PFC/RR-91-5

MAP USER'S MANUAL©

Robert D. Pillsbury, Jr

Plasma Fusion Center
Massachusetts Institute of Technology
Cambridge, MA

DECEMBER 1991

DISCLAIMER

This program and documentation is provided as is. No warranty is implied or stated by the author, the Plasma Fusion Center, or MIT with respect to the accuracy of the program, the documentation, or any results obtained from using the program.

TABLE OF CONTENTS

TABLE OF CONTENTS – (continued)

TABLE OF CONTENTS – (continued)

Chapter 1.0 INTRODUCTION

The program **MITMAP** represents a set of general purpose, two-dimensional, finite element programs for the calculation of magnetic fields. It consists of the programs **MAP** and **MAP2DJ**. The two programs are used to solve different electromagnetic problems, but they have a common set of subroutines for pre- and postprocessing. Originally separate programs, they have been combined to make modification easier. The manuals, however, will remain separate. The program **MAP** is described in this manual.

MAP is applicable to the class of problems with two-dimensional – planar or axisymmetric – geometries, in which the current density and the magnetic vector potential have only a single nonvanishing component. The single component is associated with the direction that is perpendicular to the plane of the problem and is invariant with respect to that direction – *e.g.*, $\tilde{\mathbf{J}} = (0, J_\theta(r,z), 0)$ or $\tilde{\mathbf{J}} = (0, 0, \mathbf{J_z(x,y)})$.

Maxwell's equations can be reduced to a scalar diffusion equation in terms of the single, nonvanishing component of the magnetic vector potential for planar problems and to a single component of a vector diffusion equation for axisymmetric problems. The magnetic permeability appears in the governing equation. The permeability may be a (nonlinear) function of the magnetic flux density. In addition, any electrically conducting material present will have eddy currents induced by a time varying magnetic field. These eddy currents must be included in the solution process.

MAP2DJ, on the other hand, is applicable to the class of problems with two-dimensional – planar or axisymmetric – geometries, in which the magnetic flux density or field intensity has only a single nonvanishing component. The single component is associated with the direction perpendicular to the plane of the problem and is invariant with respect to that direction – *e.g.*, $\tilde{\mathbf{B}} = (0, B_\theta(r,z), 0)$ or $\tilde{\mathbf{B}} = (0, 0, B_z(x,y))$. Associated with the one- (component) dimensional field is a two- (component) dimensional current density – hence, the name **2DJ**. The governing equation contains the material electrical resistivity which may be a strong function of temperature, which, in turn, is a function of the local ohmic heating. Therefore, **MAP2DJ** has the capability of solving the weakly coupled electromagnetic/thermal diffusion problem (or any subset of the more general problem).

MITMAP is written in FORTRAN (F77), and was developed on a VAX system. The program has been successfully transferred to the CRAY machines of the National Energy Research Supercomputer Center (NERSC) site at Livermore, CA. The program uses GRAFLIB as the graphics package. Wherever possible, machine and graphics package dependent coding has been isolated in subroutines to make conversion to other systems easier.

A large portion of this manual is repeated word for word in the **MAP2DJ** manual in order that each manual stand alone. In addition, there may be gaps in section and subsection

numbering in order to retain a common structure.

Page numbering in each of the two manuals will contain the program name, **MAP** or **MAP2DJ**, the chapter number, and then the page number. This numbering system will allow updates to be made with minimal effort.

This manual provides a description of the structure of the input data and output for the program There are several example problems presented that illustrate the major program features. Appendices are included that contain a derivation of the governing equations and the application of the finite element method to the solution of the equations.

Chapter 2.0 MAP

2.1 General Program Features

MITMAP contains many user oriented features. Some of these features are:

Free Field Input. The input of data requires no knowledge of a fixed field or format. Variables are separated by commas or spaces, and are typed (*e.g.*, real, integer) by their usage in the program. A variable cannot exceed 20 characters. Variables not specified are set to zero (or blank), unless otherwise noted. This free field feature greatly simplifies the data generation. The current version is coded to allow a maximum of five alphanumeric words per data line, twenty numeric entries, and a maximum of 80 characters per line is expected (including in-line comments). There is also a continuation capability for multiple-line input.

In the file input mode, all data are decoded before the processing starts. This feature allows typographical errors in the input file to be trapped prior to any (expensive) processing.

Comments. The free field reader in **MAP** allows both in-line and full line comments within the data set. The special character, ! (exclamation point), is interpreted to mean that the following (to the end of the line) text is to be treated as a comment. It is echo printed, but not decoded for use by the program.

Commands. **MAP** is structured so that a command language is used to drive the solution of a problem. The main program acts as a driver. The other subroutines are called by the driver according to a user specified command. The commands are as mnemonic as possible for ease of use. In addition, the program is modular and the user has the flexibility of determining the sequence in which the subroutines are executed. For instance, the user may generate and plot an element mesh without having to form and solve the finite element equations.

Interactive or File Input. **MAP** can be used in either the interactive or file mode for either input or output. The primary use of the interactive input option is to generate and view plotted output on a graphics device. If the input is expected from the terminal, the program prompts for the correct input at each stage – *e.g.*, at the command level, the prompt "COMMAND" is printed. Prompts for required input at the plot level are also given (N.B.: PLOT is a command level input). Prompts are not issued for other input required by major subroutines such as SETUP, FIELD2D, *etc.*. The program includes the option of switching between an input file and the terminal (one way only). With this feature, a complicated mesh may be defined in a file. At the end of the element generation, a command TERMINAL is used. The program then expects any additional input from the terminal.

Grid and Element Generation. The grid of points is generated separately from the elements. These grid points may be generated in sections, and within the section, the spacing of points can be nonuniform. There are several grid generation options that help minimize the required input.

The **MAP** element library includes quadratic triangles and bi-quadratic quadrilaterals. Also included is a looping feature that helps reduce the required input for element generation.

Units. **MAP** allows input and output units of length to be inches, millimeters, centimeters, or meters (default). All other quantities are in mks units.

Restart. **MAP** has a restart option. If the proper information is saved (*i.e.*, on disk) at the end of a run, the program can be reentered at the stopping point in subsequent runs. This feature is particularly attractive for the time marching used in the transient problems. It can also be used in conjunction with the element generation and plotting sequences.

Chapter 3.0 INPUT DATA STRUCTURE

The primary subroutines that make up **MAP** are called from the main program, MAP, according to a sequence of user specified commands. The specified routine is called and any additional data are read. When the subroutine ends, control is returned to MITMAP, and the next command is read. The general data structure consists of a sequence of commands separated by data. The command lines, their parameters, and subsequent data input are described in following sections. The order in which they appear is a natural one, but it is not the only one possible.

In this and remaining chapters, the following conventions are adopted:

- Upper case words indicate an alphanumeric input value – *e.g.*, SETUP. **MAP** expects alphabetic input to be in lower case. If the computer/terminal system used does not allow lower case input, then changes must be made in the code. **NOTE: Alphabetic input is only printed in upper case in this manual for emphasis.**

- Lower case words indicate a numerical input value – *e.g.*, imin.

- All input is in a free field form, starting in column 1, with successive entries separated by commas or by spaces. The maximum number of entries per line is twenty, and the maximum number of characters per entry is 20. Variables not specified – *i.e.*, by an entry of ,, when succeeding entries are made or by no entry after the last nonzero entry – are set to zero by the free field reader. **NOTE: spaces between entries are ignored. Therefore, to skip and entry ,, must be used.**

- { } denotes optional parameters.

- < > denotes a default value of an optional parameter (other than zero) set by the program.

- An input line is denoted by an asterisk (*). The actual input does not have the (*) and begins in column 1.

- All coordinates are given as r (radial) and z (axial). For planar problems x corresponds to r, and y to z.

- END commands, usually terminate a read sequence. Any words after the END are not used in the program, but may be used by the user for information purposes. For example, END or END, ELEMENTS are all acceptable lines to terminate the reading of element data.

The order in which the commands and associated data are presented is not necessarily the only order. However, this manual does group sets of commands into functional groupings.

The first grouping represents those commands which should or must be defined before any other commands. This up-front stuff (technical jargon) precedes the mesh or grid generation, which, in turn, precedes the processing stuff. Finally, the postprocessing commands are described. At the end, are those commands which don't seem to fit properly in any of the categories.

Chapter 4.0 UP-FRONT STUFF

4.1 Title Line

The title line must be the first line in the data file. There must be a $ in column 1. The title can be any sequence of 79 alphanumeric characters including blanks. The title will appear on each page of the output and on each plot. Plot titles have a limit of 40 characters. The title is saved for later restart. The form is:

(*) **$ {TITLE}**

4.2 PLANE Command

MAP can solve both planar and axisymmetric problems. The default is the axisymmetric problem. In order to solve planar problems, the PLANE option must be invoked. It is good practice to make this the first data line after the title, although it isn't required until just before FIELD, TRNSPRNT, POST or PLOT commands. The PLANE flag is saved and, therefore, need not be reset if the process is restarted – although it is good practice to repeat it. The form is:

(*) **PLANE**

4.3 UNITS Command

MAP has the capability of accepting units of length in inches, centimeters, millimeters or meters (default). The unit of length is defined on the UNITS line, and holds for both input and output (*i.e.*, – if coordinates are input in inches, then the output has all coordinates in inches.) Only the coordinates are affected by the units command. All other input and output, such as fields, forces per unit length, current density, *etc.* are in the mks system.

The coordinates are actually stored in meters and processed as such by **MAP**. The UNITS flag is saved and, therefore, need not be reset if the process is restarted. The command must be issued before SETUP, RESTART, PLOT, *etc.* The form is:

(*) **UNITS,TYPE**

where

TYPE= IN, MM, CM, or M, corresponding to units of length in inches, millimeters, centimeters, and meters, respectively.

4.4 **PROBLEM** Command

The PROBLEM command is primarily used in the version **MAP2DJ** for flagging the kind of analysis. It is, however, a required entry for **MAP**. The form is:

(*) **PROBLEM,TYPE,{sysfact}**

where

TYPE = FIELD2D – solve the two-component field penetration problem.

{*sysfact*}– a factor applied to system totals. **MAP** calculates system totals such as energy, power, *etc.* over the mesh defined. Usually this is less than the full problem since symmetries are used to reduce the problem size. If the model is only defined in the plane $z > 0$, a $\{sysfact = 2\}$ would multiply the results of the integrals over the model by 2 in order to get the total for the system. < 1 >

The PROBLEM flag and other parameters are saved and need not be reset for a restart sequence. However, resetting the flags does no harm, and allows the parameters to be reprinted into the new output file.

5.1 SETUP Command

A new problem requires the definition of material properties and the generation of an element mesh. The subroutine SETUP serves as a driver for this portion of the program. SETUP is called with the command:

(*) **SETUP,{iprint}**

where

iprint– is a print level indicator. The amount of output increases with *iprint*. If *iprint* = 1 only material properties, the number of grid parts defined, and the number of elements defined are printed. If *iprint* = 4 material properties, grid point coordinates, and element connectivity and coordinates are printed. The default value causes materials, and element connectivity and coordinates to be printed. Any *iprint* greater than one will generate a large amount of output. < 1 >

Following the SETUP command line, the subroutine expects three sets of data corresponding to (in order) material properties, grid point definition, and element generation. Each set is terminated by an END command, and the third END causes SETUP to terminate and return control to MITMAP for further commands. Earlier versions of the program required boundary condition input during the setup phase. This has been shifted to after the FIELD2D or THERMAL command.

After the final END line in the SETUP phase, a binary disk file (FOR083.DAT) is written with the problem title, plane and units flags, and the PROBLEM information. Also written is all the mesh data including equation numbering. This file is necessary for a RESTART sequence.

5.1.1 Material Properties

Immediately after a SETUP command, the program expects data defining the material properties for the problem. As discussed later, each finite element in the mesh has a material number associated with it. This number refers to the material with properties defined below.

The form of input is:

(*) **NAME,n,{j_0,σ,ironfrac,nonet,ij}**

where

NAME – an alphanumeric identifier for each material, (*e.g.*, –STEEL.)

n – an integer identifier for each material. Each finite element has an material number associated with it that defines the material properties for the element. The maximum number of different materials is 20.

j_0 – conductor current density (A/m^2) either constant or as a multiplier for a curve of j versus time as described below..

σ – electrical conductivity (mho/m).

$ironfrac$ – a multiple use flag for determining the iron properties.
$\leq$ 1.0 – nonlinear BH curve described below.
$>$ 1.0 – constant μ_r. $\mu = \mu_0\mu_r$.

$nonet$ – the entire region of elements with this material number will have a zero net current – imposed as a constraint on the problem via a LaGrange multiplier.

ij – a flag for file input of j versus time. If $ij > 0$ then use data set number ij from the file JVST.DAT (in the present directory). The form of the data file is (free-format):

number of sets
number of points in set 1
t_1 $j(t_1)$
t_2 $j(t_2)$
$\vdots$ $\vdots$
t_n $j(t_n)$
number of points in set 2
t_1 $j(t_1)$
t_2 $j(t_2)$
$\vdots$ $\vdots$
t_n $j(t_n)$
etc.

NOTE: The program does a linear interpolation of the data in the file and then multiplies the result by the normalizing factor on the material line, j_0. Therefore, in the case of nonnormalized current density, either the normalizing factor on the material line should be set to 1.0, or the data in the file should be divided by j_0.

The identifier NAME is just a label for the properties associated with the material number, n. The postprocessing section of the program sums certain quantities over all elements with the same material number – *e.g.,* forces, Joule losses, *etc.* These quantities are then printed under the name, NAME.

There are several options for the driving function for transient problems. The simplest is to define a "stiff" current region by setting a current density and a zero conductivity. The effect is to impose a current in the region which is not affected by the changing magnetic field. This is equivalent to a constant current power source.

MAP can also be used to model a constant voltage condition. If both a current density and a nonzero conductivity are specified, the program interprets the entry j_0 to be a voltage. The program calculates the current density at each time step to be:

$$j = \sigma\{j_{specified} - \dot{A}\} \tag{5.1}$$

where $j_{specified}$ is actually interpreted as an electric field strength. At present, **MAP** does not invoke any sort of constraint of uniform current. In a transient problem with a voltage source, the currents in each element of the coil will not necessarily be equal. **NOTE:** This feature is invoked for static problems also. Therefore, no conductivity should be given for static problems.

Presently, **MAP** only allows one BH curve to be defined. This curve is internal to the program. The present curve is listed in Table 5.1 and is shown in Figure 5.1. **MAP** has a parameter, iron fraction, that may be used to modify the BH curve as is shown in Figure 5.1. This parameter may be varied from zero to one. The effect of this parameter is to change the BH curve. If P is the iron fraction, then:

$$B(P) = P\{B(1) - H\} + H \tag{5.2}$$

where $B(1)$ is the value of B for $P = 1$ – that is, no degradation of the curve. Modifications to **MAP** to allow more than one BH curve to be defined are easily made. Subsequent versions will have this capability.

The material properties are terminated with an END line.

TABLE 5.1
MAP – BUILT-IN BH CURVE

B (T)	H (A/m)	μ_r	$\nu = 1/\mu_r$	$\frac{\partial \nu}{\partial B^2}$
0.0000E+00	0.0000E+00	3.9930E+03	2.5044E−04	0.0000E+00
8.9443E−01	1.7825E+02	3.9930E+03	2.5044E−04	9.0461E−05
1.2000E+00	2.9444E+02	3.2432E+03	3.0833E−04	2.7244E−04
1.4000E+00	5.0134E+02	2.2222E+03	4.5000E−04	7.4943E−04
1.5000E+00	7.9657E+02	1.4985E+03	6.6733E−04	1.7541E−03
1.5500E+00	1.1531E+03	1.0697E+03	9.3484E−04	3.0169E−03
1.6000E+00	1.7953E+03	7.0922E+02	1.4100E−03	4.7385E−03
1.6500E+00	2.8624E+03	4.5872E+02	2.1800E−03	6.3284E−03
1.7000E+00	4.3831E+03	3.0864E+02	3.2400E−03	6.3768E−03
1.7500E+00	6.0439E+03	2.3041E+02	4.3400E−03	7.4930E−03
1.8000E+00	8.1217E+03	1.7637E+02	5.6700E−03	8.3302E−03
1.8500E+00	1.0585E+04	1.3908E+02	7.1903E−03	9.6519E−03
1.9000E+00	1.3608E+04	1.1111E+02	9.0000E−03	1.0909E−02
1.9500E+00	1.7225E+04	9.0090E+01	1.1100E−02	1.1139E−02
2.0000E+00	2.1168E+04	7.5188E+01	1.3300E−02	1.5309E−02
2.0500E+00	2.6754E+04	6.0976E+01	1.6400E−02	1.8313E−02
2.1000E+00	3.3757E+04	4.9505E+01	2.0200E−02	2.2722E−02
2.1250E+00	3.8217E+04	4.4248E+01	2.2600E−02	2.8070E−02
2.1500E+00	4.3799E+04	3.9063E+01	2.5600E−02	4.3466E−02
2.1750E+00	5.2443E+04	3.3004E+01	3.0300E−02	6.7659E−02
2.2000E+00	6.6002E+04	2.6525E+01	3.7700E−02	8.0248E−02
2.2500E+00	9.9471E+04	1.8000E+01	5.5555E−02	8.2708E−02
2.2797E+00	1.2094E+05	1.5000E+01	6.6667E−02	8.2212E−02
2.3069E+00	1.4121E+05	1.3000E+01	7.6923E−02	8.0314E−02
2.3443E+00	1.6959E+05	1.1000E+01	9.0909E−02	7.7065E−02
2.3996E+00	2.1217E+05	9.0000E+00	1.1111E−01	5.9373E−02
2.5085E+00	2.8517E+05	7.0000E+00	1.4286E−01	8.6594E−02
2.5627E+00	3.3989E+05	6.0000E+00	1.6667E−01	5.9041E−02
2.6706E+00	4.2504E+05	5.0000E+00	2.0000E−01	5.0540E−02
2.8498E+00	5.6695E+05	4.0000E+00	2.5000E−01	3.8476E−02
3.2074E+00	8.5078E+05	3.0000E+00	3.3333E−01	2.7574E−02
3.5644E+00	1.1346E+06	2.5000E+00	4.0000E−01	1.7863E−02
4.2782E+00	1.7023E+06	2.0000E+00	5.0000E−01	1.1418E−02
4.8134E+00	2.1280E+06	1.8000E+00	5.5555E−01	7.4036E−03
5.7052E+00	2.8375E+06	1.6000E+00	6.2500E−01	4.8174E−03
6.4186E+00	3.4052E+06	1.5000E+00	6.6667E−01	3.1996E−03
7.4887E+00	4.2567E+06	1.4000E+00	7.1429E−01	0.0000E+00

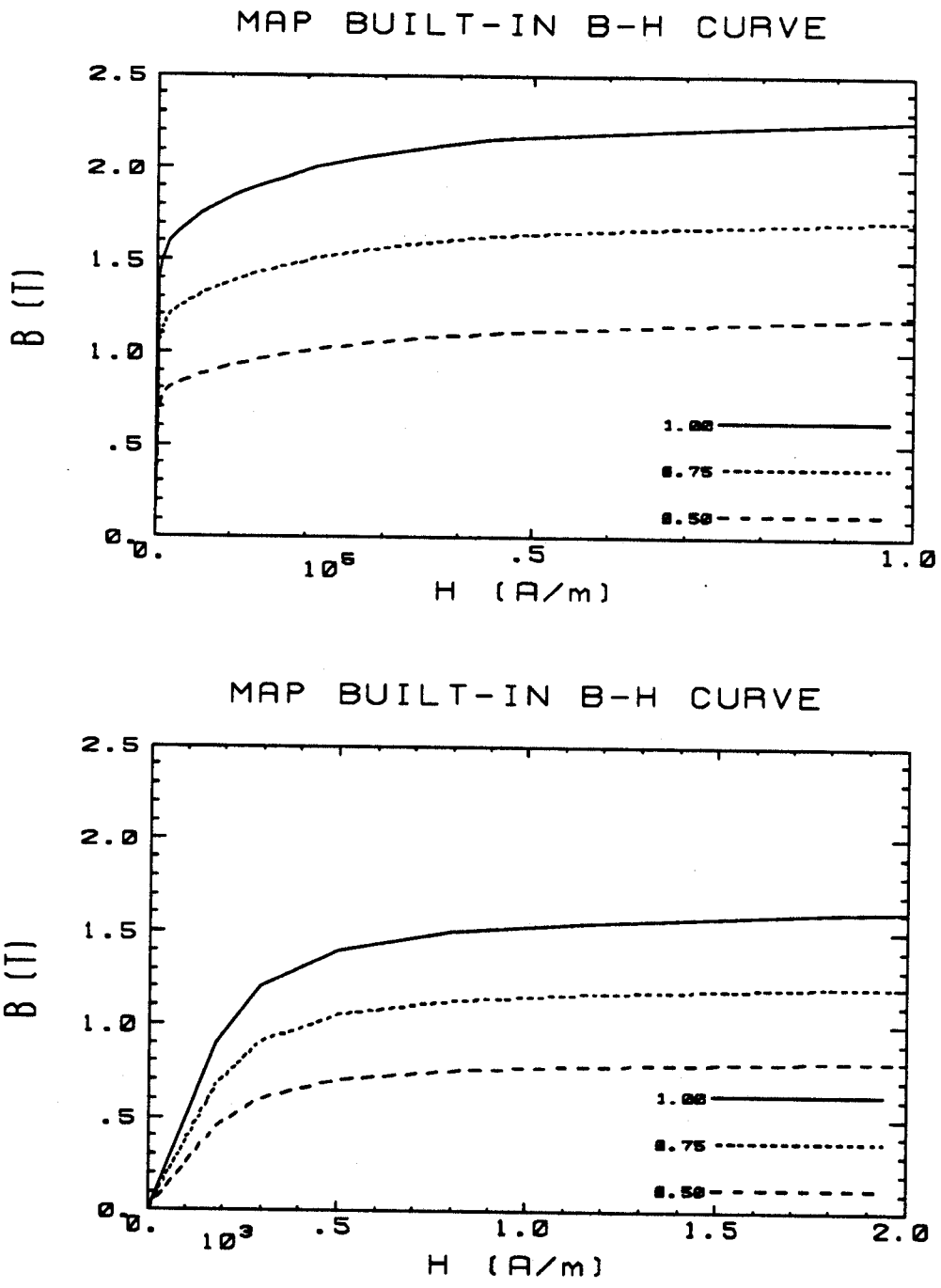


Figure 5.1 **MAP** Built-in B-H Curve

5.2 Grid Generation

After the material properties have been defined, the grid of points must be defined. **MAP** uses two sets of coordinates. There are the physical (r,z) or (x,y) coordinates and a set of logical (i,j) coordinates associated with each point. The logical coordinates range from $(1,1)$ to a maximum of $(maxi,maxj)$ (see Appendix B for the common block storage and dimensions). Each point (which may later become a corner node of an element or elements) in the problem is assigned both sets of coordinates. **Midside nodes are not given either (r,z) or (i,j) pairs.**

The grid generation begins with the definition of the two sets of coordinates of each point in the grid. Once the grid has been generated each element in the problem is defined by giving the (i,j) pairs of each of the corners of the element along with a material property number.

Points may be defined several times; **only the last definition is used.** This allows boundaries of regions to be referenced more than once. Similarly, if the spacing is regular everywhere except at a point or few points, then the regular spacing can be obtained over the part, and the few perturbations defined later.

Points may be defined that will not be used in the element definition. However, the equation solver in **MAP** orders the equations to produce a minimum storage based on the (i,j) or element ordering. Missing interior (i,j) pairs can cause larger storage requirements to be calculated than are actually necessary.

The grid of points can generated by parts. A part is defined as any of the possible input data sets defined below. There are six ways of defining a part containing one to any number of grid points. The most basic way is to define the corner points of a quadrilateral region and allow the program to fill in the intermediate points. **MAP** uses an isoparametric mapping with the option of nonuniform spacing within each part.

As described later, there are some looping features available for very regular (rectilinear) meshes that loop on parts. These features use a part number or numbers. It is up to the user to keep track of the part numbers (a good use of the in-line comment feature).

5.2.1 Basic Grid Command

The basic part definition is made up of a set of three input lines defined as:

(*) **imin,jmin,{imax,jmax,g1,g2,g3,g4,POLAR,ro,zo}**
(*) **r1{,r2,r3,r4,r5,r6,...,r12}**
(*) **z1{,z2,z3,z4,z5,z6,...,z12}**

where

imin,jmin,{imax,jmax}– are the (i,j) limits for the part. That is, the lower left and upper right corners in (i,j) space. The differences between the values determines the number of intermediate points.

