$

where B_j^e is given by (7.12), and

$$H^e = \int_{\Gamma_c^e} h \, d\Gamma$$

Therefore, a single face on the surface Γ_c^e has the square, symmetric contribution to the global stiffness matrix

$$\begin{bmatrix} K_{ij}^e & B_i^e \\ B_j^e & H^e \end{bmatrix} \begin{pmatrix} T_j \\ T_b \end{pmatrix} \tag{7.16}$$

Finally, we remark that if the bulk fluid volume is completely enclosed by a surface of the finite element mesh, then it is possible to compute the enclosed volume according to the Gauss Divergence Theorem. For the case of three-dimensional Cartesian coordinates, let

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{7.17}$$

then we may write

$$\frac{1}{3}\int_V \nabla\cdot\mathbf{x}\,d\Omega = \int_V \frac{1}{3}\left(\frac{\partial x}{\partial x}+\frac{\partial y}{\partial y}+\frac{\partial z}{\partial z}\right)\,d\Omega = \int_V d\Omega \equiv V \quad . \tag{7.18}$$

Using the Gauss divergence theorem, it is easy to show that the enclosed volume is given by

$$V = \frac{1}{3}\int_\Gamma \mathbf{x}\cdot\hat{\mathbf{n}}\,d\Gamma \quad . \tag{7.19}$$

Similarly for two-dimensions

$$V = \frac{1}{2}\int_\Gamma \mathbf{x}\cdot\hat{\mathbf{n}}\,d\Gamma \tag{7.20}$$

and for the special case of axisymmetric coordinates,

$$V = \frac{1}{2}\int_\Gamma \mathbf{r}\cdot\hat{\mathbf{n}}\,d\Gamma \tag{7.21}$$

where

$$\mathbf{r} = \begin{pmatrix} \frac{r}{2} \\ 0 \\ z \end{pmatrix} \quad .$$

The calculation of enclosed volume is implemented independently for a volume void bounded by a collection of surfaces and as a specialization for the bulk fluid element.

7.8. SURFACE RADIATION FLUX

The surface radiation flux boundary condition given in (4.16) must be linearized prior to its implementation, since it depends on the fourth power of the unknown temperature. For convenience, we repeat this flux here:

$$q_n(T) = \sigma\epsilon F\left(T^4 - T_r^4\right) \tag{7.22}$$

We discuss the details of the nonlinear solution strategy in Chapter 8, and restrict our discussion in this section to the linearization of the radiation flux. Let T_* indicate the temperature state about which we are linearizing. Currently, the linearization implemented in Aria is to lag the flux by one nonlinear iteration level, so that q_n is known. Then the boundary condition only contributes to the forcing vector. The element contribution is equivalent to a specified heat flux

$$F_i^e = -\int_{\Gamma_R^e} \sigma\epsilon F\psi_i^e\left(T_*^4 - T_r^4\right)\,d\Gamma \tag{7.23}$$

where Γ_R^e is an arbitrary finite element on the surface over which this boundary condition is to be applied.

Instead of lagging the surface radiation as indicated in (7.23), it is preferable to linearize it via a Taylor series so that it contributes to both the coefficient matrix and the forcing function. If we expand (7.22) in a Taylor series about T_*, we obtain

$$q_n\left(T_* + \delta T\right) = \sigma\epsilon F\left[T_*^4 - T_r^4 + 4T_*^3\delta T\right] \tag{7.24}$$

where $\delta T = T - T_*$. Then we may write

$$\int_{\Gamma_R^e} \psi_i^e q_n(T)\, d\Gamma = \int_{\Gamma_R^e} \sigma\epsilon F\psi_i^e\left[T_*^4 - T_r^4 + 4T_*^3\left(T - T_*\right)\right]\, d\Gamma \tag{7.25}$$

After grouping terms involving the known T_* and unknown T, we obtain the linearized form of the surface radiation boundary condition element, which are expressed as the matrix contribution

$$K_{ij}^e = \int_{\Gamma_R^e} 4\sigma\epsilon T_*^3\psi_i^e\psi_j^e\, d\Gamma \tag{7.26}$$

and the forcing vector contribution

$$F_i^e = \int_{\Gamma_R^e} \sigma\epsilon F\left(3T_*^4 + T_r^4\right)\psi_i^e\, d\Gamma \tag{7.27}$$

We plan to implement the linearization described by Equations (7.26) and (7.27) in a forthcoming version of Aria.

