SANDIA REPORT

SAND96-0903 • UC-905

Unlimited Release

Printed May 1996

RECEIVED

JUN 03 1996

OSTI

TORO II – A Finite Element Computer Program for Nonlinear Quasi-Static Problems in Electromagnetics

Part II – User's Manual

David K. Gartling

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-94AL85000

Approved for public release; distribution is unlimited.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER

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

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

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

NTIS price codes
Printed copy: A06
Microfiche copy: A01

SAND96-0903
Unlimited Release
Printed May 1996

**TORO II -
A FINITE ELEMENT COMPUTER PROGRAM
FOR NONLINEAR QUASI-STATIC PROBLEMS
IN ELECTROMAGNETICS
PART II - USER'S MANUAL**

David K. Gartling
Engineering Sciences Center
Sandia National Laboratories
Albuquerque, New Mexico 87185

ABSTRACT

User instructions are given for the finite element, electromagnetics program, TORO II. The theoretical background and numerical methods used in the program are documented in SAND95-2472. The present document also describes a number of example problems that have been analyzed with the code and provides sample input files for typical simulations.

Acknowledgments

The significant contributions of several individuals to this code development effort must be acknowledged. The formulation and capabilities included in the code were heavily influenced by the expertise of A. J. Russo (9114). Extensive testing and benchmarking of the code, as well as generation of several example problems, was performed by P. L. Hopkins (9112). R. E. Hogan (9113) developed the pointer scheme for the sparse matrix solver used in TORO II.

Contents

1	Introduction	1
2	Program Overview	2
2.1	Program Features	2
2.2	Program Organization	4
3	Input Guide	7
3.1	Input Syntax	8
3.2	Title Data Block	10
3.3	Material Data Block	11
3.4	Problem Definition Data Block	17
3.5	Solution Data Block	27
3.6	Post-processing Data Block	34
3.7	Time Function Data Block	37
3.8	Variable Function Data Block	38
3.9	User Constants Data Block	39
3.10	Termination Data	40
3.11	User Supplied Subroutines	41
3.11.1	Material Property Subroutines	41
3.11.2	Volumetric Sources	46
3.11.3	Boundary Conditions	49
3.12	Initial Conditions and Restarts	54
3.13	Error Messages	54

4	Code Installation and Access	55
4.1	FORTTRAN Coding and System Dependencies	55
4.2	File Formats	56
4.3	File Usage	57
4.4	Access to the Code	57
5	Example Problems	59
5.1	Problem 1 - Electrostatic Field Problem	59
5.2	Problem 2 - Magnetic Field Problem	64
5.3	Problem 3 - Magnetic Induction Problem	72
5.4	Problem 4 - Current Flow/Magnetostatic Problem	76
5.5	Problem 5 - Nonisothermal Magnetic Induction Problem	84
6	References	91
	Appendix A - Summary of Input Commands	93
	Appendix B - Consistent Units	99
	Appendix C - Common Block and Array Storage	100
	Appendix D - External Mesh Generator File Contents	121
	Appendix E - Post-Processing File Contents	123

1 Introduction

The TORO II computer code is a general purpose program designed for the solution of static and quasi-static problems involving electromagnetic fields. The code is based on the Galerkin form of the finite element method (FEM). The present version of TORO II represents a significantly extended edition of the original program which was limited to axisymmetric geometries.

The class of problems recognized by TORO II include the scalar potential problems associated with electrostatics and steady current flows as well as the vector potential problems of magnetostatics and eddy currents. These types of field problems are readily simulated in arbitrary two-dimensional and three-dimensional geometries. Derived electric and magnetic field variables are also provided by TORO II as are the thermo-mechanical quantities, such as Joule heating and Lorentz forces. The coupling of TORO II with other mechanics codes was a primary consideration in the code design.

A significant effort has been made during the development of TORO II to create an analysis program that is easy to use. The basic code structure and many of the programming features found in TORO II have been developed and refined from earlier versions of this and other programs. The TORO II architecture is directly derived from the thermal analysis code, COYOTE II [1]. This manual also shares large parts of the COYOTE II user's manual [2] and is again an indication of TORO's heritage. Input to TORO II follows the style used in a series of fluid and thermal codes and allows a user ready access to a variety of analysis packages with a minimum of time spent on learning input format conventions. The code has been written using standard FORTRAN 77 in an effort to increase its portability to a wide variety of computer systems.

The present document is intended to provide a detailed description of the input data necessary to access and execute the present version of the TORO II code. The theoretical basis for the code and many of the numerical procedures used in the program are described in a companion document [3].

In the next section, a brief description of the program capabilities and organization is provided. Section 3 describes the input to the program; the penultimate section provides a discussion of programming and installation details. A variety of example problems are included in the last section to illustrate typical input data files and simulation results.

2 Program Overview

The development of a reasonably efficient computer code for electromagnetics or any type of field problem 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 sections 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

TORO II is intended for the analysis of static and quasi-static electromagnetic problems such as electrostatics, steady current flows, magnetostatics and eddy currents. Geometrically, this version of the program is appropriate for treating two-dimensional, plane or axially symmetric problems and fully three-dimensional analyses. For eddy current problems the current fields may be arbitrarily time varying or time-harmonic. 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 the various field variables. Some limitations on nonlinear material behavior are imposed for the time-harmonic, eddy current option. Volumetric source terms may also be defined and allowed to vary in a general manner.

Boundary condition specification is very general and allows all of the standard types of conditions to be imposed on a problem. Scalar and vector potentials or their derivatives may be specified; all boundary conditions may be general functions of spatial location, time and/or appropriate field variables. TORO II 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 TORO II in a standard, well-documented format that allows easy coupling with other programs. The data analysis portions of the code allow derived electric and magnetic field quantities to be computed; thermal and mechanical quantities may also be derived.

Two other essential parts of TORO II are the element library and the solution procedures for the finite element equations. The element library is illustrated in Figure 1 and includes isoparametric quadrilaterals and triangles for two-dimensional applications and isoparametric tetrahedrons, wedges and hexahedrons for three-dimensional simulations. Within each of these elements, the potential variables are approximated using either linear or quadratic basis functions. For the analysis of time-independent problems, TORO II provides the user with a choice of several variants of a Picard iteration scheme. 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

Figure 1: Finite elements in the TORO II library; the linear and quadratic version of each element is available in the code.

used in a predictor/corrector mode with a fixed timestep or a dynamic timestep algorithm. Time harmonic problems utilize the various Picard schemes when nonlinearities are present. Details of these methods and their implementation can be found in [3].

2.2 Program Organization

TORO II 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, 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. The diversity of field problems treated by TORO II has required that distinct solvers be organized in the code. These are labeled A through D and correspond to static electric problems, static magnetic problems, time dependent eddy current problems and time-harmonic eddy current problems. Solver dependent routines are named with a postfixed A-D to allow easy association with a problem type.

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; 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 (**TITle**) - specifies problem title and comments
- Material Data Block (**MATerial**) - specifies material property data
- Problem Definition Data Block (**PROBLEM DEFinition**) - specifies problem type, element block properties, boundary conditions and simulation options
- Solution Data Block (**SOLution**) - specifies solution method and options
- Post-processing Data Block (**POST**) - specifies post-processing output and options
- Function Data Block (**TIME FUNction**, **VARiable FUNction**, **USER CON-stant**s) - specifies function data and user-defined constants

The large amount of data needed for a typical finite element analysis is handled in TORO II 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

Figure 2: Organization of the TORO II code.

stored on various disk files. The utilization of the files shown in Figure 2 is discussed in Section 4.3.

3 Input Guide

The structure of an input data file for the TORO II program directly reflects the steps required to formulate, solve and analyze a finite element model for an electromagnetics problem. Input to the code consists of four types of cards ¹: 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 TORO II 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 TORO II are listed below in the order in which they are discussed in the subsequent sections. 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
- 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

¹Notice 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.

3.1 Input Syntax

The TORO II 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.*, itermax.
- (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.

Figure 3: Coordinate system definition.

- (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 , the axial coordinate, z , with y and the circumferential coordinate, θ , with z . The global coordinate system used by the code is right-handed, as indicated in the sketch shown in Figure 3.
- (o) Explanatory notes for various input options and data are numbered consecutively within a section and are located at the end of each Data Block.

