UCRL-7880

University of California

Ernest O. Lawrence Radiation Laboratory

AN IRREGULAR TRIANGLE MESH GENERATOR

Livermore, California

DISCLAIMER

UCRL-7880
Physics, UC-34
TID-4500 (32nd Ed.)

UNIVERSITY OF CALIFORNIA
Lawrence Radiation Laboratory
Livermore, California

Contract No. W-7405-eng-48

AN IRREGULAR TRIANGLE MESH GENERATOR

Alan M. Winslow

August 25, 1964

AN IRREGULAR TRIANGLE MESH GENERATOR

Alan M. Winslow

Lawrence Radiation Laboratory, University of California
Livermore, California

August 25, 1964

ABSTRACT

A code is described which constructs an irregular triangle mesh. Input to the code is prepared by dividing the mesh into regions of uniform material properties and specifying the coordinates of selected points on the boundary of each region. The code then interpolates the omitted boundary points, locates the internal mesh points of each region by the pseudo-equipotential method, and assigns regional properties to each triangle. Except that a region boundary may not cross itself, there are no restrictions on the shape of a region.

The iterative procedure which relaxes the mesh uses an over-convergence factor which is recalculated each cycle and maintained close to the optimum value in order to minimize the number of cycles needed for convergence.

I. INTRODUCTION

Two-dimensional meshes composed of irregular triangles have certain advantages over rectangular meshes for the numerical solution of partial differential equations involving the Laplacian, such as the diffusion equation and Laplace's and Poisson's equations.[1] Because of the irregularity of these meshes, however, it is not practical to construct them by hand, and so a computer code is required for this purpose. Such a code, called a generator, forms the mesh – which may contain thousands of mesh points – out of regional information. These regions, typically ranging from 1 to 100 in number, are obtained by division of the mesh according to the properties of the physical system to be calculated. Each region, characterized by uniform material properties, is defined by specifying the geometrical and logical coordinates of selected points on its boundary. The generator interpolates the omitted points, calculates other variables which occur in the partial

differential equation to be solved, and prepares a tape or other output which becomes the input for the triangle mesh code. Auxiliary functions of the generator which are important in its practical use are: checking the input for errors, printing out a record of the input, and producing pictures of the completed mesh.

The first generator for triangle mesh codes of the type described in Ref. 1 was written in 1958 by R. MacLean for the IBM 704 (unpublished). It has been in use since that time on the 709, 7090, and 7094 and has recently been translated into FORTRAN II. A hand-coded version for the LARC, written by R. Clay in 1961 (unpublished), is essentially the same as MacLean's with some improvements.

This report describes triangle-mesh-generator codes in a manner useful to anyone who wishes to understand the existing generators or to code a new one. Most of the features described here are to be found in the existing generators; a few are given as desirable additions. Although the operations performed by these codes are simple to describe, they contain rather complicated logic which is best understood by consulting flow diagrams which are available for the codes.

We begin by describing the structure of the triangle mesh, the various kinds of mesh variables, and a type of intermediate storage convenient for use in a generator code.

The generator codes consist of four distinct sections which are described next under the names given them in the original code: INPT, HSTAR, SETTLE, and GENOR. A few additional features are mentioned at the end followed by input and pictures for two sample problems. In an appendix the equations are given for the pseudo-equipotential mesh lines.[2]

II. TRIANGLE MESH AND ASSOCIATED VARIABLES

The triangle mesh to be constructed is topologically regular, and can be pictured as shown in Fig. 1, in which every interior mesh point is a common vertex for six triangles. Each horizontal line of mesh points is called a row and is numbered with the index ℓ as shown, running from $\ell = 0$ at the bottom to $\ell = L$ at the top of the mesh. Within each row, mesh points are numbered with the index k, running from $k = 0$ at the beginning of each row to $k = K$ at the end. Note that the convention is made that the origin $\ell = 0$, $k = 0$ is a point at which three mesh lines meet.

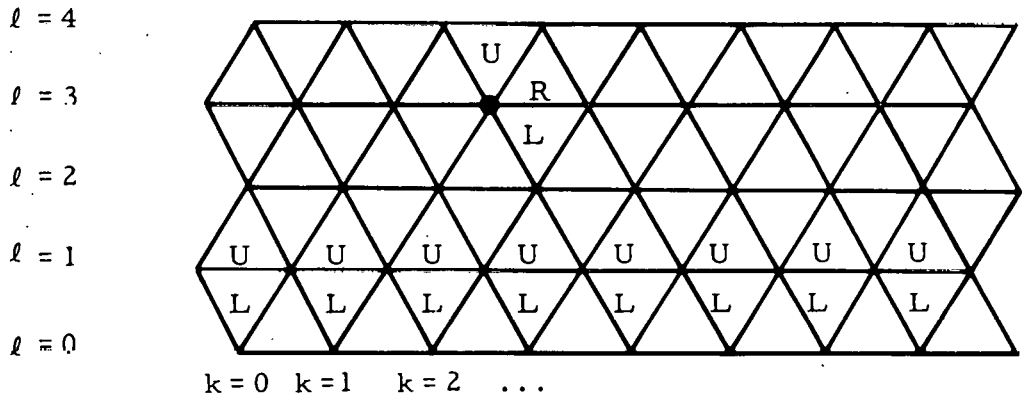


Fig. 1. Logical diagram of the triangle mesh.

Associated with each mesh point are variables such as the x and y coordinates and functions which are dependent variables for the differential equation being solved. For convenience in calculating systems which are longer in one coordinate direction than in the other, provision is made so that rows can run either in the x or in the y direction. This has the effect that clockwise traversal of a closed path in the ℓ, k space may be either clockwise or counterclockwise in x, y space. If a number called the mesh parity, which can assume the values +1 or -1, is given for each mesh and used to multiply all quantities (such as areas) which change sign upon interchange of x and y, the same coding can be used for both cases. By convention, -1 parity represents the case where the rows run in the x direction, and +1 parity the other case.

Certain mesh variables, such as triangle areas, are defined for triangles rather than for mesh points. Every mesh point ℓ, k has two triangles associated with it, called upper and lower, whose vertices are ℓ, k; ℓ, k + 1; and $\ell \pm 1$, $k + \delta$ respectively, where $\delta = 0$ for odd ℓ, and $\delta = 1$ for even ℓ. In Fig. 1 the upper and lower triangles associated with the row $\ell = 1$ are labeled U and L respectively.

A third type of mesh variable is associated with the sides of triangles. These are called upper, row, and lower sides; an example is shown in Fig. 1 for the point $\ell = 3$, $k = 3$ where they are labeled U, R, and L respectively.

If we use the letters V for vertex (mesh point) variables, T for triangle variables, and U, R, L as defined above, we have a convenient

naming scheme for all mesh variables. For example, XV and YV are the x and y coordinates of a vertex, ATU and ATL are the areas of upper and lower triangles, and CPU, CPR, and CPL are the coupling coefficients[3] connecting the given mesh point to the three points mentioned above.

The storage of information in triangle mesh codes is done by rows: all the variables for $\ell = 0$ form one record, followed by the variables for $\ell = 1$, etc. Within each row all the variables of a given type are stored together, arranged sequentially from k = 0 to k = K: for example, all the quantities ATU are stored sequentially, followed by all the quantities ATL, etc.[4]

The input to the generator is specified by regions, a region being composed of any number of whole triangles. The boundary of a region consists of straight line segments made up of triangle sides. It may have any shape, but is not permitted to cross itself. A region need not be logically convex: for example, the region shown in Fig. 2 is permissible.

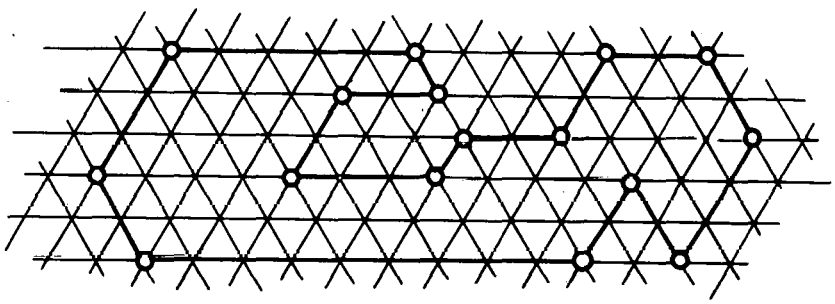


Fig. 2. Logical diagram of a region. Circled points must be specified.

To specify a region in the generator input, it is sufficient to give only those boundary points which are logical corners (see Fig. 2). Other boundary mesh points may be specified, but if not specified they will be supplied by interpolation (see Sec. IV). Non-boundary points are located by an "equipotential" method (see Sec. V).

The specified region boundary points are listed in the input as they are encountered in a traversal around the region, starting at any point and proceeding in either direction. To transform this regional input into row output, the generator sets up intermediate storage of the input information

consisting of four blocks of information, the first two of which contain, respectively, all the x coordinates and all the y coordinates, stored sequentially by rows. The third block, called the k block, consists of one word for each point which contains various kinds of information about the point, such as whether or not it is a region boundary point (see Sec. IV). The fourth block is a table listing the properties of the various regions. When these blocks have been completed, information drawn from them is used to construct row records, one at a time, which are then written as output.

The sample problems in Sec. VIII illustrate the saving in effort represented by the generator. Of the 798 mesh points in the first sample problem, only 43 were specified as input to the generator, the rest being generated by the methods described below.

III. INPT

Functions:

1. Reads problem constants, checks for errors, stores internally, prints out.
2. Reads regional input, checks for errors, prints out, stores externally.

Special Features:

a) Input is on punched cards.

b) Problem constants are divided into a required set, which must be given, and an optional set, which are assigned stored values if not given or if given as zero.

c) For each region there are region constants followed by region boundary information. Boundary points are given in succession: for each point the two logical coordinates are specified followed by the two geometrical coordinates.

d) Region numbers are arbitrary and may be in any order.

e) Specification of a region boundary may start at any boundary point and go around in either direction. The first boundary point need not be repeated to indicate closure of the region.

f) Errors checked are:
- format errors;
- problem constants: checks number of regions;
- regions: checks boundary points for logical consistency and for too many points. If problem constant or regional errors are found, the generator continues if possible to the end of INPT, thereby checking all regions.

IV. HSTAR

Functions:

1. Reads in the regional information written by INPT and constructs a list of region constants stored internally.
2. Sets up x, y, and k blocks internally, within which it:
   a) interpolates and stores region boundary points;
   b) labels region boundary points with boundary sentinels;
   c) labels triangle sides with boundary sentinels for use in distinguishing the inside of a region from the outside;
   d) assigns region numbers to triangles;
   e) assigns region numbers to non-boundary points;
   f) stores zero for coordinates of non-boundary points.

   The above operations are performed one region at a time.

Special Features:

a) The x, y, and k blocks each contain one word for each mesh point, stored sequentially. The x block contains all the x coordinates, the y block all the y coordinates, the k block a label for each point which gives the following information:
- whether or not the point is boundary point;
- if it is a boundary point, whether or not the two sides; associated with it lie on a region boundary:
- the region numbers of the two triangles associated with the point;
- the region number associated with the point (non-boundary points only).

Upper and lower sides and triangles are associated with each k block word in the same manner as in Fig. 1. In the IBM 7094 code,

two of the three tag bits are used to label upper and lower boundary sides, the third bit being used to indicate whether the point itself lies on a boundary.

b) Assignment of region numbers to triangles is done by sweeps running between rows – that is, the lower triangles for row $\ell + 1$ and the upper triangles for row ℓ are assigned alternately, as shown in Fig. 3.

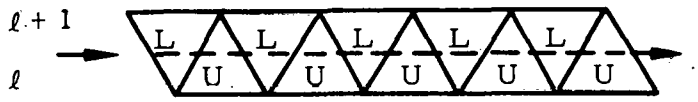


Fig. 3. Assignment of region numbers to triangles.

As each upper or lower side is crossed, track is kept of whether it is or is not a boundary side, which indicates whether the following triangles are inside or outside the region.

After all the triangles of a region have been labeled, the side boundary labels are erased so that side labels for only one region at a time will be found in the k block. On the IBM 7094 a typical k block word may read 017003700000. The first three digits mean that the upper triangle associated with the point belongs to region 17, and the second three digits mean that the lower triangle belongs to region 3. The seventh digit indicates that the point is a boundary point and that both the upper and lower sides associated with the point lie on a boundary. The k block word 003003000003 indicates a non-boundary point which itself lies in region 3. Region number zero is assigned to triangles lying outside the mesh such as lower triangles for $\ell = 0$ or upper triangles for $\ell = L$. Since the three blocks are set up one region at a time, the points can change regions, and the region numbers in the k block will be those of the last region in which the points were given. Thus a region superposition principle holds, which simplifies the preparation of input.

c) A region is assigned to a non-boundary point if all six of the triangles surrounding the point belong to that region. Since every

interior point must be completely surrounded by triangles belonging to the same region, this provides a check on the assignment of region numbers to triangles.

d) The region numbers in the k block belonging to vertices are used in SETTLE to control the mesh point iteration procedure through the list of region properties mentioned above. The region numbers belonging to triangles are used in the same way in GENOR to control the assignment of regional material properties to triangles in the construction of the row output.

e) Region boundary points not specified in the input are interpolated in two ways: by linear interpolation or by construction of a quarter-ellipse. Two points between which linear interpolation is to take place must both have the same value of k or else they must both lie on a logical straight line: that is, a line which is straight in the logical diagram. Through any point run three such lines, one being a row (constant ℓ). The linearly interpolated points will be equally spaced, but because of the superposition principle different sections of the same boundary will in general end up with different spacings. Overlapping may occur unless care is taken to specify for each region the boundary points at which regions meet. In case of an ellipse (which is restricted to an orientation with axes parallel to the logical axes), only the end points (and the center) need be specified; the other points are interpolated at equal increments of polar angle.