7.9. ENCLOSURE RADIATION SURFACE FLUX

The enclosure radiation boundary condition given by (4.21) enters our weak statement in a manner similar to that of the surface radiation flux described in Section 7.8. The enclosure radiation flux was written as (4.21), which we repeat here for convenience:

$$q_n(T) = \sigma\epsilon T^4 - \epsilon G, \tag{7.28}$$

This flux and the surface radiation flux are both nonlinear, and vary as the fourth power of the temperature, but there are important differences. First, instead of a known reference temperature T_r, the surface irradiation, G, appears in (7.28), and must be calculated as part of the solution process. To begin, let $\Gamma_{\mathcal{E}}^e$ be an arbitrary finite element on the enclosure. We call $\Gamma_{\mathcal{E}}^e$ a facet, and let $\Gamma_{\mathcal{E}}$ consist of a total of E facets. Furthermore, the flux (7.28) is constant on a facet; to emphasize this, let us rewrite it as

$$q_n^e(T) = \sigma\epsilon\left[T^e\right]^4 - \epsilon G_e(T^e), \tag{7.29}$$

where T^e is some average facet temperature, and $G_e(T^e)$ is the irradiation on facet e, which is determined by the combined effects of all the other facets in the enclosure. Recall that this was expressed in Section 4.2.5 as

$$G_e(T^e) = \sum_{f=1}^{E} F_{ef}J_f(T^e). \tag{7.30}$$

We defer the discussion of the details of our nonlinear solution strategy until Chapter 8, but it is important to note here that the unknown radiosities, J_f, depend upon the solution of the temperature. We use a decoupled approach, and calculate the viewfactors, F_{ef}, and radiosities, J_f, via the Chaparral library [12]. Recall that the radiosities are obtained by solving the system of equations

$$\sum_{f=1}^{N} \left[\delta_{ef} - (1-\epsilon_e)F_{ef}\right] J_f = \sigma\epsilon_e \left[T^e\right]^4 \tag{7.31}$$

In (7.31), it is important to realize that the J_f and T^e are constant values associated with a given facet. But in our finite element approximation, the temperatures are associated with the nodes. Therefore, the facet temperature should be considered a projection, or averaging of the temperature found at the nodes of a given facet. Hence, we define the facet temperature as

$$T^e = \frac{\sum_{i=1}^{N} \int_{\Gamma_{\mathcal{E}}^e} \psi_i^e \, d\Gamma T_i}{\int_{\Gamma_{\mathcal{E}}^e} d\Gamma} \tag{7.32}$$

It is now convenient to define the projection vector $\mathbf{P}^e$, where row i is defined by

$$P_i^e = \frac{1}{A^e} \int_{\Gamma_{\mathcal{E}}^e} \psi_i^e \, d\Gamma, \tag{7.33}$$

and A^e is the area of facet e. Thus, (7.32) may be written as

$$T^e = \sum_{i=1}^{N} P_i^e T_i \tag{7.34}$$

Now that we have appropriately defined the facet temperature, we may linearize the flux given in (7.29). Expanding $q_n(T)$ in a Taylor series about a known state T_*^e, we obtain

$$q_n\left(T_* + \delta T\right) = \sigma\epsilon \left[T_*^e\right]^4 - \epsilon G_e(T_*^e) + 4\sigma\epsilon \left[T_*^e\right]^3 \delta T \tag{7.35}$$

where $\delta T = T^e - T_*^e$, and we have chosen to evaluate the irradiation at the known state, T_*^e. If we collect terms involving the known T_*^e and unknown T^e, (7.35) may be rearranged and written as

$$q_n\left(T_* + \delta T\right) = 4\sigma\epsilon \left[T_*^e\right]^3 T^e - 3\sigma\epsilon \left[T_*^e\right]^4 - \epsilon G_e(T_*^e) \tag{7.36}$$

In order to apply this flux in our weak statement, we must integrate it against the test function, ψ_i^e. Therefore, we have

$$\int_{\Gamma_{\mathcal{E}}^e} \psi_i^e q_n(T) \, d\Gamma = \int_{\Gamma_{\mathcal{E}}^e} \psi_i^e \left\{4\sigma\epsilon \left[T_*^e\right]^3 T^e - 3\sigma\epsilon \left[T_*^e\right]^4 - \epsilon G_e(T_*^e)\right\} \, d\Gamma \tag{7.37}$$

Exploiting the fact that all quantities on a given facet are constant, except for the shape functions, (7.37) becomes

$$\begin{aligned} \int_{\Gamma_{\mathcal{E}}^e} \psi_i^e q_n(T) \, d\Gamma = &\int_{\Gamma_{\mathcal{E}}^e} \psi_i^e \, d\Gamma \left\{4\sigma\epsilon \left[T_*^e\right]^3 T^e\right\} \\ &- \int_{\Gamma_{\mathcal{E}}^e} \psi_i^e \, d\Gamma \left\{3\sigma\epsilon \left[T_*^e\right]^4 + \epsilon G_e(T_*^e)\right\} \end{aligned} \tag{7.38}$$