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 TORO II 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 TORO II 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 type, material model □  
.  
.  
Material Property Data  
.  
.  
ENDmaterial □
```

where

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

material type (C) : is a required material type. This parameter should be set to CONDuctor, for electrically conductive materials or DIElectric for nonconductive materials.

material model (C) <ISOtropic> : is the type of material model used to describe the electric and/or magnetic tensor properties. When this parameter is omitted or set to ISOtropic, the material is modeled as isotropic with electric conductivity $\sigma_{ij} = \sigma\delta_{ij}$, magnetic permeability $\mu_{ij} = \mu\delta_{ij}$ and electric permittivity $\epsilon_{ij} = \epsilon\delta_{ij}$; a parameter value of ORTHotropic indicates a material with full tensor behavior, σ_{ij} , μ_{ij} and ϵ_{ij} (see Note 2).

Material Property Data

Electromagnetic properties for each material are 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. TORO II does not contain any dimensional constants and, therefore, the units for the material properties are free to be chosen by the user, though they must be consistent with the rational MKS form of Maxwell's equations [3]. For convenience, a table of consistent units is given in Appendix B.

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

ELECTric CONDUCTivity= $\sigma_{11}, \sigma_{22}, \sigma_{33}$ □

ELECTric CONDUCTivity=USER □

ELECTric CONDUCTivity=VFUNction, idvar1, idvar2, idvar3 □

$\sigma_{11}, \sigma_{22}, \sigma_{33}$ (R) : specify the values of the principle components of the electric conductivity tensor. If the material is isotropic only the first value, σ_{11} , is required; orthotropic materials in two-dimensional problems only require the first two components of the tensor σ_{11} and σ_{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 electric 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).

MAGnetic PERMeability= $\mu_{11}, \mu_{22}, \mu_{33}$ □

MAGnetic PERMeability=USER □

MAGnetic PERMeability=VFUNction, idvar1, idvar2, idvar3 □

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

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

VFUNction (C) : specifies that the principle components of the permeability 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 magnetic permeability 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).

ELECTric PERMittivity= $\epsilon_{11}, \epsilon_{22}, \epsilon_{33}$ □

ELECtric PERMittivity=USER

□

ELECtric PERMittivity=VFUNction, idvar1, idvar2, idvar3

□

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

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

VFUNction (C) : specifies that the principle components of the permittivity 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 magnetic permittivity 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 and permeability tensors onto the global coordinate axes (see Note 2).

USER (C) : specifies that the local orientation of the conductivity and permeability tensors will be evaluated by a user subroutine, USRRROT (see Note 2).

CURrent DENsity= Jx, Jy, Jz, θ

CURrent DENsity=USER

CURrent DENsity=TFUNction, idtim1, idtim2, idtim3

Jx, Jy, Jz (R) $\langle 0.0, 0.0, 0.0 \rangle$: specifies the real values of the components of the current density. If the problem is two-dimensional only one or two components of the current density need to be specified (see Note 4). For time-harmonic problems the specified current density is the peak value for a cycle.

θ (R) $\langle 0.0 \rangle$: specifies the value of the phase angle (in degrees) for a time-harmonic current density (see Note 4).

USER (C) : specifies that the current density will be evaluated by a user-supplied subroutine, USRJ.

TFUNCTION (C) : specifies that the components of the current density will be evaluated via one or more time functions (see Note 3).

idtim1, idtim2, idtim3 (I) : sets the time function identification numbers for each component of the current density. (see Note 3).

ELECTric CHARge= ρ

ELECTric CHARge=USER

ELECTric CHARge=VFUNction, idvar

ρ (R) <0.0> : specifies the value of the electric charge density.

USER (C) : specifies that the electric charge will be evaluated by a user-supplied subroutine, USRECH.

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

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

INITial MAGnetic POTential= $Ax_{init}, Ay_{init}, Az_{init}$

$Ax_{init}, Ay_{init}, Az_{init}$ (R) <0.0, 0.0, 0.0> : specifies the initial values for the components of the magnetic potential.

INITial ELECTric POTential= V_{init}

V_{init} (R) <0.0> : specifies the initial value for the electric scalar potential.

Notes:

- 1) The material name assigned to each material data block must be unique. This name is also used in the problem definition section (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 TORO II include materials with either an isotropic or orthotropic conductivity, permeability and/or permittivity tensor. The following describes the options for the electric conductivity tensor; the alternatives for the other tensors are similar. For isotropic materials, $\sigma_{ij} = \sigma$, and the electric conductivity is specified completely by setting the value of the scalar σ . Dependence of the isotropic conductivity on other field variables can be implemented via a variable function or a user supplied subroutine. For orthotropic materials, σ_{ij} is defined as $\delta_{ij}\sigma_{ij} = \sigma_{ii}$, and the electric conductivity is determined by specifying the components of the tensor with respect to the principle material axes. The dependence of each of the principle conductivities on other field variables 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 4, 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° 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. The specified rotation is also applied to all tensors that are present under the given material name. If the tensor rotation is required to vary with spatial location or vary between different tensors 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 any of the dependent variables 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, a user subroutine is recommended. User subroutines are

Figure 4: Notation for orthotropic conductivity tensor.

described in Section 3.11. Note that there are limits to the functional dependence of the material coefficients. The electric conductivity and permittivity may depend on the electric scalar potential, V , the electric field, E and the temperature if it is available. The magnetic permeability may be a function of the magnetic vector potential, A , the magnetic flux, B and the temperature, if available. Current density may not be a function of the dependent field variables.

- 4) The current density components are volumetric sources in the eddy current and magnetostatic problems and must be specified in the coordinate system for the problem. Currents for conductors with arbitrary orientation are best input via the user subroutine where spatial location data is available. For two-dimensional problems, the orientation of the specified current dictates which components of the magnetic potential must be activated. If the current is contained within the plane of the problem (*e.g.*, J_x and J_y specified), then two components of the potential are required (see Section 3.4, **PROBLEM TYPE**). When the current is out-of-the-plane (*e.g.*, J_z specified), then only the one component of the potential is needed, though it will vary with the in-plane coordinates. Similar considerations apply to the axisymmetric case. The currents input on the **CURRENT DENSITY** card should correspond to the number and direction of the variables that are specified on the **PROBLEM TYPE** card. For time-harmonic problems the specified phase angle is used to compute the imaginary value of each component of the current.

3.4 Problem Definition Data Block

The problem definition data block is used to define the general attributes of the electromagnetic 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 TORO II. Currently the only supported data format is the EXODUS II format [4].

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 TORO II are listed below with the various parameter choices.

GEOMETRY=type □

type (C) : specifies the type of geometry used in the field 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).

PROBLEM TYPE=description, electric dof, magnetic dof x,

magnetic dof y, magnetic dof z □

description (C) : specifies the type of electromagnetic boundary value problem that is being solved. The permissible settings for this parameter are ELECTrostatic, CURrent FLOW, MAGnetostatic and EDDY CURrent (see Note 3).

electric dof (C) : specifies that the electric potential variable is active for this problem. This parameter should be set to V if the electric potential is part of the simulation and should be left blank if the electric potential is not included in the present problem (see Note 3).

magnetic dof x (C) : specifies that the x (r) component of the magnetic potential is active for this problem. This parameter should be set to AX if the component is needed in the simulation and should be left blank if the component is not to be included (see Note 3).

magnetic dof y (C) : specifies that the y (z) component of the magnetic potential is active for this problem. This parameter should be set to AY if the component is needed in the simulation and should be left blank if the component is not to be included (see Note 3).

magnetic dof z (C) : specifies that the z (θ) component of the magnetic potential is active for this problem. This parameter should be set to AZ if the component is needed in the simulation and should be left blank if the component is not to be included (see Note 3).

CURrent TYPE=description, frequency

description (C) <DC> : specifies the type of current for the problem. The permissible settings for this parameter are DC for arbitrary time varying currents and AC for time-harmonic problems (see Note 3).

frequency (R) <0.0> : specifies the frequency for time-harmonic (AC) currents. Note that this is an angular frequency in radians/sec.

HYSteresis LOSS=option

option (C) <OMIT> : specifies if hysteresis losses are to be included in the volume source computation. If this parameter is omitted or set to OMIT, the computation of volume (ohmic) heating will not include contributions from hysteresis in the B vs. H curve. If this parameter is set to INCLude, then TORO II expects a user subroutine, USRHYS (see Section 3.11.2), to be supplied for the description of the hysteretic loss MODEL. The HYSteresis LOSS option is only relevant for time-harmonic, eddy current problems.

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 ELECTric POTential, POINT SOURCE, MAGnetic POTential X, MAGnetic POTential Y, and MAGnetic POTential Z. The POINT SOURCE option is only available for electrostatic problems (see Note 4).

nodeset id (I) : specifies the previously assigned node 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: --

rbvalue, ibcvalue (R) <,0.0> : the constant values of the real and imaginary parts of the boundary condition. The ibcvalue is only required for time harmonic simulations and is defaulted to 0.0.

USER (C) : the boundary condition will be evaluated by a user-supplied subroutine. The subroutine for the ELECTric POTential boundary condition is USRPOT, the subroutine for the POINT SOURCE condition is USREPS and the subroutine for the MAGnetic POTential boundary condition is also USRPOT. The subroutine USRPOT allows the specification of both real and imaginary parts of the potential for time-harmonic applications.

TFUNCTION (C), idtim (I) : the boundary condition will be evaluated via a time function with time function identification number, idtim (see Note 6).

VFUNction (C), idvar (I) : the boundary condition will be evaluated via a variable function with variable function identification number, idvar (see Note 6). This parameter choice is only available for the POINT SOURCE boundary condition. Time-harmonic problems cannot use this option.

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

ELECtric CURrent, MAGnetic FIEld X, MAGnetic FIEld Y and MAGnetic FIEld Z (see Note 5). Note that the X, Y, Z suffix on the *boundary condition* name refers to the component of the magnetic potential equation to which the condition is applied; it does *not* refer to the direction of the magnetic field.

sideset id (I) : specifies the previously assigned side set identifier for a group of boundary conditions (see Note 5).

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

rbcvalue, ibcvalue (R) <,0.0> : the constant values of the real and imaginary parts of the boundary condition. The ibcvalue is only required for time harmonic simulations and is defaulted to 0.0.

USER (C) : the boundary condition will be evaluated by a user-supplied subroutine. The subroutine for the ELECtric CURrent boundary condition is USRFLX as is the subroutine for the MAGnetic FIEld conditions. The subroutine USRFLX allows specification of both the real and imaginary parts of a flux for time-harmonic applications.

TFUNction (C), idtim (I) : the boundary condition will be evaluated via a time function with time function identification number, idtim (see Note 6).

VFUNction (C), idvar (I) : the boundary condition will be evaluated via a variable function with variable function identification number, idvar (see Note 6). For time-harmonic problems the same function is applied to both the real and imaginary parts of the boundary flux.

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.

EXternal DATA=data source

data source (C) <FILE> : specifies the source of the external data to be used in TORO II. When this parameter is set to FILE or left blank, TORO II expects to obtain external data from an EXODUS II file that has been attached to the run. If this parameter is set to CODEs, then TORO II expects to obtain data from another code that has been attached to the run through the PVM software (see Note 7).

EXternal CODEs, MASTER=code name1, SLAVE=code name2,

MTRANslator=code name3, STRANslator=code name4

code name1 (C) : specifies the code that is to act as the master in a coupled mechanics analysis (see Note 7).

code name2 (C) : specifies the code that is to act as the slave in a coupled mechanics analysis (see Note 7).

code name3 (C) : specifies the code that is to act as the translator from the master to the slave mesh in a coupled mechanics analysis (see Note 7).

code name4 (C) : specifies the code that is to act as the translator from the slave to the master mesh in a coupled mechanics analysis (see Note 7).

EXTernal NODal FIELd=i/o status, name1, name2, ... , namen

i/o status (C) <READ> : specifies the direction that nodal data is to be transferred between TORO II and an external data source. For an external EXODUS II file, this parameter must be set to READ. When TORO II is connected to another code, this parameter may be either READ or WRITE; both reading and writing data to an external code is allowed but requires separate EXTernal NODal FIELd cards (see Note 7).

name (C) : specifies the names of nodal point variables that are to be read/written from/to an external source. Variable names that are currently recognized by TORO II for the READ option include: TEMP (temperature). Variable names that are recognized for the WRITE option include: none.

EXTernal ELEMeNt FIELd=i/o status, name1, name2, ... , namen

i/o status (C) <READ> : specifies the direction that element data is to be transferred between TORO II and an external data source. For an external EXODUS II file, this parameter must be set to READ. When TORO II is connected to another code, this parameter may be either READ or WRITE; both reading and writing data to an external code is allowed but requires separate EXTernal ELEMeNt FIELd cards (see Note 7).

name (C) : specifies the names of element variables that are to be read/written from/to an external source. Variable names that are currently recognized by TORO II for the READ option include: STATUS (element birth/death status). Variable names that are recognized for the WRITE option include: J.HEAT (Joule heating) and LOR.F (Lorentz forces).

EXTernal DATA INTerval=variable, begin value, end value

variable (C) <STEPS> : specifies the variable that controls the start and end of data exchange with an external data source. Variable names that are currently recognized by TORO II include STEPS, which indicates that the time step number is used to control the exchange and TIME, which indicates that the simulation time will be used to start and stop data exchange (see Note 7). The EXTernal DATA INTerval option is only applicable when TORO II is connected to an external file or is the master code in a PVM exchange.

begin value (I or R) <1 or 0.0> : specifies the value of the variable at which data exchange is to begin. The default values start the data exchange at the first step of the simulation.

end value (I or R) <nstep or t_{final} > : specifies the value of the variable at which data exchange will be terminated. The default values end the data exchange at the last step of the simulation.

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.

AUXiliary FIELDS=name1, name2, ... namen

namen (C) : specify the names of the auxiliary field variables that are to be computed. The following variable names are allowed: CURrent DENsity, MAGnetic FLUX, MAGnetic FIELd, ELECTric FIELd, JOULE HEATING and LORentz FORCE (see Note 8).

MAXimum CONnectivity=maxcon1, maxcon2

maxcon1 (I) : the maximum value of the element connectivity in the primary equation system (see Note 9).

maxcon2 (I) : the maximum value of the element connectivity in the secondary equation system. This value is only required for the segregated solution scheme used for magnetostatics problem with a computed current flow. The maxcon2 parameter refers to the current flow problem.

Notes:

- 1) Element blocks form a basic grouping for data within TORO II 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 TORO II through reference to the element block id. See reference [4] 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 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. 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) The need for the **PROBLEM TYPE** card is a result of the diversity of electromagnetic problems handled by TORO II and the variation in dependent variables with geometry. Electrostatic and current flow problems are the simplest and only require that the scalar potential V be active. Magnetostatic problems are more complex and the active variables depend directly on geometry and specified components of the current density. If all electric currents are specified in the problem (through the material data card) then only the appropriate magnetic vector potentials need to be activated. For two-dimensional problems with in-plane currents AX and AY are needed; two-dimensional geometries with an out-of-plane current only require the AZ component. The axisymmetric case is handled in a similar manner. Three-dimensional problems require that all three components be active. When the current is not specified directly, but is to be computed from the current flow equation, then the electric scalar variable must be specified along with the appropriate components of the magnetic potential as outlined above. Note that a problem with an out-of-plane current cannot use the steady current flow equation approach and must have the current specified directly. Eddy current problems, both time-varying and time-harmonic, require the same types of considerations for active variables.

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

Table 1: Quadrature rules for elements in TORO II

- 4) Boundary conditions for individual nodal points in a mesh are assigned in TORO II 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 TORO II through the EXODUS II file. See reference [4] for further details.
- 5) Boundary conditions for element surfaces or edges are assigned in TORO II 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 TORO II through the EXODUS II file. See reference [4] for further details.
- 6) 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 any

field variable) 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 4 and 5) or a user subroutine.

- 7) TORO II can be linked to two types of external data sources - a file or another mechanics code. For weakly coupled problems, external data can be supplied to TORO II through an EXODUS II file that was generated by some other mechanics code. This type of coupling allows TORO II to simply read field data as needed during the electromagnetics simulation and provide element and nodal point values to the material property subroutines or variable functions. TORO II is currently configured to handle temperature data from an external source, though other variables could be added with minor code modifications.

When the problem of interest is strongly coupled then TORO II can be linked directly with another mechanics code to allow two-way data exchange at any appropriate time during the simulation. This coupling is achieved through the Parallel Virtual Machine (PVM) utilities [5]. In this type of coupling one code acts as the master and directs all of the data exchange while the second code is a slave and only accepts and provides field data on command. The field variables that are to be exchanged are listed on the **EXTERNAL NODAL FIELD** and **EXTERNAL ELEMENT FIELD** cards for each code. When the coupled codes are run on the same computational mesh data exchange is direct; different meshes for the two codes require the use of intermediate mesh-to-mesh translators, such as MERLIN II [6]. The solution interval during the simulation over which data is to be exchanged, is set by the **EXTERNAL DATA INTERVAL** data card. The selection of specific times during the exchange interval for data to be exchanged is prescribed by the **EXTERNAL DATA EXCHANGE** which is described in Section 3.5. Further details on the use of the PVM coupling are provided in [7] and in the example problems of Section 5.0.

- 8) The specification of the **AUXILIARY FIELDS** option causes the x , y and z components of the various flux vectors 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. The data in Table 1 shows the integration points used for the flux evaluation. Note that the Joule heating and Lorentz force quantities are evaluated and stored as element variables, though they are computed from nodal data.
- 9) The sparse matrix format used by TORO II is a very efficient method for storing large sparse systems. However, to properly allocate memory for this scheme an

estimate of the maximum element connectivity is required. This connectivity is easily computed for structured mesh topologies but is rather expensive and difficult to find for unstructured discretizations. By definition, the maximum connectivity is the largest number of degrees of freedom (equations) that any given degree of freedom (equation) is connected to through adjoining elements. For a structured, two-dimensional mesh of four node elements with one degree of freedom per node, the maximum connectivity is 9 while for an eight node element the connectivity is 21. The maximum connectivity for a structured, three-dimensional mesh of eight node elements (one degree of freedom per node) is 27. The maximum connectivity parameter should generally be set for any mesh that is unstructured; a reasonable connectivity estimate is 130% to 150% of the connectivity for a structured mesh with the same basic element type.

3.5 Solution Data Block

The control of the solution process for the steady, time-dependent or time-harmonic electromagnetic 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, steady state or time-harmonic problem. This parameter should be set to TIME INDEpendent for a steady solution, set to TIME DEPENDent for a transient solution sequence and set to TIME HARmonic for a single frequency, time-harmonic solution.

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 TORO II 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. The permissible value for this parameter is PICard.

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 3).

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 3).

MASS MATRix=type

type (C) <CONSistent> : specifies the treatment of the various mass matrices in time dependent problems. The permissible values for this parameter are CONSistent and LUMPed (see Note 3).

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 3).

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 3).

NORM VALues= V_{norm} , Ax_{norm} , Ay_{norm} , Az_{norm}

$V_{norm}, Ax_{norm}, \dots (R) < \text{maximum value in solution} >$: specifies an appropriate (maximum) value for the electric and magnetic potentials to be used in defining an RMS norm for the change over a time step or iteration. The default scale for the norm is the maximum value of the variable found in the solution vector at each step or iteration. If the maximum values are not constant for the simulation, these parameters should be specified to ensure a consistent definition of the norms. The definition of the RMS norm is given in [3].

TIME STEP FACTor= $\beta_{explicit}$

$\beta_{explicit} (R) < 0.9 >$: 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 3).

TIME STEP= Δt

□

$\Delta t (R)$: 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.

INITial TIME= t_{init}

$t_{init} (R) < 0.0 >$: 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 4).

NUMber TIME STEP s= $nstep$

□

$nstep (I) < 1000 >$: specifies the total number of time steps allowed in a time-dependent simulation (see Note 4).

MINimum TIME STEP= Δt_{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= Δt_{max}

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

CONVergence TOLerance=tol

tol (R) < 1.0×10^{-5} > : specifies the convergence tolerance for a nonlinear, steady state problem or the steady state tolerance for a time-dependent simulation (see Note 4).

RELaxation FACTor= α

α (R) <0.0> : specifies the relaxation factor to be used in the iterative solution of a nonlinear, steady state problem.

MAXimum ITERations=itermax

itermax (I) <25> : specifies the maximum number of iterations allowed in the iterative solution of a nonlinear, steady state problem (see Note 4).

EXTernal DATA EXchange=exchange variable, exchange value

exchange variable (C) : specifies the variable or parameter that controls the reading of data from an external file or the exchange of data with an external code. The allowable settings for this parameter are currently limited to STEPs, if data is to be exchanged based on a fixed number of time steps or iterations and TIME, if an elapsed time interval is the exchange criterion (see Note 5).

exchange value (I or R) <1, > : specifies the interval for data exchange. If the exchange variable parameter is set to STEPs, then this integer parameter sets the number of time steps or iterations between a data read or a data exchange. When the exchange variable parameter is TIME, then this real parameter is the elapsed time between a data read or a data exchange (see Note 5).

PRINTed OUTput=option, frequency

option (C) <SUMmary> : specifies the type of printed output provided 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.

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 permissible 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 6).

krylov subspace (I) <10> : indicates the dimension of the Krylov subspace used with the GMRes iterative solution method.

PREConditioner TYPE=option, *polynomial order*

option (C) <JACobi> : 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 6).

polynomial order (I) < 1 > : specifies the order of the polynomial used in the POLYnomial preconditioner option.

L2 NORM= L_2

L_2 (R) < 1.0×10^{-6} > : 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 6).

RESidual NORM=resid

resid (R) < 0.1 > : 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 6).

MAXimum MATRix ITERations=matrix iter

matrix iter (I) < numnod > : specifies the maximum number of iterations allowed for the solution of the matrix problem during any given iteration or time step (see Note 6).

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 are the last computed solutions for the dependent variables, 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.
- 2) Restarts of a solution procedure from a previously computed solution require that a file in an EXODUS II format be attached to TORO II 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) All of the time integration methods available in TORO II 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.

The mass matrix may be lumped or left in a consistent form with the implicit integration methods. Mass 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.

- 4) The solution processes in TORO II 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**) or when the change in the solution (norm) between iterations has reached a nominal value (**CONVergence TOLerance**). 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**) or the attainment of a steady state (**CONVergence TOLerance**). 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.
- 5) The frequency of data exchange between codes that are linked by PVM is controlled by the **EXternal DATA EXchange** card. This directive only needs to be

supplied when TORO II is connected to an external file or is the master code in a coupled problem since the master code controls the exchange process. If TORO II is the slave code, then directives from the master will indicate when a solution step is required. The first parameter on this data card selects a variable in TORO II that will be tested to determine when a data exchange is needed. Presently, this test is limited to a fixed number of time steps or iterations or a fixed time interval. Other tests, such as the change in a field variable, are possible and could be installed with minor alterations to the code. In most cases TORO II will act as a slave and this data will not be needed. Note that the overall simulation interval during which data may be read or exchanged is set by the **EXternal DATA INterval** card that is described in Section 3.4.

- 6) The primary matrix solution methods available in TORO II 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 TORO II 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 field problems the selection of a conjugate gradient solver (CG) with JACobi preconditioning is adequate. Unsymmetric systems arising from time-harmonic problems can be solved with a generalized minimum residual method (GMRes) and the ILU preconditioner. 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].

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 TORO II 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: V_REAL (electric potential, real), V_IMAG (electric potential, imaginary), V_MAGN (electric potential, magnitude), V_PHASE (electric potential, phase angle), AX_REAL, AY_REAL, AZ_REAL (components of the magnetic vector potential, real), AX_IMAG, AY_IMAG, AZ_IMAG (components of the magnetic vector potential, imaginary), A_MAGN (magnetic vector potential, magnitude), A_PHASE (magnetic vector potential, phase angle), JX_REAL, JY_REAL, JZ_REAL (components of the current density, real), JX_IMAG, JY_IMAG, JZ_IMAG (components of the current density, imaginary), J_MAGN (current density, magnitude), J_PHASE (current density, phase angle), BX_REAL, BY_REAL, BZ_REAL (components of the magnetic flux, real), BX_IMAG, BY_IMAG, BZ_IMAG (components of the magnetic flux, imaginary), B_MAGN (magnetic flux, magnitude), B_PHASE (magnetic flux, phase angle), HX_REAL, HY_REAL, HZ_REAL (components of the magnetic field, real), HX_IMAG, HY_IMAG, HZ_IMAG (components of the magnetic field, imaginary), H_MAGN (magnetic field, magnitude),

H_PHASE (magnetic field, phase angle), EX_REAL, EY_REAL, EZ_REAL (components of the electric field, real), EX_IMAG, EY_IMAG, EZ_IMAG (components of the electric field, imaginary), E_MAGN (electric field, magnitude), E_PHASE (electric field, phase angle), PSI_REAL (magnetic stream function, real) and PSI_IMAG (magnetic stream function, imaginary). Note that most of these fields are not automatically available and must be explicitly requested via the **AUXiliary FIELDS** option (see Section 3.4).

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 following variable names are allowed: J_HEAT (Joule heating) and LOR_FX, LOR_FY, LOR_FZ (components of the Lorentz force). These fields are not automatically available and must be explicitly requested via the **AUXiliary FIELDS** option (see Section 3.4).

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 currently recognized: TIMESTEP (the integration time step) and CGITER (number of iterations for matrix solver).

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 1).

OUTput TIME STEP= Δt_{out}

Δt_{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 1).

Notes:

- 1) 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 TORO II 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  
t1, f(t1)  
t2, f(t2)  
t3, f(t3)  
.  
.  
.  
tn, f(tn)  
.  
.  
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 TORO II. 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 TORO II 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:

VARIABLE FUNCTION=function id, var id

$\xi_1, f(\xi_1)$

$\xi_2, f(\xi_2)$

$\xi_3, f(\xi_3)$

.

.

.

$\xi_n, f(\xi_n)$

.

.

ENDvariable

where

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

var id (C) : indicates the specific dependent variable that is to be used with the function.

This parameter may have the following settings: V (electric potential), A (magnetic vector potential), J (current density), B (magnetic flux), H (magnetic field), E (electric field) or T (temperature). These variables must be available in TORO II as a result of an **AUXiliary FIELDS** data card or as externally supplied data.

$\xi_n, f(\xi_n)$ (R) : is a list of variable and function values used to define the variable function.

In the current version of TORO II the variables listed above are the only variables allowed in the definition of a variable function (see Note 1).

Notes:

- 1) The sequence of dependent variables for each function must form an increasing sequence. Also, no extrapolation or continuation of data beyond the last data point is allowed by TORO II. If the current variable value falls above or below the variable range of the function, the code will terminate execution with an error message. The function data is used with a linear interpolation scheme.

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 constants 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 sections. The termination of all data input to TORO II 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 TORO II. The occurrence of variable material properties or source terms 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 sections. Skeleton versions of all user subroutines are supplied with the TORO II code and may be used as templates for specific applications.

3.11.1 Material Property Subroutines

Variations in electric or magnetic properties due to variations in the dependent variables can normally be accommodated with function type data. However, when property variations are complex, depend on spatial location or depend on more than one variable, then a subroutine evaluation may be more appropriate. When any of the material property data cards **ELECTric CONDUCTivity**, **MAGnetic PERMeability**, **ELECTric PERMittivity** or **TENsor ROTation** 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 TORO II. 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.

Electric Conductivity

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

