SANDIA REPORT

SAND89–2989 • UC–705
Unlimited Release
Printed July 1991

MERLIN II - A Computer Program to Transfer Solution Data Between Finite Element Meshes

D. 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-AC04-76DP00789

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

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: A03
Microfiche copy: A01

SAND89-2989
Unlimited Release
Printed July 1991

MERLIN II - A COMPUTER PROGRAM TO TRANSFER SOLUTION DATA BETWEEN FINITE ELEMENT MESHES

D. K. Gartling
Computational Fluid Dynamics Division
Sandia National Laboratories
Albuquerque, New Mexico 87185

ABSTRACT

The MERLIN II program is designed to transfer data between finite element meshes of arbitrary geometry. The program is structured to accurately interpolate previously computed solutions onto a given mesh and format the resulting data for immediate use in another analysis program. Data from either two-dimensional or three-dimensional meshes may be considered. The theoretical basis and computational algorithms used in the program are described and complete user instructions are presented. Several example problems are included to demonstrate program usage.

Contents

1 Introduction

The need to transfer data between different computational meshes is a frequently occurring problem in the area of computational mechanics. Typically, such data transfers become important when considering the solution of interdisciplinary problems in which the coupling between the relevant physical phenomena is relatively weak. In the present context, a problem is defined to be weakly coupled if the analysis of the participating physical processes may be carried out in a sequential manner (*e.g.*, thermal-stress analysis); strongly coupled problems demand the simultaneous solution of the relevant physical phenomena (*e.g.*, fluid-structure interaction). Though numerical solutions to weakly coupled problems have been successfully analyzed in the past, the interface between computational procedures was often of an *ad hoc* nature. As computer simulations have become more sophisticated, an increased demand has developed for the routine analysis of weakly coupled problems in general and of thermal-stress analysis problems in particular. In an effort to automate and streamline the needed data transfer process for such problems, a general interpolation program, called MERLIN II, has been developed. This code represents a significantly expanded and enhanced version of the program first documented in [1].

The MERLIN II program was designed primarily to alleviate the following problems that frequently occur in the computer analysis of weakly coupled problems. In particular, MERLIN II allows

- the use of significantly different computational meshes for each simulation as a result of differing resolution requirements for each physical phenomena.
- the use of different computational meshes for each simulation as a result of the incorporation of special analysis features in one of the computations (*e.g.*, the use of slide lines in a structural analysis which is unimportant in the thermal problem).
- the use of different computational meshes for each simulation due to the involvement of several analysts with different modeling requirements and/or personal preferences for mesh generation and analysis software.

In addition, MERLIN II can be employed as a rudimentary rezone program in which complete data fields from one simulation are simply transferred to a new mesh and the analysis continued. The new mesh may be needed because of unacceptable distortion in Lagrangian mesh descriptions or to increase/redistribute mesh resolution in Eulerian descriptions.

In the following section the general scope of the MERLIN II program is outlined. The third section briefly describes the algorithms that are used to interpolate data from one mesh to another. Finally, the last two sections provide complete instructions for use of the program, comments on the installation of the code and several examples to illustrate program capabilities.

2 Scope of Program

The primary motivation for the development of the MERLIN II program was the need to routinely transfer solution data between various computational mechanics codes. The original version of MERLIN performed this type of function mainly in the context of transferring two-dimensional, temperature data from a heat conduction program to a stress analysis program. Though the principal area of application is still assumed to be in the analysis of thermal-stress problems, a program with more general capability was desired.

To assist in the following discussion, it is appropriate to define a weakly coupled problem with generic properties. Let a physical process (*e.g.*, heat conduction) be simulated on a computational mesh, labeled A, that occupies a spatial domain Ω_A. A subsequent physical process (*e.g.*, elastic stress analysis) is simulated on mesh B that spans the domain Ω_B. The process to be simulated on mesh B requires data from the problem previously computed on mesh A. The MERLIN II program was designed to carry out the required data transfer from mesh A to mesh B.

A number of limitations had to be placed on the scope of the data transfer program in order to make the data handling and interpolation problems tractable. First, the analysis procedures for all of the weakly coupled processes were assumed to be based on the finite element method. Though it is recognized that a great deal of engineering analysis is performed with computational techniques other than the finite element method, there are a number of unresolved difficulties associated with interfacing general computational grids. For example, the topological structure of an integrated finite difference, lumped parameter mesh does not permit standard interpolation procedures to be employed. Such difficulties are not present in finite element based codes where a simple data structure allows interpolation to be carried out efficiently and accurately.

With regard to geometric restrictions, MERLIN II assumes that the physical region, Ω_B, modeled by mesh B, coincides with or is a subdomain of the region, Ω_A, modeled by mesh A. In essence this restriction implies that the solution data from the first simulation can be interpolated onto the second grid but extrapolation is not considered. Also, implicit in this assumption is the requirement that the origin of the coordinate systems used for each mesh coincide and the units chosen for length scales be identical.

The search and interpolation algorithms employed in MERLIN II are quite general with regard to their applicability to finite element meshes and the types of finite elements used in a simulation. The code was initially designed to accommodate most of the popular linear and quadratic, isoparametric elements in both two and three dimensions. Other elements can be added with simple modifications to a few subroutines. However, despite this algorithmic generality, MERLIN II is still limited in its applicability to a set of specific analysis packages. This lack of overall generality was dictated primarily by the wide variety of data formats encountered in the various finite element codes. In the present

version of MERLIN II, data transfer is limited to those codes which support the GENESIS [2] and EXODUS [3] file formats. These data formats were established at Sandia to simplify the input and output of data from finite element based mechanics programs. The GENESIS format provides a standard file for inputting mesh and boundary condition data to an analysis code; EXODUS provides a standard format for transforming solution data to a post-processing graphics program or to another analysis code. Though only the GENESIS and EXODUS formats are actively supported in MERLIN II, the addition of other formats would require relatively straightforward modifications to the program.

Finally, the concepts used in MERLIN II could form the basis for an elementary rezone capability within a larger program. Mechanics codes that use a Lagrangian mesh description often require a rezoned mesh to be generated and the dependent variables transferred to the new mesh. Eulerian-based codes could also require rezoning if the spatial (mesh) resolution requirements change significantly during the course of a solution or if Lagrangian features, such as free surfaces, are modeled. Given the old and new (rezoned) mesh descriptions, MERLIN II could function as the dependent variable interpolator for the mechanics program. In such a case, MERLIN II would be tailored to the specific program and probably made into a series of subroutines within the larger program.

3 Search and Interpolation Procedures

The present section briefly outlines the procedures used by MERLIN II to interpolate data from one mesh to another. For purposes of this discussion it is convenient to refer to the previous definitions for a weakly coupled problem. The mesh on which a solution has been previously computed is denoted as mesh A; the mesh to which the solution is to be interpolated is mesh B.

The computational procedure used in MERLIN II naturally divides itself into three phases: *i*) the reading and storing of A and B mesh data, *ii*) mesh searching and interpolation of the solution, and *iii*) the writing of the necessary output files. As the first and third steps are simply problems of data formatting and bookkeeping, they need not be considered here in any detail. Rather the discussion will be focused on the search and interpolation procedures. Also, the development will be in terms of the general three-dimensional case; the two-dimensional algorithms follow by direct analogy.

For each nodal point P_B with coordinates (x_B, y_B, z_B) in mesh B the following sequence of operations is performed:

1. MESH SEARCH: A search of mesh A is made to locate the element or group of elements which may contain node point $P_B(x_B, y_B, z_B)$.
2. ELEMENT SEARCH: A Newton iteration procedure is used to find the local isoparametric element coordinates (s_A, t_A, r_A) that describe the position of point P_B within the element from mesh A.
3. INTERPOLATION: For a given set of local element coordinates (s_A, t_A, r_A), the finite element basis functions may be used to compute the value of the dependent variable at point P_B from the mesh A solution.

Each of these operations is described in more detail in the following sections.

3.1 Mesh Search

The determination of which element in mesh A contains a given point, P_B, can become a very time consuming process if standard, sequential search methods are employed. However, a significant improvement in the search procedure can be realized by presorting the elements in mesh A into subgroups or bins according to their spatial location. During a search, mesh point P_B can then be quickly tested for general spatial location and located within a particular bin. The number of elements that must be carefully screened for each P_B can thus be (significantly) reduced from the total number of elements in mesh A to the maximum number of elements in a bin. MERLIN II implements the presorting algorithm

by construction of a rectangular parallelepiped (box) that surrounds the mesh A region, Ω_A. Each edge of the box is subdivided to form a rectangular grid of $n_x \times n_y \times n_z$ cells or bins. A simple test on the coordinates of each element in mesh A locates it within one or more bins; a list of elements for each bin is constructed for subsequent use in the mesh search portion of the code. Note that the number of bins constructed in the presort procedure is specified by the user and an optimal grid will be mesh dependent. The examples provided in a later section demonstrate the trade-offs and efficiencies associated with the presort procedure.

Upon completion of the presorting procedure, the search for the location of each point P_B within element mesh A continues in a two stage process – a "coarse" search which eliminates all elements in a particular bin from consideration except those in the "neighborhood" of point P_B and a "refined" search that finally locates the specific element in A that contains point P_B. The coarse search procedure is based on the construction of a circumscribing sphere (circle in two dimensions) for each element in the selected bin. If point P_B is found to lie within a particular sphere (circle), that element is retained for further testing; when point P_B is outside the sphere (circle) that element is omitted from further consideration.

The refined portion of the search is used to finally decide if the point P_B is within an element. Several techniques could be used for this task, depending mainly on the type of finite element being tested. For low order, straight-sided elements (*e.g.*, four node, two-dimensional quadrilaterals or eight node, three-dimensional bricks), geometric procedures could be used, as was done in the earlier version of MERLIN. Generally, the testing for the presence of a point within a higher order, curved sided element is of sufficient complexity to preclude the use of geometrically based methods. MERLIN II was designed to accommodate a wide variety of both high and low order elements. For simplicity in code design, a single procedure that was applicable to all element types was desired. Such an approach, which is more analytic in nature and appeals to the basic nature of the isoparametric mapping procedure, is the use of Newton's method. As this refined mesh search procedure is conveniently combined with the element search algorithm, its detailed description will be given in the next section.

