Title
A NEW TRIM MAGNETOSTATIC COMPUTER PROGRAM FOR THE VAX/VMS ENVIRONMENT

Authors
Colonias, J.S.
Kogaway, P.

Publication Date
1985-10-01

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

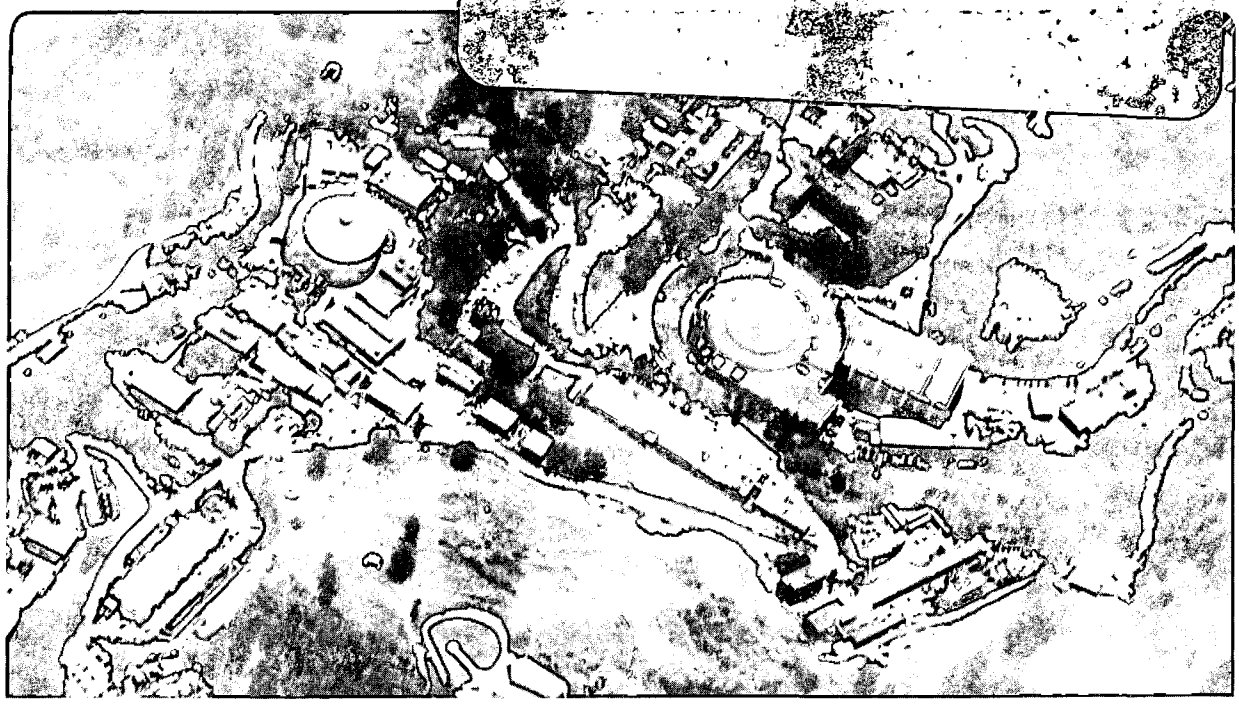
Computing Division

To be submitted for publication

A NEW TRIM MAGNETOSTATIC COMPUTER PROGRAM FOR THE VAX/VMS ENVIRONMENT

J.S. Colonias and P. Rogaway

October 1985



Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

DISCLAIMER

A New TRIM Magnetostatic Computer Program for the VAX/VMS Environment

John S. Colonias
Phillip Rogaway

Computing Division
University of California
Lawrence Berkeley Laboratory
Berkeley, California 94720

October 1985

This research was supported by the United States Department of Energy under contract DE-AC03-76SF00098.

CONTENTS

A New TRIM Magnetostatic Computer Program for the VAX/VMS Environment

John Colonias
Phillip Rogaway

Lawrence Berkeley Laboratory

ABSTRACT

This document describes a new and updated version of TRIM, a collection of computer programs that does magnetostatic simulations in two dimensions. Each of the TRIM programs has been modified somewhat, and a completely new program has been added. The modifications result in a slightly different format for TRIM-input files. The new program, named DIAGRAM, provides a user-friendly interface into TRIM. It allows the user to interactively specify a magnet's geometry, and from this description alone, it produces an irregular triangular mesh. This greatly simplifies running a magnet with TRIM.

The purpose of this paper is to describe DIAGRAM and document the expected format for TRIM input files.

1. Introduction

The VAX version of TRIM (henceforth TRIM) consists of five separately executable pieces: DIAGRAM, MESH, FIELD, TRIP, and POTPLOT. Collectively, these programs are used to do magnetostatic simulations in two dimensions. Briefly, MESH is used to generate an irregular triangular mesh, FIELD solves the non-linear Poisson equation using the mesh that MESH has generated, and TRIP provides a graphical display of the results of MESH and FIELD. POTPLOT draws curves of the potential or any of its space derivatives as a function of distance. MESH, FIELD, and TRIP require specially formatted input files: MESH.INP, FIELD.INP, and TRIP.INP, respectively. These input files are produced using your favorite VMS-supported text editor, except that MESH.INP may also be produced by running DIAGRAM. In fact this is the purpose of DIAGRAM-- to provide a user-friendly method of producing MESH input files.

In this document, we describe the exact syntax of these input files (which has changed since previous publications), and discuss DIAGRAM and POTPLOT, which are entirely new additions to TRIM. POTPLOT is a small and simple program of which there will be little discussion. The description of it will be self-contained. The description of DIAGRAM, too, is intended to be self-contained, but users new to TRIM may wish to consult some of the references listed at the end of this document for information on MESH, FIELD, and TRIP.

The organization and file structure of TRIM is diagramed in Figure 6.

2. Motivation for Developing DIAGRAM

MESH requires the user to have constructed not only a 'physical diagram'-- a description of the magnet's geometry-- but also a 'logical diagram,' the purpose of which is somewhat more obscure to the casual user. The logical diagram resembles the physical, except it is drawn on graph paper of equilateral triangles ('isometric graph paper'),

the boundaries of regions lying exclusively on the lines of this paper. We will call grid points on this diagram 'logical points,' and grid lines, 'logical lines.' The logical diagram is normally somewhat distorted, in the sense that a region in the logical diagram may be larger or smaller than the corresponding region in the physical diagram. (See Figures 1 and 2. Note: these figures do not represent real magnets; they are only meant to be illustrative.)

One can visualize what MESH does by imagining the logical lines as very tight rubber bands, and each logical points as a straight pins. In the logical diagram, the pins have been distributed to form a grid of equilateral triangles. MESH picks up the pins on the boarder of each region and tacks them down to the corresponding point in the physical diagram. The other pins move around freely, trying to minimize the stress they are now under. The result is an irregular triangular mesh.

The question naturally arises: why bother with an irregular mesh? The answer is that since there is only a finite number of points in the mesh, we might as well distribute them where they're going to be most useful. This means using a high density of mesh points in the area of interest, e.g. where a particle beam will pass, and a low density of mesh points in other places, e.g. in the iron of the magnet. In this way we can achieve a better field approximation than would be possible if the same number of points were distributed uniformly. Observe that a region will have a high mesh density if the region has a 'logical area' considerably larger than its 'physical area,' and low mesh density if the region has a logical area considerably smaller than its physical area.

Producing MESH input files has been (in the past) a tiresome undertaking because each and every logical point or line must be specified. This usually involves carefully drawing the logical diagram. A small change in the magnet geometry may require redrawing the whole thing. A curved pole face or slanted line requires every logical point be specified, and the corresponding physical point computed. To eliminate the last difficulty, CONICS, a family of subroutines called from MESH, was created. CONICS allows the user to easily specify segments of lines and arcs of parabolas, ellipses, hyperbolas and circles, thus simplifying user data preparation. CONICS is an important step towards DIAGRAM, and, indeed, its development was motivated by the same issues that led to the development of DIAGRAM.

But one might hope to do better. After all, the need to have a logical diagram is essentially an artifact of the way that numerical solutions to partial differential equations proceed; namely, the need to approximate the continuum with the discrete, i.e., a mesh. The user would like not to deal with the mesh at all, but to simply specify the magnet's physical geometry and have the program compute the magnetic field. Still, the user would like to enjoy the improved accuracy that a distorted mesh admits. DIAGRAM allows the user to do exactly that. It takes a physical diagram, specified interactively, and does everything necessary to produce a high-quality logical diagram, which it then feeds directly to MESH to yield a distorted grid for the simulation. Additionally, DIAGRAM, being interactive and menu-driven, provides a user-friendly interface into TRIM.

The part of this document devoted to DIAGRAM is organized as follows: The following section is a high level description of the program. After that is a low level description, intended for the person who maintains the code, not for the general user. Discussion on DIAGRAM is concluded with a section on the program's limitations. Useful information will also be found in the appendices, including a sample run of the program.

3. High Level Description of DIAGRAM

As you run DIAGRAM, the code maintains a description of the physical diagram you are specifying. When you ask DIAGRAM to make a MESH input file, the following steps are taken (at least conceptually):

1. DIAGRAM computes KMAX and LMAX, the number of grid points in the x- and y-directions, respectively. This is done by requesting a ceiling on the total number of points in the simulation (default: 4000), and finding the 'best' value for KMAX and LMAX that result in no more than this many points in the grid.

