

**SAND94-1179**  
Unlimited Release  
Printed October 1994

Distribution  
Category **UC-905**

**COYOTE -  
A FINITE ELEMENT COMPUTER PROGRAM  
FOR NONLINEAR HEAT CONDUCTION  
PROBLEMS  
PART II - USER'S MANUAL**

Version 2.5

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

**ABSTRACT**

User instructions are given for the finite element computer program, COYOTE. COYOTE is designed for the multi-dimensional analysis of nonlinear heat conduction problems including the effects of enclosure radiation and chemical reaction. The theoretical background and numerical methods used in the program are documented in SAND94-1173. Examples of the use of the code are presented in SAND94-1180.



# Preface

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

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 To improve the performance of the solution algorithms, the direct matrix solution methods have been replaced by iterative methods of the conjugate gradient type.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

The contributions of several individuals to the testing and evaluation of early versions of the code must be acknowledged. P. A. Sackinger (1511), J. F. Holland (6313) and R. S. Longenbaugh (6313) provided significant help in testing Version 1.0 of COYOTE. Also, M. B. Sirman (1511) generated the examples and documentation in Chapter 5 while debugging Version 2.0.



# Contents

Preface . . . . .	iii
<b>1 Introduction</b>	<b>1</b>
<b>2 Program Overview</b>	<b>3</b>
2.1 Program Features . . . . .	3
2.2 Program Organization . . . . .	4
<b>3 Input Guide</b>	<b>9</b>
3.1 Input Syntax . . . . .	10
3.2 Title Data Block . . . . .	12
3.3 Material Data Block . . . . .	13
3.4 Problem Definition Data Block . . . . .	23
3.5 Solution Data Block . . . . .	37
3.6 Post-processing Data Block . . . . .	47
3.7 Time Function Data Block . . . . .	49
3.8 Variable Function Data Block . . . . .	50
3.9 User Constants Data Block . . . . .	51
3.10 Termination Data . . . . .	52
3.11 User Supplied Subroutines . . . . .	53
3.11.1 Material Properties . . . . .	53
3.11.2 Volumetric Sources . . . . .	62
3.11.3 Material Velocity . . . . .	64
3.11.4 Boundary Conditions . . . . .	65
3.12 Initial Conditions and Restarts . . . . .	73
3.13 Error Messages . . . . .	74

<b>4</b>	<b>Code Installation and Access</b>	<b>77</b>
4.1	FORTRAN Coding and System Dependencies . . . . .	77
4.2	File Formats . . . . .	79
4.3	File Usage . . . . .	79
4.4	Access to the Code . . . . .	80
<b>5</b>	<b>Example Problems</b>	<b>83</b>
5.1	Problem 1 - Finned Radiator . . . . .	83
5.2	Problem 2 - Volume Heating in a Cylinder . . . . .	86
5.3	Problem 3 - Surface Heating of a Plate . . . . .	86
<b>6</b>	<b>References</b>	<b>101</b>
<b>7</b>	<b>Appendices</b>	<b>103</b>
	Appendix A - Summary of Input Commands . . . . .	105
	Appendix B - Consistent Units . . . . .	115
	Appendix C - Initial Time Step Estimation . . . . .	117
	Appendix D - Common Block and Array Storage . . . . .	121
	Appendix E - External Mesh Generator File Contents . . . . .	149
	Appendix F - Post-Processing File Contents . . . . .	151

# List of Figures

2.1	Finite elements in the COYOTE library; the linear and quadratic version of each element is available in the code. . . . .	5
2.2	Organization of the COYOTE code. . . . .	7
3.1	Coordinate system definition. . . . .	12
3.2	Notation for orthotropic conductivity tensor. . . . .	21
5.1	Schematic of finned radiator problem. . . . .	84
5.2	Finite element model for finned radiator. . . . .	85
5.3	Input file for COYOTE simulation of a finned radiator. . . . .	88
5.4	Temperature contours for finned radiator problem. . . . .	89
5.5	Schematic of volume heated cylinder problem. . . . .	90
5.6	Input file for COYOTE simulation of a volume heated cylinder. . . . .	91
5.7	User subroutine for COYOTE simulation of a volume heated cylinder. . . . .	92
5.7	Continuation of user subroutine for simulation of a volume heated cylinder. . . . .	93
5.8	Temperature histories for volume heated cylinder problem. . . . .	94
5.9	Schematic of plate heating problem. . . . .	95
5.10	Input file for COYOTE simulation of a surface heated plate. . . . .	96
5.11	User subroutine for COYOTE simulation of surface heated plate. . . . .	97
5.11	Continuation of user subroutine for simulation of a surface heated plate. . . . .	98
5.12	Temperature contours for the surface heated plate problem. . . . .	99
7.1	Temperature ratio versus Fourier number for smaller Biot numbers. . . . .	119
7.2	Temperature ratio versus Fourier number for larger Biot numbers. . . . .	120



# Chapter 1

## Introduction

The COYOTE computer code is a general purpose program package designed for the solution of heat conduction problems and other types of diffusion problems. The code is based on the Galerkin form of the finite element method (FEM). The present version of COYOTE represents a significantly enhanced edition of the original program which was first released in 1978.

The class of problems treated by COYOTE are basically those described by the standard heat conduction equation. The capability to simulate surface-to-surface radiation in conjunction with the thermal conduction problem is also available in COYOTE; capabilities for treating chemically reacting materials are supported in the software package. Though specifically intended for the solution of heat conduction problems, the code can be used for a wide variety of boundary and initial value problems. This generality stems from the analogy between the heat conduction equation and other diffusion equations encountered in engineering and physics. A partial list of application areas that are analogous to the heat conduction problem are listed below:

- Saturated or partially saturated flow in porous media
- Potential fluid flow
- Electrostatic fields
- Electric conduction
- Mass diffusion
- Lubrication flows

Many of these types of problems may be solved with COYOTE in either the steady-state or transient form.

A significant effort has been made during the design and development of COYOTE to create an analysis program that is easy to use. The basic code structure and many of the programming features found in COYOTE have been developed and refined from earlier versions of the program. The code has been written using standard FORTRAN 77 in an effort to increase its portability. COYOTE is one of a series of codes in the fluid mechanics and heat transfer area [1,2], all of which share a common code structure and input style. This feature allows a user ready access to a variety of analysis packages with a minimum of time spent on learning input format conventions.

The present document is intended to provide a detailed description of the input data necessary to access and execute the present version of the COYOTE code. The theoretical basis for the code and many of the numerical procedures used in the program are described in a companion document [3]. A detailed description of the use of the code on various verification, heat transfer and diffusion problems is provided in [4].

In the next chapter, a brief description of the program capabilities and organization is provided. Chapter 3 describes the input to the program; the penultimate chapter provides a discussion of programming and installation details. Several short example problems are included in Chapter 5 to illustrate typical input data files.

# Chapter 2

## Program Overview

The development of a reasonably efficient computer code for heat conduction or diffusion problems requires that the limits of applicability of the program be specifically and carefully defined. The present description is intended only as an overview of the major assumptions, capabilities and features of the program; a more complete outline of code limitations is provided in [3]. Brief chapters are also included to show the major organizational components of the program and to provide some of the background information that is a prerequisite to successful use of the code.

### 2.1 Program Features

COYOTE is intended for the analysis of heat conduction or other similar types of diffusion problems. Geometrically, this version of the program is appropriate for treating two-dimensional, plane or axially symmetric problems and fully three-dimensional analyses. The theoretical formulation assumes isotropic or orthotropic behavior for all materials. Further, the materials of interest may be heterogeneous with properties that vary in a quite general manner with spatial location, time and/or temperature and composition. Materials defined in the problem may undergo chemical reactions, the nature of which is quite arbitrary. Volumetric source terms may also be defined and allowed to vary in a general manner. The problems may be either independent of time or fully time-dependent in nature.

Boundary condition specification is very general and allows all of the standard types of thermal conditions to be imposed on a problem. Temperatures or heat fluxes may be specified as well as conditions representing convective and radiative exchange with a

surrounding environment. Radiative heat transfer across enclosures or between surfaces can also be included, subject to a few standard assumptions regarding radiation wave length and surface properties [3]. All boundary conditions may be general functions of spatial location, time and/or temperature. Material motion, in either an Eulerian or Lagrangian coordinate frame, can be simulated as well as the removal or addition of material. A general surface contact capability is also provided.

COYOTE relies on specialized external codes for mesh generation and graphical processing of solution data. Mesh data and output for graphical processing are read and written by COYOTE in a standard, well documented format that allows easy coupling with other programs. The data analysis portions of the code allow local heat fluxes and energy balances to be computed. A facility also exists for reporting results at arbitrary points within the domain in addition to nodal locations.

Two other essential parts of COYOTE are the element library and the solution procedures for the finite element equations. The element library is illustrated in Figure 2.1 and includes isoparametric quadrilaterals and triangles for two-dimensional applications and isoparametric tetrahedrons, wedges and hexahedrons for three-dimensional simulations. Specialty elements, such as bars and shells, are also available. Within each of these elements, the temperature is approximated using either linear or quadratic basis functions. For the analysis of time-independent problems, COYOTE provides the user with a choice of several variants of a Picard iteration scheme. A Newton algorithm is available, as an option, for the solution of coupled enclosure radiation and conduction problems. Transient problems are analyzed using either of two implicit integration schemes – a first-order, Euler method or a second-order trapezoid rule. Both integrators may be used in a predictor/corrector mode with a fixed timestep or a dynamic timestep algorithm. A first-order, explicit integration method is also available for use with similar types of timestep control. Details of these methods and their implementation can be found in [3].

## 2.2 Program Organization

COYOTE has been organized in a modular form in an effort to make the operation of the code understandable and to ease the task of code modification. As shown in the schematic of Figure 2.2, the code consists of a main program plus a number of major, task-oriented subroutines. Each of the major subroutines may access one or more subordinate routines plus the utilities attached to the main program.

Input to the program is organized into six distinct blocks that correspond to major tasks within the code. Within each block, keyword data is specified in a free-field format;

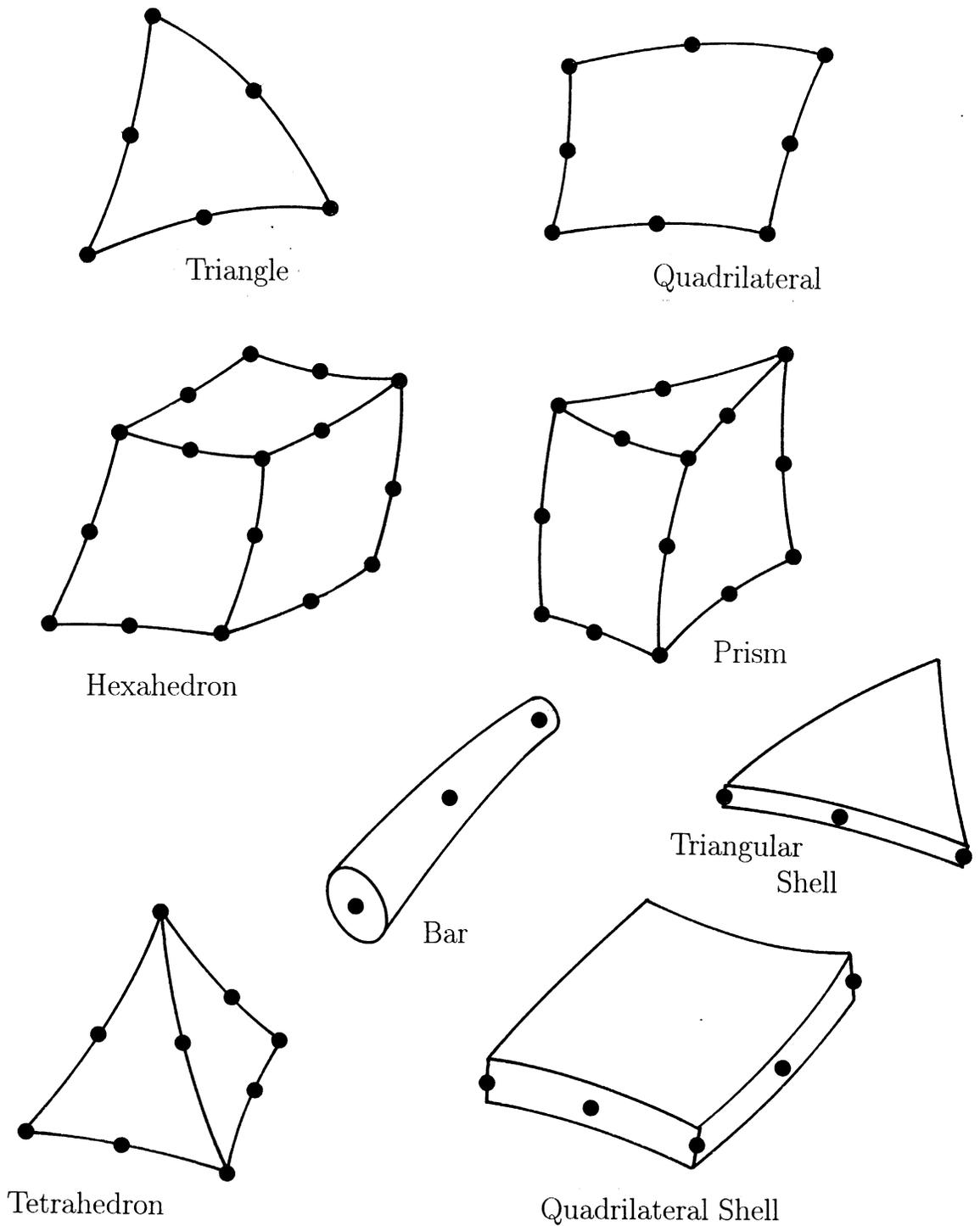


Figure 2.1: Finite elements in the COYOTE library; the linear and quadratic version of each element is available in the code.

data within the block may be ordered arbitrarily. A list of the six major data blocks, their associated keywords (with accepted abbreviations indicated), and a brief description of their function is given below.

- Title Data Block (**TIT**le) - specifies problem title and comments
- Material Data Block (**MAT**erial) - specifies material property data
- Problem Definition Data Block (**PRO**blem **DEF**inition) - specifies problem type, element block properties, boundary conditions and simulation options
- Solution Data Block (**SOL**ution) - specifies solution method and options
- Post-processing Data Block (**POST**) - specifies post-processing output and options
- Function Data Block (**TIME FUN**ction, **VAR**iable **FUN**ction, **USER CON**stants) - specifies function data and user defined constants

The large amount of data needed for a typical finite element analysis is handled in COYOTE by a combination of disk files and in-core memory. Data that is repeatedly required by several of the major subroutines (*e.g.*, nodal point coordinates, nodal point connectivity and current solution vectors) are stored in main memory. The in-core storage scheme makes use of a dynamic memory algorithm that allocates needed array space during program execution. Less frequently needed data (*e.g.*, input data file) and extremely large blocks of data (*e.g.*, previous solution vectors and viewfactors) may be stored on various disk files. The utilization of the files shown in Figure 2.2 is discussed in Section 4.3.

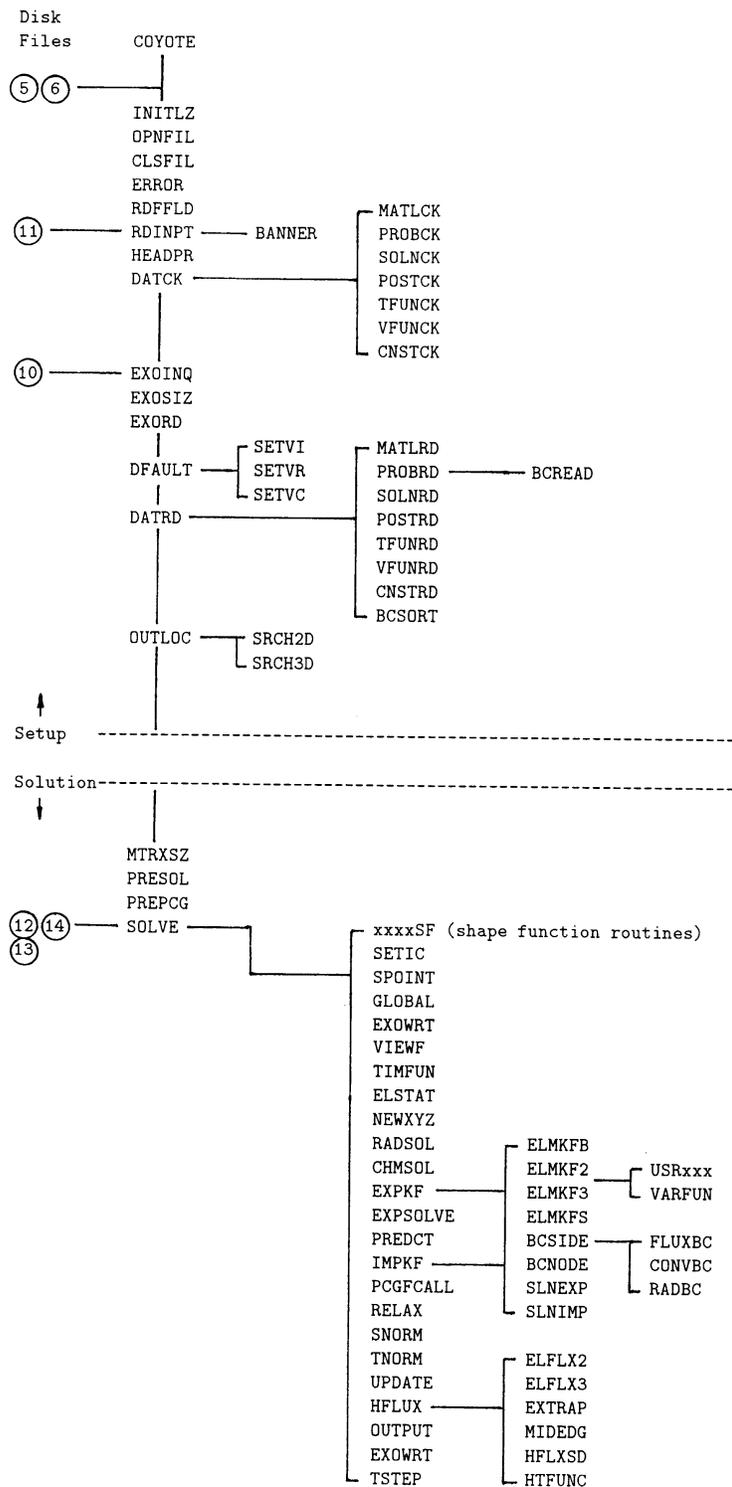


Figure 2.2: Organization of the COYOTE code.



# Chapter 3

## Input Guide

The structure of an input data file for the COYOTE program directly reflects the steps required to formulate, solve and analyze a finite element model for a heat conduction or diffusion problem. Input to the code consists of four types of cards <sup>1</sup>: a) Commentary cards, b) Data Block cards, c) Data cards and d) Termination cards. Input on any given card is in a free-field format style with the first entry usually being a keyword. Comments may appear at any point within the input file; this type of data is echoed to the output file but is otherwise ignored by the code. Data block cards and termination cards occur in pairs and bracket a series of data cards. The order in which data block cards appear in the input file is arbitrary. Data block cards may appear as many times as necessary within an input file but must always be closed by a termination card. Data cards within a data block may be ordered arbitrarily except for function data, which must appear in a logical, ascending order.

Data that is provided to COYOTE is previewed by the code in an effort to uncover obvious errors. All keywords are reviewed for syntax, proper termination of each data block is checked and the presence of all mandatory data blocks is verified. This data scan also serves to set certain critical sizing parameters needed for proper program execution.

The data block cards recognized by COYOTE are listed below in the order in which they are discussed in the subsequent chapters. This is also the order in which they would normally appear in a typical input file. A summary showing the form and content of the indicated data block cards and associated data cards is provided in Appendix A.

- Title Data Block

---

<sup>1</sup>Notice that throughout this manual input data is referred to in terms of “cards”; this is a convenient term that in actuality refers to a record on an input device.

- Material Data Block
- Problem Definition Data Block
- Solution Data Block
- Post-processing Data Block
- Time and Variable Function Data Blocks
- User Constants Data Block
- Termination Data

## 3.1 Input Syntax

The COYOTE input cards generally follow the standard format shown below:

KEYWORD=parameter1, parameter2, . . .

In some cases a keyword does not require any additional data and the parameter list is omitted.

In describing the input data, the following conventions have been adopted for the following sections:

- (a) Bold face words indicate keywords, *e.g.*, **MATerial**. Most keywords can be abbreviated to the first 3-5 characters of each word. Upper case characters indicate the shortest abbreviation that is recognized by the code.
- (b) Keyword data that is required for the proper specification of a problem and is mandatory for all input files is flagged with a □ symbol at the right hand margin.
- (c) Upper case words indicate a required alphanumeric input value, *e.g.*, USER. Most alphanumeric input can be abbreviated to the first 3-5 characters of each word. Upper case characters indicate the shortest abbreviation that is recognized by the code.
- (d) Lower case words and symbols imply that an alphanumeric or numerical value for the specified variable is expected, *e.g.*, tmax.

- 
- (e) All input values are specified in a free field format with successive variables separated by commas, equal signs or blanks. A convenient convention that is used in this manual uses blanks within keywords, equal signs to separate keywords from variable lists and commas for separating variables.
  - (f) The total number of input values allowed on any single input card (including all continuation cards) is presently limited to 50.
  - (g) Each alphanumeric input value is limited to twenty characters under the free-field format.
  - (h) The \$ character may be used to end an input card, with the remaining space on the card then being available for comments. A \$ character at the beginning of a line indicates a full comment line that is ignored by the code during execution.
  - (i) The \* character may be used to continue an input line onto the next data card. The continuation character should follow the last delimiter (comma or equal sign) on the card to be continued.
  - (j) *Italics* indicate optional parameters which may be omitted by using successive delimiters (commas or equal signs) in the input line. If the omitted parameter is not followed by any required parameters, no additional delimiters need be specified.
  - (k) ( ) indicates the data type for a particular input variable, *i.e.*, alphanumeric or character (C), real (R) or integer (I).
  - (l) < > indicates the default value for an optional parameter.
  - (m) The contents of each input line are indicated by underlining.
  - (n) All quantities associated with a coordinate direction are expressed in terms of the cartesian  $x, y$  or  $x, y, z$  coordinate system. The corresponding quantities for axisymmetric problems are obtained by the association of the radial coordinate,  $r$ , with  $x$  and the axial coordinate,  $z$ , with  $y$ . The global coordinate system used by the code is right-handed, as indicated in the sketch shown in Figure 3.1.
  - (o) Explanatory notes for various input options and data are numbered consecutively within a chapter and are located at the end of each Data Block.

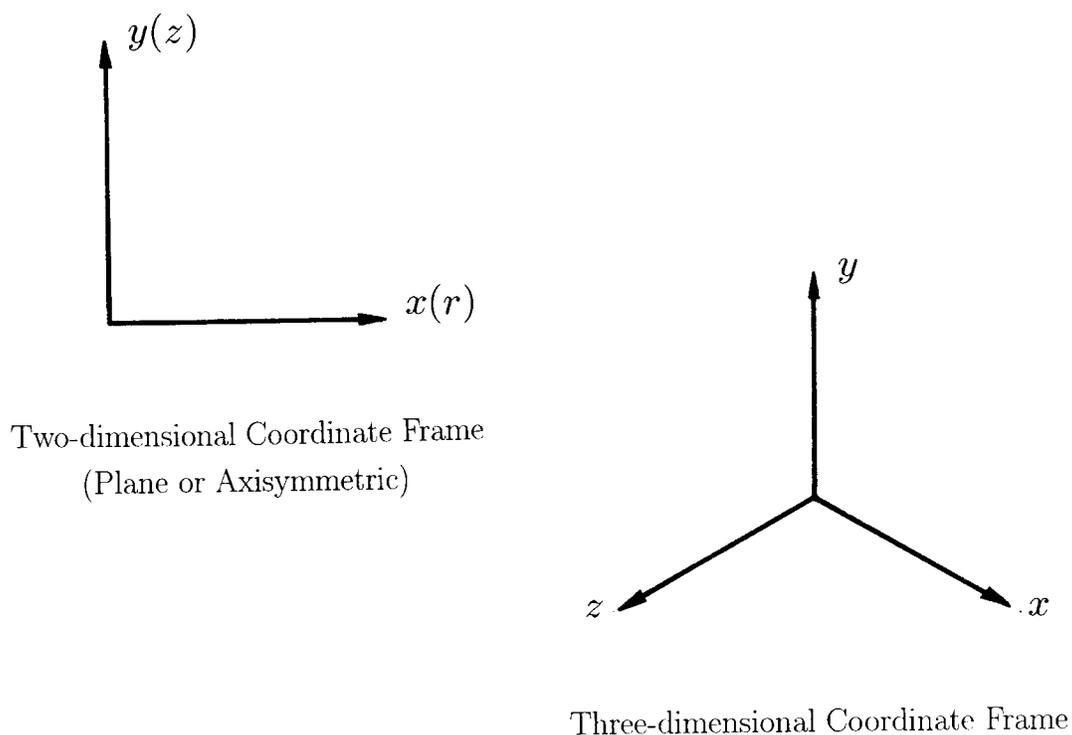


Figure 3.1: Coordinate system definition.

## 3.2 Title Data Block

The title data block allows the specification of an appropriate problem title and any additional commentary that is useful in documenting the simulation. This data block is optional. The title data block has the following form:

```

TITLe
THIS IS AN EXAMPLE OF A PROBLEM TITLE
.
.
THESE ARE EXAMPLES OF SUBTITLE LINES
UP TO 10 LINES OF PROBLEM DESCRIPTION MAY
BE INCLUDED ON THESE CARDS
.
.
ENDtitle

```

The first line following the **TITLE** keyword is taken as the problem title and is written to various output files to assist in the identification of a particular code execution. Following the title card, a number of subtitle or comment cards may be included to provide a description of the particular problem, modeling assumptions, material models, *etc.* Up to 10 subtitle lines may be specified. The commentary lines are reproduced at the beginning of the printed output file as problem documentation. The title data block is terminated with an **END** card.

### 3.3 Material Data Block

The input of material properties to COYOTE is accomplished through the material data block and its associated set of data cards. A separate material data block is required for each material used in a COYOTE simulation. Input of material data for each block is terminated by an **END** card. The material data block is of the following form:

```

MATerial, material name, material model                                □
.
.
Material Property Data
.
.
.
ENDmaterial                                                            □

```

where

material name (C) : is a required material name (see Note 1).

*material model* (C) <ISOtropic> : is the type of material model used to describe the thermal conductivity. When this parameter is omitted or set to ISOtropic, the material is modeled as isotropic with  $k_{ij} = k\delta_{ij}$ ; a parameter value of ORTHotropic indicates a material with a full conductivity tensor  $k_{ij}$  (see Note 2).

#### Material Property Data

Thermophysical property data for each material is specified by a series of data cards within the data block. Each property data card begins with a property keyword and is followed by the numerical value of the material property or an option that directs the

code to evaluate the property through a function or a user subroutine. Note that when a material is defined as a reactive material, the relevant chemical kinetic data must be specified for all of the ( $I$ ) species and ( $J$ ) reactions that describe the reactive behavior of the mixture. Thermophysical properties that depend on chemical composition should be evaluated through the user subroutine option where species concentration data is readily available. Endothermic or exothermic heat release due to chemical reaction is automatically accounted for when a reactive mixture is defined and need not be defined through the volume heating property for the material.

COYOTE does not contain any dimensional constants and, therefore, the units for the material properties are free to be chosen by the user. For convenience, a table of consistent units is given in Appendix B.

The individual material data cards recognized by COYOTE are listed below along with the various options available for each property.

### DENsity= $\rho$

### DENsity=USER

### DENsity=VFUNction, idvar

$\rho$  (R) <0.0> : specifies the value of the density.

USER (C) : specifies that the density will be evaluated by a user supplied subroutine, `USRDEN`.

VFUNction (C) : specifies that the density will be evaluated via a variable function (see Note 3).

idvar (I) : sets the variable function identification number (see Note 3).

### SPECific HEAT= $C_p$

### SPECific HEAT=USER

### SPECific HEAT=VFUNction, idvar

$C_p$  (R) <0.0> : specifies the value of the specific heat.

USER (C) : specifies that the specific heat will be evaluated by a user supplied subroutine, `USRCP`.

VFUNction (C), id (I) : specifies that the specific heat will be evaluated via a variable function (see Note 3).

idvar (I) : sets the variable function identification number (see Note 3).

**CONDUCTivity= $k_{11}, k_{22}, k_{33}$**  □

**CONDUCTivity=USER** □

**CONDUCTivity=VFUNction, idvar1, idvar2, idvar3** □

$k_{11}, k_{22}, k_{33}$  (R) : specify the values of the principle components of the thermal conductivity tensor. If the material is isotropic only the first value,  $k_{11}$ , is required; orthotropic materials in two-dimensional problems only require the first two components of the tensor  $k_{11}$  and  $k_{22}$ , to be specified (see Note 2).

USER (C) : specifies that the principle components of the conductivity tensor will be evaluated by a user supplied subroutine, **USRCON** (see Note 2).

VFUNction (C) : specifies that the principle components of the conductivity tensor will be evaluated via one or more variable functions (see Notes 2 and 3).

idvar1, idvar2, idvar3 (I) : set the variable function identification numbers for each of the principle components of the thermal conductivity tensor. If the material is isotropic only the first function idvar is required; orthotropic materials in two-dimensional problems only require the first two function idvars to be specified. Components with the same functional behavior may share a single function specification (see Note 3).

**TENSOR ROTation= $x\hat{x}, y\hat{x}, z\hat{x}, x\hat{y}, y\hat{y}, z\hat{y}$**

**TENSOR ROTation=USER**

$x\hat{x}, y\hat{x}, z\hat{x}, x\hat{y}, y\hat{y}, z\hat{y}$  (I or R)  $\langle 1,0,0,0,1,0 \rangle$  : represent the orientation of the principle material axes with respect to the global coordinate system. The six components specify the projection of the  $\hat{x}$  and  $\hat{y}$  principle material axes for the conductivity tensor onto the global coordinate axes (see Note 2).

USER (C) : specifies that the local orientation of the conductivity tensor will be evaluated by a user subroutine, **USRROT** (see Note 2).