$\{g1,g2,g3,g4\}$ – are the gradients for the grid point spacings on the four sides. The default value, < 1 >, gives an equal spacing. The gradients are defined as:

$g1$ = (length of left bottom)/(length of right bottom)
$g2$ = (length of right bottom)/(length of right top)
$g3$ = (length of left top)/(length of right top)
$g4$ = (length of left bottom)/(length of left top)

{POLAR} – indicates that the following r values are interpreted as radii and the z values as angular degrees referred to the local origin ro< 0 >, zo< 0 >. **NOTE. The angles are measured positive from the r (x) axis.**

$r1\{,r2,r3,r4,r5,\ldots,r12\}$
$z1\{,z2,z3,z4,z5,\ldots,z12\}$– are the coordinates of the four corner and eight side grid points for the part. If the side is straight only the two corner points are specified. If the side is curved either one or two interior points must be given. If one point is given, the mapping is quadratic and the point should be placed approximately at the middle of the side. If the two points are given, the mapping is cubic and the points should be placed at the 1/3 points along the side. Note that these mapping coordinate points need not be grid points.

Figure 5.6 shows the general scheme of the grid generation. The numbers associated with each of the points correspond to the location on the grid coordinate lines. The figure on the right represents a mapping of the region modeled in logical coordinate, (i,j), space. The three figures on the right represent the possible grid generated.

The top figure is generated with the default gradient spacing. The spacing of points on

the sides is uniform. Four corner coordinates are defined and, hence, the part has straight sides.

The middle figure is also straight sided – implying only four corner point coordinates were defined. The nonuniform spacing is attained with the use of gradients. The gradients are all less than one. $g1 = 0.2$ or the length of the segment at the lower left is one-fifth the segment length at the lower right corner.

The lower figure was generated by specifying four corner points and four (approximately) midside points in order to generate boundaries which are not straight (they are made up of piece-wise straight segments). The spacing is uniform.

Figure 5.7 illustrates several different grid parts that can be generated by a single set of the three data lines. Figure 5.7a shows a rectangular mesh of 25 evenly spaced points; 5 points in each direction can be generated by the set shown. The grid covers $(r, z) = (3, 0)$ to $(r, z) = (7, 8)$.

A single line of points may be defined by giving the same (I) (or (J)) and r (or z) coordinates as is shown in Figure 5.7b. The set generates 5 equally spaced points between $(r, z) = (3.5, 0)$ and $(r, z) = (3.5, 6)$.

A single point may be defined by either of the inputs shown in Figure 5.7b. In this case, only the (i, j) and (r, z) coordinates of the single point need be given. Alternatively, the POINT command defined below may be used.

A polar grid is shown in Figure 5.7c. In this case, the z-coordinates are interpreted as angular coordinates. The default value of the local origin $(ro, zo) = (0, 0)$ is used.

Finally, a more complicated set is shown in Figure 5.7d. Here, the curved boundaries are modeled with a single midside point. It can be seen that the default straight-sided edges are generated by defining only the two end points. The gradients cause the nonuniform spacing on the lower and right-hand sides.

Relatively complicated problems may require the definition of on the order of one hundred different parts. At three data lines per part, the required input becomes quite unwieldy. In order to help reduce the amount of initial data input, there are three grid part commands that are incorporated into **MAP**. These commands take advantage of the regularity of the mesh generation. Presently, these commands are valid only for straight-sided parts (or for regular annular regions with the POLAR option). These three commands are described below.

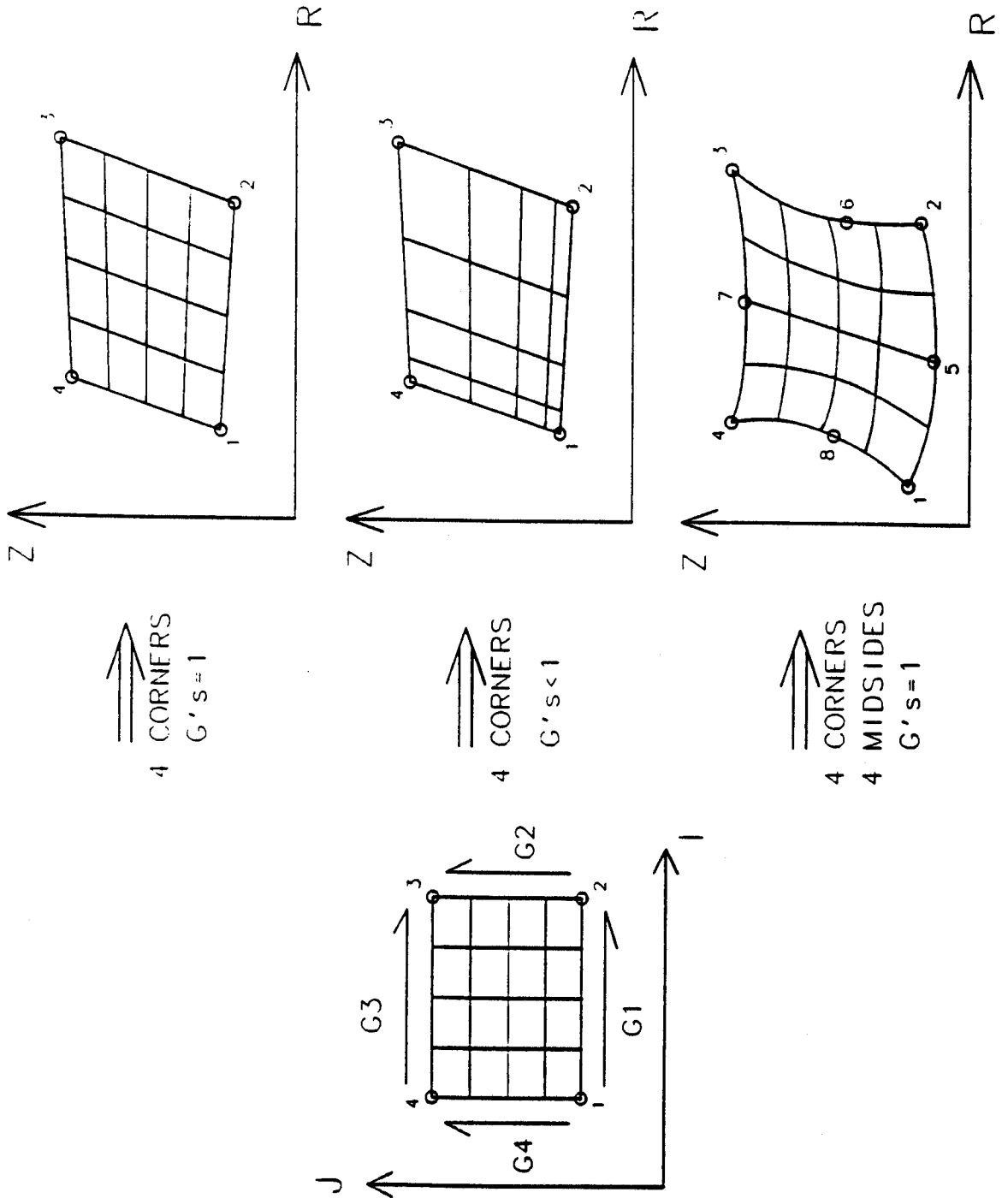


Figure 5.6 **MAP** Grid Generation Scheme

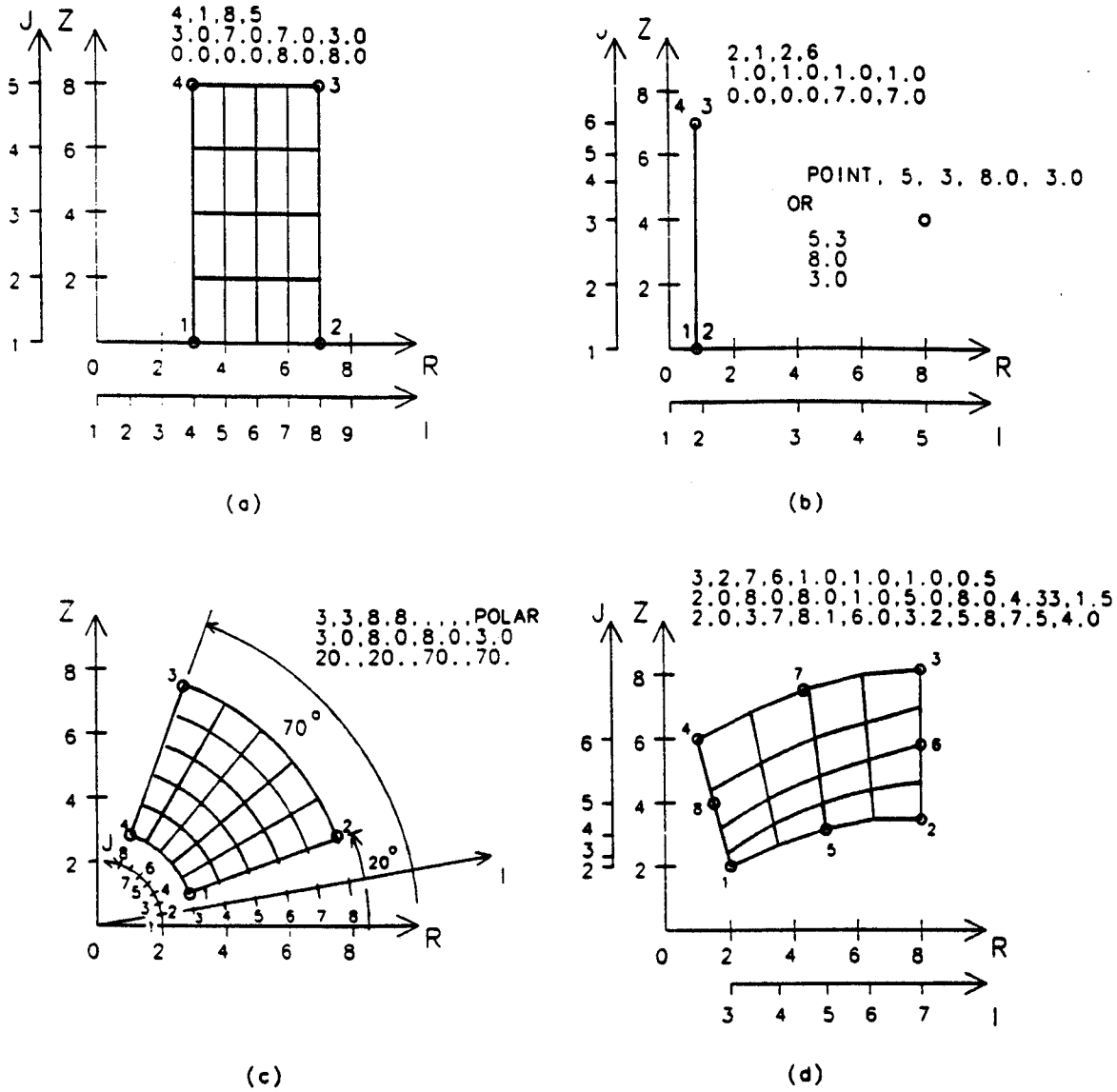


Figure 5.7 MAP Grid Generation - Examples

5.2.2 REGULAR Command

The first of the special grid commands may be used when the part to be generated is rectangular - see Figure 5.7a. For a rectangular part generated with the standard three line input, the entries $r1, r2, r3, r4$ and $z1, z2, z3, z4$ have two redundant entries each. The REGULAR command requires only $r1, r3, z1$, and $z3$.

The form of the command is:

(*) **REGULAR,imin,jmin,imax,jmax,{g1,g2,g3,g4,POLAR,ro,zo}**
(*) **rmin,zmin,rmax,zmax**

where

REGULAR – is the flag to invoke the special option.

rmin, zmin
rmax, zmax – are the minimum and maximum coordinates for the part. NOTE. The part must be rectangular (or a regular sector of an annulus for the POLAR option).

$g1, g2, g3, \ldots$ – as defined above.

Using this option reduces the required input by one line and four entries. The grid part in Figure 5.7a could be generated with the REGULAR option with the input as:

REGULAR,4,1,8,5
3.0,0.0,7.0,8.0

5.2.3 IREPEAT (JREPEAT) Command

Another possible part generation command takes advantage of the regularity of the placement of the grid parts. In many problems not only are the individual parts rectilinear, but there is also a repetition of many of the inputs between parts. For example, a sequence of rectangular parts with constant I and r limits, but with variable J and z limits is often needed. For rectangular regions, the standard part input would require 12 inputs ($imin, imax, jmin, jmax, r1 - r4$, and $z1 - z4$). Six of these would not change for the next part (say $imin, imax$, and $r1 - r4$).

The IREPEAT and JREPEAT options take advantage of this repetition of input to reduce the total number of required entries. These commands may be used to generate more than one part. That is, if the sequence of constant I and r limit parts discussed above is generated and, if the next set of parts is merely a J and z limit offset, the looping feature described below may be used to generate the next set of parts.

The general forms of the repeat commands are:

(*) **IREPEAT,n1,n2,jmin,jmax,z1,z2,z3,z4,{g2,g4}**

or

(*) **JREPEAT,n1,n2,imin,imax,r1,r2,r3,r4,{g1,g3}**

where

IREPEAT
(JREPEAT) – are the flags invoking the repeat option.

$n1, n2$ – are loop limits of the parts to be repeated (with the changes in the appropriate data). For example, if part 7 is to be repeated, then n1=7 and n2=7. If, on the other hand, parts 5 though 10 are to be repeated, then n1=5 and n2=10.

$jmin, jmax$ – are the new J (I) limits of the new part(s).
($imin, imax$) – The I (J) limits do not change.

$z1, z2, z3, z4$ – are the new z (r) limits for the new part(s)
($r1, r2, r3, r4$) – The r (z) limits do not change.

$g2, g4 (g1, g3)$ – are the gradients in the z (r) direction for the new part(s). The $g1, g3$ ($g2, g4$) gradients do not change.

If the POLAR option was invoked on the part to be repeated, then it is also invoked for the new part with the same local origin as defined for the first part.

The command IREPEAT implies repeat the I's, r's, $g1$ and $g3$ of the previous part(s) and use new J's, z's, $g2$, and $g4$. The command JREPEAT implies repeat the J's, z's, $g2$, and $g4$ of the previous part(s) and use the I's, r's, $g1$, and $g3$. If more than one part is to be repeated – *e.g.*, $n1 = 3$ and $n2 = 4$ – then the first new part repeats the appropriate data from part 3 and the second new part repeats the data from part 4.

Parts are numbered sequentially as they are generated. The user must keep track of the correct numbering in order to take advantage of the repeat options. Parts generated with the repeat option are also numbered. It is suggested that the user employ the in-line comment option to keep track of part numbers in the data set.

Figure 5.8 illustrates the use of the repeat options. Only the part limits are indicated. The first part is generated with a REGULAR part definition. Parts 2 through 5 are generated using the IREPEAT option with no loop. Parts 6-10 are generated with the JREPEAT option and looping, as are parts 11-15 and parts 16-20. It can be seen that gradient spacing may be changed.

5.2.4 POINT Command

There are two additional grid generation commands that can be used to minimize the input required to define a mesh. These commands are the POINT and FILL commands. The POINT command is simply a single data line input to define a single point. The FILL command allows the program to fill interior data points from previously defined corner data.

The form of the POINT command is:

(*) **POINT,i,j,r,z**

where

POINT – flag to invoke this command.

i, j – logical coordinates of the point.

r, z – physical coordinates of the point.

A POINT command equivalent to the three data line set shown in Fig. 5.7c would be:

(*) **POINT,5,3,8.0,3.0**

5.2.5 FILL Command

The final grid generation command is the FILL command. This option can be used to fill interior points when the four corners have been previously defined by any of the commands described above. The FILL command fills both the interior and intermediate boundary points.

The form of the command is:

(*) **FILL,imin,jmin,imax,jmax{,g1,g2,g3,g4}**

where

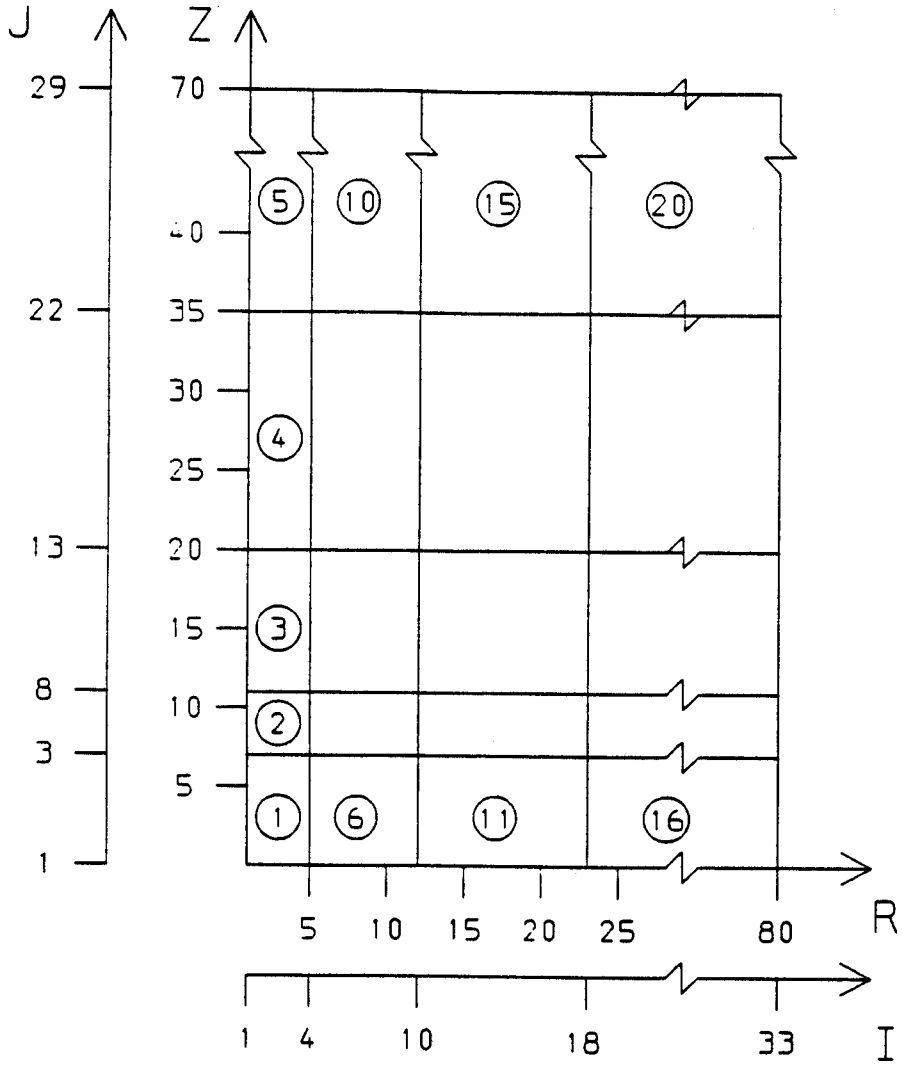
FILL – flag invoking the FILL option.

$imin, jmin, imax, jmax$ – the region logical minimum and maximum coordinates.

$g1, g2, g3, g4$ – the gradient for spacing as defined previously.

The FILL command may be used (carefully) with the POINT or other grid commands to define all the nodes on the boundaries of the problem – *e.g.,* the inner and outer boundaries of a TF coil – and, then fill the interior points. **NOTE:** The filling can not be accomplished with a single FILL command, since, as written, the intermediate boundary points are also recalculated. Instead, a sequence of FILL commands is used. Such a sequence would fill the interior using either a single boundary point on one edge and a single point on the opposite edge (*e.g., imin* = *imax* or *jmin* = *jmax*) or two boundary points on each edge (since there would be no intermediate points on the boundary to be recalculated). An example of this option is given in Chapter 11.

The grid generation phase of SETUP is halted with an END or END,GRID line.




```
REGULAR,1,1,4,3                                          !PART 1
0.0,0.0,5.0,7.0
IREPEAT,1,1, 3, 8, 7.0, 7.0,11.0,11.0                    !PART 2
IREPEAT,1,1, 8,13,11.0,11.0,20.0,20.0                    !PART 3
IREPEAT,1,1,13,22,20.0,20.0,35.0,35.0                    !PART 4
IREPEAT,1,1,22,29,35.0,35.0,70.0,70.0,0.10,0.10          !PART 5
JREPEAT,1,5, 4,10, 5.0,12.0,12.0, 5.0                    !PARTS  6-10
JREPEAT,1,5,10,18,12.0,23.0,23.0,12.0                    !PARTS 11-15
JREPEAT,1,5,18,33,23.0,80.0,80.0,23.0,0.10,0.10          !PARTS 16-20
```

Figure 5.8 MAP IREPEAT and JREPEAT Options

5.3 Element Generation

Once the grid of points has been generated, the elements must be defined. The elements are defined by giving a material number and the logical coordinates of corners. The element definitions are of the form:

(*) **ELEMENT TYPE, material no.,i1,j1,{i2,j2,i3,j3,i4,j4}**

where

ELEMENT TYPE– is an alphanumeric identifier defined below.

material no. – is the material (integer) identifier defined previously.

$i1, j1, \{i2, j2, \ldots\}$ – are the (i, j) pairs for each corner node taken in a counterclockwise manner.

The ordering of the nodal points on the element line must be counterclockwise (*e.g.*, r (x) cross z (y) according to the right-hand rule). If the ordering is not as required, then a negative Jacobian is calculated in the stiffness routine and the process stopped.

The element types available are:

TRI – triangle with quadratic interpolation of B or T.

QUAD– quadrilateral with biquadratic interpolation of B or T.

The element library is shown in Fig. 5.10 with the nodes and sides numbered. The corner node numbering is counterclockwise. The sides are also numbered counterclockwise and are used in the specification of element-based boundary conditions discussed in a later section.

Note. **All elements are straight-sided and the midside nodes are not given (i, j) pairs in either the grid or element generation.**

If the element is a quadrilateral and if the ordering of the (i,j) pairs is such that:

$$i2 = i3 = i1 + 1 \; ; \; i4 = i1$$
$$j3 = j4 = j1 + 1 \; ; \; j2 = j1$$

as shown in Fig. 5.9.

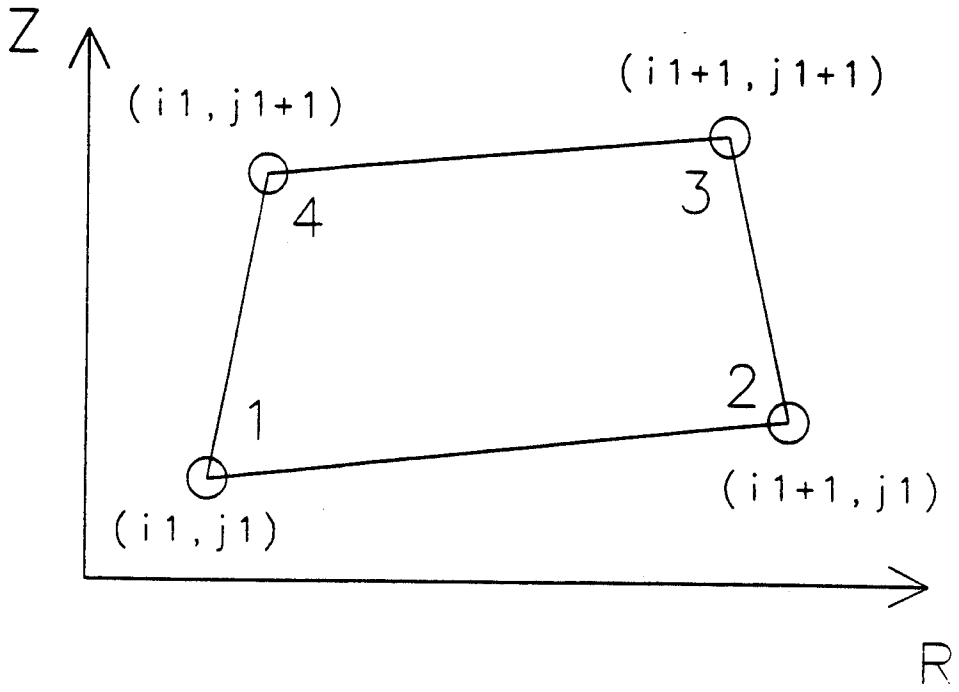


Figure 5.9 Regular Quadrilateral Nodal Point Numbering

Then only the pair $(i1, j1)$ need be specified. If the ordering is not as above, all four (i,j) pairs are needed. Similarly, if the element is a triangle, the three corner (i,j) pairs must be specified.

