A TWO-DIMENSIONAL FINITE ELEMENT MESH GENERATOR WITH AUTOMATIC TRANSITIONING CAPABILITY

by

C. C. Jara-Almonte

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

in

Mechanical Engineering

APPROVED:

C. E. Knight, Chairman

L. D. Mitchell

H. H. Mabie

December, 1982

Blacksburg, Virginia

ACKNOWLEDGEMENTS

The author wishes to acknowledge Dr. C. E. Knight for directing her research and serving as her advisor and Jaime Jara-Almonte for his help and support.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1. Introduction

Finite element analysis often involves the creation of complicated meshes involving many nodes and elements. To properly input the mesh information into the computer may require a great deal of time and effort from the analyst. In many cases, the use of a mesh generator can reduce the size of the input file and the time required to create this file. Generally, a mesh generator is a program that allows for the specification of multiple nodes and elements with a single entry or a combination of entries. This program may be part of a finite element code or it may be a separate pre-processor that produces the input file for the finite element program. The problem in creating a mesh generator is to design a program that is extremely flexible, requires minimal input to produce a satisfactory mesh, and does not require the user to know the location of every node or the specification of every element before the mesh is created.

Some of the more commonly used mesh generation techniques include string generation, a simplified finite-difference approach, simple incrementation, and multiple regions. String generation involves the use of straight-line interpolation, sometimes with nonuniform spacing, to determine the location of nodal points. The finite-difference approach requires that one specify key geometry boundary points of a grid in an integer space with correspondence to the real geometry space main domain. Boundary nodal points are located using linear interpolation while interior points are found by solving Laplace's equation in two dimensions. Simple incrementation is often used to generate a one-, two-, or three-dimensional set of elements.

All the nodal points on the generated elements must be incremented by the same amount. Multiple regions usually involves breaking the structure into simpler regions, generating a mesh in each region, and then reconnecting the regions to form the mesh for the whole structure.

All of the methods described have several common disadvantages. They all require the analyst to make a detailed plan of the mesh subdivision and of the node and element numbering scheme to be used. A transition from a coarse to fine mesh cannot be accomplished easily and may result in abrupt changes in element size or aspect ratio. If using a scheme that depends on nodal point number increments, it may be difficult to produce a satisfactory mesh with a constant increment. The major advantage of any of these methods is that they will reduce the amount of data required, over manual input of every node and element, thus reducing the time required to create a mesh.

To overcome the disadvantages associated with some of the more common mesh generation schemes, it is desirable to create a mesh generator that would be extremely flexible, easy to use, and does not require knowledge of the exact nodal point locations or numbering schemes. Flexibility, in this context, would include the ability to change the overall size of the mesh, the geometry of the structure, or the mesh size in a specific region without having to change a great deal of input data. An easy-to-use mesh generator should have a minimal number of logically arranged entries in the input file, easily understood error messages, and both printed and graphical output. The user should not need to know the number or location of every nodal point or element in the mesh. He should only be required to define the

geometry of the structure and give a description of the desired mesh size at various locations in the structure. The mesh generator should then be able to take this information and create the model. Then, with the information about loads, boundary conditions, and material properties, it should create the input file for a finite element code.

The mesh generation program to be developed in this project will follow the desirable characteristics mentioned above as guidelines. The approach used to create a mesh is as follows. In general, the input data will be limited to the definition of the geometry and the structural characteristics such as loads and boundary conditions. The program will then calculate the number of nodal points required along the boundary and locate these points. Interior nodal points will be generated using the distances between the points on the boundary to determine the locations of interior nodal points that will produce the most desirably shaped elements. Nodal points and elements are to be generated in layers going around the structure, proceeding toward the interior, until the entire structure has been divided into elements. Quadrilateral elements are to be preferred, except when transitions in mesh size necessitate the use of triangular elements. Once the mesh is created, a bandwidth reduction renumbering scheme will be employed to resequence the nodal points most efficiently. Finally, an output file will be generated which will be formatted to be used as the input file for a specified finite element code.