3.2 Element Search

Having determined that point P_B is located in the vicinity of an element, it is then required that the actual position of point P_B within the element be obtained. This "local" description of the coordinates for point P_B are most conveniently expressed in terms of the "natural" or normalized coordinates for the element [4]. Introducing the idea of isoparametric elements [5,6] permits the spatial description for the element to be written as

$$
\begin{aligned}
x &= \mathbf{\Phi}^T(s,t,r)\ \mathbf{x} \\
y &= \mathbf{\Phi}^T(s,t,r)\ \mathbf{y} \\
z &= \mathbf{\Phi}^T(s,t,r)\ \mathbf{z}
\end{aligned} \tag{1}
$$

where $\mathbf{\Phi}$ are vectors of element shape functions, $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are vectors containing the spatial coordinates for the nodal points of an element and superscript T denotes the transpose of a vector. The variables s, t and r are the normalized "local" coordinates for the element. For mesh point P_B located in an element of mesh A, equation (1) then becomes

$$
\begin{aligned}
x_B &= \mathbf{\Phi}^T(s_A,t_A,r_A)\ \mathbf{x}_A \\
y_B &= \mathbf{\Phi}^T(s_A,t_A,r_A)\ \mathbf{y}_A \\
z_B &= \mathbf{\Phi}^T(s_A,t_A,r_A)\ \mathbf{z}_A
\end{aligned} \tag{2}
$$

and the problem is to find the coordinate values for s_A, t_A and r_A that satisfy (2). As the element shape functions are usually nonlinear functions of s, t and r, the equations in (2) cannot generally be directly solved for s_A, t_A and r_A values. Within the MERLIN II program, the solution of (2) is carried out through the use of Newton's method. Details of the procedure are given in Appendix A.

The solution of equation (2) produces the s_A, t_A and r_A coordinates corresponding to point P_B and located in a particular element of mesh A. As noted previously, the "refined" mesh and element search procedures were combined into a single procedure. Upon obtaining the s_A, t_A and r_A coordinates for point P_B from the Newton procedure, a check is made to ensure that s_A, t_A and r_A lie completely within the mesh A element. When $|s_A|, |t_A|$, and $|r_A| \leq 1$, point P_B is within the element and the search procedure ends. If $|s_A|, |t_A|$, or $|r_A| > 1$, point P_B is outside of the element in question, and the search for the location of point P_B within the A mesh is continued.

It should be noted that in all inequality tests used in MERLIN II, there are tolerances incorporated to allow for finite precision arithmetic and the small differences that naturally occur in different mesh descriptions. Length tolerances are generally based on some small percentage of a typical element dimension.

3.3 Interpolation

The local coordinates that describe the location of point P_B within a particular element of mesh A are computed and stored as the result of the search procedure described above. The last remaining task in the overall algorithm is the interpolation of the solution for

the points in mesh B. The solution for mesh A is known in terms of the nodal values of the dependent variable for each element in the mesh. Thus, for each mesh A element the following relation is available

$$T(s,t,r) = \mathbf{\Psi}^T(s,t,r)\ \mathbf{T} \tag{3}$$

where $T(s,t,r)$ is the dependent variable at a point s,t,r in the element, $\mathbf{\Psi}$ is a vector of element shape functions that describes the variation of the variable within the element and $\mathbf{T}$ is a vector of nodal point values of the dependent variable. When the local coordinates for a point P_B within an element are available, as from the search procedure, the value of the dependent variable at point P_B is given by

$$T_{P_B}(s_A,t_A,r_A) = \mathbf{\Psi}^T(s_A,t_A,r_A)\ \mathbf{T}_A \tag{4}$$

Within MERLIN II, the above computation is carried out for each node in the B mesh. The resulting interpolated solution for B is properly formatted and written to a disk file for later use.

The above procedure was described in terms of a problem that required only one variable to be interpolated and that did not involve a time-dependent solution. MERLIN II is sufficiently general to accommodate both of these complications. When the mesh A solution contains more than one variable, MERLIN II will interpolate any specified variable or set of variables onto the B mesh. The relation in (4) is used repetitively for each set of nodal point variables, $\mathbf{T}$. Further, if the mesh A solution is time-dependent, MERLIN II will transfer data at all of the computed times or at times specified by the user. In this latter case, a linear interpolation in time is performed on the mesh A data.

4 User Guide

The MERLIN II program was specifically designed to be a user-oriented tool that facilitates the transfer of data between finite element meshes. In keeping with this philosophy, a conscious effort was made to minimize the data input to the program and to simplify access to the code. The following sections provide descriptions of the data cards necessary to use the MERLIN II program, comments on how to access the program and a description of the files needed by the program. A section on coding and computer implementation is also included to assist in installation and possible modifications of the code.

4.1 Program Limits

The basic version of the MERLIN II program is structured to accommodate the transfer of data between relatively large two- and three-dimensional finite element meshes. The code is dynamically dimensioned and has no specific limits on the sizes of the A and B meshes that may be considered. There are, however, some other limits that must be observed.

One limitation of the program concerns the types of finite elements that may occur in the A and B meshes. Presently, MERLIN II recognizes the following element types:

- TRI3 - three node, linear triangular element (2D)
- TRI6 - six node, quadratic triangular element (2D)
- QUAD4 - four node, linear quadrilateral element (2D)
- QUAD8 - eight node, quadratic quadrilateral element (2D)
- QUAD9 - nine node, quadratic quadrilateral element (2D)
- TETRA4 - four node, linear tetrahedral element (3D)
- TETRA10 - ten node, quadratic tetrahedral element (3D)
- PRISM6 - six node, linear prism element (3D)
- PRISM15 - fifteen node, quadratic prism element (3D)
- BRICK8 - eight node, linear brick element (3D)
- BRICK20 - twenty node, quadratic brick element (3D)
- BRICK27 - twenty seven node, quadratic brick element (3D)

This list could be expanded to include other, more specialized, elements through minor code modification. Also, note that in order to describe each of these elements, a convention for locally numbering the nodes within an element is required. The convention followed in MERLIN II is based on the EXODUS [3] standard; nodal point numbering runs counterclockwise around element edges with corner nodes listed first, followed by mid-edge nodes, if they are present.

The program is also limited to consideration of twenty variables, or less, in the mesh A solution and subsequent interpolation procedure. More variables could be treated by increasing several dimension statements in the code. Note also that the solution data supplied from the A mesh must be defined at the element nodal points; integration point data or element based data is not considered. Finally, as noted previously, the input and output formats recognized by MERLIN II are limited to the GENESIS [2] and EXODUS [3] standards.

4.2 Input Syntax

The structure of an input data set for the MERLIN II program is straightforward and consists of five command cards and associated data. Each command sequence is delimited by a specific form of termination command. To further simplify the data input, MERLIN II is equipped with a free field input routine to eliminate the problem of remembering several fixed field formats. In describing the data associated with each input card the following conventions and restrictions will be observed:

(a) Upper case bold face words indicate required command and termination words, *e.g.*, **MESH-A**.

(b) Lower case words and symbols imply that an alphanumeric or numerical value for the specified variable is expected, *e.g.*, iprint.

(c) All input data are specified in a free field format with successive variables separated by commas. All alphanumeric input data is limited to ten characters under this format; all alphanumeric input must be in upper case format.

(d) The $ character may be used to end an input line. The remaining space on the line can be used for comments.

(e) The * character may be used to continue an input line onto a second data card. The continuation character should follow the last comma on the card to be continued.

(f) *Italics* indicate optional parameters which may be omitted by using successive commas in the variable list. If the omitted parameter is not followed by any required parameters, no additional commas need be specified.

(g) ( ) indicate the data type for a particular input variable, *i.e.*, alphanumeric or character (C), real (R) or integer (I).

(h) < > indicate the default value for an optional parameter.

(i) The contents of each input card are indicated by underlining.

In the following sections, the input to MERLIN II is described in roughly the order that data would normally be supplied to the code. For convenience, a summary of the input commands is collected in Appendix B.

4.3 Title and Comment Cards

The title or header card must be the first card in an input data set for any particular problem. A $ symbol must appear in Column 1; the remaining 79 columns are available for a problem title. The problem title is used to label printed output. The title card is of the following form:

```
$ THIS IS AN EXAMPLE OF A PROBLEM TITLE
```

Following the title card, a number of comment cards may be included to provide a description of the particular problem, codes used to produce the solution, *etc.* Up to 10 comment lines may be specified; each line begins with a $ leaving the remaining 79 columns available for descriptive text. The comment lines are reproduced at the beginning of the printed output listing. Comment lines have the following form:

```
$ THESE ARE EXAMPLES OF COMMENT LINES
$ UP TO 10 LINES OF PROBLEM DESCRIPTION MAY
$ BE INCLUDED ON THESE CARDS
.
.
.
```

4.4 MESH-A Command Card

The data describing properties of the A mesh, on which a solution has been previously computed, is input through the **MESH-A** command. This command and its associated data cards have the following form:

MESH-A, *format*
element type, element block id
.
.
.
END

where

format (C) < EXODUS > : specifies the input file format for the data from mesh A. In the current version of MERLIN II this parameter should only be set to EXODUS.

element type (C) : indicates the type(s) of element(s) used to generate the solution on the A mesh. The permissible element types include: TRI3, TRI6, QUAD4, QUAD8, QUAD9, TETRA4, TETRA10, PRISM6, PRISM15, BRICK8, BRICK20, BRICK27. These elements are defined in Section 4.1.

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

Notes:

1) The element block identifiers are assigned during the mesh generation operation and must be recalled for use in the element specification. One and only one data card for each element block must appear in the input deck. See reference [2] for further information on the use of block identifiers.

4.5 MESH-B Command Card

The properties of mesh B, onto which the solution is to be interpolated, are input through the **MESH-B** command. This command and its associated data cards have the following form:

MESH-B, *format*
element type, element block id
.
.
.
END

where

format (C) < EXODUS > : specifies the input/output file format for the data from mesh B. In the current version of MERLIN II this parameter should only be set to EXODUS.

element type (C) : indicates the type(s) of element(s) used in the B mesh. The permissible element types include: TRI3, TRI6, QUAD4, QUAD8, QUAD9, TETRA4, TETRA10, PRISM6, PRISM15, BRICK8, BRICK20, BRICK27. These elements are defined in Section 4.1.

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

Notes:

1) The element block identifiers are assigned during the mesh generation operation and must be recalled for use in the element specification. One and only one data card for each element block must appear in the input deck. See reference [2] for further information on the use of block identifiers.

4.6 VARIABLES Command Card

The specification of the data fields that are to be interpolated from the A mesh to the B mesh are input through the **VARIABLES** command. The form of this command and its associated data cards is given by:

VARIABLES
variable name, *default exterior value*
.
.
.
END

where

variable name (C) : specifies the name of the variable that is to be interpolated (see Note 1).

default exterior value (R) < 0.0 > : specifies the value to be given to mesh B nodes that fall outside of the region spanned by mesh A (see Note 2).

Notes:

1) The variable names used here must correspond to the names used to define the variables in the EXODUS file for mesh A. These names will depend on the analysis code used to produce the mesh A solution.

2) For each variable to be interpolated a default value should be defined. If a node in mesh B falls outside of the A mesh, the default exterior value will be assigned to the B node.