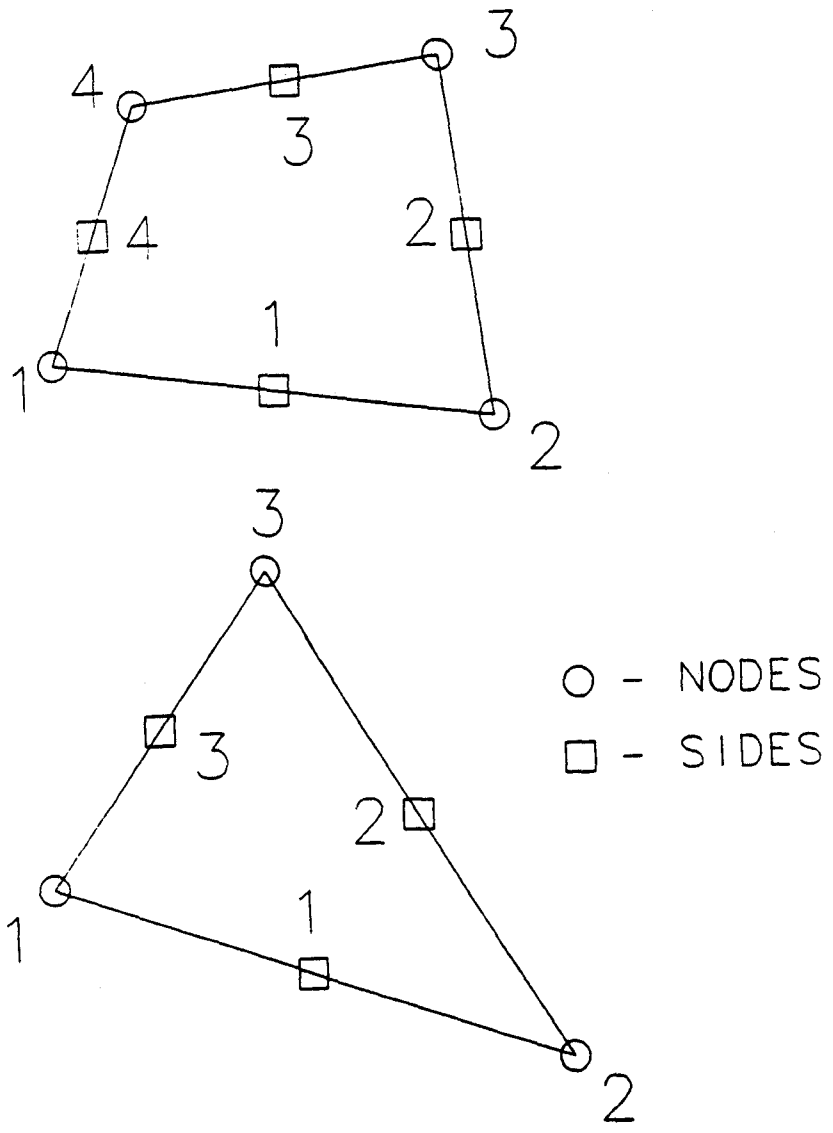


Figure 5.10 MAP Element Library

5.3.1 Looping Features

In order to facilitate the element generation phase, a looping feature is used. The two types of loops, ILOOP and JLOOP, are allowed. The loops serve the same purpose as a FORTRAN DO loop. They may be nested, but two loops on the same integer cannot be nested. The form of the loops is:

(*) **ILOOP (JLOOP), npass, {inc}**

.
.
.

(*) **IEND (JEND)**

where

ILOOP(JLOOP) – all values of $I(J)$ in all the element lines between the ILOOP(JLOOP) and IEND(JEND) are to be incremented by inc.

npass – number of times the loop is to be executed.

inc – the increment added to each $I(J)$ value in the loop during each pass. $< 1 >$.

For example, the data:

```
ILOOP,4,1
JLOOP,4,1
QUAD,2,4,1
JEND
IEND
```

would generate 16 biquadratic quadrilateral elements of material type 2 in the region (i,j)=(4,1) to (8,8) (see Figure 5.7a). Note that the increment is applied at the end of the loop. That is, QUAD,2,4,1 is generated first, then QUAD,2,4,2 ..., then QUAD,2,5,1 ..., etc.

The major difference between these loop commands and the equivalent FORTRAN do-loops is that the $I(J)$ value is incremented after a pass through the loop rather than at the start of a loop. The equivalent FORTRAN coding for the above example would be something like:

```
I = 4
DO II=1,4,1
     J = 1
     DO JJ=1,4,1
                .
          NEL=NEL+1
          DEFINE ELEMENT NEL WITH CORNER (I,J)
                .
          J=J+1
     CONTINUE
     I=I+1
CONTINUE
```

The actual coding is complicated by the fact that there may be another JLOOP command immediately following the JEND (before the IEND). In addition, the sequence could have been JLOOP then ILOOP ... IEND and JEND. The same elements would have been generated, but in a different order.

In the example above, it would possible to insert another JLOOP ... JEND sequence between the first JEND and IEND. That is, multiple nesting is possible. It would not be possible to insert another ILOOP ... IEND set, however. Such an illegal nesting would produce an error message and the process would terminate.

5.3.2 MAT command

Even with the looping features described above, the generation of the elements for a complex problem with many material boundaries can require a great deal of input. In order to facilitate the element definition portion of the input, the MAT command is included.

The MAT command allows the user to change the material number (*i.e.*, material property) of a previously defined element. The advantage of this feature is that all elements may be defined (for regular meshes) with a single set of loops (one in each direction). The material property given all elements should be the predominant material in the problem. Then, the MAT command can be used to change the material number of those elements with different numbers.

The MAT command is invoked with the line:

(*) **MAT,material no.,i1,j1**

where

MAT – is the flag that invokes the MAT option

material no.– is the desired material number of the element.

$i1, j1$ – is the I, J pair of the first node of the element whose material number is to be changed. **NOTE. Care must be taken to assure that the I, J pair is indeed the first node.**

The ILOOP(JLOOP) feature may also be used with the MAT command. Examples of the use of the MAT option are given in Chapter 11.

Care must be taken to use the MAT command and not a QUAD or TRI command. **MAP** does not check to see if elements are coincident (superposed one on top of another) as they would be if this mistake were made.

The element generation is stopped with an END or END,ELEMENTS line.

5.4 Equation Numbering

After the material properties, grid, and element definitions are completed, SETUP calls subroutine EQNNO. In EQNNO, the equation numbering for the mesh is calculated. **MAP** first searches for the minimum storage ordering. The possible orderings for minimum storage are based on: (1) an element ordering (in the order that they are generated); (2) a logical coordinate ordering with the J coordinate direction searched first; and (3) a logical coordinate ordering with the I coordinate first.

The program uses a skyline-based in-core solver [3]. It is possible to use frontal (element-based) solvers. Previous incarnations of the code have had this option.

The program prints out the storage requirements for the various orderings and the chosen ordering. If the storage requirements exceed the dimensioned values, the program halts execution. If the storage is sufficient, the program checks for boundary condition definitions. If the next line of input is not a boundary condition, the SETUP phase ends and control is returned to MITMAP. If boundary conditions are specified at this point, they are processed as described in Section 6.3.

At this point in the process, the elements may be plotted. Indeed, for a new grid, it is recommended that a plot of the elements be made and the program stopped with a STOP command. The plot or plots may be viewed and the grid checked before proceeding with the processing. The main reason for specifying BCs in the SETUP phase is to allow the user to generate plots of the elements with the boundary conditions shown for check purposes. If such a check is not used, then BC definitions (and check plots) may be made later in the execution.

If multiple materials are used in the mesh, it is recommended that all elements of each material type be plotted separately (an option for the PLOT,ELEMENTS command) in addition to the entire mesh. **MAP** does not check to see if more than one element has been defined with the same set of (i,j) corner coordinates. An inadvertent mixing of QUAD and MAT commands can easily generate elements on top of elements. These element plots by material type can be used to trap such errors.

The SETUP phase ends at this point. Processing or postprocessing commands are expected next.

Chapter 6.0 PROCESSING COMMANDS

6.1 FIELD2D Command

The major processing routine for **MAP** is FIELD2D. FIELD2D calls the subroutines that form and solve the set of algebraic equations that arise from the weak or Galerkin form of the governing partial differential equations. FIELD2D will call the postprocessing routine to calculate, print and/or write plot files. The subroutine FIELD2D is called with a command line of the form:

(*) **FIELD2D, TYPE,{itmax, tolit, prntlvl, tfinal, delt, tolss, θ, ipltlvl, prntfrq, pltfrq, modit, rlx, monit, inode, jnode, orlx, urlx, mstart}**

where

TYPE – solution technique used to solve algebraic equations.
= LINEAR linear materials only – no iteration.
= NEWTON nonlinear material present use Newton-Raphson iteration.
= MODNEWTON nonlinear material present use modified Newton iteration.

$\{itmax\}$ – maximum number of nonlinear iterations. < 15 >

$\{tolit\}$ – tolerance for iterative convergence: $\max\{\Delta \underline{A}_{(i+1)}\}/ \max\{\underline{A}_{(i+1)}\}$ < .001 >

$\{prntlvl\}$ – printed output level indicator. All levels except 2 and 4 will generate total forces by material name, system totals (power, energy, *etc.*), maximum and minimum field, and current density by material. In addition, if
= 0 Element forces, nodal point field and current density components, summary minima and maxima.
= 1 Element forces, central field ($(i,j) = (1,1)$) and summary minima and maxima.
= 2 Element forces, central field, and field and current density components at any node with current density components, summary minima and maxima (useful with transient problems).
= 3 Summary minima and maxima.
= 4 No printed output.
= 5 User defined range of nodal (i,j) values printed. Input of the (i,j)'s come after any boundary conditions. Form of input is: $i1, j1, i2, j2$. Multiple sets may be input on successive lines. An entry of END terminates definition.

$\{tfinal\}$ – final time for the time marching algorithm.
= 0 - steady state solution is sought (default).

$\{delt\}$ – time step increment.

$\{tolss\}$ – tolerance for convergence of time marching to steady state. Test is max $\{(\underline{A}^{n+1} - \underline{A}^{n})/\underline{A}^{n+1}\}$. < 0.001 >

$\{\theta\}$ – time marching parameter θ, $0 \leq \theta \leq 1$.

= 0.0 forward difference;
= 0.5 Crank-Nicholson;
= 1.0 backward difference.

{*ipltlvl*} – last digit of the logical unit numbers of the binary files of plot data written every *pltfrq* time steps.
= 0 all files written
FOR082 - element forces
FOR084 - nodal point flux density components.
FOR085 - nodal point vector potential
FOR086 - iron magnetizations
FOR087 - system totals
= ijk then files FOR08i, FOR08j, and FOR08k are written. Allows from one to three files. *.e.g.*, = 57 implies write FOR085 and FOR087.

{*prntfrq*} – frequency of printed output - print at every {*prntfrq*} time steps. < 1 >

{*pltfrq*} – frequency of plotted output.< 1 >

moditer} – number of modified Newton iterations before recalculation of l.h.s.< 1 >

{*rlx*} – relaxation factor used if no monitoring.

{*monit*} – monitor flag if = 1 implies monitoring.

{*in*, *jn*} – (i,j) of the node to be monitored. If monit = 1, and $(in, jn) = (0,0)$ are zero, the program calculates the equation number to be monitored as max(eqn.no.)/2.

{*orlx*} – over relaxation factor for accelerating monotone convergence.< 1.2 >

{*urlx*} – under relaxation factor used to dampen oscillatory behavior.< 0.8 >

{*mstart*} – iteration number at which monitoring and relaxation is to begin.< 3 >

On the NERSC CRAY version the files FOR0XX.DAT are named FILEXX.

A description of the printed output is given in more detail in Chapter 9. Nodal values of the current density are averaged from the contributing elements. It is possible to get the unaveraged element contributions printed. This is accomplished with a $prntfrq < 0$ input. Then the unaveraged element by element (corner) nodal point values of B are printed every $|prntfrq|$ time steps. If the unaveraged nodal point option is used, element forces are not printed out.

The nodal point flux density components are written on FOR084.DAT at each *pltfrq* time step. If $prntlvl = 5$, file FOR084.DAT will contain only those nodes specified with an $prntlvl = 5$. All other *prntlvl*'s will write all nodal point values of the current density on FOR084.DAT.

MAP – 6–2

Boundary conditions are defined at this point in the solution process. The assumed initial condition for the problem is that the vector potential is zero everywhere at time $t = 0^-$. If a nonhomogeneous initial condition is desired, it can be accommodated by the INITIALA option discussed later.

The FIELD2D subroutine examines the next data line in the input file for a boundary condition or an END line. (BCs may have been defined in SETUP). If no boundary conditions are specified then the present BCs are used. If any BC is defined, then all must be defined or redefined. The next section describes the boundary condition input for **MAP**

MAP uses a piecewise linear fit of the BH curve. This fit can occasionally lead to a very slow convergence of the nonlinear iteration. The use of the monitoring and relaxation options can be used to speed the convergence. The user specifies a node that the **MAP** will monitor during the iteration. If the change in the vector potential, ΔA, has the same sign for two iterations, the over-relaxation parameter is used. If the sign of ΔA is different from the previous iteration, the under-relaxation parameter is used. In this manner, **MAP** attempts to speed a monotonic behavior and damp an oscillatory one.

6.2 Initial Conditions

The default initial condition for the vector potential used in **MAP** is $A = 0$ everywhere. If a different initial condition is desired (possibly with different values in different materials), then the INITIALA command may be used. The form of the command is:

(*) **INITIALA,mat, $A_{initial}$**

where

INITIALA – invokes this option.

mat – is the number of the material that is to have the initial condition applied.

$A_{initial}$ – the value of the initial condition.

NOTE: The initial potential is set for all nodes of elements with material number *mat*. At interfaces between materials, the value of the initial condition will be taken as the last $A_{initial}$ specified. Therefore, if different initial conditions are used, there can be (non-physical) jump changes in A.

6.3 Boundary Conditions

The two types of boundary conditions for the problem are essential (or Dirichlet) and natural (or Neumann). A homogeneous (zero value) natural boundary condition is default (as discussed in Appendix A). This is the only allowed natural boundary condition in **MAP**. The natural boundary condition is that the normal derivative of A_θ or A_z is zero which is equivalent to specifying that the tangential component of the flux density is zero along the boundary - *i.e.,* the field is normal to the boundary. This is the usual symmetry condition.

The essential boundary conditions are imposed by specifying a value of the vector potential, A_θ or A_z. A constant value along a boundary is equivalent to specifying that the normal component of flux density is zero across the boundary - *i.e.,* no field lines cross the boundary.

The program now expects to read essential or Dirichlet boundary conditions for the problem. Since the problem is time dependent, the program allows the specification of values of A that can vary in a piece-wise linear manner with time.

The values of A at nodes may be specified on an element basis, or on an (i,j) nodal point basis. The two forms are:

(*) **A,nel1, nel2, nelinc, iside, A(t$_1$), {A(t$_2$), t$_1$, t$_2$}**

or

(*) **NA,i1, j1, i2, j2, A(t$_1$), {A(t$_2$), t$_1$, t$_2$}**

where

A – denotes an element boundary condition.

nel1, nel2, nelinc – is a looping feature that will apply the specified boundary condition to elements from *nel1* to *nel2* with the element number incremented by *nelinc.*

iside – side on which the boundary condition is applied (see Figure 5.10). Condition is applied to all nodes on the side.

$A(t_1), \{A(t_2)\}$ – values of the boundary condition at two times, t_1 and t_2 (which are usually, but not necessarily the start and stop of this time segment). The program will calculate the boundary condition at each time according to:

$$A(t) = \frac{(A(t_2)-A(t_1))}{(t_2-t_1)}(t-t_1) + A(t_1)$$

If $A(t_2) = t_1 = t_2 = 0$, the constant condition $A(t_1)$ is applied.

$\{t_1, t_2\}$ – two times for linear interpolation of $A(t)$.

NA – denotes a nodal point boundary condition, hence, the leading N.

$i1, j1, i2, j2$ – looping feature on the nodal point (i, j)'s with increments of one that is equivalent to the FORTRAN coding:

DO $I = I1, I2$
DO $J = J1, J2$

– **N.B. this looping does not allow a decrement in the loop. That is, if $i2 < i1$ or $j2 < j1$ the loop is not executed and no BCs are set.**

The element boundary condition imposes the value of A_θ or A_z at all nodes along side $\{iside\}$ (see Figure 5.10). The nodal point conditions are applied to the corner nodes. Any midside nodes on a side with two corner node boundary conditions is given an average value of the conditions at the corners by the program. The midside nodes are not given an (i, j) numbering.

The specification of boundary conditions is terminated by an END or END,BC line. Control is returned to calling routine (SETUP or FIELD2D).

This page lift blank intentionally

MAP – 6–6

Chapter 7.0 PLOTTING COMMANDS

For complex problems, reduction of printed output to a form that is easy to understand and to check becomes difficult. Graphics can help reduce the mass of printed data to a more easily understandable form.

MAP plotted output is generated by subroutines called from the driver routine MAPPLOT. The underlying plot package is GRAFLIB which is available on both the Plasma Fusion Center VAX system and on CRAYs of the NERSC system at Livermore.

There are four major types of plots available in **MAP**. They are: (1) element plots; (2) vector plots; (3) contour plots; and (4) function plots. In a transient problem, the last three plot types will generate a plot for each time step that was saved on the binary plot files (see the *pltfrq* parameter). There are options available to narrow down the time or times which are to be plotted.

Element and vector plots require only a single line of input. Contour and function plots require one or more additional input lines per plot as discussed below.

MAP can be used in either the file input or interactive mode. The program is presently set up to do the major computation and output of the printed solution in the file input mode. The graphical output can be generated in either mode.

Color graphics are supported by GRAFLIB and require a special switch that is selectable only in the interactive mode. Color is not available in the input file mode.

Plots may be generated at any point in execution as long as the data to be plotted have been generated. For example, element plots may be generated at any point in the process after the END, ELEMENTS line – *i.e.*, after return from SETUP. However, plots of the mesh with BCs shown cannot be made until after the BCs have been defined (to state the obvious).

7.1 The **PLOT** Command

The general form of the PLOT command is:

(*) **PLOT,TYPE,PLTTYPE,rmin,rmax,zmin,zmax,{itckr,itckz,nelov, tstart,tstop,scale,nelno,irot,TITLE}**

where

TYPE – Word identifying type of plot desired. Possible types are:
- ELEMENTS – element mesh with element numbers, material numbers or BCs.
- CONTOURS – contours of constant PLTTYPE.
- VECTORS – vectors of PLTTYPE.
- FUNCTION – function of PLTTYPE versus time or position. The independent variable is defined on the next line of input.
- TOLERANCE – Change tolerance for testing against time.

PLTTYPE – Word identifying which variable will be plotted. (see the individual TYPEs of plots).

rmin, *rmax* – r (x) limits defining the region for plotting.
zmin, *zmax* – z (y) limits defining the region for plotting.

{*itckr*, *itckz*}– no. of intermediate tick marks in the r(x) and z(y) directions < 4 >.

{*nelov*} – draw outline or elements by material number. Default is all.
- < 0 all materials except ||nelov|| outlined or drawn.
- = 0 all elements drawn or outlined (default).
- > 0 only material *nelov* outlined or drawn.
- = 99 all elements drawn and contours or vectors overlayed.

{*tstart*, *tstop*}– beginning and ending times. Default is all times in the plot file. If *tstart* = *tstop* = *t* then only a single time is plotted.

{*scale*} – scale factor for vectors < 0.08 >;
– height of element or material property numbers < 80 >;
– scale for contour labels < 1.0 >.

{*nelno*} – flag for element numbering.
- = −1 material numbers drawn at element centers.
- = 0 no numbers (default).
- = 1 element numbers drawn at element centers.
- = 2 vector potential BCs indicated at nodes.

{*irot*} – if = 1, rotate element and material numbers by 90 degrees.

{TITLE} – flag to allow input of plot titles. Primarily for function plots.

MAP does not force plots to be square – that is, *rmax* − *rmin* does not have to equal *zmax* − *zmin*. If square element, vector or contour plots are desired, it is up to the user

to choose the proper values for $rmin, rmax$ and $zmin, zmax$.

MAP can plot contours, vectors or functions for specific times or ranges of times. There is a built-in tolerance for testing the times written in the various plot files against the desired time. The default tolerance has a value of 1.e-3 seconds. If small time steps are used in the problem solution, the TOLERANCE option can be invoked to change this default. The form is: PLOT, TOLERANCE, value.

The parameters $\{itckr, itckz\}$ are included because GRAFLIB does not necessarily choose a nice minor tick interval. A major tick is one which is labeled. The minor tick marks are not labeled. Quite frequently GRAFLIB will choose minor ticks intervals of .08,.06,.04, *etc.*. These intervals may be forced to be whatever the user desires by the appropriate choice the the parameters $\{itckr, itckz\}$. Unfortunately, the user doesn't know *aprior* what minor intervals will be used and, hence, a second run of the graphics may be required.

When the FUNCTION option is requested, the inputs $rmin, rmax$ are interpreted as the physical coordinates – either r (x) or z (y) – for plots versus position, or as times $tmin, tmax$ for plots versus time. The $zmin, zmax$ inputs are the minimum and maximum for the function – *i.e.*,, the y-axis on the plots. If the user does not specify the minimum and maximum range for the function, the program computes them and scales accordingly.

The TITLE option is a way of adding a descriptive plot title for function plots. (Contour and element plots already put a title at the bottom of the plot). If title is invoked with file input, the next input line is assumed to be a title line. Because of the internal workings of the free-field reader, the title must be input in a comma separated set of a10 strings. (*E.g.*, FIELD ALON,G THE COIL) If the input is from the terminal, the commas are not necessary.

7.2 Element Plots

A plot of the element mesh may be obtained at any time after SETUP. The PLOT,ELEMENTS command is used. The parameter {*nelovl*} refers to all the elements of the particular material type – *e.g.*, an *nelovl* = 1 will cause all the elements with material number 1 (within the r,z (x,y) limits) to be plotted. If *nelovl* = 0 (the default) then all elements are drawn.

The plot may be made with the element number or the material property number printed at the element centroid, depending on the value of the parameter *nelno*. For element plots, the parameter *scale* can be used to adjust the size of the numbers. The default value is 80.0, a smaller number implies a larger print. For example, the **MITMAP** advertisement has a character size of 55. The final parameter *irot* = 1 rotates the numbers 90 degrees.

In order to plot the boundary conditions, the boundary conditions must have been defined. This type of plot could be requested after the solution phase. Alternatively, **MAP** does have a provision to define the boundary conditions in SETUP after the element definitions. Of course, if the BCs change from ramp to ramp, the last set of BCs are plotted. The boundary condition codes are plotted at the nodes. The codes for the boundary conditions are: A or TA; where A is the vector potential and TA implies a time dependent A.

Examples

To plot all elements within the region (r,z) = (0,2) to (7,9), with material numbers, the input would be:

PLOT,ELEMENTS,0,7,2,9,,,,,,-1

To plot all elements of material number 2, within the region (r,z) = (0,2) to (7,9), the input would be:

PLOT,ELEMENTS,0,7,2,9,,,2

To plot all elements except those with material type 3. Each axis would have three minor tick marks for every major ticks mark. The input would be:

PLOT,ELEMENTS,0,7,2,9,3,3,-3

7.3 Vector Plots

There are three types of vector plots available. FOR82, FOR84, and FOR86 contain the information needed to plot the force, flux density, and iron magnetization vectors, respectively. This information is available after FIELD2D or TRNSPRNT. The parameter {*scale*} on the PLOT line determines the length of the vector and the size of the arrow head.

The PLTTYPEs (third entry on the PLOT command line) are:

FIELD, FORCES, or MAGNETIZATION.

The force and magnetization vectors are centroidal or average values within each element. The tail of the vector is placed at the element centroid. Flux density vectors are nodal point values and the tail of the vectors start at each corner node. In order to have the arrowheads drawn proportional to the length of the arrow an additional entry of the letter P, is required at the end of the PLOT command line.

Once the proportional arrowhead option has been invoked, it is in force for the remainder of the session. Only a re-execution will reset this option.

Examples

To get a vector plot of the forces (with heads proportional to the shank) at time $t = 2.5$ seconds in a region (r,z) = (0,0) to (5,5) with 5 intermediate tick marks on each axis, the input would be:

PLOT,VECTORS,FORCES,0.0,5.0,0.0,5.0,5,5, ,2.5,2.5,0.2,P

The length of the force vectors would be scaled by 0.2.

7.4 Contour Plots

There are six different types of contour plots that can be generated by **MAP**.

The available plttypes are:

FLUX (A_z), B_r (B_x), B_z (B_y), B, J_θ (J_z), or HOMOGENEITY.

HOMOGENEITY refers to field homogeneity in percent and is defined as:

$(B_z(r,z)/B_z(r_o,z_o) - 1.0) * 100$

where **MAP** prompts for the point (i_o, j_o) that defines (r_o, z_o) as described below. The other plttypes are self-explanatory.