```
      SUBROUTINE USRCON (COND11, COND22, COND33, VIP, VEIP, VTIP,
*                      XIP, YIP, ZIP, NAME, NUMIPT, TIME, KSTEP,
*                      RCONST, ICONST)
C
C *****
C
C      CHARACTER*20 NAME
C
C      DIMENSION COND11(*), COND22(*), COND33(*)
C      DIMENSION VIP(*), VEIP(*), VTIP(*)
C      DIMENSION XIP(*), YIP(*), ZIP(*)
C      DIMENSION RCONST(*), ICONST(*)
C
C *****
```

```

C
C   FORTRAN coding to evaluate the electric conductivity at the element
C   integration points for each material
C
      RETURN
      END

```

where the variables in the subroutine parameter list are

COND11(NUMIPT), COND22(NUMIPT), COND33(NUMIPT) : arrays containing integration point values of the principle components of the electric conductivity tensor σ_{11} , σ_{22} and σ_{33} for the material (output). For isotropic materials set COND22=COND33=COND11; for two-dimensional in-plane problems COND33 need not be evaluated.

VIP(NUMIPT) : array containing integration point values of the electric potential (input).

VEIP(NUMIPT) : array containing integration point values of the electric field (input).

VTIP(NUMIPT) : 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.

Magnetic Permeability

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

```

      SUBROUTINE USRMPM (PERM11, PERM22, PERM33, VAIP, VBIP, VTIP,
*                      XIP, YIP, ZIP, NAME, NUMIPT, TIME, KSTEP,
*                      RCONST, ICONST)

```

```

C
C *****
C
C CHARACTER*20 NAME
C
C DIMENSION PERM11(*), PERM22(*), PERM33(*)
C DIMENSION VAIP(*), VBIP(*), VTIP(*)
C DIMENSION XIP(*), YIP(*), ZIP(*)
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C FORTRAN coding to evaluate the magnetic permeability at the element
C integration points for each material
C
C RETURN
C END

```

where the variables in the subroutine parameter list are

PERM11(NUMIPT), PERM22(NUMIPT), PERM33(NUMIPT) : arrays containing integration point values of the principle components of the magnetic permeability tensor μ_{11} , μ_{22} and μ_{33} for the material (output). For isotropic materials set PERM22=PERM33=PERM11; for two-dimensional in-plane problems PERM33 need not be evaluated.

VAIP(NUMIPT) : array containing integration point values of the magnitude of the magnetic vector potential (input).

VBIP(NUMIPT) : array containing integration point values of the magnitude of the magnetic flux (input).

VTIP(NUMIPT) : 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.

Electric Permittivity

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

```
      SUBROUTINE USREPM (PERM11, PERM22, PERM33, VIP, VEIP, VTIP,  
*                      XIP, YIP, ZIP, NAME, NUMIPT, TIME, KSTEP,  
*                      RCONST, ICONST)  
C  
C *****  
C  
C CHARACTER*20 NAME  
C  
C DIMENSION PERM11(*), PERM22(*), PERM33(*)  
C DIMENSION VIP(*), VEIP(*), VTIP(*)  
C DIMENSION XIP(*), YIP(*), ZIP(*)  
C DIMENSION RCONST(*), ICONST(*)  
C  
C *****  
C  
C FORTRAN coding to evaluate the electric permittivity at the element  
C integration points for each material  
C  
C RETURN  
C END
```

where the variables in the subroutine parameter list are

PERM11(NUMIPT), PERM22(NUMIPT), PERM33(NUMIPT) : arrays containing integration point values of the principle components of the electric permittivity tensor ϵ_{11} , ϵ_{22} and ϵ_{33} for the material (output). For isotropic materials set PERM22=PERM33=PERM11; for two-dimensional in-plane problems PERM33 need not be evaluated.

VIP(NUMIPT) : array containing integration point values of the electric potential (input).

VEIP(NUMIPT) : array containing integration point values of the electric field (input).

VTIP(NUMIPT) : 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.

Tensor Rotation

The local orientation of the material tensors for each anisotropic material are 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
C RETURN
C 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 3.3 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 3.3 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.

3.11.2 Volumetric Sources

Complex distributions of electric charge or current density may be specified through a user subroutine. If the charge density parameter, **ELEC**tric CHARge, is set to USER (see Section 3.3), then TORO II will expect the subroutine USRECH to be provided. Likewise, when the electric current parameter, **CUR**rent DENsity (see Section 3.3), is set to USER for one or more materials, then TORO II expects a source subroutine, USRJ, to be supplied by the user. Also, when the **HY**Steresis LOSS option is included (see Section 3.4) for a time-harmonic problem, the user subroutine USRHYS must be supplied to TORO II.

Electric Charge Density

The charge density subroutine is of the following form:

```

SUBROUTINE USRECH (VOLECH, VIP, VEIP, VTIP, XIP, YIP, ZIP,
*                NAME, NUMIPT, TIME, KSTEP, RCONST, ICONST)
C
```



```

C *****
C
C CHARACTER*20 NAME
C
C DIMENSION VOLECH(*), VIP(*), VEIP(*), VTIP(*)
C DIMENSION XIP(*), YIP(*), ZIP(*)
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C FORTRAN coding to evaluate the electric charge density at the
C element integration points for each material
C
C RETURN
C END

```

where the variables in the subroutine parameter list are

VOLECH(NUMIPT) : array containing the integration point values of the charge density (output).

VIP(NUMIPT) : array containing integration point values of the electric potential (input).

VEIP(NUMIPT) : array containing integration point values of the electric field (input).

VTIP(NUMIPT) : 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.

The subroutine for components of the current density must have the following form:

```

SUBROUTINE USRJ (CURX, CURY, CURZ, PANGLE, XIP, YIP, ZIP,
*              NAME, NUMIPT, TIME, KSTEP, RCONST, ICONST)
C
C *****
C
C CHARACTER*20 NAME
C
C DIMENSION CURX(*), CURY(*), CURZ(*), PANGLE(*)
C DIMENSION XIP(*), YIP(*), ZIP(*)
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C FORTRAN coding to evaluate the current density at the
C element integration points for each material
C
C RETURN
C END

```

where the variables in the subroutine parameter list are

CURX(NUMIPT), CURY(NUMIPT), CURZ(NUMIPT) : arrays containing integration point values of the components of the current density J_x , J_y and J_z for the material (output). For two-dimensional in-plane problems only CURX and CURY need to be evaluated; two-dimensional out-of-plane problems only require CURY.

PANGLE(NUMIPT) : an array containing the integration point values of the phase angle for the current density (output). This parameter need only be specified for time-harmonic problems and should be specified in degrees, if required.

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.

The model for a hysteresis loss computation is included via a subroutine of the following form:

```

      SUBROUTINE USRHYS (ELMHQ, ELMBM, ELMHM, ELMT, NAME, NNODES,
*                      RCONST, ICONST)
C
C *****
C
C   CHARACTER*20 NAME
C
C   DIMENSION ELMHQ(*), ELMBM(*), ELMHM(*), ELMT(*)
C   DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C   FORTRAN coding to evaluate the hysteresis loss at the
C   element nodal points for each material
C
C   RETURN
C   END

```

where the variables in the subroutine parameter list are

ELMHQ(NNODES) : an array containing the hysteresis heat generation at the element nodal points (output).

ELMBM(NNODES) : an array containing the magnitude of the magnetic flux density at the element nodal points (input).

ELMHM(NNODES) : an array containing the magnitude of the magnetic field at the element nodal points (input).

ELMT(NNODES) : an array containing the temperature at the element nodal points (input).

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

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

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

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

3.11.3 Boundary Conditions

Boundary conditions that vary as a simple function of time or a field variable 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. 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 TORO II. 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.

Electric and Magnetic Potential

A specified electric potential (bcname=ELEctric POTential) or a specified component of the magnetic potential (bcname=MAGnetic POTential X, Y, Z) 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 USRPOT (POTBC, POTBCI, TNODE, XNODE, YNODE, ZNODE,
*                      IDNSET, TIME, KSTEP, RCONST, ICONST)
C
C      *****
C
C      DIMENSION RCONST(*), ICONST(*)
C
C      *****
C
C      FORTRAN coding to evaluate a scalar or vector potential at a node
C
      RETURN
      END
```

where the variables in the subroutine parameter list are

POTBC : the real part of the potential at the node (output).