V. SETTLE

Function:

1. x and y coordinates of non-boundary points are calculated by an iterative procedure using intersecting "equipotentials," for mesh lines.

Special Features:

a) The entire mesh is swept each iteration cycle. Region boundary points are skipped.

b) Three sets of iterative equations may be used optionally, to produce approximately equilateral or approximately right triangles (see Appendix). The choice of equations for each region is determined by a region constant.

c) The option is provided for causing the mesh to satisfy a reflection condition at surfaces defined by specified lines of constant x or constant y (see Appendix).

d) The convergence criterion may be specified in the problem input. Five successive convergences are required for completion.

VI. GENOR

Functions:

1. An output tape is written containing, in addition to the coordinates, other quantities required as input by the triangle mesh code, arranged by rows.
2. If triangles with negative areas have been created due to crossing of zone lines (this may occur even without input errors when a sharply idented boundary is zoned with too few triangles), an error printout is made giving the location of these triangles.
3. Picutres of the mesh are made.

Special Features:

a) Since the output tape is written one row at a time, only one row need be set up at a time in memory. In practice it is convenient to set up two rows, the upper quantities for row ℓ and the lower quantities for row $\ell + 1$ being calculated at the same time. This is the same sequence in which the triangle region numbers are stored in the k block (see HSTAR).

b) The output is written and the picture plotted even when there are negative areas.

VII. SPECIFICATION OF ADDITIONAL KINDS OF INFORMATION

The following extra specifications are sometimes useful:

1. Calculating area of a region:
   In many problems it is desirable to know the area of a region in order to transform some region input quantity into other units.

For example, in the magnetostatic code[1] one specifies the total current, in amperes, for a given coil region, but it is necessary to transform this to amperes/cm^2 for use in the code. Instead of calculating the region area by hand, it is simple to calculate it during generation of the problem by making two sweeps through the mesh in GENOR. Only the region areas are calculated in the first sweep. They are stored with the region constants, and then in the second sweep are divided into the appropriate region input quantities.

2. Specification of regional quantities at mesh points:
Normally the specified regional properties are triangle quantities. If it is desired to specify a regional property which is defined at mesh points, the problem arises of determining what value to assign the quantity at a mesh point which lies on a region boundary, and which therefore may be a boundary point for as many as six regions. The simplest procedure is to count the number of regions which meet at the point and average the regional values, giving each region equal weight. This is done in GENOR, at the time the rows are being constructed, by scanning the six neighboring triangles of each region boundary point.

3. Specification of values of a variable at mesh points:
Sometimes it is necessary to prescribe values of some variable in the input for some set of mesh points, as, for example, to satisfy some given initial or boundary condition for a particular problem. Since these numbers are associated with a set of mesh points rather than with a region, this is best accomplished by adding a list of the numbers and their logical coordinates at the end of the input. These numbers need not be read in until the rows are being written out, then they can be inserted into the appropriate rows.

4. Specifying the location of non-boundary points:
Points within a region whose coordinates normally would be determined in SETTLE but which it is desired to specify could be given in the region input, and their k block sentinels labeled in HSTAR so that SETTLE would skip them.

VIII. SUMMARY

The most important features of the four sections of the generator may be summarized as follows:

INPT: maximum simplicity of input; maximum error checking; ability to change input format easily to adapt to different triangle mesh codes.

HSTAR: ability to interpolate between specified boundary points; ability to draw elliptical arcs; ability to properly label triangles in regions of any shape.

SETTLE: ability to produce different kinds of triangles; speed of convergence of iteration procedure.

GENOR: ability to change format of output tape easily to adapt to different triangle mesh codes.

IX. SAMPLE PROBLEMS

We shall now describe in detail the generation of two problems for the magnetostatic code.[1] The first is a C magnet of the type used in high-energy accelerators, the second a hypothetical H magnet.

In Fig. 4 are shown the materials and dimensions of the upper half of the C magnet. The lower half is assumed to be a mirror image of the upper half, so that by placing appropriate boundary conditions on the median plane, only the upper half need be calculated.

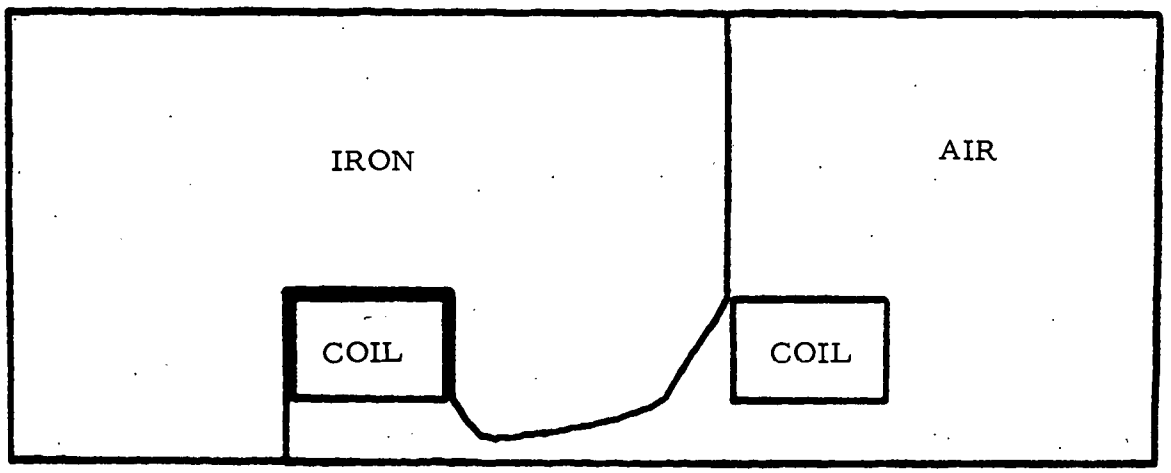


Fig. 4. C magnet showing material interfaces.

The problem divides naturally into four regions: the air, the iron, and the two coils. These regions are shown in Fig. 5, which is a logical map of the problem in ℓ, k space showing the connections of mesh points and the zoning. More zones are placed in the air directly below the pole tip, where more accuracy is desired, whereas in the coils and the air space to the right fewer zones are used.

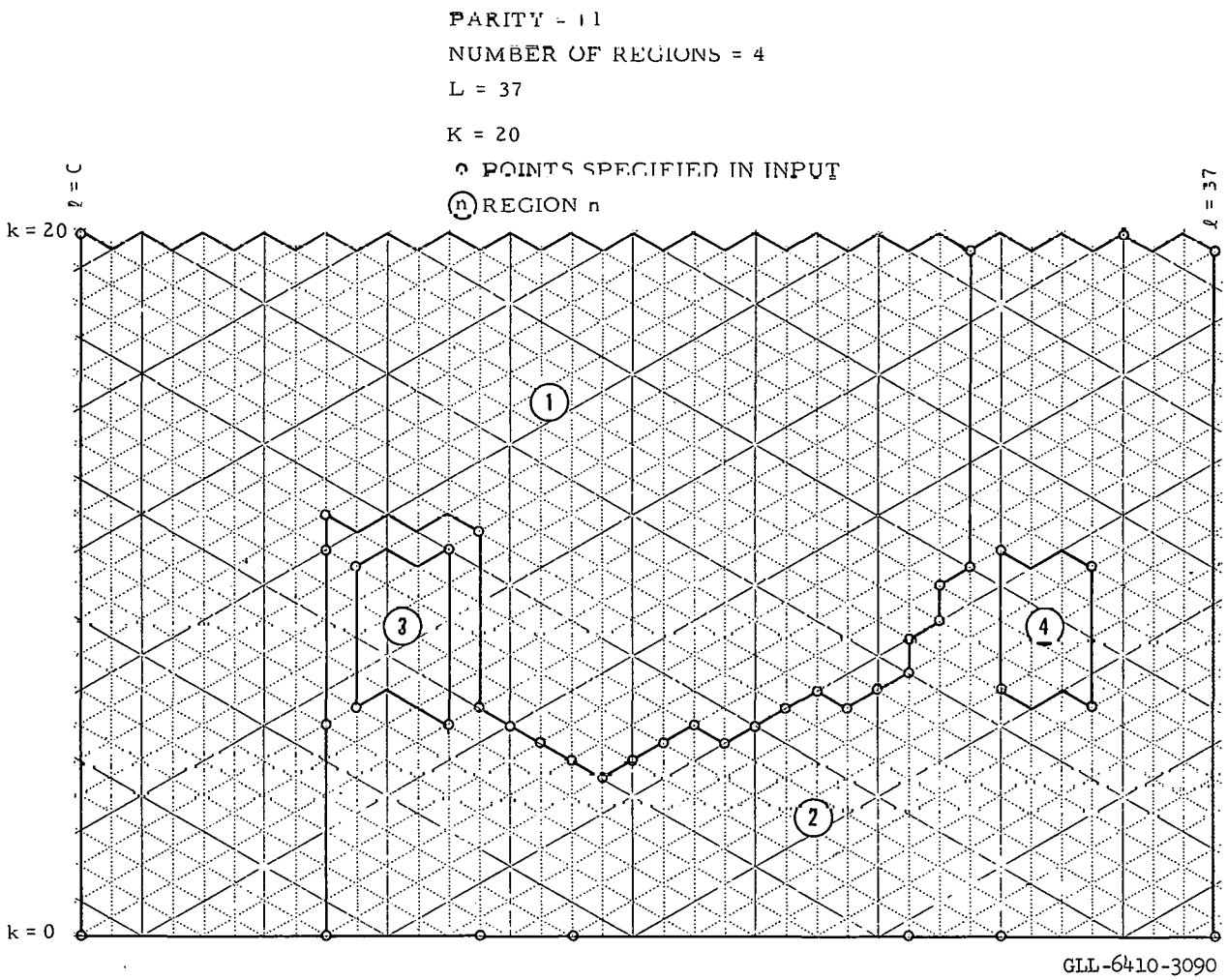


Fig. 5. Logical diagram of C-magnet mesh.

The card input to the generator for this problem reads as follows:

	Card	Input
	1	TRIM CMAG5
	2	+1 A4A37A20 Y
Region 1	3	A1 D2 + 0 + 0A1
	4	D0A0 + 0 + 0 D8A0 + 0 + 36 D37A0 + 0 + 150 D37A20 + 58 + 150 D29A20 + 58 + 94 D0A20 + 58 + 0
	5	-0 Y
Region 2	6	A2 D1 + 0 + 0 A1
	7	D8A0 + 0 + 36 D8A6 + 7.4 + 36 D8A11 + 21.3 + 36 D8A12 + 22 + 36 D13A12 + 22 + 58
	8	D13A7 + 8.526 + 58 D14A6 + 5.46 + 60
	9	D15A6 + 3.68 + 62 D16A5 + 3.34 + 64 D17A5 + 3.551 + 66 D18A5 + 3.762 + 68 D19A6 + 4.01 + 70
	10	D20A6 + 4.293 + 72 D21A6 + 4.62 + 74 D22A6 + 5.0 + 76 D23A7 + 5.448 + 78 D24A7 + 5.985 + 80
	11	D25A7 + 6.639 + 82 D26A7 + 7.454 + 84 D27A8 + 8.497 + 86 D27A9 + 11.596 + 88
	12	D28A9 + 14.73 + 90 D28A10 + 17.865 + 92 D29A11 + 21 + 94 D29A20 + 58 + 94
	13	D33A20 + 58 + 115 D37A20 + 58 + 150 D37A0 + 0 + 150 D33A0 + 0 + 115 D30A0 + 0 + 95
	14	-0 Y
Region 3	15	A3 D1 -1.0E5 + 0 A1
	16	D9A7 + 8 + 37 D9A11 + 21 + 37 D12A11 + 21 + 57 D12A6 + 8 + 57 D11A7 + 8 + 50
	17	-0 Y
Region 4	18	A4 D1 + 1.0E5 + 0 A1
	19	D30A7 + 8 + 95 D30A11 + 21 + 95 D33A11 + 21 + 115 D33A7 + 8 + 115
	20	-0 Y

The letters A and D used here are peculiar to the card-reading routine used on the IBM 7094, and indicate numbers to be placed in the address and

decrement parts of words, respectively. Numbers preceded by ± signs are floating point numbers, whose power of 10 can be indicated by the letter E followed by the power.

All cards except the first are written in variable-field format; that is, spaces are ignored when the cards are read, so that the numbers may be placed anywhere on the cards.

Card 1 carries the code name (TRIM) and the problem name (CMAG5).