2. DIAGRAM 'produces' a 'distorted physical diagram', distorted about the region of interest. This diagram is made by mapping each physical point (x,y) in the universe to the point $g(x,y)$, where g is a rather *ad. hoc.* function inspired by writing down some necessary boundary values, defining it to have a particular functional form, and solving the resulting boundary value problem.

3. We map each point of the distorted physical diagram to the 'closest' approximating point of the logical diagram.

4. We smooth-out the logical diagram that results, elliminating sharp corners that are just a by-product of the mapping method. We do our best to deal with acute angles and other problems that may result in a logical diagram that is not homeomorphic to the physical diagram.

5. We collect the logical points that correspond to each physical region and associate with each logical point a 'good' physical point.

6. We ship all of these points to MESH, which produces an irregular triangular mesh.

The user may control the degree of distortion by a single positive valued real constant, the 'distortion factor.' A value of 1.0 corresponds to no distortion; 2.0 is a fair amount; 3.0 is a lot; 5.0 is the maximum DIAGRAM will allow you to enter. The program defaults to 2.5. The larger the distortion factor, the more likely MESH will be unable to make a distorted mesh. If this happens, MESH issues a message that a triangle of the mesh has negative or zero area. The square of the distortion factor will be the approximate ratio of the density of grid points near the center of interest to the density of grid points far from it.

If you get a negative triangle, first, check the physical diagram for mistakes, and the logical diagram for any obvious irregularities. If there are none, try decreasing the distortion factor, or adding air-rectangles to partition the space surrounding the magnet into a few blocks, which seems to solve the problem in most cases.

4. Low Level Description of DIAGRAM

This section is primarilly intended for the person who maintains DIAGRAM; it is not necessary to read this in order to use the program.

4.1. Organization

DIAGRAM is written in Fortran-77 and runs on a VAX under VMS. DIAGRAM uses DI-3000 graphics routines. It resides on the CSA cluseter at LBL. Source for DIAGRAM is distributed in five files: DIAGRAM.FOR, LOGICAL.FOR, UTILS.FOR, CONICS.FOR, and MESH4DGRM.FOR. Source files are not made available to the general community, but the executable image of DIAGRAM is available for general use.

Though the program is quite large, it is sensibly commented and well abstracted, consisting of very many short subroutines. An effort has been made to isolate the graphics-dependent code to as few routines as possible.

4.2. Data Structures

The main data structure in DIAGRAM is the (real) array W and its companion R (a real two-dimensional array). W is the 'business-end' of things: it is a linear list of the information needed to specify each region, including both 'codes' to specify the arc types, and 'data' to give these arcs physical coordinates. R contains pointers into W and

global information about each region. The representation of the physical diagram as encoded in W was chosen to make providing a graphical output easy; it is not an ideal data structure for manipulating the regions.

R and W contain both real numbers (for coordinates) and number that are properly integers (e.g. R's indices into W). Consequently, DIAGRAM uses a generous number of NINTs to effect the conversion, where NINT is the Fortran 'nearest integer' function.

NINT(R(1,1)) is the number of regions in the diagram. NINT(R(k,1)), k>1, is the index in W at which region k begins. Region 1 always begins at index 1 of W.

A region description in W consists, first, of a code '0.','1.', or '4.' A rectangle is the code '0.', followed by the coordinate of the lower left hand point, followed by the coordinate of the upper right hand point. Since region 1 is always a rectangle (the universe), we see that R(2,1) is always 5.

A polygon is given by the code '1.', followed by k, the (real) value for the number of points in the region. Next are k x-values, followed by the k corresponding y-values.

An arc region is given by the code '4.', followed by NARCS, the number of arcs in this region. Next we have NARCS pointers into W, which point to the beginning of the descriptor for the given arc. The convention is established that the code '4.' specifying that this is an arc region is indexed as 1.

An arc descriptor is a code '0.','1.','2.','3.','4.', or '5.', followed by the data for this arc.

The code '0.' indicates a line segment; it is followed by two ordered pairs of reals, the first giving the tail of the segment, the second, the tip.

The code '1.' indicates an arc of a circle. It is followed by three ordered pairs, and specifies the (unique) conic section that is part of the circle containing these three points. The same scheme is used for '3.' (hyperbolas), '4.' (x-parabolas), and '5.' (y-parabolas).

The code '2.' indicates an arc from an ellipse. It is followed by three points: two points on the ellipse, and the center of the ellipse. It is then followed by A, the length of the semi-major axis, and B, the length of the semi-minor axis. Last, we specify θ, the angle of orientation of the ellipse. Note that some of this information is redundant, and the user will not usually be querried for all of it when he specifies an ellipse-arc.

A summary of this data structure is given in Appendix 5.

4.3. Mathematical Chicanery

To draw arcs of conics, we use a polygon with lots of points. The routine DRWPHY draws the physical diagram by calling DRAW on each region. DRAW employs KCIRCL, KELIPS, KHYPER, KXPARA and KYPARA to assemble the requisite lists of points. These routines perform the dual function of finding a sequence of logical points to approximate a curve and assigning a physical point to each logical point. We obtain the equation of the arc we are drawing from its description in W, and we trace out the shape by incrementing a counter on x,y, or θ. In the case of ellipses, we translate the region and rotate it, as appropriate, using the 2x2 transformation matrix. If our interest is just in drawing the region, we are done, but if we wish to make a list of logical and physical points for the region, we next take each (x,y) on the approximating list of points and map it using our distortion function to its distorted counterpart. From this point we obtain the nearest (K,L) in our logical diagram. We assume that we are using enough approximating points that the (K,L) - images of any two successive points will be the same or neighboring logical points. From this list of logical points, then, we elliminate redundancies and then eliminate the sharp corners that otherwise would give the approximation a saw-tooth appearance. Now knowing the number of grid points in the approximation, we can go back and break the arc cleanly into this many approximating points, using these as the physical values for each logical point.

While an earlier version of DIAGRAM used CONICS to obtain the list of logical and physical points for an arc, the above method has proven more versatile and generates superior results.

An 'ideal' distortion function was never found; what is implemented now works as follows: Suppose the universe is a rectangle $[x_{min}, x_{max}]$ x $[y_{min}, y_{max}]$, with center of interest (o_x, o_y). If we distort independently in the x- and y-dirctions, we may take (o_x, o_y) to be (0,0). and reduce the problem to that of producing a distortion function for the line segment $[0,T]$. We imagine a smooth stretching of the points on this segment without disruption to the endpoints. Thus we seek a function f with $f(0)=0$, and $f(T)=T$. We could imagine the distortion by requiring $f'(0)=s_0$, and $f'(T)=s_T$, where s_0 and s_T are suitably chosen constants. If we simply *assume* that f is a cubic polynomial, then we get a funtion that seems to describe a reasonable distortion. When the user enters a distortion factor, he is entering s_0; we simply take s_T as 1. because it seems to give good results.

In MESH, if a line segment (or any other arc) contains on its interior a point specified in the physical diagram, then this point has to be specified in describing the line segment. For example, the universe can not be specified by giving simply its four corners; instead, each and every point lying along the border needs to be listed. The reason for this apparent duplication is the following: Suppose the arc C contains the point P along its interior, and suppose that P is specified as a point in another region R. Then R gives for P both a physical coordinate, (x,y), and a logical coordinate, (K,L). On the other hand, when MESH interpolates to get a physical value for the (K,L) along the boarder of C, a (generally distinct) physical point will be generated. Thus the logical point (K,L) has two (different) physical points associated with it, which is clearly not desirable!

To rectify this problem, DIAGRAM was made intelligent enough to notice when this situation comes up. Suppose DIAGRAM would like to specify that there is a line segment from the point (x_1,y_1) to the point (x_2,y_2). By examining W, DIAGRAM produces a list of all points in the universe. It now goes through this list and picks out the points that lie on the segment we're interested in. This is easy: the point (x,y) lies on the segment if and only if the distance between (x,y) and (x_1,y_1) plus the distance between (x,y) and (x_2,y_2) is equal to the distance between (x_1,y_1) and (x_2,y_2). Next, DIAGRAM sorts this list of points by distance from the point (x_1,y_1). Then, it removes any duplicate entries on this list. Finally, it generates a call to produce a line segment between successive pairs of points on the list.

This eliminates the need to 'redundantly' specify points lying along line segments. But adding analogous code for other types of arcs seems to be too much work for the prospective benefit. Thus the user must 'redundantly' specify points on arcs other than line segments.

If the techniques to produce a logical diagram were not supplemented by some 'correcting mechanism,' many magnets would give rise to unacceptable logical diagrams. A variety of corrective mechanisms have been tried, but all of them work the same way: when a problem is detected, the logical points that are at fault are 'pushed away' from one another, and the result is smoothed out and cleaned up. While this sounds simple enough, a host of technical difficulties prevents it from being implemented in a very satisfactory way. Consequently, there are some realistic magnet geometries that cannot be accomodated without some sort of user intervention.