POTBCI : the imaginary part of the potential at the node (output). This parameter need only be set for time-harmonic problems.

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).

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

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

Point Source

A specified point source (bcname=POINT 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 USRPS (PSBC, VNODE, XNODE, YNODE, ZNODE, IDNSET, TIME,
*                   KSTEP, RCNST, ICONST)
C
C *****
C
C   DIMENSION RCNST(*), ICONST(*)
C
C *****
C
C   FORTRAN coding to evaluate the point source at a node
C
C   RETURN
C   END

```

where the variables in the subroutine parameter list are

PSBC : the point source at the node (output).

VNODE : the electric potential 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).

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

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

Electric Current and Magnetic Field

An applied boundary current (bcname=ELECtric CURrent) and the application of magnetic field components (bcname=MAGnetic FIELd X, Y, Z) 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, FLUXI, XS, YS, ZS, NNODES, IDSSET,
*                      TIME, KSTEP, RCNST, ICONST)
C
C *****
C
C      DIMENSION FLUX(*), FLUXI(*)
C      DIMENSION XS(*), YS(*), ZS(*)
C      DIMENSION RCNST(*), ICONST(*)
C
C *****
C
C      FORTRAN coding to evaluate the applied flux at the element
C      edge or surface nodes for each side set
C
C      RETURN
C      END
```

where the variables in the subroutine parameter list are

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

FLUXI(NNODES) : an array containing nodal point values of the imaginary part of the applied flux (output). This vector need only be set for time-harmonic problems.

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 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.

3.12 Initial Conditions and Restarts

The analysis of a time-dependent eddy current problem requires the specification of a set of initial conditions for all of the dependent variables. For cases where the potentials 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, TORO II allows the initial conditions to be input from an external file. In order to be compatible with TORO II, the initial conditions must be written as a standard EXODUS II [4] file. The mechanics of attaching a restart file to a TORO II execution are described in Section 4.3. Note that static and time-harmonic eddy current problems cannot be restarted.

The output file containing the solutions generated by TORO II has the same EXODUS II format as the initial condition file. Therefore, solutions obtained from TORO II may generally be used directly as initial conditions for subsequent problems. 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 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

TORO II 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 should 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.

4 Code Installation and Access

The use of a program such as TORO II 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 sections, all of the above topics are discussed in some detail. The main emphasis will be on the operation of TORO II 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 TORO II 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 TORO II resides in a general software package developed by Shadid and Schunk [8] and is written in standard C. The compilation and linking of the various 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, TORO II is a modular program that consists of a main driver routine and a number of subordinate, task-oriented subroutines and utilities. Communication 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 TORO II is included in Appendix C. This information would normally be of value only if the code is to be modified.

A dynamic dimensioning procedure is used in TORO II 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 [9]. 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 [9].

TORO II 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 [9] 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 TORO II are in the EXODUS II format [4], which is a random access file structure based on NetCDF [10]. 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 TORO II must provide data in the EXODUS II format. The contents of an EXODUS II file are summarized in Appendices D and E and are given in more detail in [4].

The standard version of TORO II 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 TORO II with respect to the size or complexity of the electromagnetic 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 [8,11] suitable for use with the iterative matrix solver.

4.2 File Formats

The disk files used internally by TORO II 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 section, the pre- and postprocessing files utilize the EXODUS II format. Within TORO II the pre- and postprocessing 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 D). The postprocessing file contains a copy of the pre-processing data in addition to the computed solution variables (see Appendix E). Instructions for accessing EXODUS II data through the subroutine interface is available in [4].

4.3 File Usage

TORO II 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 2 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 sections have commented on those file formats that would normally be of concern to the user.

When files are to be attached to a TORO II simulation 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 [9] obtains needed file names from the operating system for use in opening files during program execution. As described in the next section, a script is available for TORO II that simplifies the association of local file names with unit numbers. Note that within TORO II all files are opened in the OPNFIL subroutine.

Unit Number	Fortran Name/ID	File Usage	File Access	File Format
25	NIN	Input file	Sequential	Coded
26	NOUT	Output file	Sequential	Coded
30	NTP0	EXODUS II input data	Random Access	Binary
31	NTP1	Free field input	Sequential	Binary
32	NTP2	EXODUS II output data	Random Access	Binary
33	NTP3	EXODUS II external data	Random Access	Binary
34	NTP4	EXODUS II restart data	Random Access	Binary
35	NTP5	Unused		Binary

Table 2: Files used in TORO II

4.4 Access to the Code

A source version of TORO II 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, 9100. The current location of the program source code for any particular computer system may be obtained from the author.

The utility libraries needed for the execution of TORO II 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 [4,5,8,9]. The required libraries include the EXODUS II file access routines, the iterative matrix solution package, the PVM libraries and the SUPES utilities. When user-supplied sub-routines are employed with TORO II 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 TORO II and its required libraries. The makefiles for various computer environments at Sandia are located with the source code and may be obtained from the author.

In order to assist users in executing TORO II, a UNIX script is also available on the 9100 LAN's and through the SEACAS system [12]. 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.

5 Example Problems

A series of five problems have been included in this section to demonstrate the use of the previously described data cards and a few of the capabilities of the TORO II 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. The input and output files for the problems described in the following sections are available with the source code.

5.1 Problem 1 - Electrostatic Field Problem

The first problem is an electrostatic field problem that is adapted from an example problems manual for the commercial electromagnetics code, MSC/EMAS [13]. Figure 5 shows the geometry which consists of two electrodes embedded in a cylindrical porcelain insulator. The electrodes are maintained at potential values of 0 and 10 Kilovolts, respectively and the porcelain is surrounded by air. The permittivity of the porcelain is 4.4 compared to an air permittivity of 1.0; the electrodes are included in the analysis with a permittivity of 25.0. The insulator is 6 cm. in length with a radius of 3 cm. Detailed dimensions of the remaining parts can be obtained from [13].

Figure 5: Schematic of porcelain insulator and electrodes.

This problem is axisymmetric and a finite element mesh for the region was developed using the FASTQ program [14]. The outer boundary for the air space was moved sufficiently far from the insulator that a constant potential boundary condition could be applied and have a minimal influence on the solution near the electrodes. The axisymmetric mesh of four-node, quadrilateral elements is shown in Figure 6. A three-dimensional mesh was also generated by sweeping the axisymmetric mesh through 180° using the GEN3D utility [15]. The resulting mesh of eight-node hexahedral elements is also shown in Figure 6.

The input data file for this problem is listed in Figure 7. The only differences in the axisymmetric and three-dimensional input data are in the GEOM card and the POST definition of the variables that are written to the output file for plotting. This problem is symmetric and linear, and therefore requires only a single iteration of the Picard scheme with the default (iterative) matrix solver.

Shown in Figure 8 are contour plots of the electric scalar potential and the magnitude of the electric field for the three-dimensional simulation. The plots were obtained from the BLOT graphics package [16]. The results shown in Figure 8 compare well with the results illustrated in reference [13] and are exactly reproduced by the axisymmetric model. Other quantities, such as vector plots of the electric field and contours and vectors of the current density, could also have been illustrated with small changes to the MATERIAL data (electric conductivity), AUXILLIARY FIELDS specification (current density) and POST data (additional output variables).

Figure 6: Axisymmetric and three-dimensional meshes for insulator, electrodes and surrounding air space.

```

TITLE
PORCELAIN INSULATOR
EXAMPLE ELECTROSTATICS PROBLEM FOR TORO II
END
MATERIAL,PORCELAIN,CONDUCTOR
ELECTRIC PERMITTIVITY=4.4
END
MATERIAL,AIR,CONDUCTOR
ELECTRIC PERMITTIVITY=1.0
END
MATERIAL,COND1,CONDUCTOR
ELECTRIC PERMITTIVITY=25.0
END
MATERIAL,COND2,CONDUCTOR
ELECTRIC PERMITTIVITY=25.0
END
PROBLEM DEFINITION
GEOMETRY=AXISYMMETRIC
ELEMENT BLOCK=10,PORCELAIN
ELEMENT BLOCK=20,AIR
ELEMENT BLOCK=30,COND1
ELEMENT BLOCK=40,COND2
PROBLEM TYPE=ELECTROSTATIC,V
BC=ELECTRIC POTENTIAL,10,0.0
BC=ELECTRIC POTENTIAL,20,10000.
BC=ELECTRIC POTENTIAL,30,1000.
MAXIMUM CONNECTIVITY=40
AUXILIARY FIELDS=ELECTRIC FIELD
END
SOLUTION=1,TIME INDEPENDENT
ITERATIVE METHOD=PICARD
NORM VALUE=1.0
MAXIMUM ITERATIONS=1
PRINT OUT=SUMMARY,1
END
POST
NODAL DATA=V_REAL,EX_REAL,EY_REAL,E_MAGN
OUTPUT FREQ=1
END
STOP

```

Figure 7: Input file for TORO II simulation of electrostatic field problem.

Electric Potential

Electric Field

Figure 8: Contour plots of the electric potential and magnitude of the electric field.

5.2 Problem 2 - Magnetic Field Problem

The second example problem concerns the prediction of the magnetic field produced by a circular, current carrying wire loop. Two cases are considered with the first being a specified, steady current in the wire and the second introducing a time-harmonic (AC) current to the wire. An analytic solution for the case of a circular line source centered in the x-y plane ($z=0$) can be derived [17], where the single nonzero component of the magnetic potential is given by

$$A_\theta = \frac{\mu}{\pi k} J_0 \left(\frac{a}{r} \right)^{1/2} \left[\left(1 - \frac{1}{2} k^2 \right) K - E \right]$$

where

$$k^2 = 4ar / [(a+r)^2 + z^2]$$

and r, z are the radial and axial coordinates (in the axisymmetric coordinate system), a is the radius of the wire loop, J_0 is the real (or imaginary) part of the current and K, E are complete elliptic integrals of the first and second kind. The computational solution of this problem, which is shown schematically in Figure 9, considers a small but finite radius wire ($r_w=0.02$ m) and a large loop radius ($a=2.0$ m), such that the analytic solution will be approximated and can be used for comparison.

Figure 9: Schematic and mesh for wire loop problem.

The solution domain and finite element mesh as generated by FASTQ [14] is shown in Figure 9 where the axial and rotational symmetry of the problem has been used to minimize the computational region. The coil has a finite diameter and was modeled with approximately eighty linear elements. The far-field boundaries were located a reasonable distance from the wire in an attempt to minimize their influence on the near-field solution. It was found that highly accurate solutions could not be obtained on this mesh unless the analytic solution was applied along the exterior boundaries which was done for the results reported here.

The TORO II input file for the time-independent magnetostatic case is shown in Figure 10. The time-harmonic, eddy current input is very similar as shown by the right-hand-side additions in Figure 10. The boundary conditions on the magnetic potential are supplied by the user subroutine USRPOT as listed in Figure 11. Note that for the DC case only the real part of the boundary condition is used by the code. The AC boundary condition is for the case of a 75° phase angle which is also specified in the current density input.