**LATent HEAT= $L$**

$L$  (R)  $\langle 0.0 \rangle$  : specifies the value of the latent heat of fusion for the material

**SOLIDus TEMPerature= $T_{sol}$**

$T_{sol}$  (R) <0.0> : specifies the solidus temperature for the material

### LIQuidus TEMPerature= $T_{liq}$

$T_{liq}$  (R) <0.0> : specifies the liquidus temperature for the material

### ENTHalpy=USER

### ENTHalpy=VFUNction, idvar

USER (C) : specifies that the enthalpy for the material will be evaluated by a user supplied subroutine, **USRENT** (see Note 4).

VFUNction (C) : specifies that the enthalpy for the material will be evaluated via a variable function (see Notes 3 and 4).

idvar (I) : sets the variable function identification number (see Note 3).

### PHASe CHANge=*property, derivative method*

*property* (C) <SPECific HEAT> : specifies the material property that is used to evaluate the effective specific heat during a material change of phase. If this parameter is set to SPECific HEAT or left blank, the effective specific heat is computed from the latent heat value and the specified liquidus and solidus temperatures. A parameter setting of ENTHalpy causes the effective specific heat to be derived from the enthalpy specification. Note that the inclusion of the **PHASe CHANge** data card automates the inclusion of latent heat effects for the material. Phase change behavior could also be included directly through the proper manipulation of the **SPECific HEAT** function; use of an enthalpy function requires the **PHASe CHANge** procedure (see Note 4).

*derivative method* (C) <SPATial> : defines the method for computing the derivative of the enthalpy versus temperature function when the ENTHalpy option for phase change is specified. If this parameter is set to SPATial or left blank, spatial gradients of the enthalpy and temperature are used to compute the effective specific heat. A parameter setting of TIME causes time derivatives of the enthalpy and temperature to be used to evaluate the effective specific heat (see Note 4).

### EMISsivity= $\epsilon$

### EMISsivity=USER

**EMISSivity=VFUNction, idvar**

$\epsilon$  (R) <0.0> : specifies the value of the surface emissivity.

USER (C) : specifies that the surface emissivity will be evaluated by a user supplied subroutine, **USREMS**.

VFUNction (C) : specifies that the surface emissivity will be evaluated via a variable function (see Note 3).

idvar (I) : sets the variable function identification number (see Note 3).

**VOLume HEATing=Q****VOLume HEATing=USER****VOLume HEATing=TFUNction, idtim****VOLume HEATing=VFUNction, idvar**

$Q$  (R) <0.0> : specifies the value of the volumetric heat source.

USER (C) : specifies that the volumetric heat source will be evaluated by a user supplied subroutine, **USRVHS**.

TFUNction (C) : specifies that the volumetric heat source will be evaluated via a time function (see Note 3).

idtim (I) : sets the time function identification number (see Note 3).

VFUNction (C) : specifies that the volumetric heat source will be evaluated via a variable function (see Note 3).

idvar (I) : sets the variable function identification number (see Note 3).

**INITial TEMPerature= $T_{init}$** 

$T_{init}$  (R) <0.0> : specifies the initial temperature for the material

**REACtive MIXture=nspecies, nreactions, USER**

nspecies (I) : specifies the number of species to be defined for this material (see Note 5).

nreactions (I) : specifies the number of chemical reactions to be defined for this material (see Note 5).

**SPECies=name<sub>1</sub>, name<sub>2</sub>, ..., name<sub>nspecies</sub>**

$\text{name}_1 \dots \text{name}_{n_{\text{species}}}$  (C) : are the names of the chemical species defined for this material (see Notes 6 and 7).

**SPECies PHASe=phase<sub>1</sub>, phase<sub>2</sub>, ..., phase<sub>n<sub>species</sub></sub>**

$\text{phase}_1 \dots \text{phase}_{n_{\text{species}}}$  (C) : are the phases of each of the species for this material. The permissible phases are GAS or CONDensed (see Note 10).

**FRACTION CONDensed= frac**

frac (R) : is the condensed fraction of this material (see Note 10).

**INITial CONcentration= $N_1^0, N_2^0, \dots, N_{n_{\text{species}}}^0$**

$N_1^0 \dots N_{n_{\text{species}}}^0$  (R) <0.0> : specify the initial concentrations of the species for this material (see Note 7).

**MINimum CONcentration= $N_1^{\text{min}}, N_2^{\text{min}}, \dots, N_{n_{\text{species}}}^{\text{min}}$**

$N_1^{\text{min}} \dots N_{n_{\text{species}}}^{\text{min}}$  (R) <1.0E-8> : specify the minimum allowed concentrations of the species for this material (see Note 7).

**STERIC COEFFicients= $\beta_1, \beta_2, \dots, \beta_{n_{\text{reaction}}}$**

$\beta_1 \dots \beta_{n_{\text{reaction}}}$  (R) : specify the coefficients for the steric factors in the reactions defined for this material (see Note 7).

**PREExponential FACTor= $A_1, A_2, \dots, A_{n_{\text{reaction}}}$**

$A_1 \dots A_{n_{\text{reaction}}}$  (R) : specify the pre-exponential factors in the reactions defined for this material (see Notes 7 and 8).

**LPREExponential FACTor= $\ln A_1, \ln A_2, \dots, \ln A_{n_{\text{reaction}}}$**

$\ln A_1 \dots \ln A_{n_{\text{reaction}}}$  (R) : specify the natural logarithm of pre-exponential factors in the reactions defined for this material (see Notes 7 and 8).

**ACTivation ENERgy= $E_1, E_2, \dots, E_{n_{\text{reaction}}}$**

$E_1 \dots E_{nreaction}$  (R) : specify the activation energies in the reactions defined for this material (see Note 7).

**ENERgy RELease= $q_1, q_2, \dots, q_{nreaction}$**

$q_1 \dots q_{nreaction}$  (R) : specify the endothermic or exothermic energy release in the reactions defined for this material (see Note 6). The energy release data must have the correct sign to distinguish an exothermic reaction from an endothermic reaction. The standard convention in COYOTE requires that heat addition (exothermic reaction) to a material be positive and heat extraction (endothermic reaction) from a material be negative.

**CONcentration EXPonents, species no.= $\mu_{i1}, \mu_{i2}, \dots, \mu_{i,nreaction}$**

species no. (I) : specifies the species number ( $i$ ) for the exponent data

$\mu_{i1} \dots \mu_{i,nreaction}$  (R) : specify the concentration exponents for the ( $i$ th) species and reactions defined for this material (see Note 7).

**STOichiometric COEFFicients, species no.= $\nu_{i1}, \nu_{i2}, \dots, \nu_{i,nreaction}$**

species no. (I) : specifies the species number ( $i$ ) for the stoichiometric data

$\nu_{i1} \dots \nu_{n\text{spec},n\text{reaction}}$  (R) : specify the stoichiometric coefficients for the ( $i$ th) species and reactions defined for this material (see Note 7).

**CHEMistry ACTivation TEMPerature= $T_{chem}$**

$T_{chem}$  (R) <0.0> : sets the threshold temperature above which the reaction kinetics are activated in the solution process (see Note 9).

**Notes:**

- 1) The material name assigned to each material data block must be unique. This name is also used in the problem definition chapter (see Section 3.4) to allow material properties to be associated with individual blocks of elements. One and only one set of material property data cards should be supplied for each material name. No abbreviations of the name are allowed; the name must be less than 20 characters.

- 2) The material models allowed in COYOTE include materials with either an isotropic or orthotropic conductivity tensor. For isotropic materials,  $k_{ij} = k$ , and the conductivity is specified completely by setting the value of the scalar  $k$ . Temperature dependence of the isotropic conductivity can be implemented via a variable function or a user supplied subroutine. Bar elements are limited to an isotropic conductivity model. For orthotropic materials,  $k_{ij}$  is defined as  $\delta_{ij}k_{ij} = k_{ii}$ , and the conductivity is determined by specifying the components of the tensor with respect to the principle material axes. Temperature dependence for each of the principle conductivities can be specified independently through the variable functions or a user subroutine. If the principle material axes are not aligned with the global coordinate axes then the **TENSOR ROTation** data card must be used to supply the appropriate coordinate transformation (rotation). Referring to Figure 3.2, the  $\hat{x}$ ,  $\hat{y}$  and  $\hat{z}$  axes indicate the principle material axes while  $x$ ,  $y$ , and  $z$  denote the global reference frame. The parameters  $x\hat{x}$ ,  $y\hat{x}$ ,  $z\hat{x}$  on the **TENSOR ROTation** data card refer respectively to the  $x$ ,  $y$ ,  $z$  projections of a vector in the positive  $\hat{x}$  direction. Similarly, the parameters  $x\hat{y}$ ,  $y\hat{y}$ ,  $z\hat{y}$  refer to the  $x$ ,  $y$ ,  $z$  projections of a vector in the positive  $\hat{y}$  direction. Note that these vectors  $(\hat{x}, \hat{y})$  need not be unit vectors. For two-dimensional applications only the  $x\hat{x}$ ,  $y\hat{x}$  projections need to be specified for a vector in the positive  $\hat{x}$  direction. To illustrate the definition of a tensor rotation, a  $45^\circ$  rotation about the  $y$  axis is specified by  $\langle x\hat{x}, y\hat{x}, z\hat{x} \rangle = \langle 1, 0, 1 \rangle$  and  $\langle x\hat{y}, y\hat{y}, z\hat{y} \rangle = \langle 0, 1, 0 \rangle$ . Rotations defined by parameters on the **TENSOR ROTation** card are applied uniformly to all elements with the given material name. If the tensor rotation is required to vary with spatial location and must be computed on an element-by-element basis, then the user subroutine, **USRROT**, must be employed (see Section 3.11.1).
- 3) Material data that has a simple functional dependence on temperature or time will usually be specified via a variable function (**VFUNction**) or a time function (**TFUNction**). The correspondence between each material property and the appropriate function is determined from the integer identification (**idvar** or **idtim**) number. Within each category of function, the **id** must be unique. Function definitions are described in Sections 3.7 and 3.8. For material data that has a complex functional form or depends on chemical composition, a user subroutine is recommended.
- 4) Latent heat release during a material change of phase may be specified by any of several methods, two of which are available through the **PHASe CHANge** specification. In the specific heat method, the input latent heat,  $L$ , is divided by the liquidus and solidus temperature difference,  $T_{liq} - T_{sol}$ , to generate an equivalent specific heat function. This equivalent specific heat is added to the base specific heat specified by the **SPECific HEAT** card to produce a function that is valid over the phase change temperature range. A second option allows the enthalpy for

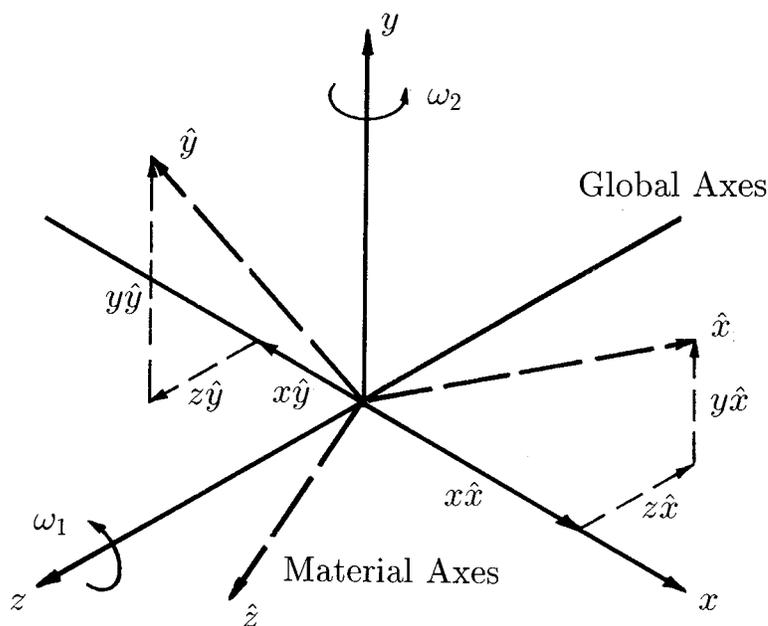


Figure 3.2: Notation for orthotropic conductivity tensor.

the material to be input as a function of temperature. In this case a numerical derivative of the enthalpy function with respect to temperature is used to define an effective specific heat. For this option the enthalpy must be defined over the entire temperature range for the problem and the specific heat card is not used. Two methods for computing the local slope of the enthalpy versus temperature function are included in COYOTE; these methods are explained in detail in [3]. Other methods for including latent heat release are possible and do not involve the **PHASe CHANge** specification. The **SPECific HEAT** input can be defined such that latent heat is accounted for in the functional definition; either a variable function or a user subroutine can be used for this type of definition. The effects of latent heat release could also be included as part of a temperature-dependent, volumetric heat release. Again, a variable function or user subroutine could be used for this definition under the **VOLume HEATIng** input. Details and cautions for the use of these methods can be found in [3,4].

- 5) The number of species and reactions that can be defined for any material is currently limited to 45. This limit may be increased by altering the number of data fields read by the free field reader.
- 6) The species defined for all reactive materials must have globally unique names so that they may be correctly identified in the post-processing file. Specie names may

be 20 characters long, however, only the first 8 characters are used for checking uniqueness.

- 7) In specifying chemical kinetic data the order of the data entries is important. Species data must always be ordered in the same sequence from 1 to *nspecies* and must correspond to the ordering set by the names of the species under the **SPECies** card. Reaction data must always be ordered in the same sequence from 1 to *nreaction* where the sequence is set (implicitly) by the user. For two-dimensional data arrays, such as the concentration exponents and the stoichiometric coefficients, the data entry is sequenced from 1 to *nreaction* with a separate card for each species.
- 8) The pre-exponential factors for a reactive mixture may be specified in either of two forms. The **PREexponential FACTor** format allows this data to be specified directly; the **LPRExponential FACTor** format allows the natural log of the pre-exponential factors to be input. This latter format permits smaller numbers to be input and lessens the chance for data errors.
- 9) For simulations involving chemical reactions, the reaction rates and heat generation are strongly coupled to the temperature level within the material. In many situations, the temperature rise in the material is slow and the chemical reactions are not significant until late in the process when some critical temperature level is reached. The **CHEMistry ACTivation TEMPerature** parameter sets a threshold temperature, by material, for the activation of the chemical reaction equations. Prior to the nodal point temperatures within an element reaching  $T_{chem}$ , the chemistry computations for the element are skipped with the species concentrations remaining at their initial values. When any nodal temperature within an element reaches the specified threshold, all chemistry computations are performed for the element.
- 10) For simulations involving chemical reactions, the reacted gas fraction can be computed and made available for post processing or for input to a reactive constitutive model in a solid mechanics code. The gas fractions are computed for each element in a reactive material at the chemistry points (integration points). The computation of gas fractions is specified by the presence of a **SPECies PHASe** card for any reactive material. Both a **SPECies PHASe** card and a **FRACTION CONDensed** card must be present for each reactive material. The gas fractions (at element centroids) can be output to the post-processing EXODUS file by including **GASFrac** as a parameter on the **CHEMistry DATA** card in the post processing data block.

## 3.4 Problem Definition Data Block

The problem definition data block is used to define the general attributes of the heat conduction problem that is to be solved. Normally, only one such data block is required. Input of problem definition parameters is terminated by an **END** card. The problem definition data block is of the following form:

```

PROBLEM DEFINITION, format □
.
.
Problem Definition Data
.
.
.
ENDproblem □

```

where

*format* (C) <EXODus> : is the name of the format used to provide mesh data to COYOTE. Currently the only supported data format is the EXODUS II format [6].

### Problem Definition Data

The setting of various program and modeling options is accomplished through a series of data cards within the problem definition data block. Each option begins with a keyword and may be followed by one or more parameters that define the option. The individual problem definition data cards recognized by COYOTE are listed below with the various parameter choices.

**GEOMETRY=type** □

type (C) : specifies the type of geometry used in the heat conduction problem. The type parameter should be set to 2D for planar, two-dimensional models, AXISym for axisymmetric, two-dimensional models or 3D for fully three-dimensional models.

**ELEMENT BLOCK= block id, material name, integration rule** □

block id (I) : specifies the previously assigned identifier for a block of elements in the mesh. The elements within this block are expected to all be the same material and be the same type of finite element (see Note 1).

material name (C) : specifies the material name for the element block. This identifier should correspond to one of the names used on the **MATerial** data block card.

*integration rule* (I) : specifies the integration (quadrature) rule used to construct the finite element coefficient matrices (see Note 2).

### BCtype=bcname, nodeset id, bc specification, *mfactor*

bcname (C) : specifies the name for the type of boundary condition applied to a node set. The permissible node set boundary conditions are TEMPerature and HEAT SOURce (see Note 3). Note that the HEAT SOURce option specifies a (nodal) point source of energy and should therefore have units of energy/time.

nodeset id (I) : specifies the previously assigned node set identifier for a group of boundary conditions (see Note 3).

bc specification : specifies the method for evaluating the boundary condition. This parameter may be replaced with any of the following:

bcvalue (R) : the constant value of the boundary condition.

USER (C) : the boundary condition will be evaluated by a user supplied subroutine. The subroutine for the TEMPerature boundary condition is **USRT** and the subroutine for the HEAT SOURce is **USRQ**.

TFUNction (C), idtim (I) : the boundary condition will be evaluated via a time function with time function identification number, idtim (see Note 5).

VFUNction (C), idvar (I) : the boundary condition will be evaluated via a variable function with variable function identification number, idvar (see Note 5). This parameter choice is only available for the HEAT SOURce boundary condition.

*mfactor* (R) <1.0> : is a multiplying factor that is applied to the value of the boundary condition. This factor is not used with the user subroutine option.

### BCtype=bcname, sideset id, bc specification, *mfactor*

bcname (C) : specifies the name for the type of boundary condition applied to a side set. The permissible side set boundary condition for this format is HEAT FLUX (see Note 4).

- sideset id (I) : specifies the previously assigned side set identifier for a group of boundary conditions (see Note 4).
- bc specification : specifies the method for evaluating the boundary condition. This parameter may be replaced with any of the following:
- bcvalue (R) : the constant value of the boundary condition.
- USER (C) : the boundary condition will be evaluated by a user supplied subroutine, **USRFLX**.
- TFUNCTION (C), idtim (I) : the boundary condition will be evaluated via a time function with time function identification number, idtim (see Note 5).
- VFUNCTION (C), idvar (I) : the boundary condition will be evaluated via a variable function with variable function identification number, idvar (see Note 5).
- mfactor* (R) <1.0> : is a multiplying factor that is applied to the value of the boundary condition. This factor is not used with the user subroutine option.

**BCtype=bcname, sideset id, COEFFicient = coef specification, *mfactor*,**

**TREference = ref specification, *mfactor***

- bcname (C) : specifies the name for the type of boundary condition applied to a side set. The permissible side set boundary conditions for this format are CONvection and RADiation (see Note 4).
- sideset id (I) : specifies the previously assigned side set identifier for a group of boundary conditions (see Note 4).
- coef specification : specifies the method for evaluating the convection heat transfer coefficient or radiation heat transfer form factor. This parameter may be replaced with any of the following:
- cvalue (R) : the constant value of the coefficient or form factor.
- USER (C) : the coefficient or form factor will be evaluated by a user supplied subroutine, **USRHTC** or **USRFF**.
- TFUNCTION (C), idtim (I) : the coefficient or form factor will be evaluated via a time function with time function identification number, idtim (see Note 5).
- VFUNCTION (C), idvar (I) : the coefficient or form factor will be evaluated via a variable function with variable function identification number, idvar (see Note 5).
- mfactor* (R) <1.0> : is a multiplying factor that is applied to the value of the coefficient or form factor. This factor is not used with the user subroutine option.

ref specification : specifies the method for evaluating the reference temperature for convection or radiation. This parameter may be replaced with any of the following:

tvalue (R) : the constant value of the reference temperature.

USER (C) : the reference temperature will be evaluated by a user supplied subroutine, `USRTRC` or `USRTRR`.

TFUNction (C), idtim (I) : the reference temperature will be evaluated via a time function with time function identification number, idtim (see Note 5).

VFUNction (C), idvar (I) : the reference temperature will be evaluated via a variable function with variable function identification number, idvar (see Note 5).

*mfactor* (R)  $<1.0>$  : is a multiplying factor that is applied to the value of the reference temperature. This factor is not used with the user subroutine option.

**BCtype**=bcname, SURFace=sideset id1, SURFace=sideset id2,

COEFFicient = coef specification

**BCtype**=bcname, BLOCk=block id1, SURFace=sideset id1,

COEFFicient = coef specification

**BCtype**=bcname, BLOCk=block id1, BLOCk=block id2,

COEFFicient = coef specification

bcname (C) : specifies the name for the type of boundary condition applied between element blocks, side sets or an element block and a side set. The permissible boundary condition for this format is `CONTACT` (see Note 6).

sideset id1, sideset id2 (I) : specifies the previously assigned side set identifiers for two groups of boundary conditions (see Note 6).

block id1, block id2 (I) : specifies the previously assigned element block identifiers for two element blocks (see Note 6).

coef specification : specifies the method for evaluating the contact or gap heat transfer coefficient. This parameter may be replaced with any of the following:

cvalue (R) : the constant value of the coefficient.

USER (C) : the coefficient will be evaluated by a user supplied subroutine, `USRHTG`.

TFUNction (C), idtim (I) : the coefficient will be evaluated via a time function with time function identification number, idtim (see Note 5).

VFUNction (C), idvar (I) : the coefficient will be evaluated via a variable function with variable function identification number, idvar (see Note 5).

**BCtype=bcname, sideset id, enclosure no.**

bcname (C) : specifies the name for the type of boundary condition applied to a side set. The permissible side set boundary condition for this format is ENClosure RADiation (see Note 4).

sideset id (I) : specifies the previously assigned side set identifier for a group of boundary conditions (see Note 4).

enclosure no. (I) : the number of the radiation enclosure that contains this side set (see Note 7).

**ENClosure=enclosure no., enclosure type, blocking option,**

**smoothing option, row-sum tolerance,  $Area_{\infty}$ ,  $T_{\infty}$ ,  $\epsilon_{\infty}$**

enclosure no. (I) : specifies the number of the radiation enclosure (see Note 7).

enclosure type (C) : specifies if the enclosure is fully contained or has one or more open surfaces. For a completely contained enclosure this parameter should be set to FULL; an enclosure that contains an open boundary should have this parameter set to PARTial. Partial enclosures must also have specified the last three (optional) parameters on this card.

*blocking option* (C) <NOBL> : indicates if third surface shadowing needs to be considered within the enclosure. This parameter should be set to BLOCKing if third surface shadowing is present and should be omitted or set to NOBLOCKing if the enclosure is unobstructed.

*smoothing option* (C) <NOSM> : indicates if a least squares smoothing of the view factors is required. This parameter should be set to SMOOth if least squares smoothing is desired and should be omitted or set to NOSMOoth if smoothing is not required.

*row-sum tolerance* (R) < $1.0 \times 10^{-4}$ > : specifies the maximum tolerance for row-sums of view factors, requires  $\sum F_{ij} - 1.0 < tolerance$ , for each row.

$Area_{\infty}$  (R) <0.0> : specifies the total area associated with the open boundary.

$T_{\infty}$  (R) <0.0> : specifies the temperature of the open boundary.

$\epsilon_{\infty}$  (R) <0.0> : specifies the emissivity of the open boundary.

**VIEWfactor IO=type, filename, format**

type (C) : specifies the direction of the view factor data transfer. When this parameter is set to READ, view factors are transferred to COYOTE/CHAPARRAL from an existing file. A parameter setting of WRITe saves the view factors computed by COYOTE/CHAPARRAL.

filename (C) : specifies the file name for the view factor data (see Note 8).

*format* (C) <BIN> : specifies the format under which the view factor data is written to a file. The allowable formats are specified by the following parameter settings: ASCIi, BINary (native machine binary), XDR and NETcdf. A format setting is not required for reading view factor data.

### **VIEWfactor COMPuTation=method, storage format, resolution, print option**

method (C) : specifies the method used to compute the view factors. The hemicube algorithm is employed with a parameter setting of HEMICube. Access to the double area integration, contour integration and/or semi-analytic algorithms are obtained by a parameter setting of FACET (see Note 9).

*storage format* (C) <WRLE> : specifies the compression format used internally by CHAPARRAL to store the view factor data. The allowable options are selected by the following parameter settings: NOSTore (no storage), NOCompress (no data compression), WRLE (word run length encoding), BRLE (byte run length encoding) and LZWE (LZW encoding) (see Note 9).

*resolution* (I) <50> : specifies the resolution for the hemicube view factor algorithm. The resolution must be specified as an even integer (see Note 9).

*print option* (C) <NONE> : specifies the printout level during view factor computation (see Notes 9 and 10). The permissible values for this parameter are NONE, SUMmary and EXTended.

### **VIEWfactor GRID=enclosure no., max surface intervals, no. rotational divisions, no. x-grid divisions, no. y-grid divisions, no. z-grid divisions, min separation distance, clipping plane scale factor, no. of threads**

enclosure no. (I) : specifies the number of the radiation enclosure (see Note 7).

*max surface intervals* (I) <5> : specifies the maximum number of subdivisions used for each element face or edge during the computation of a view factor by the area integration method (see Note 10).

*no. rotational divisions* (I) <8> : specifies the number of rotational intervals used to resolve an axisymmetric geometry during the computation of view factors by the area integration method (see Note 10).

*no. x-grid, y-grid, z-grid divisions* (I)  $\langle 1,1,1 \rangle$  : specifies the number of sorting bins in each coordinate direction that are used during the checking for third surface shadowing (see Note 10).

*min separation distance* (R)  $\langle 5.0 \rangle$  : specifies the minimum distance within which surface subdivision will be employed (see Note 10).

*clipping plane scale factor* (R)  $\langle 0.5 \rangle$  : specifies the near clip plane distance to be the product of the clipping plane scale factor and the minimum element effective radius. This parameter may be set to USER to have the code compute the near clip plane distance. (see Note 10).

*no. of threads* (I)  $\langle 1 \rangle$  : specifies the number of threads for multithreading.

### VELOCITY=type, block id, velocity specification

type (C) : specifies the type of reference frame used to describe the motion of the material. If this parameter is set to EULERian, the material in the element block is assumed to move continuously through a fixed mesh with the velocity field specified on this card. A specification of LAGRangian indicates that the element block and its associated mesh will be moved according to a set of prescribed velocities or displacements. A parameter setting of EXTERNAL implies that the motion of the element block is Lagrangian in nature and will be supplied from an external source, such as a solid mechanics code (see Notes 11 and 12).

block id (I) : specifies the previously assigned block identifier for the block of elements for which material motion is allowed.

velocity specification : specifies the method for evaluating the velocity of the material. This parameter may be replaced with any of the following:

$U_x, U_y, U_z$  (R) : the constant values of the  $x, y$  and  $z$  components of material velocity.

USER (C) : the material velocity will be evaluated by a user supplied subroutine, `USRVEL`.

TFUNCTION (C),  $idtim1, idtim2, idtim3$  (I) : the  $x, y$  and  $z$  velocity components for the material will be evaluated via time functions with separate time function identification numbers (see Note 5).

### EXTERNAL NODAL FIELD=name1, name2, ... , namen

name (C) : specifies the names of nodal point variables that are to be read from an external EXODUS II file for use in the COYOTE simulation. Variable

names that are currently recognized by COYOTE include: DISPLX, DISPLY, DISPLZ (components of the material displacement), VELX, VELY, VELZ (components of the material velocity), MAG (magnetic field) and JHEAT (Joule heating) (see Note 12).

**EXTernal ELEMENT FIELD=name1, name2, ... , namen**

name (C) : specifies the names of element variables that are to be read from an external EXODUS II file for use in the COYOTE simulation. Variable names that are currently recognized by COYOTE include: STATUS (element birth/death status) (see Notes 12 and 15).

**PRINTed OUTPUT=type**

type (C) <SUMmary> : specifies the extent of printed output that is written to the output file for the problem setup portion of the run. The permissible values for type include SUMmary, EXTended and DEBug.

**OUTPUT LOCations=e1, e2, e3, ... , en TO em, ...**

e1, e2, e3, ... , en TO em, ... (I) : is a list of element numbers for which output data will be printed during the solution process. Element numbers may be listed individually or with the syntax en TO em, which indicates an inclusive string of elements. A maximum of forty data entries may be listed on a single card; multiple data cards are permitted.

**SPECIAL OUTPUT=number of points, x1, y1, z1, x2, y2, z2 ....**

number of points (I) : specifies the total number of special points at which the solution is to be computed (see Note 13).

x1, y1, z1, x2, y2, z2 ... (R) : the coordinates of the special output points. If the problem is two-dimensional, only the x, y coordinates should be listed.

**HEAT FLUX=sideset id1,sideset id2, ... sideset idn**

sideset idn (I) : specifies the previously assigned side set identifiers for which total energy computations are to be performed. If the optional sideset id's are omitted, only nodal values of the flux components will be computed (see Note 14). A maximum of twenty sideset id's may be listed.

**HEAT FUNCTION= $\mathcal{H}_0$** 

$\mathcal{H}_0$  (R) <0.0> : specifies the value of the heat function at the first node in the first element processed (see Note 14).

**DEAth, block id, level, *variable*, *mode***

block id (I) : specifies the previously assigned identifier for the block of elements that is to have individual elements removed from the simulation as the specified criteria is reached (see Note 15).

level (R) : sets the critical level for the variable used to remove the elements.

*variable* (C) <TEMPerature> : specifies the name of the variable that controls elimination of elements within the block. This parameter may be set to TEMPerature, TDOT (temperature rate) or to one of the names of the species in a chemically reactive material (see Note 15).

*mode* (C) <MAXimum> : sets the type of criteria used for removal of the elements. This parameter should be set to MAXimum if the element is to be removed when a representative element value exceeds the specified critical level and should be set to MINimum if the removal occurs when the element value falls below the critical level (see Note 15).

**DELeTe MATerial, block id, *variable*, *value***

block id (I) : specifies the previously assigned identifier for the block of elements that is to be removed from the simulation as the specified criteria is reached (see Note 15).

*variable* (C) : specifies the name of the variable that controls elimination of the element block. This parameter is presently limited to the value TIME (see Note 15).

*value* (R) : sets the critical value for the variable used to remove the element block.

**ADD MATerial, block id, *variable*, *value***

block id (I) : specifies the previously assigned identifier to the block of elements that is to be activated during the simulation as the specified criteria is reached (see Note 15).

*variable* (C) : specifies the name of the variable that controls activation of the element block. This parameter is presently limited to the value TIME (see Note 15).

value (R) : sets the numerical value for the criteria used to activate the element block.