4.7 TIMEPLANE Command Card

The **TIMEPLANE** command is used to specify the times at which the mesh A solution is to be interpolated to mesh B. This command is required even when the solution on mesh A is independent of time. The form of the command and its data card is as follows:

TIMEPLANE
option, *parameter1, parameter2, ...*
END

where

option (C) : indicates which of the timewise interpolation schemes is to be employed. There are three methods of selecting the time planes when data are transferred between meshes. When the "option" parameter is set equal to ALL, all of the time planes found in the mesh A solution file are interpolated onto mesh B. There are no additional parameters with this option. When "option" is set to INCREMENT, data from mesh A is transferred to mesh B at constant intervals as specified by the parameters 1 through 3. For this option *parameter1* = $t_{initial}$, *parameter2* = t_{final} and *parameter3* = Δt. Interpolation occurs between meshes starting at $t_{initial}$ and running through t_{final} in time steps equal to Δt. When "option" is set to SPECIFIED, data from mesh A is transferred to mesh B at those times specified in the parameter list. In this case, *parameter1* should be set to the number of times to be specified. The remaining parameters in the list are set to the actual times at which data is to be interpolated onto the B mesh. The current version of MERLIN II requires that *parameter1* be less than or equal to 125 under this option.

4.8 EXECUTE Command Card

The actual search and interpolation process in MERLIN II is initiated by the **EXECUTE** command. This command causes the code to perform the necessary data transfer processes and format the output files. Program termination occurs at the end of this command when the **STOP** command is encountered. This command has the following form:

EXECUTE, *nx,ny,nz,nelbin,printout*
STOP

where

nx,ny,nz (I) < 1 > : specifies the number of grid cells (bins) to be used in partitioning the mesh A region (see Note 1).

nelbin (I) < 2× NUMELA/NUMBIN > : specifies the maximum number of elements allowed in any given bin (see Note 1).

printout (C) < SUMMARY > : indicates the amount of printed output that is generated during program execution. For *printout* set to SUMMARY or left blank, a minimal amount of output is generated. When *printout* is set to DETAILED or DEBUG, an increasing amount of information is output regarding the location of the mesh B nodes within mesh A.

Notes:

1) The selection of the grid cell parameters must be coordinated with the definition of *nelbin*. Prior to sorting of the elements in mesh A, it is usually not possible to predict the maximum number of elements that will be located in a particular bin. The parameter *nelbin* serves as a sizing parameter to allocate space for the element list for each bin. The default value for *nelbin* is set to the total number of elements in mesh A when *nx*=*ny*=*nz*=1 (*i.e.*, when only one grid cell is defined). When more than one grid cell is defined, the default for *nelbin* is the total number of elements in mesh A (NUMELA) divided by the total number of grid cells (NUMBIN) times two. By setting the *nelbin* parameter to a relatively small value, it can also serve as an indicator when too many elements are found in a bin and the search process will be impeded. In this latter case, the number of grid cells should be increased.

4.9 File Usage

The nature of the tasks performed by MERLIN II necessarily requires that a number of files be attached to the program for the input and output of data. All of the files used by the program are listed in the following table along with a brief description of their function, their internal FORTRAN identification and format type.

When files are to be attached to a MERLIN II job or saved after processing, the unit numbers used in the program execution statements must conform to those indicated in the following table. File names are arbitrary but must conform to the syntax rules for the particular operating system. A utility program from the SUPES library [7] obtains needed file names from the operating system for use in opening files during program execution. All files are opened in the `OPNFIL` subroutine. Prior to program execution, the proper relation between file name and unit number must be established via a system command; appropriate commands for a variety of operating systems are described in [7].

Unit Number	FORTRAN Name	File Usage	File Type
5	`NIN`	Input file	Formatted
6	`NOUT`	Output file	Formatted
10	`NTP0`	Scratch	Unformatted
11	`NTP1`	Free field input	Unformatted
12	`NTP2`	Mesh & solution data, mesh A (EXODUS)	Unformatted
13	`NTP3`	Mesh data, mesh B (GENESIS)	Unformatted
14	`NTP4`	Mesh & solution data, mesh B (EXODUS)	Unformatted
15	`NTP5`	Interpolated solution, mesh B	Unformatted

To understand the relationship between MERLIN II and its required files, reference should be made to Figure 1. This schematic indicates the interaction between the analysis codes and the files needed to successfully complete a data transfer. Most of the files shown in the table and Figure 1 that are of concern to the user have standard formats and their utility is self-explanatory. Note that there are three output files. Unit 6 contains printed output from the execution of MERLIN II. Unit 14 contains an EXODUS file with mesh and interpolated solution data for mesh B that can be used with a graphics program to verify the accuracy of the interpolation process. Finally, unit 15 contains the interpolated solution field for mesh B that is formatted for direct use in a mechanics program.

4.10 File Formats

The disk files used in MERLIN II are generally sequential access, unformatted files. The specific formats for the GENESIS and EXODUS files are well documented in [2,3] and need not be given here. The only other file format that is of concern to the user involves the interpolated solution data that goes to the mechanics code and is written to unit 15.

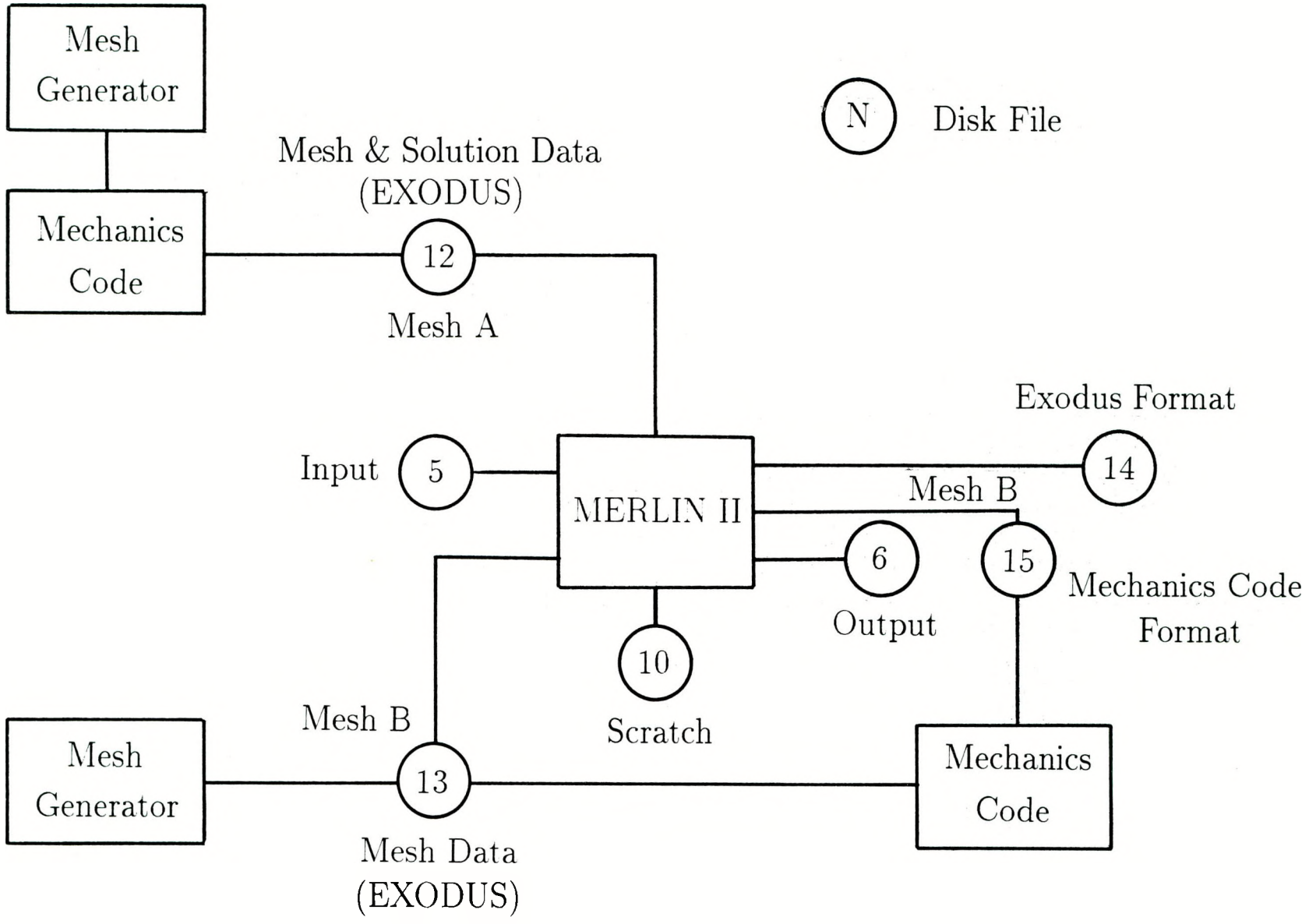

Figure 1: Schematic of data flow to and from MERLIN II.

Data for unit 15 is written with the following series of unformatted FORTRAN write statements

```
      IF (NUMVAR.GT.1) THEN
      WRITE (NTP5) NUMNOD,NUMVAR
      END IF
        .
        .
        .

      DO 10 NT=1,NTIMES
      WRITE (NTP5) TIME(NT), ((SOLN(I,J,NT), J=1,NUMNOD), I=1,NUMVAR)
   10 CONTINUE
```

where

`NUMNOD` : is the number of nodes in the mesh

`NUMVAR` : is the number of dependent variables listed under the **VARIABLES** command

`TIME` : is the current time

`SOLN(I,J)` : is the two-dimensional array containing the dependent variables. The dependent variables are listed 1 through `NUMVAR` in the order in which they appear in the mesh A EXODUS file.

4.11 Access to the Code

The source version of MERLIN II is maintained on the central file system (CFS) of the Central Computer Facility at Sandia National Laboratories. The program may be accessed by the following MASS utility [8] command

```
? get filnam : /FECODES/MERLIN/MERLIN2
```

The `filnam` parameter is the local file name for the code on the host computer. This is the standard UNIX[1] version of the program (source code) that is designed to run on the Sandia National Laboratories mainframe and distributed computers.

[1] UNIX is a trademark of AT&T Bell Laboratories

A script for executing MERLIN II is also available from the Sandia Engineering Analysis Code ACCESS System (SEACAS) [9]. This system is supported on the Central Computing Facility Unicos CRAY computers and the Directorate 1500 Local Area Networks. Access to MERLIN II through SEACAS consists of a single-line, UNIX-style command with a variety of options for file specification and data conversion.

The utility library needed for the execution of MERLIN II is 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 [7].

4.12 FORTRAN Coding and System Dependencies

The MERLIN II program is written in ANSI standard FORTRAN 77 and should therefore be usable on any computer system that supports such a compiler. The program was developed primarily for use on large mainframe computers, such as the CRAY-1S, CRAY X-MP and CRAY Y-MP, though it is also functional on smaller computers such as VAX, STARDENT and SUN. The code makes use of a few system dependent utilities. To increase portability of the code, most of the system dependencies have been isolated in a few subroutines that are located in a utility library that accompanies the main code. The utility library [7] includes such functions as calls to the date and time functions, routines needed for the dynamic memory manager algorithm and the free field reader used for decoding input to the code. The library routines are heavily commented to ease the task of converting these utilities to other computers. MERLIN II can be configured (via `SUBROUTINE OPNSSD`) to make use of a Solid State Disk (SSD) on CRAY computers, if such a device is available.