Card 2 carries the problem constants, as follows:

+1: the parity

A4: the number of regions

A37: L, the number of the rows minus one

A20: K, the number of points in a row minus one

Y: a sentinel indicating the end of the problem constants.

These four constants are the required set for the magnetostatic code. Since no values for the optional constants are given, standard values will be assigned to them in this problem.

Cards 3-5 describe region 1, as follows:

Card 3:

Constants for region 1:

A1: the region number

D2: a label describing the material of the region (iron)

+0: region current in amperes

+0: region current density in amperes/cm^2

A1: a sentinel indicating that the region is to be zoned into equilateral triangles

Card 4:

Boundary points, with the format DℓAk + y + x.

Card 5:

-0 Y: A sentinel indicating the end of the region boundary points.

Cards 6-14, 15-17, and 18-20 describe regions 2-4 in a similar fashion.

Note that these regions all carry the material label D1, indicating non-ferromagnetic material (air or copper). Also note that regions 3 and 4, which are the coils, carry currents $\mp 1.0 \times 10^5$ amperes, respectively.

It will be observed that region 1 covers the entire space of the problem with iron, areas of which are then changed to air and coils by regions 2-4. This use of superposition avoids having to repeat the many points needed to specify the curved pole tip.

In Figs. 6-9 are shown pictures of the mesh for this problem. Figures 10-13 show pictures of the same problem with the zoning changed to right triangles. Figures 14-17 show the effect of using constant weights ($w_i = 1$). Notice the tendency of the mesh lines to crowd together near the lower corners of the right-hand coil (compare Fig. 17 with Figs. 9 and 13).

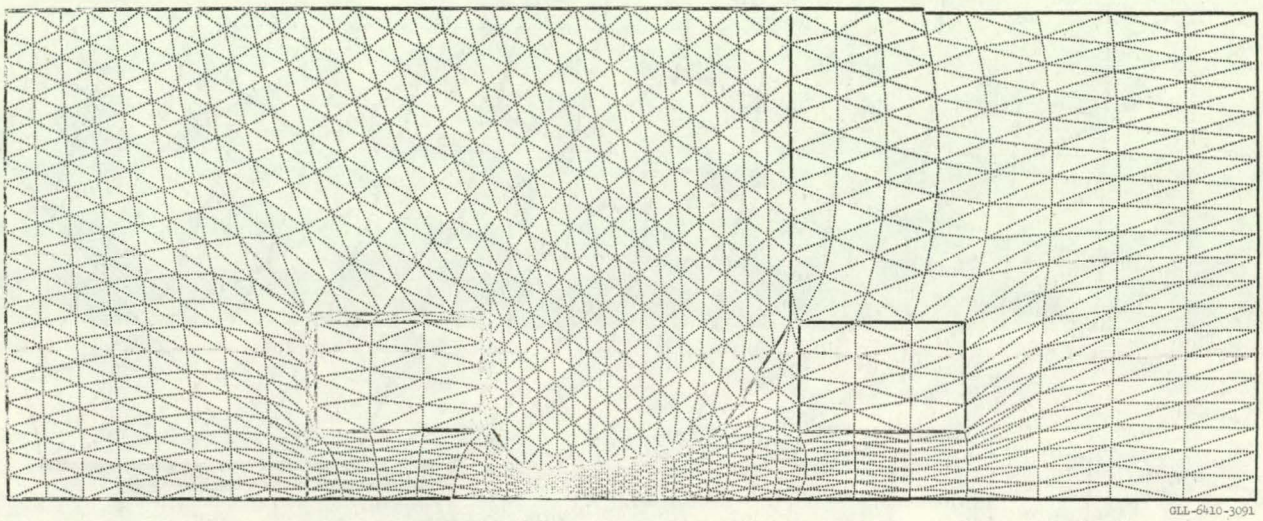


Fig. 6. C magnet showing triangle zones, equilateral-triangle zoning.

The second sample problem, an H magnet, illustrates the use of the elliptical arc routine mentioned in section IV. An arc is made up of two straight lines in the logical diagram, one slanting and one lying along a row. It is specified by giving the sentinel ⊙100000 (⊙ is the letter O, 0 is zero) followed by: $(\ell, k, x, y)_1$, $(\ell, k)_0$, θ, $(\ell, k, x, y)_2$ where subscripts 1 and 2 refer to the first and last point of the arc, subscript 0 refers to the logical center of the arc, and θ is the logical angle given by

$$\theta = \frac{\text{number of triangle sides along logical slant line}}{\text{total number of triangle sides along arc}} .$$

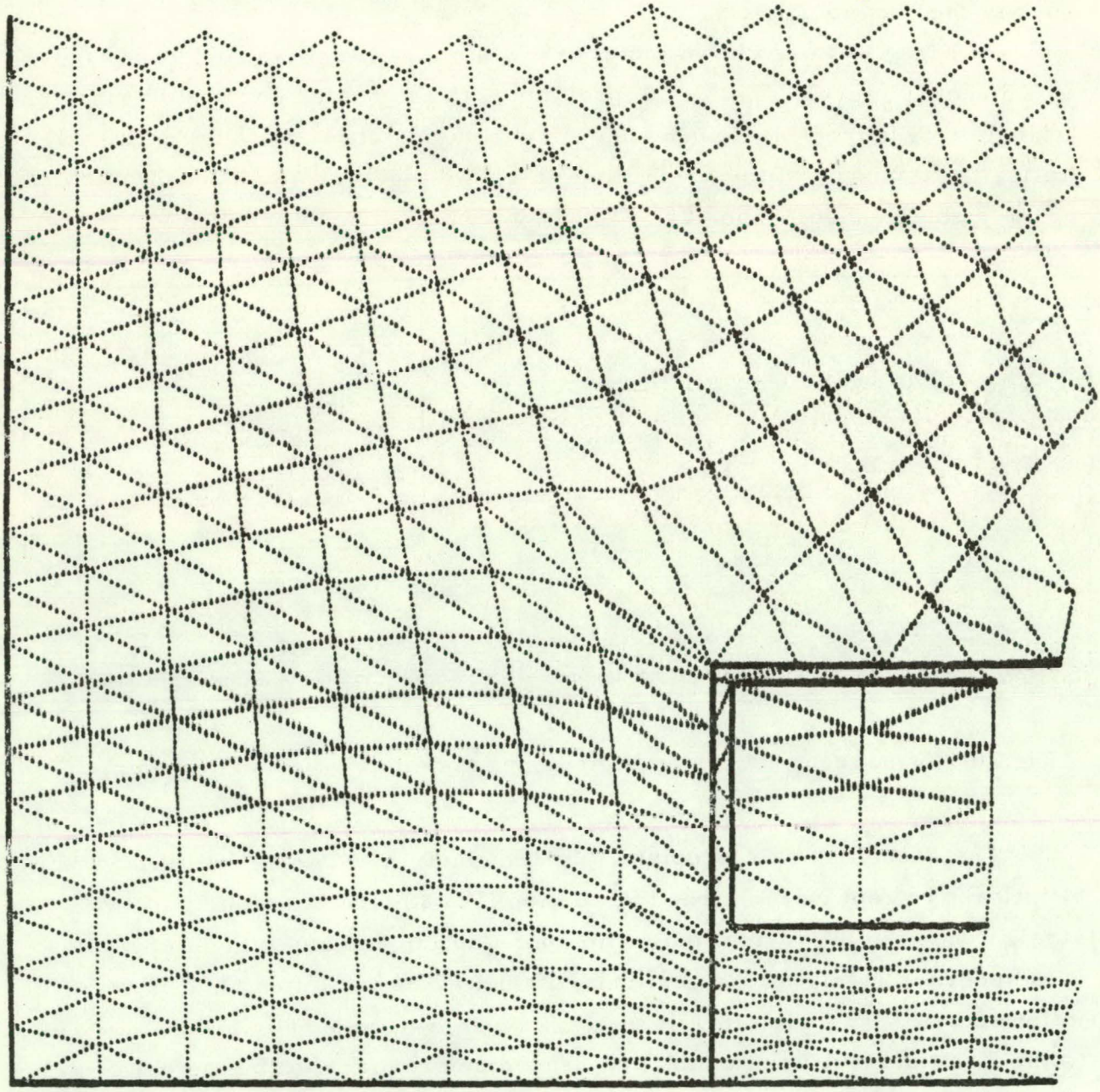


Fig. 7. C magnet, equilateral triangles, left side.

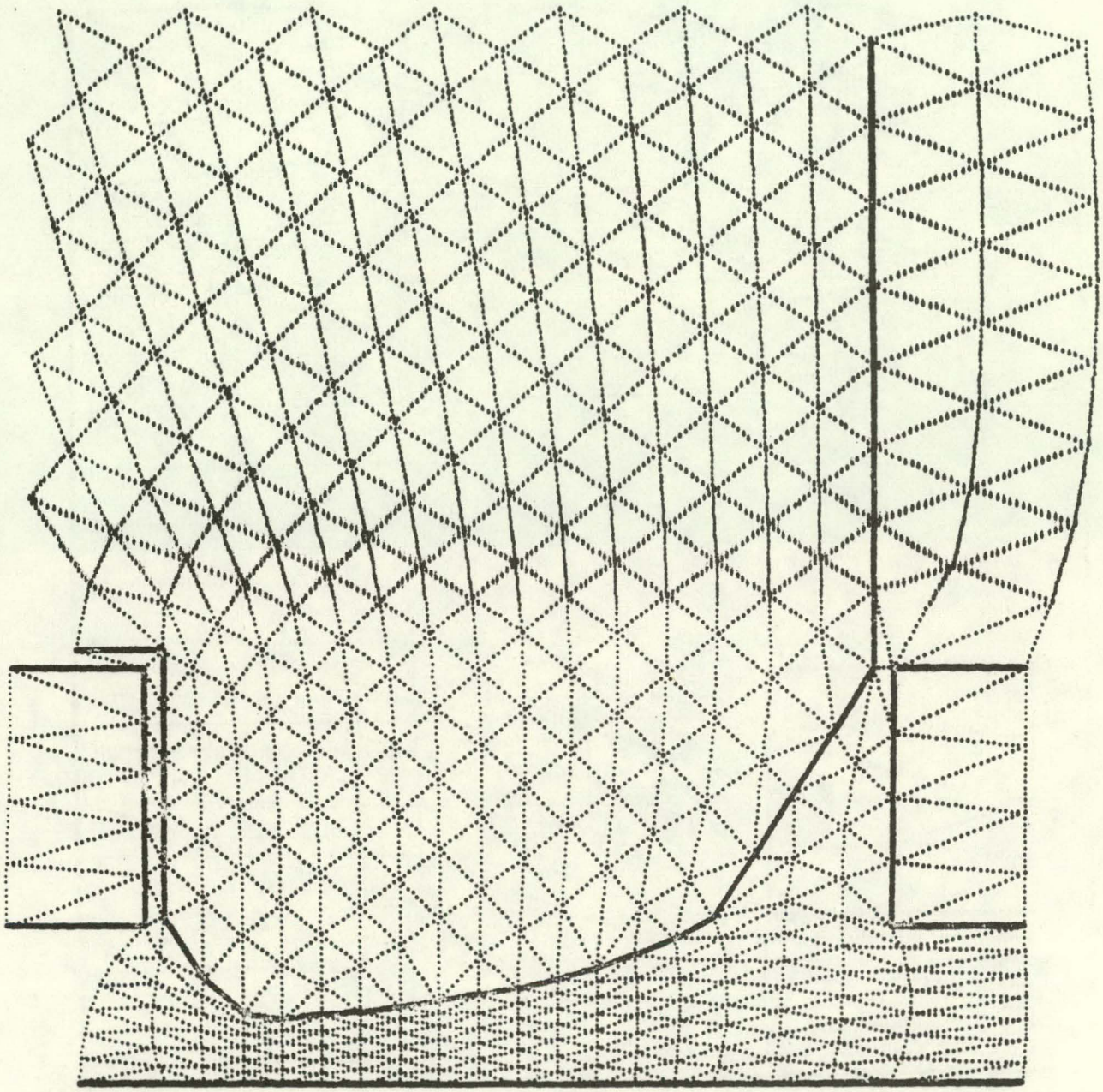


Fig. 8. C magnet, equilateral triangles, center.