### SIGma= $\sigma$

$\sigma$  (R) : the value of the Stefan-Boltzmann constant

### GAS CONstant= $\mathcal{R}$

$\mathcal{R}$  (R) : the value of the gas constant

### Notes:

- 1) Element blocks form a basic grouping for data within COYOTE and also within the input and output EXODUS II files. An element block consists of a group of finite elements that must share the same geometry (shape and number of nodes), material and number of element attributes. The definition of an element block occurs during the mesh generation process and its precise configuration is at the discretion of the user as long as the previously defined restrictions are respected. Note that an element block is not necessarily an inclusive grouping; multiple element blocks with the same element type, material and attributes may be defined if this is convenient or required for the particular application. Associated with each element block is a unique integer identifier (block id) that is assigned by the user at the time of block creation. Data assignments for an element block are always accomplished within COYOTE through reference to the element block id. See reference [5] for further details regarding element blocks.
- 2) The quadrature rules used to integrate the finite element matrices can be varied from their default definitions through specification of the optional integration rule number. Such changes are not generally recommended but in some circumstances (*e.g.*, the occurrence of highly distorted elements or the need to reduce the cost of element construction) may be warranted. Listed in Table 3.1 are the quadrature rule numbers available for each element type, the number of integration points in each formula and an indication of the default rule. Note that for problems involving reactive materials, changing the quadrature formula will normally change the number of chemistry computation points within the element. Quadrature rules are defined for individual element blocks; the same rule should be defined for all elements of the same type. If multiple rules are specified for one element type, the last rule input will be used.

- 3) Boundary conditions for individual nodal points in a mesh are assigned in COYOTE by association of the name (type) of the boundary condition with each node set identifier (nodeset id). The nodeset id is a unique integer that is assigned by the user to one or more nodes during the mesh generation process. Note that spatial variations in the nodal boundary condition can be controlled through a distribution factor that is also specified during mesh generation. The nodeset ids and distribution factors are passed to COYOTE through the EXODUS II file. See reference [5] for further details.
- 4) Boundary conditions for element surfaces or edges are assigned in COYOTE by association of the name (type) of the boundary condition with each side set identifier (sideset id). The sideset id is a unique integer that is assigned by the user to one or more element surfaces (edges) during the mesh generation process. Note that spatial variations in the boundary condition over the element surface can be controlled through a set of distribution factors that are also specified during mesh generation. The sideset ids and distribution factors are passed to COYOTE through the EXODUS II file. See reference [5] for further details.
- 5) Simple variations in boundary conditions assigned to element nodes or surfaces (edges) may be conveniently specified by assigning a functional variation to the boundary condition. Time functions and variable functions (*i.e.*, functions of temperature) are described in Sections 3.7 and 3.8. Each function is assigned an integer id which is then used in the boundary condition specification. Boundary condition node sets and side sets may share functions; the *mfactor* can be used to scale individual functions. Spatial variations in a boundary condition are controlled through the use of distribution factors (see Notes 3 and 4) or a user subroutine.
- 6) The contact boundary condition allows a general (gap) heat transfer condition to be specified between any two surfaces in a problem. The three different forms of this boundary condition allow the participating surfaces to be conveniently identified with increasing generality. The most restrictive case is paired surface contact in which two surfaces with distinct side set ids are allowed to interact thermally with a specified heat transfer coefficient. A subset of this specification is self-contact in which a single surface can interact with itself. This case is invoked by replicating the side set id in the boundary condition specification. A more general contact specification identifies an element block and a surface that may interact. In this case, all surfaces of the element block are considered for thermal interaction with the specified surface. The third option is the most general and allows two element blocks to interact. All surfaces in each block are considered for contact with the opposing block. Self-contact for a single element block can be invoked by replicating the block id in the boundary condition specification. In all cases the gap heat

Element Type	Rule Number	Number of Integration Points	Default Rule for Each Element	Rule for Flux Evaluation
Bar	1	1		
	2	2	Linear	*
	3	3	Quadratic	
Triangle	1	1		
	2	3	Linear	*
	3	4		
	4	7	Quadratic	
Quad	1	1		
	2	4 ( $2 \times 2$ )	Linear	*
	3	9 ( $3 \times 3$ )	Quadratic	
Tetrahedron	1	1		
	2	4	Linear	*
	3	5	Quadratic	
Prism	1	1		
	2	6	Linear	*
	3	12	Quadratic	
	4	21		
Hexahedron	1	1		
	2	6		
	3	8 ( $2 \times 2 \times 2$ )	Linear	*
	4	14	Quadratic	
	5	27 ( $3 \times 3 \times 3$ )		
Shell (Tri)	1	1		
	2	3	Linear	*
	3	4		
	4	7	Quadratic	
Shell (Quad)	1	1		
	2	4 ( $2 \times 2$ )	Linear	*
	3	9 ( $3 \times 3$ )	Quadratic	

Table 3.1: Quadrature rules for elements in COYOTE

transfer condition is applied for all nodes and elements that are found to be in contact during a given time step or iteration. The gap condition can be extended to finite gap separations if the user subroutine option is used for the coefficient specification. Note that bar and shell elements are not included in this capability; only the linear continuum elements can be used with the current contact algorithm.

- 7) COYOTE allows multiple enclosures to be defined for the simulation of surface-to-surface radiation heat transfer. Each enclosure is identified by a user defined integer enclosure number; enclosures should be numbered consecutively starting with 1. The element surfaces (edges) that form each enclosure are identified by one or more sideset ids; a **BCtype= ENClosure RADiation** card must be provided for each side set in the enclosure. Enclosure data is specified on the **ENClosure** data card, one of which is required for each enclosure. Note that bar elements are not included in the enclosure radiation algorithm. Shell elements are acceptable in an enclosure when the shell face is the radiating surface; radiation from the edge of a shell is not permitted.
- 8) View factor data is computed by the code CHAPARRAL [6] and made available to COYOTE through an internal transfer. This data may also be stored on a disk file, in various formats, for future use. The local filename (pathname) is required when view factor data is to be written or read from this file. The filename is limited to 20 characters.
- 9) The CHAPARRAL code [6] contains numerous options for the computation, compression and storage of view factor data during a COYOTE run. Many of these options are of little interest to the casual user and can be safely left at their default settings. For extremely large problems, the user may wish to consult [6] for details on other parameter settings to ensure an efficient solution process and storage of data.
- 10) View factors computed by the standard double area integration, contour integration, *etc.* methods require the specification of a background grid for efficient operation. These methods have been adopted from the FACET code [7] and are available as part of CHAPARRAL [6]. The background grid is primarily used for the sorting of interacting surfaces, especially when third-surface shadowing is present. The *max surface intervals*, *no. rotational divisions*, *min seperation distance*, and *clipping plane scale factor* parameters determine how accurately the surface of any element is gridded during a view factor computation. These parameters can have a very strong influence on the overall view factor computation time.
- 11) Lagrangian velocity specifications are used in COYOTE to move the appropriate element block and compute an updated set of nodal coordinates. Nodal velocities

are not stored as variables but simply used to generate total displacements which are stored and made available in the EXODUS II output file. When kinematic data is supplied from an external source, either velocity or displacement fields may be used. Internally supplied kinematics are limited to velocity fields. Note that it is not permissible to mix internal and external specifications for kinematic data since the external source is expected to supply the appropriate fields for the entire mesh.

- 12) Dependent variables from other mechanics codes may be supplied to COYOTE for use in coupled simulations. The two **EXternal ... FIEld** commands indicate to COYOTE that an EXODUS II file has been properly named and attached to the code (see Section 4.3). The listed nodal point and element variables are read from the external file and made available to various COYOTE routines, including some of the user supplied subroutines.
- 13) The **SPECIAL OUTput** option allows a number of spatial locations to be defined at which the temperature can be computed, labeled and stored for post-processing. These locations are arbitrary and need not be nodal points. Special points should not be located in bar or shell elements as these types of elements are excluded from the search process. A limit of 50 such points may be defined, the first 20 of which are labeled TP1, TP2, TP3, ... TP20. Data at all of the special points is printed to the output file at the selected output intervals. Labeled data may also be written to the post-processing file for later plotting; special output points are defined as Global data (see Section 3.6).
- 14) The specification of the basic **HEAT FLUX** option causes the  $x$ ,  $y$  and  $z$  components of the heat flux vector to be computed for each time step or iteration. The fluxes are evaluated at a fixed number of (integration) points within each active element. These discontinuous flux values are subsequently extrapolated to the corner nodes of the element and averaged between adjoining elements to produce a continuous nodal point flux field. For higher-order elements, midedge (midside) values of the fluxes are generated by interpolation. The flux components should be written to the post-processing file as no printed output of these quantities is available. Note that the integration points used to define flux locations may not be the same (quadrature) points used to evaluate element matrices and define chemistry points. This is only of concern when a material is reactive and the thermal conductivity is dependent on species concentration. In this case, the flux and quadrature points must be made to coincide if fluxes are to be computed. The data in Table 3.1 shows the integration points used for the flux evaluation.

When *sideset id*'s are specified, some additional processing of the flux data occurs. For each element face (side) in a given sideset, the heat flux normal to the surface is computed and integrated over the area (length) of the element face (side). These

elemental energy contributions are summed to produce the total energy transfer for the surface defined by a sideset; these values are reported in the printed output for each time step or iteration.

The **HEAT FUNCTION** option is only available for two-dimensional geometries and causes a scalar potential to be constructed that is useful in the visualization of energy transport through a domain. The definition of the heat function and its numerical construction is outlined in [3]. Since the heat function requires nodal values of the heat flux, this option automatically invokes the basic HEAT FLUX computation, if it has not been previously specified. Also, as the heat function is constructed incrementally, on an element-by-element basis, the sequencing of elements must be continuous or the process will fail. The heat function should be written to the post-processing file as no printed output of this quantity is available.

- 15) Material removal from a problem may be accomplished by either of two techniques, both of which require some expertise on the part of the user. The **DEATH** option removes individual elements from a specified element block whenever a local element criteria is reached. The removal criteria is based on the comparison of an element centroidal temperature, temperature rate or species concentration with the critical level specified on the DEATH data card. Note that when an element is removed from the problem it cannot be reactivated. The **DELETE MATERIAL** option is more global in nature and removes an entire block of elements at a defined, critical time in the analysis. The **ADD MATERIAL** option is the reverse of the deletion process and activates a block of elements at a specified critical time; the added block is assumed to have a uniform initial temperature and chemical composition as specified by the material property data for the block. The proper use of material addition and deletion requires special attention to the boundary conditions that are applied to the activated and deactivated elements. Examples of the use of this option are given in [4].

## 3.5 Solution Data Block

The control of the solution process for the steady or time-dependent heat conduction problem is accomplished through the solution data block and its associated data cards. A separate solution block is required for each change in solution algorithm or change in any of the associated solution parameters. Input of data for each solution block is terminated by an **END** card. The solution data block is of the following form:

SOLution, solution block number, time dependence

□

.  
 Solution Data  
 .  
 .  
 .  
**ENDsolution**

□

where

solution block number (I) : is the number of the solution block. Solution blocks are executed in numerical order and should be numbered consecutively starting with 1 (see Note 1).

time dependence (C) : indicates if the solution block should be treated as a transient or steady state problem. This parameter should be set to TIME INDEpendent for a steady solution and set to TIME DEPendent for a transient solution sequence.

### Solution Data

The parameters that control the steady or transient solution process within each solution block are specified by a series of data cards. Each solution option data card begins with a keyword and is followed by one or more parameters. Data supplied within a given solution block applies only within that block; parameter settings are not maintained across solution blocks and are reset (or defaulted) by subsequent solution blocks.

The individual solution data cards recognized by COYOTE are listed below along with the various options available for each parameter.

#### **REStart TIME= $t_{restart}$**

$t_{restart}$  (R) <0.0> : specifies the timeplane (time) on the restart file that is to be used for initializing the solution field (see Note 2).

#### **REStart STEP= $step\ number$**

$step\ number$  (I) <last solution on the file> : specifies the timeplane (number) on the restart file that is to be used for initializing the solution field (see Note 2). The default setting selects the last solution available on the restart file.

#### **ITERative METHod=scheme**

scheme (C) <PICard> : specifies the type of iterative method employed for non-linear, time independent problems. For a heat conduction problem without enclosure radiation, the permissible value for this parameter is PICard. Combined enclosure radiation and conduction problems may have parameter settings of PICard or NEWton (see Note 3).

#### **INTEGRATION METHOD=scheme, predictor option**

scheme (C) <TRAPezoid> : specifies the type of time integration method employed for time dependent problems. The permissible values for this parameter are EULER, TRAPezoid and EXPLICIT (see Note 4).

*predictor option* (C) <PREDict> : specifies if the explicit predictor is to be used in conjunction with an implicit integration method. This parameter may be set to PREDict or left blank, if the predictor step is to be employed; a parameter setting of NOPredict eliminates the predictor step (see Note 4).

#### **CAPACITANCE MATRIX=type**

type (C) <CONSistent> : specifies the treatment of the capacitance matrix in time dependent problems. The permissible values for this parameter are CONSistent and LUMPed (see Note 4).

#### **TIME STEP OPTION=type**

type (C) <AUTOstep> : specifies the type of time step control used in time dependent problems. The permissible values of this parameter are FIXed or AUTOstep (see Note 4).

#### **INTEGRATION TOLERANCE=integ tol**

integ tol (R) <  $1.0 \times 10^{-4}$  > : specifies the integration tolerance for the time step control under the AUTOstep option. This parameter is only applicable to the implicit integration methods (see Note 4).

#### **NORM TEMPERATURE= $T_{norm}$**

$T_{norm}$  (R) < maximum T in solution > : specifies an appropriate (maximum) temperature to be used in defining an RMS norm on the temperature change over a time step or iteration. The default temperature scale for the norm is the

maximum temperature found in the solution vector at each step or iteration. If the maximum temperature is not constant for the simulation, this parameter should be specified to ensure a consistent definition of the norm. The definition of the RMS norm is given in [3].

**TIME STEP FACTor= $\beta_{explicit}$**

$\beta_{explicit}$  (R)  $\langle 0.9 \rangle$  : specifies the scale factor to be applied to the maximum stable explicit time step. This parameter is only applicable to the explicit integration method (see Note 4).

**TIME STEP= $\Delta t$**  □

$\Delta t$  (R) : specifies the time step for a time dependent simulation. For the FIXed time step option this time step is used throughout the solution block. When the AUTOstep option is selected, this is the initial time step for the integration process. A method for properly estimating the initial time step is described in Appendix C.

**INITial TIME= $t_{init}$**

$t_{init}$  (R)  $\langle 0.0 \rangle$  : specifies the initial time for a time dependent simulation. Note that this parameter only needs to be specified for the first solution block except in the case of a restart problem, where the initial time is obtained from the restart file.

**FINal TIME= $t_{final}$**  □

$t_{final}$  (R) : specifies the final time for a time dependent simulation (see Note 5).

**NUMber TIME STEP $s$ = $nstep$**  □

$nstep$  (I)  $\langle 1000 \rangle$  : specifies the total number of time steps allowed in a time dependent simulation (see Note 5).

**ABSolute TEMPerature LIMit= $T_{lim}$**

$T_{lim}$  (R)  $\langle 1.0 \times 10^{20} \rangle$  : specifies the allowable upper limit on the temperature field. If the temperature at any node exceeds  $T_{lim}$  then the simulation is terminated (see Note 5).

**MINimum TIME STEP= $\Delta t_{min}$** 

$\Delta t_{min}$  (R) <  $\Delta t_{init} \times 0.0001$  > : specifies the minimum time step allowed during a time dependent simulation.

**MAXimum TIME STEP= $\Delta t_{max}$** 

$\Delta t_{max}$  (R) <  $\Delta t_{init} \times 10000.0$  > (R) : specifies the maximum time step allowed during a time dependent simulation.

**MAXimum TEMPerature STEP= $\Delta T_{max}$** 

$\Delta T_{max}$  (R) : specifies the maximum change in the temperature that is allowed during any time step in a time dependent simulation. This parameter is only valid when the AUTOstep time step option has been selected (see Note 4).

**CHEMistry STEP MULTIplier= $X_{chem}$** 

$X_{chem}$  (R) < 100.0 > : specifies the ratio of the maximum allowed conduction time step to the smallest, current chemistry time step (see Note 6).

**CONVergence TOLerance= $tol$** 

$tol$  (R) <  $1.0 \times 10^{-5}$  > : specifies the temperature convergence tolerance for a nonlinear, steady state problem or the steady state tolerance for a time dependent simulation (see Note 5).

**RELaxation FACTor= $\alpha$** 

$\alpha$  (R) < 0.0 > : specifies the relaxation factor to be used in the iterative solution of a nonlinear, steady state problem. For coupled heat conduction and enclosure radiation problems solved with Newton's method, both the temperatures and the radiative heat fluxes are relaxed.

**MAXimum ITERations= $itermax$** 

$itermax$  (I) < 5 > : specifies the maximum number of iterations allowed in the iterative solution of a nonlinear, steady state problem (see Note 5).

**PRINTed OUTput=option, frequency**

option (C) <SUMmary> : specifies the type of printed output that is writtend to the output file during the solution process. The allowable options are NONE, SUMmary, EXTended and DEBug.

frequency (I) <1> : specifies how often data is printed during the solution process; data is printed every n iterations or time steps where n=frequency.

### RADIation SOLution=method, tolerance, itermax, relax factor

method (C) <GAUSs> : specifies the algorithm to be used for solution of the enclosure radiation problem. When this parameter is set to GAUSs the net radiation problem is solved using a Gauss-Seidel iterative method. A parameter setting of PROGressive selects the progressive refinement algorithm.

tolerance (R) <  $1.0 \times 10^{-3}$  > : specifies the convergence tolerance for the enclosure radiation solution; based on the change in the magnitude of the radiosity.

itermax (I) < 200,  $2 \times$  nsurfaces > : specifies the maximum number of iterations allowed for the enclosue radiation solution. The first default value corresponds to the Gauss-Seidel method and the second default is for the progressive refinement algorithm.

relax factor (R) < 1.2 > : specifies the over-relaxation factor to be used with the Gauss-Seidel solution method.

### VIEWfactor UPDate=option, frequency

option (C) : specifies the type of procedure used to determine when view factors should be updated during the solution process. If this parameter is set to STEP, the view factors are recomputed every n iterations or time steps, where n is equal to the frequency parameter. A parameter setting of TIME indicates that the view factors will be updated at a fixed time interval,  $\Delta t$ , set by the frequency parameter (see Note 7).

frequency (I or R) : specifies the integer number of steps (iterations) between view factor updates or the time interval between recomputation of the view factors.

### MATRix SOLver=method, krylov subspace

method (C) < CG > : specifies the type of method used to solve the linear algebra problem at each iteration or time step. The parameter setting CG specifies that a conjugate gradient method will be used for the matrix problem. Other permissable parameter settings include CGS, for the conjugate gradient squared method, GMRes for the generalized minimum residual method

and QMR for the quasi-minimum residual method. A final parameter setting of DIRect allows access to a direct (non-iterative) matrix solution method (see Note 8).

*krylov subspace* (I)  $\langle 10 \rangle$  : indicates the dimension of the Krylov subspace used with the GMRes iterative solution method.

#### PREConditioner TYPE=option, *polynomial order*

option (C)  $\langle \text{JACobi} \rangle$  : specifies the type of preconditioner to be used in conjunction with the selected iterative matrix solver. The permissible values for this parameter are NONE, JACobi, POLYnomial, ILU and IC (see Note 8).

*polynomial order* (I)  $\langle 1 \rangle$  : specifies the order of the polynomial used in the POLYnomial preconditioner option.

#### L2 NORM= $L_2$

$L_2$  (R)  $\langle 1.0 \times 10^{-6} \rangle$  : specifies the value of the  $L_2$  norm for terminating the preconditioned conjugate gradient solution of the matrix problem during any given iteration or time step (see Note 8).

#### RESidual NORM=resid

resid (R)  $\langle 0.1 \rangle$  : specifies the value of the residual norm for terminating the preconditioned conjugate gradient solution of the matrix problem during any given iteration or time step (see Note 8).

#### MAXimum MATRix ITERations=matrix iter

matrix iter (I)  $\langle \text{NUMNOD} \rangle$  : specifies the maximum number of iterations allowed for the solution of the matrix problem during any given iteration or time step (see Note 8).

#### EPSilon MINimum= $\epsilon_{min}$

$\epsilon_{min}$  (R)  $\langle 0.0001 \rangle$  : specifies a lower bound convergence criteria and chemistry time scale check (see Note 9).

#### EPSilon MAXimum= $\epsilon_{max}$

$\epsilon_{max}$  (R) < 10.0 > : specifies an upper bound convergence criteria and chemistry time scale check (see Note 9).

**MINimum CHEMistry TIMEstep= $\Delta t_{min}^{chem}$**

$\Delta t_{min}^{chem}$  (R) <  $1.0 \times 10^{-15}$  > : specifies the minimum time step allowed during the solution of the chemical rate equations (see Note 9).

**PERcentage ASYMptotics=pctasymp**

pctasymp (R) < 0.0 > : specifies the percentage of the chemical rate equations that will always be solved using asymptotics (see Note 9).

**TOLerance ASYMptotics=tolasymp**

tolasymp (I) < 100.0 > : specifies the asymptotic selection criteria for the chemical rate equations (see Note 9).

**Notes:**

- 1) Each solution block should be constructed such that all the parameters required for that portion of the solution process are defined. The only data that is carried across solution blocks is the last computed temperature field, the current time and a global step number. Any parameters that are defined in one block but not defined in subsequent blocks revert to their default values. Though the input order of solution blocks is arbitrary, execution of the blocks is carried out according to the solution block number, which must form a consecutive sequence. Also, note that changes in the solution algorithms across solution blocks are generally arbitrary with one exception. The use of the Newton method with enclosure radiation requires substantially more memory than the decoupled algorithms that treat conduction and radiation in a sequential solution process. Switching between the coupled and decoupled methods across solution blocks is not permitted.
- 2) Restarts of a solution procedure from a previously computed solution require that a file in an EXODUS II format be attached to COYOTE with a specified file name (see Section 4.3). The particular solution field used to initialize the new solution process is specified by providing either the time or the step (iteration) number for the field located on the external file. Further details on the restart process and restrictions are provided in Section 3.12.

- 3) The weakly nonlinear problems normally encountered in time independent heat conduction simulations are adequately treated with a simple Picard iteration method. However, the inclusion of enclosure radiation in the steady problem significantly increases the nonlinearity of the system and makes convergence more difficult. The selection of Newton's method for the combined radiation and conduction problem usually leads to improved convergence of the iterative process. Note that when the Newton algorithm is selected, the conduction and enclosure radiation equations are solved as a completely combined system. This circumstance leads to large increases in the computer memory requirements for the problem; large three-dimensional geometries with a large number of radiating surfaces may not be solvable with the Newton option. When the Picard method is applied to the enclosure radiation and conduction problem, the equation systems are solved in a decoupled, sequential algorithm. This leads to the ability to accommodate larger problem sizes though very large relaxation factors (*e.g.*,  $\alpha > 0.9$ ) are normally required to achieve convergence. When neither of the above algorithms are suitable a transient approach is a viable alternative.
- 4) All of the time integration methods available in COYOTE may be employed with either a fixed time step or an adaptive time step control. For the implicit integrators (Euler and Trapezoid methods) the adaptive time step is based on an accuracy criteria that is specified by the **INTEGRATION TOLERANCE** parameter. Adaptive time step control for these methods also requires that the *predictor option* be activated; if the predictor is deactivated, the time step option is automatically set to a fixed time step procedure. The adaptive time step control for the explicit integrator is based on stability considerations; the **TIME STEP FACTOR** allows the maximum stable explicit time step to be scaled by the user. Under the adaptive time step control option, a new time step is computed at the end of each step. If the **MAXIMUM TEMPERATURE STEP** parameter has been specified, the computed time step may be reduced to ensure that the limit on the maximum temperature change is respected.

The capacitance matrix may be lumped or left in a consistent form with the implicit integration methods. Capacitance lumping for the explicit integrator is automatically invoked by the code. The use of the lumping option with higher-order (quadratic temperature) elements is not recommended.

- 5) The solution processes in COYOTE may be terminated by any of several criteria. For time-independent, iterative methods, the solution process may be terminated after a certain number of iterations (**MAXIMUM ITERATIONS**), when the change in the solution (norm) between iterations has reached a nominal value (**CONVERGENCE TOLERANCE**) or when the temperature exceeds a specified value (**ABSOLUTE TEMPERATURE LIMIT**). The time integration methods are terminated based on

the total number of time steps taken (**NUMBER TIME STEPS**), the end of the integration interval (**FINAL TIME**), the attainment of a steady temperature state (**CONVergence TOLerance**) or when the temperature exceeds a specified value (**ABSolute TEMPerature LIMit**). These stopping parameters are generally set to values that should not prematurely terminate the analysis, though the (**CONVergence TOLerance**) parameter may need to be reset for problems that evolve slowly at early times. For coupled heat conduction and enclosure radiation, only temperatures are checked for convergence.

- 6) The coupling between the time scales for conduction and chemical reaction is accomplished with the **CHEMistry STEP MULTiplier** parameter. At the conclusion of each time step the newly recommended time step for thermal conduction is compared to the newly computed time step for the chemical kinetic equations. If the time step for conduction is more than  $X_{chem}$  times the chemistry time step, the conduction time step is reduced so that the  $X_{chem}$  ratio is satisfied. If the new conduction time step is less than  $X_{chem}$  times the chemistry time step, then both recommended time steps are accepted and used in the next solution. This procedure allows the chemistry to control the integration process during the rapid reaction phase of a problem while keeping thermal conduction in control during slow reaction intervals.
- 7) When enclosure radiation is part of the conduction problem, the view factors for the geometry are normally computed only once under the assumption that the problem geometry is not variable. In cases where (Lagrangian) material motion is prescribed, the view factors should be periodically updated to reflect the evolving geometric configuration. A similar updating is appropriate when material is added or deleted from the simulation. The **VIEWfactor UPDate** input allows updating to occur based on the number of time steps between updates or the time interval between updates. Note that the default for this parameter corresponds to the case where updating of the view factors will occur for every time step - a process that can be very expensive.
- 8) The primary matrix solution methods available in COYOTE consist of iterative methods based on the Krylov or conjugate direction algorithms. To use these methods effectively also requires the selection of an acceptable preconditioning technique. The solution package called by COYOTE was developed by Schunk and Shadid [8] and contains a wide spectrum of symmetric and unsymmetric matrix methods with a variety of preconditioners. For most standard conduction problems the selection of a conjugate gradient solver (CG) with JACobi preconditioning is adequate. Unsymmetric systems arising from a combined enclosure radiation, heat conduction problem can be solved with a generalized minimum residual method (GMRes) and

the ILU preconditioner. This choice is also appropriate for the unsymmetric system developed in the advection-diffusion form of the energy equation. Further details on these solvers and their overall performance on different types of problems can be found in [8]. Though a direct solution method is also available in this package, it is not recommended except for modest-sized, two-dimensional simulations. The matrix solver parameters **L2 NORM** and **RESidual NORM** do not usually require adjustment; the parameter **MAXimum MATRIX ITERations** is set for medium-sized problems and may need to be set substantially higher for large simulations. The precise definition and function of these parameters is given in [8].

- 9) The adjustment of the parameters that control the operation of the stiff, ordinary equation solver, CHEMEQ, requires a very experienced user. The default parameter settings are those recommended in the original CHEMEQ document [9] and have proven suitable for a large class of reaction equations. The definitions for the listed control parameters are provided in [9].

## 3.6 Post-processing Data Block

The control of the output of data to the post-processing file is accomplished through the post-processing data block and its associated data cards. This data block is optional. Input of post-processing data is terminated by an **END** card. The post-processing data block is of the following form:

```
POST
.
.
Post-processing Data
.
.
.
ENDpost
```

### Post-processing Data

The output of various solution variables to the post-processing EXODUS file is managed through a series of data cards within the post-processing data block. Each data card begins with a keyword and is followed by one or more parameters that define the various options. The individual post-processing data cards recognized by COYOTE are listed below.

**NODal DATA=name1, name2,....**

name1, name2,.. (C) : specify the names of the nodal point variables that are to be written to the post-processing output file. The following variable names are allowed: TEMPerature, TDOT (time rate of change of temperature), QXFLux, QYFLux, QZFLux (components of the heat flux), HFUNction (two-dimensional heat flow function) and DISPLX, DISPLY, DISPLZ (components of the material displacement).

**ELEMent DATA=name1, name2,....**

name1, name2,.. (C) : specify the names of the element variables that are to be written to the post-processing output file. The allowed variable name is currently limited to STATUS (variable that specifies the element status for the birth/death option) (see Note 1).

**CHEMistry DATA=name1, name2,....**

name1, name2,.. (C) : specify the names of the chemistry variables that are to be written to the post-processing output file. The allowed variable names are those specified as **SPECies** under each material definition and GASFrac for gas fractions at element centroids.

**GLOBAL DATA=name1, name2,....**

name1, name2,.. (C) : specify the names of the global variables that are to be written to the post-processing output file. The following variable names are recognized: TIMESTEP (the integration time step), CGITER (the number of iterations taken by the conjugate gradient solver), TP1, TP2, ... TP $n$ , ... TP20 (temperature at special output points) and QN1, QN2, ... QN $n$ , ... QN20 (integrated heat fluxes on element side sets). The  $n$  in the TP name refers to the special point number as given by the sequence of points listed on the special point card (see Section 3.4). The  $n$  in the QN name refers to the sequence number of the side sets specified on the **HEAT FLUX** option (see Section 3.4).

**OUTput FREQuency=nsteps**

nsteps (I) < 1 > : specifies the frequency at which data is written to the post-processing file. All requested data is written every nsteps time steps or iteration cycles.

**OUTput TIMES**=  $t_1, t_2, \dots, t_n$ 

$t_1, t_2, \dots, t_n$  (R) : specify the set times at which data is written to the post-processing file. All requested data is written whenever the simulation time exceeds one of the specified output times. A maximum of fifty specific times may be specified; the times must be ordered in a strictly increasing sequence (see Note 2).

**OUTput TIME STEP**=  $\Delta t_{out}$ 

$\Delta t_{out}$  (R) : specifies the time interval at which data is written to the post-processing file. All requested data is written whenever the simulation time exceeds one of the times given by the sequence  $t_{out} = t_{init} + n\Delta t_{out}$  where  $n = 1, 2, 3, \dots$  (see Note 2).