Contours of constant A_z or FLUX may be plotted after FIELD2D, or RESTART (if FOR083 and FOR085 are available). The other contour plots cannot be obtained unless FOR084 is also available, as it is after FIELD2D or TRNSPRNT. Therefore, if FOR084 is not available, but FOR085 is, then a RESTART followed by one or more TRNSPRNT commands can reconstruct the necessary data.

In addition to the PLOT command, the program requires one additional line of data (two additional lines for homogeneity plots). The line next defines the values of the contours to be plotted and the manner in which the contours are to be labeled. The form of the input line for specifying contours is:

(*) **ncontrs,{min, delta, iplt, lblfrq, ilbl, htd, ndecNONREG,PRINT,NODASH}**

where

ncontrs – number of contours (≤ 100). Must be specified. In interactive mode, = 0 will skip the present time step and process the next step. If < 0, exit contour routine and return to command level. The latter two options are useful in time dependent problems.

{*min*} – value of the first contour to be plotted.

{*delta*} – contour increment. If = 0 **MAP** finds the minimum and maximum values in the region of interest and calculates the delta as $(max - min)/ncontrs$

{*iplt*} – index that determines whether contours are to be labeled and, if so, to determine the distance between labels on a given contour as a percentage of the plot width *e.g.*, {*iplt*} = 10 implies that a label will be placed about every 10% of the plot width.
= 0 no labels.
> 0 every *lblfrq* contour labeled with the value.

{*lblfrq*} – frequency of contour labels.

{*ilbl*} – type of contour labeling.
= 0 values of contours plotted.
= 1 contour numbers are plotted.

{*htd*} – height of numbers used to label contours. < 80 >

{*ndec*} – number of decimal places to be plotted (-1 implies integer). < 2 >

{NONREG} – special flag for nonregular meshes described below.

{PRINT} – special flag to write the contouring data into a special file FOR031.DAT. Output consists of an unsorted contour number and the beginning and end coordinates of a line segment making up the contour.

{NODASH} – special flag to disable the plotting of positive and negative contour values with different styles of lines. Negative contour values are plotted as a dashed line for regular ordering and a thick line for nonregular ordering. This option overrides the default and plots all contours with the same style line. If NONREG is used, then either PRINT or NODASH (but not both) may be used.

MAP computes the minimum and maximum values of the function to be contoured within the region of the plot for each time step written to the plot file (according to the parameter *pltfrq* in the FIELD2D command. These values are printed before the data discussed above are required. In the interactive (terminal input) mode, the user is able to see the minimum and maximum values before specifying the number and interval between contours. That is, the program prints the minimum and maximum values found at each time and prompts for the above input.

In the noninteractive (file input) mode, this is not possible. The program basically uses the above input across all times. If the minimum and delta values are not specified, the program computed values are used and these may not be the same values across times. In order to have the same contour values across times, the minimum and delta values must be specified.

Since the free-field reader sets nonspecified data to zero, if $\{min\}$ is unspecified, the minimum value of the function found by the program is used as the first contour value. If the user wishes to start at any other value, then $\{min\}$ should be specified on the data line. If, however, the user wishes to start at a contour value of zero and the program computed $\{min\}$ is not zero, then a value of 1.e-9 must be input to override the default.

For contour plots, the parameter $\{scale\}$ on the PLOT line takes on a different meaning. It is a scale factor for the contour labeling with default of 1. The values of the function being contoured are multiplied by this scale factor for the purpose of labeling the contours. For example, if a B contour plot is requested, and if the user requires the labeling to be in gauss instead of tesla then $\{scale\}$=1.e-4 would produce the desired labels.

GRAFLIB has a built-in contouring routine for nice rectilinear meshes. This routine will not work for meshes containing triangles. There are also problems with very crude rectilinear meshes with large holes in them. Although there are general mesh contouring algorithms, they do not work well with certain kinds of mesh geometries. **MAP** has a cruder and more expensive contouring algorithm built-in. This routine is invoked via the NONREG option on the contour definition line. **NOTE:** Contour labeling is disabled for the NONREG option.

The PRINT option can only be invoked in conjunction with the NONREG option.

If HOMOGENEITY is chosen, an additional line of input is required to specify the point of the normalizing field. The input is

(*) $\underline{\{i_o, j_o\}}$

where (i_o, j_o) defines the node at which the normalizing field is computed. Default node is assumed to be (1,1).

Examples

If twenty contours of constant FLUX are required for the region $(r, z) = (0, 0)$ to $(7, 7)$, the input would be:

PLOT,CONTOURS,FLUX,0,7,0,7,7
20

with seven tick marks between numbers on each axis. The routine would determine the contour spacing. There would be no contour labels.

If contours of constant field are desired in the region $(r, z) = (0, 0)$ to $(3, 3)$ m., with 20 contours starting at 1.0 T and increments of 0.1 T, and if the mesh is irregular, the input would be:

PLOT,CONTOURS,B,0,3,0,3
20, 1.0, 0.1, NONREG

If twenty contours of B_z are desired starting at -1 T with increments of 0.1 T and with every other contour labeled with a one decimal place label and with labels placed approximately 20% of the plot-width apart, the input would be:

PLOT,CONTOURS,B,0,3,0,3
20, -1.0, 0.1, 20, , ,1

7.5 Function Plots

MAP has extensive capabilities for generating function plots. The system quantities such as energy stored, power, total dissipative losses, or material or total current, may be plotted versus time. Local quantities such as vector potential or flux, flux density components, or current density at a point may be plotted as a function of time. Finally, the local quantities may be plotted as a function of position at various times.

All function plots are initiated with the same PLOT,FUNCTION command described above. As mentioned previously, the $rmin, rmax, zmin$, and $zmax$ inputs are interpreted as $xmin, xmax, fmin$, and $fmax$ (for a plot of f(x) versus x). If the user desires, the program can compute the minimum and maximum values of the function to be plotted – *i.e.*, auto-scale. The auto-scale is invoked by entries of 0,0 for $fmin, fmax$.

For a function plot, a background grid is drawn with a dashed line horizontally and vertically at each major (labeled) tick mark. This grid can be used to aid in reading data from the plots. However, if the user does not desire the background grid, and entry of $\{nelov\} = 1$ on the PLOT, FUNCTION line will disable the background grid.

The $\{scale\}$ parameter on the PLOT, FUNCTION line may be used to scale the data being plotted. For example, if a plot of flux density versus time is desired with the flux density in gauss instead of tesla, then an entry of $\{scale\} = 1.e-4$ would scale the data.

An additional input is required to define the independent variable (x-axis). The form of the command depends on the type of function plot desired.

7.5.1 System Totals Versus Time

The available PLTTYPEs for plotting system totals versus time are:

STORENERGY, TOTNI, TOTLOSS, POWER,

or

MATNI, MATFR (MATFX) or MATFZ (MATFY)

where

STORENERGY – the system magnetic stored energy.
TOTNI – the total system ampere-turns.
TOTLOSS – the system energy dissipation.
POWER – the system power.
MATNI – the material ampere-turns (summed over all elements of each material)
MATFR – the material radial force (summed over all elements of each material)
MATFZ – the material vertical force (summed over all elements of each material)

See Appendix A for a description of how these terms are calculated. In order to plot one of the system totals versus time, the second input in the function plot sequence has the form:

(*) **T,{mat}**

where

T – sets the independent variable to be time.

{*mat*} – sets the material number for plots of MATNI, MATFR or MATFZ. Unused for other PLTTYPEs. Multiple material numbers may be entered by setting *mat* = −1; **MAP** then prompts for material numbers until an END or 0 is entered.

Examples

To plot the total ampere-turns in the problem versus time between 0 and 10 seconds with auto-scaling for the current, the input would be:

PLOT,FUNCTION,TOTNI,0,10
T

To plot material vertical force for materials 1, 2, and 4, the entries would be:

```
PLOT,FUNCTION,MATFZ,0,10
T,-1
1
2
4
END
```

If only material 2 force was desired, the entry **T,2** would be sufficient.

7.5.2 Local Quantities Versus Time

In addition to plotting system totals versus time, **MAP** can plot local variables versus time. The local variables may be flux, flux density components, or current density at a point (or points). The auto-scaling, background gridding, and data scaling described in the previous sections also hold for this type of function plot.

The plttypes are:

FLUX (AZ), BR (BX), BZ (BY), B, or JTH (JZ)

The second line of input required is:

(*) **T, i_o, j_o, {PRINT}**

where

i_o, j_o – the (i, j) pair of the (spatial) point at which the function is to be plotted versus time. If $i_o = -1$, the program expects a list of (i, j) pairs on successive lines (terminated with an END). The value of the function at each (i, j) pair is then plotted versus time.

{PRINT}– special flag to print the data to the terminal (or log file) as well as plot it.

Examples

To plot the x-component of flux density at the node with $(i, j) = (5, 3)$ versus time between 0 and 10 seconds, with auto-scaling and no background grid, the set of commands would be:

```
PLOT,FUNCTION,BX,0,10,0,0,,,1
T,5,3
```

To plot the magnitude of the current density versus time (0 to 10 seconds) at three points – say $(i, j) = (1, 1), (4, 5)$, and $(10, 3)$, the set of entries would be:

```
PLOT,FUNCTION,J,0,10,0,0
T,-1
1,1
4,5
10,3
END
```

7.5.3 Local Quantities Versus Position

In addition to plots of local quantities at point versus time, **MAP** has the capability of plotting these functions versus position for various times.

The available plttypes are:

FLUX (AZ), BR (BX), BZ (BY), B, or JTH (JZ)

Plots of these functions may be along lines of constant I or J – Note: not necessarily constant r or z. In addition, the plots may be forced to be along a constant r or constant z line by invoking the NONREG parameter on the command line as described below.

There are two distinct forms of the second command line in the set. The first form is for plotting along constant I or J lines in a region with a rectilinear mesh. The second form must be used with a nonrectilinear mesh, and it may be used with a rectilinear mesh to plot along a constant r or z line that does not coincide with a constant I or J line. The two forms are:

(*) **R(X),j1,j2,jinc,{ ,PRINT,MORE}**
(*) **Z(Y),i1,i2,iinc,{ ,PRINT,MORE}**

or

(*) **R(X),zo(yo), NONREG{,PRINT,MORE}**
(*) **Z(Y),ro(xo), NONREG{,PRINT,MORE}**

where

R(X) –
Z(Y) – The independent variable name.

$j1, j2, jinc$ –
$i1, i2, iinc$ – starting, ending, and increment value of the independent variable in (i, j) space. May be used to define multiple curves.

$zo(yo)$ –
$ro(xo)$ – fixed value of the coordinate held constant for the plot. This option must be used with the NONREG parameter.

NONREG – special flag for nonrectilinear grids and for plotting a function along a constant r or z line which is not a constant I or J line.

{PRINT} – special flag to print the data to the terminal (or log file) as well as plot it.

{MORE} – special flag to allow specification of times at which function should be plotted. When invoked, the program expects a sequence of times to be specified (one per line) until an END line is encountered.

If there is an interface between two regions with different permeabilities, the tangential component of the flux density will be discontinuous. **MAP** calculates the flux density components at points within each element and extrapolates the values to the corner nodes. It then averages the values from all the elements at each node. If the node is on an interface between regions with different permeabilities then the averaging is done separately on each side of the interface. Therefore, there will be two or more values of the components of flux density at such nodes.

The plotting algorithm recognizes the existence of the interface. Away from the interface, the function plot is continuous; at the interface, however, the two values of flux density or flux density component are plotted with a small open square. For example, if a single curve is requested that crosses an interface, a continuous line will be plotted up to the last node before the interface, then two squares are plotted and then the line will continue at the first node past the interface. If the plot is along an interface, then two sets of squares will appear all along the interface.

Examples

To plot B_z versus z, ($0 \leq z \leq 5$ and $0 \leq B_z \leq 3.0$) for $r = 0$ (assumed to be $I = 1$) and all times saved, the input would be:

PLOT,FUNCTION,BZ,0,5,0,3.0
Z,1,1,1

That is: plot B_z versus z for $I = 1$ to 1 by 1. If three curves for $I = 1, 3, 5$ (corresponding to, say, r=0.0,0.15,0.3) are desired, then the second line of input would be:

Z,1,5,2

In order to plot the flux versus r for $0.1 \leq r \leq 0.3$ along a line $z = 0.2$ for all times with auto-scaling, the input would be:

PLOT,FUNCTION,FLUX,0.1,0.3,0,0
R,0.2,NONREG

NOTE the use of the NONREG parameter.

To plot the flux versus r for $0.1 \leq r \leq 0.3$ for $j = 1$ at time $t = 5$ seconds with auto-scaling, the command would be:

PLOT,FUNCTION,FLUX,0.1,0.3,0,0,,,,5,5
R,1,1,1

If all times between 5 and 10 seconds (inclusive) were desired, then the second 5 on the PLOT line would be changed to a 10.

If a run comprises times from 0 to 25 seconds with data saved every 0.5 seconds, a plot versus position for all times might contain too many lines. The MORE option can be used to select only certain times. In the previous example, if times at 5, 10, 15, 20, and 25 seconds were desired the input would be:

PLOT,FUNCTION,FLUX,0.1,0.3,0,0
R,1,1,1,MORE
5.0
10.0
15.0
20.0
25.0
END

Chapter 8.0 MISCELLANEOUS COMMANDS

8.1 RESTART Command

The restart option is a powerful feature that allows the user a great deal of flexibility in the generation of element meshes, plotting, and the solution of transient problems. The RESTART command allows the program to be entered at any one of the primary subroutine levels.

The form of the command is:

(*) **RESTART, COMMAND**

where

COMMAND – is the subroutine name or major program module to be entered. Choices are:

FIELD	– Continue with a magnetic field problem from the last time stored on disk.
PLOT	– Generate one or more plots from the data stored on disk.
POST	– Postprocess the data stored on disk.
TRNSPRNT	– Enter the **MAP** postprocessor for one or more times written on disk.
MORE	– A slightly different way of restarting a transient problem. Under the MORE option, the new time step solutions are appended to the old. A RESTART, FIELD2D will start a new problem using the previous data on disk as the initial conditions, but it will not save the previous data.

The COMMAND must be specified. The entry identifies for **MAP** which disk files are to be read. A RESTART with one COMMAND followed by a different COMMAND may lead to spurious results. There is an exception to this rule. A RESTART, TRNSPRNT may be followed by a set of boundary conditions on the field to allow the postprocessor to correctly capture the total NI and stored energy.

Of course, in order to restart, certain data must be available. The data necessary for restarting are written to the (binary) disk files given in Table 8.1. The information stored in the files is given in Appendix B. An X implies that the disk file must be available. A Y indicates that the need for a particular file depends on what output is desired. For example, if element plots are desired, then only FOR083 is required. If flux contour or function plots are desired, then only FOR083 and FOR085 are needed.

RESTART reads the data stored on FOR083. This data includes the title, units, plane or axisymmetric flag, problem flag, and the data from the initial problem command. It also contains the required grid data, including material properties, coordinates, element

connectivities, and equation numbers. **Therefore FOR083 must be available for any RESTART.**

Of course, FOR083 may be generated directly from a data input file by running through the SETUP phase. At that point, a RESTART command may be issued assuming the other required files are available.

Table 8.1 Required Disk Files for RESTARTing **MAP**

	FIELD	PLOT	TRNSPRNT	POST	MORE
FOR083	X	X	X	X	X
FOR084		Y		Y	Y
FOR085	X	Y	X	X	X
FOR086		Y		Y	Y
FOR087		Y		Y	Y
FOR089					X

8.2 TRNSPRNT Command

The TRNSPRNT command is included so that the user need not make an *a priori* choice of frequency of printed and plotted output by specifying the *prntfrq*,*pltfrq*, and *prntlvl* parameters on the FIELD2D or THERMAL commands. Without this command, if proper choices were not made then the entire run would have to be repeated. Similarly, a too conservative approach of specifying a very frequent output could generate a huge amount of printed and plotted data. For example, if Lorentz forces are desired at only a few time points out of many, a full run with a prntlvl=2 would generate the force output at each time step and result in a huge amount of printed output. Instead, TRNSPRNT can be used to pick certain time points after a run and postprocess only the desired points.

A single file, FOR085, is used to store the field solution at each time step – regardless of the frequency or level parameters of the FIELD command. TRNSPRNT can be called after a RESTART, TRNSPRNT - (Note: FOR083 and FOR085 must be available). FOR085 is read until the time of the write and the requested print time are equal. The file is backspaced and the solution at the previous time is also read. Then TRNSPRNT calls the print routine with the specified print and plot flags, and printed and plotted output can be obtained.

A large number of time steps may be run with a very low level of requested output – *e.g.*, $prntlvl = 3$, *prntfrq* and *pltfrq* at large intervals. The output can be examined to determine intermediate times of interest, and TRNSPRNT used to get more detailed output at these times. For example, an element by element print of the in-plane forces at a single time can take 1000 lines of output. If the detailed forces are required only at a few times, it does not make sense to print all the element forces at all the times. Instead, a $prntlvl = 3$ could be used to determine the time(s) at which the material totals are a maximum. Then, a RESTART, TRNSPRNT could be used to postprocess the information on FOR085 at the times of interest.

The form of the command is:

(*) **TRNSPRNT,time, {prntlvl, ipltlvl, prntfrq, pltfrq, irewind, ALL}**

where

time – time at which output is desired.

{*prntlvl*} – printed output level indicator. See FIELD2D (6.1).

{*ipltlvl*} – last digit of unit numbers of the binary files of plot data written every $pltfrq$ time steps. See FIELD2D (Sec. 6.1)

{*prntfrq*} – frequency of printed output - print at every {$prntfrq$} time steps. < 1 >

{*pltfrq*} – frequency of plotted output. MUST be specified if plots are desired. < 1 >

{*irewind*} – special flag to rewind plot files before postprocessing. If = 1 then all files are rewound. Basically, a holdover from previous versions. It can be used to minimize disk storage if printed output is desired, and plotted output is not needed.

ALL – Special flag to invoke continuous processing of all data on disk. This option would basically reconstruct the printed output and plot disk files from a run.

The entries $prntlvl, ipltlvl, prntfrq$, and $pltfrq$ are identical flags to those for the FIELD2D command.

Since two solutions are needed to calculate the time rate of change of the energy – *i.e.*, magnetic power, the use of TRNSPRNT for the first time step saved on FOR085 may lead to spurious results for this term (unless the initial condition is B = 0). TRNSPRNT sets the $t - \Delta t$ solution to zero if the first step is requested. The values of the field and current density components, however, will be correct. In this case, **MAP** prints a warning message, but proceeds with the printing.

FOR085 will be overwritten by a TRNSPRNT command if *ipltlvl* is specified with a digit 5. The default for TRNSPRNT excludes this file from the write. If the file is overwritten, other times cannot be recovered. For peace of mind, it is a good idea to rename or otherwise save a copy of FOR085 before running TRNSPRNT.

Example

If a transient analysis has been run, and the solutions at time .01, .02, ... 10. have been saved on FOR85, a RESTART,TRNSPRNT with the following sequence of commands would generate plot files and in-plane Lorentz forces on each element for times 1., 5., and 10.0:

TRNSPRNT,1,1,0,1,1
TRNSPRNT,5,1,0,1,1
TRNSPRNT,10,1,0,1,1

The $prntlvl = 1$ parameter implies print in-plane Lorentz forces. The $ipltlvl = 0$ parameter will cause disk files FOR082, FOR084 and FOR087 to be written. The last two parameters $prntfrq$ and $pltfrq$ are set to their default values.

8.3 POST command

In addition to the postprocessing of the field data that is automatically performed in the FIELD2D, **MAP** has two options for further postprocessing. They are invoked using the POST command. The POST command can invoke one of the built-in postprocessing functions defined below.

The form of the POST command is:

(*) **POST, COMMAND**

where COMMAND is one of the possible postprocessing commands defined below.

ADDFORCE– A routine to sum in-plane forces for multiple elements.

FINDEQN – A routine to find (1) element numbers associated with (i, j) pairs; or (2) the equation number associated with a physical (r, z) or logical (i, j) coordinate pair.

Each of theses routines requires additional input as defined below.

8.3.1 ADDFORCE Command

The ADDFORCE option under the POST command will allow the user to sum the in-plane Lorentz forces for a user-defined set of elements. Files FOR083 and FOR082 must be available. FOR082 contains the element forces and is written according to the *pltfrq* parameter on the FIELD2D or TRNSPRNT commands.

For axisymmetric problems, the forces are total forces per unit (circumferential) length acting on each element. In planar problems, the forces are per unit length. The ADDFORCE routine expects additional input defining the elements over which the force components are to be summed. The input can be of two forms:

$I1, J1, I2, J2$
$\vdots$
END

or

SUM,$N1, N2, \ldots$
$\vdots$
END

where the entries $I1, J1, I2, J2$ refer to the looping limits of the (i,j) pairs of the first corner of each element to be summed; *e.g.*, an entry of $2, 3, 10, 11$ would sum the element forces for the elements with the first corner (logical) coordinates of $(2, 10), (2, 11), (3, 10)$, and $(3, 11)$. Multiple $I1, J1, I2, J2$ sets may be defined in order to get multiple summed forces.

In addition, the SUM option allows the summation of forces across specified element numbers $N1, N2, \ldots$. Multiple SUM commands may be used. The element numbering for a particular mesh may be found by (1) a print level option of 1 on the SETUP command; (2) an element plot with element numbers; (3) a print level of 1 on the FIELD2D command – which will print the element number and (i,j) pair of the first node along with the in-plane force components; or (4) use the FINDEQN option discussed below.

The $I1, J1, I2, J2$ and SUM options may be mixed. There is only one END command. In a transient run, forces for all times stored on FOR082 will be processed.

EXAMPLES

Assume a coil is defined by elements with (i,j)'s running from $1,1$ to $11,9$. Also assume the force on elements 131, 145, 159, and 163 is desired. To sum the forces over the coil

and this other region, the command set would be (after a restart, or at the end of a time marching run):

1,1,11,9
SUM,131,145,159,173
END

There would be two lines of output for each time stored on FOR082, one, for the coil and one for the additional sum.

8.3.2 FINDEQN COMMAND

The FINDEQN command can be used to find either the element number associated with an (r,z) or (i,j) coordinate pair, or an equation number associated with a coordinate pair. Conversely, the command can be used to determine the first corner (i,j) pair associated with an element number or an equation number. This command may be used to determine element numbers for use in the ADDFORCE command described above. Alternatively, it can be used to determine the (i,j) pair associated with an equation number for monitor acceleration of the nonlinear iteration.

FINDEQN expects additional input. The first input (prompted for an interactive mode) is the type of find to be done. The options are:

ELEMENT – Determine the element numbers corresponding to a set of (i,j) pairs. The (i,j) pair is assumed to be the first corner node. The program expects input of: $I1, J1, I2, J2$ or END where $I1, J1, I2, J2$ are the usual looping limits. After a set of (i,j)'s the program loops back for another set of (i,j)'s. An END terminates the routine and returns control to MAP.

EQUATION – Given an equation number, find the (i,j) pair associated with it.

COORDINATE– Given an (i,j) pair or (r,z) pair, find the (r,z) or (i,j) coordinates, the element number with this set as its first node, and the equation number. The input is: RZ,r,z or IJ,i,j or END.

END – Terminates routine and returns control to MAP.

8.4 TERMINAL Command

The TERMINAL command allows the user to switch the input from file to terminal during a run. The switch is only one way. The command may be used with a complicated mesh defined in a file. If the command following the element generation is:

(*) **TERMINAL**

then the file is closed and additional input is expected from the terminal. The standard terminal input prompt of COMMAND is issued.

8.5 END Command

In the command mode, if an END line is encountered, the program assumes another problem is to be solved. A new title line must follow immediately. In addition all flags such as plane/axisymmetric, units, and problem are reset to the default values. The form of the command is:

(*) **END**

Like other end lines, anything following is ignored, so that END,PROBLEM is acceptable.

8.6 STOP Command

In the command mode, a STOP line causes **MAP** to terminate execution. All attached disk files are detached and control returns to the system command level. The form of the command is:

(*) **STOP**

Chapter 9.0 PROGRAM OUTPUT

9.1 MAP Output

The first lines of an output file contain the **MITMAP** version number, and a description/history of recent changes. The additional output generated in the main program MITMAP consists of a line-by-line echo print of the input data file. The example of input decks shown in the next chapter are reproductions of this print. The CPU time taken in each of the major subroutines called by MITMAP is also printed. In addition, certain data such as the PROBLEM definitions are printed in a formatted form.

Each page of output is headed by a banner that includes **MITMAP** version number, current date and time as well as the run title input via the TITLE line. For transient problems, the printed output also includes the time at which the print takes place.

Examples of output from the various sections of the program are given in Chapter 11.

9.2 SETUP Output

The output generated in SETUP consists of three distinct parts. They are: material properties including a separate listing of the BH-curve (if any of the materials have an iron fraction between 0 and 1); grid point generation; element generation; and equation numbering data.