A dynamic memory manager is used in MERLIN 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. Further details on the memory manager and its operation are given in [7]. Some computer systems may not permit execution time changes in dynamic memory allocation; modifications to the code and memory manager to handle this case are outlined in [7].

5 Example Problems

Three example problems are included in this section to demonstrate the use of the MERLIN II program and provide some verification of its capabilities. The first two problems are contrived two- and three-dimensional examples that were developed to test the ability of the program to accommodate dissimilar geometries and element types in the A and B meshes. The third example illustrates the use of the program for an engineering analysis.

5.1 Cylinder to Rectangle Data Transfer

The first test problem consists of a steady state, heat conduction solution that was generated on the cylindrical section shown in Figure 2. The linear conduction problem was solved using the COYOTE II [10] finite element program; the mesh in Figure 2 was generated by the internal mesh generator in COYOTE II and consists of nine node quadrilateral and six node triangular elements. A contour plot of the isotherms produced for this problem are shown in Figure 3.

For purposes of this example, the rectangular region indicated in Figure 3 was discretized using four node linear elements via the FASTQ program [11]. MERLIN II was used to interpolate the COYOTE II solution from the cylindrical domain in Figure 3 onto the rectangular mesh. The results of the interpolation procedure are shown in Figure 4. The input data required by MERLIN II to perform this task is listed below. Note that mesh A (the cylinder) was only partitioned into a single grid cell due to the small size of the problem.

```
$EXAMPLE PROBLEM 1 FOR MERLIN II
MESH-A,EXODUS
QUAD9,1
TRI6,2
END
MESH-B,EXODUS
QUAD4,1
END
VARIABLES
TEMP,100.
END
TIMEPLANE
ALL
END
EXECUTE,,,,,SUMMARY
STOP
```

MERLIN II input for Example Problem 1

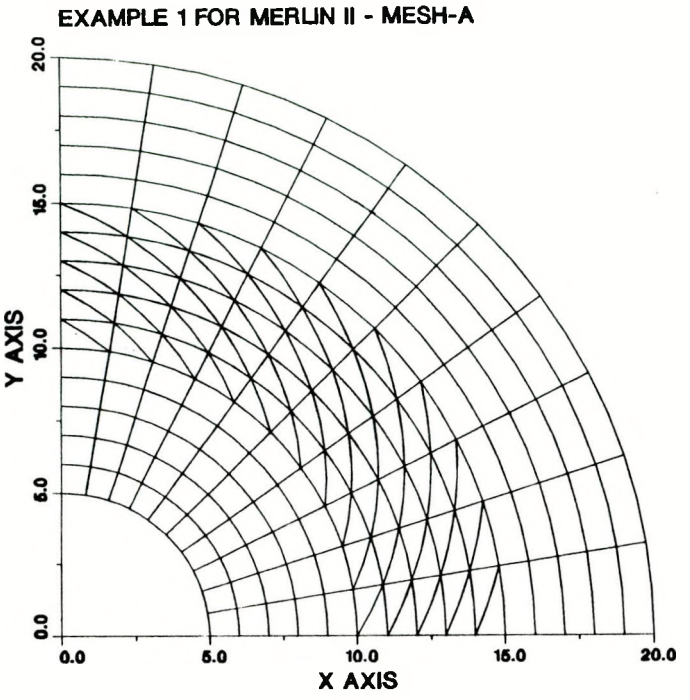

Figure 2: Finite element mesh for example problem 1 (Mesh A)

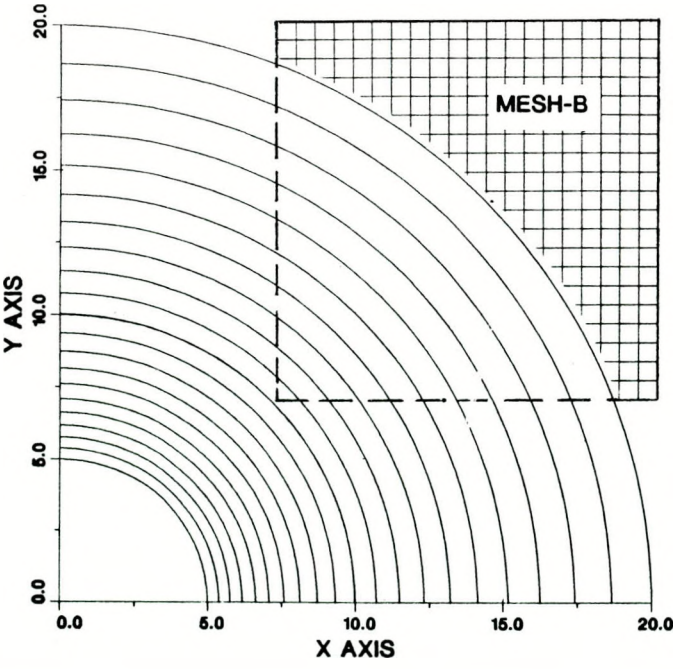

Figure 3: Temperature contours for mesh A. Location of Mesh B is indicated.

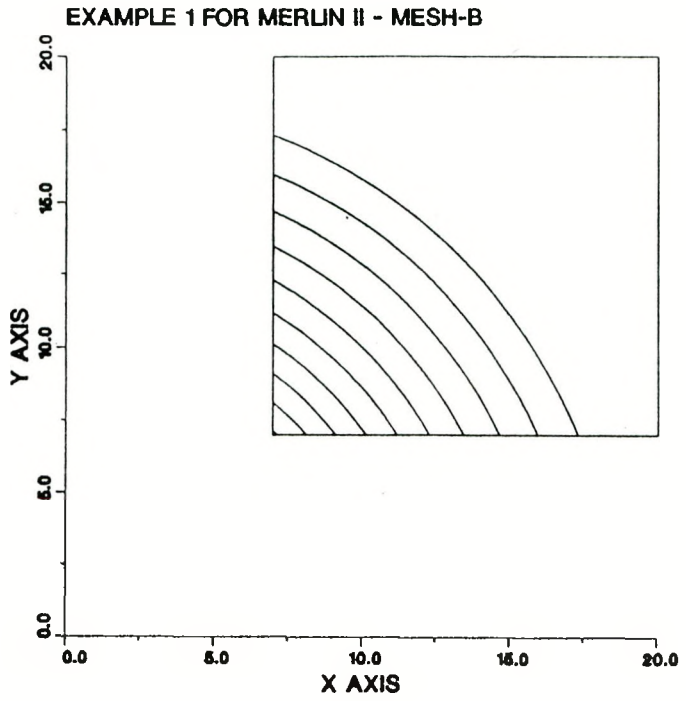

Figure 4: Temperature contours for solution interpolated to mesh B.

5.2 Sphere to Y-Block Data Transfer

The second example represents a test of a three-dimensional geometry and consists of the transfer of a diffusion solution on a sphere to an embedded rectangular block with a Y shaped cross-section. Shown in Figure 5 is the mesh developed for one-half of the solid sphere; the discretization was produced using the FASTQ program [11] and the GEN3D utility [12] and contains approximately 9200 eight-node hexahedral elements. The solid sphere was subjected to a uniform volumetric heat source and a constant surface temperature. The steady-state heat conduction problem was solved using COYOTE II [10]. Plots of the surface temperature on the half-sphere and a quarter section of the sphere are shown in Figure 6.

The mesh B geometry for this example consisted of a rectangular block with the cross-sectional shape of a Y as shown in Figure 7. The mesh in Figure 7 was also produced using FASTQ and GEN3D and contains approximately 4200 eight-node hex elements. The center of the Y section was located at the center of the sphere and the front face of the Y was coplanar with the face of the half sphere. Shown in Figure 8 is the temperature field as transferred to the Y block by MERLIN II. The characteristic spherical pattern of the isotherms is clearly evident on the Y geometry.

To demonstrate the effects of partitioning the mesh A region into grid cells or bins, a series of data transfers were computed. Listed in the following table are the results from eight runs where the number of grid cells were uniformly increased in all three coordinate directions. Three additional runs were performed in which the number of cells in the z direction (perpendicular to the face of the half sphere) was half the number of cells in

the x-y plane. It is clear from the execution times that a significant improvement in code performance can be achieved by presorting the elements in the A mesh. Also, there is an optimal grid configuration, where the work done in sorting the elements in mesh A is balanced by the search procedure within each bin. Though the optimal subdivision of mesh A will not usually be known *a priori*, some experimentation and experience can produce a near optimal or acceptable configuration.

# Grid Cells			Execution Time
nx	ny	nz	seconds
1	1	1	280
2	2	2	74
3	3	3	46
4	4	4	26
5	5	5	19
6	6	6	18
7	7	7	19
8	8	8	23
4	4	2	35
6	6	3	19
8	8	4	17

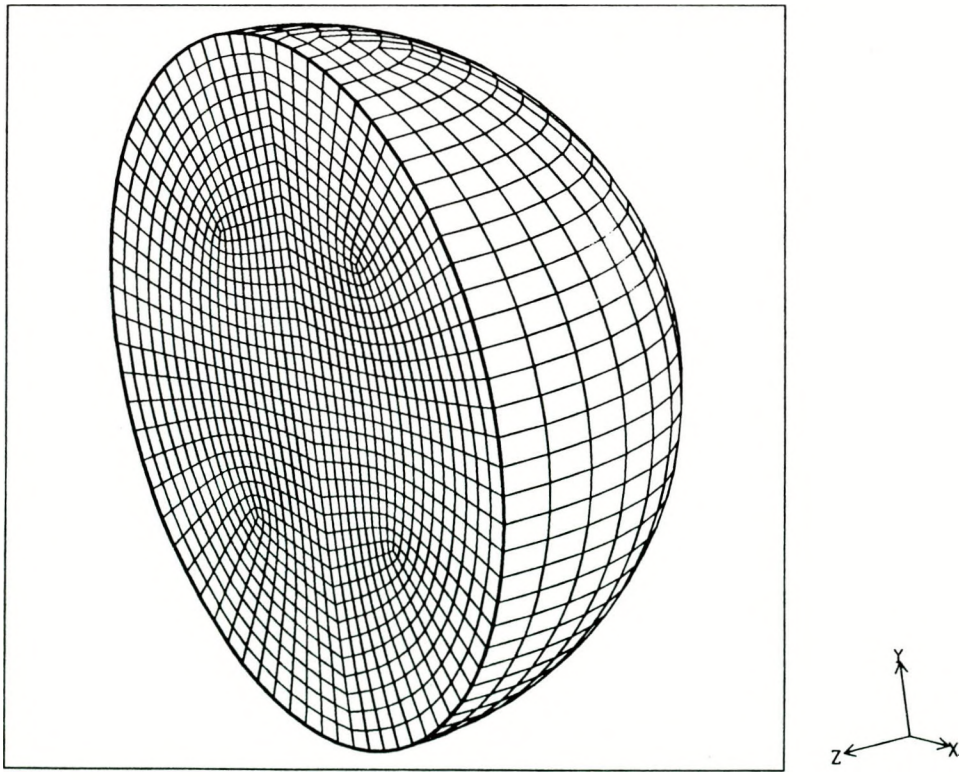

Figure 5: Finite element mesh for example problem 2 (Mesh A).