**Notes:**

- 1) The element variable STATUS indicates if an element within an element block is currently active or inactive. The status variable is set to zero for an active element and is unity for a deactivated element. The setting of this variable is controlled by the material addition and deletion options; the element activation status may also be controlled from an external solution file through the **EXternal ELEMENT FIELD** input in the Problem Definition Data Block (see Section 3.4).
- 2) When requesting output at specified times or time intervals it is important to coordinate these requested times with the time step being used in the solution of the finite element equations. In general, output time intervals must be larger than solution time steps. If successive output times fall within a single computational time step, the test for output of a solution will fail and no further output will be generated. The output processor always checks to see if the next requested output time has been passed by the solution time; once the output time falls behind the solution time the output check will always fail.

## 3.7 Time Function Data Block

The specification of time functions for use in boundary conditions or material models is accomplished with the time function data block and its associated data cards. A separate time function block is required for each time function used in a COYOTE simulation.

Input to each time function is terminated by an **END** card. This data block is optional. The time function data block is of the following form:

**TIME FUNction=function id**

$t_1, f(t_1)$

$t_2, f(t_2)$

$t_3, f(t_3)$

.

.

.

$t_n, f(t_n)$

.

.

**ENDtime**

where

function id (I) : is a unique, integer identifier for the time function.

$t_n, f(t_n)$  (R) ; is a list of times and function values used to define the time function (see Note 1).

#### Notes:

- 1) The sequence of times for each function must form an increasing sequence. Also, no extrapolation or continuation of data beyond the last data point is allowed by COYOTE. If the solution time exceeds the time for the function, the code will terminate execution with an error message. The function data is used with a linear interpolation scheme to provide variable values at intermediate times.

## 3.8 Variable Function Data Block

The specification of variable functions for use in boundary conditions or material models is accomplished with the variable function data block and its associated data cards. A separate variable function block is required for each variable function used in a COYOTE

simulation. Input to each variable function is terminated by an **END** card. This data block is optional. The variable function data block is of the following form:

```

VARiFUNCTION=function id
  T1, f(T1)
  T2, f(T2)
  T3, f(T3)
  .
  .
  .
  Tn, f(Tn)
  .
  .
ENDVARIABLE

```

where

function id (I) : is a unique, integer identifier for the variable function.

$T_n, f(T_n)$  (R) ; is a list of variable and function values used to define the variable function.

In the current version of COYOTE the temperature is the only variable allowed in the definition of a variable function (see Note 1).

#### Notes:

- 1) The sequence of temperatures for each function must form an increasing sequence. Also, no extrapolation or continuation of data beyond the last data point is allowed by COYOTE. If the current temperature falls above or below the temperature range of the function the code will terminate execution with an error message. The function data is used with a linear interpolation scheme to provide variable values at intermediate temperatures.

## 3.9 User Constants Data Block

The specification of real or integer constants that may be useful within user supplied boundary condition or material property subroutines is controlled by the user constant

data block and its associated data cards. The data supplied in this block is recorded in two vectors (real and integer) and passed to all user supplied subroutines through their parameter lists. Subroutine evaluations that make use of these constants may therefore be altered through the input file, without modification and recompilation of the subroutines. This data block is optional. The user constants data block is of the following form:

**USER CONstants**

·  
·

**USER REAL=n, rvalue**

·  
·  
·

**USER INTeger=m, ivalue**

·  
·

**ENDconstant**

where

$n, m$  (I) : are unique, integer indices that locate each constant within its appropriate vector (see Note 1).

$rvalue, ivalue$  (R and I) ; are the real and integer values of the constants

**Notes:**

- 1) Real and integer user defined constants are stored in two vectors called RCONST and ICONST, respectively. The indices specified on the data cards allow the user to precisely determine where in each vector the constant will be stored. Note that the RCONST and ICONST vectors are dynamically allocated and the length of each vector corresponds to the largest  $n$  and  $m$  values encountered in the input.

## 3.10 Termination Data

The termination of data within a particular data block is signaled with an **END** command and has been described in the previous chapters. The termination of all data input to

COYOTE is signaled by use of either the **EXIT** or **STOP** data cards. This command must be the last data entry in the input file to ensure proper program termination.

## 3.11 User Supplied Subroutines

There are several instances which require the user to supply FORTRAN coded subroutines to COYOTE. The occurrence of variable material properties or source terms, the definition of some types of material motion and the specification of certain types of variable boundary conditions result in the need for one or more user supplied subroutines. The required formats for these subroutines will be described in the following chapters. Skeleton versions of all user subroutines are supplied with the COYOTE code and may be used as templates for specific applications.

### 3.11.1 Material Properties

Variations in thermophysical properties due to temperature variations can normally be accommodated with function type data. However, when property variations are complex or depend on variables other than temperature, then a subroutine evaluation may be more appropriate. When any of the material property data cards **DENSity**, **SPECific HEAT**, **CONDUCTivity**, **TENSor ROTation** or **EMISSivity** are set to USER for any material in the problem (see Section 3.3), a user subroutine describing that material property must be supplied to COYOTE. Each user subroutine is used to evaluate the appropriate material property for *all* materials labeled as USER on the material data card; the user assigned material name allows specific materials to be identified within each subroutine.

### Density

The density for each material is evaluated with a subroutine of the following form:

```

SUBROUTINE USRDEN (RHO, TEMP, SPEC, XIP, YIP, ZIP, NAME, NUMIPT,
*                MXSPEC, NSPEC, TIME, KSTEP, RCONST, ICONST)
C
C *****
C
CHARACTER*20 NAME
```

```

C
  DIMENSION RHO(*), TEMP(*), SPEC(MXSPEC,*)
  DIMENSION XIP(*), YIP(*), ZIP(*)
  DIMENSION RCONST(*), ICONST(*)
C
C
C *****
C
C   FORTRAN coding to evaluate the density at the element
C   integration points for each material
C
  RETURN
  END

```

where the variables in the subroutine parameter list are

RHO(NUMIPT) : an array containing integration point values of the material density (output).

TEMP(NUMIPT) : an array containing integration point values of the temperature (input).

SPEC(MXSPEC,NUMIPT) : an array containing integration point values of the chemical species concentrations for this material (input).

XIP(NUMIPT), YIP(NUMIPT), ZIP(NUMIPT) : arrays containing integration point values of the x, y and z coordinates (input).

NAME : a character variable specifying the material name (input).

NUMIPT : an integer specifying the number of integration points in the element (input).

MXSPEC : an integer specifying the maximum number of chemical species in any material (input).

NSPEC : an integer specifying the number of chemical species in the current material (input).

TIME : the value of the current time (input).

KSTEP : an integer specifying the number of the current iteration cycle or time step (input).

RCONST(\*) : an array containing (real) user defined constants

ICONST(\*) : an array containing (integer) user defined constants

## Specific Heat

The specific heat for each material is evaluated with a subroutine of the following form:

```

      SUBROUTINE USRCP (CP, TEMP, SPEC, XIP, YIP, ZIP, NAME, NUMIPT,
*                    MXSPEC, NSPEC, TIME, KSTEP, RCONST, ICONST)
C
C *****
C
C CHARACTER*20 NAME
C
C DIMENSION CP(*), TEMP(*), SPEC(MXSPEC,*)
C DIMENSION XIP(*), YIP(*), ZIP(*)
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C FORTRAN coding to evaluate the specific heat at the element
C integration points for each material
C
C RETURN
C END

```

where the variables in the subroutine parameter list are

CP(NUMIPT) : an array containing integration point values of the material specific heat (output).

TEMP(NUMIPT) : an array containing integration point values of the temperature (input).

SPEC(MXSPEC,NUMIPT) : an array containing integration point values of the chemical species concentrations for this material (input).

XIP(NUMIPT), YIP(NUMIPT), ZIP(NUMIPT) : arrays containing integration point values of the x, y and z coordinates (input).

NAME : a character variable specifying the material name (input).

NUMIPT : an integer specifying the number of integration points in the element (input).

MXSPEC : an integer specifying the maximum number of chemical species in any material (input).

NSPEC : an integer specifying the number of chemical species in the current material (input).

TIME : the value of the current time (input).

KSTEP : an integer specifying the number of the current iteration cycle or time step (input).

RCONST(\*) : an array containing (real) user defined constants

ICONST(\*) : an array containing (integer) user defined constants

## Conductivity

The thermal conductivity for each material is evaluated with a subroutine of the following form:

```

      SUBROUTINE USRCON (COND11, COND22, COND33, TEMP, SPEC, XIP, YIP,
*                      ZIP, NAME, NUMIPT, MXSPEC, NSPEC, TIME, KSTEP,
*                      RCONST, ICONST)
C
C *****
C
C CHARACTER*20 NAME
C
C DIMENSION COND11(*), COND22(*), COND33(*)
C DIMENSION TEMP(*), SPEC(MXSPEC,*)
C DIMENSION XIP(*), YIP(*), ZIP(*)
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C FORTRAN coding to evaluate the thermal conductivity at the element
C integration points for each material
C
C RETURN
C END

```

where the variables in the subroutine parameter list are

COND11(NUMIPT), COND22(NUMIPT), COND33(NUMIPT) : arrays containing integration point values of the principle thermal conductivities  $k_{11}$ ,  $k_{22}$  and  $k_{33}$  for the material (output). For isotropic materials set COND22=COND33=COND11; for two-dimensional problems COND33 need not be evaluated.

TEMP(NUMIPT) : an array containing integration point values of the temperature (input).

SPEC(MXSPEC,NUMIPT) : an array containing integration point values of the chemical species concentrations for this material (input).

XIP(NUMIPT), YIP(NUMIPT), ZIP(NUMIPT) : arrays containing integration point values of the x, y and z coordinates (input).

NAME : a character variable specifying the material name (input).

NUMIPT : an integer specifying the number of integration points in the element (input).

MXSPEC : an integer specifying the maximum number of chemical species in any material (input).

NSPEC : an integer specifying the number of chemical species in the current material (input).

TIME : the value of the current time (input).

KSTEP : an integer specifying the number of the current iteration cycle or time step (input).

RCONST(\*) : an array containing (real) user defined constants

ICONST(\*) : an array containing (integer) user defined constants

## Tensor Rotation

The local orientation of the conductivity tensor for each anisotropic material is evaluated with a subroutine of the following form:

```

SUBROUTINE USRROT (XXHAT, YXHAT, ZXHAT, XYHAT, YYHAT, ZYHAT, XIP,
*                YIP, ZIP, NAME, NUMIPT, TIME, KSTEP, RCONST,
*                ICONST)
C
C *****
C
C CHARACTER*20 NAME
C
C DIMENSION XXHAT(*), YXHAT(*), ZXHAT(*)
C DIMENSION XYHAT(*), YYHAT(*), ZYHAT(*)
C DIMENSION XIP(*), YIP(*), ZIP(*)
C DIMENSION RCONST(*), ICONST(*)
C

```

```

C      *****
C
C      FORTRAN coding to evaluate the orientation of the principle material
C      axes with respect to the global coordinate frame at the element
C      integration points for each material
C
      RETURN
      END

```

where the variables in the subroutine parameter list are

**XXHAT(NUMIPT)**, **YXHAT(NUMIPT)**, **ZXHAT(NUMIPT)** : arrays containing integration point values of the  $x, y, z$  projection of a vector in the positive  $\hat{x}$  direction. The  $x, y, z$  axes denote the global reference frame and the  $\hat{x}, \hat{y}, \hat{z}$  axes are the principle material axes. For two-dimensional problems only the **XXHAT** and **YXHAT** variables need to be evaluated. See Section 3.4 for a further discussion of the reference frames.

**XYHAT(NUMIPT)**, **YYHAT(NUMIPT)**, **ZYHAT(NUMIPT)** : arrays containing integration point values of the  $x, y, z$  projection of a vector in the positive  $\hat{y}$  direction. The  $x, y, z$  axes denote the global reference frame and the  $\hat{x}, \hat{y}, \hat{z}$  axes are the principle material axes. For two-dimensional problems these variables need not be evaluated. See Section 3.4 for a further discussion of the reference frames.

**XIP(NUMIPT)**, **YIP(NUMIPT)**, **ZIP(NUMIPT)** : arrays containing integration point values of the  $x, y$  and  $z$  coordinates (input).

**NAME** : a character variable specifying the material name (input).

**NUMIPT** : an integer specifying the number of integration points in the element (input).

**TIME** : the value of the current time (input).

**KSTEP** : an integer specifying the number of the current iteration cycle or time step (input).

**RCONST(\*)** : an array containing (real) user defined constants

**ICONST(\*)** : an array containing (integer) user defined constants

## Enthalpy

Change of phase problems may require the enthalpy of the material to be specified as a function of the temperature and/or chemical composition. The enthalpy is evaluated for each material with a subroutine of the following form:

```

SUBROUTINE USRENT (ENT, TEMP, SPEC, XIP, YIP, ZIP, NAME, NUMIPT,
*                MXSPEC, NSPEC, TIME, KSTEP, RCONST, ICONST)
C
C *****
C
C CHARACTER*20 NAME
C
C DIMENSION ENT(*), TEMPS(*), SPEC(MXSPEC,*)
C DIMENSION XIP(*), YIP(*), ZIP(*)
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C FORTRAN coding to evaluate the enthalpy at the element
C integration points for each material
C
C RETURN
C END

```

where the variables in the subroutine parameter list are

ENT(NUMIPT) : an array containing integration point values of the material enthalpy (output).

TEMP(NUMIPT) : an array containing integration point values of the temperature (input).

SPEC(MXSPEC,NUMIPT) : an array containing integration point values of the chemical species concentrations for this material (input).

XIP(NUMIPT), YIP(NUMIPT), ZIP(NUMIPT) : arrays containing integration point values of the x, y and z coordinates (input).

NAME : a character variable specifying the material name (input).

NUMIPT : an integer specifying the number of integration points in the element (input).

MXSPEC : an integer specifying the maximum number of chemical species in any material (input).

NSPEC : an integer specifying the number of chemical species in the current material (input).

TIME : the value of the current time (input).

KSTEP : an integer specifying the number of the current iteration cycle or time step (input).

RCNST(\*) : an array containing (real) user defined constants

ICONST(\*) : an array containing (integer) user defined constants

## Emissivity

Though the surface emissivity is a material property, it occurs in the thermal diffusion problem only in conjunction with boundary conditions. Unlike the other material properties that are evaluated at element integration points, the emissivity is evaluated at nodal points on the surface or edge of an element. The surface emissivity for each material is evaluated with a subroutine of the following form:

```

      SUBROUTINE USREMS (EMISS, TEMPS, XS, YS, ZS, NAME, NNODES, TIME,
*                   KSTEP, RCNST, ICONST)
C
C   *****
C
C   CHARACTER*20 NAME
C
C   DIMENSION EMISS(*), TEMPS(*)
C   DIMENSION XS(*), YS(*), ZS(*)
C   DIMENSION RCNST(*), ICONST(*)
C
C   *****
C
C   FORTRAN coding to evaluate the emissivity at the element surface
C   or edge nodes for each material
C
      RETURN
      END

```

where the variables in the subroutine parameter list are

EMISS(NNODES) : an array containing surface/edge node values of the material surface emissivity (output).

TEMPS(NNODES) : an array containing surface/edge node values of the temperature (input).

XS(NNODES), YS(NNODES), ZS(NNODES) : arrays containing surface/edge node values of the x, y and z coordinates (input).

NAME : a character variable specifying the material name (input).

NNODES : an integer specifying the number of nodes on the element surface/edge (input).

TIME : the value of the current time (input).

KSTEP : an integer specifying the number of the current iteration cycle or time step (input).

RCONST(\*) : an array containing (real) user defined constants

ICONST(\*) : an array containing (integer) user defined constants

## Chemistry

The use of reactive materials may require that the chemical reaction rates be specified through a user subroutine. The selection of this option is accomplished by setting the USER parameter on the **REACTive MIXture** card (Section 3.3) and supplying a subroutine of the following form:

```

      SUBROUTINE USRRR (RRATES,AK,TEMP,SPEC,NAME,NUMIPT,MXSPEC,MXREAC,
*                   NSPEC,NREAC,STERIC,PREX,AENRGY,AMUSP,RCONST,ICONST)
C
C *****
C
C CHARACTER*20 NAME
C
C DIMENSION RRATES(MXREAC,*), AK(MXREAC,*)
C DIMENSION TEMP(*), SPEC(MXSPEC,*)
C DIMENSION STERIC(*), PREX(*)
C DIMENSION AENRGY(*), AMUSP(MXSPEC,*)
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C USER SUPPLIED FORTRAN CODE TO EVALUATE THE REACTION RATES
C FOR ALL REACTIONS IN A MATERIAL
C
C RETURN
C END

```

where the variables in the subroutine parameter list are

RRATES(NUMIPT) : an array containing integration point values of the chemical reaction rates (output).

AK(MXREAC,NUMIPT) : an array containing integration point values of the kinetic coefficients for the current material (output).

TEMP(NUMIPT) : an array containing integration point values of the temperature (input).

SPEC(MXSPEC,NUMIPT) : an array containing integration point values of the chemical species concentrations for this material (input).

NAME : a character variable specifying the material name (input).

NUMIPT : an integer specifying the number of integration points in the element (input).

MXSPEC : an integer specifying the maximum number of chemical species in any material (input).

NSPEC : an integer specifying the number of chemical species in the current material (input).

NREAC : an integer specifying the number of chemical reactions in the current material (input).

STERIC(NUMIPT) : an array specifying the steric coefficients for the current material (input).

PREX(NUMIPT) : an array specifying the pre-exponential factors for the current material (input).

AENRGY(NUMIPT) : an array specifying the activation energy for the current material (input).

AMUSP : an array specifying the activation energy for the current material (input).

RCNST(\*) : an array containing (real) user defined constants

ICONST(\*) : an array containing (integer) user defined constants

### 3.11.2 Volumetric Sources

When the volumetric heating parameter, **VOLume HEATing** (see Section 3.3) for one or more materials is set to **USER**, then **COYOTE** expects a source subroutine (**USRVHS**) to be supplied by the user. This subroutine must have the following form:

```

SUBROUTINE USRVHS (VOLHT, TEMP, XIP, YIP, ZIP, NAME, NUMIPT,
*                TIME, KSTEP, RCONST, ICONST)
C
C *****
C
C CHARACTER*20 NAME
C
C DIMENSION VOLHT(*), TEMP(*)
C DIMENSION XIP(*), YIP(*), ZIP(*)
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C FORTRAN coding to evaluate the volumetric heat source at the
C element integration points for each material
C
C RETURN
C END

```

where the variables in the subroutine parameter list are

VOLHT(NUMIPT) : an array containing integration point values of the volumetric source (output).

TEMP(NUMIPT) : an array containing integration point values of the temperature (input).

XIP(NUMIPT), YIP(NUMIPT), ZIP(NUMIPT) : arrays containing integration point values of the x, y and z coordinates (input).

NAME : a character variable specifying the material name (input).

NUMIPT : an integer specifying the number of integration points in the element (input).

TIME : the value of the current time (input).

KSTEP : an integer specifying the number of the current iteration cycle or time step (input).

RCONST(\*) : an array containing (real) user defined constants

ICONST(\*) : an array containing (integer) user defined constants

### 3.11.3 Material Velocity

Simple velocity fields for one or more blocks of elements can usually be supplied directly to COYOTE through the **VELocity** data card that is defined in Section 3.4. More complex velocity fields, for either Eulerian or Lagrangian descriptions, require the use of a user supplied subroutine. This option is invoked by setting the velocity specification parameter on the **VELocity** data to USER and supplying COYOTE with a subroutine of the following form:

```

      SUBROUTINE USRVEL (IDBLK, NNODES, XNODE, YNODE, ZNODE, UVEL, VVEL,
*                      WVEL, TIME, RCONST, ICONST)
C
C *****
C
      DIMENSION XNODE(*), YNODE(*), ZNODE(*)
      DIMENSION UVEL(*), VVEL(*), WVEL(*)
      DIMENSION RCONST(*), ICONST(*)
C
C *****
C
      FORTRAN coding to evaluate the components of the velocity vector
      for each node in an element
C
      RETURN
      END

```

where the variables in the subroutine parameter list are

IDBLK : an integer specifying the id for the current element block (input).

NNODES : an integer specifying the number of nodes in the element (input).

XNODE(NNODES), YNODE(NNODES), ZNODE(NNODES) : arrays containing node values of the x, y and z coordinates (input).

UVEL(NNODES), VVEL(NNODES), WVEL(NNODES) : arrays containing node values of the x, y and z velocity components (output).

TIME : the value of the current time (input).

RCONST(\*) : an array containing (real) user defined constants

ICONST(\*) : an array containing (integer) user defined constants

### 3.11.4 Boundary Conditions

Boundary conditions that vary as a simple function of time or temperature can generally be specified with function data. Complex boundary conditions may require the use of subroutines that describe the appropriate functional behavior of the boundary values or required coefficients. These subroutines may also be used to modify boundary conditions as a function of the iteration cycle and thereby improve convergence of the solution process. When any of the boundary condition data cards include the USER specification, a user subroutine of the appropriate type must be supplied to COYOTE. The correspondence between a particular boundary condition and its variation is established through the node set or side set parameter, which appears on the **BCtype** data card (Section 3.4). All boundary conditions of the same type are evaluated via a single subroutine; the node set or side set identification number allows specific boundary conditions to be identified within the subroutine.

#### Temperature

A specified temperature (bcname=TEMPERature) for a group of nodal points is evaluated by setting the bc specification parameter (Section 3.4) equal to USER and supplying a subroutine of the following form:

```

      SUBROUTINE USRT (TEMPBC, XNODE, YNODE, ZNODE, IDNSET, TIME,
*                   KSTEP, RCONST, ICONST)
C
C *****
C
C   DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C   FORTRAN coding to evaluate the temperature at a node
C
      RETURN
      END

```

where the variables in the subroutine parameter list are

TEMPBC : the temperature at the node (output).

XNODE, YNODE, ZNODE : nodal point values of the x, y and z coordinates (input).

IDNSET : an integer specifying the current node set id (input).

TIME : the value of the current time (input).

KSTEP : an integer specifying the number of the current iteration cycle or time step (input).

RCONST(\*) : an array containing (real) user defined constants

ICONST(\*) : an array containing (integer) user defined constants

## Heat Source

A specified heat source (bcname=HEAT SOURCE) for a group of nodal points is evaluated by setting the bc specification parameter (Section 3.4) equal to USER and supplying a subroutine of the following form:

```

        SUBROUTINE USRQ (QBC, TNODE, XNODE, YNODE, ZNODE, IDNSET, TIME,
*
          KSTEP, RCONST, ICONST)
C
C
C *****
C
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C FORTRAN coding to evaluate the heat source at a node
C
        RETURN
        END

```

where the variables in the subroutine parameter list are

QBC : the heat source at the node (output).

TNODE : the temperature at the node (input).

XNODE, YNODE, ZNODE : nodal point values of the x, y and z coordinates (input).

IDNSET : an integer specifying the current node set id (input).

TIME : the value of the current time (input).

KSTEP : an integer specifying the number of the current iteration cycle or time step (input).

RCONST(\*) : an array containing (real) user defined constants

ICONST(\*) : an array containing (integer) user defined constants

## Heat Flux

An applied heat flux (bcname=HEAT FLUX) for an element edge or element surface is evaluated by setting the bc specification parameter (Section 3.4) equal to USER and supplying a subroutine of the following form:

```

      SUBROUTINE USRFLX (FLUX, TEMPS, XS, YS, ZS, NNODES, IDSSET, TIME,
*                      KSTEP, RCONST, ICONST)
C
C *****
C
      DIMENSION FLUX(*), TEMPS(*)
      DIMENSION XS(*), YS(*), ZS(*)
      DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C   FORTRAN coding to evaluate the applied heat flux at the element
C   edge or surface nodes for each side set
C
      RETURN
      END

```

where the variables in the subroutine parameter list are

FLUX(NNODES) : an array containing nodal point values of the applied heat flux (output).

TEMPS(NNODES) : an array containing nodal point values of the temperature (input).

XS(NNODES), YS(NNODES), ZS(NNODES) : arrays containing nodal point values of the x, y and z coordinates (input).

NNODES : an integer specifying the number of the nodes on the element edge or surface (input).

IDSSET : an integer specifying the current side set id (input).

TIME : the value of the current time (input).

KSTEP : an integer specifying the number of the current iteration cycle or time step (input).

RCONST(\*) : an array containing (real) user defined constants

ICONST(\*) : an array containing (integer) user defined constants

## Convective Heat Flux

The use of convection boundary condition (bcname=CONVection) may require that the variation of the heat transfer coefficient and/or the reference temperature be specified through a user subroutine. The heat transfer coefficient for an element edge or surface is evaluated by setting COEFFicient=USER (Section 3.4) and supplying a subroutine of the following form:

```

      SUBROUTINE USRHTC (HCOEF, TEMPS, TREF, XS, YS, ZS, NNODES, IDSSET,
*                      TIME, KSTEP, RCONST, ICONST)
C
C *****
C
C   DIMENSION HCOEF(*), TEMPS(*), TREF(*)
C   DIMENSION XS(*), YS(*), ZS(*)
C   DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C   FORTRAN coding to evaluate the heat transfer coefficient at the
C   element edge or surface nodes for each side set
C
      RETURN
      END

```

where the variables in the subroutine parameter list are

HCOEF(NNODES) : an array containing nodal point values of the heat transfer coefficient (output).

TEMPS(NNODES) : an array containing nodal point values of the temperature (input).

TREF(NNODES) : an array containing nodal point values of the reference temperature (input).

`XS(NNODES)`, `YS(NNODES)`, `ZS(NNODES)` : arrays containing nodal point values of the x, y and z coordinates (input).

`NNODES` : an integer specifying the number of the nodes on the element edge or surface (input).

`IDSSET` : an integer specifying the current side set id (input).

`TIME` : the value of the current time (input).

`KSTEP` : an integer specifying the number of the current iteration cycle or time step (input).

`RCONST(*)` : an array containing (real) user defined constants

`ICONST(*)` : an array containing (integer) user defined constants

The reference temperature for convection on an element edge or surface is evaluated by setting `TREF=USER` (Section 3.4) and supplying a subroutine of the following form:

```

      SUBROUTINE USRTRC (TREF, TEMPS, XS, YS, ZS, NNODES, IDSSET, TIME,
*                      KSTEP, RCONST, ICONST)
C
C *****
C
      DIMENSION TREF(*), TEMPS(*)
      DIMENSION XS(*), YS(*), ZS(*)
      DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C   FORTRAN coding to evaluate the reference temperature for
C   convection at the element edge or surface nodes for each side set
C
      RETURN
      END

```

where the variables in the subroutine parameter list are

`TREF(NNODES)` : an array containing nodal point values of the reference temperature (output).

`TEMPS(NNODES)` : an array containing nodal point values of the temperature (input).

`XS(NNODES)`, `YS(NNODES)`, `ZS(NNODES)` : arrays containing nodal point values of the x, y and z coordinates (input).

`NNODES` : an integer specifying the number of the nodes on the element edge or surface (input).

`IDSSET` : an integer specifying the current side set id (input).

`TIME` : the value of the current time (input).

`KSTEP` : an integer specifying the number of the current iteration cycle or time step (input).

`RCONST(*)` : an array containing (real) user defined constants

`ICONST(*)` : an array containing (integer) user defined constants

## Radiative Heat Flux

The use of radiation boundary condition (`bcname=RADIATION`) may require that the variation of the radiation form factor and/or the reference temperature be specified through a user subroutine. The form factor for an element edge or surface is evaluated by setting `COEFFICIENT=USER` (Section 3.4) and supplying a subroutine of the following form:

```

      SUBROUTINE USRFF (FORMF, TEMPS, EMISS, XS, YS, ZS, NNODES, IDSSET,
*                   TIME, KSTEP, RCONST, ICONST)
C
C *****
C
      DIMENSION FORMF(*), TEMPS(*), EMISS(*)
      DIMENSION XS(*), YS(*), ZS(*)
      DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C   FORTRAN coding to evaluate the radiation form factor at the
C   element edge or surface nodes for each side set
C
      RETURN
      END

```

where the variables in the subroutine parameter list are

- FORMF(NNODES) : an array containing nodal point values of the radiation form factor (output).
- TEMPS(NNODES) : an array containing nodal point values of the temperature (input).
- EMISS(NNODES) : an array containing nodal point values of the surface emissivity (input).  
These values are supplied only if emissivity data has been supplied with the material property specification.
- XS(NNODES), YS(NNODES), ZS(NNODES) : arrays containing nodal point values of the x, y and z coordinates (input).
- NNODES : an integer specifying the number of the nodes on the element edge or surface (input).
- IDSSET : an integer specifying the current side set id (input).
- TIME : the value of the current time (input).
- KSTEP : an integer specifying the number of the current iteration cycle or time step (input).
- RCONST(\*) : an array containing (real) user defined constants
- ICONST(\*) : an array containing (integer) user defined constants

The reference temperature for radiation on an element edge or surface is evaluated by setting TREF=USER (Section 3.4) and supplying a subroutine of the following form:

```

SUBROUTINE USRTRR (TREF, TEMPS, XS, YS, ZS, NNODES, IDSSET, TIME,
*                 KSTEP, RCONST, ICONST)
C
C *****
C
C DIMENSION TREF(*), TEMPS(*)
C DIMENSION XS(*), YS(*), ZS(*)
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C FORTRAN coding to evaluate the reference temperature for
C radiation at the element edge or surface nodes for each side set
C
C RETURN
C END

```

where the variables in the subroutine parameter list are

TREF(NNODES) : an array containing nodal point values of the reference temperature (output).

TEMPS(NNODES) : an array containing nodal point values of the temperature (input).

XS(NNODES), YS(NNODES), ZS(NNODES) : arrays containing nodal point values of the x, y and z coordinates (input).

NNODES : an integer specifying the number of the nodes on the element edge or surface (input).

IDSSET : an integer specifying the current side set id (input).

TIME : the value of the current time (input).

KSTEP : an integer specifying the number of the current iteration cycle or time step (input).

RCONST(\*) : an array containing (real) user defined constants

ICONST(\*) : an array containing (integer) user defined constants

## Contact or Gap Heat Transfer

The use of a contact boundary condition (BCtype=CONtact) may require that the variation of the effective heat transfer coefficient be specified through a user subroutine. The selection of this option is accomplished by setting the parameter COEFfient=USER (Section 3.4) and supplying a subroutine of the following form:

```

      SUBROUTINE USRHTG (HCOEF, TEMPM, TEMPS, XS, YS, ZS, GAPS, NNODES,
*                      IDSSET, TIME, KSTEP, RCONST, ICONST)
C
C *****
C
      DIMENSION HCOEF(*), TEMPM(*), TEMPS(*)
      DIMENSION XS(*), YS(*), ZS(*) ,GAPS(*)
      DIMENSION RCONST(*), ICONST(*)
C
C *****
C
      FORTRAN coding to evaluate the effective heat transfer coefficient

```

```
C      at the element edge or surface nodes for each contact surface
C
      RETURN
      END
```

where the variables in the subroutine parameter list are

HCOEF(NNODES) : an array containing nodal point values of the effective heat transfer coefficient (output).