If we introduce the definitions (7.34) and (7.33) into (7.38), we see that the linearized form of the element matrix contribution is

$$K_{ij}^e = 4\sigma\epsilon \left[T_*^e\right]^3 A^e P_i^e P_j^e \tag{7.39}$$

and the contribution to the source vector is

$$F_i^e = A^e\epsilon \left\{3\sigma \left[T_*^e\right]^4 + G_e(T_*^e)\right\} P_i^e \tag{7.40}$$

We remark that, when calculating terms in Equations (7.39) and (7.40), experience has shown that it is important to project the exponential power of the temperature, as opposed to projecting the temperature, then raising it to some exponential power. In other words, spurious oscillations are less likely to occur if

$$[T^e]^4 = \sum_{i=1}^{N} P_i^e T_i^4$$

is used instead of

$$[T^e]^4 = \left[\sum_{i=1}^{N} P_i^e T_i\right]^4$$

7.10. THERMAL CONTACT

The thermal contact algorithm is implemented as a Discontinuous Galerkin (DG)-like method that most closely follows the "Generalized Algorithm" discussed in [4] and references therein, and users are encouraged to read that report for more details regarding the nuances associated with the various types of contact enforcement.

Thermal contact is often enforced across an interface with a noncontiguous mesh, as seen in Figure 7.10-1. The surfaces between two subdomains may be meshed independently, so that there may be a physical gap between them, or the nodes on the two surfaces may not align. Moreover, in parallel computations, these surfaces will in general have different processor decompositions so that all of the required information may not be available locally on a given processor. This nonlocality of information complicates the implementation, because this information must be constructed in a consistent way on all processors. Aria uses the stk::search and Dash packages to detect when facets that lie on separate surfaces are close enough to be in contact in a parallel consistent manner.

Even in cases of collocated nodes, the DG-nature of the method means that the temperature is double defined along the contact interface, which is shown as Γ_{ij} in Figure 7.10-2. The preferred contact enforcement strategy for the case with no contact resistance, i.e., physically a continuous temperature field, is TIED_TEMPERATURE, which is derived similarly to Discontinuous Galerkin (DG) and Interior Penalty (IP) methods ([1, 4]). In practice, the continuity in the temperature is only enforced weakly; however, the jump is convergent with mesh refinement. TIED_TEMPERATURE is useful for analyzing systems whose geometry is so complex that a contiguous mesh is difficult or impossible to obtain. Also available as an enforcement strategy is GAP_CONDUCTANCE, which allows a discontinuous temperature across the interface, and specifies a flux according to a contact resistance.

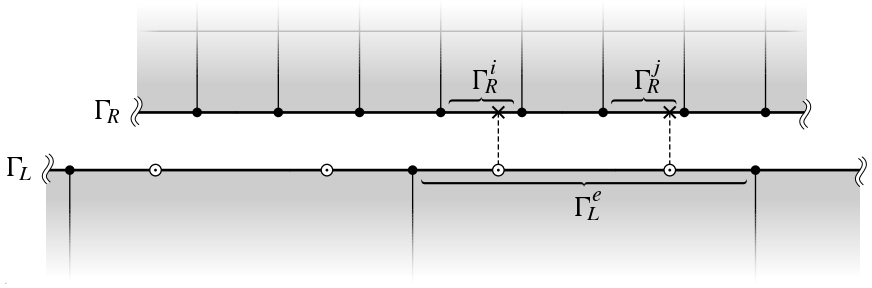

Figure 7.10-1.. In a Discontinuous Galerkin-based contact enforcement strategy, the elements involved on the right side Γ_R, are those whose sides intersect the quadrature points of the element on the left side Γ_L. In general, the sides of elements on the right side, Γ_R^i, and Γ_R^j, may not be contiguous.

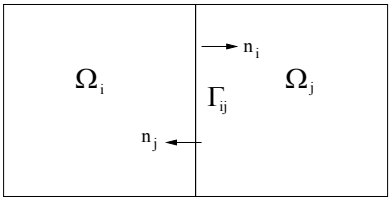

Figure 7.10-2.. Contact interface between subdomains Ω_i and Ω_j

Contact resistance is generally used when the system being analyzed consists of at least two parts that touch, and it is important to model their imperfect fit with a finite thermal resistance.