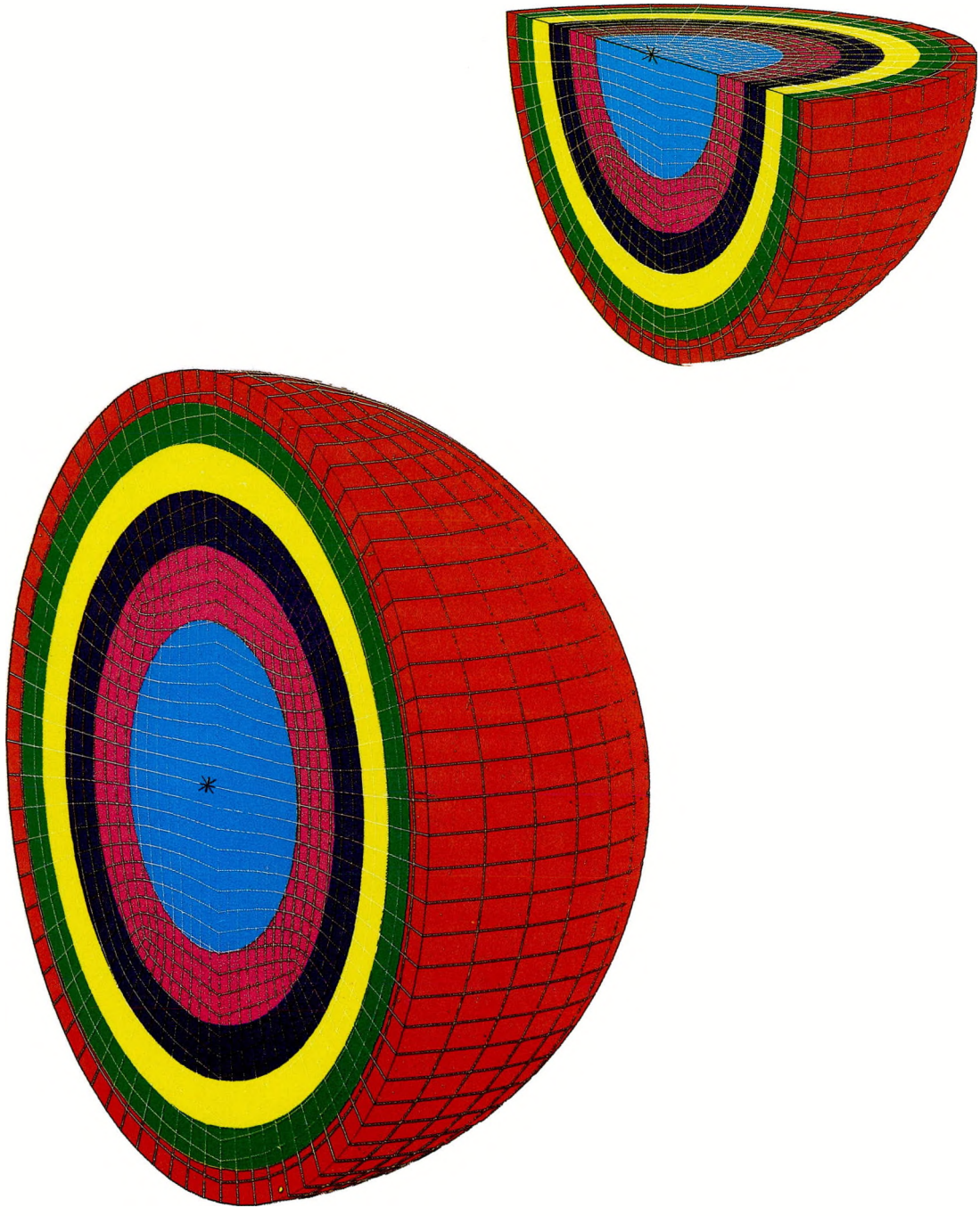

Figure 6: Temperature contours for Mesh A.

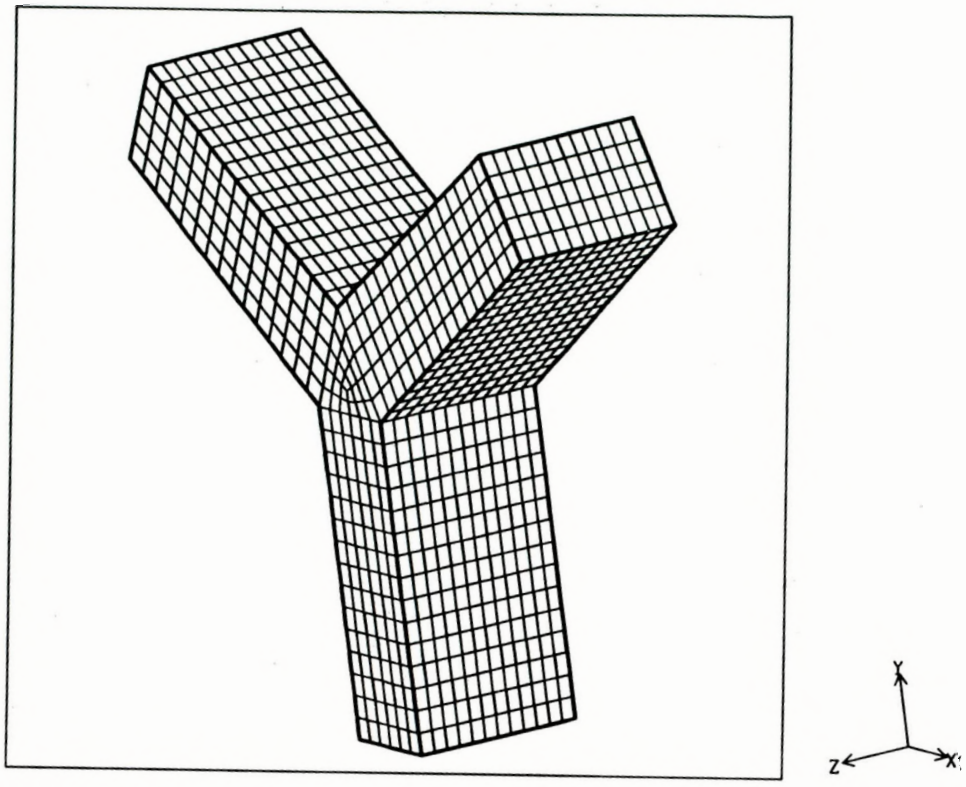

Figure 7: Finite element mesh for example problem 2 (Mesh B).

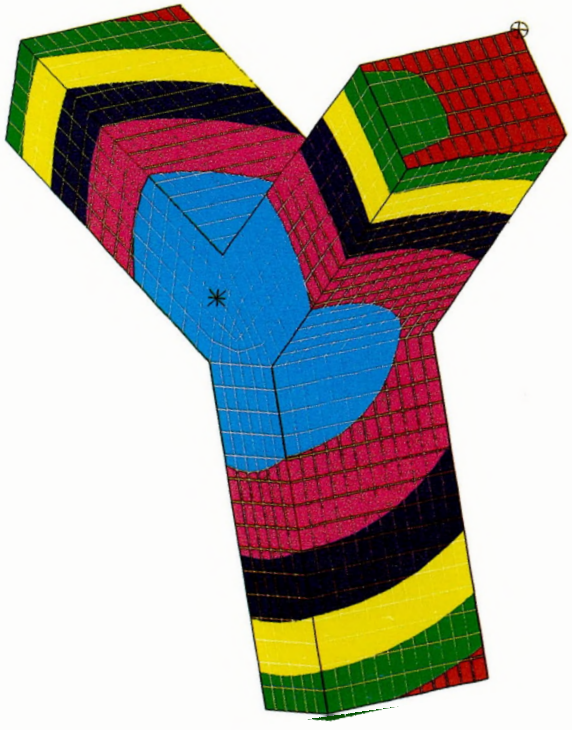

Figure 8: Temperature contours for solution interpolated to Mesh B.

5.3 Engineering Example

The final example concerns a data interpolation process that was required to perform a stress analysis of an electromagnetically loaded structure. Shown in Figure 9 is a schematic of the upper half of an axisymmetric diode for the Particle Beam Fusion Accelerator II (PBFA II). The simulation problem consists of determining the time-dependent magnetic field, currents and heating induced in the structure due to the application of a time varying current in the coils. The magnitude of the applied current is sufficiently large that magnetic pressure loads and Joule heating in the structure can affect the structural response of the diode components. MERLIN II is required to transfer the temperature and load data computed in the magnetic diffusion simulation over the entire geometry to a simpler structural analysis mesh that contains only the solid components of interest. Further details of the analysis are given in [13].

Figure 10 illustrates two views of the mesh used for the magnetic diffusion problem; the first figure shows the entire mesh while the second figure shows the mesh for the vacuum region with the solid components removed. This mesh was generated by FASTQ [11] and contains approximately 7500 four-node quadrilateral elements. The time-dependent magnetic diffusion and heat transfer problem was simulated using the TORO code [13]. Contour plots of the temperature and the magnitude of the magnetic pressure, as computed by TORO, are shown in Figures 11 and 12. Note that the contour plots show the fields at a time when the current applied to the main coil (Coil 1) is at a maximum.

The computational mesh used for the structural analysis is shown in Figure 13 and was also generated using the FASTQ program. This model contains only the solid components of interest and disregards most of the vacuum region. The magnetic and temperature fields were transfered to this mesh by MERLIN II with the temperature and magnetic pressure results shown in Figures 14 and 15. Note that all timeplanes computed in the magnetic diffusion simulation were transferred to the structural mesh.