Results for the AC case are shown in Figures 12-15; the DC case is exactly the same as the real part of the time-harmonic problem. Figures 12 and 13 show the computed solutions for the real part of the magnetic potential and the magnitude of the magnetic flux. The imaginary part of the potential has the same spatial form as the real part but its magnitude is scaled by the tangent of the phase angle, as expected. Shown in Figures 14 and 15 are contour maps of the percent error in the computed potential (real part) and magnetic flux magnitude as derived from the analytic solution. For the potential, the errors are quite small ($< 2\%$) and are concentrated around the wire. This is expected since the analytic solution only approximates the field close to the finite-sized wire. The maximum error in the magnetic flux is also close to the wire though some error also occurs in the coarser mesh above the wire. The magnetic flux, being a derivative quantity, would be expected to lose some accuracy in the larger finite elements.

```

TITLE
CIRCULAR COIL IN AIR - DC CURRENT $ CIRCULAR COIL IN AIR - AC CURRENT
ENDTITLE
MATERIAL,COIL,CONDUCTOR
CURRENT DENSITY=0.0,0.0,7.95775E8 $ CURRENT DENSITY=,,7.95775E8,75.0
ELECTRIC CONDUCTIVITY=5.8E7
MAGNETIC PERMEABILITY=1.256E-6
ENDMATERIAL
MATERIAL,AIR,DIELECTRIC,ISOTROPIC
MAGNETIC PERMEABILITY=1.256E-6
ENDMATERIAL
PROBLEM DEFINITION
GEOMETRY=AXISYMMETRIC
ELEMENT BLOCK=10,COIL
ELEMENT BLOCK=20,AIR
PROBLEM TYPE=MAGNETOSTATIC,,,,AZ $ PROBLEM TYPE=EDDY CURRENT,,,,AZ
$ CURRENT TYPE=AC,1000.0

BC=MAGNETIC POTENTIAL Z,10,USER
AUXILIARY FIELDS=MAGNETIC FLUX
MAXIMUM CONNECTIVITY=20 $ MAXIMUM CONNECTIVITY=40
ENDPROBLEM
SOLUTION,1,TIME INDEPENDENT $ SOLUTION,1,TIME HARMONIC
MAX ITER=1
CONV TOL=1.0E-7

$ MATRIX SOLVER=GMR
$ PREC TYPE=ILU

PRINTED OUTPUT=SUMMARY,1
ENDSOLUTION
POST
NODAL DATA=AZ_REAL,B_MAG $NODAL DATA=AZ_REAL,AZ_IMAG,B_MAG
OUTPUT FREQ=1
ENDPOST
STOP

```

Figure 10: Input file for TORO II simulation of magnetic field problem.

```

SUBROUTINE USRPOT (POTBC,POTBCI,TNODE,RNODE,ZNODE,DUM, DNSET,
*               TIME,KSTEP,RCONST,ICONST)
C
C *****
C
C   IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C   DOUBLE PRECISION K
C
C   REAL POTBC,POTBCI,TNODE,RNODE,ZNODE,DUM,TIME,RCONST(*)
C   DIMENSION ICONST(*)
C
C   COMMON /CONCOM/ PI,AMUO
C
C   DATA CIR,CII,RC,ZC/1.0D6, 3.73203D6, 2.0D0, 0.0D0/
C   DATA PI /3.141592653/
C
C *****
C
C   THIS ROUTINE COMPUTES THE POTENTIAL VALUE AT A
C   BOUNDARY NODE, BASED ON AN ANALYTIC SOLUTION
C
C   THE FOLLOWING DATA IS PROBLEM DEPENDENT AND IS SET FOR A
C   PHASE ANGLE OF 75 DEGREES
C
C   CIR - CURRENT IN THE COIL, REAL COMPONENT (AMPS)
C   CII - CURRENT IN THE COIL, IMAGINARY COMPONENT (AMPS)
C   RC  - COIL RADIUS (M)
C   ZC  - Z COORDINATE OF THE COIL
C
C   THE COMPUTATION IS CARRIED OUT IN DOUBLE PRECISION
C
C   AMUO=4.0E-7*PI
C
C   K=4.0*RNODE*RC/((RNODE+RC)**2+(ZNODE-ZC)**2)
C   K=DMIN1(K,0.999999)
C   ELIPK=ELK(K)
C   ELIPE=ELE(K)

```

Figure 11: User subroutines for boundary condition in magnetic field problem.

```

IF(DABS(K).LT. 1.0D-10) THEN
  POTBC=0.0
  POTBCI=0.0
ELSE
  TERM=AMUO*DSQRT(RC/(RNODE*K))*((1.0-0.5*K)*ELIPK-ELIPE)/PI
  POTBC=CIR*TERM
  POTBCI=CII*TERM
END IF

C
RETURN
END

DOUBLE PRECISION FUNCTION ELK(P)
C
C *****
C
C ELLIPTIC INTEGRAL OF THE FIRST KIND
C
C *****
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
COMMON /CONCOM/ PI,AMUO
C
RATO=1.0
SUM=1.0
AN=1.0
AD=2.0
C
10 CONTINUE
RAT=AN/AD
TERM=RAT**2*RATO*P
SUMS=SUM
SUM=SUM+TERM
AN=AN+2.0
AD=AD+2.0
RATO=TERM
IF(DABS(SUMS-SUM)/SUM.GT.0.000001) GO TO 10
ELK=SUM*PI*0.5
C
RETURN
END

```

Figure 11: Continued: User subroutines for boundary condition in magnetic field problem.

```

      DOUBLE PRECISION FUNCTION ELE(P)
C
C *****
C
C      ELLIPTIC INTEGRAL OF THE SECOND KIND
C
C *****
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /CONCOM/ PI,AMUO
C
      RATO=1.0
      SUM=1.0
      AN=1.0
      AD=2.0
C
      10 CONTINUE
      RAT=AN/AD
      TERM= RAT**2*RATO*P
      SUMS=SUM
      SUM=SUM-TERM/AN
      AN=AN+2.0
      AD=AD+2.0
      RATO=TERM
      IF(DABS(SUMS-SUM)/SUM.GT.0.000001) GO TO 10
      ELE=SUM*PI*0.5
C
      RETURN
      END

```

Figure 11: Continued: User subroutines for boundary condition in magnetic field problem.

Figure 12: Real part of the magnetic potential for wire loop problem.

Figure 13: Magnitude of the magnetic flux for wire loop problem.

Figure 14: Difference in the computed and analytic solutions - real part of the magnetic potential.

Figure 15: Difference in the computed and analytic solutions - magnitude of the magnetic flux.

5.3 Problem 3 - Magnetic Induction Problem

One of the primary purposes of TORO II is the prediction of inductively coupled field problems. The third example is a geometrically simple case of induction that can be compared to a semi-analytic solution and other numerical simulations. The basic problem involves a cylindrical steel rod encircled by a square cross-section coil. The coil is energized with a time-harmonic current which induces an eddy current in the steel rod. The geometry for this particular example is taken from the work of Wang, *et al.* [18] and is shown in Figure 16 along with the computational mesh. The mesh was constructed with FASTQ [14] and was designed to correspond to the simulation reported in [18]. The rod is 40 mm in diameter with the 4×4 mm square coil having a mean radius of 24 mm. The rod extends to the top of the symmetric mesh ($H=150$ mm) and the lateral edge of the mesh was located at a radius of 250 mm. A boundary condition of zero potential (real and imaginary components) was applied on the lateral boundary of the mesh. Current to the coil was set at a peak value of 7×10^9 A/m² with a circular frequency of 120π rad/s or 60 Hz. The steel was assumed to have a relative magnetic permeability of 90.

Figure 16: Schematic and mesh for magnetic induction problem.

The input file for the TORO II simulation is listed in Figure 17. The axisymmetric nature of the problem leads to the conclusion that only one component of the magnetic potential is required for this problem. The problem is of the eddy current type and since it is linear, only one iteration is required.

```

TITLE
STEEL CYLINDER WITH COIL
ENDTITLE
MATERIAL,STEEL,CONDUCTOR,ISOTROPIC
ELECTRIC CONDUCTIVITY=3.0E6
MAGNETIC PERMEABILITY=1.131E-4
ENDMATERIAL
MATERIAL,AIR,DIELECTRIC,ISOTROPIC
MAGNETIC PERMEABILITY=1.256E-6
ENDMATERIAL
MATERIAL,COIL,CONDUCTOR,ISOTROPIC
CURRENT DENSITY=0.0,0.0,4.95E9
ELECTRIC CONDUCTIVITY=3.0E8
MAGNETIC PERMEABILITY=1.26E-6
ENDMATERIAL
PROBLEM DEFINITION
GEOMETRY=AXISYM
ELEMENT BLOCK=1,STEEL
ELEMENT BLOCK=2,AIR
ELEMENT BLOCK=3,COIL
PROBLEM TYPE=EDDY CURRENT, , , ,AZ
CURRENT TYPE=AC,376.99
BCTYPE=MAGNETIC POTENTIAL Z,1,0.0,0.0
MAXIMUM CONNECTIVITY=50
AUXILIARY FIELDS=CURRENT DENSITY,JOULE HEAT
ENDPROBLEM
SOLUTION,1,TIME HARMONIC
MAXIMUM ITERATIONS=1
MATRIX SOLVER=GMR
PREC TYPE=ILU
PRINTED OUTPUT=EXTENDED,1
ENDSOLUTION
POST
NODAL DATA=AZ_REAL,AZ_IMAG,JZ_REAL,JZ_IMAG,J_MAGN
ELEMENT DATA=J_HEAT
OUTPUT FREQ=1
ENDPOST
STOP

```

Figure 17: Input file for TORO II simulation of magnetic induction problem.

A semi-analytic (Green's function) solution to this problem was developed by Dodd [19] which was subsequently evaluated by Wang, *et al.* [18]. This solution could not be reproduced with the current method since the solution in [18] omitted a term in the axisymmetric formulation. The present method was compared to a numerical solution generated by the commercial code, EMAS [13]; the EMAS solution was produced by J. Kotulski, Dept. 9352. Shown in Figure 18 is a profile plot of the real and imaginary parts of the magnetic potential along the radial axis ($z=0$) of the domain. These profiles compare extremely well with the results from EMAS, including the location of the change in sign of both components within the rod. Peak values are essentially identical between the two codes. Figure 19 shows the Joule heating contours for this problem; comparison of this field with the result from EMAS was not pursued since EMAS does not directly compute this quantity.

Figure 18: Magnetic potential components along the radius of the induction problem.

Figure 19: Contour plot of the Joule heating field for the induction problem.

5.4 Problem 4 - Current Flow/Magnetostatic Problem

A magnetostatics problem in which the electric potential and two components of the magnetic potential are active is considered for the fourth example problem. The geometry of the problem is a right circular cylinder of a conducting material with a current distribution specified over the top surface. The problem is shown schematically in Figure 20 along with the finite element mesh used in the simulation. The cylinder is grounded along its lateral boundary and at the bottom. The steady electric current within the cylinder produces a magnetic field and a spatially varying Joule heating and Lorentz force. This is a very simplified model for the electromagnetic problem associated with the vacuum arc remelting process [20]. The simplicity of the geometry and boundary conditions allows an analytic solution to be generated for this problem based on a separation of variables technique. The components of the current density vector are given by

$$J_r = \frac{I}{2\pi R^2} j_0^1 \frac{\sinh[j_0^1 z/R]}{\cosh[j_0^1 L/R]} \frac{J_1(j_0^1 r/R)}{J_1(j_0^1)}$$

$$J_z = -\frac{I}{2\pi R^2} j_0^1 \frac{\cosh[j_0^1 z/R]}{\cosh[j_0^1 L/R]} \frac{J_0(j_0^1 r/R)}{J_1(j_0^1)}$$

and the magnetic induction field is described by

$$B_\theta = -\mu \left(\frac{I}{2\pi R} \right) \frac{\cosh[j_0^1 z/R]}{\cosh[j_0^1 L/R]} \frac{J_1(j_0^1 r/R)}{J_1(j_0^1)}$$

where I is the current, R is the cylinder radius, L is the height of the cylinder, J_0 and J_1

Figure 20: Schematic of cylindrical current flow/magnetostatic static problem.

are Bessel functions, j_0^1 is a zero of the Bessel function and μ is the magnetic permeability. These two relations may also be used to compute the Joule heating and Lorentz forces.

The TORO II simulation for this problem can be carried out in either of two ways. If the analytic current density given above is used, then a magnetostatics problem with the r and z components of the magnetic potential can be solved directly with J as a source term. A second approach uses the analytic current density to provide a flux boundary condition (to the top of the cylinder) for the steady current flow equation. The electric potential solution can then be used to compute the current field and then provide source terms to the magnetic potential equations. Both of these solution methods were used in TORO II. Shown in Figure 21 is the input file for the sequential solution to the current flow and magnetic potential problem. The input file for the magnetic potential with specified current field differs only in the few material and boundary condition cards shown on the right-hand-side of Figure 21. The current flux boundary condition for the electric potential equation is listed in Figure 22; the current density subroutine used in the second problem is shown in Figure 23.

Figures 24 and 25 show contour plots of the electric potential and the magnitude of the magnetic potential as produced from the TORO II simulation. The Joule heating field is contoured in Figure 26 and the magnitude of the Lorentz force is shown in Figure 27. The computed fields agree very well with the analytic solutions and are quite similar between the methods of solution. Table 3 lists a number of maximum values for several field quantities as obtained from the analytic and two numerical solutions. The quantitative agreement is very good; the Lorentz forces show the largest error of $\sim 6\%$ which is not unexpected since this is a derivative quantity that is maintained as an element variable.