5. Limitations of DIAGRAM

As mentioned in the section entitled 'High Level Description,' a high value for the distortion factor may result in negative triangles.

Specifying physical points that are distinct but very close to one another will sometimes cause problems. Two instances in which this is common are when a magnet contains a small 'shim,' a notch in the iron, and when the coils of a magnet are separated from the iron by a very small gap.

We have spent much effort to develop methods to deal with these problems in a user-transparent way, yet nothing seems to work in all cases. The user, at present, has two alternatives. One is to approximate a 'difficult' geometry with a geometry that causes no problems: small shims can be ignored or entered by hand in MESH.INP; if coils are separated by a small distance from the iron yolk, they can be made flush. Another alternative, which we call 'rezoning,' has also been provided. This is a mechanism by which the user tells DIAGRAM to give to certain physical points logical values different from those that would otherwise have been assiged to them. Since this mechanism is undergoing rapid development, the user is refered to the help menu associated with the 'Rezone diagram' selection. Rezoning appears to be necessary to make DIAGRAM truly general, but if a geometry requires extensive rezoning, the advantage of using DIAGRAM over creating MESH.INP by hand is compromised.

It is worth noting that inaccuracies introduced by such approximations as making a coil flush are generally small compared to the inaccuracy introduced by the fact that the magnet is finite in depth.

Alas, for the experienced user of TRIM there are occasionally cases in which MESH input files will have to be prepared by hand. Perhaps the user wishes to have a dozen mesh lines spread out in a gap just a tenth of an inch; perhaps the user needs a very high density of mesh points in one area, but wants a low density quite near. DIAGRAM is intended to make running a 'normal' simulation easy, but user-intervention will always be necessary in the most unusual cases.

6. MESH and its Input, MESH.INP

MESH takes an input file, MESH.INP, describing the 'physical' and 'logical' geometry of a magnet. It produces a distorted mesh, stored in a form only the machine understands in a file named FOR035.DAT. The output of MESH will be the first 'dump' on this file.

MESH.INP may be created 'by hand,' or may be created by running DIAGRAM. If you use exclusively the latter method, then you need not know the contents of this section.

The input to MESH specifies (in this order): (1) A name for the problem you are solving; (2) Values for the problem constants PRBCON (See Appendix 2); (3) A description of the physical and logical diagram.

(1) The problem name is the first line of the input file. It may contain of up to 80 alphanumeric characters.

(2) Next come the PRBCON constants. Specify them by a list of ordered pairs, 'n PRBCON(n)', one ordered pair per line. The first number is the index into the PRBCON array (an integer between 1 and 50); the second number is the value to be assigned to the indicated constant. Enter '0 0' to signify that there are no more PRBCON constants specified. 'n PRBCON(n)' lines in MESH.INP may be annotated by putting a comment to the right of the pair of numbers.

Appendix 2 describes the meaning of each of the PRBCON constants, as well as the TVAR and PBMK constants, which are used in FIELD's input file.

(3) Specify the problem geometry as follows: for each region, list 5 numbers: region number, material, current, current density, and zoning code, as described in references [1] and [2]. Then specify, on successive lines, either ordered 4-tuples, (L,K,Y,X), or tuples $(A_1,A_2,\ldots,A_m)$,$A_1<0$, as described in Appendix 4. The latter notation is for describing conic sections in a convenient manner. When you are done specifying the

region, enter the 4-tupple '0 0 0 0'.

Two comments: First, there's no need to worry about writing 'real' numbers with a decimal point. Second, you may put blank lines anywhere in the file after the first line without ill effects. One might wish to put blank lines between (1) and (2), between (2) and (3), and between successive regions in (3). This helps make the file easier to read.

A sample MESH input file is given in Appendix 1.

7. FIELD and its Input, FIELD.INP

FIELD takes as input a file FIELD.INP, which describes various constants pertinent to the problem. It also takes the file FOR035.DAT that MESH has created. Each time FIELD is run, it drops another 'dump' on FOR035.DAT. If FIELD was able to solve the simulation in the specified amount of time, then this dump represents the solution FIELD arrived at. If FIELD did not have enough time to solve the problem, then the dump FIELD drops represents the approximation it had at the time when time ran out. You will be able to pick up this dump and run FIELD again, starting at this point.

In addition, FIELD will save a synopsis of its run for you in a file named FIELD.OUT. This will include information on the convergence of the program and the magnetic field for the geometry.

FIELD.INP is of the following format: First, on a line by itself, give the number of permeability tables you want to specify. We establish the convention that 0 is a request to use only the built in table; -1 indicates that no table is appropriate, i.e. that this is a constant gamma or infinite mu problem. See PRBCON(4), TVAR(15).

If some permeability tables are to be specified, then they are given next. Enter the number of entries in the table, n, on a line by itself. Then enter $B_1, B_2, \ldots, B_n$, followed by $H_1, H_2, \ldots, H_n$, where the B-values are specified in gauss, the H-values in oersteds. The H-values should begin on a line of their own after the B-values have been completed. Count the number of entries in your table to make sure you haven't missed one or typed one twice.

Next comes the dump number. For example, enter 1 to pick up the dump dropped on FOR035.DAT by MESH.

Next, we enter desired values for the PBMK and TVAR constants. As with MESH's input, these are specified by ordered pairs, the first element an index between 1 and 50, the second element the desired value for the indicated constant. When you've specifed all of the PBMK constants you want to use, enter '0 0' and begin specifying the TVAR constants. When you've specified all of the TVAR constants that you want to use, enter '0 0' once again. A summary of PBMK and TVAR constants is given as Appendix 2.

If this is the only dump you want to pick up, then enter a 0 at this point of the input file and you are done. Otherwise, specify the next dump to pick up, PBMK and TVAR constants as above,and then a 0 to terminate the file or another dump number for FIELD to pick up, etc.

FIELD.INP has a slightly different format for boundary value problems. Following the second '0 0' in the FIELD input (the marker indicating the end of TVAR definitions), enter 4 integers, separated by spaces. These numbers indicate whether or not you are specifying boundary values for the top, bottom, right, and left boundaries (in this order). A '1' or a '2' mean you are specifying boundary values for this side; a '0' indicates that you are not. Corresponding to a code of '1', list the potentials you would like to fix at successive logical points along this side. You will need exactly KMAX entries for a top or bottom boundary, and LMAX entries for a left or right boundary. Corresponding to a code of '2', give a single potential, and it will be used as the potential for every logical point along the indicated side.

Don't forget that you are specifying potentials, not B-values, when running boundary value problems.

Since running boundary value problems requires fixing potentials to logical, not physical, points, knowledge of KMAX, LMAX, and the physical coordinates corresponding to each logical coordinate along the boundary of the diagram is needed. Thus if MESH.INP was created by DIAGRAM and you are running a boundary value problem, you will probably have to examine MESH.INP after running DIAGRAM to glean this information.

A sample FIELD input file is given in Appendix 1.

8. TRIP and its Input, TRIP.INP

The input to TRIP, TRIP.INP, usually consists of one line for each frame you would like plotted. The line consists of eight numbers:

YMIN, YMAX, XMIN, XMAX, SCALE, TSENT, NOL, NDUMP,

specified in the indicated order.

The first four number describe the 'window' into the magnet that you'd like to examine.

SCALE tell the program how big to draw the magnet. It is a number larger than 0 and less than or equal to 1. A scale factor of SCALE means that the plot is to be SCALE times as large as possible. On a 'strip' plotter, such as the Varian, SCALE=1 means that the y-axis is to fill the width of the paper, resulting in a very long plot if the magnet is much longer in the x-direction than the y-direction. SCALE=.33, for example, yields a plot in which the y-axis takes up one third of the available space.

On a graphics display terminal, such as a Tektronix terminal or ADM-3A with Retro-Graphics, the plotting surface will always be chosen so that it fits entirely on the screen. Use SCALE=1 to make the picture as large as possible.

TSENT is 1 if you want the triangles plotted, and 0 otherwise.

NDUMP is the dump that you'd like TRIM to pick up.

If NOL $\geq$ 0, it specifies the number of flux lines you'd like drawn on the plot.

If NOL=-1, then TRIM expects the following lines to contain n, and, on subsequent line(s), $A_1, A_2, \ldots, A_n$. TRIM will plot a flux line at $A = A_i$ for each i in 1,2,...,n.

If NOL=-2, then TRIM expects the next line to contain A_0, δ. TRIM will plot a flux line at $A = A_0 + i * \delta$ for each i in 0,1,2,....

If NOL=-3, then TRIM expects the next line to contain A_0, A_f, δ. TRIM will plot a flux line at $A = A_0 + i * \delta$, where $i = 0,1,\ldots,m$, and m is the greatest positive integer such that $A_0 + m * \delta \leq A_f$.

A sample TRIP input file is given in Appendix 1.

9. POTPLOT

A short program, POTPLOT, has been provided to draw curves of the potential or any of its space derivatives as a function of position. The plot is produced interactively on a Tektronix or Tektronix-compatable terminal. POTPLOT uses the graphics package DISSPLA, and, consequently, can only be run a machine supporting it. POTPLOT picks up the potentials and their derivatives, which FIELD computes after it converges. These are left on the file FOR034.DAT when the user has specified that this file should be written (using PRBCON(44) or PBMK(32)).