The amount of grid and element output is determined by the *iprint* option on the SETUP command line. The material properties, the number of grid parts defined, and the number of elements generated are printed for all values of *iprint*.

Additional output is generated according to:

iprint = 1 no additional output.

= 2 data for each grid part and for each element two integers are printed, the first is $1000 * j1 + i1$ where $(i1, j1)$ is the (i, j) pair of the first node, and the second is $\{100 * \{element\ type\} + material\ number\}$.

= 3 same as 2, but also the $1000 * j + i$ for each node of each element is printed.

= 4 the grid data, plus coordinates of each point generated in each grid part. Each element has its type, material number, the (i, j) pairs and coordinates of each corner printed.

= 5 same element print as *iprint* = 4 but without any grid point output.

An *iprint* = 4 or 5 will generate many pages of output and should only be used during the initial stages of checking out a very complicated new mesh or for transmitting detailed mesh data between programs. Once the mesh has been checked out an *iprint* = 1 is suggested.

9.3 FIELD2D Output

Output generated by the FIELD2D subroutine consists of an informative echo print of the various parameters on the FIELD2D command line, and timing for the various secondary subroutines called. For nonlinear problems, the output includes the iterative convergence information such as the maximum change in the potential and several different convergence norms. The equation number at which the maximum delta was found is also printed – the equation number and FINDEQN in POST can be used to find the nodal point (i,j) and then used in a subsequent run with the adaptive iterative acceleration option.

The postprocessing routine is called automatically at the end of a static run. This routine is also called in transient problems according to the parameters *prntfrq* and *pltfrq* or at the final time. This subroutine not only calculates and prints the various values of field, current, *etc,* but also writes the necessary files for plot postprocessing. The amount of printed output is determined by the *prntlvl* parameter on the FIELD2D line. The frequency of printed output (in transient analyses) is determined by the *prntfrq* parameter. This parameter also determines the type of printed output, as described below. The frequency at which the plot files are written is determined by the parameter *pltfrq*.

A *prntlvl* of 4 produces no printed output. However, the plot files are written with the appropriate data, if *pltfrq* has the required value.

If *prntlvl* is 0, 1, 2 the routine will print:

- an element by element output of the Lorentz force components. These forces are on a per unit length basis. The element number, the (i,j) pair of the first corner node, the material number, the force components F_r, F_z and $|F|$ are printed at each time requested by the *prntfrq* parameter. This output can be more than 1000 lines per time step. If only a small number of times is required, then the TRNSPRNT command should be used.
- the total Lorentz force components acting on each different material. These forces are on a per radian basis. Also calculated and printed are the coordinates of the force centroid, *i.e.*, the coordinates at which the total force acts to give the same moment about the appropriate axis.
- system totals such as ampere-turns, stored, dissipated, and total energy, and power.
- a table with the area, ampere-turns, and the maximum and minimum flux density, current density, and field intensity for each material.

If *prntlvl* is 0, then following the total force components by material, the nodal values of $i, j, r, z, Flux, J_\theta, B_r, B_z, |B|, H$, and *dir* are printed. *dir* is the direction of the flux density vector in degrees (measured counterclockwise from the x- or r-axis. This value of *prntlvl* can generate a great deal of output.

A *prntlvl* of 3 will print the total force, system totals, and summary material table for each time.

A *prntlvl* of 5 was added to reduce the amount of required output. It is a special option to allow the user to specify a range of nodal point (i, j)'s for which the field, current density, *etc.* is to be printed – *i.e.*, *prntlvl* = 0. See the FIELD2D command for an explanation.

The values of the flux density components that are printed are the extrapolated, averaged, nodal point values [9]. That is, the gradients are calculated at the 2 x 2 Gauss-Legendre integration points in each quadrilateral (and at the vertices of each triangle). These values are extrapolated to the corner nodes and averaged. If the corner lies along a material interface between materials with different permeabilities, then the averaging is done only over elements of like material. Multiple lines of output are generated for each interface node. An asterisk, (*), is used to denote the second side of the interface. If required, an exclamation point (!) is used to denote a third side of an interface.

MAP has the option of printing the extrapolated, but unaveraged, values of the flux density at the corner points of each element. This option is available for *prntfrq* negative. The output consists of the element number, element type, the material number, the coordinates of each nodal point and the values of $Flux, J_\theta, B_r, B_z, |B|, |H|$ and angle for each node. This option can generate a great deal of output, and is not generally recommended.

Appendix A gives a list of the formulae used in the calculation of the various output quantities.

9.4 Other Output

The other subroutines in **MAP** generate a limited amount of output. The output consists mainly of timing and echo prints of input. The choice of UNITS and PLANE options are echo printed. The TRNSPRNT command will generate the same output as described above. The POST routines generate output as described in section 8.3.

MAPPLOT output consists of the minimum and maximum values of the contours, vectors, and function within the limits of the plot. A PRINT option for the function plot will print the data to the terminal or log file. A PRINT option for a contour plot will cause the unsorted data to be printed to a disk file, FOR031. Additional software (CONTOSTK) is used to sort the data and output a sorted set of contour data.

Finally, all routines have error checks and error handling. An error, when detected, generates an informative print, as defined in the next chapter.

Chapter 10.0 DIAGNOSTIC AND ERROR MESSAGES

MITMAP has a large number of internal error checks, but Murphy's 3rd Law states that it won't trap your error. Errors are classified as fatal and nonfatal. If an error is encountered, a diagnostic and realted message appears.

In a batch environment, a fatal error causes a message to appear in the log file and at the end of the output file, then execution is terminated with a call to STOP 1. A nonfatal error will cause a message to appear in both the log file (or at the terminal) and in the output file. Execution will continue.

In the interactive environment certain errors, such as unrecognizable commands, cause a diagnostic to appear, but execution does not terminate. After the diagnostic, the program will print a new prompt for the input.

In either environment, the error message is printed as a row of asterisks, an error number, the subroutine where the error occurred, an explanation and then, depending on the error, an additional word and two integers to help in the debug. The general form of an error message is:

```
**********************************************************************
Error Number iiii encountered in subroutine "subroutine"
Explanation "word" jjjj kkkk
**********************************************************************
```

where

iiii – is an integer error code defined in the next section.

subroutine – subroutine name in which error was encountered.

explanation – an abbreviated text string defining the error.

word – optional additional word of explanation – *e.g.*,an incorrectly spelled or unrecognized command will invoke the error routine. The entry "word" will be the unrecognized command.

jjjj,kkkk – optional two integers to help define the error.

The error messages incorporated in **MAP** and their meanings are listed in Table 10.1 along with the subroutines in which they occur. The following notation is adopted for the

error type. If the error type is printed in lower case, the actual computer output would have the alphanumeric variable printed. If the error type is in upper case, the message is as it appears below. The entry B/I refers to batch and interactive mode. The notation F and N imply fatal and nonfatal. A blank implies it's not really applicable.

Error numbers are grouped by major module or routine. Since **MITMAP** contains both **MAP** and **MAP2DJ**, all errors are included.

B I	Error #	Routine Name	Explanation	word *jjjj,kkkk*	Meaning
F N	0001	MITMAP	INCORRECT COMMAND	cmmnd	Misspelled command, cmmnd?
F	0002	FFLD	FREE-FIELD ERROR		Error in input file. Misplaced or omitted period or comma.
F N	0003	RESTART	INCORRECT RESTART COMMAND	cmmnd	Misspelled command, cmmnd?
F N	0004	TRNSPRNT	WRONG TIME		Requested time not found on FOR085
F	0005	READ3	END OF FILE ON FOR083		Mesh file missing?
F	0006	READ4	END OF FILE ON FOR084		Current density file missing?

B I	Error #	Routine Name	Explanation	word *jjjj,kkkk*	Meaning
F	1001	MATREAD	MAXIMUM NUMBER OF MATERIALS EXCEEDED	*mat,maxmat*	Material number *mat* exceeds dimension, *maxmat*
F	1002	GRID	MAXIMUM NUMBER OF PARTS EXCEEDED	word *part,maxprt*	Part number *part* exceeds dimension, *maxprt*
F	1003	GRID	MAXI EXCEEDED	*part,i*	Part number *part* has maximum I, i, which exceeds dimension.
F	1004	GRID	MAXJ EXCEEDED	*part,j*	Part number *part* has maximum J, j, which exceeds dimension.
F	1005	GRID	UNDEFINED NODES DURING FILL		FILL was used with corner nodes not yet defined.
F	1006	ELMDEF	INVALID ELEMENT COMMAND	cmmnd	Incorrect element or loop command, cmmnd.
F	1007	ELMDEF	TWO LOOPS OF SAME TYPE	cmmnd *ij,inc*	Two loops of the same type. Missing IEND or JEND?
F	1008	ELMDEF	NO MATCH FOR MAT COMMAND	cmmnd $i1, j1$	MAT command, cmmnd, with node $(i1, j1)$ did not match any elements.
F	1009	ELMDEF	MAXIMUM NUMBER OF ELEMENTS EXCEEDED	cmmnd *nel,maxel*	Element,*nel*, exceeds dimension, *maxel*.
F	1010	ELMDEF	ELEMENTS WITH UNDEFINED NODES		Element nodes defined without coordinates.
F	1011	EQN	PATHOLOGICAL ORDER	n, m	Bizarre (I, J) ordering. element, n, side, m.
F	1012	EQN	TOO MANY UNKNOWNS	n, *maxeqn*	Maximum unknowns exceeds dimension, *maxeqn*.
F	1013	EQN	CAN'T FIND CORRECT ORDER		Something's really screwed up.
F	1014	EQN	NOT ENOUGH SKYLINE STORAGE	n, *maxs*	Required skyline storage, n, exceeds dimension, *maxs*.

B I	Error #	Routine Name	Explanation	word $jjjj, kkkk$	Meaning
F	2001	FIELD1D	TFINAL LESS THAN DELTA T		Final time or time step in FIELD command is incorrect.
F	2002	FIELD1D	Bmax IS ZERO		Solution trivial. B=0 everywhere check the driving function.
F	2101	READBC	ILLEGAL BOUNDARY CONDITION TYPE	cmmnd i, j	Misspelled BC command, cmmnd?
F	2102	READBC	MORE THAN 2 SURFACE CONDITIONS APPLIED	cmmnd nel	Only 2 surface conditions per element side allowed.
F	2103	READBC	TOO MANY SURFACE BC's	cmmnd $lsrf, mxsrfbc$	Surface condition $lsrf$ exceeds allowed $mxsrfbc$
F	2104	READBC	ZERO DELTA TIME	cmmnd nel1,nel2	Time dependent BC defined with zero delta time.
F	2201	NUCDPTH	PARALLEL BOUNDARY SEGMENTS	$nel, nel1$	Shouldn't happen. Something's screwy with the BCs
F	2202	NUCDPTH	NO BOUNDARY FOUND	$nel1, nel1$	Something's screwy. No boundary between element,$nel1$,and plasma.
F	2203	NUCDPTH	MORE THAN TWO BOUNDARY POINTS	n, nn	Intermediate structure def'n messed up.
F	2204	FOFPHI	PHI GREATER THAN 180		Peaking factor for $f(\theta)$ only goes from 0 to 180
F	2301	STIFF1D	ZERO OR NEGATIVE JACOBIAN (AREA)		Wrong ordering. Clockwise order of nodes? Mesh messed up.

B I	Error #	Routine Name	Explanation	word *jjjj,kkkk*	Meaning
F	2401	POST1D	NO CONDUCTING MATERIAL		Something's messed up. No material found with nonzero σ.
F	2402	POST1D	ZERO OR NEGATIVE JACOBIAN (AREA)	*n,idum*	Shouldn't happen. Error would be found in STIFF1D. Mesh screwy after restart?
F	2403	POST1D	NO MATERIAL MATCH	*nel,mat*	Shouldn't happen. Interface testing is confused.
F	2404	CURRVST	NI = 0		Total NI is zero and not at EOP. Check VOLT and VOLTAGE commands.
F N	2405	CURRVST	MAXIMUM RISE TIME REACHED		Not at NI max. after max. rise time reached. Time stepping stops.

B I	Error #	Routine Name	Explanation	word *jjjj,kkkk*	Meaning
F	3001	THERMAL	TFINAL IS LESS THAN DELTA T		Final time or time step on THERMAL command is screwed up.
F	3002	THERMAL	Tmax IS ZERO		Solution trivial. T=0 everywhere, check the driving function.
F	3101	TSTIFF	ZERO OR NEGATIVE JACOBIAN (AREA)		Wrong ordering. Clockwise order of nodes? Mesh messed up.
F	3201	Q0DATA	TOO MANY Q0DATA SETS	*nsets,maxsets*	Data sets for Q0, *nsets*, exceeds dimension, *maxsets*.
F	3202	Q0DATA	TOO MANY Q0DATA POINTS	*npts,maxpts*	Points in set for Q0,*npts*, *npts*, exceeds dimension, *maxpts*.
F	3203	Q0DATA	COULDN'T FIND TEMP	*idum,nset*	Couldn't find Temp in data set, *nset*. Data screwy?
F	3204	Q0DATA	NO Q0DATA FILE FOUND		Couldn't find file Q0DATA.DAT in current directory.
F	3205	Q0DATA	EOF ENCOUNTERED Q0DATA.DAT		Premature EOF encountered during read. *npts* or *nsets* wrong?
F	3301	RHODATA	TOO MANY RHODATA SETS	*nsets,maxsets*	Data sets for RHO, *nsets*, exceeds dimension, *maxsets*.
F	3302	RHODATA	TOO MANY RHODATA POINTS	*npts,maxpts*	Points in set for RHO, *npts*, exceeds dimension, *maxpts*.
F	3303	RHODATA	COULDN'T FIND TEMP	*idum,nset*	Couldn't find Temp in data set, *nset*. Data screwy?
F	3304	RHODATA	NO RHODATA FILE FOUND		Couldn't find file RHODATA.DAT in current directory.
F	3305	RHODATA	EOF ENCOUNTERED RHODATA.DAT		Premature EOF encountered during read. *npts* or *nsets* wrong?

B I	Error #	Routine Name	Explanation	word *jjjj,kkkk*	Meaning
F	3401	CPDATA	TOO MANY CPDATA SETS	*nsets,maxsets*	Data sets for CP, *nsets*, exceeds dimension, *maxsets*.
F	3402	CPDATA	TOO MANY CPDATA POINTS	*npts,maxpts*	Points in set for CP, *npts*, exceeds dimension, *maxpts*.
F	3403	CPDATA	COULDN'T FIND TEMP	*idum,nset*	Couldn't find Temp in data set, *nset*. Data screwy?
F	3404	CPDATA	NO CPDATA FILE FOUND		Couldn't find file CPDATA.DAT in current directory.
F	3405	CPDATA	EOF ENCOUNTERED CPDATA.DAT		Premature EOF encountered during read. *npts* or *nsets* wrong?
F	3501	XKDATA	TOO MANY XKDATA SETS	*nsets,maxsets*	Data sets for XK, *nsets*, exceeds dimension, *maxsets*.
F	3502	XKDATA	TOO MANY XKDATA POINTS	*npts,maxpts*	Points in set for XK, *npts*, exceeds dimension, *maxpts*.
F	3503	XKDATA	COULDN'T FIND TEMP		Couldn't find Temp in data set, *nset*. Data screwy?
F	3504	XKDATA	NO XKDATA FILE FOUND		Couldn't find file XKDATA.DAT in current directory.
F	3505	XKDATA	EOF ENCOUNTERED XKDATA.DAT		Premature EOF encountered during read. *npts* or *nsets* wrong?

B I	Error #	Routine Name	Explanation	word *jjjj, kkkk*	Meaning
F	3601	HDATA	TOO MANY HDATA SETS	*nsets, maxsets*	Data sets for H, *nsets*, exceeds dimension, *maxsets*.
F	3602	HDATA	TOO MANY HDATA POINTS	*npts, maxpts*	Points in set for H, *npts*, exceeds dimension, *maxpts*.
F	3603	HDATA	COULDN'T FIND TEMP		Couldn't find Temp in data set, *nset*. Data screwy?
F	3604	HDATA	NO HDATA FILE FOUND		Couldn't find file HDATA.DAT in current directory.
F	3605	HDATA	EOF ENCOUNTERED HDATA.DAT		Premature EOF encountered during read. *npts* or *nsets* wrong?
F	3701	SSCPVST	INCONEL CP FIT INVALID BELOW 70 K		Mixture command is really only valid above 70 K

B I	Error #	Routine Name	Explanation	word *jjjj,kkkk*	Meaning
F	4001	FIELD2D	TFINAL LESS THAN DT		Final time less than time step on FIELD2D command line.
F	4002	FIELD2D	INCORRECT ITERATIVE SOLUTION	type	Solution type misspelled?
F	4003	FIELD2D	Amax IS ZERO		Solution trivial. A=0 everywhere check the driving function.
F	4004	FIELD2D	ITERATIVE DIVERGENCE	*iter,istep*	Nonlinear iteration is not converging..
F	4005	FIELD2D	ITERATION DID NOT CONVERGE	*iter,istep*	Nonlinear iteration failed to converge.
F	4101	STIFF2D	ZERO OR NEGATIVE JACOBIAN (AREA)		Wrong ordering. Clockwise order of nodes? Mesh messed up.
F	4201	JVST	TOO MANY J DATA SETS	*nsets,maxsets*	Data sets for J, *nsets*, exceeds dimension, *maxsets*.
F	4202	JVST	TOO MANY J DATA POINTS	*npts,maxpts*	Points in set for J,*npts*, *npts*, exceeds dimension, *maxpts*.
F	4203	JVST	COULDN'T FIND J	*idum,nset*	Couldn't find Temp in data set, *nset*. Data screwy?
F	4204	JVST	NO JVST.DAT FILE FOUND		Couldn't find file JVST.DAT in current directory.
F	4205	JVST	EOF ENCOUNTERED ON JVST.DAT		Premature EOF encountered during read. *npts* or *nsets* wrong?
F	4301	POST2D	ZERO OR NEGATIVE JACOBIAN (AREA)	*n,idum*	Shouldn't happen. Error would be found in STIFF2D. Mesh screwy after restart?
F	4302	POST2D	NO MATERIAL MATCH	*nel,mat*	Shouldn't happen. Interface testing is confused.

B I	Error #	Routine Name	Explanation	word $jjjj, kkkk$	Meaning
F	4901	SKYFAC	ZERO DIAGONAL		Zero Diagonal during elimination Mesh or properties?
F N	5001	POST	ILLEGAL POST COMMAND	cmmnd	Incorrect POST command, cmmnd
F	5101	ADDFORCE	NO MATCH ON FOR082	l, nel	Element, nel, not found on FOR082. Incompatible FOR082 and mesh?
F	5102	ADDFORCE	NO ELEMENT MATCH	$i1, j1$	Requested element with 1^{st} node $(i1, j1)$ not found.
F N	5201	FINDEQN	INCORRECT FINDEQN COMMAND	cmmnd	Incorrect FINDEQN command, cmmnd.
F N	5202	FINDEQN	NO COORDINATE MATCH	cmmnd i, j	No match found for coordinate command, cmmnd. (i, j) (if cmmnd=ij)
F N	5203	FINDEQN	NO ELEMENT MATCH	i, j	No match found for element with requested node, (i, j).
F N	5204	FINDEQN	NO EQUATION MATCH	$neq, idum$	No element match found for equation, neq.
F	5301	PFFORCES	ZERO OR NEGATIVE JACOBIAN		Wrong ordering. Clockwise order of nodes? Mesh messed up.
F	5302	PFFORCES	NO TIME MATCH ON FOR085		Time point not found on file FOR085.
F N	5401	MACRO	DIDN'T FIND FOUR CORNERS		Not a documented routine You shouldn't be here anyway.

B I	Error #	Routine Name	Explanation	word *jjjj, kkkk*	Meaning
N[1] N	6001	MAPPLOT	INCORRECT PLOT TYPE	cmmnd	Incorrect PLOT command, cmmnd
N N	6101	VECTOR	ZERO MAXIMUM VALUE	cmmnd	Vector routine found a 0 maximum value for plot type, cmmnd.
N N	6102	VECTOR	NO DATA AVAILABLE	cmmnd	Vector routine can't find correct time for plot type, cmmnd.
N N	6201	CONTOUR	INCORRECT CONTOUR TYPE	cmmnd	Incorrect contour command, cmmnd.
N N	6202	CONTOUR	TOO MANY CONTOURS	*ncon, maxcon*	No. of requested contours, *ncon*, exceeds dimension, *maxcon*.
N N	6202	CONTOUR	NO DATA AVAILABLE	cmmnd	Contour routine can't find correct time for plot type, cmmnd,
N N	6301	FUNCPLT	INCORRECT FUNCTION TYPE	cmmnd	Incorrect plot type, cmmnd.
N N	6302	FUNCPLT	INCORRECT INDEPENDENT VARIABLE	cmmnd	Incorrect independent variable name, cmmnd.
N N	6303	FUNCPLT	MAXIMUM NUMBER OF CURVES EXCEEDED	*idum, maxcurves*	Number of curves exceeds dimension, *maxcurves*.
N N	6304	FUNCPLT NFUNCPLT	MAXIMUM NUMBER OF POINTS EXCEEDED	*idum, maxpoints*	Number of points exceeds dimension, *maxpoints*.
N N	6305	FUNCPLT	NO MIN/MAX FOUND		Routine couldn't find dependent variable min. or max. No data?
N N	6306	FUNCPLT	NO DATA AVAILABLE	cmmnd	Function routine can't find correct time for plot type, cmmnd

[1] MITMAP assumes the next line is a data line for CONTOURS or FUNCTION plots and reads it before control is returned to MITMAP. If TYPE should have been ELEMENTS or VECTORS, or if the command was to call for more than one extra data line, the next command will be screwed up and might invoke an incorrect fatal command error.

Chapter 11.0 EXAMPLE PROBLEMS

In order to facilitate the understanding of the input deck structure, several example problems with their input data are presented in this chapter. Limited output is also given.

The examples presented are for: (1) a static analysis of an iron bound solenoid; (2) a RESTART postprocessing run for the solenoid; and (3) a current diffusion problem.

11.1 Iron-bound Solenoid

The first example problem is that of determining the static field distribution for an iron-bound solenoid. The input file is reproduced in Table 11.1. The UNITS, CM option is used. The IREPEAT and JREPEAT grid generation options are used. The comment character, !, is used to comment the grid part numbers into the data deck for convenience only. These numbers (after !) are not used by **MAP**. It is essential, however, that the user keep track of the part numbers if IREPEAT or JREPEAT are to be used.

QUADs with regular ordering are used exclusively. All elements are defined by the first ILOOP, JLOOP, QUAD set as being material type 1 elements – air in this case. The remaining ILOOP, JLOOP, and MAT lines change the material property designation for the elements in the coil and iron to the proper values. The boundary conditions are set such that $A_\theta = 0$ along the outer boundaries of the problem. Along the r-axis, the boundary condition is the natural condition and need (can) not be set.

The finite element grid is shown in Fig. 11.1. Only half the problem is modeled due to the symmetry about the $Z = 0$ plane. The material regions are outlined, and boundary conditions are indicated.

Included are a large number of plot calls to show the flexibility of the routines. The plots generated by these commands are shown in Figs. 11.2a-g. Figure 11.2a shows a contour plot of constant flux – the plot is also sometimes called a field line plot. The magnetic field is directed along the lines of constant flux. The magnitude of the flux density is inversely proportional to the spacing for plane problems – in axisymmetric problems there is also a 1/r variation.

Figure 11.2b shows lines of constant field homogeneity and the contour labeling option. Figures 11.2c and d show the total flux density versus radius at z=0 and versus axial position along the axis of symmetry, respectively. Since these two paths cross the air-vacuum boundaries, the tangential component of the flux density experiences a jump across the boundary. **MAP** handles the jump by plotting to small squares that represent the two values of the flux density on either side of the interface. Since the tangential component of flux density is zero along the axis, the squares are (almost) coincident in the second figure. Finally, there are three vector plots of flux density, force per unit length, and iron

magnetization shown in Figs. 11.2e-g, respectively.

In addition, selected portions of the printed output are extracted and shown in Table 11.2. The * is used to denoted the two sides of the interface between the air and iron in the nodal point print section of the output.