2. Literature Review

Numerous papers have been written on methods and techniques for generating finite element meshes. Some of these deal exclusively with node generation, others with element generation, and a few with entire mesh generation. Bush and Buell (1)* summarize thirty papers dealing with some of the more commonly used methods for generating nodes and/or elements. These methods all require that the user have already designed his mesh and know how many elements and nodes are required, as well as an idea of the approximate coordinates of all the nodes. A summary of seven of the papers reviewed by Bush and Buell (1) follows.

Among the node generation methods presented in Bush and Buell (1) is that of straight-line interpolation. This very simple technique requires that the user specify the coordinates and nodal point numbers for two ends of a string of nodal points. The number of nodal points to be generated between the two ends is calculated using a user specified nodal point increment. The intermediate points are located by dividing the length of the segment by the required number of nodal points. This technique produces a string of equally spaced, equally incremented nodal points. An extension of this method which is common in the larger finite element codes allows for two- or three-dimensional strings of nodes to be generated.

The idea of constant incrementation can also be extended to element generation. If an element is specified by listing the nodal points on that element and a generation increment, succeeding elements

* Numbers in parenthesis refer to references at the end of this thesis

can be specified by adding the generation increment to each of the nodal point numbers on the previous element, and incrementing the element number by one. Both this method and the method of nodal point generation by constant increment require that the user have worked out a numbering system that is compatible with the generation requirements.

Another method of nodal point generation that also has a counterpart in element generation is sides and parts (1). The structure is divided into a number of parts, each of which is bounded by either straight line segments or circular arcs. Within each part, the number of nodal points to be located on each side is specified. The major disadvantage of this scheme is that the number of nodal points on opposite sides of a part must be the same and that the part must have only four sides. This allows the interior nodal points to be generated by connecting corresponding boundary nodal points on opposite sides. The intersections of the connections specifies the interior nodal points. Once each part has been divided into a mesh, the parts are reconnected to form the structure and the nodes are renumbered to eliminate duplication. For the parts to be reconnected correctly, the common sides between parts must have the same number of nodes and have been specified identically.

The element generation form of sides and parts transforms the mesh into a rectangular grid located in an integer space with correspondence to the actual geometry. This I-J grid is oriented such that I is parallel to sides 1 and 3 of the part, and J is parallel to sides 2 and 4. To generate the elements, one may start at the minimum value of I and move in the direction of increasing J. The nodal points

on each element are those located at (I,J), (I,J+1),(I+1,J+1), and (I+1,J). Once the maximum value of J is reached, I is incremented by one and the process repeats. The combination of nodal point and element generation by the sides and parts method produces a mesh that has linearly interpolated interior nodal points and a systematic numbering scheme for both nodal points and elements. This method requires that the user locate all the boundary nodal points, align the opposite sides within parts, and align all the parts.

The node generation method of finite difference is similar to that of sides and parts in that the structure must be defined by a combination of straight line segments and arcs (1). The locations of the boundary nodal points must also be specified on these segments. All of this information is transformed to a grid in the I-J domain, as was done in sides and parts element generation. It is necessary that the number of nodal points on opposite sides of the grid match. A simplified version of Laplace's equation is used to locate the interior nodal points. A central finite difference approximation is used to derive the following equations:

$$x_{i+1,j} + x_{i-1,j} + x_{i,j+1} + x_{i,j-1} - 4x_{i,j} = 0$$

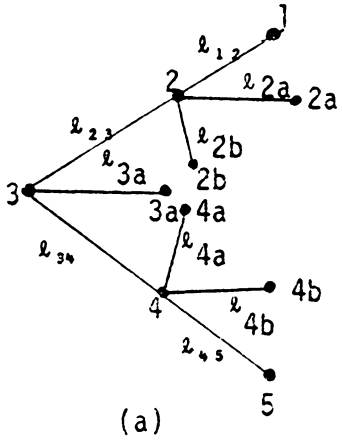
$$y_{i+1,j} + y_{i-1,j} + y_{i,j+1} + y_{i,j-1} - 4y_{i,j} = 0$$