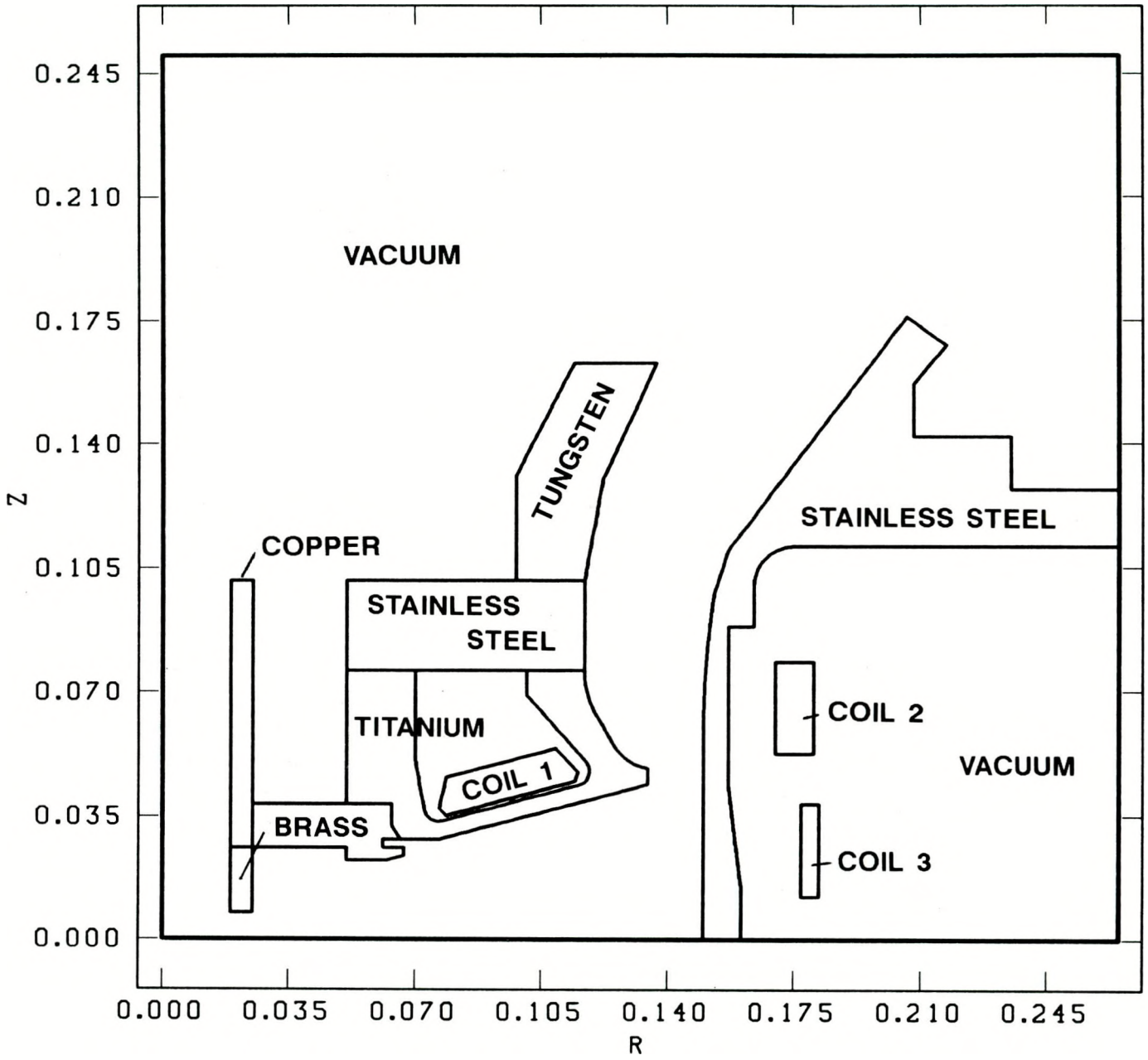

Figure 9: Schematic of PBFA II diode.

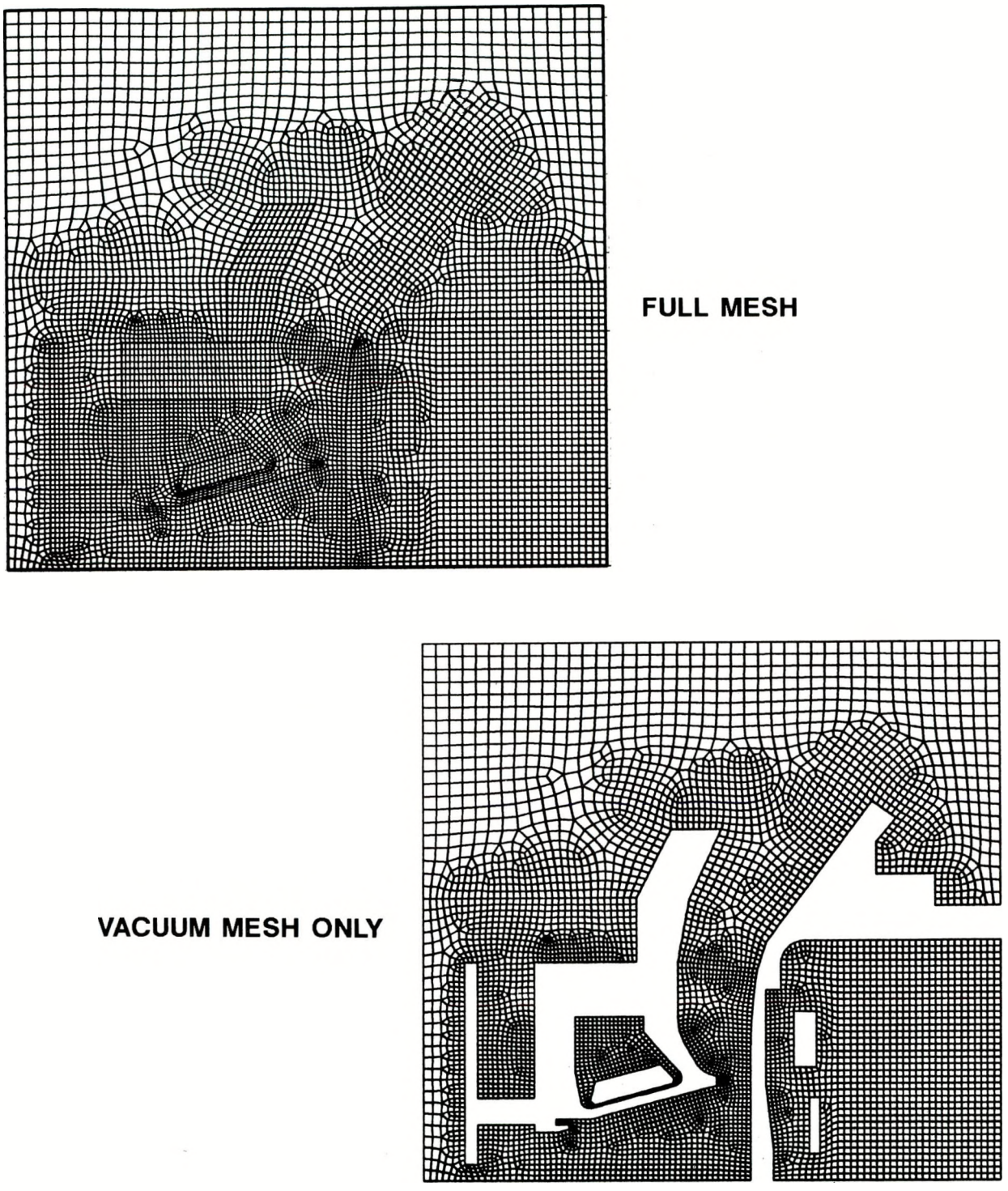

Figure 10: Finite element mesh for diode problem (mesh A).

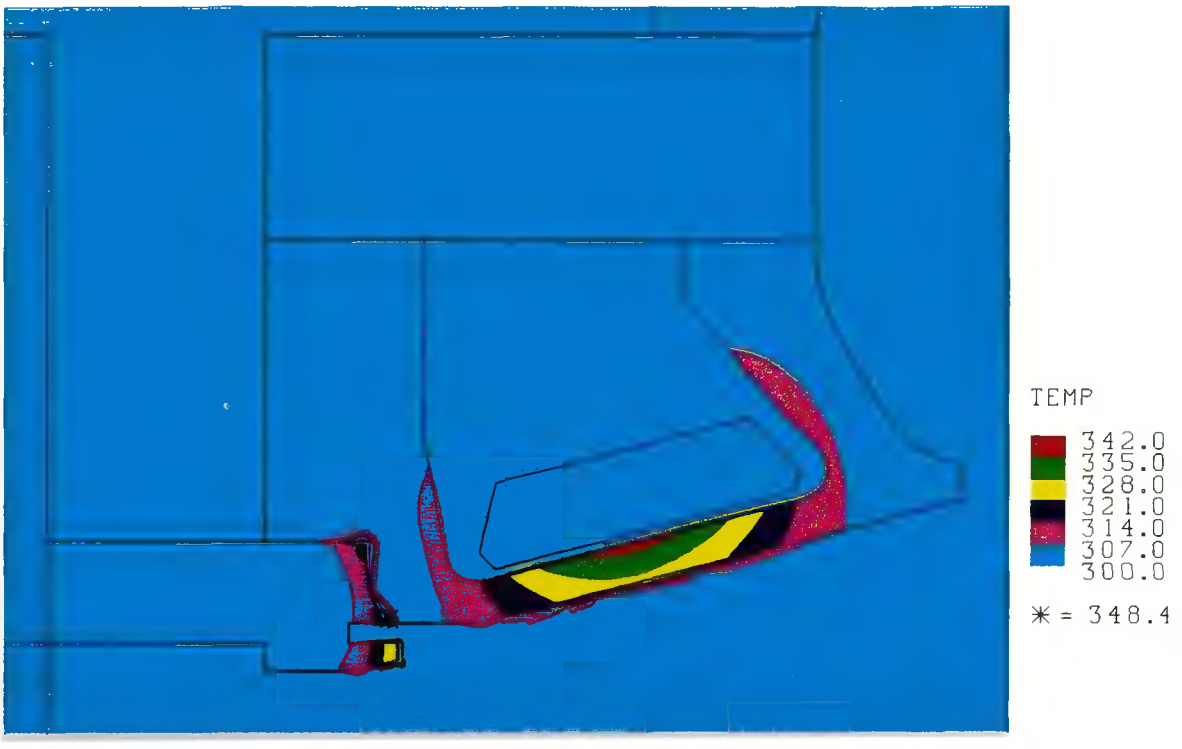

Figure 11: Temperature field in the diode at the peak of the current pulse.