Field Variable	Analytic Solution	V, A_r, A_z Solution	A_r, A_z Solution
Maximum V	9.444	9.420	—
Maximum J_r	1.264×10^5	1.263×10^5	1.263×10^5
Maximum J_z	-2.172×10^5	-2.141×10^5	-2.197×10^5
Maximum B_θ	69.06	72.06	72.09
Maximum Q_J	3.039	2.936	2.939
Maximum F_r	921.7	866.2	873.0
Maximum F_z	933.7	895.3	896.1

Table 3: Comparison of computed field variables with analytic solution.

```

TITLE
VACUUM ARC REMELT INGOT
END
MATERIAL, INGOT, CONDUCTOR
ELECTRIC CONDUCTIVITY=1.0E6
MAGNETIC PERMEABILITY=1.256E-6
                                $ CURRENT DENSITY=USER
END
PROBLEM DEFINITION
GEOM=AXISYMMETRIC
ELEMENT BLOCK=10, INGOT
PROBLEM TYPE=MAGNETOSTATIC, V, AX, AY    $ PROBLEM TYPE=MAGNETOSTATIC, , AX, AY
BC=ELECTRIC POT, 10, 0.0                $
BC=ELECTRIC POT, 20, 0.0                $
BC=ELECTRIC CURRENT, 30, USER            $
BC=MAGNETIC POT X, 100, 0.0
BC=MAGNETIC POT X, 300, 0.0
BC=MAGNETIC POT Y, 200, 0.0
MAXIMUM CONNECTIVITY=40
AUXILIARY FIELDS=CURRENT DENSITY,*
MAGNETIC FLUX, JOULE HEAT, LORENTZ FORCE
END
SOLUTION, 1, TIME INDEPENDENT
ITERATIVE METHOD=PICARD
MAXIMUM ITERATIONS=1
PRINTED OUTPUT=SUMMARY, 1
NORM VALUES=1.0, 1.0, 1.0
END
POST
NODAL DATA=V_REAL, AX_REAL, AY_REAL,*    $ NODAL DATA=AX_REAL, AY_REAL,*
    JX_REAL, JY_REAL, J_MAGN, BZ_REAL      $    JX_REAL, JY_REAL, J_MAGN, BZ_REAL
ELEMENT DATA=J_HEAT, LOR_FX, LOR_FY
OUTPUT FREQUENCY=1
END
STOP

```

Figure 21: Input file for TORO II simulation of current flow/magnetostatic problem.


```

SUBROUTINE USRFLX (FLUX,FLUXI,XS,YS,ZS,NNODES,IDSSET,TIME,
*                  KSTEP,RCONST,ICONST)
C
C *****
C
C REAL JI
C
C DIMENSION FLUX(*), FLUXI(*), XS(*), YS(*), ZS(*)
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C ELECTRIC CURRENT (FLUX) ON TOP BOUNDARY OF VAR INGOT
C CURRENT DISTRIBUTION IS BASED ON AN ANALYTIC PROFILE
C
C PROFILE IS GIVEN BY :
C    $J(r) = (JI/2 \times \pi \times RI^2) \times b_0 \times J_0(b_0 r / RI) / J_1(b_0)$ 
C
C JI = Input current magnitude
C RI = Ingot radius
C J_0 = Bessel function
C J_1 = Bessel function
C b_0 = First zero of Bessel function
C
C SET PARAMETERS FOR INGOT (210mm diameter ingot at 3.2kA)
C
C PI=ACOS(-1.0)
C RI=0.1042
C JI=3200.0
C B0=2.404825577
C
C COMPUTE CURRENT VALUES FOR ELEMENT EDGE
C
C COEF=(JI/(2.0*PI*RI**2))*B0
C DO 10 I=1,NNODES
C FLUX(I)=COEF*BESJO(B0*XS(I)/RI)/BESJ1(B0)
10 CONTINUE
C
C RETURN
C END

```

Figure 22: Boundary condition subroutine for current input to cylindrical geometry.

```

SUBROUTINE USRJ (CURX,CURY,CURZ,PANGLE,XIP,YIP,ZIP,NAME,NUMIPT,
*              TIME,KSTEP,RCONST,ICONST)
C
C *****
C
C CHARACTER*20 NAME
C REAL JI
C
C DIMENSION CURX(*), CURY(*), CURZ(*), PANGLE(*)
C DIMENSION XIP(*), YIP(*), ZIP(*)
C DIMENSION RCONST(*), ICONST(*)
C
C *****
C
C ELECTRIC CURRENT DENSITY WITHIN A CYLINDRICAL VAR INGOT
C CURRENT DISTRIBUTION IS BASED ON AN ANALYTIC PROFILE
C
C PROFILE IS GIVEN BY :
C
C 
$$J_r = \frac{(JI/2 \times \pi \times RI^{**2}) * b_0 * \sinh(b_0 z / RI) * J_1(b_0 r / RI)}{\cosh(b_0 L / RI) * J_1(b_0)}$$

C
C 
$$J_z = -\frac{(JI/2 \times \pi \times RI^{**2}) * b_0 * \cosh(b_0 z / RI) * J_1(b_0 r / RI)}{\cosh(b_0 L / RI) * J_1(b_0)}$$

C
C JI = Input current magnitude
C RI = Ingot radius
C J_0 = Bessel function
C J_1 = Bessel function
C b_0 = First zero of Bessel function
C L = Height of ingot
C
C SET PARAMETERS FOR INGOT (210mm diameter ingot at 3.2kA)
C
C PI=ACOS(-1.0)
C RI=0.1042
C JI=3200.0
C B0=2.404825577
C AL=0.3647

```

Figure 23: Source current subroutine for cylindrical geometry

```

C
C      COMPUTE CURRENT DENSITY VALUES
C
      COEF=(JI/(2.0*PI*RI**2))*B0
      ZL=B0*AL/RI
      DO 10 I=1,NUMIPT
      R=B0*XIP(I)/RI
      Z=B0*YIP(I)/RI
      CURX(I)= COEF*SINH(Z)*BESJ1(R)/(COSH(ZL)*BESJ1(B0))
      CURY(I)=-COEF*COSH(Z)*BESJ0(R)/(COSH(ZL)*BESJ1(B0))
10  CONTINUE
C
      RETURN
      END

```

Figure 23: Continued: Source current subroutine for cylindrical geometry.

Figure 24: Contour plot of the electric potential in the cylindrical geometry.

Figure 25: Contour plot of the magnitude of the magnetic potential in the cylindrical geometry.

Figure 26: Contour plot of the Joule heating field for magnetostatic problem.

Figure 27: Contour plot of the magnitude of the Lorentz force for the magnetostatic problem.

5.5 Problem 5 - Nonisothermal Magnetic Induction Problem

The last example is representative of an inductive heating problem that requires the use of a coupled thermal and electromagnetic analysis. The axisymmetric geometry is similar to the third example and consists of a cylindrical steel workpiece encircled by a hollow, square cross-section coil that carries an AC current. The current is of sufficient magnitude to produce a large eddy current in the workpiece; Joule heating within the workpiece raises the temperature beyond the Curie temperature resulting in a large change in the magnetic permeability of the part. This intimate coupling between the temperature and the magnetic properties necessitates the use of the PVM option for transferring data between TORO II and a thermal code such as COYOTE II [1,2].

A schematic of the problem is shown in Figure 28 along with the finite element mesh of the part, coil and surrounding air space. The coil is driven with an 11 kA current at 4400 Hz. The workpiece is 1050 steel and the coil is copper; the Curie temperature for the steel is assumed to be 770° C. The electromagnetic and thermal properties for the two materials are available from the input listings shown in Figures 29 and 30. Note that the magnetic permeability for the workpiece was specified as a variable function of temperature and included the sharp transition at the Curie temperature. This problem was run using the same mesh for TORO II and COYOTE II. Data from the two codes was exchanged at every time step. The thermal code was designated as the master code since the time constant for the thermal diffusion problem is significantly longer than the

Figure 28: Schematic of nonisothermal eddy current problem.

electromagnetic response. The thermal code used an adaptive timestep, trapezoid rule integration method that accurately followed the temperature response of the part. The time-harmonic solution to the eddy current problem provided a stepwise update to the volume heating within the part. Thermal boundary conditions on the part consisted of simple radiation conditions; a more accurate method would employ an enclosure radiation model.

The simulation was run for a time interval of 6.0 seconds (heating for 5.0 seconds) with the part reaching the Curie temperature at about 3.0 seconds. Shown in Figure 31 is a time history of the temperature at the midheight of the workpiece surface. Contour plots of the part temperature field at two times during the inductive heating process are shown in Figure 32. These fields correspond to the Joule heating and magnetic potential profiles shown in Figures 33 and 34. It is very apparent that the permeability transition at the Curie temperature has a significant effect on the EM fields and consequently on the temperature penetration of the workpiece.

```
TITLE
PLAIN CYLINDER IN A COIL
AXISYMMETRIC TEST MODEL FOR EM HEATING
ELECTROMAGNETICS INPUT
END
MATERIAL,CYLINDER,CONDUCTOR
ELECTRIC CONDUCTIVITY=1.0E4
MAGNETIC PERMEABILITY=VFUN,10
END
MATERIAL,COIL,CONDUCTOR
ELECTRIC CONDUCTIVITY=5.8E5
MAGNETIC PERMEABILITY=1.26E-8
CURRENT DENSITY=0.,0.,3580.0
END
MATERIAL,AIR,DIELECTRIC
MAGNETIC PERMEABILITY=1.26E-8
END
PROBLEM DEFINITION
GEOMETRY=AXISYMMETRIC
ELEMENT BLOCK=10,CYLINDER
ELEMENT BLOCK=20,COIL
ELEMENT BLOCK=30,AIR
PROBLEM TYPE=EDDY CURRENT,, ,AZ
CURRENT TYPE=AC,27700.0
```

Figure 29: Input file for TORO II simulation of nonisothermal eddy current problem.

```

BC=MAGNETIC POTENTIAL Z,10,0.
BC=MAGNETIC POTENTIAL Z,20,0.
AUXILIARY FIELDS=CURRENT DENSITY,ELECECTRIC FIELD,MAGNETIC FLUX,*
    MAGNETIC FIELD,JOULE HEAT,LORENTZ FORCE
MAXIMUM CONNECTIVITY=30
EXTERNAL DATA=CODES
EXTERNAL CODE,MASTER=COYOTE2,SLAVE=TORO2
EXTERNAL NODAL FIELD=READ,TEMP
EXTERNAL ELEMENT FIELD=WRITE,J_HEAT
END
SOLUTION,1,TIME HARMONIC
MAXIMUM ITERATIONS=100
CONVERGENCE TOLERANCE=0.0
MATRIX SOLVER=GMR
PRECONDITIONER TYPE=ILU
EXTERNAL DATA EXCHANGE=STEPS,1
END
POST
NODAL DATA=A_MAGN,A_PHASE,BX_REAL,BY_REAL,B_MAGN,B_PHASE,*
    EZ_REAL,JZ_REAL,JZ_IMAG,J_MAGN,J_PHASE
ELEMENT DATA=J_HEAT,LOR_FX,LOR_FY
OUTPUT FREQUENCY=1
END
VARIABLE FUNCTION=10,T
200.0,2.52E-5
1043.0,2.52E-5
1050.0,1.26E-6
2000.0,1.26E-6
END
STOP

```

Figure 29: Continued: Input file for TORO II simulation of nonisothermal eddy current problem.


```

TITLE
PLAIN CYLINDER IN A COIL
AXISYMMETRIC TEST MODEL FOR EM HEATING
END
MATERIAL,CYLINDER
DENSITY=7.85
SPECIFIC HEAT=0.467
CONDUCTIVITY=.4925
INITIAL TEMPERATURE=300.
EMISSIVITY=0.5
VOLUME HEATING=EXTERNAL
END
MATERIAL,COIL
DENSITY=8.3
SPECIFIC HEAT=0.419
CONDUCTIVITY=3.72
INITIAL TEMPERATURE=300.
EMISSIVITY=0.6
VOLUME HEATING=EXTERNAL
END
MATERIAL,AIR
DENSITY=1.205E-3
SPECIFIC HEAT=1.006
CONDUCTIVITY=2.5E-4
INITIAL TEMPERATURE=300.
END
PROBLEM DEFINITION
GEOMETRY=AXISYMMETRIC
SIGMA=5.669E-12
BC=TEMPERATURE,10,300.
BC=TEMPERATURE,20,300.
BC=RADIATION,10,COEF=1.0,,TREF=300.
BC=RADIATION,20,COEF=1.0,,TREF=300.
ELEMENT BLOCK=10,CYLINDER
ELEMENT BLOCK=20,COIL
ELEMENT BLOCK=30,AIR
MAXIMUM CONNECTIVITY=20

```

Figure 30: Input file for COYOTE II simulation of nonisothermal eddy current problem.

```

EXTERNAL DATA=CODES
EXTERNAL CODE,MASTER=COYOTE2,SLAVE=TOR02
EXTERNAL NODAL FIELD=WRITE,TEMP
EXTERNAL ELEMENT FIELD=READ,VOLHEAT
END
SOLUTION,1,TIME DEPENDENT
INITIAL TIME=0.
FINAL TIME=6.0
TIME STEP=0.1
INTEG METH=TRAP
CONVERGENCE TOLERANCE=0.0
NUMBER TIME STEPS=75
EXTERNAL DATA EXCHANGE=STEPS,1
END
POST
NODAL DATA=TEMPERATURE
OUTPUT FREQUENCY=1
END
STOP

```

Figure 30: Continued: Input file for COYOTE II simulation of nonisothermal eddy current problem.

Figure 31: Temperature history for the surface of the workpiece.

Figure 32: Temperature contours for nonisothermal eddy current problem, workpiece only.

Figure 33: Joule heating contours for nonisothermal eddy current problem, workpiece only.

Figure 34: Magnetic potential contours (real part) for nonisothermal eddy current problem.

6 References

1. 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)
2. D. K. Gartling and R. E. Hogan "COYOTE II - A Finite Element Computer Program for Nonlinear Heat Conduction Problems, Part II - User's Manual," SAND94-1179, Sandia National Laboratories, Albuquerque, NM (1994)
3. D. K. Gartling, "TORO II - A Finite Element Computer Program for Nonlinear, Quasi-static Problems in Electromagnetics, Part I - Theoretical Background," SAND95-2472, Sandia National Laboratories, Albuquerque, NM (1996)
4. L. A. Schoof and V. R. Yarberry, "EXODUS II - A Finite Element Data Model," SAND92-2137, Sandia National Laboratories, Albuquerque, NM (1994)
5. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, "PVM 3 User's Guide and Reference Manual," ORNL/TM-12187, Oak Ridge National Laboratory, Oak Ridge, TN (1994)
6. D. K. Gartling, "MERLIN II - A Computer Program to Transfer Solution Data Between Finite Element Meshes," SAND89-2989, Sandia National Laboratories, Albuquerque, NM (1991)
7. D. K. Gartling and S. E. Gianoulakis, "The Coupling of Mechanics Codes Via PVM," SAND96-yyyy, (in preparation) Sandia National Laboratories, Albuquerque, NM (1996)
8. P. R. Schunk and J. N. Shadid, "Iterative Solvers in Implicit Finite Element Codes," SAND92-1158, Sandia National Laboratories, Albuquerque, NM (1992)
9. 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)
10. "NetCDF User's Guide; An Interface for Data Access," Version 1.11, Unidata Program Center (1991)
11. Y. Saad, "SPARSKIT: A Basic Tool for Sparse Matrix Computations," Tech. Report, Research Institute for Advanced Computer Science, NASA Ames, Moffitt Field, CA (1990)
12. G. D. Sjaardema, "Overview of the Sandia National Laboratories Engineering Analysis Code Access System," SAND92-2292, Sandia National Laboratories, Albuquerque, NM (1993)

13. "Semi-Guided Exercises for MSC/EMAS," MSC/EMAS Version 2, MacNeal/Schwendler Corp. (1992)
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. J. A. Stratton, *Electromagnetic Theory*, McGraw Hill, New York, NY (1941)
18. K. F. Wang, S. Chandrasekar and H. T. Y. Yang, "Finite Element Simulation of Induction Heat Treatment," *J. Mat. Eng. and Performance*, **1**, 97-112 (1992)
19. C. V. Dodd, "Solutions to Electromagnetic Induction Problems," ORNL TM-1842, Oak Ridge National Laboratory, Oak Ridge, TN (1967)
20. F. J. Zanner and L. A. Bertram, "Computational and Experimental Analysis of a 6 wt% Nb Vacuum Consumable Arc Remelted Ingot," SAND80-1156, Sandia National Laboratories, Albuquerque, NM (1981)

Appendix A - Summary of Input Commands

In this section all of the data blocks and associated data cards recognized by TORO II 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 type, *material model*

☐

.