POTPLOT plots values as a function of position, where the position is given by a line segment in the physical universe. The program linearly interpolates the potentials or their derivatives for nearby logical points since logical lines may 'bend' with respect to

physical space. POTPLOT is capable of plotting two curves on the same graph.

10. Executing TRIM

To run DIAGRAM, type: RUN DIAGRAM, where DIAGRAM is a logical name pointing to the executable image of DIAGRAM (if it has not been moved to your own directory). If the version of DIAGRAM you are using has been linked for debugging, you may wish to type instead: RUN /NODEBUG DIAGRAM. DIAGRAM produces the file MESH.INP for MESH, and, if requested, automatically runs MESH, producing MESH.OUT and FOR035.DAT.

To run MESH type: RUN MESH (or RUN /NODEBUG MESH), where MESH is a logical name pointing to the executable image of MESH (if it has not been moved to your own directory). MESH produces the files MESH.OUT and FOR035.DAT.

To run FIELD type: RUN FIELD (or RUN /NODEBUG FIELD), where FIELD is a logical name pointing to the executable image of FIELD (if it has not been moved to your own directory). FIELD produces the file FIELD.OUT, and drops another dump on FOR035.DAT. To see what cycle FIELD is on while it is running, you can peak at FIELD.OUT.

Since executing FIELD usually take a long time, it is generally preferable to run FIELD as a batch job ('in the background.') A simple command procedure can be written to do this.

If you want to try to improve the speed at which FIELD converges on multiple runs of related geometries, you can feed FIELD a file named A0.DAT, which is just a renamed FOR035.DAT. This may give FIELD better initial potentials, hastening the problem's convergence. See PRBCON(42) and PRBCON(43), or PBMK(30), and PBMK(31).

To get a TRIP plot while logged onto a Tektronix terminal or and ADM-3A with Retro-Graphics, type: RUN TRIPT (or RUN /NODEBUG TRIPT), where TRIPT is a logical name pointing to the executable image of TRIPT (if it has not been moved to your own directory). TRIPT will pause after each frame to let you have a chance to see it or make a hardcopy. To get the program to continue, just hit <RETURN>.

To get a TRIP plot on the Varian plotter, type: RUN TRIPV (or RUN /NODEBUG TRIPV). Then type 'PLOTSEND' and answer the questions asked of you. The file TRIPV creates is named FOR010.DAT.

To run POTPLOT type: RUN POTPLOT (or RUN /NODEBUG POTPLOT), where POTPLOT is a logical name pointing to the executable image of POTPLOT (if it has not been moved to your own directory). POTPLOT expects the file FOR034.DAT to be available, which is produced by setting the appropriate constant to 1 in MESH.INP or FIELD.INP (PRBCON(44) and PBMK(32), respectively). If you wish to do this 'after the fact,' just pick up the converged dump and set PBMK(32) to 1.

References

[1] Colonias, John S., 'TRIM: A Magnetostatic Computer Program for the CDC 6600,' UCRL-18439, Aug 29, 1968.

[2] ----, Particle Accelerator Design: Computer Programs, Academic Press, New York, 1974.

[3] Winslow, Alan, 'Equipotential Zoning of Two-Dimensional Meshes,' UCRL-7312, June 5, 1963.

[4] ----, 'An Irregular Triangular Mesh Generator,' UCRL-7880, Aug 25, 1964.

[5] ----, 'Numerical Solution of the Quasilinear Poisson Equation in a Nonuniform Triangular Mesh,' J. Comput. Physics 1, 149-1972, 1966.

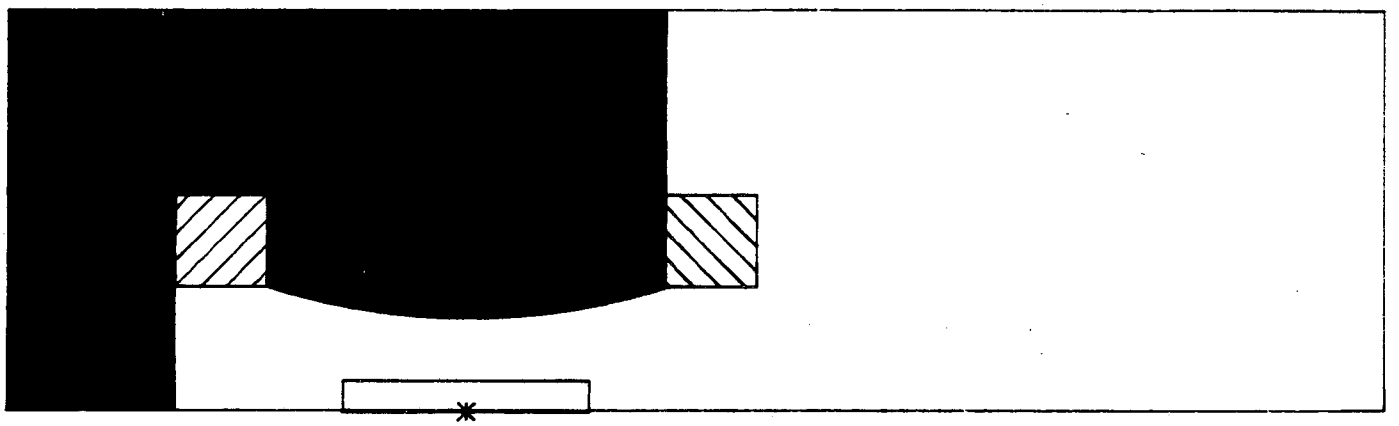


Figure 1: Physical diagram for the C-Magnet produced by DIAGRAM in the sample session of Appendix 3. The asterisk marks the center of interest.

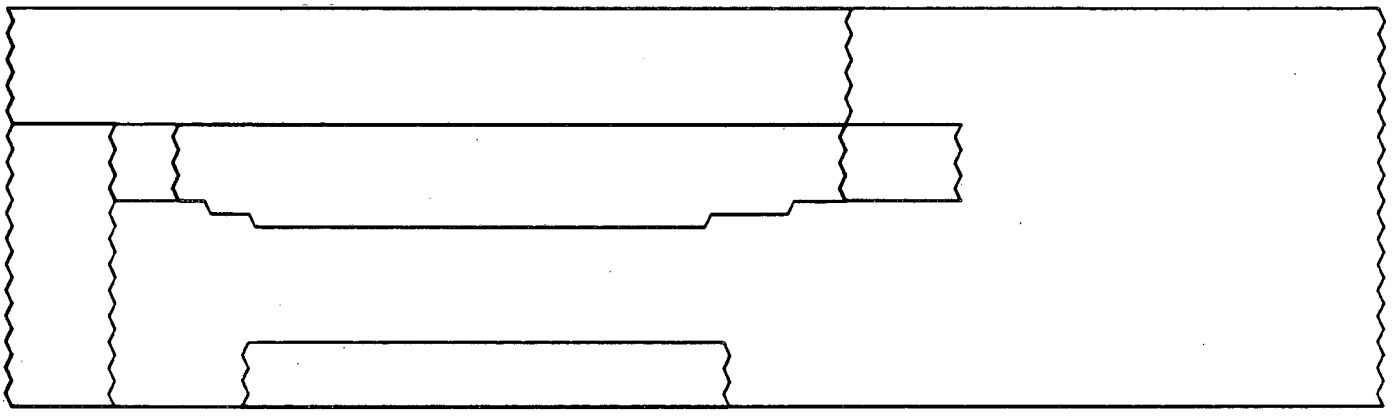


Figure 2: Logical diagram produced in DIAGRAM for the magnet of Figure 1. Here 4000 was selected in response to the question, "Ceiling on total number of points in the simulation?"

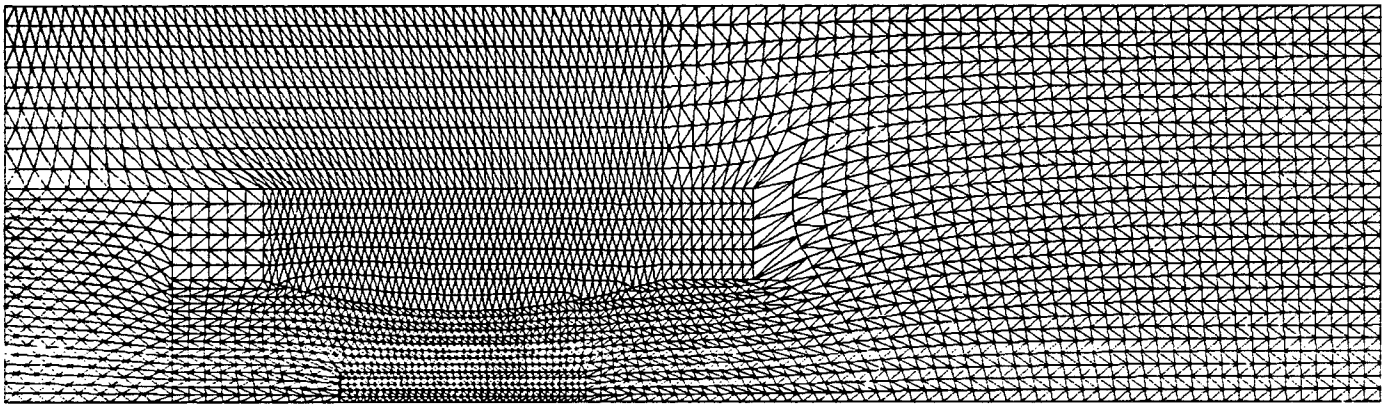


Figure 3: A distorted mesh, produced by MESH from DIAGRAM's MESH.INP. The plot was produced by TRIP.