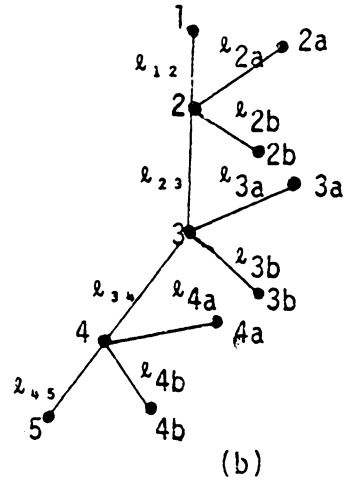
The values of $x_{i,j}$ and $y_{i,j}$ can be found by iterating on these equations. Once all the nodal points are located, the elements are generated using the same method described above. This approach produces a smoother mesh than linear interpolation, especially in cases where there is a large change in element sizes.

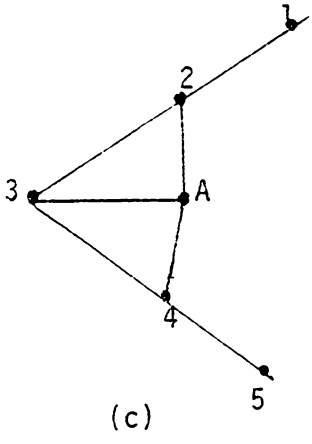
Though all of these methods reduce the amount of input required for the preparation of the finite element mesh, they still impose several restrictions upon the user. It is not possible for the user to generate a mesh with these techniques unless he has already developed the mesh. Any scheme that requires a very specific numbering scheme will force a tradeoff between convenience and bandwidth size, as the bandwidth is directly affected by the numbering scheme used. Often, the most convenient method of numbering the nodal points for generation will result in a larger bandwidth than if the nodal points had been specified individually. As the use of finite element analysis increased and models became more complex more flexible mesh generation techniques were developed.

Two of the more recent mesh generation programs developed are described in Sadek (2) and Schoof (3). Sadek (2) describes a program developed to generate triangular element meshes once the user specifies the location of all the boundary nodal points. His program does not require that there be the same number of nodal points on opposite sides of the structure or that the points be equally spaced along each side. His criteria for the generation of interior nodal points and specification of elements is that the element formed must be as close as possible to an equilateral triangle.

He begins by locating the first defined corner boundary point, that is, a point at which the interior angle is not equal to π. This node and the nodes on either side of it are used to generate the first interior node. Figure 2-1a shows an example where node 3 is the first corner node and nodes 2 and 4 are the adjacent nodes. The number of







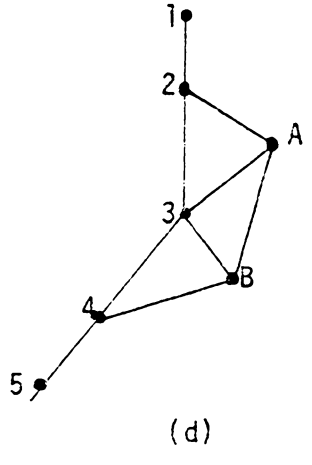


Figure 2-1
Node and Element Generation by Sadek's Method

elements that can be formed at any node is calculated by dividing the interior angle at the node by sixty degrees and rounding to the nearest integer. In Figure 2-1a, three elements may be formed at nodes 2 and 4 and two elements at node 3. The number of interior nodal points that can be generated from any node is one less than the number of elements that can be formed at that node. This leads to the generation of points 2a, 2b, 3a, 4a, and 4b from points 2, 3, and 4. The lengths ℓ_{2a}, ℓ_{2b}, ℓ_{3a}, ℓ_{4a}, and ℓ_{4b}, are proportional to the lengths of segments ℓ_{12}, ℓ_{23}, ℓ_{34}, and ℓ_{45}, by relationships derived from the properties of equilateral triangles. For example,