TEMPM(NNODES) : an array containing nodal point values of the temperature for the master surface (input).

TEMPS(NNODES) : an array containing nodal point values of the temperature for the slave surface (input).

XS(NNODES), YS(NNODES), ZS(NNODES) : arrays containing nodal point values of the x, y and z coordinates (input).

GAPS(NNODES) : an array containing nodal point values of the separation distance (gap) between the master and slave surfaces (input).

NNODES : an integer specifying the number of the nodes on the element edge or surface (input).

IDSSET : an integer specifying the current side set id (input).

TIME : the value of the current time (input).

KSTEP : an integer specifying the number of the current iteration cycle or time step (input).

RCNST(\*) : an array containing (real) user defined constants

ICNST(\*) : an array containing (integer) user defined constants

## 3.12 Initial Conditions and Restarts

The analysis of a transient conduction or diffusion problem requires the specification of a set of initial conditions for all of the dependent variables. For cases where the temperature (and chemical composition) may be assumed uniform over each material, the initial conditions may be set through the input parameters available in the material data block (see Section 3.3). For problems in which the initial fields are more complex,

COYOTE allows the initial conditions to be input from an external file. In order to be compatible with COYOTE, the initial conditions must be written as a standard EXODUS II [5] file. The mechanics of attaching a restart file to a COYOTE execution are described in Section 4.3.

The output file containing the solutions generated by COYOTE has the same EXODUS II format as the initial condition file. Therefore, solutions obtained from COYOTE may generally be used directly as initial conditions for subsequent problems. An exception to this circumstance occurs for problems that include chemical reactions. The current limitations of the EXODUS II format requires that species data be output as element-based data that is uniform over the element volume (area). For purposes of computation, the species are defined as unknowns at the element integration points and are not uniform over the element. This incompatibility prevents the species data from being properly redistributed during a restart operation and this option is therefore not allowed in the present version of COYOTE.

The fact that the input and output files for the solution share a common format means that the code can be easily restarted from a particular point in a previous solution. Such a procedure consists of attaching a previous solution file, re-executing the input file and specifying within the solution data block (see Section 3.5) the iteration/timeplane number or time at which the solution is to be restarted. The input file must be re-executed since there is no explicit provision made for storing all of the data from a previous run. Typically, the computation time for problem setup and initialization is small enough that the storage of this information for a possible restart operation is not warranted. Note that when restarts are carried out using the above procedure, the resulting solution file contains the time history of the problem only from the restart timeplane.

### 3.13 Error Messages

COYOTE contains a number of error checks and tests for bad or inconsistent data, storage problems, *etc.* When an error is encountered, an error message and other pertinent data are printed and the program is terminated with a STOP ERROR. The error citations usually contain sufficient information to allow the problem to be diagnosed and corrected. For each listed error condition, the subroutine encountering the error is given along with an explanatory message. Also, any relevant integers, floating point data and/or character data that help to explain the problem are output by the code.

The general format for error messages takes the following form,

```
ERROR FOUND IN - ‘‘SUBROUTINE NAME’’
```

DESCRIPTION - ‘‘ERROR MESSAGE’’

RELEVANT PARAMETERS -

MESSAGE 1 = ‘‘INTEGER DATA’’

MESSAGE 2 = ‘‘INTEGER DATA’’

MESSAGE 3 = ‘‘FLOATING POINT DATA’’

MESSAGE 4 = ‘‘FLOATING POINT DATA’’

MESSAGE 5

MESSAGE 6 = ‘‘CHARACTER DATA’’



# Chapter 4

## Code Installation and Access

The use of a program such as COYOTE generally requires some knowledge of how the program is installed and operates on a specific computer system. In particular, details regarding the use of system and utility libraries, file usage, file formats, FORTRAN coding standards and access to the program are important to the successful and efficient use of the code. These topics become especially critical when the code is used for the analysis of very complex problems, when the code is modified for a new application or when the program is transported to a new computer system. In the following chapters, all of the above topics are discussed in some detail. The main emphasis will be on the operation of COYOTE on the Sandia National Laboratories computer systems; the operation of the code on other computers should involve only minor alterations to the code and operating procedures.

### 4.1 FORTRAN Coding and System Dependencies

The major part of the COYOTE program is written in ANSI standard Fortran 77 and should therefore be usable on any computer system that supports such a compiler. The iterative matrix solver used in COYOTE resides in a general software package developed by Shadid and Schunk [8] and is written in standard C. The enclosure radiation algorithms contained in the program CHAPARRAL by Glass [6] contain both Fortran and C components. The compilation and linking of the various COYOTE modules is most conveniently handled through a standard UNIX makefile which is available with the source distribution of the code.

As indicated in Section 2.2, COYOTE is a modular program that consists of a main driver routine and a number of subordinate, task oriented subroutines and utilities. Com-

munication between the main program and the various subroutines is accomplished primarily through named common blocks and subroutine parameter lists. Named common blocks generally contain pointers, option flags, sizing parameters and data that is invariant with respect to problem specification. Data that is stored in vectors and arrays are normally passed through subroutine parameter lists due to the use of a dynamic memory allocation scheme that is described below. A description of all of the major common blocks and arrays used in COYOTE is included in Appendix D. This information would normally be of value only if the code is to be modified.

A dynamic dimensioning procedure is used in COYOTE to allocate and release computer memory during program execution. Under this algorithm the individual vectors and arrays needed by the program are stored in (noncontiguous) blocks of memory under a single vector name. Pointers indicating the location of individual arrays within memory are maintained by the memory manager as is the allocation of needed storage space. All memory allocation is performed in the main program. Further details on the memory manager and its operation are given in [10]. Some computer systems may not permit execution-time changes in memory allocation; modifications to the code and memory manager to handle this circumstance are outlined in [10].

COYOTE also makes use of a few system dependent utilities. To increase portability of the code all of the system dependencies have been isolated in a few subroutines that are located in a utility library that accompanies the main code. The utilities available in the SUPES library [10] include date and time stamping, the memory manager and parsing of input for use in the free field input routines. These library routines are available for a number of commonly used computer systems and are heavily commented to ease the task of converting these utilities to other, nonsupported computers.

The pre- and postprocessing files currently used by COYOTE are in the EXODUS II format [5], which is a random access file structure based on NetCDF [11]. Access to these files is accomplished through a library of C subroutines that read and write data in the specified format. All of the mesh generation programs and graphics packages that are to be used in conjunction with COYOTE must provide data in the EXODUS II format. The contents of an EXODUS II file are summarized in Appendices E and F and are given in more detail in [5].

The standard version of COYOTE was developed for use on a spectrum of computer systems ranging from large vector supercomputers to engineering workstations. The code is also suitable for smaller computer systems, provided that the memory is sufficient to handle the required problem size. There are relatively few built-in limits in COYOTE with respect to the size or complexity of the heat conduction model. The code is designed to handle problems with an arbitrary number of elements, materials, boundary conditions, *etc.* up to the capacity of the computer memory. The largest portion of memory

is used for the storage of the assembled global diffusion matrix; this array is stored in a sparse matrix format [8,12] suitable for use with the iterative matrix solver.

## 4.2 File Formats

The disk files used internally by COYOTE are generally sequential access, unformatted files. The specific form of these files is not usually of concern to the user except for the pre- and postprocessing files and the restart (or solution) file (see Section 3.12).

As noted in the previous chapter, the pre- and postprocessing files utilize the EXODUS II format. Within COYOTE the pre- and post processing files are maintained separately to allow for reuse, though they share a substantial amount of data. The pre-processing file contains information from the mesh generator regarding element coordinates, connectivity, element types and boundary condition flags (see Appendix E). The postprocessing file contains a copy of the pre-processing data in addition to the computed solution variables (see Appendix F). Instructions for accessing EXODUS II data through the subroutine interface is available in [5].

## 4.3 File Usage

COYOTE makes use of a series of disk files for the storage of large blocks of input and element data, storage of solution vectors and input and output functions. All of the files used by the program are listed in Table 4.1 with a brief description of their function, their internal FORTRAN identification and format type. Specific formats for these files are not given as that information is readily available from a code listing. Previous chapters have commented on those file formats that would normally be of concern to the user.

When files are to be attached to a COYOTE job or saved after processing, the unit numbers used in the job control statements must conform to those indicated in the table. File names are arbitrary but must conform to the syntax rules for the particular operating system. The proper relation between file name and unit number must also be established in the job control statements. A utility program from the SUPES library [10] obtains needed file names from the operating system for use in opening files during program execution. As described in the next chapter, a script is available for COYOTE that simplifies the association of local file names with unit numbers. Note that within COYOTE all files are opened in the `OPNFIL` subroutine.

## 4.4 Access to the Code

A source version of COYOTE is maintained on the Network Storage System (NSS) of the Central Computer Facility at Sandia National Laboratories and on the Local Area Networks (LANs) operated by the Engineering Sciences Center, 1500. The current location of the program source code for any particular computer system may be obtained from the authors.

The utility libraries needed for the execution of COYOTE are available for several computer systems. Access to these routines varies depending on the particular operating system. Instructions for accessing and loading these routines are available in [5,6,8,10]. The required libraries include the CHAPARRAL radiation routines, the EXODUS II file access routines, the iterative matrix solution package and the SUPES utilities. When user supplied subroutines are employed with COYOTE they must be compiled and then made available to the loader as a library to be searched for unsatisfied external references. To simplify this compiling and loading process, a series of standard UNIX makefiles are available for COYOTE and its required libraries. The makefiles for various computer environments at Sandia are located with the source code and may be obtained from the authors.

In order to assist users in the running of COYOTE, a UNIX script is also available on the 1500 LAN's and through the SEACAS system [13]. This script has a standard UNIX, single-line, input style and allows execution of the code in either a batch or interactive mode on the host computer.

Unit Number	Fortran Name/ID	File Usage	File Access	File Format
5	NIN	Input file	Sequential	Coded
6	NOUT	Output file	Sequential	Coded
10	NTP0	EXODUS II input data	Random Access	Binary
11	NTP1	Free field input	Sequential	Binary
12	NTP2	EXODUS II output data	Random Access	Binary
13	NTP3	EXODUS II external data	Random Access	Binary
14	NTP4	EXODUS II restart data	Random Access	Binary
15	NTP5	Unused		Binary

Table 4.1: Files used in COYOTE



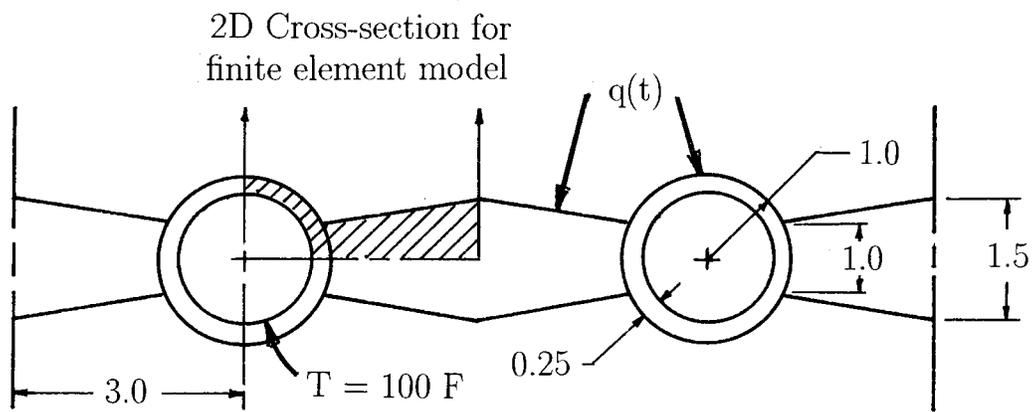
# Chapter 5

## Example Problems

A series of three problems have been included in this chapter to demonstrate the use of the previously described data cards and a few of the capabilities of the COYOTE program. Though the examples were designed to illustrate the basic features of the code, all possible options and subtleties of the program could not be covered. A more extensive demonstration of the use of COYOTE is available in [4].

### 5.1 Problem 1 - Finned Radiator

The first example problem consists of the finned tube radiator chapter shown in Figure 5.1 and was chosen to illustrate the use of a simple time dependent boundary condition. Symmetry considerations allowed the physical model in Figure 5.1 to be reduced to the three-dimensional, finite element model shown in Figure 5.2. Note that this model could have been reduced to a planar, two-dimensional geometry. The radiator is assumed to be at an initial temperature of 100° F. The temperature of the inside surface of the aluminum tube is maintained at 100° F while the outside surfaces of the radiator are subjected to the heat flux history shown in Figure 5.2. The planes of symmetry and the front and back surfaces of the model are treated as insulated. The thermal properties for the aluminum are also listed in Figure 5.2.



All dimensions in inches

Figure 5.1: Schematic of finned radiator problem.

The input data file for this problem is shown in Figure 5.3. The mesh was constructed using the programs FASTQ [14] and GEN3D [15] and consists of eight-node hexahedral elements with eight integration points in each element. The time dependent flux boundary condition was specified through a **TIME FUNCTION** data block in the input file. The time integration method used for the transient analysis of the problem was the trapezoid rule (predictor/corrector) with automatic time step control. Temperature results for each time step were written to an EXODUS II file for post-processing. A typical result from the transient analysis of this problem is shown in Figure 5.4 in the form of temperature contours on the surface of the radiator. These plots were generated using the graphics program BLOT [16].

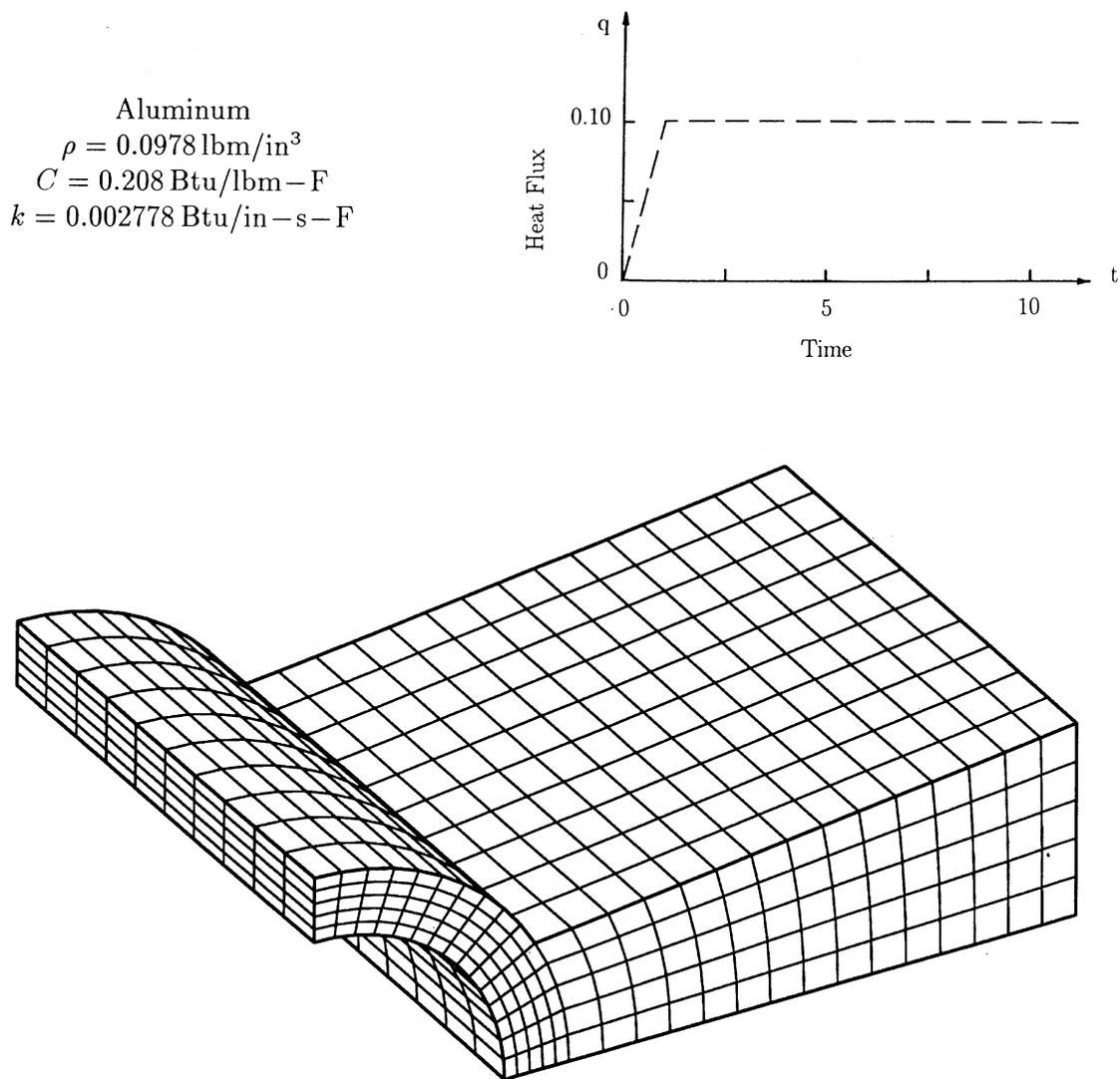


Figure 5.2: Finite element model for finned radiator.

## 5.2 Problem 2 - Volume Heating in a Cylinder

The second example problem considers a simplified one-dimensional model of an axisymmetric, two-dimensional geometry and allows the demonstration of a user supplied subroutine. A cylindrical region of heat generating material is encased by a thin layer of low conductivity material and a thicker layer of material having a relatively high thermal conductivity. The outer surface of the composite cylinder loses heat to the surrounding environment by natural convection. The geometry, boundary conditions, and material properties for this problem are shown in Figure 5.5. To estimate the radial temperature profile through the cylinder, a one-dimensional chapter at the mid-height of the cylinder is considered for the finite element model.

The input data file and user subroutine for this problem are listed in Figure 5.6 and 5.7. The mesh is constructed of four-node quadrilateral elements with four integration points in each element; the program FASTQ [14] was used to construct the mesh. The volumetric heating history for the inner cylindrical region was specified through a **TIME FUNCTION** data block in the input file as shown in Figure 5.6. The heat transfer coefficient for the outside surface of the cylinder was provided in the user supplied subroutine USRHTC. This subroutine contains correlations for laminar and turbulent natural convection along a vertical cylinder and selects the appropriate heat transfer coefficient based on the current surface and film temperatures. The initial temperature of the cylindrical region was set at 20° C; the reference temperature for the convection process was also 20° C.

The time integration method used for the transient analysis of this problem was the trapezoid rule with the predictor/corrector option and the automatic time step selection. The time dependent response of the cylinder was computed for 30 seconds, during which time the volume heating was held at two different levels and then turned off. The resulting temperature histories at several radial positions are shown in Figure 5.8. This plot was generated with the graphics program BLOT[16].

## 5.3 Problem 3 - Surface Heating of a Plate

The last example simulates the thermal response of a simple three-dimensional plate that is subjected to a moving surface heat flux. A schematic of the problem is shown in Figure 5.9. A cylindrical heat flux of constant magnitude is constrained to move at a constant velocity on the square aluminum plate. The heat source travels through one complete circular revolution and is then removed from the surface. The plate is assumed

to be insulated and loses no energy through any of its surfaces. The plate is initially at a uniform temperature of 20° C.

The input data file for the COYOTE analysis of this problem is shown in Figure 5.10. The mesh of eight-node hexahedral elements was generated with the FASTQ [14] and GEN3D [15] programs. The moving heat flux was described in the user supplied subroutine USRFLX which is listed in Figure 5.11. A number of **USER CONstants** were defined to allow the parameters for the motion of the flux distribution to be defined through the input file and avoid recompilation of the user subroutine.

The time dependent simulation of the plate was carried out for 150 seconds at which point the plate returns to a uniform but higher temperature state. The simulation was run using the automatic time step option and the trapezoid rule integration method. The resulting temperature contours on the surface of the plate are shown in Figure 5.12 for a series of times during simulation. The contour plots were generated with the BLOT code [16]. A more extensive analysis of this problem is included in the COYOTE Example Problems manual [4].

```
TITLE
HEAT CONDUCTION IN A FINNED RADIATOR
END
MATERIAL,ALUMINUM
DENSITY=0.0978
SPECIFIC HEAT=0.208
CONDUCTIVITY=2.778E-3
INITIAL TEMPERATURE=100.
END
PROBLEM DEFINITION
GEOMETRY=3D
ELEMENT BLOCK=10,ALUMINUM
BCTYPE=TEMPERATURE,100,100.
BCTYPE=HEAT FLUX,500,TFUNCTION=100
END
SOLUTION,1,TIME DEPENDENT
INTEGRATION METHOD=TRAPEZOID
TIME STEP OPTION=AUTOSTEP
TIME STEP=0.05
INITIAL TIME=0.0
FINAL TIME=10.0
MINIMUM TIME STEP=.05
END
POST
NODAL DATA=TEMPERATURE
OUTPUT FREQUENCY=1
END
TIME FUNCTION=100
0.0,0.0
1.0,0.10
15.0,0.10
END
STOP
```

Figure 5.3: Input file for COYOTE simulation of a finned radiator.

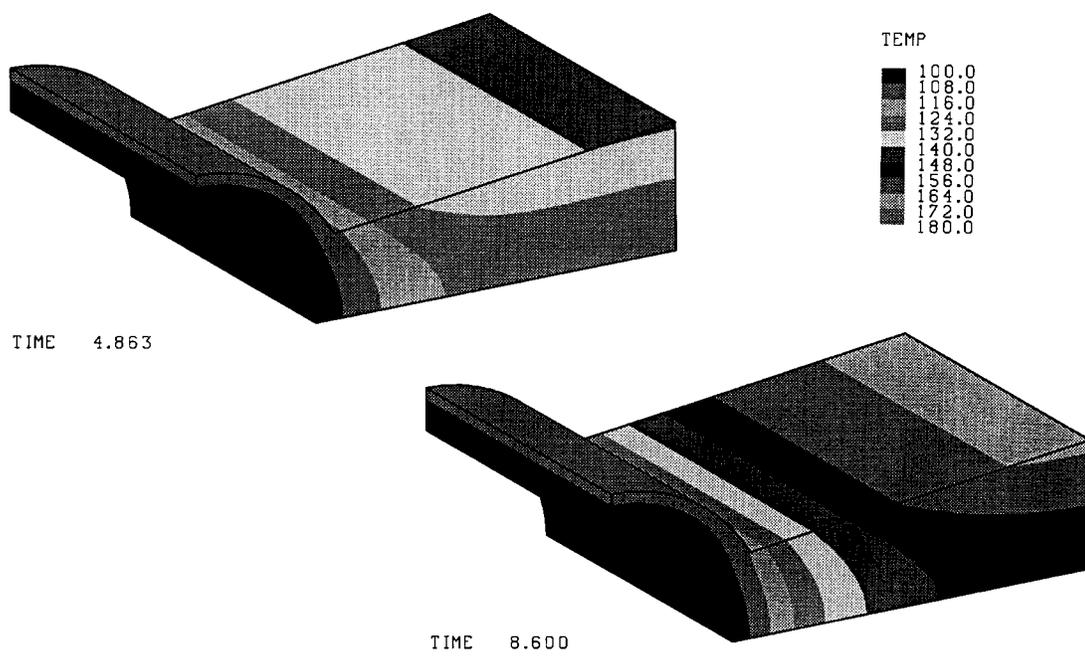


Figure 5.4: Temperature contours for finned radiator problem.

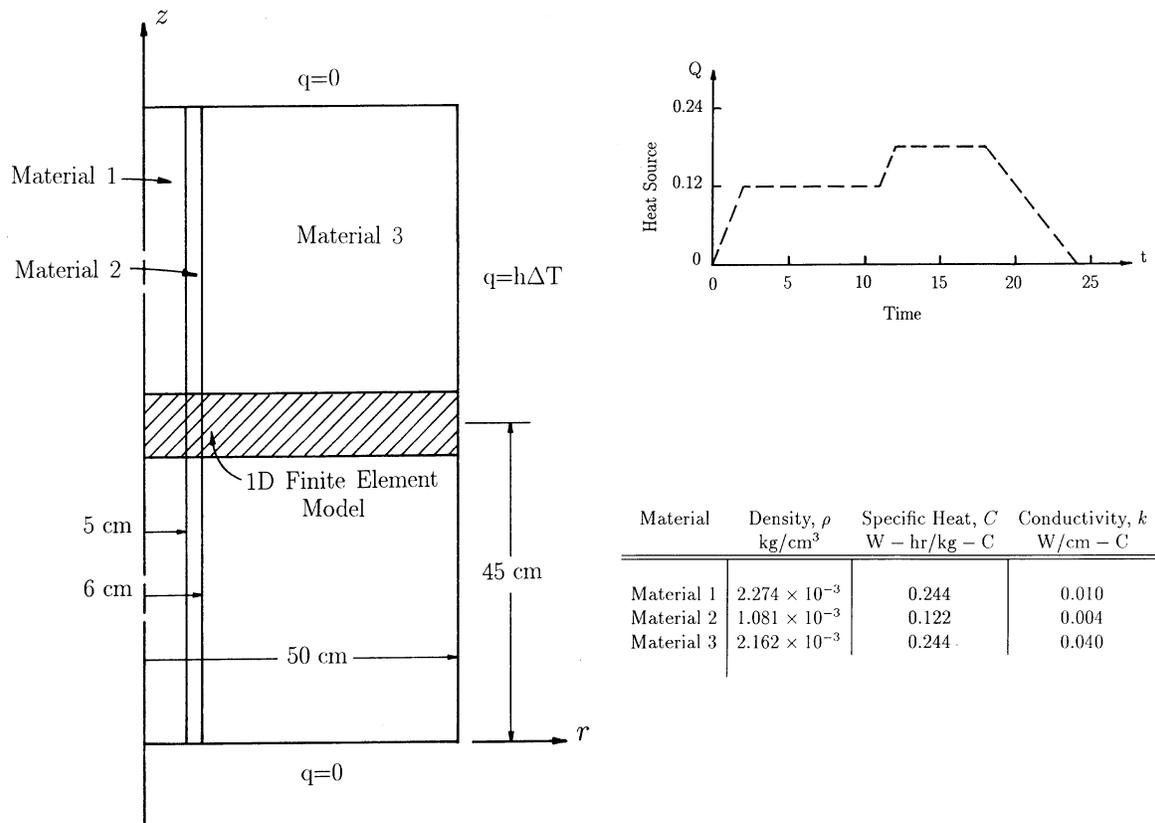


Figure 5.5: Schematic of volume heated cylinder problem.

```

TITLE
ONE-DIMENSIONAL CONDUCTION PROBLEM
VOLUMETRIC HEATING IN A CYLINDER
FOUR NODE QUADRILATERAL ELEMENT
END
MATERIAL,MAT1
DENSITY=2.274E-3
SPECIFIC HEAT=0.244
CONDUCTIVITY=0.01
VOLUME HEATING=TFUNCTION,100
INITIAL TEMPERATURE=20.0
END
MATERIAL,MAT2
DENSITY=1.081E-3
SPECIFIC HEAT=0.122
CONDUCTIVITY=0.004
INITIAL TEMPERATURE=20.0
END
MATERIAL,MAT3
DENSITY=2.162E-3
SPECIFIC HEAT=0.244
CONDUCTIVITY=0.04
INITIAL TEMPERATURE=20.0
END
PROBLEM DEFINITION
GEOMETRY=AXISYMMETRIC
ELEMENT BLOCK=10,MAT1
ELEMENT BLOCK=20,MAT2
ELEMENT BLOCK=30,MAT3
BCTYPE=CONVECTION,10,COEFFICIENT=USER,,*
  REFERENCE=20.0
END
SOLUTION,1,TIME DEPENDENT
INTEGRATION METHOD=TRAPEZOID
TIME STEP OPTION=AUTOSTEP
TIME STEP=0.10
INITIAL TIME=0.0
FINAL TIME=30.0
END
POST
NODAL DATA=TEMPERATURE
OUTPUT FREQUENCY=1
END
TIME FUNCTION=100
0.0,0.0
2.0,0.12
11.0,0.12
12.0,0.18
18.0,0.18
24.0,0.0
35.0,0.0
END
STOP

```

Figure 5.6: Input file for COYOTE simulation of a volume heated cylinder.

```

SUBROUTINE USRHTC (HCOEF,TEMPS,TREF,XS,YS,ZS,NNODES,IDSSET,TIME,
*                 KSTEP,RCONST,ICONST)
C
C *****
C
C DESCRIPTION:
C   USER SUBROUTINE TO EVALUATE THE CONVECTIVE HEAT TRANSFER
C   COEFFICIENT FOR AN ELEMENT SURFACE OR EDGE
C
C PARAMETERS:
C   HCOEF      (REAL) - Heat transfer coefficient evaluated at the
C                   element surface or edge nodes (output)
C   TEMPS      (REAL) - Temperatures at the element surface or edge
C                   nodes (input)
C   TREF       (REAL) - Reference temperatures for the element
C                   surface or edge nodes (input)
C   XS,YS,ZS   (REAL) - Coordinates for the element surface or edge
C                   nodes (input)
C   NNODES     (INTEGER) - Number of surface or edge nodes (input)
C   IDSSET     (INTEGER) - Boundary condition side set id (input)
C   TIME       (REAL) - Current time (input)
C   KSTEP      (INTEGER) - Current iteration/time step number (input)
C   RCONST     (REAL) - User constants (input)
C   ICONST     (INTEGER) - User constants (input)
C
C *****
C
C   DIMENSION HCOEF(*), TEMPS(*), TREF(*)
C   DIMENSION XS(*), YS(*), ZS(*)
C   DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C THIS ROUTINE IS FOR THE FREE CONVECTION FLOW OF AIR ALONG A
C VERTICAL CYLINDRICAL SURFACE
C THE HEAT TRANSFER COEFFICIENT IS EVALUATED AT A HEIGHT AL (FT)
C AIR PROPERTIES ARE SPECIFIED IN ENGLISH UNITS
C THE HEAT TRANSFER COEFFICIENT IS COMPUTED IN BTU/HR-FT**2-F AND
C THEN CONVERTED TO METRIC UNITS AS USED IN THE PROBLEM DEFINITION
C

```

Figure 5.7: User subroutine for COYOTE simulation of a volume heated cylinder.

```

      AL = 1.5
      G = 32.2
      CONVERT = 5.678E-4
C
C   LOOP ON EVALUATION POINTS
C
      DO 10 I = 1, NNODES
      TW = TEMPS(I)
      TR = TREF(I)
C
C   CONVERT TEMPERATURES TO FAHRENHEIT (RANKINE)
C   AND EVALUATE AIR PROPERTIES AND GRASHOF NUMBER
C
      DELT = (TW-TR)*(9./5.)
      TFILM = ((TW+TR)*0.5 + 273.)*(9./5.)
C
      BETA = 1./TFILM
      AMU = (7.3094E-7)*((TFILM**1.5)/(TFILM+198.))
      RHO = 39.68/TFILM
C
      GR = ((RHO**2)*G*BETA*(AL**3)*DELT)/(AMU**2)
C
C   EVALUATE HEAT TRANSFER COEFFICIENT - CORRELATION DUE TO McADAMS
C   LAMINAR FLOW (GR < 1.0E9)
C   TURBULENT FLOW (GR > 1.0E9)
C
      IF (GR .LT. 1.0E4) THEN
        HT = 0.0
      ELSE IF (GR .GE. 1.0E4 .AND. GR .LT. 1.0E9) THEN
        HT = .29*(DELT/AL)**.25
        HT = HT * CONVERT
      ELSE
        HT = .19*(DELT**.333333)
        HT = HT * CONVERT
      END IF
C
      HCOEF(I) = HT
10  CONTINUE
      RETURN
      END

```

Figure 5.7: Continuation of user subroutine for simulation of a volume heated cylinder.