.

ELECtric CONDUCTivity= $\sigma_{11}, \sigma_{22}, \sigma_{33}$

☐

ELECtric CONDUCTivity=USER

☐

ELECtric CONDUCTivity=VFUNction, idvar1, idvar2, idvar3

☐

.

.

MAGnetic PERMeability= $\mu_{11}, \mu_{22}, \mu_{33}$

☐

MAGnetic PERMeability=USER

☐

MAGnetic PERMeability=VFUNction, idvar1, idvar2, idvar3

☐

.

.

ELECtric PERMittivity= $\epsilon_{11}, \epsilon_{22}, \epsilon_{33}$

☐

ELECtric PERMittivity=USER

☐

ELECtric PERMittivity=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

.

.

CURrent DENsity= J_x, J_y, J_z, θ
CURrent DENsity=USER
CURrent DENsity=TFUNction, idtim1, idtim2, idtim3

ELECtric CHARge= ρ
ELECtric CHARge=USER
ELECtric CHARge=VFUNction, idvar

INITial MAGnetic POTential= $A_{x_{init}}, A_{y_{init}}, A_{z_{init}}$

INITial ELECtric POTential= V_{init}

ENDmaterial □

Problem Definition Data Block :

PROBLEM DEFinition, *format* □

GEOMetry=type □

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

PROBLEM TYPE=description, electric dof, *magnetic dof x,*
magnetic dof y, magnetic dof z □

HYSteresis LOSS=*option*

CURrent TYPE=description, *frequency*

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

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

EXternal DATA=data source

.

EXternal CODES, MASTER=code name1, SLAVE=code name2,
MTRANslator=code name3, STRANslator=code name4

.

EXternal NODal FIELd=i/o status, name1, name2, ... , namen

.

EXternal ELEMEnt FIELd=i/o status, name1, name2, ... , namen

.

PRINTed OUTput=type

.

OUTput LOCations=e1, e2, e3, ... , en TO em, ...

.

AUXiliary FIELds=name1, name2, ... namen

.

MAXimum CONnectivity=maxcon1, maxcon2

.

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*

.

MASS MATRix=type

.
.
TIME STEP OPTion=type

.
.
INTEgration TOLerance=integ tol

.
.
NORM VALues= V_{norm} , Ax_{norm} , Ay_{norm} , Az_{norm}

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

.
.
TIME STEP= Δt

☐

.
.
INITial TIME= t_{init}

.
.
FINal TIME= t_{final}

☐

.
.
NUMBER TIME STEPs=nstep

☐

.
.
MINimum TIME STEP= Δt_{min}

.
.
MAXimum TIME STEP= Δt_{max}

.
.
CONVergence TOLerance=tol

.
.
RELaxation FACTor= α

.
.
MAXimum ITERations=itermax

.
.
EXTernal DATA EXchange=variable, value

PRINted OUTput=option, frequency

.

MATRix SOLver=method, *krylov subspace*

.

PREConditioner TYPE=option, *polynomial order*

.

L2 NORM= L_2

.

RESidual NORM=resid

.

MAXimum MATRix ITERations=matrix iter

.

ENDsolution

□

Post-processing Data Block :

POST

.

NODal DATA=name1, name2,....

.

ELEMent DATA=name1, name2,....

.

GLOBal DATA=name1, name2,....

.

OUTput FREQuency=nsteps

.

OUTput TIMEs= $t_1, t_2, \dots t_n$

.

OUTput TIME STEP= Δt_{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, var id

$\xi_1, f(\xi_1)$

$\xi_2, f(\xi_2)$

$\xi_3, f(\xi_3)$

.

.

.

$\xi_n, f(\xi_n)$

.

.

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 TORO II.

Quantity	MKS Units
Length	meter (m)
Time	second (s)
Mass	kilogram (kg)
Force	Newton (N)
Energy	joules (J)
Temperature	Centigrade (C) or Kelvin (K)
Density	kg/m ³
Power	J/s (Watt)
Capacitance	Farad (F)
Resistance	Ohm (Ω)
Inductance	Henry (H) = Ohm-s
Charge	Coulomb (C)
Conductivity (σ)	1/Ohm-m = mho/m
Permittivity (ϵ)	Farads/m
Permeability (μ)	Henry/m
Reluctivity (ν)	m/Henry
Electric Field (E)	Volts/m
Magnetic Field (H)	Amp/m
Magnetic Flux (B)	Tesla= Webers/m ²
Vector Potential (A)	Tesla-m
Scalar Potential (V)	Volts
Current Density (J)	Amps/m ²
Charge Density (ρ)	Coulomb/m ³
Dielectric Constant (ϵ_0)	8.854×10^{-12} F/m
Free Space Permeability (μ_0)	$4\pi \times 10^{-7}$ H/m

Appendix C - Common Block and Array Storage

The labeled common blocks and major arrays used in TORO II 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 TORO II 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 TORO II 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 TORO II

Problem Data

```

COMMON /PRBDAT/  IGEOM,ITMDEP,IEXTFL,IADAPT,IFLUX,IPOST
COMMON /EMDAT/   IEMTYP,IGAUGE
COMMON /CURDAT/  IACDC,OMEGA,IHYSTR
COMMON /VARDAT/  IELECT,IMAGX,IMAGY,IMAGZ,NMAG

```

<u>Variable</u>	<u>Description</u>
IGEOM	Problem geometry flag, 1=2D, 2=axisymmetric, 3=3D
ITMDEP	Time dependence flag, 0=steady, 1=transient
IEXTFL	External variable file 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
IEMTYP	Flag for type of EM problem, 1=electrostatic, 2=current flow, 3=magnetostatic with current flow, 4=magnetostatic with no current flow, 5=eddy current
IGAUGE	Flag for gauge condition, 0=omitted, 1=included
IACDC	Flag for current type, 0=DC current, 1=AC current
OMEGA	Frequency for AC current
IHYSTR	Flag for hysteresis loss, 0=omitted, 1=included
IELECT	Flag for electric potential variable, 0=omitted, 1=included
IMAGX	Flag for magnetic potential x variable, 0=omitted, 1=included
IMAGY	Flag for magnetic potential y variable, 0=omitted, 1=included
IMAGZ	Flag for magnetic potential z variable, 0=omitted, 1=included
NMAG	Number of active magnetic variables

Sizing Data

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

<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
MXDFEL	Maximum number of degrees of freedom per element
MXATEL	Maximum number of element attributes per element
NUMBLK	Number of element blocks
NUMSLN	Number of solution blocks

Boundary Condition Data

```
COMMON /BCDAT/ NUMNS,LNSNL,LNSDF,NUMSS,LSEL,LSSDF,NUMNBC,NUMSBC,
*              MXNSNL,MXNSDF,MXSEL,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
LSEL	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),NFACE(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
NFACE	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/ NTIME,MXPTTF,NPTSTF,NVARF,MXPTVF,NPTSVF
COMMON /USRDAT/ NUSRR,NUSRI
```

<u>Variable</u>	<u>Description</u>
NTIME	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

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
COMMON /RSTDAT/IRSTRT,NSTEPS,RDTIME
COMMON /PCGDAT/ NDOFND(2),MXCON(2)
COMMON /EQNDAT/ NNZERO(2),NEQNS(2)
```

<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
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
NDOFND	Multiple degrees of freedom per node flag, 1=1 dof, 0=n dof, each equation system
MXCON	Maximum number of equations connected to any equation each equation system
NNZERO	Number of nonzero entries in global matrix, each equation system
NEQNS	Number of equations, each equation system

External Variable Data

```
COMMON /EXTDAT/ NUMXNV(2),NUMXE(2),IPTXNV(2,10),IPTXE(2,10)
COMMON /EXTNAM/ NAMXNV(2,10),NAMEV(2,10)
COMMON /PVMDAT/ NAMTSK(4),IDTSK(4,0:0),NUMTSK,IMASTR
```

<u>Variable</u>	<u>Description</u>
NUMXNV(1)	Number of external nodal variables to read
NUMXNV(2)	Number of external nodal variables to write
NUMXEV(1)	Number of external element variables to read
NUMXEV(2)	Number of external element variables to write
IPTXNV	Flags for nodal variables to read and write
IPTXEV	Flags for element variables to read and write
NAMXNV	Names of nodal variables to read and write (CHARACTER*8)
NAMXEV	Names of element variables to read and write (CHARACTER*8)
NAMTSK	Names of externally coupled codes (CHARACTER*8)
IDTSK	Task ids for externally coupled codes
NUMTSK	Number of externally coupled codes
IMASTR	Flag to indicate master code, 0=TORO II slave code, 1=TORO II master code

Norm Data

```

COMMON /MAXVAR/ AXMAX,AYMAX,AZMAX,VMAX,IFNDMX
COMMON /NORMS/  ANRMX,ANRMY,ANRMZ,ANRMV,
*               BNRMX,BNRMY,BNRMZ,BNRMV,
*               AXDIFF,AYDIFF,AZDIFF,VDIFF

```

<u>Variable</u>	<u>Description</u>
AXMAX	Maximum x magnetic potential in the current solution vector
AYMAX	Maximum y magnetic potential in the current solution vector
AZMAX	Maximum z magnetic potential in the current solution vector
VMAX	Maximum electric potential in the current solution vector
IFNDMX	Flag for computation/input of norms
ANRMX	Norm on x magnetic potential change between iterations/time steps
ANRMY	Norm on y magnetic potential change between iterations/time steps
ANRMZ	Norm on z magnetic potential change between iterations/time steps
ANRMV	Norm on electric potential change between iterations/time steps
BNRMX	Norm on x magnetic potential integration error
BNRMY	Norm on y magnetic potential integration error
BNRMZ	Norm on z magnetic potential integration error

BNRMV	Norm on electric potential integration error
AXDIFF	Maximum x magnetic potential change between steps or iterations
AYDIFF	Maximum y magnetic potential change between steps or iterations
AZDIFF	Maximum z magnetic potential change between steps or iterations
VDIFF	Maximum electric potential change between steps or iterations

Postprocessing Data

```
COMMON /EXODAT/ NVARNP,NVAREL,NVARGL
COMMON /EXONAM/ NAMECO(3),NAMEEL(20),NAMENV(50),NAMEEV(5),NAMEGV(5)
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
NAMECO	Names for coordintes (CHARACTER*8)
NAMEEL	Names for elements (CHARACTER*8)
NAMENV	Names for nodal variables (CHARACTER*8)
NAMEEV	Names for 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

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 ED2, ED3	Quadrature weights for each element at each integration point. One-dimensional, element edge shape functions evaluated at the quadrature points. The first index is the shape function (node) number; the second index is the quadrature point number.
ED2G, ED3G	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 HX8F,HX20F	Three-dimensional, tetrahedral, wedge and hexahedral, 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.
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 FC3,FC6 FC4,FC8	Quadrature weights for each element at each integration point. 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.