$$\ell_{3a} = \sqrt{\ell_{23}\,\ell_{34}}$$

$$\ell_{2b} = \sqrt[3]{\ell_{23}^{2}\,\ell_{12}}$$

Using Figure 2-1a as a reference, the interior nodal point A will be located using points 2b, 3a, and 4a. The coordinates x_A and y_A will be the weighted averages of coordinates x_{2b}, x_{3a}, x_{4a}, and y_{2b}, y_{3a}, y_{4a}. The weighting factors used are:

$$w_{2b} = (\ell_{2b} + \ell_{3a} + \ell_{4a})/(\ell_{2b})$$

$$w_{3a} = (\ell_{2b} + \ell_{3a} + \ell_{4a})/(\ell_{3a})$$

$$w_{4a} = (\ell_{2b} + \ell_{3a} + \ell_{4a})/(\ell_{4a})$$

The coordinates x_A and y_A will be

$$x_A = (w_{2b}x_{2b} + w_{3a}x_{3a} + w_{4a}x_{4a})/(w_{2b} + w_{3a} + w_{4a})$$

$$y_A = (w_{2b}y_{2b} + w_{3a}y_{3a} + w_{4a}y_{4a})/(w_{2b} + w_{3a} + w_{4a})$$

If the angle at node 3 is such that three elements (two interior nodal points) can be generated there, the situation of Figure 2-1b would result. In this case, two interior nodal points would be generated. One would be from the weighted average of points 2b and 3a, the other from the weighted average of points 3b and 4a. The weighting factors and the final coordinates would be calculated as above, except there would be only two points used instead of three.

The elements that result from the situations in Figure 2-1a and 2-1b are shown in Figure 2-1c and 2-1d respectively. Sadek's program would then find the next corner node and begin the entire process over again. If all corner nodes have been used, the newly generated nodal points in conjuction with the boundary nodes would be used to generate more nodal points. For example, in Figure 2-1c, nodes A, 4, and 5 could be used to generate another interior point. Nodal point generation continues around the current boundary of nodes until all of the nodes on the boundary have been incorporated into elements. The set of newly generated nodes forms a new boundary around the structure and the entire process begins over again on that boundary. This continues until the entire structure has been divided into elements and there are no free nodes from which to generate more nodes.

Sadek's method is convenient in that the only input required is a definition of the structure geometry and the location of the boundary nodal points. The user does not need to know how many nodes or elements will be present in the final structure, nor a specific numbering scheme for the nodes and elements. Also, there is no restriction on how the

nodal points may be arranged on the boundary. In his paper, Sadek presents the results of his program on two different structures. The results in both cases show that the program can handle large changes in element sizes well, and produces a satisfactory triangular mesh.

Another type of general purpose mesh generator is the program TRIQUAMESH, written by A. J. G. Schoofs (3). This program generates meshes composed of either triangles or quadrilaterals but not both. The required input describes the structure as a series of segments defined by key geometry boundary points. As part of the input, the user defines a roughness function at each specified boundary point and an overall reference element length. This function is the key to locating the boundary nodal points on the structure. The roughness function multiplied by the reference length determines the element side length in the region of the specified boundary point. By defining different values of the roughness function at the different boundary points, the boundary nodal point spacing will be nonuniform. This is the basis for developing a mesh that transitions from fine to coarse in different regions.