In Aria TIED_TEMPERATURE and GAP_CONDUCTANCE are implemented in a variational form. Consider a 2D or 3D domain Ω that consists of a set of subdomains Ω_i with interfaces Γ_{ij} at the intersection of each pair Ω_i and Ω_j, $i \neq j$ where $n_{i,j}$ denote the unit outward normal vector on each subdomain boundary $\partial\Omega_{i,j}$, as shown in Figure 7.10-2. On each subdomain Ω_i, we pose a standard heat conduction problem of the form

$$\nabla \cdot \boldsymbol{q} = f, \quad x \in \Omega_i, \tag{7.41}$$

where $\boldsymbol{q}$ is the heat flux, and f is a volumetric source term, along with appropriate boundary conditions on $\partial\Omega_i/\Gamma$. The heat flux is defined as $\boldsymbol{q} \equiv -k\nabla T$, where T is the temperature field, and k is the thermal conductivity. The notion of average and jump notations are useful in describing this method. The average notation is defined as

$$\{T\} = \frac{1}{2}(T_i + T_j) \tag{7.42}$$

for both scalar and vector quantities and the jump operator is defined as

$$[\![T]\!] \equiv T_i \boldsymbol{n}_i + T_j \boldsymbol{n}_j \tag{7.43}$$

and

$$[\![\boldsymbol{q}]\!] \equiv \boldsymbol{q}_i \cdot \boldsymbol{n}_i + \boldsymbol{q}_j \cdot \boldsymbol{n}_j \tag{7.44}$$

for scalar and vector variables, respectively.

To write (7.41) in a variational statement, let v be suitable a test function that is continuous within each subdomain and possibly discontinuous across Γ. We first multiply the differential energy equation (7.41)

in each subdomain Ω_i by a suitable test function v and integrate by parts to get the weak form

$$\int_{\Omega_i} \nabla v \cdot \boldsymbol{q}\, d\Omega + \int_{\Omega_i} v f\, d\Omega - \int_{\partial\Omega_i} v\,(\boldsymbol{q}_i \cdot \boldsymbol{n}_i)\; dS = 0 \tag{7.45}$$

Assuming for simplicity Dirichlet boundary conditions for T have been specified on $\partial\Omega_i \backslash \Gamma$, we can assume that $v = 0$ on $\partial\Omega_i \backslash \Gamma$. Summing over domains gives

$$\int_{\Omega_i} \nabla v \cdot \boldsymbol{q}\, d\Omega + \int_{\Omega_i} v f\, d\Omega - \sum_K \int_{\partial\Gamma_{ij}} v\,(\hat{\boldsymbol{q}}_k \cdot \boldsymbol{n}_k)\; dS = 0 \tag{7.46}$$

where $\hat{\boldsymbol{q}}$ is an arbitrary numerical flux to be chosen, and the summation is over all surface elements along the interface Γ_{ij}. Using the jump operators and noting that $[\![\hat{\boldsymbol{q}}]\!] = 0$ is necessary for conservation, (7.46) can be written compactly as

$$\int_{\Omega} \nabla v \cdot \boldsymbol{q}\, d\Omega_i + \int_{\Omega_i} v f\, d\Omega - \int_{\partial\Gamma_{ij}} [\![v]\!] \cdot \{\hat{\boldsymbol{q}}\}\, dS = 0 \tag{7.47}$$

As discussed in [1], there are many valid choices for closing the numerical flux. One such method is the Interior Penalty method, which is the method listed as the "Generalized Algorithm" in [4]. In this case the numerical flux is chosen as $\hat{\boldsymbol{q}} \equiv -\alpha_h k \nabla T + \beta_h [\![T]\!]$, and this choice can be shown to be both consistent and conservative in the weak form of the problem. The resultant weak form of the problem becomes

Interior Penalty Method Variational Form

$$\begin{aligned} &\int_{\Omega} \nabla v \cdot k \nabla T\, d\Omega_i - \int_{\Omega_i} v f\, d\Omega \\ &- \int_{\partial\Gamma_{ij}} [\![v]\!] \cdot \{\alpha_h k \nabla T\}\, dS + \int_{\partial\Gamma_{ij}} \beta_h [\![v]\!] \cdot [\![T]\!]\, dS = 0 \end{aligned} \tag{7.48}$$

For the TIED_CONTACT case (an Interior Penalty Method), coefficient $\alpha_h \equiv 1$ and β_h is defined as

$$\beta_h \equiv C\overline{kh^{-1}}, \tag{7.49}$$

where the overline denotes an average from both sides of the interface Γ_{ij}, C is an arbitrary positive number, nominally $1/2$, and h is a local mesh length scale.