Table 11.1 – MAP Example 1 – Input

```
$ MAP EXAMPLE PROBLEM 1
problem,field2d
units,cm
setup,1
! Define material properties
air, 1
coil,2,5.5e7
iron,3,,,1.0
end,materials
! Define grid points
1, 1, 6, 6                                    !Part 1
 0, 25, 25,  0
 0,  0, 25, 25
!
jrepeat,1,1, 6, 8, 25, 30, 30, 25             !Part 2
jrepeat,1,1, 8,11, 30, 40, 40, 30             !Part 3
jrepeat,1,1,11,14, 40, 47, 47, 40             !Part 4
jrepeat,1,1,14,21, 47,100,100, 47,.2,.2       !Part 5
!
irepeat,1,5, 6, 9, 25, 25, 40, 40             !repeat parts 1-5
irepeat,1,5, 9,12, 40, 40, 47, 47
irepeat,1,5,12,16, 47, 47,100,100,.2,.2
end,grid
! Define elements
iloop,20,1                !Define all elements as air
 jloop,15,1
  quad,1,1,1
 jend
iend
iloop,2,1                 !Redefine coil elements
 jloop,5,1
  mat,2,6,1
 jend
iend
iloop,3,1                 !Redefine iron elements in the return frame
 jloop,8,1
  mat,3,11,1
 jend
iend
iloop,13,1                !Redefine iron elements in the pole
 jloop,3,1
  mat,3,1,9
 jend
iend
end,elements
! Define BC's here for plotting -- they could be immediately after FIELD2D
na,1,  1, 1,16,0
na,1, 16,21,16,0
na,21,  ,21,16,0
end,bc
plot,elements,0,110,0,110,,,  ,,,,2           !Plot BC's
```

MAP – 11-3

```
plot,elements,0, 60,0, 60,,,-1
!
field2d,newton,25,1.e-4
plot,contours,flux,0,100,0,100
35
plot,contours,flux,0,60,0,60
35
plot,contours,homogeneity,0,20,0,20
1,1
20,-18,2,20,1,0,65,-1
!
plot,function,b,0,60,0,0
r,1,1,1
plot,function,b,0,60,0,0
z,1,1,1
plot,vectors,fields,0,60,0,60,,,-1
plot,vectors,forces,0,60,0,60,,,-1
plot,vectors,magnetization,0,60,0,60,,,-1
stop
```

MAP - 11-4

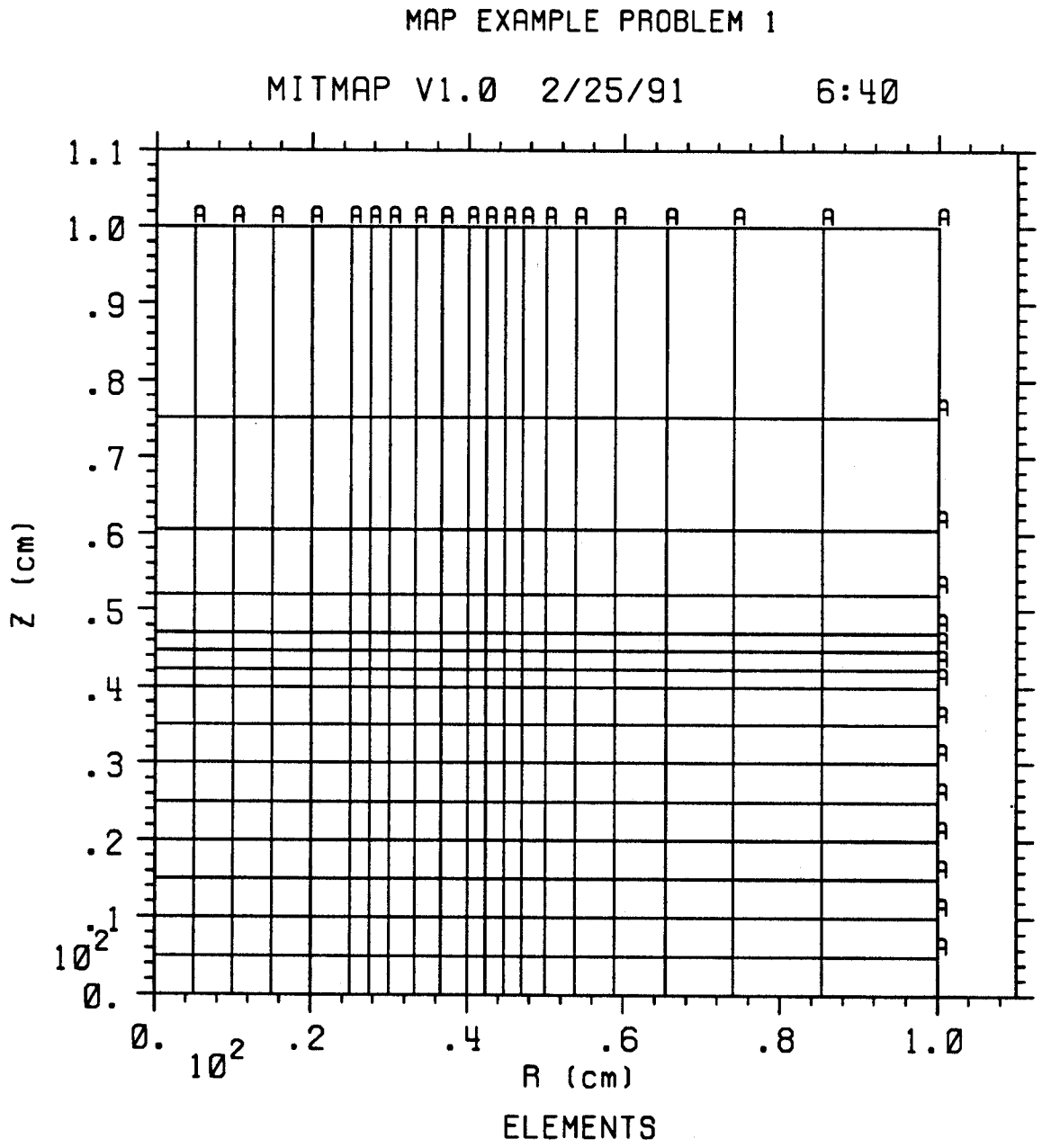


Figure 11.1 **MAP** Finite Element Grid

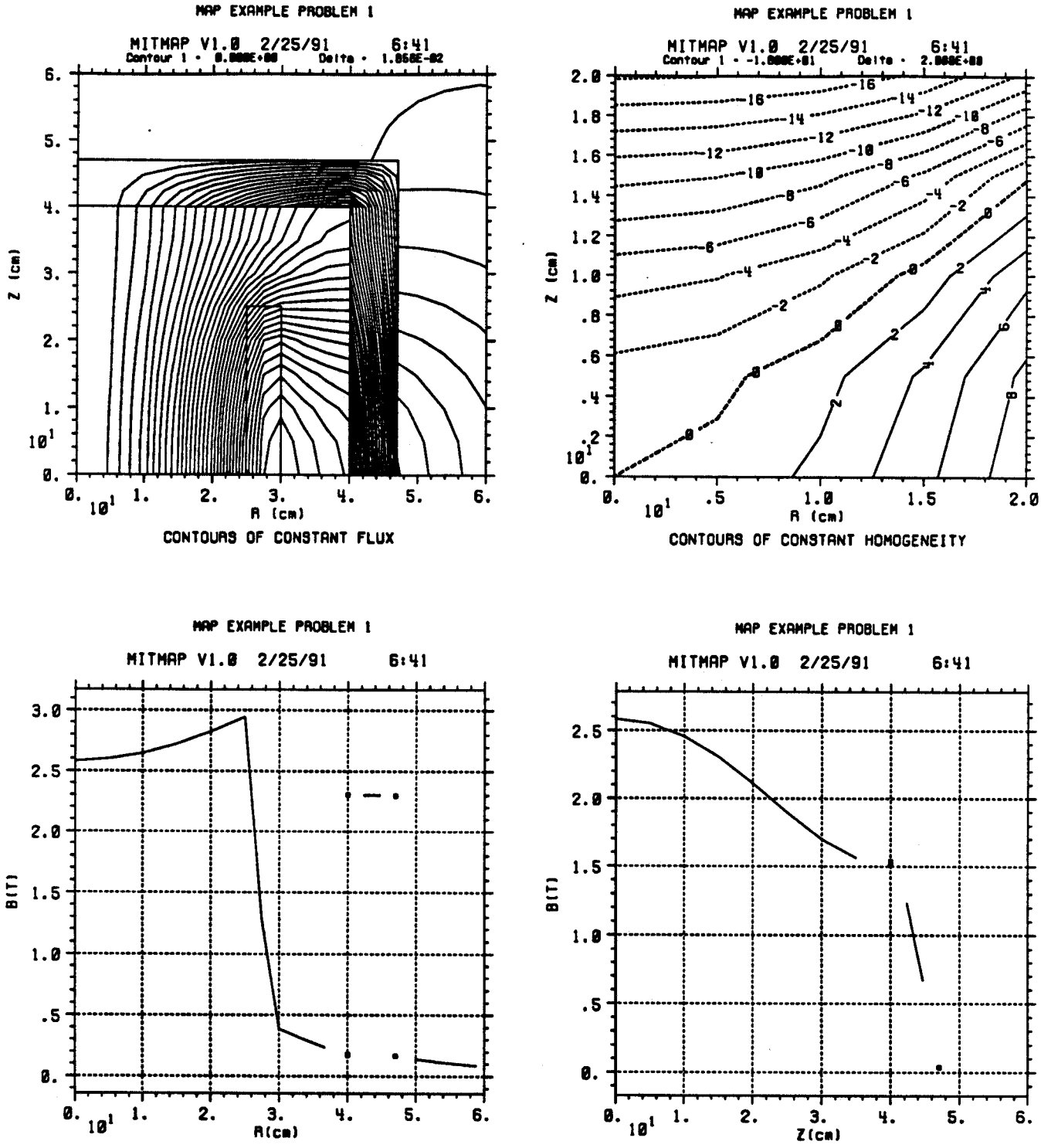


Figure 11.2 (a)-(d) **MAP** Plotted Output

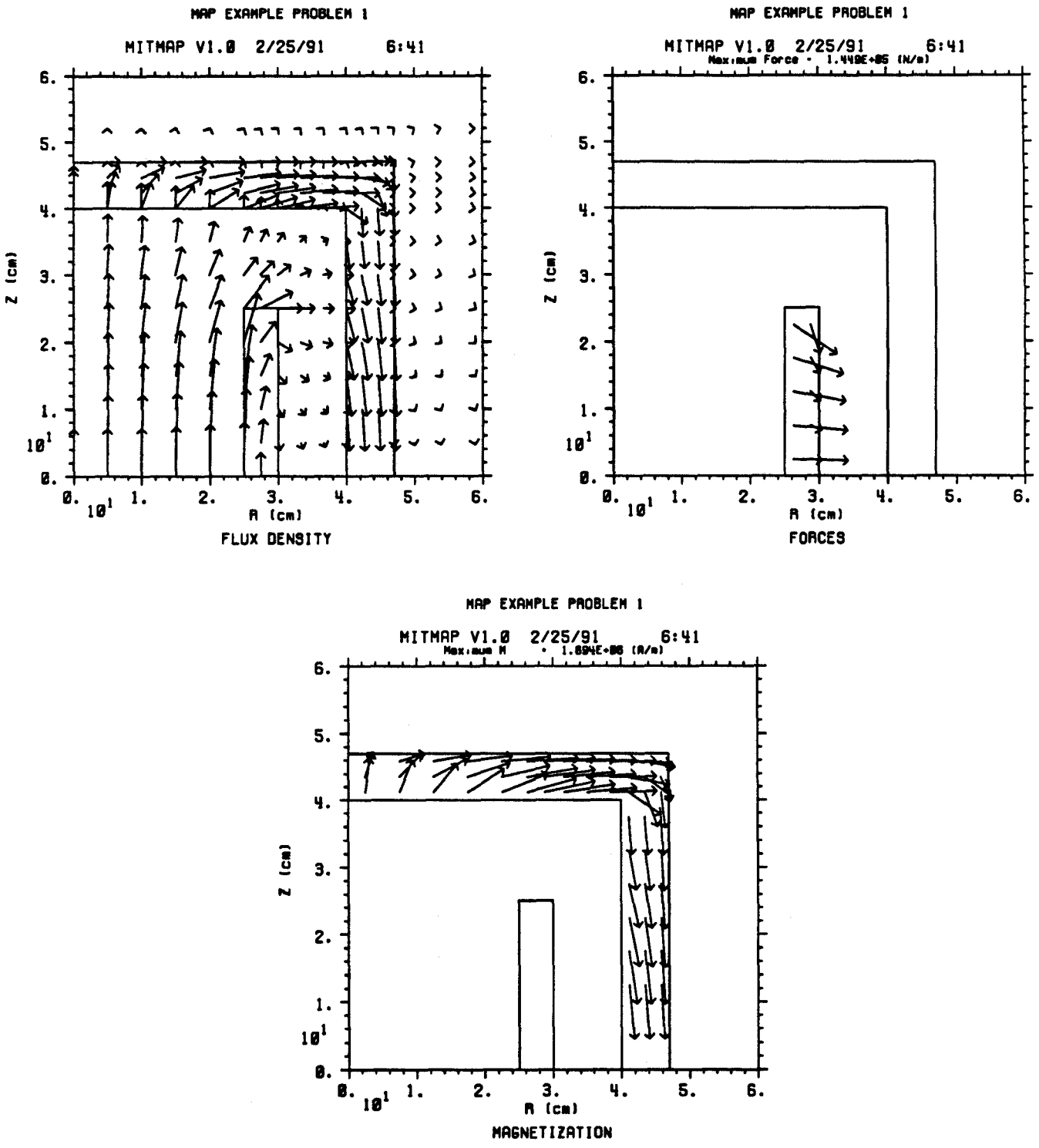


Figure 11.2 (e)-(g) **MAP** Plotted Output

Table 11.2 – MAP Example 1 – Selected Output

```
        MITMAP - VAX VERSION 1.0 - LAST MODIFIED 05/24/91

        MITMAP V1.0  5/28/91    8:43

LINE   DIRECT LIST OF INPUT DATA
    1  $ MAP EXAMPLE PROBLEM 1
    2 problem,field2d
        .
        .
        .
   68 plot,vectors,magnetization,0,60,0,60,,,-1
   69 stop

 Time in     reader =  1.01000E+00 seconds

      Two Dimensional Field - One Dimensional Currents

                Factor for System Totals =  1.000E+00

 Time in problem    =  0.00000E+00 seconds

            The unit of length for input and output is cm

 Time in units      =  0.00000E+00 seconds

            Electromagnetic and Thermal Material Properties

no  material      Lamda J      Conductivity    Iron        Constraint    J FLAG
                  (A/m**2)       (Mho/m)     Fraction
1  air            0.000E+00    0.000E+00    0.000E+00    0.000E+00    0.000E+00
2  coil           5.500E+07    0.000E+00    0.000E+00    0.000E+00    0.000E+00
3  iron           0.000E+00    0.000E+00   -1.000E+00    0.000E+00    0.000E+00

        BH CURVE FOR MATERIAL iron       WITH MATERIAL NUMBER    3
    B (T)       H (A/m)     MU-relative      NU        D(NU)/D(B*B)
 0.0000E+00   0.0000E+00   3.9931E+03   2.5043E-04   0.0000E+00
 8.9443E-01   1.7825E+02   3.9930E+03   2.5044E-04   9.0466E-05
                             .
                             .
                             .
 7.4887E+00   4.2567E+06   1.4000E+00   7.1429E-01   0.0000E+00

            Data for  20 grid parts have been read
            Data for  300 elements have been read
```

Table 11.2 – MAP Example 1 – Selected Output – continued

```
MITMAP V1.0  5/28/91    8:43

   MAP EXAMPLE PROBLEM 1

     Ordering j first
Equations              =     971
Constraints            =       0
Total Equations        =     971
Inside  interface eqn  =      19
Outside interface eqn  =     199
Sky line storage       =   50426

     Ordering i first
Equations              =     971
Constraints            =       0
Total Equations        =     971
Inside  interface eqn  =      19
Outside interface eqn  =     199
Sky line storage       =   50426

     Ordering element
Equations              =     971
Constraints            =       0
Total Equations        =     971
Inside  interface eqn  =      35
Outside interface eqn  =     292
Sky line storage       =   46743

     Sky-Line Solver Data
Equations              =     971
Constraints            =       0
Total Equations        =     971
Min interface eqn      =     292
Max storage available  =  900000
Max storage required   =   32036
```

Table 11.2 – MAP Example 1 – Selected Output – continued

```
                MITMAP V1.0  5/28/91    8:43

                   MAP EXAMPLE PROBLEM 1

             Boundary conditions

  type      element    side       value
             or  i     or j

na              1        1     0.000E+00
na              1        2     0.000E+00
                .
                .
                .
na              1       15     0.000E+00
na              1       16     0.000E+00
na              1       16     0.000E+00
na              2       16     0.000E+00
                .
                .
                .
na             20       16     0.000E+00
na             21       16     0.000E+00
na             21        1     0.000E+00
na             21        2     0.000E+00
                .
                .
                .
na             21       16     0.000E+00
.
Time in setup      =  1.87000E+00 seconds
```

Table 11.2 – MAP Example 1 – Selected Output – continued

```
              MITMAP V1.0  5/28/91    8:43

                  MAP EXAMPLE PROBLEM 1
                Steady State Solution Is Sought
        Solution of nonlinear algebraic equations by newton     iteration
        maximum number of iterations is    25
        tolerance for iterative convergence is  1.000E-04
        number of iterations before left hand side is updated is       1
        relaxation factor is  1.000E+00

         Print level =   0          Plot level =   0

Time in form stiff =  2.73000E+00 seconds
Time in solve eqns =  3.05000E+00 seconds

 iteration   1 delta A max  3.787E-01 (normalized  1.000E+00 ) equation  704
sum dasq/sum asq  1.000E+00  sum (dasq/asq)  3.038E-02   max(da/a)  1.000E+00
Time in form stiff =  2.83000E+00 seconds
Time in solve eqns =  2.38000E+00 seconds

 iteration   2 delta A max  5.765E-02 (normalized  1.575E-01 ) equation  375
sum dasq/sum asq  8.196E-02  sum (dasq/asq)  2.200E-02   max(da/a)  9.904E-01

 iteration   3 delta A max -7.693E-03 (normalized  2.095E-02 ) equation  500
sum dasq/sum asq  1.358E-02  sum (dasq/asq)  2.285E-03   max(da/a)  2.573E-01

 iteration   4 delta A max  4.073E-03 (normalized  1.110E-02 ) equation  399
sum dasq/sum asq  4.687E-03  sum (dasq/asq)  6.814E-04   max(da/a)  8.419E-02
                          .
                          .
                          .
 iteration  10 delta A max  6.183E-05 (normalized  1.685E-04 ) equation  447
sum dasq/sum asq  2.775E-05  sum (dasq/asq)  1.561E-06   max(da/a)  6.522E-04

 iteration  11 delta A max  1.658E-05 (normalized  4.516E-05 ) equation  447
sum dasq/sum asq  7.934E-06  sum (dasq/asq)  4.889E-07   max(da/a)  2.057E-04

  iterative procedure converged to specified tolerance after  11 iterations
```

Table 11.2 – MAP Example 1 – Selected Output – continued

```
                    MITMAP V1.0  5/28/91     8:43

                        MAP EXAMPLE PROBLEM 1

                          Element Forces per Unit Legnth

 Nel    mat  i1   j1     rc(cm)      zc(cm)      Fr(N/m)      Fz(N/m)      Ft(N/m)       NI(A)

  76     2    6     1   2.625E+01   2.500E+00   1.4611E+05  -4.4415E+03   1.4618E+05   6.8750E+04
  77     2    6     2   2.625E+01   7.500E+00   1.4423E+05  -1.4023E+04   1.4491E+05   6.8750E+04
  78     2    6     3   2.625E+01   1.250E+01   1.4009E+05  -2.6084E+04   1.4250E+05   6.8750E+04
  79     2    6     4   2.625E+01   1.750E+01   1.3210E+05  -4.3752E+04   1.3916E+05   6.8750E+04
                              :
                              :
  94     2    7     4   2.875E+01   1.750E+01   2.5321E+04  -4.3228E+04   5.0098E+04   6.8750E+04
  95     2    7     5   2.875E+01   2.250E+01   2.5307E+04  -7.8122E+04   8.2118E+04   6.8750E+04

                          Components of Total Force By Material

 mat   name          Rc(cm)      Zc(cm)      Rbar(cm)     Zbar(cm)      Fr(N/m)      Fz(N/m)
  2    coil       2.7500E+01  1.2500E+01  2.7491E+01   1.1927E+01   2.1246E+05  -8.9258E+04

   i    j     r(cm)       z(cm)      Flux(V-s)     Jth(A/m^2)       Br(T)        Bz(T)        Bt(T)        H(A/m)       dir(deg)

   1    1   0.000E+00   0.000E+00   0.00000E+00   0.00000E+00   3.53670E-07   2.59768E+00   2.59768E+00   2.06717E+06   9.000E+01
   1    2   0.000E+00   5.000E+00   0.00000E+00   0.00000E+00  -1.73913E-05   2.56643E+00   2.56643E+00   2.04228E+06   9.000E+01
   1    3   0.000E+00   1.000E+01   0.00000E+00   0.00000E+00  -1.07259E-04   2.47313E+00   2.47313E+00   1.96796E+06   9.000E+01
   1    4   0.000E+00   1.500E+01   0.00000E+00   0.00000E+00  -3.14880E-04   2.32142E+00   2.32142E+00   1.84709E+06   9.001E+01
                              :
                              :
 * 1    9   0.000E+00   4.000E+01   0.00000E+00   0.00000E+00  -2.37770E-03   1.54849E+00   1.54849E+00   8.71002E+02   9.009E+01
   1   10   0.000E+00   4.233E+01   0.00000E+00   0.00000E+00   2.19388E-03   1.25811E+00   1.25811E+00   2.95157E+02   8.990E+01
   1   11   0.000E+00   4.467E+01   0.00000E+00   0.00000E+00   3.04638E-03   6.94623E-01   6.94629E-01   1.19829E+02   8.975E+01
   1   12   0.000E+00   4.700E+01   0.00000E+00   0.00000E+00  -6.73115E-04   5.81728E-02   5.81767E-02  -4.89598E+00   9.066E+01
 * 1   12   0.000E+00   4.700E+01   0.00000E+00   0.00000E+00  -4.54416E-06   5.33352E-02   5.33352E-02   4.24374E+04   9.000E+01
   1   13   0.000E+00   8.960E+01   0.00000E+00   0.00000E+00  -1.83882E-06   1.89311E-02   1.89311E-02   1.50582E+04   9.001E+01
                              :
                              :
  21   15   5.000E+02   2.870E+02   0.00000E+00   0.00000E+00  -3.58587E-06  -1.19674E-04   1.19728E-04   8.83522E+01  -9.172E+01
  21   16   5.000E+02   5.000E+02   0.00000E+00   0.00000E+00  -8.19618E-07   7.18577E-06   7.23237E-06  -1.16774E+01   9.651E+01
```

Table 11.2 – MAP Example 1 – Selected Output – continued

```
                MITMAP V1.0  5/28/91    8:43

                   MAP EXAMPLE PROBLEM 1

      Total Ampere-turns          =  6.8749994E+05  Amperes
      Total Stored Energy         =  1.9436267E+05  Joules
      Total Joule Losses          =  0.0000000E+00  Joules
                                     ----------------------
      Total Stored and Dissipated =  1.9436267E+05  Joules
                                     ----------------------
      Total       Power           =  0.0000000E+00  Watts


                    Area and Maximum and Minimum J , B, and H by Material

mat   material    Area       Amp-Turns     gphi        Jmax         Jmin         Bmax       Bmin        Hmax         Hmin
                  (m**2)     (A-t)         (V/m)       (A/m**2)     (A/m**2)     (T)        (T)         (A/m)        (A/m)

 1 air         2.4927E+01  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  2.9635E+00  7.2324E-06  2.3579E+06 -3.1245E+03
 2 coil        1.2500E-02  6.8750E+05  0.0000E+00  5.5000E+07  5.5000E+07  2.9643E+00  3.6603E-01  2.3586E+06 -2.7081E+05
 3 iron        6.0900E-02  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  2.2803E+00  5.8177E-02  1.2119E+05 -1.5165E+03

Time in field2d    =  6.04900E+01 seconds

at time =  0.000E+00  max =  6.547E-01  min =  0.000E+00

Time in plot       =  3.11000E+00 seconds

at time =  0.000E+00  max =  6.547E-01  min =  0.000E+00

Time in plot       =  2.73000E+00 seconds

stop

Time in stop       =  6.92100E+01 seconds
```

11.2 Iron-bound Solenoid – Restart

After a successful run of **MAP**, there are frequently calls for additional graphics output, processing or for additional postprocessing using either the POST option or a UPOST routine. The RESTART option has been built into **MITMAP** to allow for these eventualities.

For example, the previous example problem could have been run without any of the plotting requests at the end of the input of file. Then, an interactive session with terminal input and output could be used to generate the plots. Alternatively, the input file shown in Table 11.3 could be run. Another alternative would be to delete or to comment out the FIELD2D command line in the original input file in Table 11.1 and add the RESTART,PLOT command at that point. There is a small amount of additional processing associated with the mesh generation, but the advantage is that the full input file with material properties, mesh, boundary conditions, *etc.* is available for documentation purposes.