To actually locate the boundary nodal points, TRIQUAMESH uses the specified roughness function values at the ends of each segment to calculate an overall piecewise harmonic roughness function for the segment. Using curvilinear coordinates, the length of element side i is:

$$\ell = s_i - s_{i-1}$$

where s_i is the curvilinear coordinate of point i. By Schoofs'

definition of the roughness function, this same length must also be :

$$\ell = \frac{1}{2}\left[g2_{i-1} + g2_i\right]$$

where g2 is the roughness function along the segment. Combining equations of these types, one can derive an equation in terms of the curvilinear coordinates and roughness functions of points i, i+1, and i-1. This equation is:

$$\frac{s_i - s_{i-1}}{g2_i + g2_{i-1}} = \frac{s_{i+1} - s_i}{g2_{i+1} + g2_i} \qquad i=1,2,\ldots n-1$$

where n is the number of nodal points located on the segment. Knowing the maximum value of s for the segment and iterating on this equation, one can find the curvilinear coordinates of all the nodal points on the segment. The spacing between these points will be dependent upon the user specified roughness value at the endpoints of the segment. This procedure is repeated for the rest of the segments until all the boundary nodal points have been located.

After all the boundary nodal points have been located, TRIQUAMESH cuts the structure into convex regions, regions in which no interior angle is greater than π. The cutting lines are from boundary nodal point to boundary nodal point and are chosen using the following criteria:

1. The angles between the cutting line and the boundary must be as close as possible to sixty degrees (triangular elements) or ninety degrees (quadrilateral elements)

2. The length of the cutting line should be as short as possible.

3. The number of nodal points on the cutting line should be as low as possible. Nodal points are located on the cutting lines using the roughness function values for the endpoints of the cutting line. The procedure is the same as for locating the boundary nodal points.

After cutting the structure, regions result in which no angle is greater than π. Nodal points are generated along the cutting lines using the roughness functions at the boundary nodal points. If the number of nodal points around a convex region is three or four, a triangular or quadrilateral element is generated. If the number of nodal points around the area is greater than three or four, additional cutting lines are introduced into the region using the above criteria. Elements are generated between each pair of cutting lines. This continues until the area is completely divided up or until the region becomes concave again. If all possible elements have been generated in the region, the program repeats the procedure in another region, continuing until the whole structure is divided up. If the region becomes concave again, it is split into two convex regions with a cutting line, and the two regions are divided into elements.

The result of these procedures is a mesh composed of either triangular or quadrilateral elements that will transition from a fine to coarse mesh as specified by the user. The required input is the definition of the structure and an indication of the relative size of the mesh at each specified point. The mesh that results can be refined overall by changing the specified reference length for the whole structure. It can also be refined in a specific region by changing the

roughness function values in that region.

In a general purpose mesh generator, post-processing is often needed to prepare the output of the mesh generator for efficient use in a finite element code. Available post-processing in TRIQUAMESH include shape improvement and bandwidth reduction. The output file can be formatted to be used with three different finite element programs. Schoofs includes several examples of the result of TRIQUAMESH in his paper. The meshes shown demonstrate the versatility of the program and the ease with which a mesh can be refined locally or overall. The major drawback of the program is that transitions are not always well handled due to the exclusive use of triangular or quadrilateral elements. Transitioning a quadrilateral mesh without the use of triangular elements can result in strangely shaped elements. However, this is a relatively minor drawback when the overall savings in mesh generation time is considered.

A general purpose mesh generator for triangular elements, TRIFEM II, is described by Kleinstreuer (4). Like Sadek, (2) this program requires that the user specify the location of all of the boundary nodal points. In the first stage of generation, the region is intially divided into elements by connecting nodal points on opposite boundaries to form triangles. The result of this step is a mesh composed of elements with all three nodes on boundaries. This mesh is then refined by locating new nodal points at the centroids of the triangular elements and forming three new triangular elements. This procedure continues until all elements are as close as possible to equilateral triangles.

As shown by the examples in Kleinstreuer (4), TRIFEM II can

generate an appropriate triangular mesh for a given geometry. The need for the user to specify the boundary nodal points and the exclusive use of triangular elements limits the versatility of the program. Also, because nodal points on opposite boundaries are connected in the first stage of element generation, strangely shaped elements may result if there is a great disparity in the number of nodal points on the opposite boundaries.