For the GAP_CONDUCTANCE case, α_h and β_h are chosen as in [4], which is repeated here for convenience as

$$(\alpha_h, \beta_h) \equiv \begin{cases} \left(1 - \frac{1}{H}, C\overline{kh^{-1}}\right) & H > 1, \\ (0, R^{-1}) & H \le 1, \end{cases} \tag{7.50}$$

where

$$H = \frac{R^{-1}}{C\overline{kh^{-1}}}. \tag{7.51}$$

As before, the overline notation is used to denote the average from both sides of the interface.

While the above formulation is consistent, i.e. $\hat{\boldsymbol{q}}(T) = \boldsymbol{q}(T)|_\Gamma$, and conservative, i.e., $\hat{\boldsymbol{q}}$ is single valued along Γ, the discrete form of the integrals is only conservative in the case of collocated nodes, which is generally not the case in practical usage. However, [4] has shown that the error norms do decrease according to the order of the underlying finite-element scheme, as expected.

8. SOLUTION STRATEGY

In previous chapters, we have discussed how to generate linearized, fully discrete approximations to the weak statements for the transient and stationary problems that were presented in Sections 5.1 and 5.4. In both cases, the linearized system of equations may be written as

$$\mathbf{A}(\mathbf{T}_*)\mathbf{T} = \hat{\mathbf{f}}(\mathbf{T}_*, \mathbf{b}(\mathbf{T}_*), \mathbf{n}(\mathbf{T}_*)), \tag{8.1}$$

where $\mathbf{T}$ is the vector of unknown temperatures, $\mathbf{T}_*$ is the known temperature state about which we have linearized, $\mathbf{A}(\mathbf{T}_*)$ is a matrix and $\mathbf{f}$ is the forcing vector.

Recall that in the stationary case, we must solve a system of the form (8.1) with

$$\mathbf{A}(\mathbf{T}_*) = \hat{\mathbf{K}}(\mathbf{T}_*)$$

and

$$\hat{\mathbf{f}}(\mathbf{T}_*, \mathbf{b}(\mathbf{T}_*), \mathbf{n}(\mathbf{T}_*)) = \mathbf{f},$$

with each component of $\mathbf{f}$ given by (5.16c). For the transient case, we must solve (8.1) with

$$\mathbf{A}(\mathbf{T}_*) = \frac{1}{\Delta t^n}\mathbf{M} + \theta\hat{\mathbf{K}}(\mathbf{T}_*)$$

and

$$\hat{\mathbf{f}}(\mathbf{T}_*, \mathbf{b}(\mathbf{T}_*), \mathbf{n}(\mathbf{T}_*)) = \theta\mathbf{f} + (1-\theta)\mathbf{M}\dot{\mathbf{T}}_* + \frac{1}{\Delta t^n}\mathbf{M}\mathbf{T}_*$$

at each timestep.

As indicated, the forcing vector $\hat{\mathbf{f}}$ is, in the most general case, a function of the radiosities, $\mathbf{b}(\mathbf{T}_*)$ and the chemical species concentrations $\mathbf{n}(\mathbf{T}_*)$. The radiosities satisfy (7.31), which we rewrite here in vector form as

$$\mathbf{F}\mathbf{b} = \mathbf{g}(\mathbf{T}_*), \tag{8.2}$$

where $\mathbf{F}$ is the viewfactor matrix, and $\mathbf{g}(\mathbf{T}_*)$ is the vector of emitted radiative heat fluxes. Finally, the chemical concentrations satisfy the system of ordinary differential equations given by (3.6), which we rewrite here in vector form as

$$\frac{d\mathbf{n}}{dt} = \mathbf{p}(\mathbf{T}_*, \mathbf{n}) \tag{8.3}$$

with appropriate initial conditions.

The linearization (8.1) is equivalent to a fixed point iteration known as successive substitution applied to the nonlinear system (5.15) for the steady case and (5.22) for the transient case. This method is also known as functional iteration (see [10]). The method requires no Jacobian calculation and the

convergence is asymptotically linear. Since boundary conditions, volumetric heating sources and thermal conductivities are often given as tabulated values or user subroutines, formation of a full, analytic Jacobian is not practical. In fact, (8.1) includes linearizations of these these quantities. The simplicity of implementation of this method accounts for its popularity for the solution of a wide variety of problems. The rate of convergence of the basic iterative procedure of (8.1) can often be improved by the use of an acceleration or relaxation parameter.

We solve the coupled system represented by Equations (8.1)-(8.3) via a decoupled approach. In Section 8.1 we discuss the details of this strategy for the stationary case, then in Section 8.2 we discuss the minor differences that arise in the transient case. Generally speaking, the matrix $\mathbf{A}$ will be symmetric and positive definite, with the following important exceptions:

1. If the convection operator given in (7.3) is used, then $\mathbf{A}$ will be nonsymmetric.
2. If either the tied contact or resistance contact resistance enforcement schemes described in Section 4.1 are used, then $\mathbf{A}$ will be nonsymmetric.
3. If the $h-$adaptive, hanging node constraint enforcement scheme is used, then $\mathbf{A}$ will be symmetric, but indefinite.