Table 11.3 – **MAP** Example 2 – Input

```
$ MAP EXAMPLE PROBLEM 1
problem,field2d
units,cm
restart,plot
plot,contours,flux,0,100,0,100
35
plot,contours,flux,0,60,0,60
35
plot,contours,homogeneity,0,20,0,20
1,1
20,-18,2,20,1,0,65,-1
!
plot,function,b,0,60,0,0
r,1,1,1
plot,function,b,0,60,0,0
z,1,1,1
plot,vectors,fields,0,60,0,60,,,-1
plot,vectors,forces,0,60,0,60,,,-1
plot,vectors,magnetization,0,60,0,60,,,-1
stop
```

11.3 Eddy Current Analysis

The final example is that of a transient or eddy current analysis. A long conductor (and return) are inside a pair of parallel long conducting strips. Symmetry is used to model only one quarter of the problem. The coil current is ramped linearly from zero to its full value in 0.25 seconds; it is held constant for 0.25 seconds; and then ramped down to zero in 0.25 seconds.

The input file for this example is listed in Table 11.5. It can be seen that the PLANE option is used to define the problem as two-dimensional rather than axisymmetric. The grid is essentially the same as in the first example with a finer mesh in the conducting shell.

In addition, the J-flag option on the coil material property line is used to indicate that the current density versus time is to be read from the file JVST.DAT and the result multiplied by the J_0 value of 5.5×10^7 A/m^2. The JVST.DAT file contained the following information:

Table 11.4 – JVST.DAT

```
1                  ! one set of data
  5                ! number of data points in this set
  0.00      0.0    ! start at zero
  0.25      1.0    ! ramp up   to full current density in  0.25 s
  0.50      1.0    ! flattop   at full current density for 0.25 s
  0.75      0.0    ! ramp down to zero current density in  0.25 s
 10.00      0.0    ! zero current
```

The last time point is used if the time marching goes past the end of current in order to capture the final decay of the eddy currents. In this example, the time marching continued for 0.25 seconds after the end of the ramp down. The FIELD2D command requests 0.05 second steps from 0 to 1 second with plotted output written at every time step.

Figure 11.3a shows the NI (ampere-turns) for materials 2 and 3, corresponding to the coil and conducting material as functions of time. The coil NI follows the JVST data since this analysis assumes the coil has a "stiff" current source. The conducting shell has eddy currents induced in the opposite direction.

Figure 11.3b shows a plot of B_y versus x along the line $y = 0$ for four time points corresponding to 0.25, 0.50, 0.75, and 1.00 seconds. The MORE option has been used to select these times. If this option were not used, curves for all 20 time points would have appeared on the plot.

Figures 11.3c-f show the field lines (lines of constant A_z) at times 0.25, 0.50, 0.75, and 1.00. A constant ΔA_z is used to indicate the transient effects. The only way in the present version of **MAP** to generate such a sequence is to make two runs – one to get the correct increment and one to plot.

Table 11.5 – **MAP** Example 3 – Input

```
$ MAP EXAMPLE PROBLEM 3
problem,field2d
plane
setup,1
air,  1
coil, 2,5.5e7,    ,,,1
metal,3,     ,5.8e7
end,materials
!
1, 1, 6, 6                                    !Part 1
 0.0, 0.25, 0.25, 0.0
 0.0, 0.0,  0.25, 0.25
!
jrepeat,1,1, 6, 8, 0.25, 0.30, 0.30, 0.25       !Part 2
jrepeat,1,1, 8,11, 0.30, 0.40, 0.40, 0.30       !Part 3
jrepeat,1,1,11,16, 0.40, 0.45, 0.45, 0.40       !Part 4
jrepeat,1,1,16,26, 0.45, 5.00, 5.00, 0.45,.1,.1 !Part 5
!
irepeat,1,5, 6, 9, 0.25, 0.25, 0.40, 0.40
irepeat,1,5, 9,12, 0.40, 0.40, 0.47, 0.47
irepeat,1,5,12,26, 0.47, 0.47, 5.00, 5.00,.1,.1
end,grid
iloop,25,1
 jloop,25,1
  quad,1,1,1
 jend
iend
iloop,2,1
 jloop,5,1
  mat,2,6,1
 jend
iend
iloop,5,1
 jloop,10,1
  mat,3,11,1
 jend
iend
end,elements
na,1,  1, 1,26,0
na,1, 26,26,26,0
na,26, 1,26,26,0
end,bc
plot,elements,0,.6,0,.6,,,-1
field2d,linear,,,3,1.0,0.05,,,,1,1  ! March 1.0 secs with 0.05 sec step
plot,function,totni,0,1.00
t
plot,function,matni,0,1.00
t,-1
2,0
3,0
0,0
```

```
plot,function,by,0,0.6
x,1,1,more
0.25
0.50
0.75
1.00
end
plot,contours,az,0.0,0.60,0.0,0.60,,,,0.25,0.25
31,,0.02
plot,contours,az,0.0,0.60,0.0,0.60,,,,0.50,0.50
31,,0.02
plot,contours,az,0.0,0.60,0.0,0.60,,,,0.75,0.75
31,,0.02
plot,contours,az,0.0,0.60,0.0,0.60,,,,1.00,1.00
31,,0.02
stop
```

MAP - 11-17

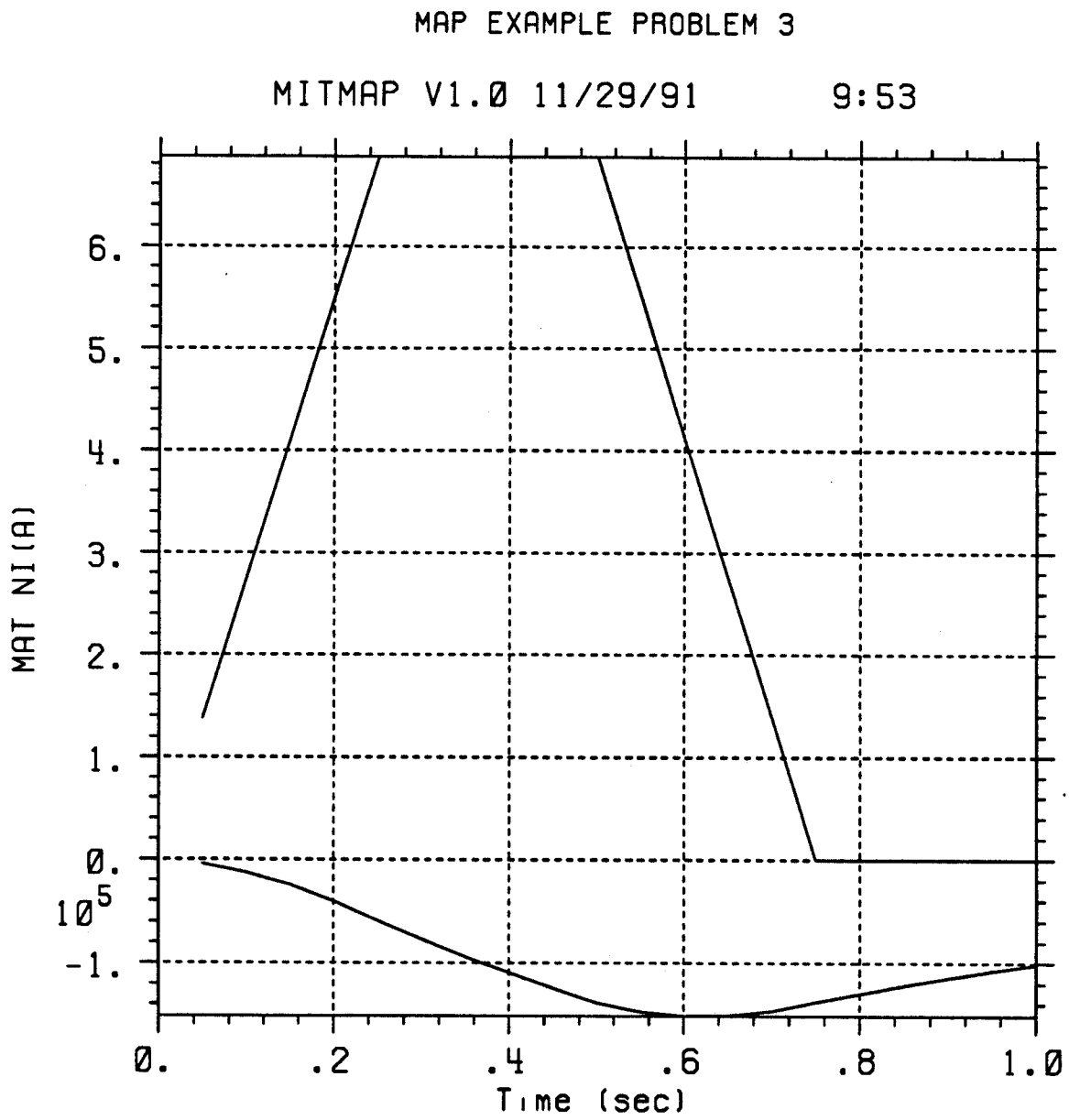


Figure 11.3a – Component NI versus time

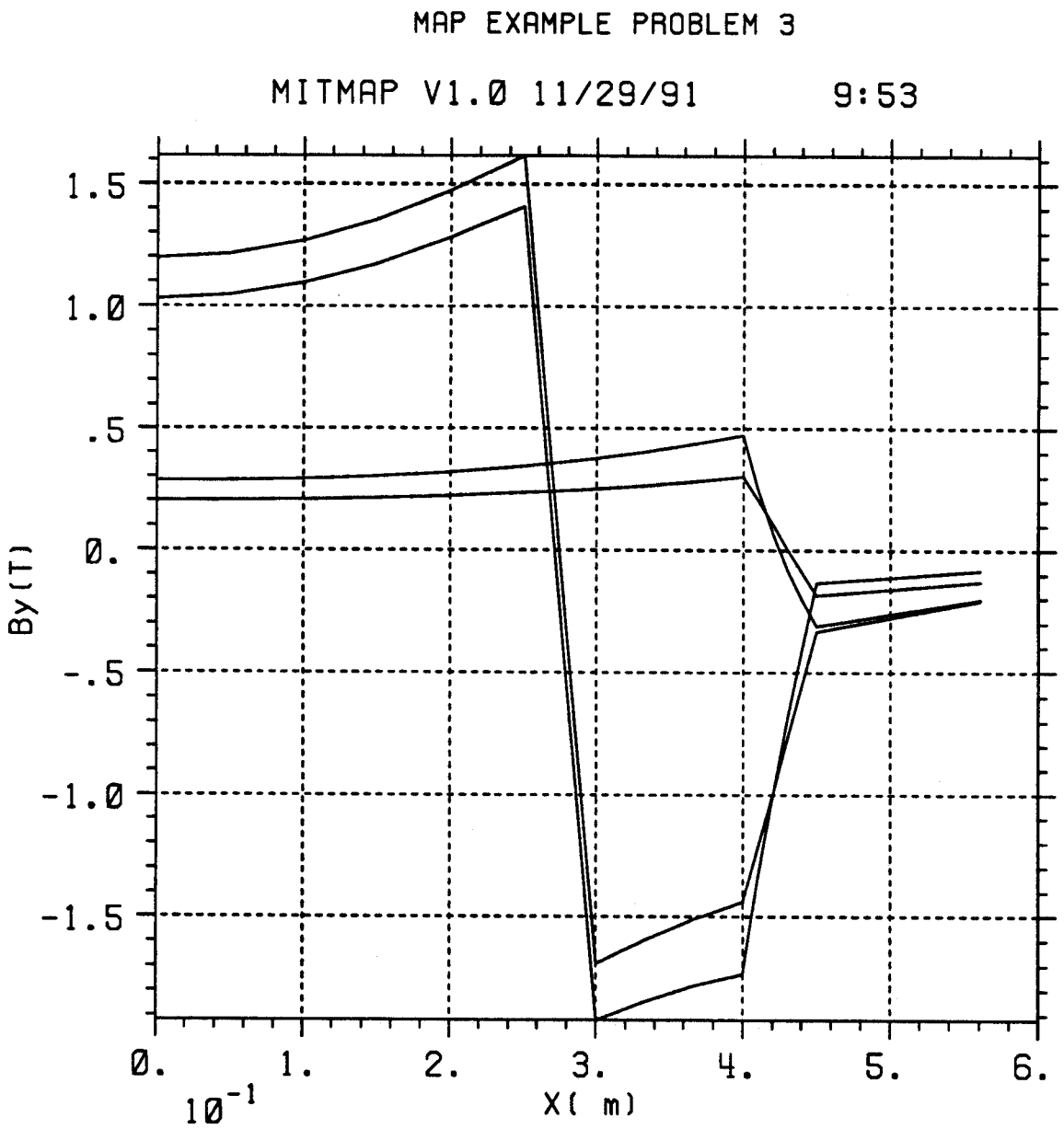


Figure 11.3b – By versus x for various times

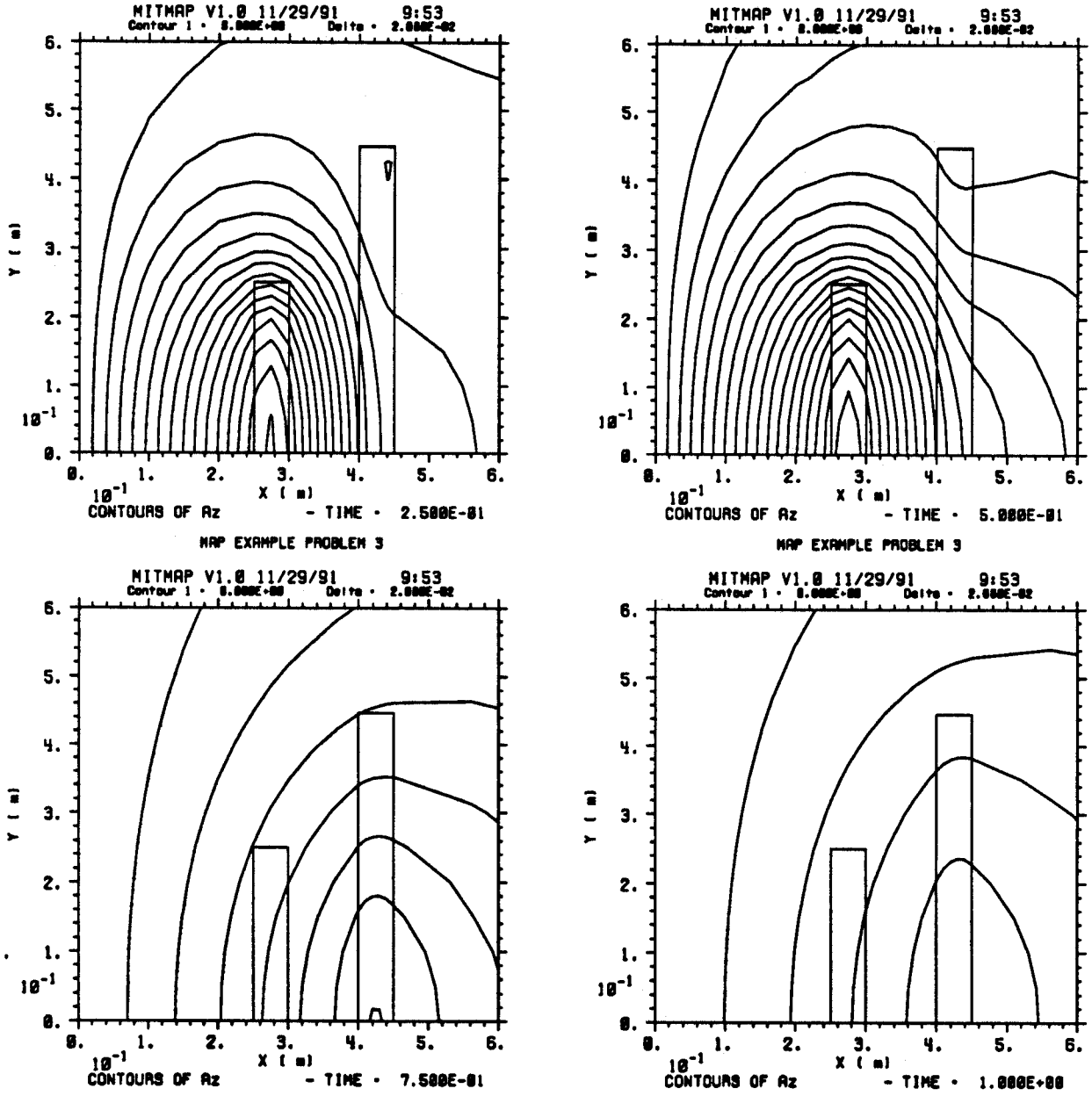


Figure 11.3 (c)-(f) Contours of Constant A_z at various times

APPENDIX A. THE TWO-DIMENSIONAL FIELD PROBLEM

A.1 Governing Equation

The penetration of the magnetic field into stationary, conducting media produced by a known current (density) distribution is governed by the following set of partial differential equations in terms of the magnetic vector potential, $\tilde{A}$ (where $\tilde{B} = \tilde{\nabla} \times \tilde{A}$).

$$\sigma \dot{\tilde{A}} + \tilde{\nabla} \times \frac{1}{\mu(|\tilde{B}|)} \tilde{\nabla} \times \tilde{A} = \tilde{0} \quad \text{in } \Omega_i \times (0,T),\ i \neq j \tag{A.1}$$

$$\tilde{\nabla} \times \frac{1}{\mu(|\tilde{B}|)} \tilde{\nabla} \times \tilde{A} = \tilde{J} \quad \text{in } \Omega_j \times (0,T) \tag{A.2}$$

subject to the constraint that

$$\tilde{\nabla} \cdot \tilde{A} = 0 \quad \text{in } \Omega \times (0,T). \tag{A.3}$$

The regions denoted by Ω_i are those over which the permeability varies continuously. The regions Ω_i have boundaries $\partial\,\Omega_i$ across which the permeability has finite jumps. The behavior of the vector potential across $\partial\,\Omega_i$ is governed by:

$$\tilde{n} \times \left[\left[\frac{1}{\mu(|\tilde{B}|)} \tilde{\nabla} \times \tilde{A} \right]\right] = \tilde{0} \quad \text{on } \partial\,\Omega_i \times (0,T) \tag{A.4}$$

where $[[\cdot]]$ denotes a jump across the interface, *i.e.*, $(\cdot)_2 - (\cdot)_1$, and $\tilde{n}$ is the unit normal vector from material (1) into material (2).

In order to complete the statement of the initial/boundary value problem, the following conditions are imposed.

$$\tilde{A}\,(\tilde{x},0) = \tilde{0} \quad \text{in } \Omega \tag{A.5}$$

$$\tilde{A}\,(\tilde{x},t) = \tilde{0} \quad \text{on } \partial\,\Omega \times (0,T). \tag{A.6}$$

where the last condition is equivalent to $\tilde{A} \mapsto 0$ as $r \mapsto \infty$.

If the geometry of the region is planar or axisymmetric, and if the known current acts in and is invariant with respect to the direction perpendicular to the plane of the field, then (A.1) - (A.6) reduce to scalar equations in terms of the out-of-plane component of the magnetic vector potential. For instance, in the axisymmetric case, with a circumferential current distribution that is independent of θ, the only nonvanishing component of the potential is A_θ (which is also independent of θ).

For the remainder of this chapter, the axisymmetric problem is treated. Furthermore, the convention that $\sigma = 0$ when $J_\theta \neq 0$, and vice versa, is adopted for convenience. Equations (A.1) - (A.6) written in terms of $\tilde{A} = (0, A_\theta(r,z,t), 0)$ become:

$$\sigma \dot{A}_\theta - \left[\frac{\partial}{\partial r}\left(\frac{1}{\mu r}\frac{\partial(rA_\theta)}{\partial r}\right) + \frac{\partial}{\partial z}\left(\frac{1}{\mu r}\frac{\partial(rA_\theta)}{\partial z}\right)\right] = J_\theta \quad \text{in } \Omega_i \times (0,T) \qquad (A.7)$$

$$n_r\left[\left[\frac{1}{r\mu}\frac{\partial(rA_\theta)}{\partial r}\right]\right] + n_z\left[\left[\frac{1}{r\mu}\frac{\partial(rA_\theta)}{\partial z}\right]\right] = 0 \quad \text{on } \partial\Omega_i \times (0,T) \qquad (A.8)$$

$$A_\theta(r,z,0) = 0 \quad \text{in } \Omega \qquad (A.9)$$

$$A_\theta(r,z,t) = 0 \quad \text{on } \partial\Omega \times (0,T). \qquad (A.10)$$

Since A_θ is independent of θ, the constraint that $\tilde{\nabla} \cdot \tilde{A} = 0$ is identically satisfied, and the vector identity,

$$\tilde{\nabla} \times \tilde{\nabla} \times \tilde{A} = \tilde{\nabla}(\tilde{\nabla} \cdot \tilde{A}) - (\tilde{\nabla} \cdot \tilde{\nabla})\tilde{A} \qquad (A.11)$$

is used to obtain (A.7).

A.2 Finite Element Equations

The variational form of the classical initial/boundary value problem given by (A.7) - (A.10) and the set of finite element equations are obtained using a Galerkin technique. A discussion of this method can be found in [2,3]. A scalar test function u(r,z) that satisfies the essential boundary conditions on A_θ multiplies equations (A.7) and (A.8), and the result is integrated to yield:

$$\sum_i \int\int_{\Omega_i} \left[\sigma \dot{A}_\theta - \frac{\partial}{\partial r}\left(\frac{1}{r\mu}\frac{\partial(rA_\theta)}{\partial r}\right) - \frac{\partial}{\partial z}\left(\frac{1}{r\mu}\frac{\partial(rA_\theta)}{\partial z}\right) - J_\theta\right] u \, r dr dz$$

$$+ \oint_{\partial\Omega_i} \left\{ n_r \left[\left[\frac{1}{r\mu}\frac{\partial(rA_\theta)}{\partial r}\right]\right] + n_z \left[\left[\frac{1}{r\mu}\frac{\partial(rA_\theta)}{\partial z}\right]\right] u \right\} ds = 0 \qquad (A.12)$$

An integration by parts of the second and third terms in the first integral yields:

$$\sum_i \left\{ \int\int_{\Omega_i} \left[\sigma \dot{A}_\theta u + \frac{1}{\mu(|\tilde{B}|)}\left(\frac{1}{r^2}\frac{\partial(rA_\theta)}{\partial r}\frac{\partial(ru)}{\partial r} + \frac{\partial A_\theta}{\partial z}\frac{\partial u}{\partial z}\right) - J_\theta u\right] r dr dz \right\} = 0. \qquad (A.13)$$

It can be seen that the boundary integral from the integration by parts equilibrates the weak form of the material interface jump conditions. Therefore, the jump conditions are satisfied in an average sense without any further attention! Equation (A.13) must hold for any function u(r,z) satisfying the essential boundary conditions.

The entire region, Ω, is divided into a number of simple subdomains or elements, Ω_e, $e = 1, 2, \ldots, E$. In each element a number of points called nodal points are identified. These points are usually chosen to be on interelement boundaries. Over each Ω_e, the unknown potential, A_θ, and the test function, u, are approximated by polynomials in terms of

$$A_\theta(r,z,t) = \sum_{i=1}^{n} N_i(r,z)\, A_e^i(t)$$

and

$$u(r,z) = \sum_{i=1}^{n} N_i(r,z)\, u_e^i \qquad (A.14)$$

where $A_e^i(t)$ and u_e^i are the unknown values of A_θ and u at the nodal points in the element, and where the N_is are the interpolating polynomials [4].