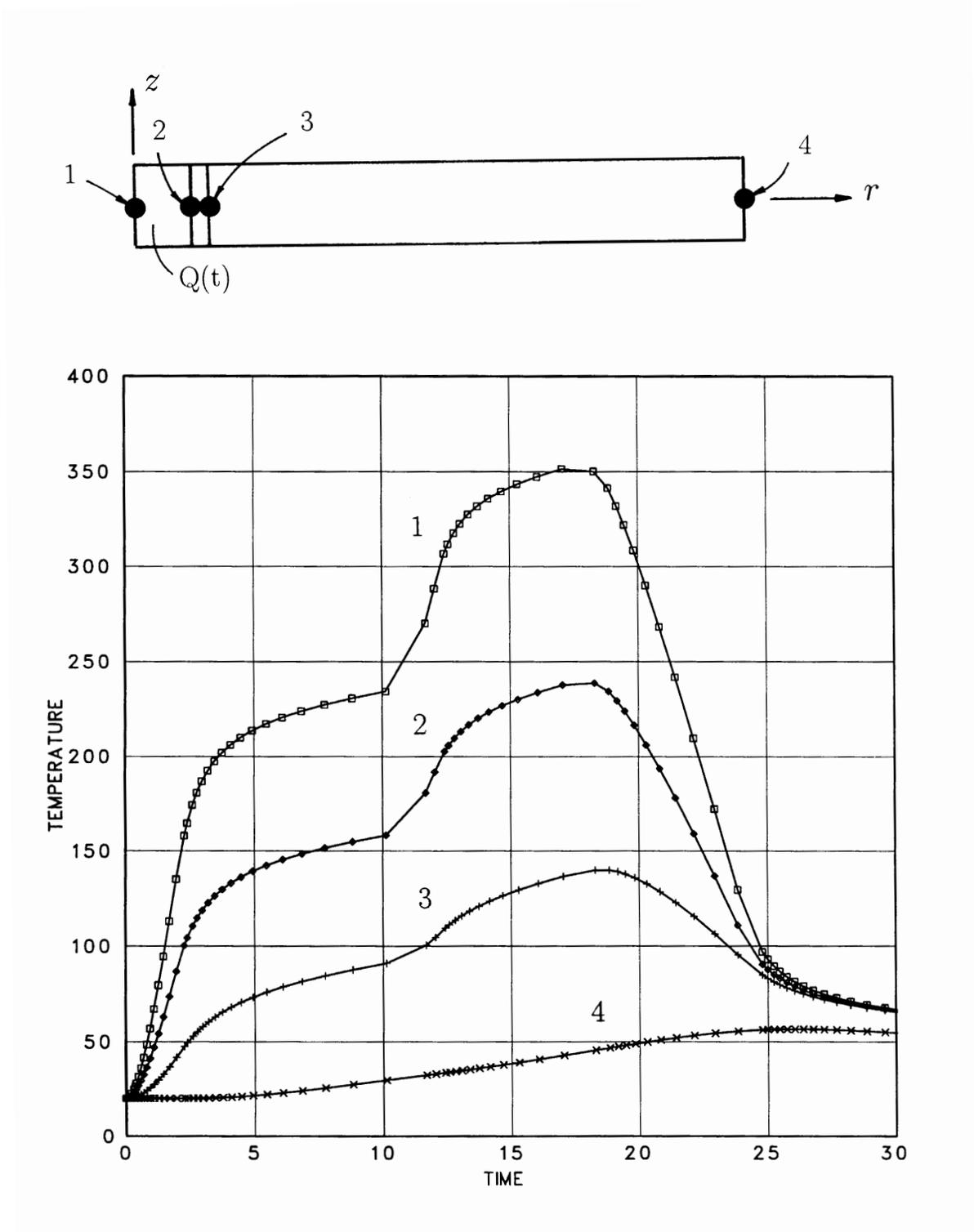


Figure 5.8: Temperature histories for volume heated cylinder problem.

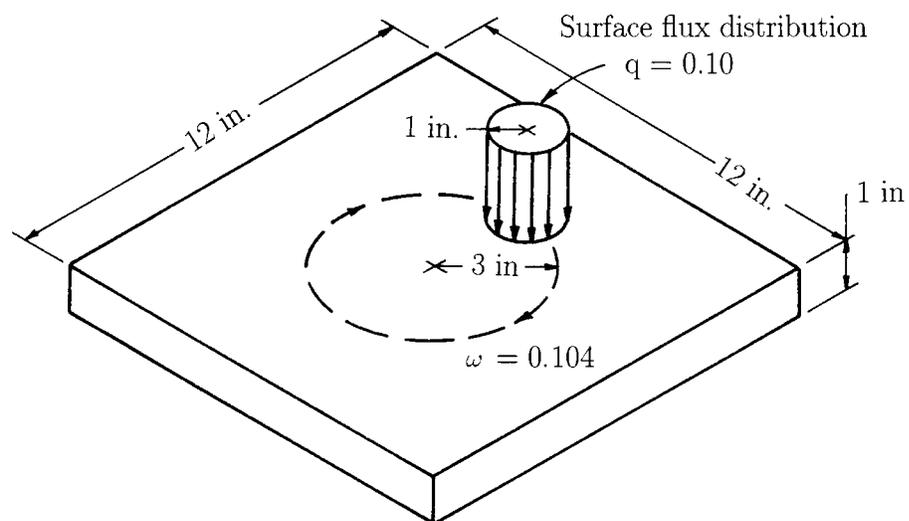


Figure 5.9: Schematic of plate heating problem.

```
TITLE
MOVING HEAT SOURCE
CYLINDRICAL FLUX DISTRIBUTION MOVING IN A CIRCULAR PATH
ON A RECTANGULAR BLOCK OF ALUMINUM
EIGHT NODE HEXAHEDRAL ELEMENT
END
MATERIAL,ALUMINUM
DENSITY=0.0978
SPECIFIC HEAT=0.208
CONDUCTIVITY=2.778E-3
INITIAL TEMPERATURE=20.
END
PROBLEM DEFINITION
GEOMETRY=3D
ELEMENT BLOCK=10,ALUMINUM
BCTYPE=HEAT FLUX,100,USER
END
SOLUTION,1,TIME DEPENDENT
INTEGRATION METHOD=TRAPEZOID
TIME STEP OPTION=AUTOSTEP
TIME STEP=0.1
INITIAL TIME=0.0
FINAL TIME=150.0
NUMBER TIME STEP=500
INTEGRATION TOLERANCE=1.0E-3
END
POST
NODAL DATA=TEMPERATURE
OUTPUT FREQUENCY=1
END
USER CONSTANTS
USER REAL=1,3.0
USER REAL=2,0.1047198
USER REAL=3,1.00
USER REAL=4,60.0
USER REAL=5,0.10
END
STOP
```

Figure 5.10: Input file for COYOTE simulation of a surface heated plate.

```

SUBROUTINE USRFLX (FLUX,TEMPS,XS,YS,ZS,NNODES,IDSSET,TIME,KSTEP,
*                RCONST,ICONST)
C
C *****
C
C DESCRIPTION:
C   USER SUBROUTINE TO EVALUATE THE APPLIED HEAT FLUX FOR AN
C   ELEMENT SURFACE OR EDGE
C
C PARAMETERS:
C   FLUX      (REAL) - Heat flux evaluated at the element surface or
C                 edge nodes (output)
C   TEMPS     (REAL) - Temperatures at the element surface or edge
C                 nodes (input)
C   XS,YS,ZS  (REAL) - Coordinates for the element surface or edge
C                 nodes (input)
C   NNODES    (INTEGER) - Number of surface or edge nodes (input)
C   IDSSET    (INTEGER) - Boundary condition side set id (input)
C   TIME      (REAL) - Current time (input)
C   KSTEP     (INTEGER) - Current iteration/time step number (input)
C   RCONST    (REAL) - User constants (input)
C   ICONST    (INTEGER) - User constants (input)
C
C *****
C
C DIMENSION FLUX(*), TEMPS(*)
C DIMENSION XS(*), YS(*), ZS(*)
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C USER SUPPLIED FORTRAN CODE TO EVALUATE THE HEAT FLUX VECTOR, FLUX
C
C THIS ROUTINE DESCRIBES THE MOTION OF A SURFACE FLUX THAT HAS A
C UNIFORM CIRCULAR DISTRIBUTION (RADIUS=RFLUX) ABOUT A CENTER POINT.
C THE CENTER OF THE FLUX DISTRIBUTION MOVES IN A CIRCULAR PATH WITH
C VELOCITY V=RCENTR*OMEGA. THE FLUX IS NONZERO FOR 0 < TIME < ENDTIM
C

```

Figure 5.11: User subroutine for COYOTE simulation of surface heated plate.

```

C     SET PARAMETERS FROM INPUT
C
C     RCENTR = RCONST(1)
C     OMEGA = RCONST(2)
C     RFLUX = RCONST(3)
C     ENDTIM = RCONST(4)
C     FLUXVAL = RCONST(5)
C
C     CHECK TIME FOR ACTIVE FLUX
C
C     IF (TIME .LE. ENDTIM) THEN
C
C     COMPUTE LOCATION OF CENTER OF HEAT SOURCE
C
C     XC = RCENTR*COS(OMEGA*TIME)
C     ZC = RCENTR*SIN(OMEGA*TIME)
C
C     EVALUATE FLUX AT ELEMENT SURFACE NODES
C
C     DO 10 I = 1,NNODES
C       RAD = SQRT((XS(I) - XC)**2 + (ZS(I) - ZC)**2)
C       IF(RAD .LE. RFLUX) THEN
C         FLUX(I) = FLUXVAL
C       ELSE
C         FLUX(I) = 0.0
C       END IF
C     10 CONTINUE
C
C     FLUX INACTIVE
C
C     ELSE
C
C     DO 20 I =1,NNODES
C       FLUX(I)=0.0
C     20 CONTINUE
C     END IF
C
C     RETURN
C     END

```

Figure 5.11: Continuation of user subroutine for simulation of a surface heated plate.

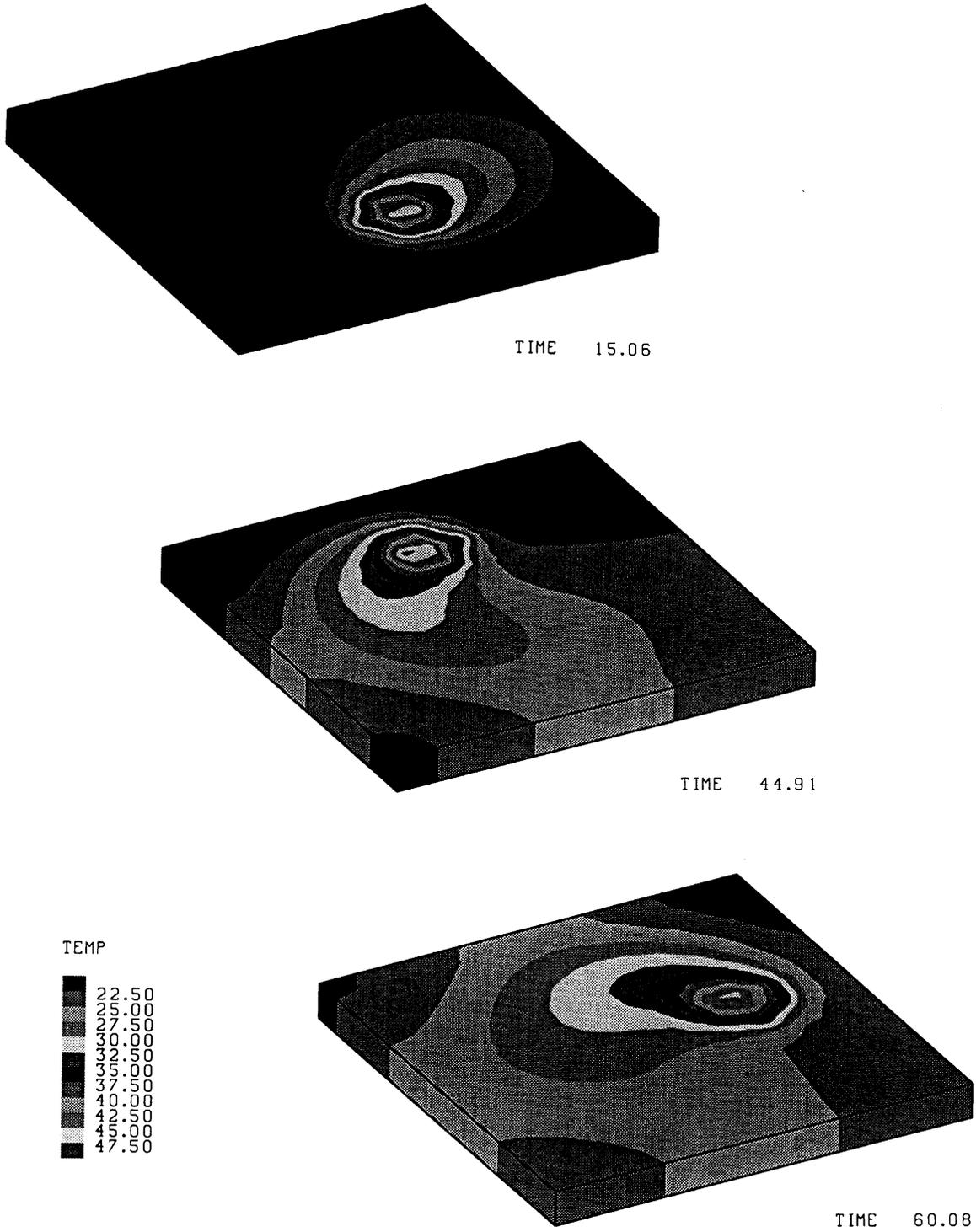


Figure 5.12: Temperature contours for the surface heated plate problem.



# Chapter 6

## References

1. D. K. Gartling, “NACHOS II - A Finite Element Computer Program for Incompressible Flow Problems, Part I - Theoretical Background,” SAND86-1816, Sandia National Laboratories, Albuquerque, NM (1986)
2. D. K. Gartling and C. E. Hickox, “MARIAH - A Finite Element Computer Program for Incompressible Porous Flow Problems,” SAND79-1622, Sandia National Laboratories, Albuquerque, NM (1979)
3. D. K. Gartling and R. E. Hogan, “COYOTE II - A Finite Element Computer Program for Nonlinear Heat Conduction Problems, Part I - Theoretical Background,” SAND94-1173, Sandia National Laboratories, Albuquerque, NM (1994)
4. D. K. Gartling and M. B. Sirman, “COYOTE II - A Finite Element Computer Program for Nonlinear Heat Conduction Problems, Part III - Example Problems,” SAND94-1180, Sandia National Laboratories, Albuquerque, NM (1994)
5. L. A. Schoof and V. R. Yarberr, “EXODUS II - A Finite Element Data Model,” SAND92-2137, Sandia National Laboratories, Albuquerque, NM (1994)
6. M. W. Glass, “CHAPARRAL - A Library Package for Solving Large Enclosure Radiation Heat Transfer Problems,” (in preparation), Sandia National Laboratories, Albuquerque, NM (1994)
7. A. B. Shapiro, “FACET - A Radiation View Factor Computer Code for Axisymmetric, 2D Planar, and 3D Geometries with Shadowing,” UCID-19887, Lawrence Livermore Laboratory, Livermore, CA (1983)
8. P. R. Schunk and J. N. Shadid, “Iterative Solvers in Implicit Finite Element Codes,” SAND92-1158, Sandia National Laboratories, Albuquerque, NM (1992)

9. T. R. Young, "CHEMEQ - A Subroutine for Solving Stiff Ordinary Differential Equations," NRL Memorandum Report 4091, Naval Research Laboratory, Washington, DC (1980)
10. J. R. Red-Horse, W. C. Mills-Curran, and D. P. Flanagan, "SUPES Version 2.1 - A Software Utilities Package for the Engineering Sciences," SAND90-0247, Sandia National Laboratories, Albuquerque, NM (1990)
11. "NetCDF User's Guide; An Interface for Data Access," Version 1.11, Unidata Program Center (1991)
12. Y. Saad, "SPARSKIT: A Basic Tool for Sparse Matrix Computations," Tech. Report, Research Institute for Advanced Computer Science, NASA Ames, Moffitt Field, CA (1990)
13. G. D. Sjaardema, "Overview of the Sandia National Laboratories Engineering Analysis Code Access System," SAND92-2292, Sandia National Laboratories, Albuquerque, NM (1993)
14. T. D. Blacker, "FASTQ Users Manual, Version 1.2," SAND88-1326, Sandia National Laboratories, Albuquerque, NM (1988)
15. G. D. Sjaardema, "GEN3D: A GENESIS Database 2D to 3D Transformation Program," SAND89-0485, Sandia National Laboratories, Albuquerque, NM (1989)
16. A. P. Gilkey and J. H. Glick, "BLOT - A Mesh and Curve Plot Program for the Output of a Finite Element Analysis," SAND88-1432, Sandia National Laboratories, Albuquerque, NM (1989)
17. I. M. Levi, "User's Guide to the Transient Heat Conduction Finite Element Code HTCON," MM70-5424-17, Bell Telephone Laboratories Memorandum, Whippany, NJ (1970)
18. A. F. Emery, K. Sugihara and A. T. Jones, "A Comparison of Some of the Thermal Characteristics of Finite-Element and Finite-Difference Calculations of Transient Problems," *Num. Heat Trans.*, **2**, 97-113 (1979)
19. F. Damjanic and D. R. J. Owen, "Practical Considerations for Thermal Transient Finite Element Analysis Using Isoparametric Elements," *Nucl. Engrg. Des.*, **69**, 109-126 (1982)
20. E. Rank, C. Katz and H. Werner, "On the Importance of the Discrete Maximum Principle in Transient Analysis Using Finite Element Methods," *Int. J. Num. Meth. Engrg.*, **19**, 1771-1782 (1983)
21. H. S. Carslaw and J. C. Jaeger, "*Conduction of Heat in Solids*," Clarendon Press, Oxford, 2nd Edition (1959)

# Chapter 7

## Appendices



## Appendix A - Summary of Input Commands

In this section all of the data blocks and associated data cards recognized by COYOTE are summarized. No attempt is made to define the parameters on each input line since these descriptions are available in the main text.

### Title Data Block :

**TITLe**

THIS IS AN EXAMPLE OF A PROBLEM TITLE

.

.

THESE ARE EXAMPLES OF SUBTITLE LINES

UP TO 10 LINES OF PROBLEM DESCRIPTION MAY

BE INCLUDED ON THESE CARDS

.

.

.

**ENDtitle**

### Material Data Block :

MATerial, material name, *material model* □

.

.

DENsity= $\rho$

DENsity=USER

DENsity=VFUNction, idvar

.

.

SPECific HEAT= $C_p$

SPECific HEAT=USER

SPECific HEAT=VFUNction, idvar

.

.

CONDUCTivity= $k_{11}, k_{22}, k_{33}$  □

CONDUCTivity=USER □

CONDUCTivity=VFUNction, idvar1, idvar2, idvar3 □

.

.

TENsor ROTation= $x\hat{x}, y\hat{x}, z\hat{x}, x\hat{y}, y\hat{y}, z\hat{y}$

TENsor ROTation=USER

.

.

LATent HEAT= $L$

.

SOLidus TEMPerature= $T_{sol}$

.

LIQuidus TEMPerature= $T_{liq}$

.

ENTHalpy=USER

ENTHalpy=VFUNction, idvar

.

PHASe CHANge=*property, derivative method*

.

EMISSivity= $\epsilon$

EMISSivity=USER

EMISSivity=VFUNction, idvar

.

VOLume HEATIng= $Q$

VOLume HEATIng=USER

VOLume HEATIng=TFUNction, idtim

VOLume HEATIng=VFUNction, idvar

.

INITial TEMPerature= $T_{init}$

.

REACTive MIXture= $n_{species}$ ,  $n_{reactions}$ , *USER*

.

SPECies= $name_1$ ,  $name_2$ ,  $\dots$ ,  $name_{n_{species}}$

.

SPECies PHASe= $phase_1$ ,  $phase_2$ ,  $\dots$ ,  $phase_{n_{species}}$

.

FRACTION CONDensed=  $frac$

.

INITial CONcentration= $N_1^0$ ,  $N_2^0$ ,  $\dots$ ,  $N_{n_{species}}^0$

.

·  
MINimum CONcentration= $N_1^{min}, N_2^{min}, \dots, N_{n_{species}}^{min}$

·  
STERic COEFFicients= $\beta_1, \beta_2, \dots, \beta_{n_{reaction}}$

·  
PRExponential FACTor= $A_1, A_2, \dots, A_{n_{reaction}}$

·  
LPRExponential FACTor= $\ln A_1, \ln A_2, \dots, \ln A_{n_{reaction}}$

·  
ACTivation ENERgy= $E_1, E_2, \dots, E_{n_{reaction}}$

·  
ENERgy RELease= $q_1, q_2, \dots, q_{n_{reaction}}$

·  
CONcentration EXPonents, species no.=  $\mu_{i1}, \mu_{i2}, \dots, \mu_{i,n_{reaction}}$

·  
STOichiometric COEFFicients, species no.=  $\nu_{i1}, \nu_{i2}, \dots, \nu_{i,n_{reaction}}$

·  
CHEMistry ACTivation TEMPerature= $T_{chem}$

·  
ENDmaterial □

Problem Definition Data Block :

PROBLEM DEFINITION, format □

·  
GEOMETRY=type □

·  
ELEMENT BLOCK= block id, material name, integration rule □

·  
BCtype=bcname, nodeset id, bc specification, mfactor

·  
BCtype=bcname, sideset id, bc specification, mfactor

BCtype=bcname, sideset id, COEFFicient = coef specification, *mfactor*,  
TREference = ref specification, *mfactor*  
 .  
 .  
BCtype=bcname, SURFace=sideset id1, SURFace= sideset id2,  
COEFFicient = coef specification  
BCtype=bcname, BLOCk=block id1, SURFace=sideset id1,  
COEFFicient = coef specification  
BCtype=bcname, BLOCk=block id1, BLOCk=block id2,  
COEFFicient = coef specification  
 .  
 .  
BCtype=bcname, sideset id, enclosure no.  
 .  
 .  
ENClosure=enclosure no., enclosure type, *blocking option*,  
*smoothing option, row-sum tolerance, Area<sub>∞</sub>, T<sub>∞</sub>, ε<sub>∞</sub>*  
 .  
 .  
VIEWfactor IO=type, filename, *format*  
 .  
 .  
VIEWfactor COMPutation=method, *storage format, resolution, print option*  
 .  
 .  
VIEWfactor GRID=enclosure no., *max surface intervals, no. rotational divisions,*  
*no. x-grid divisions, no. y-grid divisions, no. z-grid divisions, min seperation distance,*  
*clipping plane scale factor, no. of threads*  
 .  
 .  
VELocity=type, block id, velocity specification  
 .  
 .  
EXTernal NODal FIELd=name1, name2, ... , namen  
EXTernal ELEMEnt FIELd=name1, name2, ... , namen  
 .  
 .  
PRINted OUTput=type  
 .  
 .  
OUTput LOCations=e1, e2, e3, ... , en TO em, ...  
 .  
 .  
SPECial OUTput=number of points, x1, y1, z1, x2, y2, z2 ....

.  
 .  
HEAT FLUX=*sideset id1,sideset id2, ... sideset idn*  
 .  
 .  
HEAT FUNction= $\mathcal{H}_0$   
 .  
 .  
DEAth, block id, level, *variable, mode*  
 .  
 .  
DElete MATerial, block id, variable, value  
 .  
 .  
ADD MATerial, block id, variable, value  
 .  
 .  
SIGma=value  
 .  
 .  
GAS CONStant=value  
 .  
 .  
ENDproblem □

**Solution Data Block :**

SOLution, solution block number, time dependence □  
 .  
 .  
REStart TIME= $t_{restart}$   
 .  
 .  
REStart STEP=*step number*  
 .  
 .  
ITERative METHod=scheme  
 .  
 .  
INTEgration METHod=scheme, *predictor option*  
 .  
 .  
CAPacitance MATRix=type  
 .  
 .

TIME STEP OPTion=type

.

INTEgration TOLerance=integ tol

.

NORM TEMPerature= $T_{norm}$

.

TIME STEP FACTor= $\beta_{explicit}$

.

TIME STEP= $\Delta t$  □

.

INITial TIME= $t_{init}$

.

FINal TIME= $t_{final}$  □

.

NUMber TIME STEPs=nstep □

.

ABSolute TEMPerature LIMit= $T_{lim}$

.

MINimum TIME STEP= $\Delta t_{min}$

.

MAXimum TIME STEP= $\Delta t_{max}$

.

MAXimum TEMPerature STEP= $\Delta T_{max}$

.

CHEMistry STEP MULTiplier= $X_{chem}$

.

CONVergence TOLerance=tol

.

RELaxation FACTor= $\alpha$

.

.  
MAXimum ITERations=itermax

.  
PRINTed OUTput=option, frequency

.  
RADiation SOLution=method, *tolerance, itermax, relax factor*

.  
VIEWfactor UPDate=option, frequency

.  
MATRix SOLver=method, *krylov subspace*

.  
PREConditioner TYPE=option, *polynomial order*

.  
L2 NORM= $L_2$

.  
RESidual NORM=resid

.  
MAXimum MATRix ITERations=matrix iter

.  
EPSilon MINimum= $\epsilon_{min}$

.  
EPSilon MAXimum= $\epsilon_{max}$

.  
MINimum CHEMistry TIMEstep= $\Delta t_{min}^{chem}$

.  
PERcentage ASYMptotics=pctasymp

.  
TOLerance ASYMptotics=tolasymp

.  
ENDsolution

□

Post-processing Data Block :

```

POST
.
.
NODal DATA=name1, name2,....
.
.
ELEMent DATA=name1, name2,....
.
.
CHEMistry DATA=name1, name2,....
.
.
GLOBAL DATA=name1, name2,....
.
.
OUTput FREQuency=nsteps
.
.
OUTput TIMEs=  $t_1, t_2, \dots t_n$ 
.
.
OUTput TIME STEP=  $\Delta t_{out}$ 
.
.
ENDpost

```

Time Function Data Block :

```

TIME FUNction=function id
 $t_1, f(t_1)$ 
 $t_2, f(t_2)$ 
 $t_3, f(t_3)$ 
.
.
.
 $t_n, f(t_n)$ 
.
.
ENDtime

```

Variable Function Data Block :

```

VARIABLE FUNction=function id
 $T_1, f(T_1)$ 

```

$T_2, f(T_2)$   
 $T_3, f(T_3)$

.

.

.

$T_n, f(T_n)$

.

.

**ENDvariable**

User Constants Data Block :

**USER CONstants**

.

.

**USER REAL=n, rvalue**

.

.

.

**USER INTeger=m, ivalue**

.

.

**ENDconstant**

Termination Data :

**EXIT or STOP**



## Appendix B - Consistent Units

The following list provides examples of consistent units for quantities that may be encountered in the use of COYOTE.

Quantity	English	Metric	S I
Length	foot (ft)	centimeter (cm)	meter (m)
Time	second (s)	second (s)	second (s)
Mass	lb <sub>m</sub>	gram (gm)	kilogram (kg)
Force	lb <sub>m</sub> -ft/s <sup>2</sup>	gm-cm/s <sup>2</sup>	Newton (N)
Energy	Btu	erg	joules (J)
Temperature	Fahrenheit (F) or Rankine (R)	Centigrade (C) or Kelvin (K)	Centigrade (C) or Kelvin (K)
Velocity	ft/s	cm/s	m/s
Density	lb <sub>m</sub> /ft <sup>3</sup>	gm/cm <sup>3</sup>	kg/m <sup>3</sup>
Specific Heat	Btu/lb <sub>m</sub> -F	erg/gm-C	J/kg-K
Power	Btu/s	erg/s	J/s (Watt)
Heat Flux	Btu/ft <sup>2</sup> -s	erg/cm <sup>2</sup> -s	J/m <sup>2</sup> -s
Heat Transfer Coefficient	Btu/ft <sup>2</sup> -s-F	erg/cm <sup>2</sup> -s-C	J/m <sup>2</sup> -s-K
Thermal Conductivity	Btu/ft-s-F	erg/cm-s-C	J/m-s-K
Volumetric Heat Source (Distributed)	Btu/ft <sup>3</sup> -s	erg/cm <sup>3</sup> -s	J/m <sup>3</sup> -s
Heat Source (Point)	Btu/s	erg/s	J/s
Emissivity	-	-	-
Chemical species	-	-	-
Pre-exponential*	1/s	1/s	1/s
Activation energy	ft-lb <sub>f</sub> /lb-mol	erg/mol	J/mol
Stefan-Boltzmann Constant, $\sigma$	$4.761 \times 10^{-13}$ Btu/s-ft <sup>2</sup> -R <sup>4</sup>	$5.669 \times 10^{-5}$ erg/s-cm <sup>2</sup> -K <sup>4</sup>	$5.669 \times 10^{-8}$ J/s-m <sup>2</sup> -K <sup>4</sup>
Gas Constant, $R$	$1.544 \times 10^3$ ft-lb <sub>f</sub> /lb-mol-R	$8.3143 \times 10^7$ erg/K-mol	8.3143 J/K-mol

\* The units for the pre-exponential will change with the use of a steric factor.