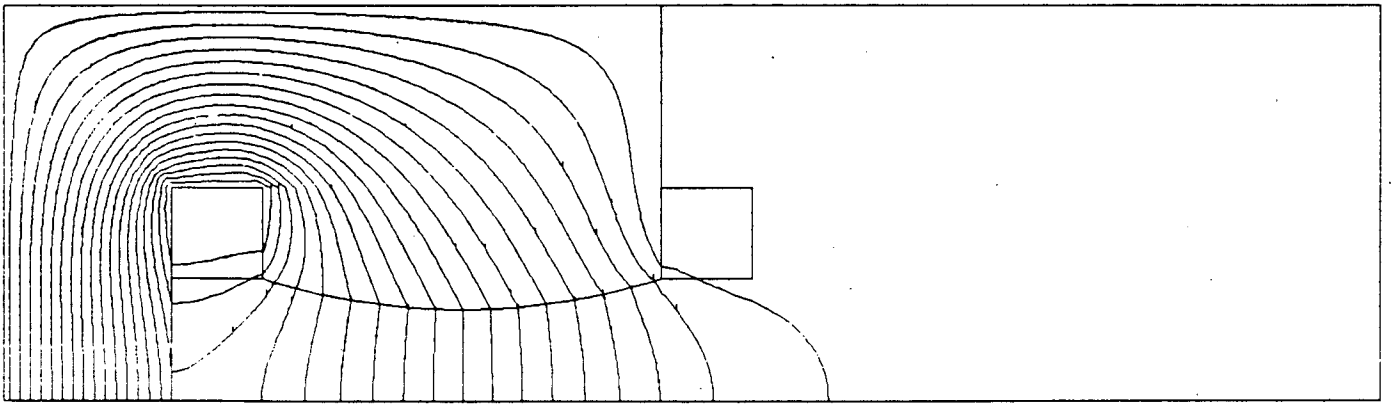


Figure 4: A flux diagram for the magnet of Figure 1, produced by FIELD and plotted by TRIP.

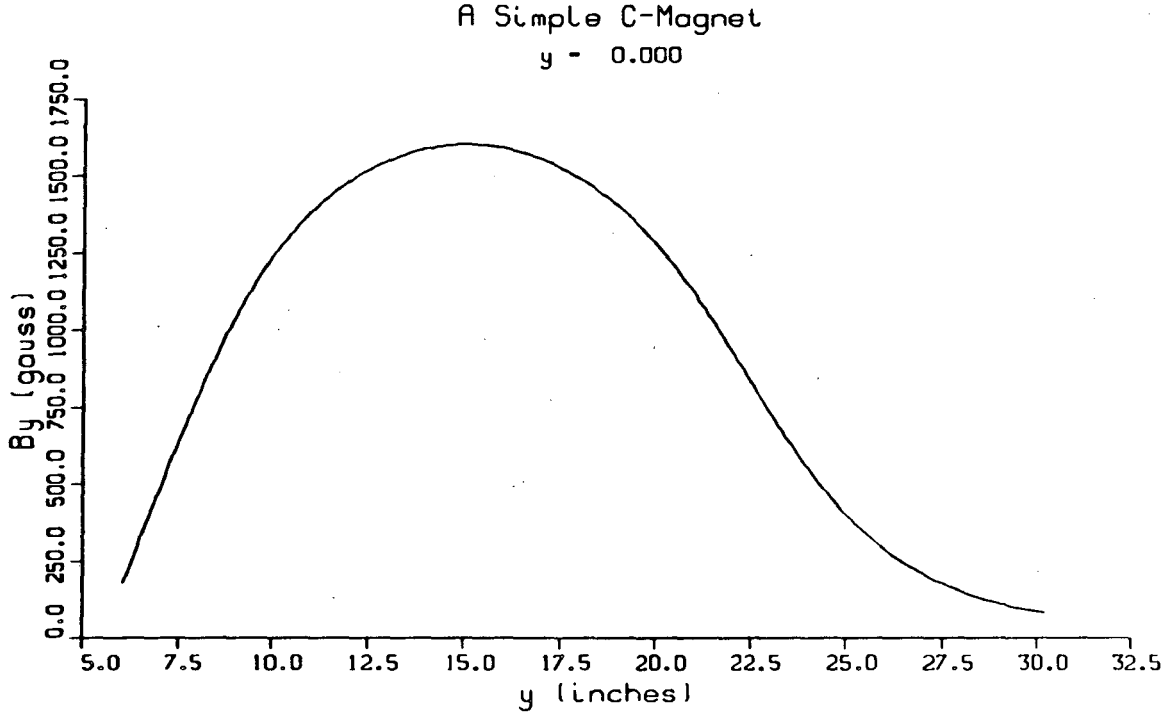


Figure 5: POTPLOT output for the magnet of Figure 1. Here we plot B_y along the median plane.

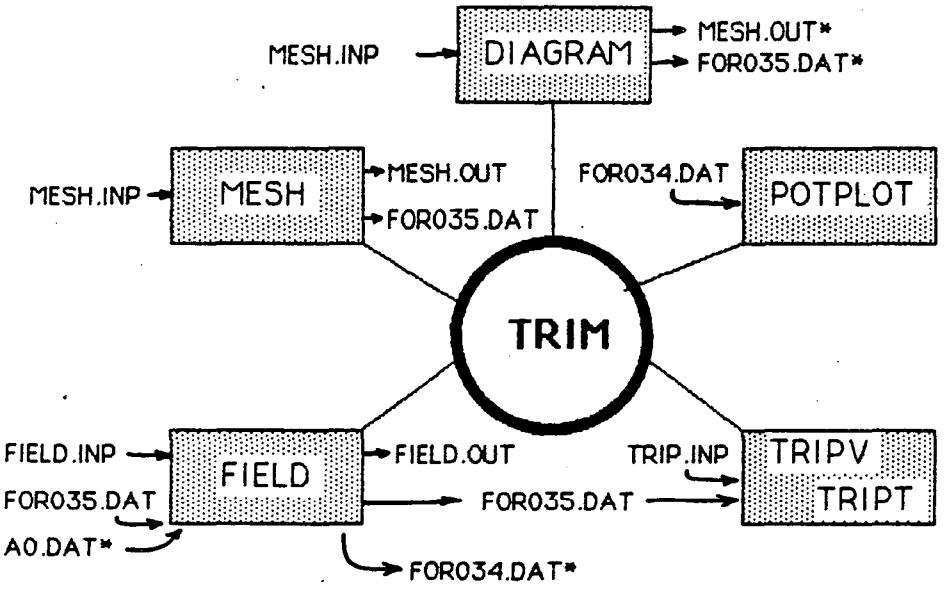


Fig.6 Organization of program TRIM
Executable programs are in boxes. Asterisks mark optional files. Conceptually TRIPV and TRIPT are but one program, therefore they are drawn in the same box

Appendix 1: Sample TRIM Test Cases

MESH input file, MESH.INP, for a sample quadrupole magnet. This magnet uses both (L,K,Y,X)-style point descriptors as well as CONICS codes.

A Sample Quadrupole Magnet

```
1 7      number of regions
2 50     LMAX
3 50     KMAX
6 1      lower reflecting boundary
8 1      left reflecting boundary
0 0

1 1 0 0 2                                   region 1
1 1 0 0
1 23 0 3.675
1 26 0 4.3
1 38 0 7.35
1 47 0 9.825
1 50 0 12
50 50 12 12
50 1 12 0
47 1 9.825 0
38 1 7.35 0
26 1 4.3 0
23 1 3.675 0
0 0 0 0

2 2 0 0 1                                   region 2
-1 38 1 7.35 0        30 10 5.756 1.672
-1 30 10 5.756 1.672  29 12 5.29 2.06
-1 29 12 5.29 2.06    27 14 4.824 2.526
-1 27 14 4.824 2.526  14 27 2.526 4.824
-1 14 27 2.526 4.824  12 29 2.06 5.29
-1 12 29 2.06 5.29    10 31 1.672 5.756
-1 10 31 1.672 5.756  1 38 0 7.35
1 47 0 9.825
-1 1 47 0 9.825       16 34 2.831 6.993
-1 16 34 2.831 6.993  18 32 3.297 6.527
-1 18 32 3.297 6.527  20 30 3.763 6.061
-1 20 30 3.763 6.061  33 17 6.061 3.763
-1 33 17 6.061 3.763  35 15 6.527 3.297
-1 35 15 6.527 3.297  37 13 6.993 2.831
-1 37 13 6.993 2.831  47 1 9.825 0
0 0 0 0

3 1 20000 0 2                               region 3
-1 26 5 4.525 .363    30 10 5.756 1.672
-1 30 10 5.756 1.672  29 12 5.29 2.06
-1 29 12 5.29 2.06    24 6 4.059 .829
-1 24 6 4.059 .829    26 5 4.525 .363
0 0 0 0

4 1 20000 0 2                               region 4
-1 29 12 5.29 2.06    24 6 4.059 .829
-1 24 6 4.059 .829    23 4 3.593 .363
-1 23 4 3.593 .363    20 6 3.127 .829
-1 20 6 3.127 .829    27 14 4.824 2.526
-1 27 14 4.824 2.526  29 12 5.29 2.06
0 0 0 0

5 1 -20000 0 2                              region 5
-1 6 20 .829 3.127    4 23 .363 3.593
-1 4 23 .363 3.593    6 25 .829 4.059
-1 6 25 .829 4.059    12 29 2.06 5.29
-1 12 29 2.06 5.29    14 27 2.526 4.824
-1 14 27 2.526 4.824  6 20 .829 3.127
0 0 0 0

6 1 -20000 0 2                              region 6
-1 6 25 .829 4.059    12 29 2.06 5.29
-1 12 29 2.06 5.29    10 31 1.672 5.756
-1 10 31 1.672 5.756  4 26 .363 4.525
-1 4 26 .363 4.525    6 25 .829 4.059
0 0 0 0

7 2 0 0 1                                   region 7
-1 27 14 4.824 2.526  14 27 2.526 4.824
-1 14 27 2.526 4.824  6 20 .829 3.127
-4 6 19 .829 2.8284   10 10 1.414 1.414  19 5 2.8284 .8284
-1 20 6 3.127 .829    27 14 4.824 2.526
0 0 0 0
```