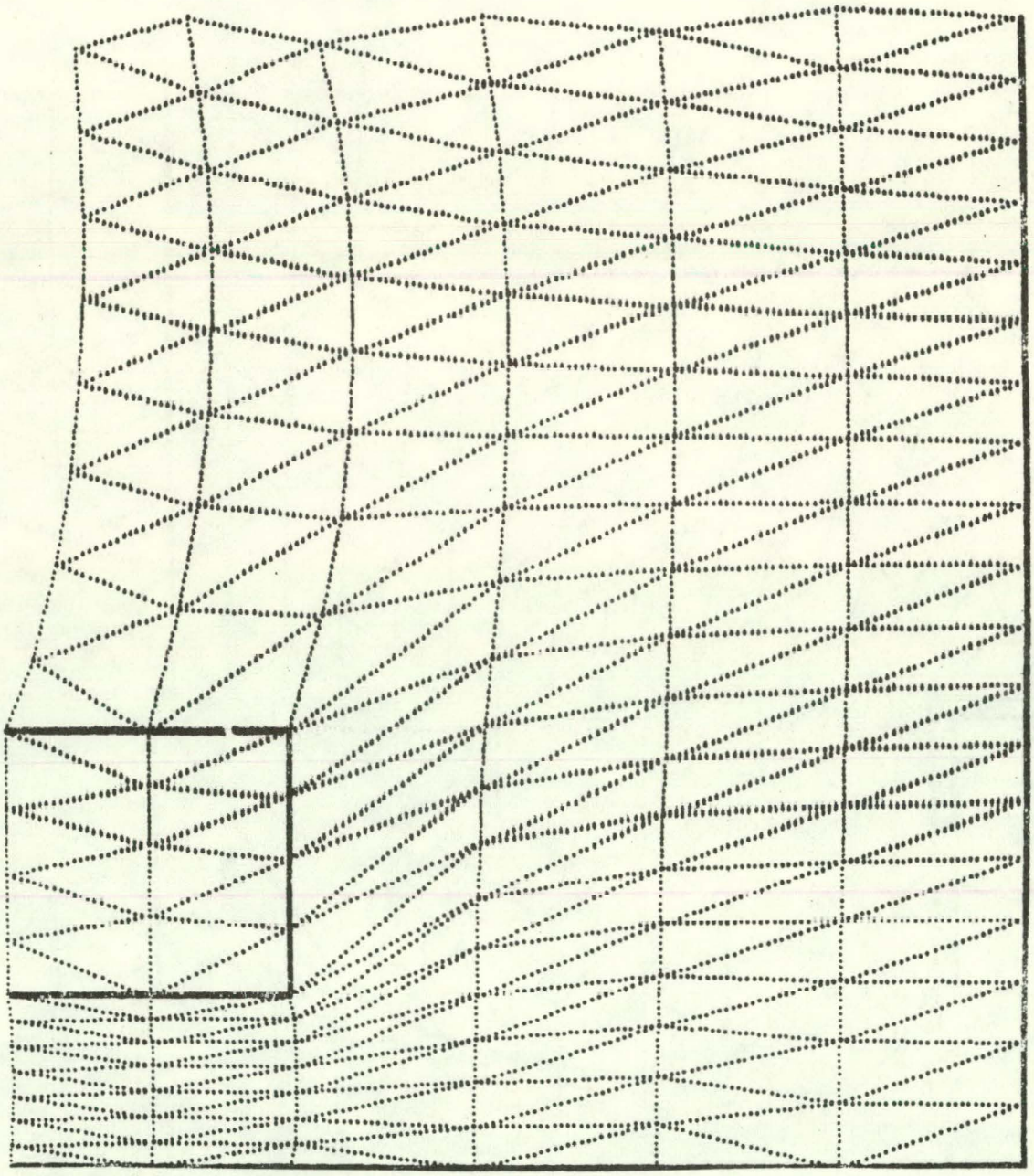


Fig. 9. C magnet, equilateral triangles, right side.

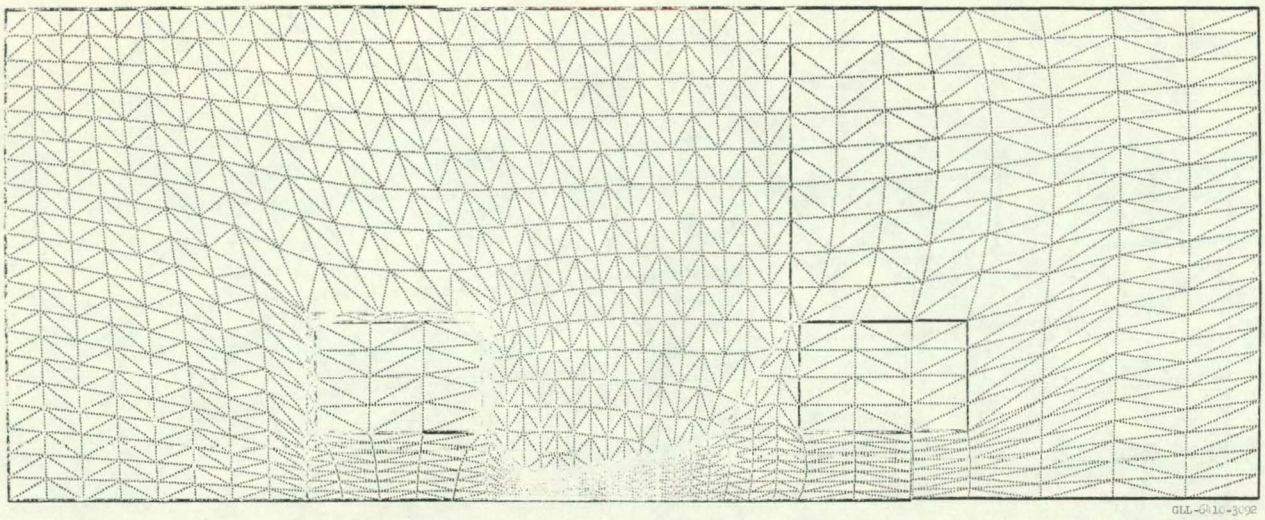


Fig. 10. C magnet, right-triangle zoning.

Figure 18 shows the material interfaces for the H magnet, and Fig. 19 the logical mesh. The card input for this problem reads as follows:

	Card	Input
	1	TRIM HMAGR1
	2	-1 A4A21A28 Y
Region 1	3	A1 D2 + 0 + 0 A1
	4	D0A0 + 0 + 0 D21A0 + 40 + 0 D21A28 + 40 + 70 D0 A28 + 0 + 70 D0A25 + 0 + 60 D0A24 + 0 + 58
	5	D0A16 + 0 + 46 D0A12 + 0 + 24 D0A4 + 0 + 12 D0A3 + 0 + 10
	6	-0 Y
Region 2	7	A2 D1 +0 + 0 A1
	8	D0A3 + 0 + 10 A17A3 + 30 + 10 ⊙100000 D17A6 + 30 + 16 D17A14 + 1.0472 D9A14 + 15 + 35
	9	⊙100000 D9A14 + 15 + 35 D17A14 + 0.6667 D17A22 + 30 + 54
	10	D17A25 + 30 + 60 D0A25 + 0 + 60 D0A24 + 0 + 58 D0A16 + 0 + 46 D0A12 + 0 + 34 D0A4 + 0 + 12
	11	D0A3 + 0 + 10
	12	-0 Y

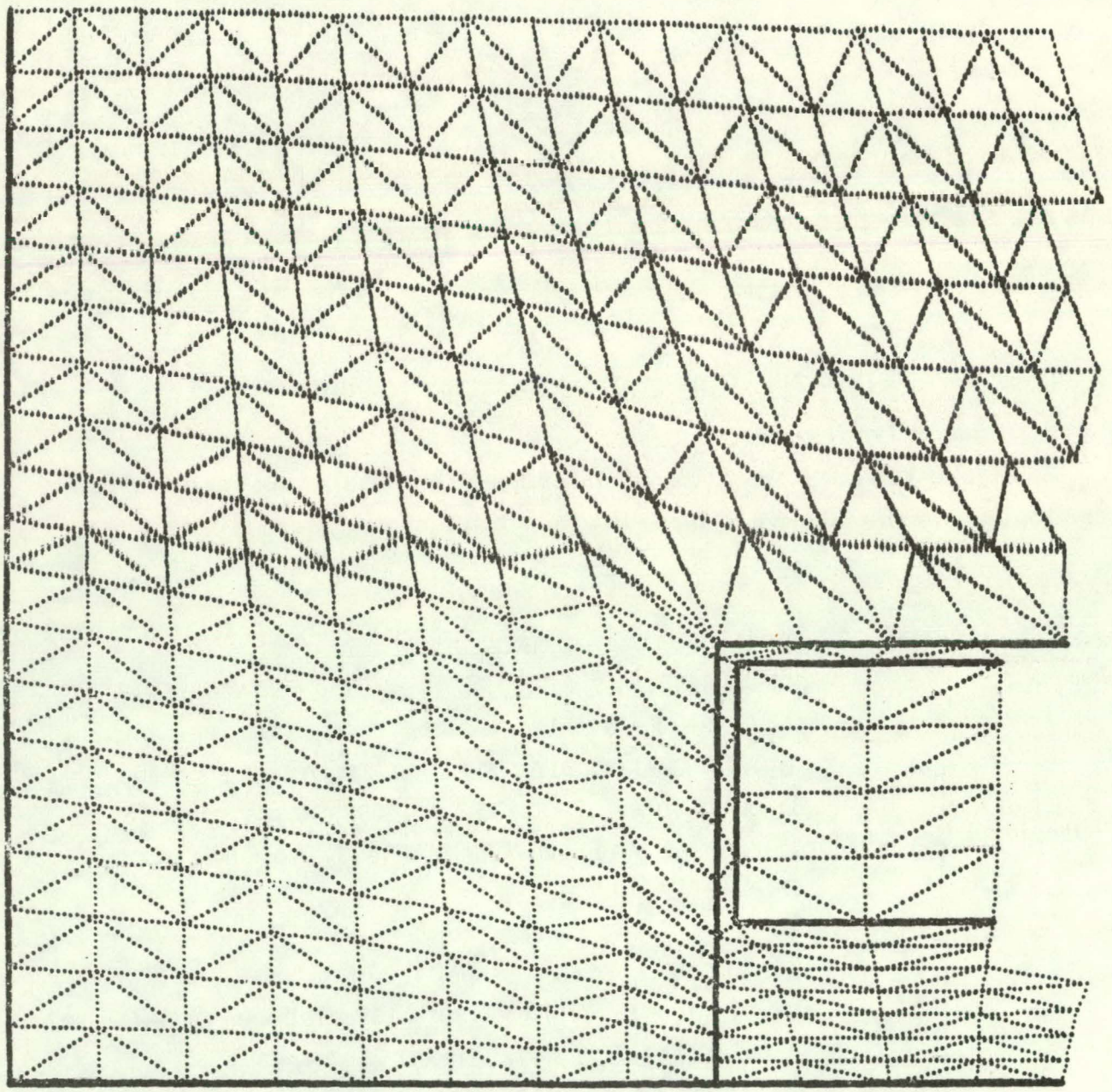


Fig. 11. C magnet, right triangles, left side.

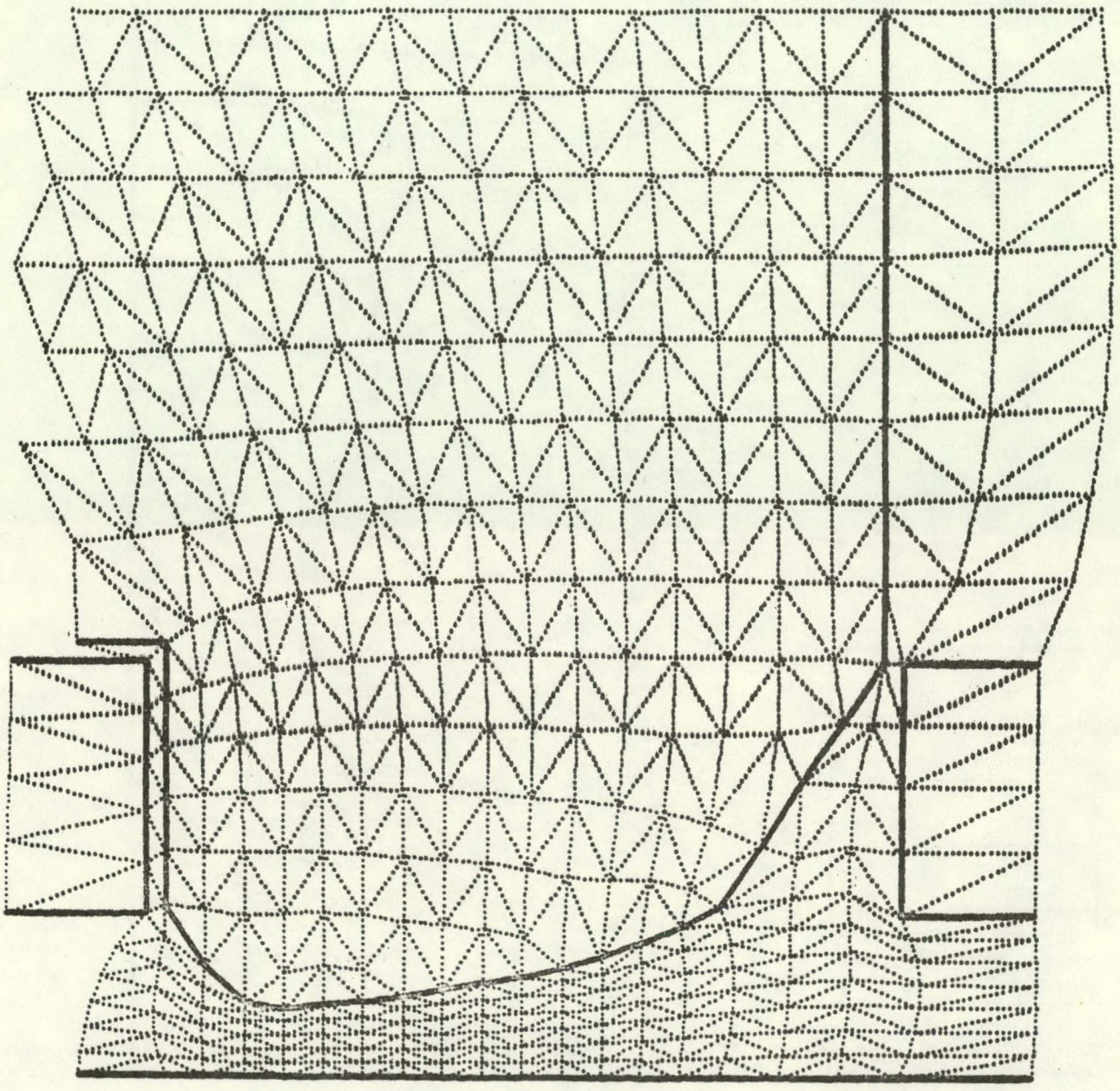


Fig. 12. C magnet, right triangles, center.