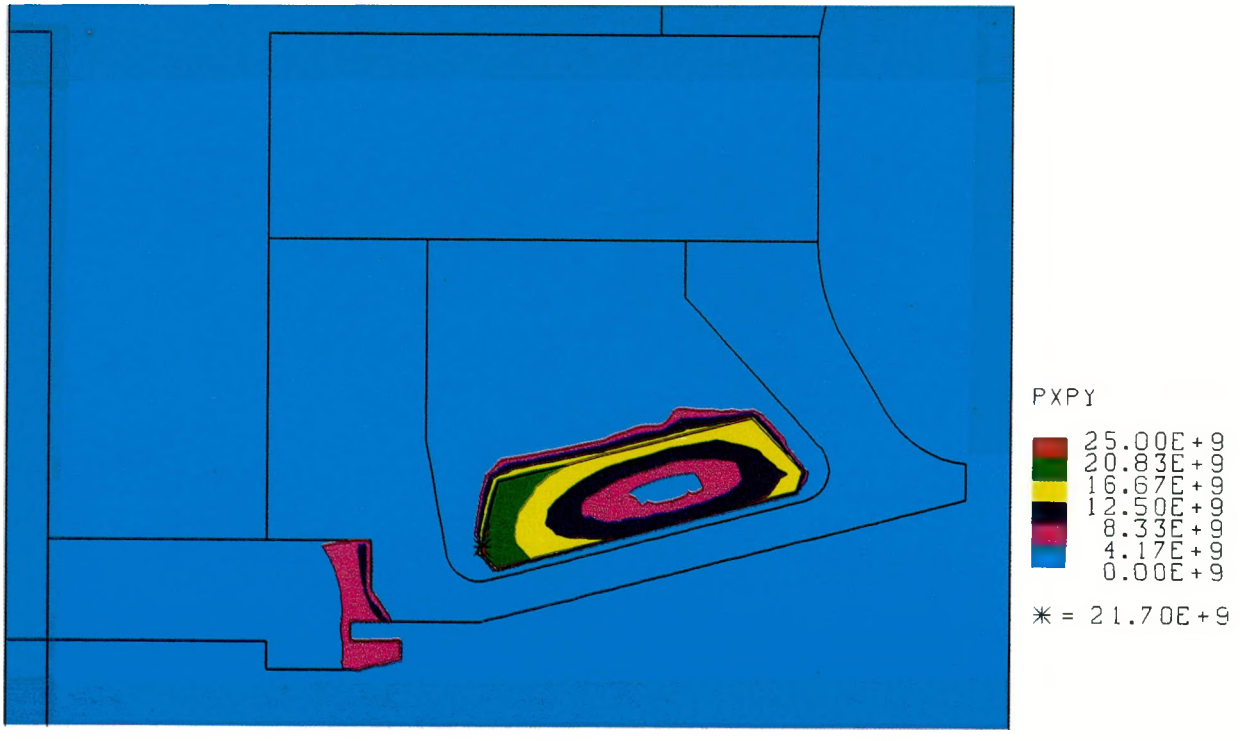

Figure 12: Magnitude of the magnetic pressure in the diode at the peak of the current pulse.

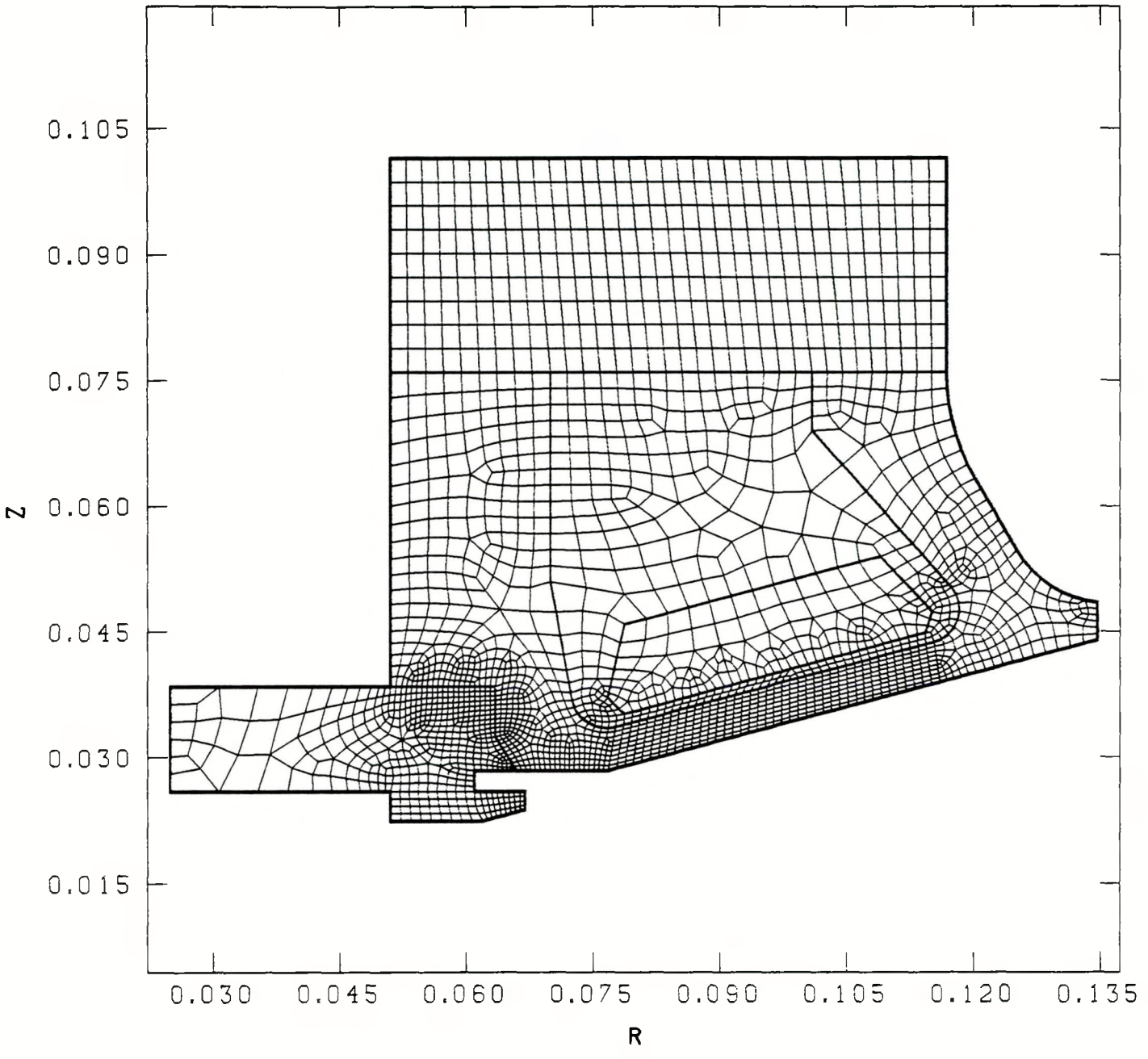

Figure 13: Finite element mesh for structural analysis of PBFA II diode (mesh B).

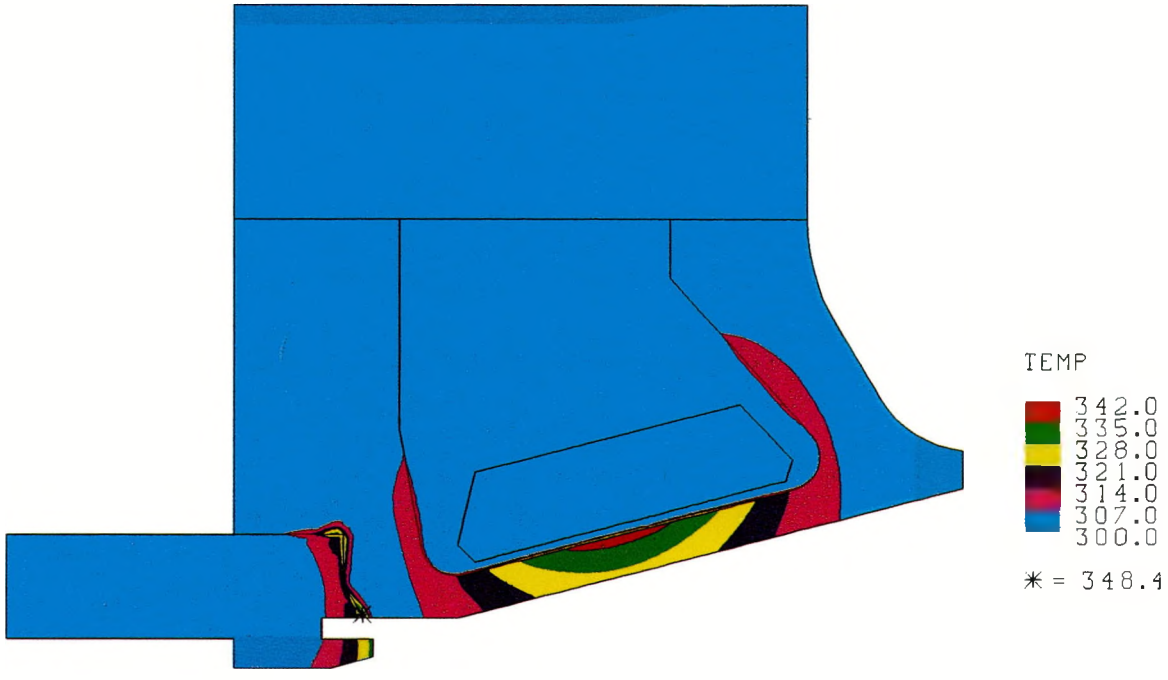

Figure 14: Temperature contours for solution interpolated to mesh B.

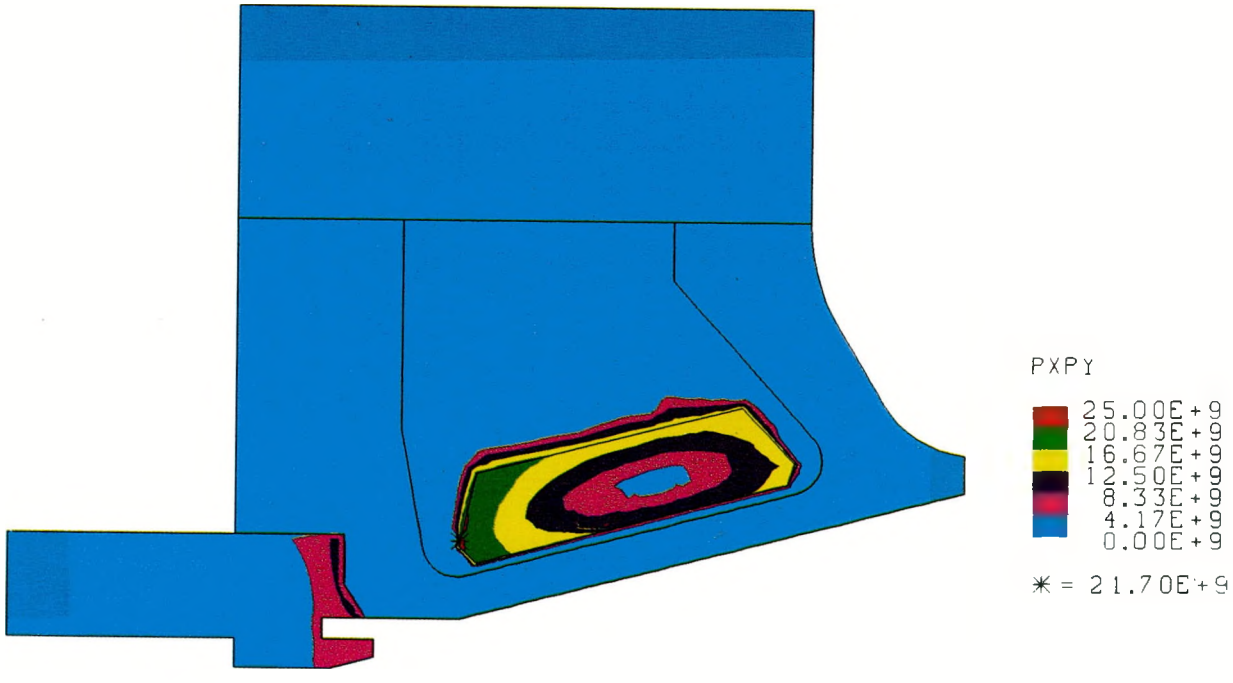

Figure 15: Magnetic pressure for solution interpolated to mesh B.