If none of the above exceptions apply, then the method of conjugate gradients (CG) is always recommended to solve (8.1). Usually in this case, Jacobi preconditioning is also adequate; however, if the number of iterations is excessive, then incomplete Cholesky decomposition preconditioning, ILU or ILU-T, are recommended. If exception 3 applies, then the Finite Element Interface (FEI) [5, 24] library preprocesses the linear system before passing it to the underlying linear solver. The purpose of this preprocessing is to eliminate algebraically the slave equations from the augmented system. After this process, if the original augmented system was symmetric, then the reduced system is symmetric and positive definite, and CG is again recommended. If $\mathbf{A}$ is nonsymmetric, then an iterative method such as the generalized minimum residual method (GMRES) or the biconjugate gradient stabilized method (BI-CGSTAB) is recommended.

8.1. STATIONARY PROBLEMS

For the stationary case, the linearized thermal problem given by (8.1) is implied by (5.15). The solution strategy for this case is shown in Algorithm 8.1. In the input, it is assumed that a starting iterate $\mathbf{T}_*$ is available. Furthermore, a convergence tolerance, ϵ, a maximum number of iterations, I, and a relaxation parameter, α, are also required. The Chaparral library is used to compute the viewfactors required in Step 2, as well as the radiosities in Step 5. Of course, if no enclosure radiation boundary conditions have been applied, then Steps 2 and 5 are skipped. Recall that the ACME library is used to perform the contact search of Step 1. Of course, if no contact enforcement conditions have been specified, this step is also skipped. For this stationary case, we do not consider the time evolution of the chemical species, and so Step 6 is omitted. The solution in Step 13, the newly computed solution is modified via the usual relaxation formula. If $\alpha = 1$, then no relaxation is applied. If $\alpha < 1$, then the solution is under−relaxed, which is sometimes helpful if Algorithm 8.1 is divergent. Occasionally, over−relaxation, or $\alpha > 1$, will accelerate convergence. In Step 17, element death is omitted for stationary problems.

Algorithm 8.1 Strategy for solving the coupled nonlinear system.

Input: $\mathbf{T}_*, \epsilon, I, \alpha$
Output: $\mathbf{T}$

1: Perform contact search
2: Form $\mathbf{F}$
3: $i = 0$
4: while not finished do
5: Solve (8.2) for $\mathbf{b}$
6: Integrate (8.3) to obtain new $\mathbf{n}$
7: Form $\mathbf{A}(\mathbf{T}_*), \hat{\mathbf{f}}(\mathbf{T}_*, \mathbf{b}(\mathbf{T}_*), \mathbf{n}(\mathbf{T}_*))$
8: $\mathbf{r} = \hat{\mathbf{f}} - \mathbf{A}\mathbf{T}_*$
9: if $(\|\mathbf{r}\| < \epsilon)$ or $(i > I)$ then
10: finished = true
11: end if
12: Solve (8.1) for $\mathbf{T}$
13: $\mathbf{T} = \mathbf{T}_* + \alpha(\mathbf{T} - \mathbf{T}_*)$
14: $\mathbf{T}_* = \mathbf{T}$
15: end while
16: Perform dynamic load rebalance
17: Compute element death

8.2. TRANSIENT PROBLEMS

The solution strategy for the transient case, presented as Algorithm 8.2, adds additional steps to the stationary case discussed in Section 8.1. Essentially, Algorithm 8.1 is enclosed in an outer loop that performs the time integration. In this case, the linearized system given by (8.1) is implied by (5.22). Note that the viewfactor calculations and the contact searches are expensive operations, requiring significant communication across parallel processors. Accordingly, this work is only performed during a given timestep if the mesh topology changes. For example, the mesh topology might change due to element death, or h–adaptivity. Furthermore, the viewfactors for a given enclosure are recomputed only if that particular enclosure requires updating.

The time loop in Step 1 iterates from the initial time value, t_0, at which the initial conditions are known, up through the final time specified by the user, t_1. The predicted temperature is computed in Step 2 according to the methodology described in Section 5.6. As indicated in Step 3, this predicted temperature is subsequently used for the starting iterate of the nonlinear solution. Recall that, as discussed in Section 5.6, if the adaptive timestep option is used, the predicted temperature is also used to calculate the new timestep in Step 5. Given an initial condition for the chemical species, in Step 4, we use the CHEMEQ library [26] to integrate (8.3) over the time interval. CHEMEQ uses a stiff ordinary differential equation integrator, which adaptively subcycles species equations according to their stiffness.