## Appendix C - Initial Time Step Estimation

The analysis of a transient diffusion problem requires the selection of a suitable time step for the integration procedure. This process can be automated with the time step being adaptively selected to preserve a specified time truncation error. The AUTOstep option provides this capability in COYOTE (see Section 3.5). However, the time step selection process is not self-starting and some initial estimate of an appropriate time step is still required. In the following the discussion is primarily limited to implicit integration methods, since these techniques are usually the methods of choice for most conduction models.

The selection of too *large* an initial time step can result in the loss of temporal accuracy in the solution and produce a nonphysical oscillatory response. Likewise, an inappropriately *small* initial time step may produce nonphysical, spatial oscillations in the early time temperature field due to the limited resolution ability (of temperature gradients) of the finite element mesh. Either of these difficulties may lead to stability problems if the boundary value problem is nonlinear. In any event, the solution during the oscillatory period is not accurate and these occurrences are to be avoided. A method for estimating an appropriate initial time step is outlined below. This procedure is originally due to Levi [17] though it has been discussed by several other authors [18–20].

In the following development, it is assumed that a finite element mesh has been constructed that will adequately model the thermal phenomena of interest (*e.g.*, thermal shock problems will require a fine mesh near a boundary while slower thermal transients will be less demanding on mesh refinement). For a given spatial discretization, a local characteristic length,  $\Delta x$ , is chosen based on element size. Typically, this characteristic length is measured normal to a boundary on which a temperature or heat flux (source) disturbance occurs. Based on the characteristic length, the local heat transfer coefficient,  $h$ , and the local thermal conductivity,  $k$ , a local element Biot number ( $Bi = h\Delta x/k$ ) can be computed. Note that for a prescribed temperature, heat flux or heat source boundary condition, the heat transfer coefficient,  $h$ , is assumed to be large leading to a large Biot number.

In order to bound the thermal gradient that will occur at the boundary in the first time step, the ratio of the temperature at a distance  $\Delta x$  (characteristic length) from the boundary to the temperature on the boundary is selected. Let this ratio be defined by  $\Theta = T(\Delta x)/T_{surface}$ . Typical values of  $\Theta$  will range from 0.10 to 0.25. With the estimated values for local Biot number and temperature ratio  $\Theta$ , a local Fourier number ( $Fo = \alpha\Delta t/\Delta x^2$ ) may be found from the charts in Figures 7.1 and 7.2. These graphs are based on an analytic solution for one-dimensional conduction with a convective boundary

condition [21]. A value of the local Fourier number and values for the characteristic length and thermal diffusivity,  $\alpha$ , then allow a time step to be computed.

As an example of the procedure outlined above, consider the transient problem described in Section 5.1. The pertinent material properties (aluminum) were given as  $\rho = 0.0978 \text{ lbm/in}^3$ ,  $C = 0.208 \text{ Btu/lbm-F}$ ,  $k = 0.002778 \text{ Btu/in-s-F}$  and  $\alpha = 0.1365 \text{ in}^2/\text{s}$ . The characteristic length is based on the average size of the elements along the radiator fin (see Figure 5.2) and is equal to  $\Delta x = 0.125 \text{ in}$ . Since the boundary condition is a specified heat flux, the Biot number is assumed large (infinite). Using a temperature ratio of  $\Theta = 0.10$ , then Figure 7.1 yields a Fourier number of  $Fo \sim 0.20$ . Thus,

$$\Delta t = \frac{(Fo)(\Delta x^2)}{(\alpha)} = \frac{(0.20)(0.125^2)}{(0.1365)} = 0.023 \text{ seconds.}$$

In the analysis of Example 1, the initial time step was set a factor of two higher than this estimate since the very early time response was not of interest.

The above procedure can also be used to estimate the overall response time for a region. In this case the length scale is a characteristic length for the entire region and the temperature ratio should be order unity. The Fourier number will then produce the approximate time interval required to reach equilibrium.

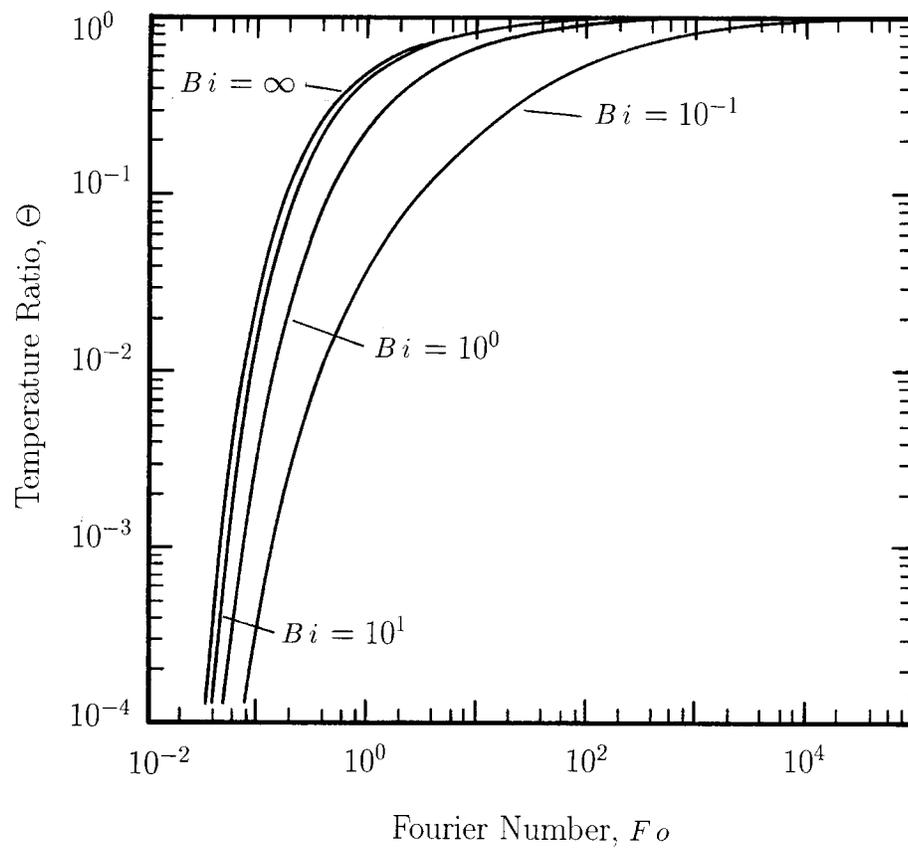


Figure 7.1: Temperature ratio versus Fourier number for smaller Biot numbers.

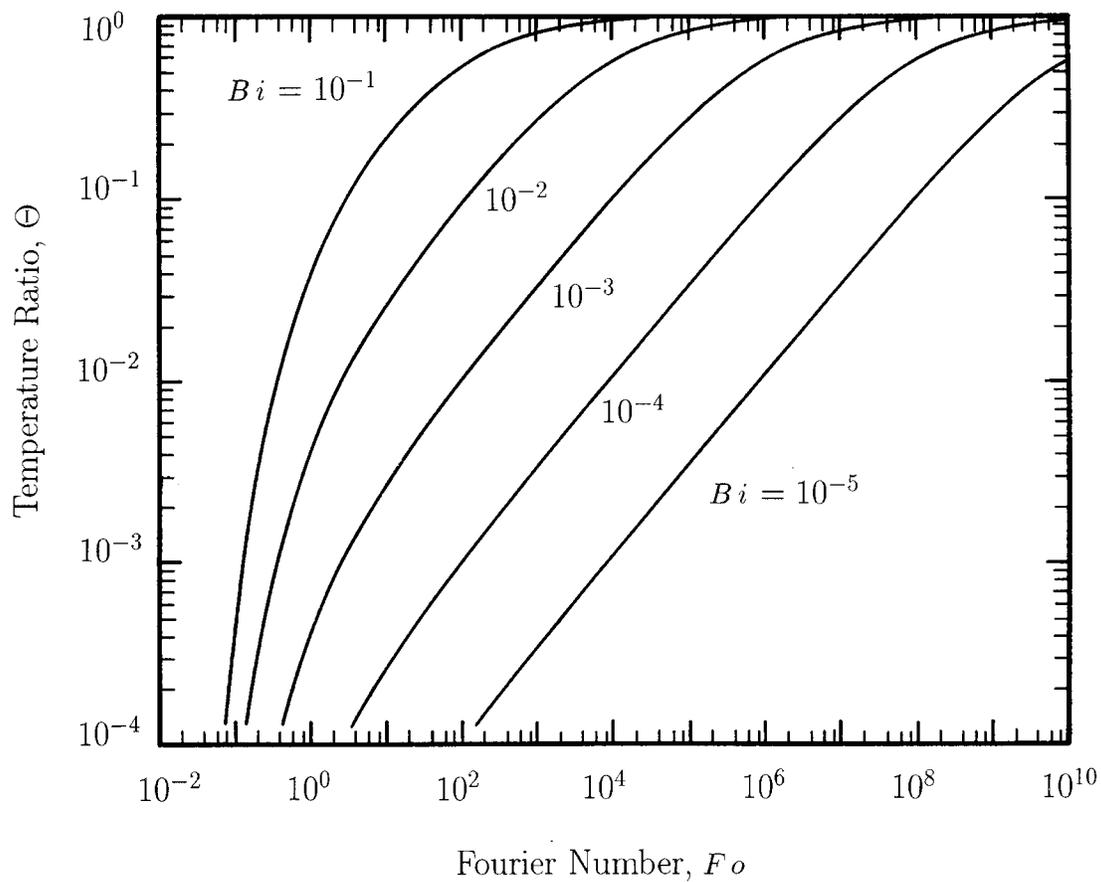


Figure 7.2: Temperature ratio versus Fourier number for larger Biot numbers.

## Appendix D - Common Block and Array Storage

The labeled common blocks and major arrays used in COYOTE are defined in the following two sections. In most cases, sizing and dimensioning parameters, option flags, pointers and problem invariant data (*e.g.*, quadrature rules) are stored in named common blocks. The common block name should give some indication of the function of the data within the block. The large data arrays used by COYOTE are allocated by the dynamic memory manager within the main program and are contained within two vectors labeled **A** and **CA**; the **CA** vector is for character data. Integer pointers partition **A** and **CA** into specific arrays that are passed through subroutine parameter lists to other parts of the code. When these arrays are used within the various subroutines they have a descriptive name that is cataloged in a subsequent section. Comment lines within the source code also help to define the use of various arrays and vectors.

### Common Blocks

The common blocks used in COYOTE are grouped below according to function.

#### File Data

```
COMMON /TAPES/  NIN,NOUT,NTP0,NTP1,NTP2,NTP3,NTP4,NTP5
COMMON /NTPDAT/ IFILES(8)
COMMON /FILDAT/ INFILE
```

<u>Variable</u>	<u>Description</u>
NIN,NOUT,NTP0..	Disk file names and ids
IFILES	Disk status for each file, 1=open, 0=closed
INFILE	Type of mesh input file, 1=EXODUS, 2=Other

#### Header Data

```
COMMON /HEADER/ PRBHED,MSHHED,CMMNT(10)
COMMON /RUNDAT/ CODNAM,VERSN,RDATE,RTIME,HRDWRE,SFTWRE,VERSNX
```

<u>Variable</u>	<u>Description</u>
PRBHED	Input title (CHARACTER*80)
MSHHED	Input title from mesh file (CHARACTER*80)
CMMNT	Input comment lines (CHARACTER*80)
CODNAM	Code name (CHARACTER*8)
VERSN	Code version number (CHARACTER*8)
RDATE	Run date (CHARACTER*8)
RTIME	Run time (CHARACTER*8)
HRDWRE	Computer hardware (CHARACTER*8)
SFTWRE	Computer operating system (CHARACTER*8)
VERSNX	Version number of EXODUS II used in COYOTE

## Problem Data

```
COMMON /PRBDAT/ IGEOM,ITMDEP,IRAD,IEXTFL,IREACT,IADAPT,IFLUX,
IPOST
```

<u>Variable</u>	<u>Description</u>
IGEOM	Problem geometry flag, 1=2D, 2=axisymmetric, 3=3D
ITMDEP	Time dependence flag, 0=steady, 1=transient
IRAD	Enclosure radiation flag, 0=omitted, 1=included
IEXTFL	External variable file flag, 0=omitted, 1=included
IREACT	Reaction kinetics flag, 0=omitted, 1=included
IADAPT	Adaptive mesh flag, 0=omitted, 1=included
IFLUX	Flux computation flag, 0=omitted, 1=included
IPOST	Post-processing file flag, 0=omitted, 1=included

## Sizing Data

```
COMMON /MATDAT/ NUMMAT
COMMON /MSHDAT/ NUMDIM,NUMEL,NUMNOD,NUMVAR,NUMDOF
COMMON /SZDAT/ MXELBK,MXNDEL,MXATEL
COMMON /BLKDAT/ NUMBLK
```

```
COMMON /SLNDAT/ NUMSLN
COMMON /CHMDAT/ MXSPEC, MXREAC, MXCHPT, NUMREL, NUMRMT, NUMSPC, IGASF
```

<u>Variable</u>	<u>Description</u>
NUMMAT	Number of materials
NUMDIM	Number of coordinate dimensions
NUMEL	Number of elements
NUMNOD	Number of nodal points
NUMVAR	Number of variables per node
NUMDOF	Number of degrees of freedom (total)
MXELBK	Maximum number of elements per element block
MXNDEL	Maximum number of nodes per element
MXATEL	Maximum number of element attributes per element
NUMBLK	Number of element blocks
NUMSLN	Number of solution blocks
MXSPEC	Maximum number of chemical species in any material
MXREAC	Maximum number of chemical reactions in any material
MXCHPT	Maximum number of chemistry points in any element
NUMREL	Number of reactive elements
NUMRMT	Number of reactive materials
NUMSPC	Number of species
IGASF	Gas fraction computation flag, 0=omitted, 1=included

## Boundary Condition Data

```
COMMON /BCDAT/ NUMNS, LNSNL, LNSDF, NUMSS, LSSEL, LSSDF, NUMNBC, NUMSBC,
*           MXNSNL, MXNSDF, MXSSEL, MXSSDF
```

<u>Variable</u>	<u>Description</u>
NUMNS	Number of nodal point sets
LNSNL	Length of concatenated node set node list
LNSDF	Length of concatenated node set distribution factor list
NUMSS	Number of element side sets
LSSEL	Length of concatenated side set element list
LSSDF	Length of concatenated side set distribution factor list

NUMNBC	Number of node boundary conditions
NUMSBC	Number of side boundary conditions
MXNSNL	Maximum number of nodes in any node set list
MXNSDF	Maximum number of distribution factors in any node set list
MXSSEL	Maximum number of elements in any side set list
MXSSDF	Maximum number of distribution factors in any side set list

## Element Data

```
COMMON /ELMDAT/ NNELM(20),NNCOR(20),NFACES(20),NNFACE(20,6),
*              NCFACE(20,6),NDFACE(20,6,8)
```

<u>Variable</u>	<u>Description</u>
NNELM	Number of nodes for each element type
NNCOR	Number of corner nodes for each element type
NFACES	Number of faces/edges for each element type
NNFACE	Number of nodes on each face/edge for each element type
NCFACE	Number of corner nodes on each face/edge for each element type
NDFACE	Local node numbers on each face/edge for each element type

## Function and Constant Data

```
COMMON /FNCDAT/ NTIMF,MXPTTF,NPTSTF,NVARF,MXPTVF,NPTSVF
COMMON /USRDAT/ NUSRR,NUSRI
```

<u>Variable</u>	<u>Description</u>
NTIMF	Number of time functions
MXPTTF	Maximum number of points in any time function
NPTSTF	Number of time function points
NVARF	Number of variable functions
MXPTVF	Maximum number of points in any variable function
NPTSVF	Number of variable function points
NUSRR	Number of user defined (real) constants

NUSRI                      Number of user defined (integer) constants

## Special Point Data

```
COMMON /SPTDAT/ ISPT, NSPT, NELSPT(50), PTS(50,3), SPTVAL(50),
*              SPTXYZ(50,3)
COMMON /TOLDAT/ TOL, EPS, STRLMT, ITERMX
COMMON /FLXDAT/ NUMFLX, IDFLUX(20)
```

<u>Variable</u>	<u>Description</u>
ISPT	Special output points flag, 0=omitted, 1=included
NSPT	Number of special points, maximum 50
NELSPT	Number of the element containing each special point
PTS	Local element coordinates $(s, t, r)$ for each special point
SPTVAL	Current value of dependent variable at each special point
SPTXYZ	Global coordinates $(x, y, z)$ for each special point
TOL	Spatial tolerance for locating special points within an element
EPS	Spatial tolerance for locating special points coincident with a node
STRLMT	Tolerance for normalized element coordinates
ITERMX	Maximum Newton iterations for convergence of special point coordinates
NUMFLX	Number of side sets for integrated flux computations
IDFLUX	Side set ids for integrated flux computations

## Radiation Enclosure Data

```
COMMON /ENCDAT/ NUMENC, MXSRF, NUMSRF, MXNDE, NUMNDE
COMMON /VFDDAT/ IVFMTH, IVFCMP, IVFRES, IVFRD, IVFWRT, IVFFMT, IVFOUT,
*              NAMVFR, NAMVFW
```

<u>Variable</u>	<u>Description</u>
-----------------	--------------------

NUMENC	Number of radiation enclosures
MXSRF	Maximum number of surfaces in any enclosure
NUMSRF	Sum of the number of surfaces in all enclosures
MXNDE	Maximum length of vector containing uncompressed list of nodes for all enclosures
NUMNDE	Sum of the number of nodes in all enclosures
IVFMTH	View factor computation method, 1=FACET, 2=Hemicube
IVFCMP	View factor storage and compression, 0=no storage, 1=no compression, 2=word run length encoding, 3=byte run length encoding, 4=LZW encoding
IVFRES	Resolution for hemicube algorithm
IVFRD	Read view factor flag, 0=no read, 1=read
IVFWRT	Write view factor flag, 0=no write, 1=write
IVFFMT	Write view factor format, 0=ASCII, 1=native machine binary, 2=XDR, 3=NetCDF
IVFOUT	Print option for FACET, 0=no print, 1=summary, 2=debug
NAMVFR	Name of view factor file, read option (CHARACTER*20)
NAMVFW	Name of view factor file, write option (CHARACTER*20)

## Input/Output Data

```
COMMON /IODAT/ IEDIT, IPRNTD, IPRNTS, NPRNTS
```

<u>Variable</u>	<u>Description</u>
IEDIT	Solution output editing flag, 0=print all elements, 1=print selected elements
IPRNTD	Print level for input data, 1=summary, 2=extended, 3=debug
IPRNTS	Print level for solution data, 0=none, 1=summary, 2=extended, 3=debug
NPRNTS	Print frequency for solution data

## Solution Algorithm Data

```
COMMON /ALGDAT/ IALGOR, IAUTO, ILUMP, IPRDCT, IFXDPT
```

```
COMMON /ITRDAT/ INEWT, IDELTA
COMMON /RSTDAT/ IRSTRT, NSTEPS, RDTIME
COMMON /PCGDAT/ NUMCHT, NUMTER, NUMQER, NNZERO
COMMON /STFSOL/ EPSMIN, EPSMAX, DTMIN, PAYSI, TCRASY
```

<u>Variable</u>	<u>Description</u>
IALGOR	Solution algorithm flag, steady - 1=Picard, 2=Newton, transient - 1=Euler, 2=trapezoid, 3=explicit
IAUTO	Automatic time step selection flag, 1=omitted, 2=included
ILUMP	Lumped capacitance matrix flag, 1=omitted, 2=included
IPRDCT	Explicit predictor flag, 1=omitted, 2=included
IFXDPT	Iterative method flag, 1=Picard, 2=Newton
INEWT	Newton iteration flag, 0=omitted, 1=included
IDELTA	Newton formulation approach, 0=T&Q, 1=dT&dQ
IRSTRT	Restart flag, 0=new solution, 1=restart from old solution
NSTEPS	Time step number at which to restart
RDTIME	Time at which to restart
NUMCHT	Number of coefficients in heat conduction equations ( for Newton's method, includes the coefficients for radiative fluxes in the heat conduction equations )
NUMTER	Number of coefficients of nodal temperatures in the enclosure radiation equations
NUMQER	Number of coefficients of radiative fluxes in the enclosure radiation equations
NNZERO	Number of nonzero entries in global matrix
EPSMIN	Convergence tolerance for stiff solver CHEMEQ
EPSMAX	Convergence tolerance for stiff solver CHEMEQ
DTMIN	Minimum time step allowed for stiff solver CHEMEQ
PAYSI	Percentage of equations treated by asymptotic method
TCRASY	Tolerance for asymptotic algorithm

## External Variable Data

```
COMMON /EXTDAT/ NUMXNV, NUMXEV, IXVEL, IXDSPL, IXVOLH, IXMAG, IXSTAT
COMMON /EXTNAM/ NAMXNV(10), NAMXEV(10)
```

<u>Variable</u>	<u>Description</u>
-----------------	--------------------

NUMXNV	Number of external nodal variables
NUMXEV	Number of external element variables
IXVEL	External velocity variable flag, 0=omitted, 1=included
IXDSPL	External displacement variable flag, 0=omitted, 1=included
IXVOLH	External volume heating variable flag, 0=omitted, 1=included
IXMAG	External magnetic variable flag, 0=omitted, 1=included
IXSTAT	External element status flag, 0=omitted, 1=included
NAMXNV	Names of external nodal variables (CHARACTER*8)
NAMXEV	Names of external element variables (CHARACTER*8)

## Mesh Modification and Movement

```
COMMON /MODDAT/ IMDMSH, IADDBK, IDELBK, IKILL
COMMON /MOVDAT/ IMVMSH, ILAGRN, IEULRN
```

<u>Variable</u>	<u>Description</u>
IMDMSH	Mesh modification flag, 0=omitted, 1=included
IADDBK	Material addition flag, 0=omitted, 1=included
IDELBK	Material deletion flag, 0=omitted, 1=included
IKILL	Element death flag, 0=omitted, 1=included
IMVMSH	Mesh movement flag, 0=omitted, 1=included
ILAGRAN	Lagrangian velocity flag, 0=omitted, 1=included
IEULRN	Eulerian velocity flag, 0=omitted, 1=included

## Norm Data

```
COMMON /MAXVAR/ TMAX, IFNDMX
COMMON /NORMS/ ANRMT, BNRMT, TDIFF
```

<u>Variable</u>	<u>Description</u>
TMAX	Maximum temperature in the current solution vector
IFNDMX	Flag for computation/input of norm temperature

ANRMT	Norm on temperature change between iterations/time steps
BNRMT	Norm on temperature integration error
TDIFF	Maximum temperature change between steps or iterations

## Postprocessing Data

```
COMMON /EXODAT/ NVARNP,NVAREL,NVARGL,NUMEV,NUMCV
COMMON /EXONAM/ NAMECO(3),NAMEEL(20),NAMENV(10),NAMEEV(5),NAMECV(45),
*           NAMEGV(25)
COMMON /EXOUT/ IEXOPT,NWFREQ,DELTEX,NTIMEX,TIMEX(50)
```

<u>Variable</u>	<u>Description</u>
NVARNP	Number of nodal variables to be written on EXODUS II file
NVAREL	Number of element variables to be written on EXODUS II file
NVARGL	Number of global variables to be written on EXODUS II file
NUMEV	Number of nonchemistry element variables
NUMCV	Number of chemistry element variables
NAMECO	Names for coordinates (CHARACTER*8)
NAMEEL	Names for elements (CHARACTER*8)
NAMENV	Names for nodal variables (CHARACTER*8)
NAMEEV	Names for nonchemistry element variables (CHARACTER*8)
NAMECV	Names for chemistry element variables (CHARACTER*8)
NAMEGV	Names for global variables (CHARACTER*8)
IEXOPT	EXODUS II output option flag, 1=write output at every NWFREQ steps, 2=write output at every DELTEX time interval, 3=write output at every time specified in TIMEX
NWFREQ	Frequency for writing to EXODUS II output file
DELTEX	Time step for writing EXODUS II output file
NTIMEX	Number of output times stored in TIMEX
TIMEX	Specified times for writing EXODUS II output file

## Constants

```
COMMON /CNSTNT/ SMA,GASR
```

<u>Variable</u>	<u>Description</u>
SIGMA	Stefan-Boltzman constant
GASR	Gas constant

## Shape Function Data (2D)

COMMON /TRI3/	TR3(3,7), TR3G(3,2,7), TR3F(3,3), TR3FG(3,2,3), TR3WT(7)
COMMON /TRI6/	TR6(6,7), TR6G(6,2,7), TR6F(6,3), TR6FG(6,2,3), TR6WT(7)
COMMON /QUAD4/	QD4(4,9), QD4G(4,2,9), QD4F(4,4), QD4FG(4,2,4), QD4WT(9)
COMMON /QUAD8/	QD8(8,9), QD8G(8,2,9), QD8F(8,4), QD8FG(8,2,4), QD8WT(9)
COMMON /EDGE2/	ED2(2,3), ED2G(2,3), ED2WT(3)
COMMON /EDGE3/	ED3(3,3), ED3G(3,3), ED3WT(3)

<u>Variable</u>	<u>Description</u>
TR3, TR6 QD4, QD8	Two-dimensional, triangular and quadrilateral, element shape functions evaluated at the quadrature points. The first index is the shape function (node) number; the second index is the quadrature point number.
TR3G, TR6G QD4G, QD8G	Two-dimensional, triangular and quadrilateral, element shape function derivatives evaluated at the quadrature points. The first index is the shape function (node) number, the second index is the local spatial direction for the derivative and the third index is the quadrature point number.
TR3F, TR6F QD4F, QD8F	Two-dimensional, triangular and quadrilateral, element shape functions evaluated at the flux evaluation points. The first index is the shape function (node) number; the second index is the flux evaluation point number.
TR3FG, TR6FG QD4FG, QD8FG	Two-dimensional, triangular and quadrilateral, element shape function derivatives evaluated at the flux evaluation points. The first index is the shape function (node) number, the second index is the local spatial direction for the derivative and the third index is the flux evaluation point number.
TR3WT, TR6WT QD4WT, QD8WT	Quadrature weights for each element at each integration point.
ED2, ED3	One-dimensional, element edge shape functions evaluated at the quadrature points. The first index is the shape function (node)

ED2G,ED3G	number; the second index is the quadrature point number. One-dimensional, element edge shape function derivatives evaluated at the quadrature points. The first index is the shape function (node) number and the second index is the quadrature point number.
ED2WT,ED3WT	Quadrature weights for each element edge at each integration point.

### Shape Function Data (3D)

```

COMMON /TETR4/  TT4(4,5),TT4G(4,3,5),TT4F(4,4),TT4FG(4,3,4),TT4WT(5)
COMMON /TETR10/ TT10(10,5),TT10G(10,3,5),TT10F(10,4),TT10FG(10,3,4),
*              TT10WT(5)
COMMON /WEDG6/  WD6(6,21),WD6G(6,3,21),WD6F(6,6),WD6FG(6,3,6),WD6WT(21)
COMMON /WEDG15/ WD15(15,21),WD15G(15,3,21),WD15F(15,6),WD15FG(15,3,6),
*              WD15WT(21)
COMMON /HEX8/   HX8(8,27),HX8G(8,3,27),HX8F(8,8),HX8FG(8,3,8),HX8WT(27)
COMMON /HEX20/  HX20(20,27),HX20G(20,3,27),HX20F(20,8),HX20FG(20,3,8),
*              HX20WT(27)
COMMON /FACE3/  FC3(3,7),FC3G(3,2,7),FC3WT(7)
COMMON /FACE6/  FC6(6,7),FC6G(6,2,7),FC6WT(7)
COMMON /FACE4/  FC4(4,9),FC4G(4,2,9),FC4WT(9)
COMMON /FACE8/  FC8(8,9),FC8G(8,2,9),FC8WT(9)

```

<u>Variable</u>	<u>Description</u>
TT4,TT10 WD6,WD15 HX8,HX20	Three-dimensional, tetrahedral, wedge and hexahedral, element shape functions evaluated at the quadrature points. The first index is the shape function (node) number; the second index is the quadrature point number.
TT4G,TT10G WD6G,WD15G HX8G,HX20G	Three-dimensional, tetrahedral, wedge and hexahedral, element shape function derivatives evaluated at the quadrature points. The first index is the shape function (node) number, the second index is the local spatial direction for the derivative and the third index is the quadrature point number.
TT4F,TT10F WD6F,WD15F	Three-dimensional, tetrahedral, wedge and hexahedral, element shape functions evaluated at the flux evaluation points.

HX8F, HX20F	The first index is the shape function (node) number; the second index is the flux evaluation point number.
TT4FG, TT10FG WD6FG, WD15FG HX8FG, HX20FG	Three-dimensional, tetrahedral, wedge and hexahedral, element shape function derivatives evaluated at the flux evaluation points. The first index is the shape function (node) number, the second index is the local spatial direction for the derivative and the third index is the flux evaluation point number.
TT4WT, TT10W WD6WT, WD15WT HX8WT, HX20WT	Quadrature weights for each element at each integration point.
FC3, FC6 FC4, FC8	Two-dimensional, triangular and quadrilateral element face shape functions evaluated at the quadrature points. The first index is the shape function (node) number; the second index is the quadrature point number.
FC3G, FC6G FC4G, FC8G	Two-dimensional, triangular and quadrilateral element face shape function derivatives evaluated at the quadrature points. The first index is the shape function (node) number, the second index is the local (surface) spatial direction for the derivative and the third index is the quadrature point number.
FC3WT, FC6WT FC4WT, FC8WT	Quadrature weights for each element face at each integration point.