Another program that generates triangular elements is discussed by Bykat (5). This routine also requires that the user define the location of all boundary nodal points. However, it has the ability to redefine or to add boundary nodal points at reentrant corners to develop a finer mesh. The three major parts of this program are triangulation of convex regions, division of a structure into convex regions, and the grading of the mesh at reentrant corners.

Convex regions, regions in which no angle is greater than π, are established by joining boundary nodal points. The criteria for cutting the structure into convex regions is that no extremely small angles may be created and the number of regions created must be as few as possible. Nodal points are located along the cutting lines using the nodal point spacing at the endpoints of the cutting lines as a guide. The ratio between the lengths of any two consecutive segments on the boundary must be less than two.

Once a convex region is established, all corners with interior angles of less than eighty degrees are cut off from the structure. Then, a node is chosen to use as a beginning point for nodal generation. The points on either side of this node are also used in the generation

process. This is illustrated in Figure 2-2 with point 2 as the starting point and points 1 and 3 as the adjacent points. An interior point R is located such that the following relations are satisfied

$$\ell_{1R} = \ell_{3R} = \frac{(\ell_{12} + \ell_{23})}{2} \leq \ell_{max}$$

where ℓ_{max} is the maximum length of an element side on the boundary. Elements are defined by nodes 1, 2, and R and by nodes 2, 3, and R. The procedure then repeats with point 3 as the central point and points R and 4 as the adjacent points. This is repeated until all of the nodes on the current boundary have been incorporated into elements. A new boundary is established by the generated elements and the procedure is repeated.

If a reentrant corner is found in the structure, the node at that corner is marked for grading. Additionally, the user can specify nodes at which he wants grading to be done. During the process of grading, the program will introduce nodes into the boundary or will respace existing nodes on the boundary. This grading method produces a fine mesh in the area around the indicated nodes but does not greatly affect the mesh outside of the immediate region of the node.

Like the programs of Sadek (2) and Kleinstreuer (4), this mesh generator only makes triangular elements and requires that the user specify the location of the boundary nodal points. The grading option in this program is helpful and can allow the user to develop a graded mesh without having to set up the boundary nodal points with varied spacing.