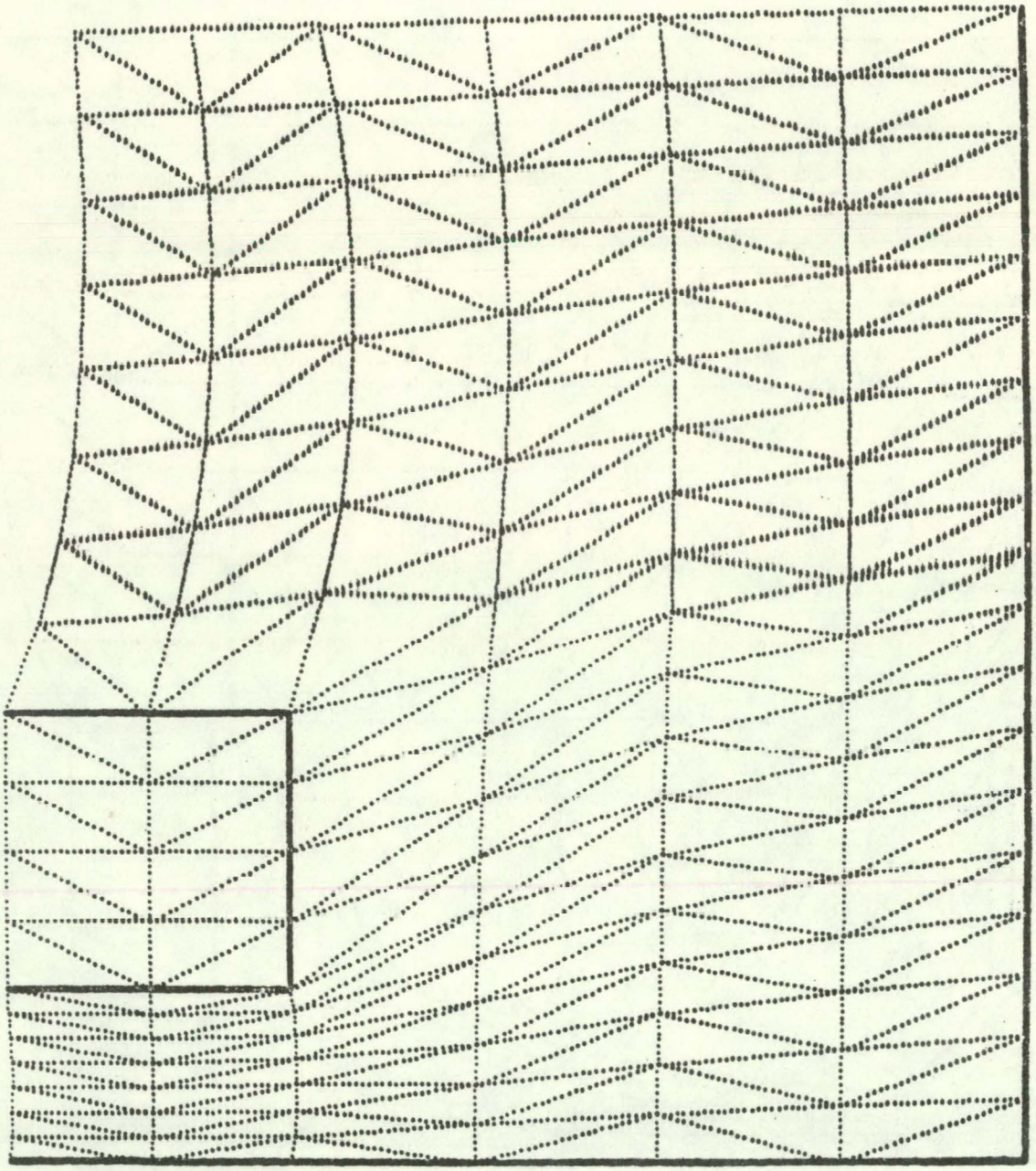


Fig. 13. C magnet, right triangles, right side.

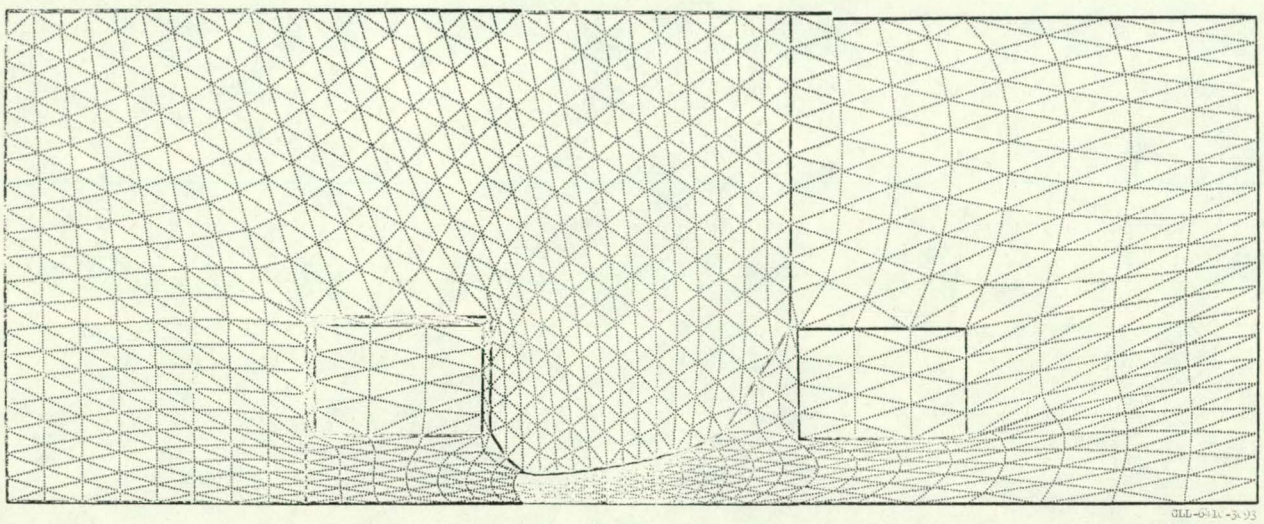


Fig. 14. C magnet, equal-weight zoning.

	Card	Input
Region 3	13	A3 D1 - 100 + 0 A1
	14	⊙100000 D0A4 + 0 + 12 D0A8 +1.0472 D4A8 + 6 + 18
	15	⊙100000 D4A8 + 6 + 18 D0A8 + 1.0472 D0A12 + 0 + 24
	16	-0 Y
Region 4	17	A4 D1 +100 + 0 A1
	18	⊙100000 D0A16 + 0 + 46 D0A20 + 0.6667 D4A20 + 6 + 52
	19	⊙100000 D4A20 + 6 + 52 D0A20 +0.6667 D0A24 +0 + 58
	20	-0 Y

Figures 20-23 show the completed mesh, with equilateral triangle zoning.

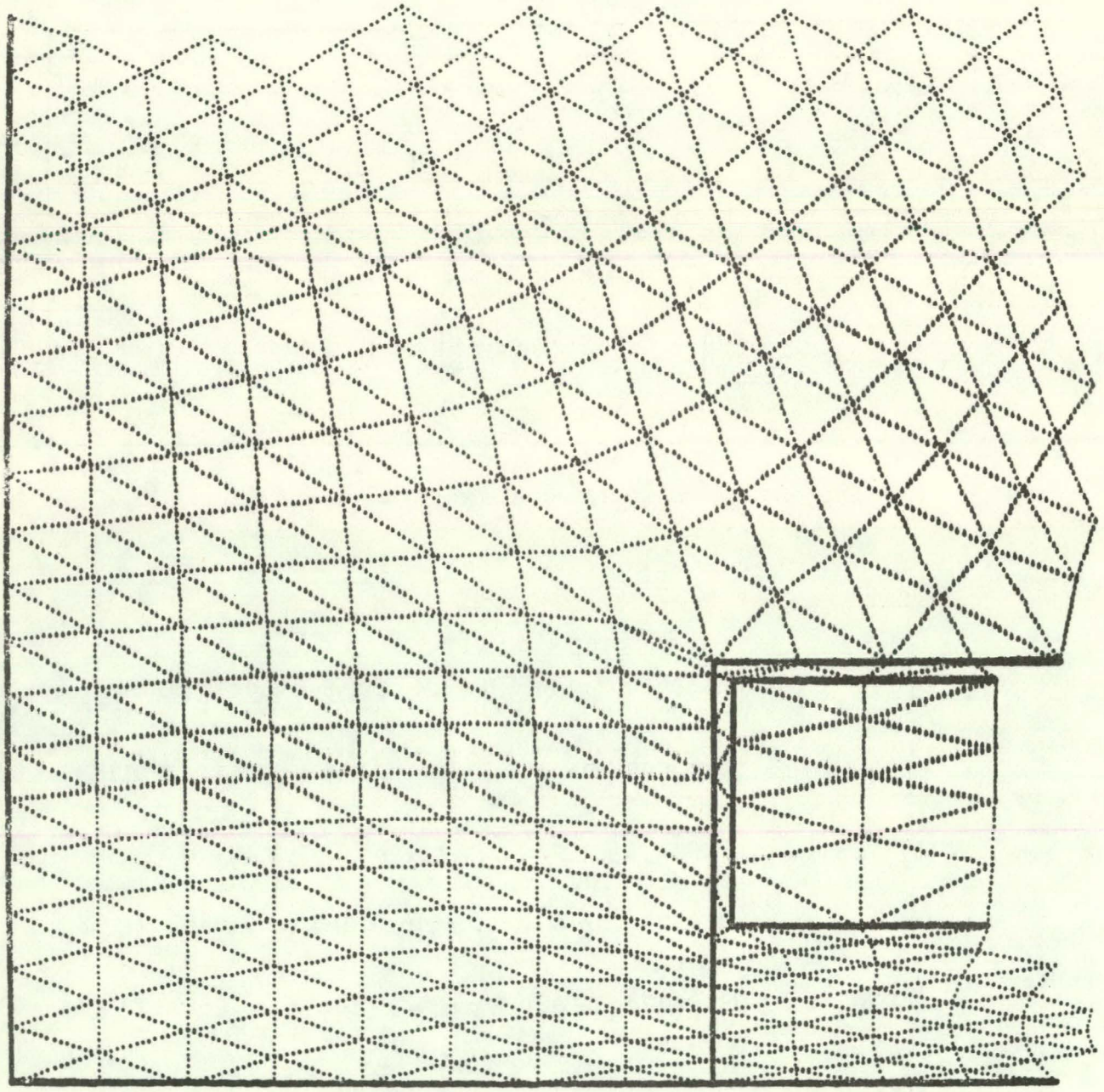


Fig. 15. C magnet, equal-weight zoning, left side.