FIELD input file, FIELD.INP, for the magnet of Figure 1.

```
0       no permeability tables
1       pick up dump 1
32 1     to save FOR034.DAT for POTPLOT
0 0     no more PBMK constants
23 1     L-printout from 1
24 1       to 1
25 1     K-printout from 1
26 1       to 1
0 0     no more TVAR constants
0       end of data
```

TRIP input file, TRIP.INP, for the magnet of Figure 1.

```
0 13 0 45 .3  1  0  1
0 13 0 45 .3  0 20  2
```

The first line reads: Plot the magnet between YMIN=0, YMAX=13, XMIN=0, XMAX=45, with a scale of 0.3 (use 1.0 on a Tektronix terminal), plotting the mesh lines but no flux lines, picking up dump 1.

The second line reads: Plot the magnet between YMIN=0, YMAX=13, XMIN=0, XMAX=45, with a scale of 0.3, not plotting the mesh but plotting 20 flux lines, picking up dump 2.

Appendix 2: Problem Constants

Name	PRBCON Index	PBMK Index	TVAR Index	Description	Range	Def.
NREG*	1			Number of regions	1→150	**
LMAX*	2			Number of horizontal logical grid lines	1→150	**
KMAX*	3			Number of vertical logical grid lines	1→150	**
MODE	4	3		0—An all-points problem. −1—Air, then iron: first an infinitely permeable solution, then a finite-μ solution −2—Air only: iron is infinitely permeable	0,−1,−2	0
IUP*	5	4		Upper reflecting boundary	0,1	0
ILO*	6	5		Lower reflecting boundary	0,1	0
IRT*	7	6		Right reflecting boundary	0,1	0
ILF*	8			Left reflecting boundary	0,1	0
SFACT*	14	19		Stacking factor: fraction of Fe-region that is really iron. To allow for laminated magnets	0.→1.	1.
EPSOC	15	11		FIELD convergence criterion	10^{-7}→10^{-4}	10^{-6}
IPRFQ	16	12		FIELD print frequency	1→20	20
CONVER*	18	23		Conversion factor: used to convert specified units to cm. If input dimensions in inches, specify 2.54	0.→∞	1.
CMULT	19	20		Current multiplier. Scales current by this factor	−∞→+∞	1.
KBZERO*	26	21		If BDES≠0., current will be scaled to achieve BDES Gauss at K=KBZERO, L=1	1→KMAX	NA
LPRNT	27	22		L-row of FIELD printout it problem has not converged	1→LMAX	1
EPS	28			MESH convergence criterion	10^{-7}→10^{-4}	10^{-5}
BDES*	29	24		If nonzero, desired magnetic field at K=KBZERO, L=1	−∞→+∞	0.
MODEL*	31	26		Linear (0) or cylindrical (1) coordinates	0,1	0
KBOUND	32	27		1 if boundary value problem, 0 otherwise	0,1	0
GAMA	33	28		If nonzero, value of $\gamma = 1/\mu$ Used for constant-γ problems	0.→1.	0.
L_o	34		23	Lower Limit of L-printout (in FIELD)	0→L_f	1
L_f	36		24	Upper limit of L-printout (in FIELD)	L_o→LMAX	LMAX
K_o	37		25	Lower limit of K-printout (in FIELD)	0→K_f	1
K_f	38		26	Upper limit of K-printout (in FIELD)	K_o→KMAX	KMAX
TIMLIM	41		27	Time limit: CPU min. until FIELD drops a dump on FOR035.DAT and exits, even if it has not yet converged	1.→+∞	60.
NDMPA0	42	30		Dump on A0.DAT to pick up for initial potentials 0—No dump; 1—last dump; 2,3,...—specified dump	0→# dumps on A0.DAT	0
A0MULT	43	31		Scaling factor for initial potentials.	−∞→+∞	1.
HAPESV	44	32		1 to save harmonic polynomial edit, 0 otherwise. Use it you will run POTPLOT	0,1	0

*Automatically specified in MESH.INP as a result of running DIAGRAM.
**No default—these constants *must* be specified in MESH.INP.

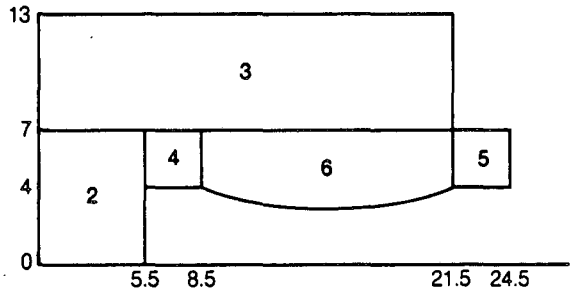
Appendix 3: A Sample Run of DIAGRAM—Part I

```
DIAGRAM—Version 2.0:
   Automated mesh production for TRIM simulations.

Should I fetch an old diagram for you? (def: NO) no
Name of diagram: A Simple C-Magnet
Units: cm, dm, m, in (default is cm): in
Lower left hand coordinate of your universe: 0 0
Upper right hand coordinate of your universe: 45,13
Location of the center of interest: (15,0)
Cylindrical coordinates? (def: NO) <CR>
Upper reflecting boundary? (def: NO) <CR>
Lower reflecting boundary? (def: NO) yes
Right reflecting boundary? (def: NO) <CR>
Left reflecting boundary? (def: NO) <CR>
B-Desired at center of Interest (def: not applicable): <CR>
Stacking factor (def: 1.0): <CR>

MAIN MENU ...
-1: Exit this program          3: Modify diagram
 0: Add region(s)              4: List region(s)
 1: Show geometry              5: Rezone diagram
 2: Make MESH input file       6: Help!
-> 0

-1: Return to main menu
Region 2:
  0: Is a rectangle
  1: Is a polygon
  2: Is made of various arcs
-> 0
Lower left hand coordinate of rectangle: 0 0
Upper right hand coordinate of rectangle: 5.5 7
What's the region made of:
0 = air  1 = Fe  n = Fe (n>1, use table #n): 1

-1: Return to main menu
Region 3:
  0: Is a rectangle
  1: Is a polygon
  2: Is made of various arcs
-> 0
Lower left hand coordinate of rectangle: 0 7
Upper right hand coordinate of rectangle: 21.5 13
What's the region made of?
0 = air  1 = Fe  n = Fe (n>1, use table #n): 1

-1: Return to main menu
Region 4:
  0: Is a rectangle
  1: Is a polygon
  2: Is made of various arcs
-> <CR>
Lower left hand coordinate of rectangle: 5.5 4
Upper right hand coordinate of rectangle: 8.5 7
What's the region made of?
0 = air  1 = Fe  n = Fe (n>1, use table #n): 0
Any current in the region? (def: NO) yes
Do you want to specify a current density? (def: NO) no
Current: 10000

-1: Return to main menu
Region 5:
  0: Is a rectangle
  1: Is a polygon
  2: Is made of various arcs
-> 0
Lower left hand coordinate of rectangle: 21.5 4
Upper right hand coordinate of rectangle: 24.5 7
What's the region made of?
0 = air  1 = Fe  n = Fe (n>1, use table #n): 0
Any current in the region? (def: NO) yes
Do you want to specify a current density? (def: NO) no
Current: -10000
```

The user invokes Diagram by typing **diagram**. *The program clears the screen and print these two lines. The program asks if we wish to recall a previously entered magnet. We say 'no.' We could also just enter* **<CR>** *(return), and achieve the same thing, since it is the default. 'yes' and 'no' may be shortened to 'y' and 'n'.*

Ordered pairs of numbers may be separated by spaces or commas (or both). They may even be enclosed in parenthesis. The "center of interest" is the area around which you would like the simulation to be most accurate, e.g., where a particle beam will be.