Another method of developing a finite element mesh is described

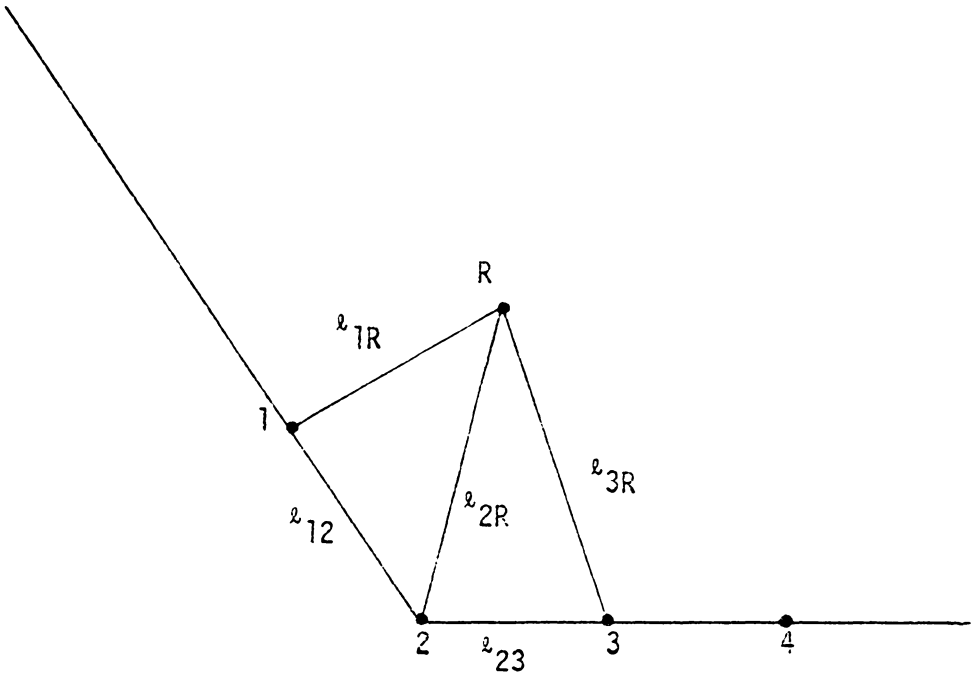


Figure 2-2
Node and Element Generation by Bykat's Method

by Brown (6) and Harber (7). These papers describe the use of mappings to locate internal nodes and to define elements. Brown (6) uses Schwarz-Christoffel transformations and Harber (7) uses discrete transfinite mappings. Both of these methods involve a great deal of complex mathematical theory but basically, they map the actual structure into a specified regular geometry. The nodes and elements are located and then transformed back to the actual structure. The meshes that result are composed of only one type of element, usually triangular, and are well transitioned from a fine to coarse mesh.

If any mesh generator does not automatically number the nodes most efficiently, a bandwidth reduction routine must be employed before the mesh is submitted to a finite element code. There are several popular schemes for doing this, including the methods developed by Collins (8) and by Cuthill-McKee (9). The bandwidth of a structure is proportional to the maximum difference between the node numbers on an element. Collins' method (8) attempts to renumber the structure by moving the origin of the numbering system to another node in the structure. The routine begins by renumbering all the nodes connected to the node at the origin. Then, in turn, it renumbers all of the nodes connected to the newly numbered nodes. As a node is renumbered, the bandwidth at that point is calculated. If the bandwidth exceeds the minimum bandwidth from all of the previous renumberings, the process stops and the origin of the numbering system is moved to a new node and the process repeated. If all of the nodes in the structure are renumbered with a smaller bandwidth than the current minimum value, that bandwidth is taken as the current minimum and the procedure begun over

again with a new nodal point as the origin. Once all nodes have been used as the origin of the numbering system, the numbering method that gave the minimum bandwidth is used.

The Cuthill-McKee method (9) selects a node as a starting point, gives it a number of one, and then renumbers the nodes connected to the starting node beginning at two. Each of the nodes connected to the starting point is then used as a starting point in turn, and the nodes connected to it renumbered sequentially. This continues until all nodes in the structure have been renumbered.

The primary difference between Collins' method and the Cuthill-McKee method is the number of different numbering schemes tried. Because Collins' method tries as many different numbering schemes as there are nodes, it is more likely to arrive at the numbering pattern that produces the minimum bandwidth for the structure.

3. Problem Approach

The mesh generator described here is a general purpose program designed to create meshes using a combination of quadrilateral and triangular elements. Quadrilateral elements predominate, with triangular elements incorporated where needed to provide a transition between a fine and coarse mesh. The program is structured such that the mesh can be easily changed or redefined with minimum effort from the user, thus making it simple to use and very versatile. There are four major stages of operation in this program, 1) reading and interpreting input data, 2) generation of boundary nodal points, 3) generation of interior nodal points and definition of elements, and 4) bandwidth reduction and output file creation.

The input data needed for the first stage of the program is described in Table 3-1. The input data file is free formatted with blank lines used as separators between sections of data. Comments are allowed in the input file if they are preceeded by a dollar sign ($) in column one. Entries A through C are used to generate nodal points and elements while entries D through H are used to define the loading and constraints on the model.

The geometry of the structure and the desired mesh size are defined in the first three entries of Table 3-1. Entry A, the reference length, is used to describe the overall size of the mesh. Doubling the value of the reference length will double the size of all the elements, while halving the value of the reference length will halve the size of all the elements.