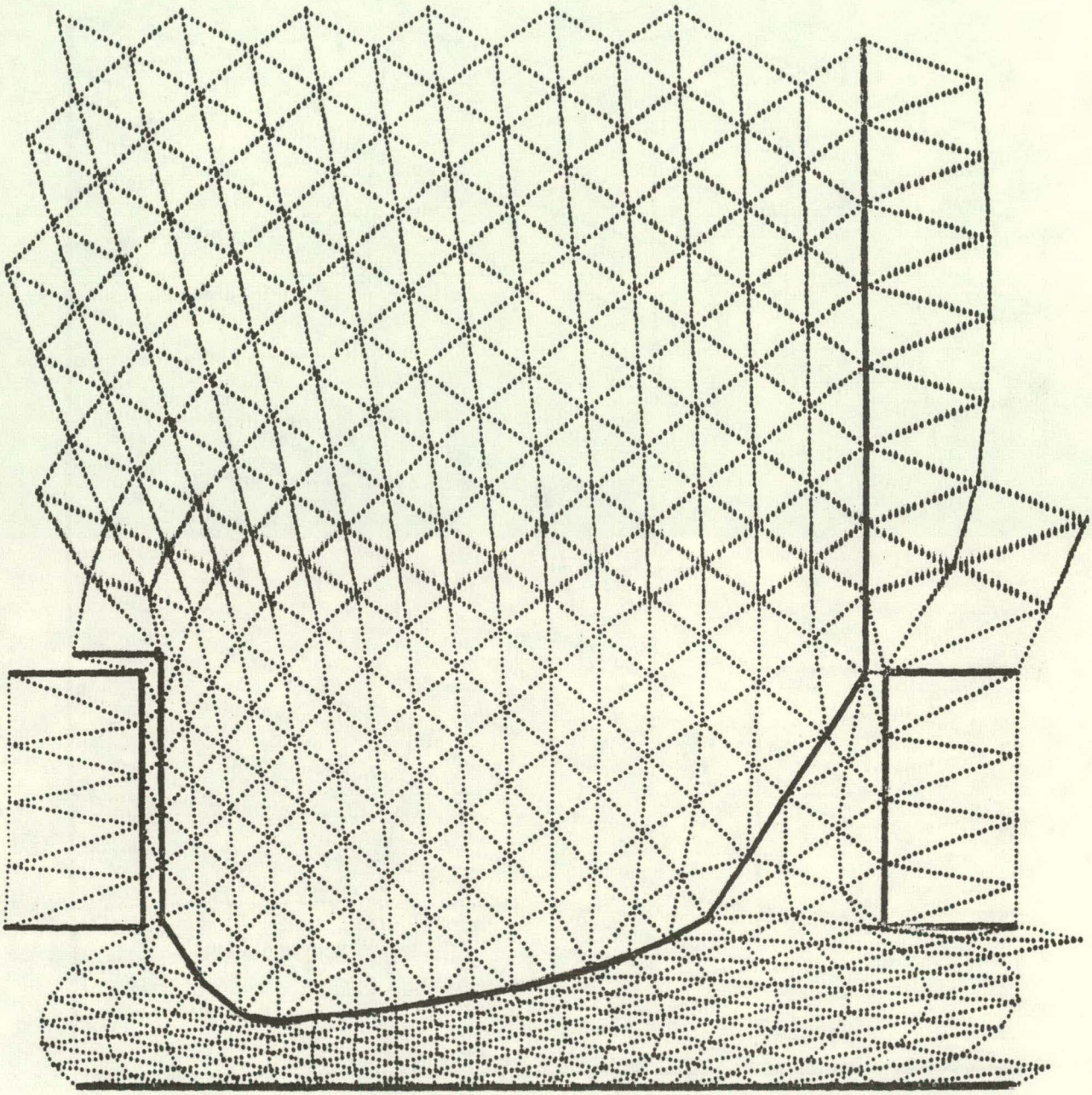


Fig. 16. C magnet, equal-weight zoning, center.

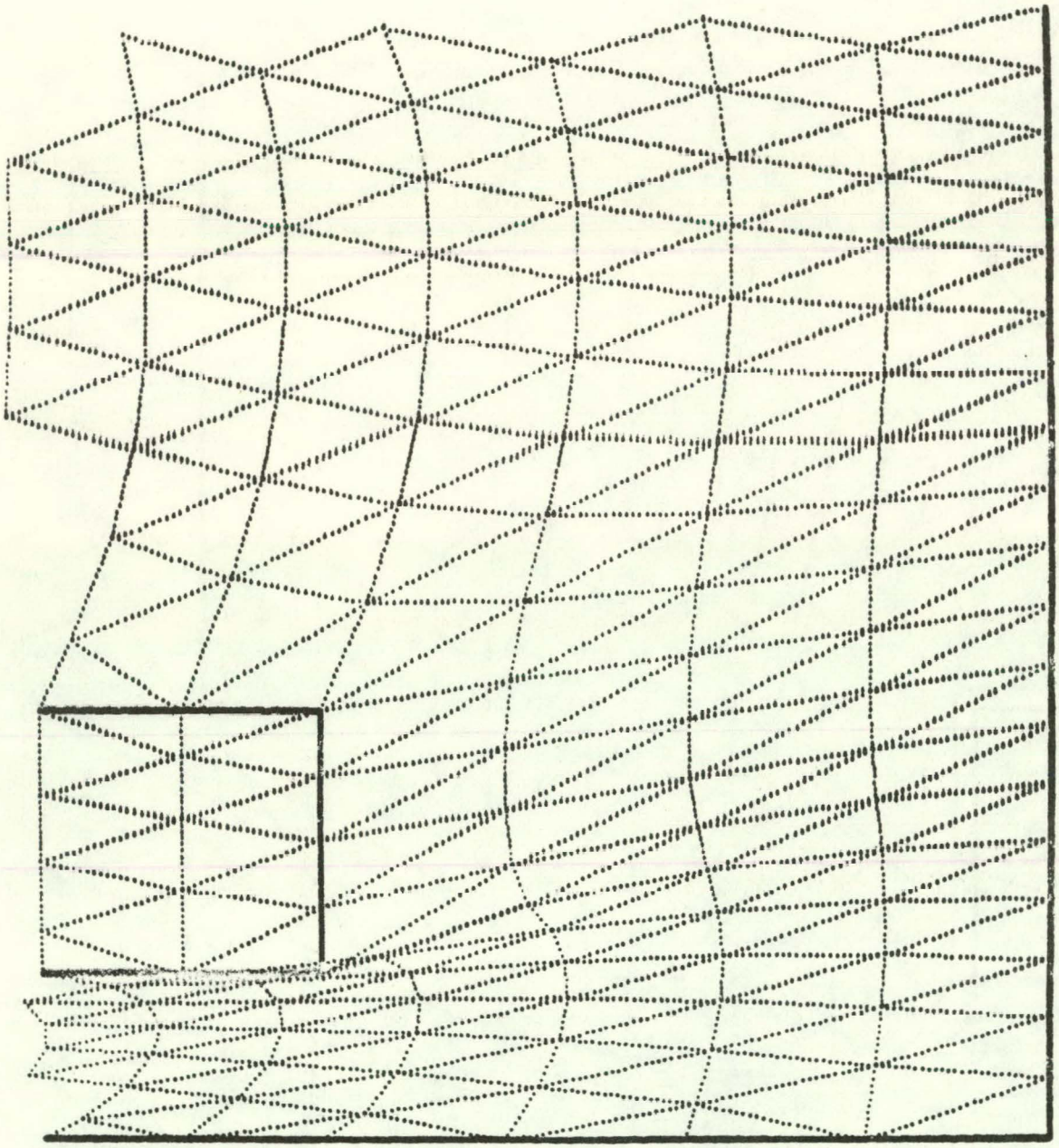


Fig. 17. C magnet, equal-weight zoning, right side.

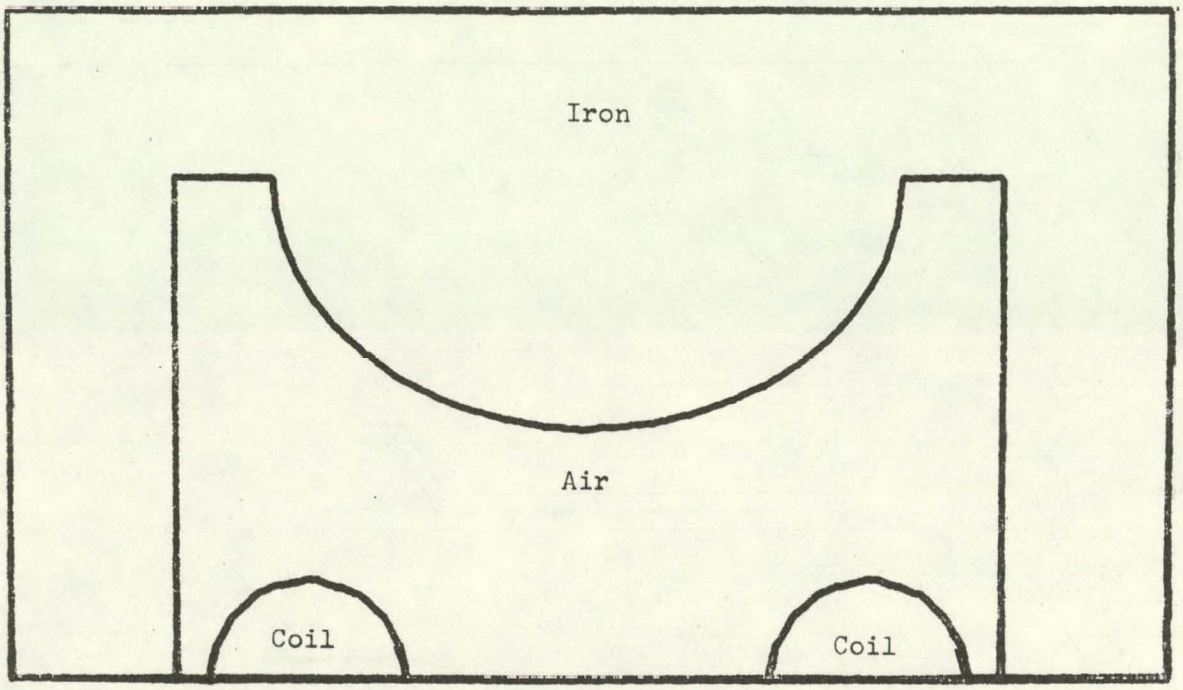


Fig. 18. H magnet showing material interfaces.

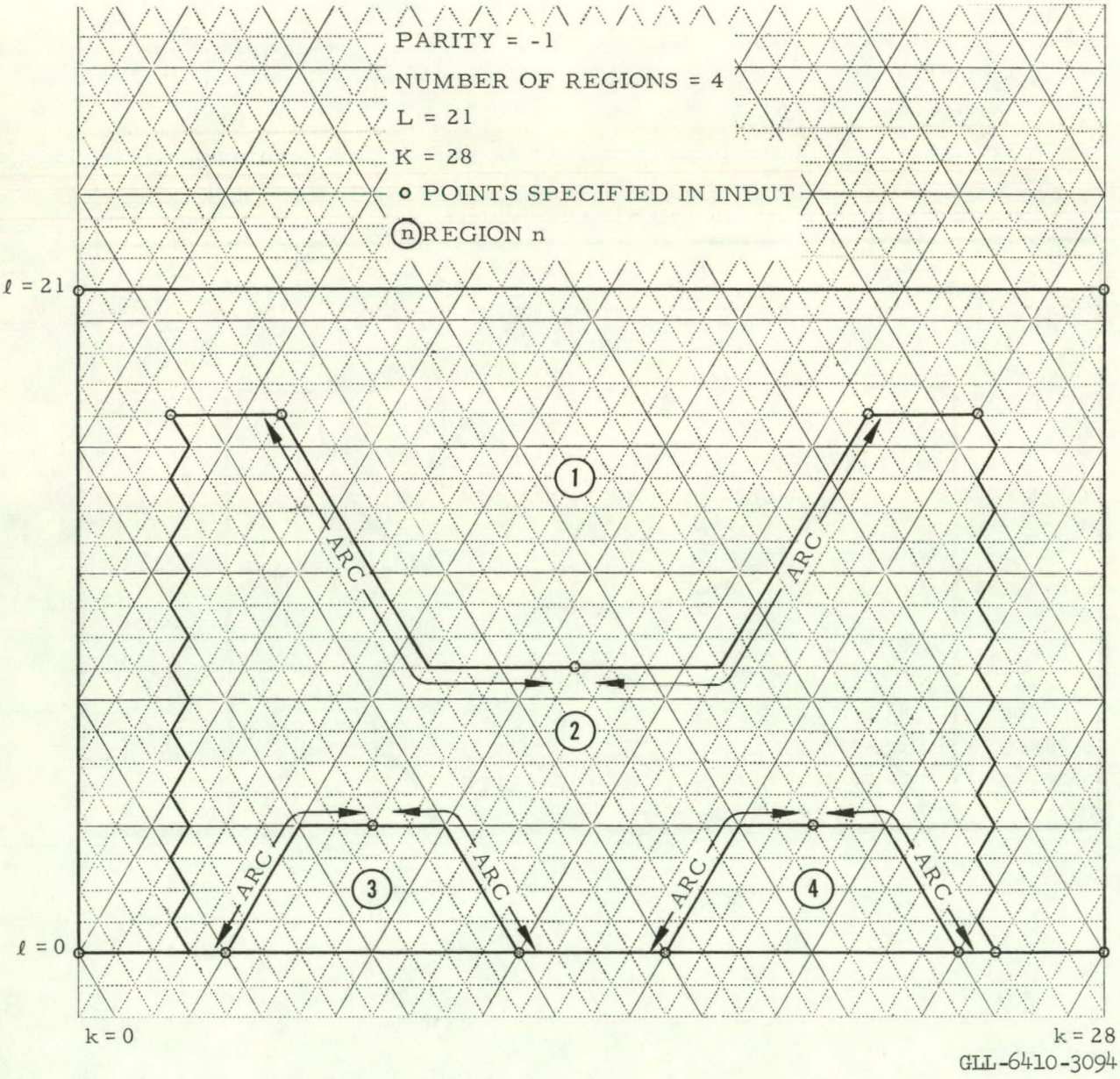


Fig. 19. Logical diagram of H magnet mesh.