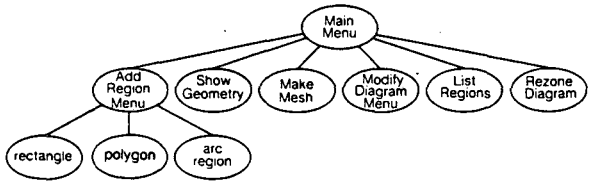
We begin with a sketch of the magnet we want to create:



Then we break it into regions, avoiding big, peculiarly-shaped ones (e.g., making the entire iron yoke a single region). Number the pieces, beginning at 2.

We add the regions, one at a time, until we're done.

DIAGRAM's menu structure looks like this:



On responses where the machine wants an integer, **<CR>** *as a response means 0.*

Specify air (0) for current carrying regions. Region code >1 is used in conjunction with permeability tables, specified in FIELD.

While DIAGRAM performs some checks on the legality of regions, it is primarily your responsibility to check you have not entered a meaningless shape.

At most points in the program, you can type 'about' to get 'out of' where you are.

```
-1: Return to main menu
Region 6:
  0: Is a rectangle
  1: Is a polygon
  2: Is made of various arcs
-> 2

To terminate, close region or type -1.

-4: List arcs
-3: Delete last arc
-2: Close region
-1: Abort region
Next arc part of a:
  0: Line              3: Hyperpola
  1: Circle            4: X-parabola
  2: Ellipse           5: Y-parabola
-> 5
First, second, third points: 8.5,4  15,3 21.5,4

-4: List arcs
-3: Delete last arc
-2: Close region
-1: Abort region
Next arc part of a:
  0: Line              3: Hyperbola
  1: Circle            4: X-parabola
  2: Ellipse           5: Y-parabola
-> 0
Line from (21.50, 4.00) to 21.5,7

-4: List arcs
-3: Delete last arc
-2: Close region
-1: Abort region
Next arc part of a:
  0: Line              3: Hyperbola
  1: Circle            4: X-parabola
  2: Ellipse           5: Y-parabola
-> 0
Line from (21.50, 7.00) to 8.5,7

-4: List arcs
-3: Delete last arc
-2: Close region
-1: Abort region
Next arc part of a:
  0: Line              3: Hyperbola
  1: Circle            4: X-parabola
  2: Ellipse           5: Y-parabola
-> -2 <CR>
What's the region made of:
0 = air  1 = Fe  n = F (n>1, use table #n): 1

-1: Return to main menu
Region 7:
  0: Is a rectangle
  1: Is a polygon
  2: Is made of various arcs
-> 0
Lower left hand coordinate of rectangle: 11 0
Upper right hand coordinate of rectangle: 19 1
What's the region made of?
0 = air  1 = Fe  n = Fe (n>1, use table #n): <CR>
Any current in the region? (def: NO) <CR>

-1: Return to main menu
Region 8:
  0: Is a rectangle
  1: Is a polygon
  2: Is made of various arcs
-> -1

MAIN MENU ...
-1: Exit this program      3: Modify diagram
  0: Add region(s)         4: List region(s)
  1: Show geometry         5: Rezone diagram
  2: Make MESH input file  6: Help!
-> 1

Show entire magnet? (def: YES) <CR>
```

When specifying arc regions, never list points twice. For example, to specify a line you give only the point on one end of the arc. You may specify the region by listing the arcs clockwise or counterclockwise.

To achieve good results, we should add one more region–an air region surrounding the area of interest. This gives DIAGRAM something to "grab on to" when producing the distorted grid.

The magnet is now completely specified. To take a look, select entry 1 from the main menu. The next question is primarily used for magnets that are very much longer in one dimensions or the other. If so, you may only want to look at a piece of the magnet, setting up a "window" into the physical universe.

DIAGRAM clears the screen and produces the plot labeled Fig. 1. The asterisk marks the center of interest in the problem.

```
MAIN MENU ...
-1: Exit this program          3: Modify diagram
 0: Add region(s)              4: List region(s)
 1: Show geometry              5: Rezone diagram
 2: Make MESH input file       6: Help!
-> 2

Ceiling on number of points in simulation (def:  4000): <CR>

I select LMAX = 30
I select KMAX = 113
Making total number of points = 3910

Computing logical diagram...

Logical diagram created.

Do you want a picture of the logical diagram? (def:  YES) <CR>

Do you want to save this as MESH.INP? (def:  YES) <CR>
Do you want to make a distorted mesh? (def:  YES) <CR>

A Simple C-Magnet

Number of regions = 7
LMAX = 32   KMAX = 108
RHOX = 1.60  RHOY = 1.60  EPSO = 0.10E-04
Conversion factor = 2.54
KBZERO = 1
This is a mode     IRON problem
This problem has symmetry about the LOWER boundary

Region:  1   Material: AIR      Zoning: Right
Region:  2   Material: Fe (2)   Zoning: Equilateral
Region:  3   Material: Fe (2)   Zoning: Equilateral
Region:  4   Material: AIR      Current:  10000.0   Zoning: Right
Region:  5   Material: AIR      Current: -10000.0   Zoning: Right
Region:  6   Material: Fe (2)   Zoning: Equilateral
Region:  7   Material: AIR      Zoning: Right
```

We are now in a position to make a logical diagram. All we have to do is choose the number of points in the simulation, (LMAX+4)(KMAX+2).

It's a good idea to always look at the logical diagram to make sure there was no problems (like distinct regions running into one another). DIAGRAM look for some problems, but doesn't check everything.

Again, DIAGRAM clears the screen, drawing what is shown in Fig. 2.

DIAGRAM has now invoked MESH to produce the irregular triangular mesh.

cycle	georx	geory	rhox	rhoy
1	0.1164936059	0.0591877678	1.60	1.60
2	0.0523231258	0.0273570071	1.60	1.60
3	0.0381431678	0.0197205791	1.60	1.60
4	0.0281017243	0.0142184168	1.60	1.60
5	0.0211255676	0.0108385363	1.61	1.62
6	0.0165416502	0.0084667614	1.61	1.62
7	0.0126587265	0.0066775619	1.61	1.62
8	0.0090806666	0.0053564380	1.61	1.62
9	0.0070010043	0.0079306632	1.61	1.62
10	0.0053276634	0.0055897779	1.63	1.62
11	0.0045266876	0.0041925120	1.63	1.62
12	0.0036652552	0.0033171755	1.63	1.62
13	0.0031169058	0.0026798586	1.63	1.62
14	0.0026289148	0.0023211649	1.63	1.62
15	0.0021961229	0.0019551882	1.66	1.65
16	0.0020089471	0.0019430453	1.66	1.65
17	0.0015989647	0.0016403315	1.66	1.65
18	0.0013374708	0.0015270593	1.66	1.65
19	0.0010650729	0.0013140322	1.66	1.65
20	0.0008874618	0.0012257257	1.68	1.72
21	0.0007875557	0.0013591288	1.68	1.72
22	0.0006586540	0.0012622618	1.68	1.72
23	0.0005326353	0.0011375287	1.68	1.72
24	0.0004445386	0.0010545148	1.68	1.72
25	0.0003699157	0.0009705064	1.70	1.76
26	0.0003367263	0.0010531921	1.70	1.76
27	0.0002845421	0.0009708567	1.70	1.76
28	0.0002448048	0.0009061777	1.70	1.76
29	0.0002127672	0.0008391035	1.70	1.76
30	0.0001886182	0.0007862857	1.73	1.80
31	0.0001886167	0.0008498353	1.73	1.80
32	0.0001710584	0.0007931846	1.73	1.80
33	0.0001565121	0.0007322542	1.73	1.80
34	0.0001448761	0.0006807868	1.73	1.80
35	0.0001349811	0.0006280087	1.77	1.82