Algorithm 8.2 Nonlinear solution strategy for transient problem.

Input: $\mathbf{T}_0, \Delta t, \mathbf{n}_0, t_0, t_1, \epsilon, I, \alpha$
Output: $\mathbf{T}$

1: for $t = t_0$ to t_1 do
2: Calculate the predicted temperature, $\mathbf{T}_p$
3: $\mathbf{T}_* = \mathbf{T}_p$
4: Use Algorithm 8.1 to solve $\mathbf{T}$
5: Calculate new Δt
6: end for

9. GENERAL SCHEME OF NOTATION

Table 9.0-1.. General notation.

Expression	Meaning
$\mathring{\mathcal{R}}$	interior of $\mathcal{R}$
$\partial\mathcal{R}$	boundary of $\mathcal{R}$
$\mathcal{R} \cup \mathcal{F}$	union of $\mathcal{R}$ and $\mathcal{F}$
$\mathcal{R} \cap \mathcal{F}$	intersection of $\mathcal{R}$ and $\mathcal{F}$
$\mathcal{R} \subset \mathcal{F}$	$\mathcal{R}$ is a subset of $\mathcal{F}$
$\mathcal{R} \setminus \mathcal{F}$	set complement
$x \in \mathcal{R}$	x is an element of the set $\mathcal{R}$
$f : \mathcal{R} \to \mathcal{F}$	f maps the set $\mathcal{R}$ into the set $\mathcal{F}$; $\mathcal{R}$ is the domain, $\mathcal{F}$ the codomain
$x \mapsto f(x)$	mapping that carries x into $f(x)$; e.g., $x \mapsto x^2$ is the mapping that carries every real number x into its square
$f \circ g$	composition of the mappings f and g; i.e., $(f \circ g)(x) = f(g(x))$
$\{x \mid R(x)$ holds$\}$	the set of all x such that $R(x)$ holds; e.g., $\{x \mid 0 \leq x \leq 1\}$ is the interval $[0, 1]$
$\delta(\mathbf{x} - \mathbf{x}_0)$	Dirac delta function centered at $\mathbf{x}_0$
$[\![\cdot]\!]$	measure of jump in a quantity across an interface
$\mathbb{R}$	real numbers
$\mathbb{N}$	natural numbers

Table 9.0-2.. Index of frequently used symbols.

Symbol	Name	First occurrence
C_P	constant pressure specific heat	11
C_v	constant volume specific heat	12
$\mathcal{F}$	set of fluid subdomains	12
f_{RG}	reacted gas fraction of a chemical species	17
q_n	flux normal to a surface	13
$\mathbf{K}$	conductivity tensor	11
$\hat{\mathbf{n}}$	outward unit normal vector	13
$\dot{q}$	volumetric heating	11
$\dot{q}$	magnitude of point heat source (or sink)	18
$\mathbf{q}$	flux vector	11
r	reaction rate in a chemical reaction	16
$\mathcal{S}$	set of solid subdomains	12
t	time	11
t_0	initial time	13
T	temperature	11
T_0	initial temperature at t_0	13
$\mathbf{v}$	velocity vector	12
$\mathbf{x}$	position vector	10
$\mathfrak{R}$	endo- or exothermic energy release in chemical reaction	16
ρ	density	11
Ω	domain	10
$\partial\Omega$	boundary of the domain	10
$\partial\Omega_{i-j}$	interface of two subdomains	10
Γ	subset of (sub)domain boundary	10
Γ_a	subset of boundary that is adiabatic	10
Γ_h	subset of boundary with convective flux applied	10
Γ_q	subset of boundary with flux applied	10
Γ_r	subset of boundary with radiative flux applied	10
Γ_T	subset of boundary with temperature applied	10
μ	concentration exponent in chemical reaction	17
ν	stoichiometric coefficient of reactant species	16
∇	gradient	11

BIBLIOGRAPHY

[1] Arnold, D., Cockburn, B. & Marini, L. D., 'Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems', SIAM J. Numer. Anal., 39, (2002), pp. 1749–79.

[2] Becker, E. B., Carey, G. F. & Oden, J. T., Finite Elements: An Introduction, Prentice-Hall Inc., New Jersey, 1981.

[3] Bova, S. W., Copps, K. D., Glass, M. W., Newman, C. K. & Okusanya, T., Calore: A Computational Heat Transfer Program, Volume 2, User Reference Manual., Tech. Rep. SAND2008-0098P, Sandia National Laboratories, Albuquerque, NM, USA, 2008.