A substitution of (A.14) into (A.13) written as a sum on e yields:

$$\sum_{e=1}^{E} u_e^i \int\int_{\Omega_e} \sigma N_i N_j \, r dr dz \dot{A}_e^j +$$

$$\sum_{e=1}^{E} u_e^i \int\int_{\Omega_e} \frac{1}{\mu(|\tilde{B}|)} \left[\frac{\partial N_i}{\partial r}\frac{\partial N_j}{\partial r} + \frac{1}{r}\left(N_i \frac{\partial N_j}{\partial r} + N_j \frac{\partial N_i}{\partial r} \right) + \frac{1}{r^2} N_i N_j + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right] r dr dz A_e^j$$

$$- \sum_{e=1}^{E} u_e^i \int\int_{\Omega_e} J N_i \, r dr dz \; = \; 0. \qquad (A.15)$$

The unknown nodal point values of A_θ and u are taken outside the integrals since they are no longer functions of position. Equation (A.15) must vanish for each u_e^i since the function u is arbitrary. The result can be written in matrix form as a sum on terms.

$$\underline{\underline{C}}^e \underline{\dot{A}}_e \; + \; \underline{\underline{K}}^e(|\tilde{B}|) \, \underline{A}_e \; = \; \underline{J}^e \qquad (A.16)$$

where

$$C_{ij}^e \; = \; \int\int_{\Omega_e} \sigma N_i N_j r dr dz$$

$$K_{ij}^e \; = \; \int\int_{\Omega_e} \frac{1}{\mu(|\tilde{B}|)} \left[\frac{\partial N_i}{\partial r}\frac{\partial N_j}{\partial r} + \frac{1}{r}\left(N_i \frac{\partial N_j}{\partial r} + N_j \frac{\partial N_i}{\partial r} \right) + \frac{1}{r^2} N_i N_j + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right] r dr dz$$

$$J_i^e \; = \; \int\int_{\Omega_e} J_\theta \, N_i \, r dr dz$$

for $i, j \; = \; 1, \; 2, \; \ldots, \; n$. For the linear problem, μ = constant, the integrals can be evaluated - **MAP** uses Gauss-Legendre quadrature to integrate the terms numerically. The nonlinear problem is discussed below. Once these element equations are formed, they are assembled, as implied by the summation, into a set of global, nonlinear, first order ordinary differential equations.

$$\underline{\underline{C}}\underline{\dot{A}} \; + \; \underline{\underline{K}}(|\tilde{B}|)\underline{A} \; = \; \underline{J}. \qquad (A.17)$$

The time derivative term can be treated by any one of the many numerical integrators such as Runge-Kutta or one of the time marching algorithms such as Crank-Nicholson. **MAP** employs a time marching scheme that arises from approximating the time derivative term as:

$$\dot{\underline{A}} = \frac{1}{\Delta t}[\underline{A}(t + \Delta t) - A(t)] = \frac{1}{\Delta t}(\underline{A}^{n+1} - \underline{A}^{n}) \qquad (A.18)$$

where Δt is the time step increment and n is the time step number, *i.e.*, $t = n\Delta t$. Substitution of (A.18) into (A.17) yields the following general difference form:

$$\left[\frac{1}{\Delta t}\underline{\underline{C}} + (1-\theta)\underline{\underline{K}}(\underline{A}^{n+1})\right]\underline{A}^{n+1} = \left[\frac{1}{\Delta t}\underline{\underline{C}} - \theta\underline{\underline{K}}(\underline{A}^{n})\right]\underline{A}^{n} + (1-\theta)\underline{J}^{n+1} + \theta\underline{J}^{n}. \quad (A.19)$$

The factor θ, $0 \leq \theta \leq 1$, is a constant that determines the type of time marching to be used. If $\theta = 0$, a forward difference form of (A.17) results, and $\theta = \frac{1}{2}$ yields a Crank-Nicholson form. The term $\underline{\underline{K}}(\underline{A}^{n+1})$ implies

$$\underline{\underline{K}}(\underline{A}^{n+1}) = \underline{\underline{K}}(|\tilde{\nabla} \times \tilde{A}^{n+1}|) = \underline{\underline{K}}(|\tilde{B}|) = \underline{\underline{K}}\left(\frac{1}{\mu(|\tilde{B}|)}\right).$$

The material nonlinearity can be treated by one of the iterative techniques such as successive substitution or Newton-Raphson iteration. **MAP** uses a Newton-Raphson scheme. The scheme is applied to the set of element equations in difference form. If Equation (A.19) were written as:

$$\underline{F}^{e}(\underline{A}_{e}^{n+1}) = \underline{0}, \qquad (A.20)$$

then the Newton-Raphson scheme consists of finding in the (i+1) iterate $\Delta\underline{A}_{i+1}^{n+1}$ such that

$$\underline{F}^{e}(\underline{A}_{i}^{n+1} + \Delta\underline{A}_{i+1}^{n+1}) = \underline{0}, \qquad (A.21)$$

A Taylor expansion, truncated after the first order term yields:

$$\frac{\partial F^{e}(\underline{A}^{n+1})}{\partial \underline{A}^{n+1}} \Big|_{\underline{A}_{i}^{n+1}} \Delta\underline{A}_{i+1}^{n+1} = -F^{e}(\underline{A}_{i}^{n+1}). \qquad (A.22)$$

Evaluating A.22 yields:

$$\left[\frac{1}{\Delta t}\underline{\underline{C}} + (1-\theta)\left\{\underline{\underline{K}}(\underline{A}_i^{n+1}) + \underline{\underline{K}}'(\underline{A}_i^{n+1})\right\}\right] \Delta\underline{A}_{i+1}^{n+1} = \left[\frac{1}{\Delta t}\underline{\underline{C}} - \theta\underline{\underline{K}}(\underline{A}^n)\right] \underline{A}^n$$

$$+ (1-\theta)\underline{J}^{n+1} + \theta J^n - \left[\frac{1}{\Delta t}\underline{\underline{C}} + (1-\theta)\underline{\underline{K}}(\underline{A}_i^{n+1})\right] \underline{A}_i^{n+1} \qquad (A.23)$$

where the e-superscript has been omitted for convenience. The term $\underline{\underline{K}}'(\underline{A}^{n+1})$ is calculated according to :

$$\underline{\underline{K}}'(\underline{A}_i^{n+1}) \equiv \frac{\partial\underline{\underline{K}}(\underline{A}^{n+1})}{\partial\underline{A}^{n+1}}\Big|_{\underline{A}_i^{n+1}\underline{A}_i^{n+1}} = \int\int_{\Omega_e} \underline{\underline{k}}\frac{\partial}{\partial\underline{A}^{n+1}}\left[\frac{1}{\mu(|\tilde{B}|)}\right] r dr dz \Big|_{\underline{A}_i^{n+1}\underline{A}_i^{n+1}} \qquad (A.24)$$

where $\underline{\underline{k}}$ is the matrix formed by the products of the polynomials $\underline{N}$ and its derivatives. Using the chain rule

$$\frac{\partial}{\partial A^{n+1}}\left[\frac{1}{\mu(|\tilde{B}|)}\right] = \frac{\partial}{\partial|\tilde{B}|^2}\left[\frac{1}{\mu(|\tilde{B}|)}\right]\frac{\partial|\tilde{B}|^2}{\partial\underline{A}^{n+1}}.$$

However

$$|\tilde{B}|^2 = (\tilde{\nabla}\times\tilde{A})\cdot(\tilde{\nabla}\times\tilde{A}) = \left(\frac{1}{r}\frac{\partial(rA_\theta)}{\partial r}\right)^2 + \left(\frac{\partial A_\theta}{\partial z}\right)^2$$

and, in light of (A.15)

$$|\tilde{B}|^2 = \underline{A}^{n+1^T}\,\underline{\underline{k}}\,\underline{A}^{n+1}. \qquad (A.25)$$

Finally,

$$\frac{\partial}{\partial\underline{A}^{n+1}}\left[\frac{1}{\mu(vert\tilde{B}|)}\right] = \nu'(|\tilde{B}|^2)\,2\left(\underline{\underline{k}}\underline{A}^{n+1}\right)^T$$

and

$$\underline{\underline{K}}'(\underline{A}_i^{n+1}) = \int\int_{\Omega_e} 2\nu'(|\tilde{B}|^2)\left(\underline{\underline{k}}\underline{A}_i^{n+1}\right)^T\left(\underline{\underline{k}}\underline{A}_i^{n+1}\right) r dr dz. \qquad (A.26)$$

MAP employs a pointwise BH curve used to determine ν ($|\tilde{B}|$). Both ν and ν' are found by linear interpolation.

Using equation (A.25) and the pointwise BH data, the element equations (A.23) can be integrated (numerically) and the results assembled into a form that can be written:

$$\underline{\underline{T}}(\underline{A}_i^{n+1})\Delta\underline{A}_{i+1}^{n+1} = -\underline{F}(\underline{A}_i^{n+1}). \qquad (A.27)$$

For each time step, equation (A.27) is formed and solved and $\underline{A}_{i+1}^{n+1} = \underline{A}_i^{n+1} + \Delta\underline{A}_{i+1}^{n+1}$ is calculated. The iterative procedure is continued until convergence is achieved.

A.3 Calculation of Output Quantities

The postprocessing routines in **MAP** calculate a number of secondary quantities derived from the nodal (single component) vector potential data that result from the solution of matrix equations. These secondary quantities are:

$$\tilde{B} = \tilde{\nabla} \times \tilde{A} \qquad (A.28)$$

$$\tilde{H} = \frac{1}{\mu(|B|)}\tilde{B} \qquad (A.29)$$

$$J = J_0 - \sigma(\dot{A} + \phi/r) \qquad (A.30)$$

where ϕ is the "gap voltage" arising from the zero-net current condition. In an planar problem, the last term in (A.30) is not divided by r. Other terms are then calculated according to:

$$Area = \int\int drdz \qquad (A.31)$$

$$NI = \int\int J\ dr\ dz \qquad (A.32)$$

element forces:

$$F_r = \int\int JB_z \; dr \; dz \tag{A.33}$$

$$F_z = -\int\int JB_r \; dr \; dz \tag{A.34}$$

energy stored and dissipated:

$$E_{stored} = 2\pi \int\int \frac{1}{\mu(\|B\|)} \tilde{B} \cdot \tilde{B} \; r \; dr \; dz \tag{A.35}$$

$$POWER = 2\pi \int\int \frac{1}{\sigma} J^2 \; r \; dr \; dz \tag{A.36}$$

$$TOTLOSS = \int E_{dissipated} dt \tag{A.37}$$

$$\tilde{M} = \frac{1}{\mu_0} \tilde{B} \; - \; \tilde{H} \tag{A.37}$$

Total component or material forces are output on a per radian basis.

APPENDIX B: MAP DISK FILES AND PARAMETERS

B.1 MAP Disk Files

UNIT	VARIABLE	INFORMATION
5	nin	terminal input
6	nout	terminal output
15	nin	file input – not decoded
16	nout	file output
81	ntape1,nin	decoded data input (unit renamed after FFLDSB)
82	ntape2	t,iunit – for each time point nc,rr,zz,fr,fz,dum,ft,NI – element by element forces
83	ntape3	property, grid, and element data defined below.
84	ntape4	nodal point flux and current densities at each time defined below
85	ntape5	nodal (single component) vector potential defined below
86	ntape6	t,iunit – for each time point nc,rr,zz,mr,mz,dum,mt – element by element magnetization
87	ntape7	t, energy, totni, power, matni, matfr, matfz
88	ntape8	unused
89	ntape9	unused
90	ntape10	unused

The vast majority of data written by and/or read by **MAP** is contained in three primary files, 83, 84, and 85. The FORTRAN routines to read these files are called READ3, READ4, and READ5, respectively. They are reproduced below. A user may write an interface routine such as a pre- or postprocessor using these routines.

These routines are lifted "as is" and require MAPCOMMONS.FOR (described below). In addition, these same routines are used in **MAP2DJ**. The flag I1D2D is used to differentiate between the two. I1D2D = 2 flags the problem as a two-dimensional field and **MAP** is used.

B.1.1 READ3

The first routine, READ3, is used to read in previously stored problem, property, grid, and element data. This file is written after SETUP and is read in RESTART.

```
      subroutine read3
      include 'mapcommons.for'
      data sub/'READ3'/
      rewind ntape3
      read(ntape3,end=999) hed,unit,i1d2d,axisym,itemp,wdth,ntf,
     &                     nturns,totfac
      read(ntape3,end=999) numel,maxeqn,imx,jmx,nmat,nmix,
     &   ((prop(i,j),tprop(i,j),itflag(i,j),i=1,maxprp),j=1,nmat),
     &   (matnme(i),i=1,nmat),
     &   ((xlammix(i,j),gammix(i,j),j=1,3),i=1,nmix),
     &   ((ijk(i,j),i=1,maxc1),(ldes(i,j),i=1,maxdof1),j=1,numel),
     &   ((r(i,j),z(i,j),i=1,imx),j=1,jmx)
c.....    Determine the number of characters in the header.  I forgot to
c.....    save it, and instead of changing the routine WRITE3 and READ3
c.....    and introducing an incompatibility, I'll recalculate it here.
      do 10 i=1,80
        j=81-i
        if(hed(j).ne.' ') go to 11
   10 continue
   11 nchart=j
      return
  999 call error(0,0005,0,0,sub,' ','END OF FILE ON FOR083')
      return
      end
```

B.1.2 READ4

The READ4 routine is written in POST2D each PRNTFRQ time steps or after a converged iterative solution. It contains the nodal point flux density and current density data. The routine is complicated by the fact that the flux density can experience jumps across material interfaces and for completeness the values of the flux density at such interfaces are stored separately. In addition, the option of calculating and storing flux densities at user-selected points (ILVL=5) must be handled.

```
      subroutine read4(ilvl,v,vl,vi,ipt,lst,kpts,kmax)
      include 'mapcommons.for'
      data sub/'READ4'/
      dimension v(maxi,maxj,kmax),ipt(2,maxb),lst(maxb),vl(maxb),
     &          vi(2,maxb)
      if(ilvl.eq.5) then
        read(ntape4,end=999) ll,(vl(k),lst(k),k=1,ll)
        do 10 k=1,ll
          ij=lst(k)
          i=mod(ij,1000)
          j=ij/1000
          v(i,j,1)=vl(k)
   10   continue
      else
        read(ntape4,end=999) ((v(i,j,1),i=1,imx),j=1,jmx)
        if(ilvl.eq.9) then
          read(ntape4,end=999) kpts
          if(kpts.ne.0) then
            backspace ntape4
            read(ntape4,end=999) kpts,((ipt(i,k),vi(i,k),i=1,2),
     &                                 k=1,kpts)
          end if
          do 30 k=1,kpts
            do 20 l=1,2
              ij=ipt(l,k)
              if(ij.ne.0) then
                i=mod(ij,1000)
                j=ij/1000
                v(i,j,l+1)=vi(l,k)
              end if
   20       continue
   30     continue
        end if
      end if
      return
  999 call error(0,0006,0,0,sub,' ','END OF FILE ON FOR084')
      return
      end
```

B.1.3 READ5

The final routine is READ5; the vector potential at each node is written every PRNTFRQ time steps by POST2D. The correspondence between the equation order vector A and nodal points is contained in the destination vector LDES (read in READ3).

```
      subroutine read5(ieof)
      include 'mapcommons.for'
      data sub/'READ5'/
c.....
      ieof=0
      if(i1d2d.ne.3) then
        if(itemp.eq.0) then
          read(ntape5,end=999) t,maxeqn,(a(i),i=1,maxeqn)
        else
          read(ntape5,end=999) t,maxeqn,(a(i),temp(i),i=1,maxeqn)
        end if
      else
        read(ntape5,end=999) t,maxeqn,(temp(i),i=1,maxeqn)
      end if
      return
  999 ieof=1
      return
      end
```

B.2 MAP Parameters

maxa	8500	Max. # of unknowns – including midside nodes and constraints.
maxb	100	Max. # of user defined (i,j)s for printout.
maxc	4	Max. # of corner nodes in an element (FIXED).
maxcon	20	Max. # of constraint equations – nonet current regions.
maxc1		maxc+1
maxct2		2*maxc
maxct3		3*maxc
maxdof	8	Max. # of vector potential degrees of freedom in an element.
maxdof1		maxdof+1
maxdof2		maxdof+2
maxel	2750	Max. # of elements.
maxi	120	Max. # of is.
maxj	60	Max. # of js.
maxi4		4*maxi
maxj4		4*maxj
maxint	9	Max. # of integration points (FIXED).
maxmat	20	Max. # of materials.
maxmix	5	Max. # of mixtures (**MAP2DJ** only).
maxpf	20	Max. # of PF coils (**MAP2DJ** only).
maxprp	6	Max. # of electromagnetic properties for each material.
maxprt	100	Max. # of grid parts.
maxs	950000	Max. Skyline storage.
maxst		maxdof1*maxdof/2
mxchars	20	Max. # of characters per word.
mxnmes	5	Max. # of alphanumeric words per line.
mxprln	80	Max. # of characters per line.
mxsrfbc	100	Max. # of surface boundary conditions (Thermal portion of **MAP2DJ** only).
mxvals	20	Max. # of numeric values per line.

```
      common /bc      / bc(maxa),tbc(maxa),abc(maxa),bbc(maxa),
     &                  lstsrf(maxc,maxel),surfbc(maxc,mxsrfbc)
      common /chars   / matnme(maxmat),hed(81),iunit,header,type,
     &                  vsupply
      common /cnstnt  / pi,rad,xmu0,unit,nchart,bcdum,zero,idate(6)
      common /coildat/ ntf,nturns,wdth
      common /emprop  / prop(maxprp,maxmat)
      common /elmdat  / mat,kind,nc,ndof,ip(maxc),jp(maxc),rr(maxc1),
     &                  zz(maxc1),lldes(maxdof),ebc(maxdof),aa(maxdof),
     &                  aan(maxdof),aai(maxdof),tt(maxdof)
      common /flags   / axisym,resolv,timebc,ivec,interm,outterm,
     &                  first,jrestrt,itemp,i1d2d,iplot,totfac,
     &                  voltage,rise,flat,down,rwnd
      common /iodata  / data(mxvals)
      common /ionames/ name(mxnmes)
      common /ijkrz   / ijk(maxc1,maxel),r(maxi,maxj),z(maxi,maxj),
     &                  ldes(maxdof2,maxel)
      common /maxdim  / numel,maxeqn,nmat,imx,jmx,neqint,ntie,ncn
      common /mixture/ nmix,xlammix(maxmix,3),gammix(maxmix,3)
      common /nuclear/ qnom,rp,ap,delvv,xftest,xf1,xf2,q(maxel)
      common /rhs     / a(maxa),an(maxa),ai(maxa)
      common /shapef  / sm(maxc,maxint),sn(maxdof,maxint),w(maxint),
     &                  dmdl(maxc,2,maxint),dndl(maxdof,2,maxint),
     &                  dndx(maxdof,2),dmdx(maxc,2),ff(maxdof),
     &                  extrap(4,4),cc(maxdof,maxdof),ss(maxdof,maxdof),
     &                  sspr(maxdof,maxdof),ldl(maxdof2)
      common /source  / vo,xnio,psresis,tflat,tdown,xni,risemax,teoru,
     &                  teoft,teord,xieof,voltavl,voltreq,volt,xnilst,
     &                  psedis,psdata(20),xltmdt,deltrup,deltflt,deltdwn
      common /tapes   / nin,nout,ntape1,ntape2,ntape3,ntape4,ntape5,
     &                  ntape6,ntape7,ntape8,ntape9,ntape10,ltape1
      common /thprop  / tprop(maxprp,maxmat),itflag(maxprp,maxmat)
      common /totals  / totni,storeng,pwrres,totloss,totnuc,
     &                  eloss(maxmat),qnuclst(maxmat),rhojlst(maxmat),
     &                  ennuc(maxmat)
      common /trhs    / temp(maxa),tempn(maxa)
      common /work    / tfinal,delt,thet,t,iter
c.....
      logical axisym,resolv,timebc,interm,outterm,ivec,first,
     &        voltage,rise,flat,down,rwnd
      character*1 hed
      character*20 name,matnme,iunit,sub,vsupply,type
      character*120 header
      data ldl/0,1,3,6,10,15,21,28,36,45/
```

APPENDIX C: MAP COMMAND SUMMARY

```
(*) $ Title - 80 characters maximum

(*) PLANE

(*) UNITS,  M (or MM, CM, or IN)

(*) PROBLEM,FIELD2D,sysfact

(*) SETUP,iprint

Material Property Definition

   (*)  MATERIAL NAME,n,j0,sigma,iron fraction,nonet,ij

   (*)  END,MATERIALS

Grid Part Definition

   (*)   imin,jmin,imax,jmax,g1,g2,g3,g4,POLAR,ro,zo
   (*)   r1,r2,r3,r4,....,r12 or x1,x2,etc.
   (*)   z1,z2,z3,z4,....,z12 or y1,y2,etc. or theta1,theta2,etc
   (*)   REGULAR,imin,jmin,imax,jmax,g1,g2,g3,g4,POLAR,ro,zo
   (*)   rmin,zmin,rmax,zmax
   (*)   IREPEAT,n1,n2,jmin,jmax,z1,z2,z3,z4,g2,g4
   (*)   JREPEAT,n1,n2,imin,imax,r1,r2,r3,r4,g1,g3
   (*)   POINT,i,j,r,z
   (*)   FILL,i1,j1,i2,j2

   (*)   END,GRID

Element Definition

   (*)   ILOOP,npass,inc
   (*)   JLOOP,npass,inc
   (*)   ELEMENT TYPE,mat,i1,j1,i2,j2,i3,j3,i4,j4
   (*)   MAT,material number,i1,j1
   (*)   JEND
   (*)   IEND

   (*)   END,ELEMENTS

(*) FIELD2D,TYPE,itmax,tolit,prntlvl,tfinal,delt,tolss,theta,ipltlvl
       prntfrq,pltfrq,modit,rlx,monit,inode,jnode,orlx,urlx,mstart

Boundary Condition Definition

   (*)   INITIALA,mat,value
   (*)   A,nel1,nel2,ninc,iside,v(t1),(v(t2),t1,t2)
   (*)   NA,i1,j1,i2,j2,v(t1),(v(t2),t1,t2)

   (*)   END,BC
```

Plotting Commands

(*) PLOT,TYPE,PLTTYPE,rmin,rmax,zmin,zmax,(itckr,itckz,nelov,
tstart,tstop,scale,nelno,irot,TITLE)

where TYPE = ELEMENTS, CONTOURS, VECTORS, FUNCTION, TOLERANCE

nelovl < 0 all but material abs(nelovl) drawn or outlined
= 0 no outline
> 0 material nelovl outlined
= 77 all materials outlined
= 99 all elements outlined (like an overlay)

scale = height or element material numbers [80]
= length of VECTORS [0.08]
= factors for labels of CONTOURS [1.0]
= size of squares at interfaces in FUNCTIONS [0.08]

nelno = -1 material numbers written
= 0 no numbers written
= 1 element numbers written
= 2 potential BCs written

Vector PLTTYPES: FIELD, FORCES, MAGNETIZATION

Contour PLTTYPES: FLUX (AX), BR (BX), BZ (BY), B, JTH (JZ), HOMOGENEITY

(*) ncon,(min,delta,iplt,lblfrq,ilbl,htd,ndec,NONREG,PRINT,NODASH)

if dp=0 program computes min, max, and delta for the region

iplt < 0 only first contour numbered once
= 0 no labels
> 0 label every lblfrq contours with labels every
iplt% of the plot width

ilbl = 0 contour values printed
= 1 contour number printed

Function PLTTYPES

System Totals vs Time: STORENERGY, TOTNI, TOTLOSS, POWER
MATNI, MATFR (MATFX), MATFZ (MATFY)

(*) t mat, PRINT

Local Quantities vs Time: FLUX (AZ), BR (BX), BZ (BY), B, JTH (JZ)

(*) t io, jo, PRINT

Local Quantities vs Position: FLUX (AZ), BR (BX), BZ (BY), B, JTH (JZ)

(*) r (x) or i jostart, jofinal, jinc, PRINT, MORE
(*) z (y) or j iostart, iofinal, iinc, PRINT, MORE

```
(*) RESTART,NEXT COMMAND

(*) TRNSPRNT,time,prntlvl,ipltlvl,prntfrq,pltfrq,irew,ALL

(*) POST, ADDFORCE
   (*)  i1,j1,i2,j2  or sum,n1,n2,...
                .
   (*)   END

(*) POST, FINDEQN
   (*)  ELEMENT
       (*)  i1,j1,i2,j2
                .
       (*)   END
   (*) EQUATION
       (*)  neq
                .
       (*)   END
   (*) COORDINATE
       (*)  RZ, r, z or IJ, i,j
                .
       (*)   END

(*) TERMINAL

(*) END, PROBLEM

(*) STOP
```

MAP - C-3

This page lift blank intentionally

REFERENCES

[1] Pillsbury, R.D., "A Finite Element Method for the Analysis of Electromagnetic Field Penetration Problems," Ph.D. Dissertation, The University of Texas at Austin, December 1976.

[2] Strang, G., and Fix, G.J., *An Analysis of the Finite Element Method*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

[3] Oden J.T., and Reddy J.N., *An Introduction to the Mathematical Theory of Finite Elements*, John Wiley and Sons, New York, 1976.

[4] Zienkiewicz O.C., *The Finite Element Method in Engineering Science*, McGraw-Hill, London, 1971.

[5] Philipa C.A."Solution of Linear Equations With Skyline-Stored Symmetric Matrix",*Computers and Structures*,Vol 5, Pergamon Press; 1975.

[6] Hinton, E., and Campbell, J.S., "Local and Global Smoothing of Discontinuous Finite Element Functions Using a Least Squares Method," *International Journal for Numerical Methods in Engineering*, Vol. 8, 1974.