The mesh is defined to have converged when both georx and geory are less than 10^{-3} for five successive cycles.

```
36      0.0001457307    0.0006361261    1.77    1.82
37      0.0001360186    0.0005846817    1.77    1.82
38      0.0001272944    0.0005383072    1.77    1.82
39      0.0001190161    0.0004933870    1.77    1.82
40      0.0001114192    0.0004518219    1.80    1.83
41      0.0001174816    0.0004379629    1.80    1.83
42      0.0001093016    0.0003988915    1.80    1.83
43      0.0001014261    0.0003623262    1.80    1.83
44      0.0000939140    0.0003283787    1.80    1.83
45      0.0000867927    0.0002969487    1.82    1.83
46      0.0000863228    0.0002766039    1.82    1.83
47      0.0000791861    0.0002487911    1.82    1.83
48      0.0000723302    0.0002231083    1.82    1.83
49      0.0000658508    0.0001995488    1.82    1.83
50      0.0000597387    0.0001779714    1.83    1.84
51      0.0000561422    0.0001604826    1.83    1.84
52      0.0000504913    0.0001422822    1.83    1.84
53      0.0000451730    0.0001258262    1.83    1.84
54      0.0000402465    0.0001109272    1.83    1.84
55      0.0000357121    0.0000975001    1.83    1.84
56      0.0000319117    0.0000855634    1.83    1.84
57      0.0000280167    0.0000747224    1.83    1.84
58      0.0000244875    0.0000650640    1.83    1.84
59      0.0000212886    0.0000564605    1.83    1.84
60      0.0000184063    0.0000488118    1.83    1.84
61      0.0000157334    0.0000417777    1.83    1.84
62      0.0000134281    0.0000358496    1.83    1.84
63      0.0000114075    0.0000306573    1.83    1.84
64      0.0000096191    0.0000261069    1.83    1.84
65      0.0000080495    0.0000221317    1.83    1.83
66      0.0000065842    0.0000184722    1.83    1.83
67      0.0000054208    0.0000155098    1.83    1.83
68      0.0000044292    0.0000129569    1.83    1.83
69      0.0000035944    0.0000107632    1.83    1.83
70      0.0000028952    0.0000088841    1.83    1.83
71      0.0000023035    0.0000072075    1.83    1.83
72      0.0000018608    0.0000058675    1.83    1.83
73      0.0000015283    0.0000047428    1.83    1.83
74      0.0000012818    0.0000038011    1.83    1.83

Positive current:  10000.00
Negative current:  -10000.00

Generation completed.

Distorted mesh successfully created.

Do you want to see the distorted mesh? (def: YES) no

MAIN MENU ...
-1: Exit this program          3: Modify diagram
 0: Add regions(s)             4: List region(s)
 1: Show geometry              5: Rezone diagram
 2: Make MESH input file       6: Help!
-> -1
Save this file? (def:  YES) <CR>
Name of file:  sample
File has been sucessfully written.
Make a hardcopy of the diagram? (def:  NO) <CR>
You sure you want to exit? (def:  YES) <CR>
```

We are now back in DIAGRAM. A 'yes' response clears the screen and draws the magnet of Fig. 3.

We save the file. (DIAGRAM appends .DAT to our file name if we haven't specified a suffix).

The hard copy it refers to is a picture of the physical and logical diagrams, made on the zeta plotter.

DIAGRAM again clears the screen, prints 'Thank you,' and you're done.

Appendix 4: Conics Codes

The following codes may be useful when preparing MESH-input files by hand; they are an alternative to the form:

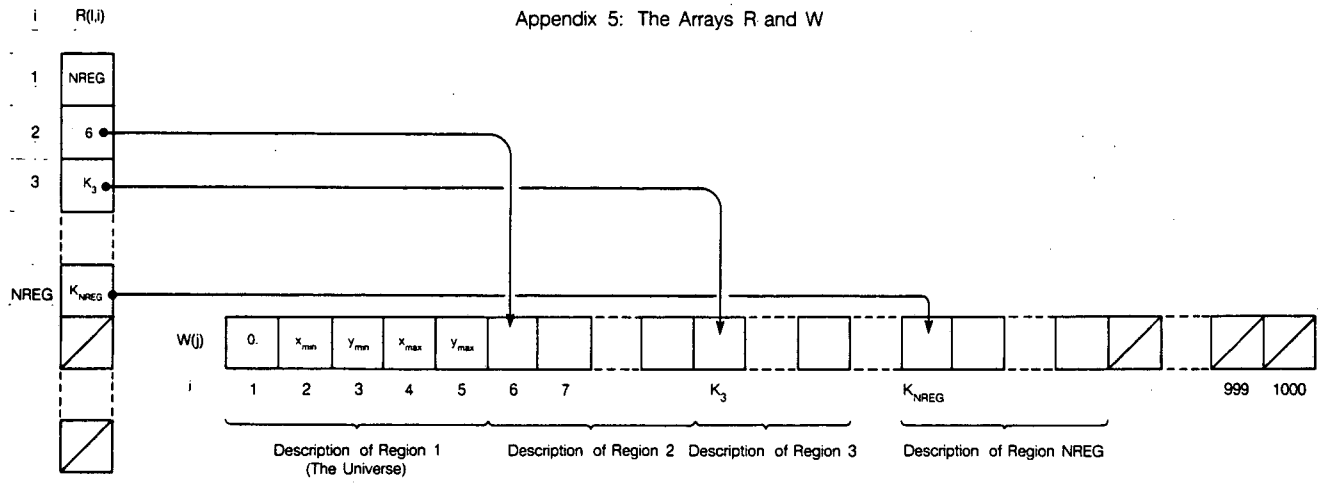
L K Y X

used to connect successive points along logical lines; for here (L_i, K_i), need *not* be on the same logical line as (L_j, K_j):

−1	L_1	K_1	Y_1	X_1	L_2	K_2	Y_2	X_2					a straight LINE
−4	L_1	K_1	Y_1	X_1	L_2	K_2	Y_2	X_2	L_3	K_3	Y_3	X_3	an arc of a CIRCLE
−5	A	B	X_c	Y_c	θ	L_1	K_1	Y_1	X_1	L_2	K_2	Y_2	X_2 an arc of an ELLIPSE
−6	L_1	K_1	Y_1	X_1	L_2	K_2	Y_2	X_2	L_3	K_3	Y_3	X_3	an arc of a 90° HYPERBOLA
−8	L_1	K_1	Y_1	X_1	L_2	K_2	Y_2	X_2	L_3	K_3	Y_3	X_3	an arc of a Y-PARABOLA
−9	L_1	K_1	Y_1	X_1	L_2	K_2	Y_2	X_2	L_3	K_3	Y_3	X_3	an arc of an X-PARABOLA

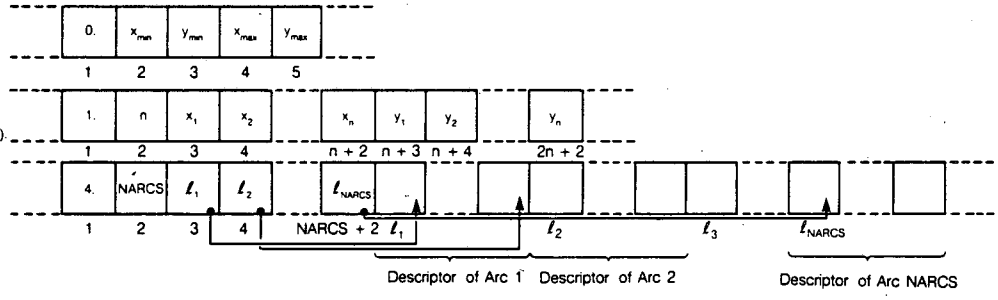
Here (L_1,K_1), (L_2,K_2), and (L_3,K_3) are the logical coordinates of the first, second, and third points; (Y_1,X_1), (Y_2,X_2), (Y_3,X_3) are the physical coordinates of the first, second, and third points; A, B, (X_c,Y_c), and θ are the length of the semi-major axis, the length of the semi-minor axis, the coordinate of the center of the ellipse, and the angle of rotation of the ellipse, measured counter-clockwise in degrees, with 0° meaning that the semi-major axis is parallel to the x-axis. An X-Parabola is a parabola opening in the ±X-direction, and a Y-Parabola is a parabola opening in the ±Y-direction.

Appendix 5: The Arrays R and W



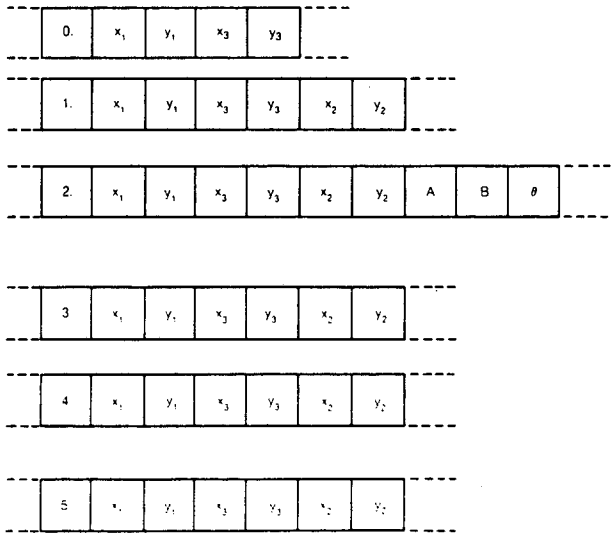
Arc Descriptors:

- Rectangles –
  a rectangle with vertices (x_{min}, y_{min}), (x_{max}, y_{min}), (x_{max}, y_{max}), (x_{min}, y_{max}).
- Polygons –
  a polygon with vertices (x_1, y_1), (x_2, y_2), ..., (x_n, y_n).
- Arc Regions –
  a region composed of arcs 1, 2, ..., NARCS.



Region Descriptors:

- Line –
  a line segment directed from (x_1, y_1) to (x_3, y_3).
- Circle –
  an arc of a circle, directed from (x_1, y_1) to (x_3, y_3), passing through (x_2, y_2).
- Ellipse
  an arc of an ellipse of semi-major axis A, semi-minor axis B, angle at rotation θ, from (x_1, y_1) to (x_3, y_3), passing through (x_2, y_2).
- X-Parabola
  an arc of a parabola opening in the ± x-direction, from (x_1, y_1) to (x_3, y_3), passing through (x_2, y_2).
- Y-Parabola
  an arc of a parabola opening in the ± y-direction, from (x_1, y_1) to (x_3, y_3), passing through (x_2, y_2).



XBL 855-10515

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720