[4] Carnes, B. R. & Copps, K. D., Thermal Contact Algorithms in SIERRA Mechanics, Tech. Rep. SAND2008-2607, Sandia National Laboratories, Albuquerque, New Mexico, 2008.

[5] Clay, R., Mish, K., Otero, I., Taylor, L. & Williams, A., An Annotated Reference Guide to the Finite Element Interface Specification Version 1.0, Technical Report SAND99-8229, Sandia National Laboratories, Albuquerque, New Mexico and Livermore, California, 1999.

[6] Crandall, S. H., 'The symmetry relations for anisotropic heat conduction', Physica, 21, (1955), pp. 251–252.

[7] Donea, J., 'A Taylor-Galerkin Method for Convective Transport Problems', Int. J. Numer. Methods Engrg., 20, (1984), pp. 101–119.

[8] Edwards, H. C. & Stewart, J. R., 'SIERRA: A Software Environment for Developing Complex Multi-Physics Applications', in K. J. Bathe, ed., 'First MIT Conference on Computational Fluid and Solid Mechanics', pp. 1147–1150, Elsevier, Amsterdam, 2001.

[9] Finlayson, B. A., The Method of Weighted Residuals and Variational Principles, Academic Press, New York, 1972.

[10] Gartling, D. K., Hogan, R. E. & Glass, M. W., Coyote—A Finite Element Computer Program for Nonlinear Heat Conduction Problems, Part I—Theoretical Background, Version 4.05, Tech. Rep. SAND94-1173, Sandia National Laboratories, Albuquerque, New Mexico, 2000.

[11] Gartling, D. K., Hogan, R. E. & Glass, M. W., Coyote—A Finite Element Computer Program for Nonlinear Heat Conduction Problems, Part II—User's Manual, Version 4.05, Tech. Rep. SAND94-1179, Sandia National Laboratories, Albuquerque, New Mexico, 2000.

[12] Glass, M. W., CHAPARRAL: A Library for Solving Enclosure Radiation Heat Transfer Problems, Tech. Rep. SAND01-xxxx, Sandia National Laboratories, Albuquerque, New Mexico, 2001.

[13] Glassman, I., Combustion, Academic Press, New York, 1977.

[14] Gresho, P., Lee, R., Sani, R. & Stullich, T., 'On the Time-Dependent FEM Solution of the Incompressible Navier-Stokes Equations in Two and Three Dimensions', Recent Advances in Numerical Methods in Fluids, 1.

[15] Kraus, A. D. & Bar-Cohen, A., Thermal Analysis and Control of Electronic Equipment, Hemisphere, Washington DC, 1983.

[16] LeVeque, R. J., Numerical Methods for Conservation Laws, Birkhäuser-Verlag, Boston, 1992.

[17] Lewis, R. W., Morgan, K., Thomas, H. R. & Seetharamu, K. N., The Finite Element Method in Heat Transfer Analysis, John Wiley and Sons, New York, 1996.

[18] Lienhard IV, J. H. & Lienhard V, J. H., A Heat Transfer Textbook, Phlogiston Press, Cambridge, Massachusetts, 3rd ed., 2004, URL http://web.mit.edu/lienhard/www/ahtt.html.

[19] Madhusudana, C. V., Thermal Contact Conductance, Springer-Verlag, New York, 1996.

[20] Marsden, J. E. & Tromba, A. J., Vector Calculus, W. H. Freeman and Company, New York, 1981.

[21] Mills, A. F., Basic Heat and Mass Transfer, Prentice Hall, New Jersey, 1999.

[22] Morel, J., McGhee, J. & Larsen, E., 'A 3-D Time-Dependent Unstructured Tetrahedral-Mesh SPn Method', Nuclear Science and Engineering, 123, (1996), pp. 319–327.

[23] Reddy, J. N. & Gartling, D. K., The Finite Element Method in Heat Transfer and Fluid Dynamics, CRC Press, Boca Raton, 2001.

[24] Saint-Georges, P., Notay, Y. & Warzee, G., 'Efficient Iterative Solution of Constrained Finite Element Analyses', Comput. Methods Appl. Mech. Engrg., 160, (1998), pp. 101–114.

[25] Siegel, R. & Howell, J. R., Thermal Radiation Heat Transfer, Taylor and Francis, Washington, D.C., 1992.

[26] Young, T. R., CHEMEQ—A Subroutine for Solving Stiff Ordinary Differential Equations, Tech. Rep. NRL Memorandum Report 4091, Naval Research Laboratory, Washington D.C., 1980.

INDEX

DISTRIBUTION

Email—Internal (encrypt for OUO)

Name	Org.	Sandia Email Address
Technical Library	01177	libref@sandia.gov

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.