## Shape Function Data (Specialty Elements)

```

COMMON /BAR2/ BR2(2,3), BR2G(2,3), BR2F(2,2), BR2FG(2,2), BR2WT(3)
COMMON /BAR3/ BR3(3,3), BR3G(3,3), BR3F(3,2), BR3FG(3,2), BR3WT(3)
COMMON /SHELL3/ SHL3(3,14), SHL3G(3,3,14), RSHL3G(3,3,14),
* SHL3F(3,3), SHL3FG(3,3,3), SHL3WT(14)
COMMON /SHELL6/ SHL6(6,14), SHL6G(6,3,14), RSHL6G(6,3,14),
* SHL6F(6,3), SHL6FG(6,3,3), SHL6WT(14)
COMMON /SHELL4/ SHL4(4,18), SHL4G(4,3,18), RSHL4G(4,3,18),
* SHL4F(4,4), SHL4FG(4,3,4), SHL4WT(18)
COMMON /SHELL8/ SHL8(8,18), SHL8G(8,3,18), RSHL8G(8,3,18),
* SHL8F(8,4), SHL8FG(8,3,4), SHL8WT(18)

```

<u>Variable</u>	<u>Description</u>
-----------------	--------------------

BR2, BR3	Three-dimensional, bar and shell element shape functions evaluated at the quadrature points. The first index is the shape function (node) number; the second index is the quadrature point number.
SHL3, SHL6	
SHL4, SHL8	
BR2G, BR3G	Three-dimensional, bar and shell element shape function derivatives evaluated at the quadrature points. The first index is the shape function (node) number, the second index is the local spatial direction for the derivative and the third index is the quadrature point number.
SHL3G, SHL6G	
SHL4G, SHL8G	
RSHL3G, RSHL6G	Three-dimensional, bar and shell product of radius and element shape functions evaluated at the quadrature points. The first index is the shape function (node) number, the second index is the local spatial direction for the derivative and the third index is the quadrature point number.
RSHL3G, RSHL6G	
BR2F, BR3F	Three-dimensional, bar and shell element shape functions evaluated at the flux evaluation points. The first index is the shape function (node) number; the second index is the flux evaluation point number.
SHL3F, SHL6F	
SHL4F, SHL8F	
BR2FG, BR3FG	Three-dimensional, bar and shell element shape function derivatives evaluated at the flux evaluation points. The first index is the shape function (node) number, the second index is the local spatial direction for the derivative and the third index is the flux evaluation point number.
SHL3FG, SHL6FG	
SHL4FG, SHL8FG	
BR2WT, BR3WT	Quadrature weights for each element at each integration point.
SHL3WT, SHL6WT	
SHL4WT, SHL8WT	

## Quadrature Data

```

COMMON /INTDAT/ NIRULE(20),NFRULE(20)
COMMON /GAUSS1/ GSPT1(3,1,3),GSWT1(3,3),NGSPT1(3)
COMMON /GAUSS2/ GSPT2(9,2,3),GSWT2(9,3),NGSPT2(3)
COMMON /GAUSS3/ GSPT3(27,3,5),GSWT3(27,5),NGSPT3(5)
COMMON /GAUSS4/ GSPT4(7,3,4),GSWT4(7,4),NGSPT4(4)
COMMON /GAUSS5/ GSPT5(5,4,3),GSWT5(5,3),NGSPT5(3)
COMMON /GAUSS6/ GSPT6(21,4,4),GSWT6(21,4),NGSPT6(4)
COMMON /GAUSS7/ GSPT7(14,4,4),GSWT7(14,4),NGSPT7(4)
COMMON /GAUSS8/ GSPT8(18,3,5),GSWT8(18,5),NGSPT8(5)

```

```
COMMON /CENTER/ STRPT(20,3)
COMMON /TRNSFM/ EXTBR(2,2),EXTTR(3,3),EXTQD(4,4),EXTTT(4,4),
*          EXTWD(6,6),EXTHX(8,8),EXTSHT(3,3),EXTSHQ(4,4)
```

<u>Variable</u>	<u>Description</u>
NIRULE	Integer indicating integration rule for each element type
NFRULE	Integer indicating flux computation rule for each element type
GSPTn	Quadrature points for each quadrature rule and each element type. The first index is the number of the quadrature point, the second index is the number of the local coordinate direction and the third index is the integration rule. The integer <b>n</b> in the array name separates the data according to element geometry. Arrays with <b>n=1</b> contain quadrature data for one-dimensional, line elements, <b>n=2</b> is for two-dimensional quadrilateral regions, <b>n=3</b> is for three-dimensional hexahedrons, <b>n=4</b> is for two-dimensional triangular areas, <b>n=5</b> is for three-dimensional tetrahedrons, <b>n=6</b> is for three-dimensional wedges, <b>n=7</b> is for triangular shells and <b>n=8</b> is for quadrilateral shells.
GSWTn	Quadrature weights for each integration point and each quadrature rule. The first index specifies the number of the quadrature point and the second index is the integration rule.
NGSPTn	Number of quadrature points for each integration rule.
STRPT	Local element coordinates for element centroid.
EXTmmm	Flux point to node point linear extrapolation operators for each element type.

## Free Field Input Data

```
COMMON /INDATR/ RDATA(50)
COMMON /INDATI/ IDATA(50)
COMMON /INDATC/ CDATA(50)
COMMON /INDATK/ KDATA(50)
```

<u>Variable</u>	<u>Description</u>
RDATA	Real data from the free field reader

IDATA	Integer data from the free field reader
CDATA	Character data from the free field reader (CHARACTER*20)
KDATA	Data type flags from the free field reader

## Array Storage

The major arrays used in COYOTE are grouped below according to function.

### Mesh Data

```

DIMENSION IDBLK(NUMBLK)
DIMENSION X(NUMNOD), Y(NUMNOD), Z(NUMNOD)
DIMENSION ICON(MXNDEL, NUMEL)
DIMENSION LSTKND(NUMEL), LSTOUT(NUMEL), LSTDEL(NUMEL)
DIMENSION ELATRB(MXATEL, NUMEL)
DIMENSION DATBLK(10, NUMBLK), IPTBLK(20, NUMBLK)
DIMENSION ICONBK(MXNDEL*MXELBK), ATRBBK(MXATEL*MXELBK)
DIMENSION BLKMAT(NUMBLK), BLKVAR(NUMBLK)

```

<u>Array</u>	<u>Description</u>
IDBLK	Block id numbers
X, Y, Z	Nodal coordinates
ICON	Element connectivity
LSTKND	Encoded list of element type and material (1000*mat number + element type)
LSTOUT	Output flags for elements, 0=no print, 1=print
LSTDEL	Activation flag for elements, 0=inactive, 1=active
ELATRB	Element attributes
DATBLK	Element block data DATBLK(1, NUMBLK) - UX velocity for block DATBLK(2, NUMBLK) - UY velocity for block DATBLK(3, NUMBLK) - UZ velocity for block DATBLK(4, NUMBLK) - Variable value for element death DATBLK(5, NUMBLK) - Time for block deletion DATBLK(6, NUMBLK) - Time for block addition

	DATBLK(7,NUMBLK) - Unused
	DATBLK(8,NUMBLK) - Unused
	DATBLK(9,NUMBLK) - Unused
	DATBLK(10,NUMBLK) - Unused
IPTBLK	Element block pointers and flags
	IPTBLK(1,NUMBLK) - Number of elements in block
	IPTBLK(2,NUMBLK) - Element type
	IPTBLK(3,NUMBLK) - Number of element attributes
	LSTBLK(4,NUMBLK) - Number of integration points
	IPTBLK(5,NUMBLK) - Material number
	IPTBLK(6,NUMBLK) - First element in block
	IPTBLK(7,NUMBLK) - Last element in block
	IPTBLK(8,NUMBLK) - Element death flag
	IPTBLK(9,NUMBLK) - Block deletion flag
	IPTBLK(10,NUMBLK) - Block addition flag
	IPTBLK(11,NUMBLK) - Element death variable
	IPTBLK(12,NUMBLK) - Element death mode
	IPTBLK(13,NUMBLK) - Unused
	IPTBLK(14,NUMBLK) - Number of flux points
	IPTBLK(15,NUMBLK) - External velocity flag
	IPTBLK(16,NUMBLK) - Velocity type
	IPTBLK(17,NUMBLK) - Velocity user subroutine flag
	IPTBLK(18,NUMBLK) - Function id for UX
	IPTBLK(19,NUMBLK) - Function id for UY
	IPTBLK(20,NUMBLK) - Function id for UZ
ICONBK	Element connectivity for a block
ATRBK	Element attributes for a block
BLKMAT	Material name for element block (CHARACTER*20)
BLKVAR	Variable name for death option (CHARACTER*20)

## Boundary Condition Data

```

DIMENSION IDNS(NUMNS), NNNS(NUMNS), NDFNS(NUMNS)
DIMENSION IPNNS(NUMNS), IPDFNS(NUMNS)
DIMENSION LSNNS(LNSNL), DFNS(LNSDF)
DIMENSION IDSS(NUMSS), NESS(NUMSS), NDFSS(NUMSS)
DIMENSION IPESS(NUMSS), IPDFSS(NUMSS)
DIMENSION LSESS(LSSEL), LSSSS(LSSEL), DFSS(LSSDF)

```



	3=radiation, 4=enclosure radiation, 5= contact surface
KPTSS(3,NUMSS)	- User subroutine flag for value 1
KPTSS(4,NUMSS)	- User subroutine flag for value 2
KPTSS(5,NUMSS)	- Function id for value 1, +id=variable function, -id=time function
KPTSS(6,NUMSS)	- Function id for value 2, +id=variable function, -id=time function
KPTSS(7,NUMSS)	- Constant bc flag, value 1
KPTSS(8,NUMSS)	- Constant bc flag, value 2
KPTSS(9,NUMSS)	- Unused
KPTSS(10,NUMSS)	- Enclosure number
KPTELM	Encoded side set list for each element

## Material Data

DIMENSION DATMAT(21,NUMMAT)  
 DIMENSION PTMAT(21,NUMMAT)  
 DIMENSION IPTMAT(25,NUMMAT)  
 DIMENSION MATNAM(NUMMAT)

<u>Array</u>	<u>Description</u>
DATMAT	Material property values
	DATMAT(1,NUMMAT) - Density
	DATMAT(2,NUMMAT) - Specific heat
	DATMAT(3,NUMMAT) - Conductivity (xx component)
	DATMAT(4,NUMMAT) - Conductivity (yy component)
	DATMAT(5,NUMMAT) - Conductivity (zz component)
	DATMAT(6,NUMMAT) - Volume heating
	DATMAT(7,NUMMAT) - Emissivity
	DATMAT(8,NUMMAT) - Conductivity tensor orientation ( $x\hat{x}$ )
	DATMAT(9,NUMMAT) - Conductivity tensor orientation ( $y\hat{x}$ )
	DATMAT(10,NUMMAT) - Conductivity tensor orientation ( $z\hat{x}$ )
	DATMAT(11,NUMMAT) - Conductivity tensor orientation ( $x\hat{y}$ )
	DATMAT(12,NUMMAT) - Conductivity tensor orientation ( $y\hat{y}$ )
	DATMAT(13,NUMMAT) - Conductivity tensor orientation ( $z\hat{y}$ )
	DATMAT(14,NUMMAT) - Latent heat

---

	DATMAT(15,NUMMAT) - Solidus temperature
	DATMAT(16,NUMMAT) - Liquidus temperature
	DATMAT(17,NUMMAT) - Enthalpy (Unused)
	DATMAT(18,NUMMAT) - Unused
	DATMAT(19,NUMMAT) - Unused
	DATMAT(20,NUMMAT) - Chemistry activation temperature
	DATMAT(21,NUMMAT) - Initial temperature
PTMAT	Material property pointers (CHARACTER*1)
	PTMAT(1,NUMMAT) - Subroutine and function flags for property 'V'=variable function, 'T'=time function 'C'=constant value, 'U'=user subroutine
	.
	PTMAT(16,NUMMAT) - Subroutine and function flag for property
	PTMAT(17,NUMMAT) - Unused
	PTMAT(18,NUMMAT) - Unused
	PTMAT(19,NUMMAT) - Unused
	PTMAT(20,NUMMAT) - Unused
	PTMAT(21,NUMMAT) - Material model, 'I'= isotropic, 'O'=orthotropic
IPTMAT	Material property function flags
	IPTMAT(1,NUMMAT) - Function id for property
	.
	.
	.
	IPTMAT(16,NUMMAT) - Function id for property
	IPTMAT(17,NUMMAT) - Unused
	IPTMAT(18,NUMMAT) - Unused
	IPTMAT(19,NUMMAT) - Unused
	IPTMAT(20,NUMMAT) - Unused
	IPTMAT(21,NUMMAT) - Phase change model, 0=none, 1=specific heat, 2=enthalpy gradient, 3=enthalpy time derivative
	IPTMAT(22,NUMMAT) - Reactive material number
	IPTMAT(23,NUMMAT) - Number of species
	IPTMAT(24,NUMMAT) - Number of reactions
	IPTMAT(25,NUMMAT) - Unused
MATNAM	Material property names (CHARACTER*20)

## Chemistry Data

```

DIMENSION IPTREL(NUMEL), IPTRMT(5,NUMMAT)
DIMENSION AMUSP(MXSPEC,MXREAC,NUMRMT), ANUSP(MXSPEC,MXREAC,NUMRMT)
DIMENSION STERIC(MXREAC,NUMRMT), PREX(MXREAC,NUMRMT)
DIMENSION AENRGY(MXREAC,NUMRMT), ENRGYR(MXREAC,NUMRMT)
DIMENSION SPEC0(MXSPEC,NUMRMT), SPECMN(MXSPEC,NUMRMT)
DIMENSION SPCNAM(MXSPEC,NUMRMT)
DIMENSION GSPHSE(MXSPEC,NUMRMT), CNDFRC(NUMRMT)

```

<u>Array</u>	<u>Description</u>
IPTRMEL	List of reactive elements
IPTRMT	Reactive material pointers IPTRMT(1,NUMMAT) - Reactive material number IPTRMT(2,NUMMAT) - Number of species IPTRMT(3,NUMMAT) - Number of reactions IPTRMT(4,NUMMAT) - Unused IPTRMT(5,NUMMAT) - Unused
AMUSP	Concentration exponents for each material
ANUSP	Stoichiometric coefficients for each material
STERIC	Steric factors for each material
PREX	Pre-exponential factors for each material
AENRGY	Activation energies for each material
ENRGYR	Endo/exothermic energy release for each material
SPEC0	Initial species concentrations for each material
SPECMN	Minimum species concentrations for each material
SPCNAM	Species names for each material (CHARACTER*20)
GSPHSE	Gas phase flag for each species for each material
CNDFRC	Condensed fraction for each material

## Function and Constant Data

```

DIMENSION DATTFN(2,NPTSTF), IPTTFN(3,NTIMF), TFNVAL(NTIMF)
DIMENSION DATVFN(2,NPTSUF), IPTVFN(3,NVARF)
DIMENSION RCONST(NUSRR), ICONST(NUSRI)

```

<u>Array</u>	<u>Description</u>
DATTFN	Time function data DATTFN(1,NPTSTF) - Time DATTFN(2,NPTSTF) - Function value
IPTTFN	Time function pointers IPTTFN(1,NTIMF) - Function id IPTTFN(2,NTIMF) - Number of points in time function IPTTFN(3,NTIMF) - Pointer to first function point in DATTFN
TFNVAL	Time function values at each time step
DATVFN	Variable function data DATVFN(1,NPTSVF) - Variable value (temperature) DATVFN(2,NPTSVF) - Function value
IPTVFN	Variable function pointers IPTVFN(1,NVARF) - Function id IPTVFN(2,NVARF) - Number of points in time function IPTVFN(3,NVARF) - Pointer to first function point in DATVFN
RCNST	User defined constants (real)
ICONST	User defined constants (integer)

## Radiation Enclosure Data

```

DIMENSION DATENC(3,NUMENC) , IPTENC(12,NUMENC)
DIMENSION ICONSF(5,MXSRF)
DIMENSION LSTSRF(MXSRF,NUMENC)
DIMENSION QSURF(MXSRF,NUMENC) , TSURF(MXSRF,NUMENC) , ESURF(MXSRF,NUMENC) ,
VF(MXSRF)

```

<u>Array</u>	<u>Description</u>
DATENC	Enclosure radiation data DATENC(1,NUMENC) - $Area_{\infty}$ , partial enclosure DATENC(2,NUMENC) - $T_{\infty}$ , partial enclosure DATENC(3,NUMENC) - $\epsilon_{\infty}$ , partial enclosure
IPTENC	Enclosure radiation pointers IPTENC(1,NUMENC) - Number of surfaces in enclosure IPTENC(2,NUMENC) - Partial enclosure flag, 0=full enclosure, 1=partial enclosure

	IPTENC(3,NUMENC) - Blocking surface flag, 0=no blocking surfaces, 1=blocking surfaces
	IPTENC(4,NUMENC) - Smoothing flag, 0=no smoothing, 1=least squares smoothing
	IPTENC(5,NUMENC) - Maximum surface subdivisions
	IPTENC(6,NUMENC) - Number of rotational intervals
	IPTENC(7,NUMENC) - Number of x-grid divisions
	IPTENC(8,NUMENC) - Number of y-grid divisions
	IPTENC(9,NUMENC) - Number of z-grid divisions
	IPTENC(10,NUMENC) - Unused
	IPTENC(11,NUMENC) - Time function id for $T_\infty$ (not implemented)
	IPTENC(12,NUMENC) - Variable function id for $\epsilon_\infty$ (not implemented)
ICONSF	Connectivity for enclosure surfaces
LSTSRF	Encoded element and face numbers for enclosure surfaces
QSURF	Heat flux for enclosure surfaces
TSURF	Temperature for enclosure surfaces
ESURF	Emissivity for enclosure surfaces
VF	Row of view factors for enclosure surfaces

## Solution Control Data

DIMENSION DATSLN(20,NUMSLN)  
 DIMENSION IPTSLN(20,NUMSLN)

<u>Array</u>	<u>Description</u>
DATSLN	Solution control data
	DATSLN(1,NUMSLN) - Initial solution time
	DATSLN(2,NUMSLN) - Final solution time
	DATSLN(3,NUMSLN) - Initial time step
	DATSLN(4,NUMSLN) - Minimum time step
	DATSLN(5,NUMSLN) - Maximum time step
	DATSLN(6,NUMSLN) - Maximum temperature change per time step
	DATSLN(7,NUMSLN) - Convergence tolerance on temperature
	DATSLN(8,NUMSLN) - Integration tolerance
	DATSLN(9,NUMSLN) - Relaxation factor
	DATSLN(10,NUMSLN) - Chemistry time step multiplier

DATSLN(11,NUMSLN) - Convergence tolerance on radiation  
 DATSLN(12,NUMSLN) - Time increment for view factor reform  
 DATSLN(13,NUMSLN) - Explicit time step scale factor  
 DATSLN(14,NUMSLN) -  $L_2$  norm for iterative solver  
 DATSLN(15,NUMSLN) - Residual norm for iterative solver  
 DATSLN(16,NUMSLN) - Absolute temperature limit  
 DATSLN(17,NUMSLN) - Temperature for norm  
 DATSLN(18,NUMSLN) - Unused  
 DATSLN(19,NUMSLN) - Unused  
 DATSLN(20,NUMSLN) - Unused

IPTSLN      Solution control pointers and flags  
 IPTSLN(1,NUMSLN) - Time dependence flag,  
                   1=steady, 2=transient  
 IPTSLN(2,NUMSLN) - Iterative method flag,  
                   1=Picard, 2=Newton  
 IPTSLN(3,NUMSLN) - Maximum number of iterations  
 IPTSLN(4,NUMSLN) - Time integration method flag,  
                   1=Euler, 2=trapezoid, 3=explicit  
 IPTSLN(5,NUMSLN) - Capacitance matrix flag,  
                   1=consistent, 2=lumped  
 IPTSLN(6,NUMSLN) - Time step option flag, 1=fixed step,  
                   2=autostep  
 IPTSLN(7,NUMSLN) - Maximum number of time steps  
 IPTSLN(8,NUMSLN) - Predictor flag, 1=no predictor ,  
                   2=predictor  
 IPTSLN(9,NUMSLN) - Unused  
 IPTSLN(10,NUMSLN) - Radiation solution method flag,  
                   1=Gauss, 2=Progressive  
 IPTSLN(11,NUMSLN) - Maximum iterations for radiation  
 IPTSLN(12,NUMSLN) - View factor reform flag, 0=no reform,  
                   1=steps reform, 2=time reform  
 IPTSLN(13,NUMSLN) - View factor reform, nsteps  
 IPTSLN(14,NUMSLN) - Type of matrix solver, 0=CG,  
                   1=GMRES, 2=CGS, 3=QMR  
 IPTSLN(15,NUMSLN) - Krylov subspace dimension  
 IPTSLN(16,NUMSLN) - Type of preconditioner, 0=None,  
                   1=Jacobi, 3=polynomial, 4=ILU, 9=IC  
 IPTSLN(17,NUMSLN) - Order of polynomial preconditioner  
 IPTSLN(18,NUMSLN) - Maximum number of solver iterations  
 IPTSLN(19,NUMSLN) - Print flag, 1=summary, 2=detailed,

IPTSLN(20,NUMSLN) - Print frequency

## Presolution and Solution Data

```

DIMENSION IPTE2N(NUMEL+NUMSRF+1)
DIMENSION ICNE2N(MXNDEL*NUMEL+NUMSRF+NUMNDE)
DIMENSION IPTN2E(NUMNOD+NUMSRF+1)
DIMENSION ICNN2E(MXNDEL*NUMEL+NUMSRF+NUMNDE)
DIMENSION IPTN2N(NUMNOD+NUMSRF+1)
DIMENSION NODLST(MXNDE), IJK(MEMCHT+NUMNOD+MEMTER+MEMQER+2)
DIMENSION TN(NUMNOD), TPNP1(NUMNOD), TDOT(NUMNOD), SCRATCH(NUMNOD)
DIMENSION BNDVAL(MXNNPS)
DIMENSION BIGK(NNZERO), BIGF(NUMNOD)
DIMENSION DSPLX(NUMNOD), DSPLY(NUMNOD), DSPLZ(NUMNOD)
DIMENSION XZERO(NUMNOD), YZERO(NUMNOD), ZZERO(NUMNOD)
DIMENSION VELX(NUMNOD), VELY(NUMNOD), VELZ(NUMNOD)
DIMENSION VHEAT(NUMNOD), AMAG(NUMNOD)
DIMENSION QX(NUMNOD), QY(NUMNOD), QZ(NUMNOD), NODFLX(NUMNOD)

```

<u>Array</u>	<u>Description</u>
IPTE2N	Pointer for element to node connectivity
ICNE2N	Vector containing element to node connectivity
IPTN2E	Pointer for node to element connectivity
ICNN2E	Vector containing node to element connectivity
IPTN2N	Pointer for node to node connectivity
NODLST	Temporary list of node numbers for an enclosure
IJK	Pointers for sparse matrix storage
TN	Solution vector (current nodal temperatures)
TPNP1	Predicted solution vector
TDOT	Temperature rate vector
SCRATCH	Scratch solution vector
BNDVAL	Boundary conditions for nodal points
BIGK	Global matrix (sparse matrix format)
BIGF	Global force vector
DSPLX	Nodal point displacements, x component
DSPLY	Nodal point displacements, y component
DSPLZ	Nodal point displacements, z component

XZERO	Reference nodal point coordinates, x component
YZERO	Reference nodal point coordinates, y component
ZZERO	Reference nodal point coordinates, z component
VELX	Nodal point velocities, x component
VELY	Nodal point velocities, y component
VELZ	Nodal point velocities, z component
VHEAT	Nodal point volume heating (external)
AMAG	Nodal point magnetic field (external)
QX	Nodal point heat flux component
QY	Nodal point heat flux component
QZ	Nodal point heat flux component
NODFLX	Number of fluxes per node

## Chemistry Solution Data

```

DIMENSION SPEC(MXSPEC,MXCHPT,NUMREL)
DIMENSION QCHEM(MXCHPT,NUMREL)
DIMENSION SPECGP(MXSPEC,MXCHPT), SPECEL(MXELBK)
DIMENSION CHMWRK(IWRK)
DIMENSION GSFR(MXCHPT,NUMREL), GSFREL(NUMREL)

```

<u>Array</u>	<u>Description</u>
SPEC	Chemical species solution array
QCHEM	Chemical heat source array
SPECGP	Chemical species solution at element integration points
SPECEL	Chemical species solution at element centroids
CHMWRK	Chemistry work array for CHMSOL
GSFR	Gas fraction at element integration points
GSFREL	Gas fraction at element centroids

## Postprocessing Data

```

DIMENSION GBLV(NVARGL)
DIMENSION IPPNOD(10), IPPELM(5), IPPGLB(25)

```

<u>Array</u>	<u>Description</u>
GLBLV	Global variable vector
IPPNOD	Output flags for nodal variables (0=omitted, 1=output) IPPNOD(1) - Flag for nodal displacement, x component IPPNOD(2) - Flag for nodal displacement, y component IPPNOD(3) - Flag for nodal displacement, z component IPPNOD(4) - Flag for temperature IPPNOD(5) - Flag for temperature rate IPPNOD(6) - Flag for heat flux, x component IPPNOD(7) - Flag for heat flux, y component IPPNOD(8) - Flag for heat flux, z component IPPNOD(9) - Flag for heat flow function (2D) IPPNOD(10) - Unused
IPPELM	Output flags for nonchemistry element variables (0=omitted, 1=output) IPPELM(1,NUMBLK) - Flag for element status variable IPPELM(2,NUMBLK) - Unused IPPELM(3,NUMBLK) - Unused IPPELM(4,NUMBLK) - Unused IPPELM(5,NUMBLK) - Unused
IPPGLB	Output flags for global variables (0=omitted, 1=output) IPPGLB(1) - Flag for timestep IPPGLB(2) - Flag for number of cg iterations IPPGLB(3) - Unused IPPGLB(4) - Unused IPPGLB(5) - Unused IPPGLB(6) - Flag for special output point 1 IPPGLB(7) - Flag for special output point 2 IPPGLB(8) - Flag for special output point 3 . . . IPPGLB(13) - Flag for special output point 8 IPPGLB(14) - Flag for special output point 9 IPPGLB(15) - Flag for special output point 10 IPPGLB(16) - Flag for integrated flux surface 1 IPPGLB(17) - Flag for integrated flux surface 2 IPPGLB(18) - Flag for integrated flux surface 3 . .

IPPGLB(23) - Flag for integrated flux surface 8  
IPPGLB(24) - Flag for integrated flux surface 9  
IPPGLB(25) - Flag for integrated flux surface 10



## Appendix E - External Mesh Generator File Contents

The file structure recognized by COYOTE for reading data from an external mesh generator is based on the EXODUS II [6] standard. The contents of the file are summarized below in the order in which data is normally read by COYOTE. Note that the EXODUS II file is a random access file and the actual reading and writing of data records is accomplished through a subroutine interface that is documented in [6]. File opening and closing is also handled through a subroutine interface. COYOTE reads and writes EXODUS II mesh data in subroutines OPNFIL, EXOINQ, EXOSIZ and EXORD.

```
C
C   Heading and Problem Sizing Parameters
C
C   MSHHED - Problem title from mesh generator (CHARACTER*80)
C   NUMNOD - Number of nodes
C   NUMDIM - Number of coordinates per node
C   NUMEL  - Total number of elements
C   NUMBLK - Number of element blocks
C   NUMNS  - Number of nodal point sets
C   NUMSS  - Number of element side sets
C
C   QA Records
C
C   QAREC(1,NQAREC) - Code name (CHARACTER*8)
C   QAREC(1,NQAREC) - Version (CHARACTER*8)
C   QAREC(1,NQAREC) - Run time (CHARACTER*8)
C   QAREC(1,NQAREC) - Run date (CHARACTER*8)
C
C   Nodal Point Coordinates
C
C   X(NUMNOD) - X coordinate
C   Y(NUMNOD) - Y coordinate
C   Z(NUMNOD) - Z coordinate
C   NAMECO(NUMDIM) - Names of coordinates
C
C   Element Block Parameters
C
C   IDBLK(NUMBLK) - Element block ids (must be unique)
```

```

C
C   For each element block
C
C       NUMELB - Number of elements in the block (the sum of NUMELB
C               over all blocks must equal NUMEL above)
C       ELTYP  - Name of element type in this block (CHARACTER*8)
C       NELNOD - Number of nodes defining the connectivity for the
C               element in this block
C       NATRIB - Number of element attributes for this element type
C
C       ICONBK(NELNOD,NUMELB) - Element connectivity for the block
C       ATRBBK(NATRIB,NUMELB) - Element attributes for the block
C
C   Nodal Point Boundary Condition Data
C
C       LNSNL - Length of the node set node list
C
C       IDNS(NUMNS)   - Node set ids
C       NNNS(NUMNS)   - Node set node counts
C       NDFNS(NUMNS) - Node set distribution factor counts
C       IPNNS(NUMNS) - Node set node pointers
C       IPDFNS(NUMNS) - Node set distribution factor pointers
C       LSNNS(LNSNL) - Node set node list
C       DFNS(LNSNL)  - Node set distribution factor list
C
C   Element Side Set Boundary Condition Data
C
C       LSSEL - Length of the element side set element list
C       LSSDF - Length of the element side sets distribution factor list
C
C       IDSS(NUMSS)   - Element side set ids
C       NESS(NUMSS)   - Element side set element counts
C       NDFSS(NUMSS) - Element side set distribution factor counts
C       IPESS(NUMSS) - Element side set element pointers
C       IPDFSS(NUMSS) - Element side set distribution factor pointers
C       LSESS(LSSEL) - Element side set element list
C       LSSSS(LSSNL) - Element side set side list
C       DFSS(LSSDF)  - Element side set distribution factor list
C

```

## Appendix F - Post-Processing File Contents

The file structure employed by COYOTE to write data for an external post-processing file is based on the EXODUS II [6] standard. The contents of the file include all of the data records from the mesh generation file (Appendix D) plus the results records which are summarized below. The EXODUS II file is a random access file and the recording of data is performed through a series of subroutine calls. Post-processing data is written by COYOTE in subroutines EXOSET, SETIC and EXOWRT.

```

C
C   Output Variable Sizing Parameters
C
C   NVARNP - Number of variables for each node
C   NVAREL - Number of variables for each element
C   NVARGL - Number of global variables
C
C   Variable Names
C
C   NAMENV(NVARNP) - Names of nodal variables (CHARACTER*8)
C   NAMEEV(NVAREL) - Names of element variables (CHARACTER*8)
C   NAMEGV(NVARGL) - Names of global variables (CHARACTER*8)
C
C   Element Variable Truth Table
C
C   ITRUTH(NVAREL,NUMBLK) - Output flag for element variables within
C                           each element block
C
C   Solution Records (repeat for each timeplane)
C
C   TIME - Current solution time
C
C   SOLNNP(NUMNP) - Solution at the nodal points (repeat for each
C                   variable, as needed)
C   SOLNEL(NUMELB) - Solution at for each element (repeat for each
C                   element block and variable, as needed)
C   SOLNGL(NVARGL) - Solution for each global variable
C

```