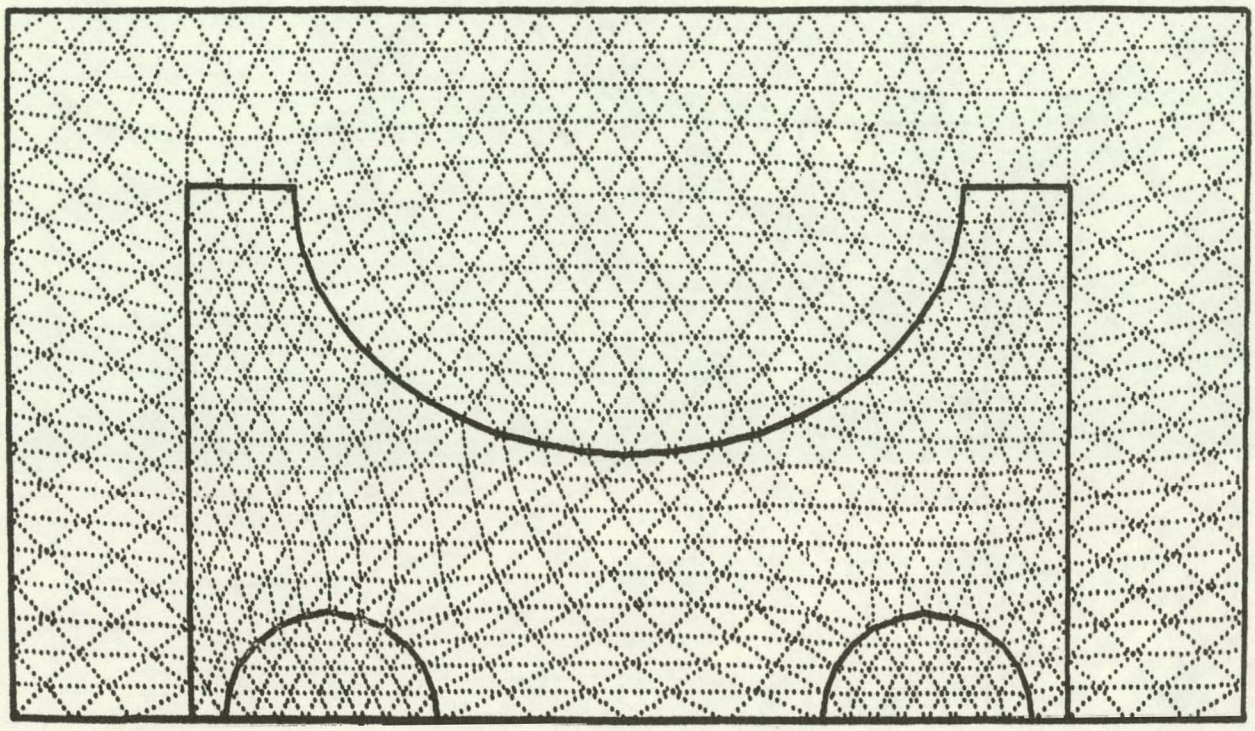


Fig. 20. H magnet, equilateral-triangle zoning.

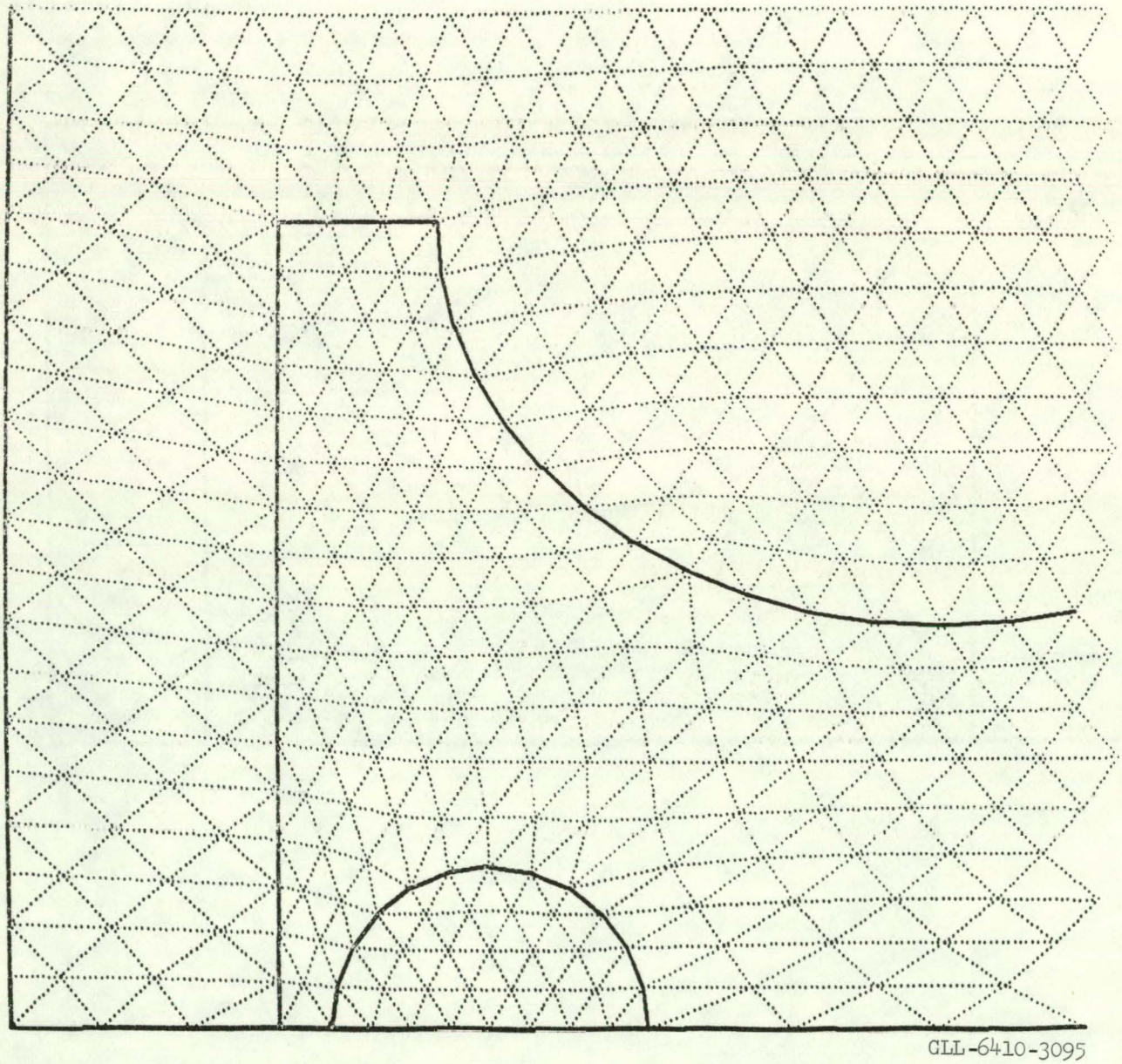


Fig. 21. H magnet, equilateral triangles, left side.

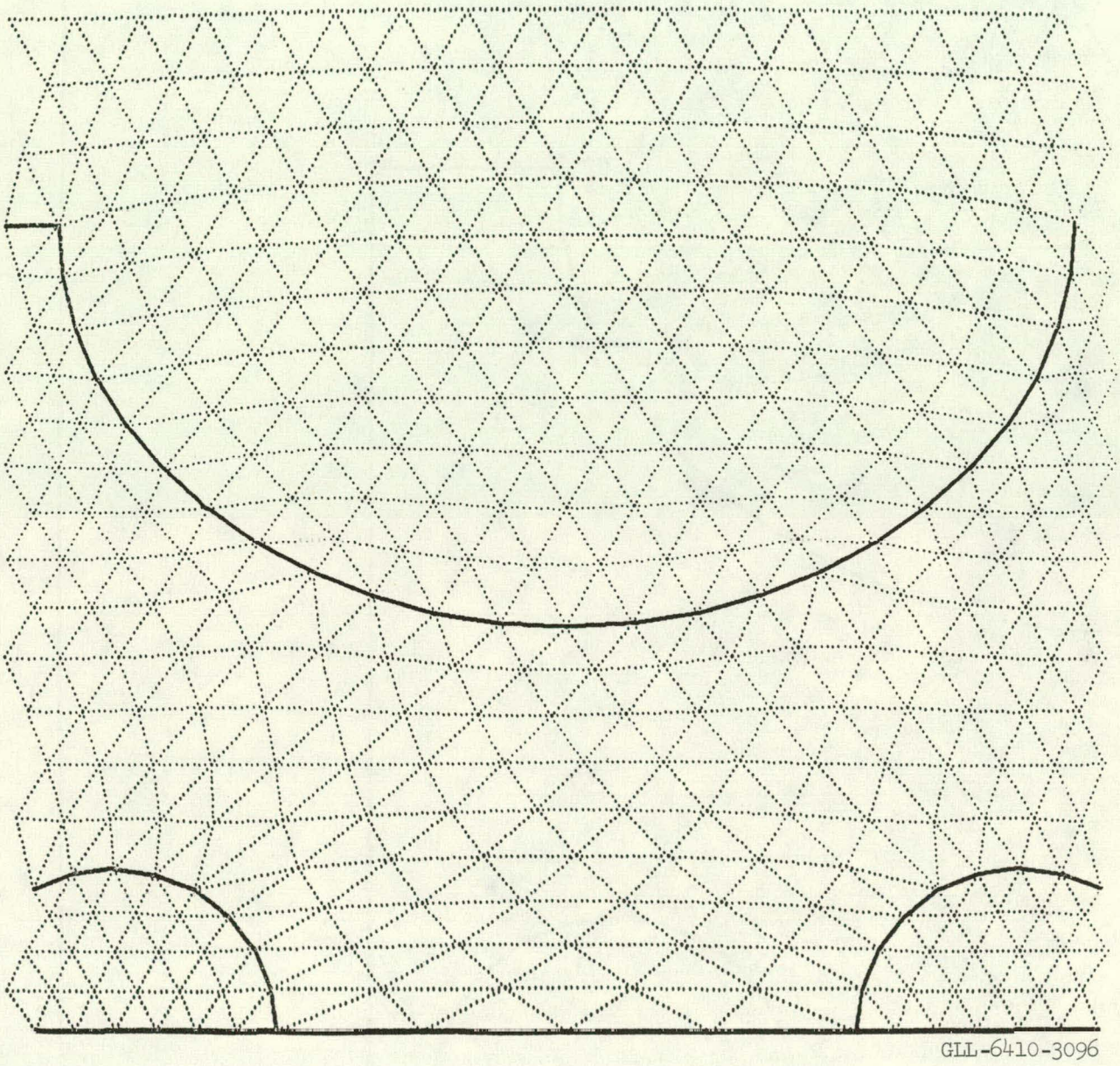


Fig. 22. H magnet, equilateral triangles, center.

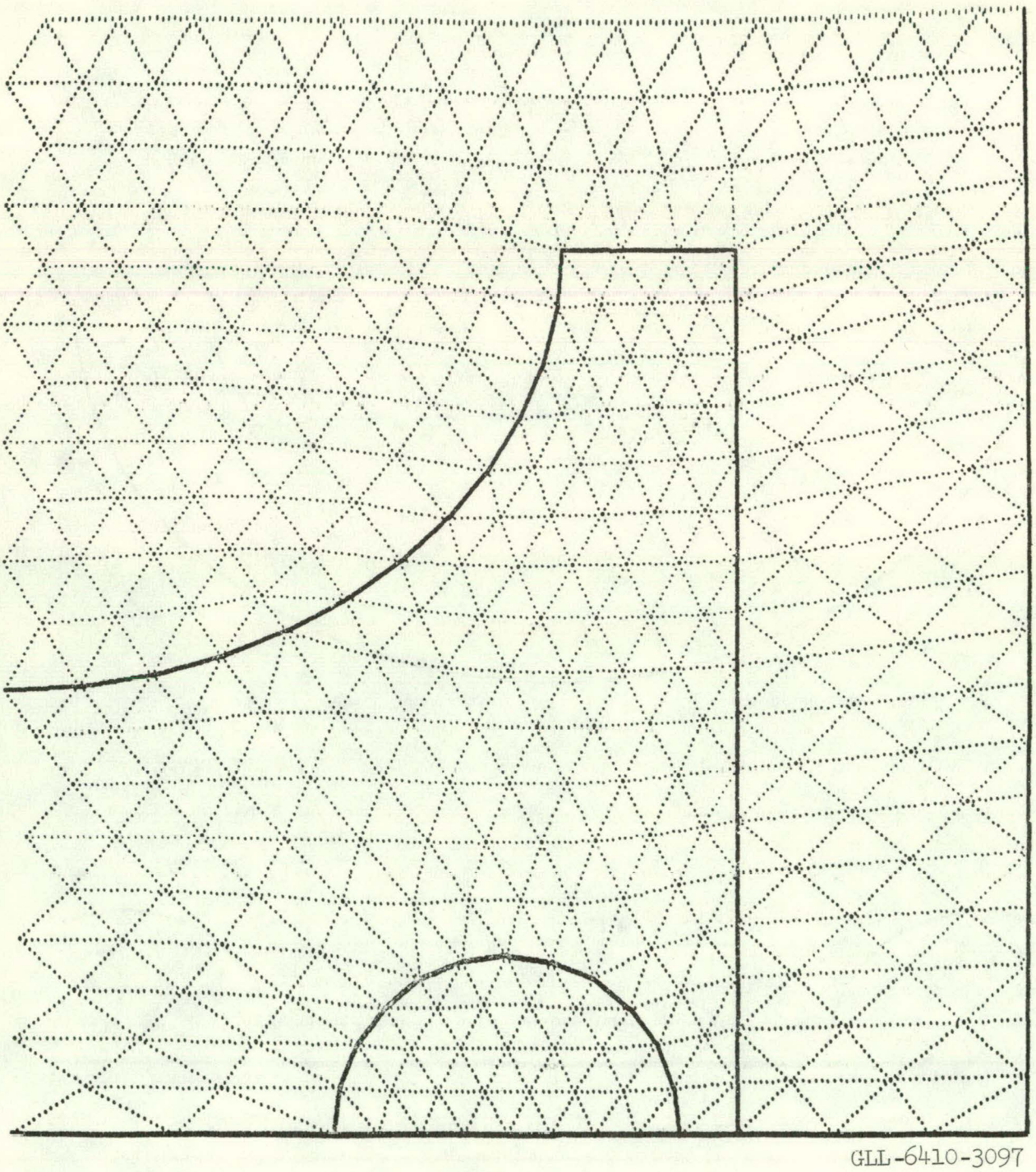


Fig. 23. H magnet, equilateral triangles, right side.