6 References

1. D. K. Gartling, "MERLIN - A Computer Program to Transfer Data Between Finite Element Meshes," SAND81-0463, Sandia National Laboratories, Albuquerque, N.M. (1981)

2. L. M. Taylor, D. P. Flanagan and W. C. Mills-Curran, "The Genesis Finite Element Mesh File Format," SAND86-0910, Sandia National Laboratories, Albuquerque, N.M. (1986)

3. W. C. Mills-Curran, A. P. Gilkey and D. P. Flanagan, "EXODUS : A Finite Element File Format for Pre- and Postprocessing," SAND87-2997, Sandia National Laboratories, Albuquerque, N.M. (1988)

4. O. C. Zienkiewicz, "*The Finite Element Method*," McGraw-Hill, London, 1977

5. I. Ergatoudis, B. M. Irons and O. C. Zienkiewicz, "Curved, Isoparametric, 'Quadrilateral', Elements for Finite Element Analysis," *Int. J. Solids Structures*," **4**, 31-42 (1968)

6. E. B. Becker, G. F. Carey and J. T. Oden, "*Finite Elements, An Introduction, Volume I*," Prentice Hall, New Jersey, 1981

7. 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, N.M. (1990)

8. "MASS Reference Manual," Version 1.0, Sandia National Laboratories, Albuquerque, N.M. (1983)

9. G. D. Sjaardema, "Sandia Engineering Analysis ACCESS System: Update (4/22/91)," memo to distribution, April 22,1991, Sandia National Laboratories, Albuquerque, New Mexico

10. D. K. Gartling,"COYOTE II - A Finite Element Computer Program for Nonlinear Heat Conduction Problems," SAND86-1844, Sandia National Laboratories, Albuquerque, N.M. (to be published)

11. T. D. Blacker, "FASTQ Users Manual, Version 1.2," SAND88-1326, Sandia National Laboratories, Albuquerque, N.M. (1988)

12. A. P. Gilkey and G. D. Sjaardema, "GEN3D : A Genesis Database 2D to 3D Transformation Program," SAND89-0485, Sandia National Laboratories, Albuquerque, N.M. (1989)

13. D. K. Gartling, "TORO - A Finite Element Program for Two-Dimensional Magnetic Diffusion Problems," SAND91-1491, Sandia National Laboratories, Albuquerque, N.M. (to be published)

Appendix A - The Element Search Algorithm

The spatial coordinates for a point P_B located within an isoparametric finite element "A" are described by the following equations

$$\begin{aligned} x_B &= \mathbf{\Phi}^T(s,t,r)\ \mathbf{x_A} \\ y_B &= \mathbf{\Phi}^T(s,t,r)\ \mathbf{y_A} \\ z_B &= \mathbf{\Phi}^T(s,t,r)\ \mathbf{z_A} \end{aligned} \tag{A.1}$$

where $\mathbf{\Phi}$ is a vector of element shape functions, $\mathbf{x_A}, \mathbf{y_A}, \mathbf{z_A}$ are vectors containing the spatial coordinates for the nodal points of element "A"and superscript T denotes a vector transpose. The variables s, t and r are the normalized, "local" coordinates for the element $(-1 \leq s,t,r \leq +1)$ [4].

During the interpolation procedure used in MERLIN II, it is required that the s,t,r coordinates corresponding to a given x_B, y_B, z_B and $\mathbf{x_A}, \mathbf{y_A}, \mathbf{z_A}$ be computed. In general, the shape functions, $\mathbf{\Phi}$, are sufficiently complex to preclude an analytic inversion of equation (A.1). MERLIN II employs an iterative procedure based on Newton's method to solve (A.1) for s,t and r.

Equation (A.1) may be conveniently written as a system of nonlinear equations of the form

$$\mathbf{f(s)} = \mathbf{0} \tag{A.2}$$

with

$$\mathbf{f} = \left\{ \begin{array}{c} f_1 \\ f_2 \\ f_3 \end{array} \right\} = \left\{ \begin{array}{c} x_B - \mathbf{\Phi}^T \mathbf{x_A} \\ y_B - \mathbf{\Phi}^T \mathbf{y_A} \\ z_B - \mathbf{\Phi}^T \mathbf{z_A} \end{array} \right\}$$

and

$$\mathbf{s} = \left\{ \begin{array}{c} s \\ t \\ r \end{array} \right\}$$

Writing a Taylor series expansion for (A.2) produces

$$\mathbf{f(s)} = \mathbf{f(s_0 + \Delta s)} = \mathbf{f(s_0)} + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{s}}\right|_{\mathbf{s_0}} \cdot (\mathbf{s} - \mathbf{s_0}) + \cdots$$

Truncating the expansion after the linear term leads to

$$\mathbf{f(s_0)} + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{s}}\right|_{\mathbf{s_0}} \cdot (\mathbf{s} - \mathbf{s_0}) = \mathbf{0}$$

or

$$\left.\frac{\partial \mathbf{f}}{\partial \mathbf{s}}\right|_{\mathbf{s_0}} \cdot (\mathbf{s} - \mathbf{s_0}) = -\mathbf{f}(\mathbf{s_0}) \qquad (A.3)$$

This last relation suggests the iterative solution procedure defined by

$$\mathbf{s}^{n+1} = \mathbf{s}^n - \mathbf{J}^{-1}(\mathbf{s}^n) \cdot \mathbf{f}(\mathbf{s}^n) \qquad (A.4)$$

where

$$\mathbf{J}(\mathbf{s}^n) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{s}}\right|_{\mathbf{s}^n} = \left[\begin{array}{ccc} \frac{\partial f_1}{\partial s} & \frac{\partial f_1}{\partial t} & \frac{\partial f_1}{\partial r} \\ \frac{\partial f_2}{\partial s} & \frac{\partial f_2}{\partial t} & \frac{\partial f_2}{\partial r} \\ \frac{\partial f_3}{\partial s} & \frac{\partial f_3}{\partial t} & \frac{\partial f_3}{\partial r} \end{array}\right]_{\mathbf{s}^n}$$

and superscript n indicates the iteration number. Equation (A.4) defines the iterative algorithm used in MERLIN II to compute the s, t, r variables.

The components of the Jacobian, $\mathbf{J}$, are easily computed once the basis functions, $\mathbf{\Phi}$, for each element have been specified. MERLIN II has a library of basis functions corresponding to the most commonly used two-dimensional and three-dimensional elements. Other elements could be added to MERLIN II by including the basis functions and derivatives to the Jacobian computation within the code.

Appendix B - Summary of Input Commands

In this section all of the command and data cards recognized by MERLIN 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 and Comments :

$ A TITLE LINE
$ A SERIES
$ OF
$ COMMENT LINES
.
.

MESH-A Command :

MESH-A, *format*
element type, block id
.
.
.
END

MESH-B Command :

MESH-B, *format*
element type, block id
.
.
.
END

VARIABLES Command :

VARIABLES
variable name, *default exterior value*
.
.
.
END

TIMEPLANE Command :

TIMEPLANE
option, *parameter1, parameter2, ...*
END

EXECUTE Command :

EXECUTE, *nx,ny,nz,nelbin,printout*
STOP

Distribution:

University of Texas (2)
Department of Aerospace Engineering
and Engineering Mechanics
Austin, Texas 78712
Attn: E. B. Becker
G. F. Carey

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

Lawrence Livermore National (2)
Laboratory
P. O. Box 808
Livermore, California 94550
Attn: P. M. Gresho
G. Goudreau

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

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

J. N. Reddy
Virginia Polytechnic Institute and
State University
Engineering Science and Mechanics
Blacksburg, Virginia 24061

Hibbitt, Karlsson & Sorrensen, Inc. (2)
100 Medway St.
Providence, Rhode Island 02906
Attn: L. M. Taylor
D. P. Flanagan

V. D. Murty
University of Portland
Department of Engineering
Portland, Oregon 97203

S. E. Benzeley
Brigham Young University
Department of Civil Engineering
Provo, Utah 84601

H. P. Wang
General Electric Company
Research and Development Center
Process Technology Branch
Schenectady, New York 12301

S. W. Key
McNeal-Schwendler Corp.
815 Colorado Blvd.
Los Angeles, California 90041

M. Crochet
Universite Catholique de Louvain
Unite de Mecanique Applique
B–1348
Louvain–la–Neuve
Belgium

Sandia Internal:

1425 J. H. Biffle
1425 S. W. Attaway
1425 T. D. Blacker
1425 M. L. Blanford
1425 J. Jung
1425 S. T. Montgomery
1425 J. A. Schutt
1425 W. R. Witkowski
1510 J. C. Cummings
1511 J. S. Rottler
1511 D. K. Gartling (25)
1511 R. R. Eaton
1511 R. C. Givler
1511 P. L. Hopkins
1511 M. J. Martinez
1511 P. A. Sackinger
1511 P. R. Schunk
1512 A. C. Ratzel
1512 M. R. Baer
1512 A. S. Geller
1512 M. W. Glass
1513 R. D. Skocypec
1513 R. L. Akau
1513 R. G. Baca
1513 B. L. Bainbridge
1513 C. E. Hickox
1513 R. E. Hogan
1513 S. N. Kempka
1513 J. L. Moya
1513 V. F. Nicolette
1513 V. J. Romero
1513 C. E. Sisson
1514 H. S. Morgan
1514 J. G. Arguello
1514 V. L. Bergmann
1514 S. N. Burchett
1514 R. S. Chambers
1514 E. P. Chen
1514 E. L. Hoffman
1514 J. F. Holland
1514 R. L. Johnson
1514 J. R. Koteras
1514 M. K. Nielsen
1514 C. M. Stone
1514 B. J. Thorne
1514 J. R. Weatherby
1514 G. W. Wellman
1540 J. R. Asay
1541 J. M. McGlaun
1544 J. R. Asay (acting)
1544 K. W. Metzinger
1544 G. D. Sjaardema
1544 P. P. Stirbis
1544 R. K. Thomas
1545 D. R. Martinez
1550 C. W. Peterson
8240 C. W. Robinson
8241 G. A. Benedetti
8242 M. R. Birnbaum
8243 M. L. Callabresi
8244 C. M. Hartwig
8245 R. J. Kee
3141 S. A. Landenberger (5)
3145 Document Processing (8)
for DOE/OSTI
3151 G. C. Claycomb (3)
8523 R. C. Christman

☆ U.S. GOVERNMENT PRINTING OFFICE 1991 - 573-122/40185

Org.	Bldg.	Name	Rec'd by	Org.	Bldg.	Name	Rec'd by

Sandia National Laboratories