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 /CENTER/ STRPT(20,3)
COMMON /TRNSFM/ EXTTR(3,3),EXTQD(4,4),EXTTT(4,4),EXTWD(6,6),
*                EXTHX(8,8)

```

<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 n in the array name separates the data according to element geometry. Arrays with n=1 contain quadrature data for one-dimensional, line elements, n=2 is for two-dimensional quadrilateral regions, n=3 is for three-dimensional hexahedrons, n=4 is for two-dimensional triangular areas, n=5 is for three-dimensional tetrahedrons, n=6 is for three-dimensional wedges, n=7 is for triangular shells and n=8 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.
EXTmmmm	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 TORO II 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)
DIMENSION ELATRB(MXATEL,NUMEL)
DIMENSION DATBLK(5,NUMBLK), IPTBLK(20,NUMBLK)
DIMENSION ICONBK(MXNDEL*MXELBK)
DIMENSION BLKMAT(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
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) - Unused DATBLK(5,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) - Number electric dof/node IPTBLK(9,NUMBLK) - Number of magnetic dof/node IPTBLK(10,NUMBLK) - Unused IPTBLK(11,NUMBLK) - Unused

	IPTBLK(12,NUMBLK) - Unused
	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
BLKMAT	Material name for element block (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)
DIMENSION DATNS(3,NUMNS), KPTNS(5,NUMNS)
DIMENSION DATSS(3,NUMSS), KPTSS(5,NUMSS), KPTELM(NUMEL)

```

<u>Array</u>	<u>Description</u>
IDNS	Node set ids
NNNS	Node set node counts
NDFNS	Node set distribution factor counts
IPNNS	Node set node pointers
IPDFNS	Node set distribution factor pointers
LSNNS	Node set node list
DFNS	Node set distribution factor list
IDSS	Side set ids
NESS	Side set element counts
NDFSS	Side set distribution factor counts
IPESS	Side set element pointers
IPDFSS	Side set distribution factor pointers
LSESS	Side set element list
LSSSS	Side set side list
DFSS	Side set distribution factor list
DATNS	Node set boundary data

	DATNS(1,NUMNPS) - Value of boundary condition (real)
	DATNS(2,NUMNPS) - Value of boundary condition (imaginary)
	DATNS(3,NUMNPS) - Multiplying factor for boundary value
KPTNS	Node set boundary pointers
	KPTNS(1,NUMNPS) - Node set id
	KPTNS(2,NUMNPS) - Type of bc, 1=scalar potential, 2=point source, 3=magnetic potential x, 4=magnetic potential y, 5=magnetic potential z, 6=gauge
	KPTNS(3,NUMNPS) - User subroutine flag
	KPTNS(4,NUMNPS) - Function id for bc, +id=variable function, -id=time function
	KPTNS(5,NUMNPS) - Constant bc flag
DATSS	Side set boundary data
	DATSS(1,NUMESS) - Value of boundary condition (real)
	DATSS(2,NUMESS) - Value of boundary condition (imaginary)
	DATSS(3,NUMESS) - Multiplying factor for boundary value
KPTSS	Side set boundary pointers
	KPTSS(1,NUMESS) - Side set id
	KPTSS(2,NUMESS) - Type of bc, 1=current flux, 2=unused, 3=magnetic field x, 4=magnetic field y, 5=magnetic field z
	KPTSS(3,NUMESS) - User subroutine flag
	KPTSS(4,NUMESS) - Function id for bc, +id=variable function, -id=time function
	KPTSS(5,NUMESS) - Constant bc flag
KPTLM	Number of side set boundary conditions for each element

Material Data

```

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

```

<u>Array</u>	<u>Description</u>
DATMAT	Material property values
	DATMAT(1,NUMMAT) - Conductivity (xx component)
	DATMAT(2,NUMMAT) - Conductivity (yy component)
	DATMAT(3,NUMMAT) - Conductivity (zz component)

Function and Constant Data

```
DIMENSION DATTFN(2,NPTSTF), IPTTFN(3,NTIMF), TFNVAL(NTIMF)
DIMENSION DATVFN(2,NPTS VF), IPTVFN(3,NVARF), PTVFN(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,NPTS VF) - Variable value (temperature) DATVFN(2,NPTS VF) - 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
PTVFN	Flag for dependent variable in function (CHARACTER*1)
RCONST	User defined constants (real)
ICONST	User defined constants (integer)

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

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 NELPND(NUMNOD), NDLST(NUMNOD), NELST(NUMEL)
DIMENSION NDFNOD(NUMNOD), NFIRST(NUMNOD), NDFELM(NUMEL)
DIMENSION NCHECK(NUMNOD)
DIMENSION NK(MXDFEL), INK(MXDFEL)
DIMENSION ITMP(MXCON), IKTMP(MXCON,NUMDOF), IJK(MXCON*NUMNOD+1)
DIMENSION VN(NUMNOD), VPNP1(NUMNOD), VDOT(NUMNOD)
DIMENSION AXN(NUMNOD), AXPNP1(NUMNOD), AXDOT(NUMNOD)
DIMENSION AYN(NUMNOD), AYPNP1(NUMNOD), AYDOT(NUMNOD)
DIMENSION AZN(NUMNOD), AZPNP1(NUMNOD), AZDOT(NUMNOD)
DIMENSION SCRTCH(NUMNOD*NUMVAR)
DIMENSION BNDVAL(MXNNPS), BNDVALI(MXNNPS)
DIMENSION BIGK(NNZERO), BIGF(NUMNOD)
DIMENSION TEMP(NUMNOD)
DIMENSION VJX(NUMNOD), VJY(NUMNOD), VJZ(NUMNOD)
DIMENSION VBX(NUMNOD), VBY(NUMNOD), VBZ(NUMNOD)
DIMENSION VHX(NUMNOD), VHY(NUMNOD), VHZ(NUMNOD)
DIMENSION VEX(NUMNOD), VEY(NUMNOD), VEZ(NUMNOD)
DIMENSION VFX(NUMEL), VFY(NUMEL), VFZ(NUMEL), VQ(NUMEL)
DIMENSION NDFLXE(NUMNOD), NDFLXM(NUMDOD)
DIMENSION VTEMP(NUMNOD)
  
```

<u>Array</u>	<u>Description</u>
NELPND	Number of elements per node
NDLST	Temporary node list (presolution)
NELST	Temporary element list (presolution)
NDFNOD	Number of degrees of freedom per node
NFIRST	Number of first degree of freedom at each node
NDFELM	Number of degrees of freedom per element
NCHECK	Scratch check array
NK	Element pointers for sparse matrix storage
INK	Element pointers for sparse matrix storage
ITMP	Temporary pointers for sparse matrix storage
IKTMP	Temporary pointers for sparse matrix storage

IJK	Pointers for sparse matrix storage
VN	Scalar potential solution
AXN	
AYN	Vector potential solution
AZN	
VPNP1	Predicted scalar potential
AXPNP1	
AYPNP1	Predicted vector potential
AZPNP1	
VDOT	Scalar potential rate
AXDOT	
AYDOT	Vector potential rate
AZDOT	
SCRATCH	Scratch solution vector
BNDVAL	Boundary conditions for nodal points (real)
BNDVALI	Boundary conditions for nodal points (imaginary)
BIGK	Global matrix (sparse matrix format)
BIGF	Global force vector
TEMP	Temperature field (external)
VJX	
VJY	Nodal point current density components
VJZ	
VBX	
VBY	Nodal point magnetic flux components
VBZ	
VHX	
VHY	Nodal point magnetic field components
VHZ	
VEX	
VEY	Nodal point electric field components
VEZ	
VFX	
VFY	Element Lorentz force components
VFZ	
VQ	Element Joule heating
NDFLXE	Number of electric fields per node
NDFLXM	Number of magnetic fields per node
VTEMP	Temporary vector for flux output

Not all of the above solution vectors will be defined for all electromagnetic problem types but will occur as demanded by code input. Also, for time-harmonic eddy current simulations additional vectors containing the imaginary part of each solution vector will be defined; the imaginary component of each vector retains the same base name and is

suffixes with an I.

Postprocessing Data

```
DIMENSION GLBLV(NVARGL)
DIMENSION IPPNOD(50), IPPELM(5), IPPGLB(5)
```

<u>Array</u>	<u>Description</u>
GLBLV	Global variable vector
IPPNOD	Output flags for nodal variables (0=omitted, 1=output)
	IPPNOD(1) - Electric potential (real)
	IPPNOD(2) - Electric potential (imaginary)
	IPPNOD(3) - Electric potential (magnitude)
	IPPNOD(4) - Electric potential (phase angle)
	IPPNOD(5) - Magnetic potential, x component (real)
	IPPNOD(6) - Magnetic potential, x component (imaginary)
	IPPNOD(7) - Magnetic potential, y component (real)
	IPPNOD(8) - Magnetic potential, y component (imaginary)
	IPPNOD(9) - Magnetic potential, z component (real)
	IPPNOD(10) - Magnetic potential, z component (imaginary)
	IPPNOD(11) - Magnetic potential (magnitude)
	IPPNOD(12) - Magnetic potential (phase angle)
	IPPNOD(13) - Current density, x component (real)
	IPPNOD(14) - Current density, x component (imaginary)
	IPPNOD(15) - Current density, y component (real)
	IPPNOD(16) - Current density, y component (imaginary)
	IPPNOD(17) - Current density, z component (real)
	IPPNOD(18) - Current density, z component (imaginary)
	IPPNOD(19) - Current density (magnitude)
	IPPNOD(20) - Current density (phase angle)
	IPPNOD(21) - Magnetic flux, x component (real)
	IPPNOD(22) - Magnetic flux, x component (imaginary)
	IPPNOD(23) - Magnetic flux, y component (real)
	IPPNOD(24) - Magnetic flux, y component (imaginary)
	IPPNOD(25) - Magnetic flux, z component (real)
	IPPNOD(26) - Magnetic flux, z component (imaginary)
	IPPNOD(27) - Magnetic flux (magnitude)
	IPPNOD(28) - Magnetic flux (phase angle)
	IPPNOD(29) - Magnetic field, x component (real)
	IPPNOD(30) - Magnetic field, x component (imaginary)
	IPPNOD(31) - Magnetic field, y component (real)

IPPNOD(32) - Magnetic field, y component (imaginary)
 IPPNOD(33) - Magnetic field, z component (real)
 IPPNOD(34) - Magnetic field, z component (imaginary)
 IPPNOD(35) - Magnetic field (magnitude)
 IPPNOD(36) - Magnetic field (phase angle)
 IPPNOD(37) - Electric field, x component (real)
 IPPNOD(38) - Electric field, x component (imaginary)
 IPPNOD(39) - Electric field, y component (real)
 IPPNOD(40) - Electric field, y component (imaginary)
 IPPNOD(41) - Electric field, z component (real)
 IPPNOD(42) - Electric field, z component (imaginary)
 IPPNOD(43) - Electric field (magnitude)
 IPPNOD(44) - Electric field (phase angle)
 IPPNOD(45) - Magnetic stream function (real)
 IPPNOD(46) - Magnetic stream function (imaginary)
 IPPNOD(47) - Unused
 IPPNOD(48) - Unused
 IPPNOD(49) - Unused
 IPPNOD(50) - Unused

IPPELM Output flags for element variables (0=omitted, 1=output)
 IPPELM(1,NUMBLK) - Joule heating
 IPPELM(2,NUMBLK) - Lorentz force, x component
 IPPELM(3,NUMBLK) - Lorentz force, y component
 IPPELM(4,NUMBLK) - Lorentz force, z component
 IPPELM(5,NUMBLK) - Unused

IPPGLB Output flags for global variables (0=omitted, 1=output)
 IPPGLB(1) - Timestep
 IPPGLB(2) - CG iterations
 IPPGLB(3) - Unused
 IPPGLB(4) - Unused
 IPPGLB(5) - Unused

Appendix D - External Mesh Generator File Contents

The file structure recognized by TORO II for reading data from an external mesh generator is based on the EXODUS II [4] standard. The contents of the file are summarized below in the order in which data is normally read by TORO II. 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 [4]. File opening and closing is also handled through a subroutine interface. TORO II 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      ELATRB(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 E - Post-Processing File Contents

The file structure employed by TORO II to write data for an external post-processing file is based on the EXODUS II [4] standard. The contents of the file include all of the data records from the mesh generation file (Appendix C) 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 TORO II in subroutines EXOSET, SETIC, EXOWRTA, EXOWRTB, EXOWRTC and EXOWRTD.

```
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
```

Distribution:

University of Texas (3)
Department of Aerospace Engineering
and Engineering Mechanics
Austin, Texas 78712
Attn: E. B. Becker
G. F. Carey
L. J. Hayes

R. I. Tanner
University of Sydney
Department of Mechanical Engineering
Sydney, New South Wales 2006
Australia

T. J. R. Hughes
Stanford University
Division of Applied Mechanics
Department of Mechanical Engineering
Stanford, California 94305

P. J. Roache
Ecodynamics Research Associates, Inc.
P.O. Box 9229
Albuquerque, New Mexico 87119

P. M. Gresho
Lawrence Livermore National
Laboratory
P. O. Box 808 L-103
Livermore, California 94550

M. S. Engelman
Fluid Dynamics International
500 Davis Street, Suite 600
Evanston, Illinois 60201

R. E. Nickell
Applied Science and Technology
16630 Sagewood Lane
Poway, California 92064

Purdue University (2)
School of Engineering and Technology
799 W. Michigan Street
Indianapolis, Indiana 46202
Attn: H. Akay
A. Ecer

J. N. Reddy
Texas A & M University
Department of Mechanical Engineering
College Station, Texas 77843

M. S. Ingber
Department of Mechanical Engineering
University of New Mexico
Albuquerque, New Mexico 87131

P. R. Dawson
Cornell University
Sibley School of Mechanical and
Aerospace Engineering
254 Upson Hall
Ithaca, New York 14853

J. J. Derby
University of Minnesota
Department of Chemical Engineering
and Materials Science
151 Amundson Hall
412 Washinton Avenue
Minneapolis, Minnesota 55455

D. L. Lesser
ISB-1 K7-15
Pacific Northwest Laboratory
Richland, Washington 99352

Sandia Internal:

MS1134 1833 B. K. Damkroger
MS1134 1833 J. B. Kelly
MS1134 1833 M. C. Maguire
MS1134 1833 F. J. Zanner
MS0841 9100 P. J. Hommert
MS0828 9102 R. D. Skocypec
MS0833 9103 J. H. Biffle

Route to: 9112, 9113, 9116

MS0828 9104 E. D. Gorham

Route to: 9111, 9114, 9115

MS0826 9111 W. Hermina
MS0826 9111 K. S. Chen
MS0826 9113 M. Christon
MS0826 9111 D. K. Gartling (30)
MS0826 9111 R. C. Givler
MS0826 9111 M. W. Glass
MS0826 9111 C. E. Hickox
MS0826 9111 S. N. Kempka
MS0826 9111 P. A. Sackinger
MS0826 9111 P. R. Schunk
MS0826 9111 J. A. Schutt

MS0834 9112 A. C. Ratzel
MS0834 9112 M. R. Baer
MS0834 9112 P. L. Hopkins
MS0834 9112 J. R. Torczynski
MS0835 9113 R. E. Hogan
MS0827 9114 R. T. McGrath
MS0827 9114 A. J. Russo
MS0443 9117 H. S. Morgan

Route to: 9118

MS0443 9117 S. W. Key
MS0443 9117 C. M. Stone
MS0443 9117 G. W. Wellman
MS1111 9221 J. N. Shadid
MS0819 9231 J. M. McGlaun
MS0819 9231 A. C. Robinson
MS1186 9542 R. S. Coats
MS0899 4414 Technical Library (5)
MS0100 7613-2 Document Processing
-- for DOE/OSTI (2)
MS0619 12615 Print Media
MS9018 8523-2 Central Technical Files

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.