APPENDIX

The calculation of the coordinates of non-boundary mesh points is done by the "equipotential" method.[2] This gives the result that the coordinates x, y of a point satisfy the different equations

$$\sum_i w_i(x_i - x) = 0$$
$$\sum_i w_i(y_i - y) = 0, \qquad \text{(A.1)}$$

where the sum is over the six nearest neighbors x_i, y_i of each point x, y, and the weights w_i are defined as follows: Let

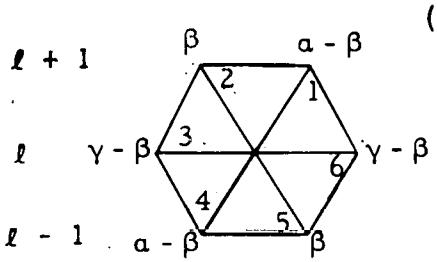
$$\alpha = x_\psi^2 + y_\psi^2$$
$$\beta = x_\phi x_\psi + y_\phi y_\psi$$
$$\gamma = x_\phi^2 + y_\phi^2,$$

where x_ϕ, x_ψ, $y_\phi y_\psi$, are defined below in two different ways, tending to produce equilateral or right triangles. The weights w_i are functions of α, β, and γ and are shown in Fig. A.1 attached to their respective vertices.

A reflection boundary condition can be imposed on the equipotential zoning. Let us assume that the plane $x = x_0$ represents the boundary, and the parity is negative, so that we have a condition as shown in Fig. A.2. It can be seen that the reflecting plane cuts across rows, and, as in the right-hand edge of Fig. 1, mesh points on this plane have alternately one and three couplings to internal mesh points. This is different from the situation with positive parity, where the reflecting plane is itself a row, like the row $\ell = 0$ in Fig. 1, and each mesh point on the reflecting plane has two internal couplings. In the latter case, where the boundary is a logical straight line, we have a logically "smooth" boundary, whereas with negative parity the boundary is logically "rough." (With reflection in a plane $y = y_0$ the situation is reversed.)

A smooth boundary can always be interpreted as a reflecting plane, and no changes in the zoning equations are needed. With a rough boundary, however, we impose a reflecting condition by assuming that the reflecting plane

(a) Equilateral Triangles

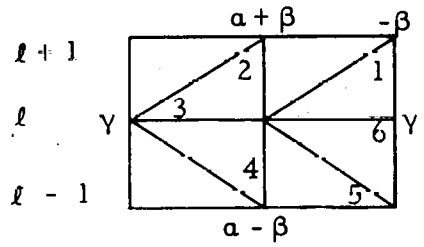


$$x_\phi = \frac{1}{6}\left[(x_2 + 2x_1 + x_6) - (x_3 + 2x_4 + x_5)\right]$$

$$x_\psi = \frac{1}{6}\left[(x_1 + 2x_6 + x_5) - (x_2 + 2x_3 + x_4)\right]$$

(b) Right Triangles

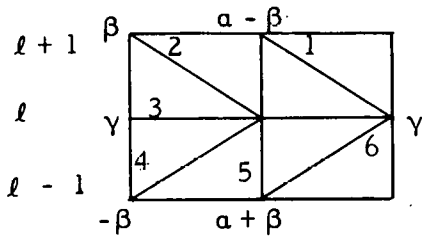
ℓ = even:



$$x_\phi = \frac{1}{6}\left[(x_1 + 2x_2) - (x_5 + 2x_4)\right]$$

x_ψ = as above

ℓ = odd:



$$x_\phi = \frac{1}{6}\left[(x_2 + 2x_1) - (x_4 + 2x_5)\right]$$

x_ψ = as above

Fig. A.1. Weights used for triangle zoning.

cuts through triangles as shown, so that the triangles which have their bases on the reflecting plane are really only half-triangles.

Consider the difference equation

$$x = \frac{\sum_{i=1}^{6} w_i x_i}{\sum_{i=1}^{6} w_i}$$

for the coordinate x at the point ℓ, k shown in Fig. A.2. The reflection condition is

$$x_6 - x_0 = x_0 - x$$

or

$$x_6 = 2x_0 - x.$$

Substituting in the above equation, we get

$$x = \frac{w_1 x_1 + \ldots + w_5 x_5 + 2w_6 x_0}{w_1 + \ldots + w_5 + 2w_6}.$$

Thus the effect of reflection on x is simply to double the weight to the boundary of the point being reflected. The weights themselves are calculated as usual.

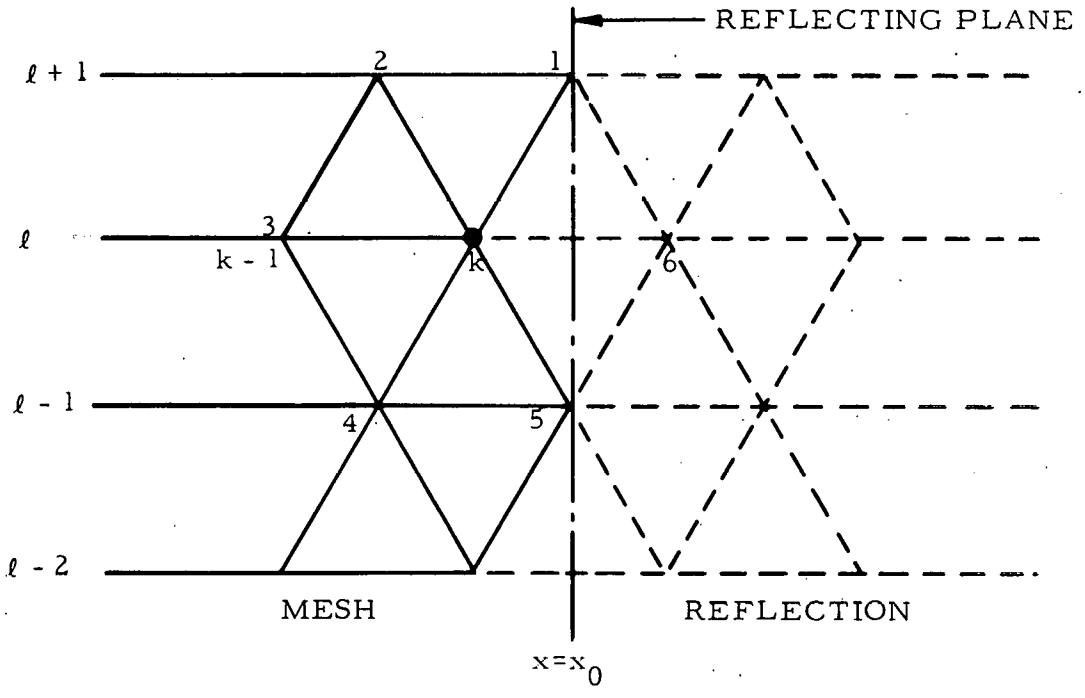


Fig. A.2. Reflection of mesh at a "rough" boundary.

The reflecting condition for y is

$$y_6 = y$$

This leads to

$$y = \frac{w_1 y_1 + \ldots + w_5 y_5}{w_1 + \ldots + w_5} .$$

We see that in the y equation the weight to the boundary point is zero.

We solve (A.1) by the method of successive overrelaxation, which gives for the (n + 1)st cycle

$$x^{n+1} = x^n + \frac{\rho}{\sum_i w_i^n} \left[\sum_i w_i^n (x_i^n - x^n) \right] \tag{A.2}$$

and

$$y^{n+1} = y^n + \frac{\rho}{\sum_i w_i^n} \left[\sum_i w_i^n (y_i^n - y^n) \right], \tag{A.3}$$

provided that the Jacobian

$$J = x_\psi y_\phi - x_\phi y_\psi$$

is not zero at the given point (if J = 0, the point is skipped). Here ρ ($1 < \rho < 2$) is the overrelaxation factor. The weights w_i^n, which are functions of the nearest neighbor coordinates x_i^n, y_i^n, are recalculated at each mesh point. New values x^{n+1}, y^{n+1} are substituted immediately for old values x^n, y^n.

The rate of convergence has a maximum when ρ assumes its optimum value ρ_{opt}, which usually changes somewhat in the course of the iteration. It is simple in practice to recalculate ρ each cycle, and the following scheme has successfully optimized ρ (we give the equations for (A.2) only; the method is the same for (A.3):

For a given value of $\rho = \rho^n$, the rate of convergence is defined to be

$$\eta^n = \sqrt{\frac{\sum_i \left(x_i^{n+1} - x_i^n \right)^2}{\sum_i \left(x_i^n - x_i^{n-1} \right)^2}} \tag{A.4}$$

summed over the whole mesh. Then theory[5] gives for the optimum value

$$\rho_{opt} = \frac{2}{1 + \sqrt{1 - \lambda^2}}, \tag{A.5}$$

where we obtain λ from the equation

$$\lambda = \frac{\rho^n + \eta^n - 1}{\rho^n \sqrt{\eta^n}}. \tag{A.6}$$

The value ρ^{n+1} to be used for the $(n+1)$st cycle is a weighted mean of ρ^n and ρ_{opt} given by

$$\rho^{n+1} = \beta(\rho_{opt} - \rho_0) + (1 - \beta)\rho^n. \tag{A.7}$$

Equations (A.4) to (A.7), together with an initial trial value of ρ, define the procedure for optimizing ρ for the calculation of x. Replacing x_i, x by y_i, y in (A.4) gives an analogous ρ to be used in (A.3).

If η is found to be ≥ 1 at any time, ρ^n is not changed. In general, ρ^n is changed only when the condition $\eta < 1$ has obtained for at least five successive cycles.

It is important that β not be too large, or the optimization procedure will tend to be unstable, slowing down or preventing convergence. Too small a value of β will also slow down the convergence. In practice, $\beta = 0.05$ gives good results.

The constant ρ_0 in (A.7) is an empirical term which is only needed when ρ exceeds ρ_{opt}. In this case, ρ_{opt} cannot be properly calculated by (A.5) and (A.6) and ρ tends to increase instead of decreasing. Setting $\rho_0 = 0.01$ is sufficient to overcome this tendency. When ρ is less than ρ_{opt}, it will increase despite the presence of ρ_0, approaching the value $\rho_{opt} - \rho_0$.

The criterion used for convergence of the x coordinates is the smallness of the ratio

$$\epsilon_x = \sqrt{\frac{\sum_i \left(x_i^{n+1} - x_i^n\right)^2}{\sum_i \left(x_i^{n+1}\right)^2}} \tag{A.8}$$

summing over all non-boundary points. An analogous expression is used for the convergence of the y coordinates. For convergence we require that

$$\epsilon_x < \epsilon_0$$

$$\epsilon_y < \epsilon_0$$

for five successive cycles, where ϵ_0 is usually 10^{-5} or 10^{-6}. Note that this criterion is independent of the dimensions of the mesh.

An alternative set of weights (for equilateral triangles) is obtained by simply using

$$w_i = 1 \qquad i = 1, 2, \ldots 6$$

for all points. For many problems these weights give nearly the same results as the ones shown above, and the calculating speed is about three times faster. However, for regions which have concave boundaries, constant weights cause the points to be placed too close to the boundary or even to fall outside the region (see p. 26). We customarily use constant weights for preliminary relaxation ($\epsilon_0 = 10^{-2}$ or 10^{-3}) followed by use of the variable weights.

All problems run so far have converged except when variable weights are used from the beginning and the initial value of ρ is greater than ρ_{opt}. The procedure mentioned in the preceding paragraph prevents this difficulty. It could also be prevented by using an initial value of ρ that is less than ρ_{opt}.

REFERENCES

[1] A. M. Winslow, "Numerical Calculation of Static Magnetic Fields in an Irregular Triangle Mesh," UCRL-7784 (1964).

[2] A. M. Winslow, "'Equipotential' Zoning of Two-Dimensional Meshes," UCRL-7312 (1963).

[3] Ref. 1, Section II.

[4] The variables may also be stored by mesh points, where all the quantities for k = 0 are stored followed by all the quantities for k = 1, etc.

[5] G. Forsythe and W. Wasow, Finite Difference Methods for Partial Differential Equations, 1960, Sec. 22.

/no

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.