Table 3.1

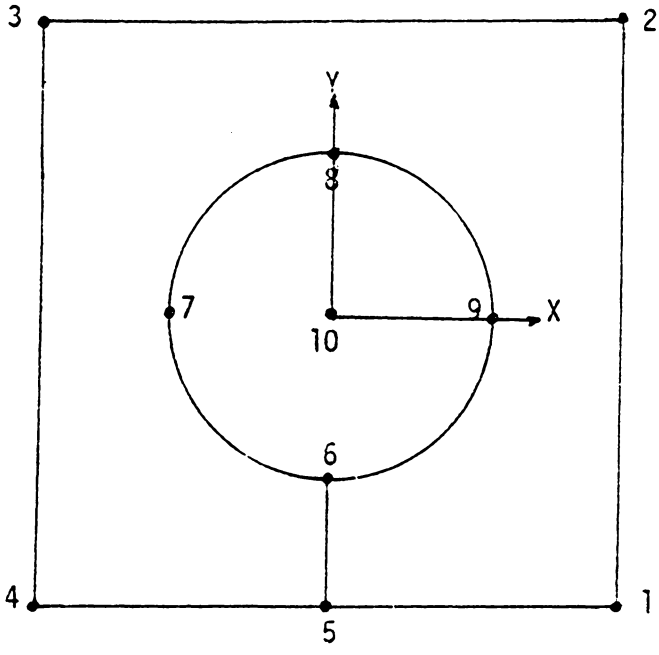
Required Input for Mesh Generation

	Variable Name	Description
A.	Reference Length	Controls the overall size of the mesh
B.	Keypoint Data	Describes key geometry points
	1. Keypoint Number	Number of the keypoint being entered
	2. Coordinates	Global X and Y coordinates
	3. Grading Value	Relative element size around point
C.	Segment Data	Describes the boundary of the model
	1. Segment Number	Number of the segment being entered
	2. Beginning	Keypoint used as the beginning
	3. End	Keypoint used as the end
	4. Center	Keypoint used as the center of an arc
D.	Load Data	Describes the loads on the structure
	1. Pressure Load	
	a. Segment	Segment on which the load is applied
	b. Magnitude	Positive value is compressive
	2. Concentrated Load	
	a. Point	Point where load applied
	b. Direction	Direction of load (X or Y)
	c. Magnitude	Value of load
E.	Segment Boundary Conditions	Describes the segment boundary conditions
	1. Segment	Segment to be fixed
	2. Condition	Specify direction to be fixed
F.	Keypoint Boundary Conditions	Describes the keypoint boundary conditions.
	1. Keypoint	Keypoint to be fixed
	2. Condition	Specify direction to be fixed
G.	Element Type	=3 is Axisymmetric element =4 is 2D Isoparametric Element
H.	Material Data	Material properties for structure
	1. Youngs Modulus	Value of Young's modulus
	2. Poisson's Ratio	Value of Poisson's ratio

Keypoint locations are described in entry B of Table 3-1. These keypoints are points on the structure where a nodal point must be located or are points used as centers of arcs on the structure. An example of which points would be defined as keypoints for a sample structure is shown in Figure 3-1a. Especially important in this figure is the way in which keypoints are defined around the interior hole. Arcs must be broken into ninety degree or less segments, relative to a local coordinate system through the center of the arc. The endpoints of an arc must lie within the same quadrant of the X-Y coordinate system. The example shown as incorrect in Figure 3-1b has the endpoints of the segments in different quadrants. The correct example shown in Figure 3-1b has the endpoints of the segments in the same quadrants. Aside from the geometrical considerations, keypoints must also be located at any point where a concentrated load is to be applied.

The grading value that must be entered along with the coordinates of the keypoints is used to determine the element size in the vicinity of the keypoint. All elements in the region of a specific keypoint will have side lengths as close as possible to the product of the grading value at that keypoint multiplied by the reference length for the whole structure. Whereas, the reference length controls the element size for the whole structure, the grading value controls the element size at a particular location in the structure. Doubling the grading value at a specific keypoint will double the size of the elements in the vicinity of that point.

The boundary of the structure is defined in entry C of Table 3-1. The segments must be specified such that they form a closed boundary



Segment	Points
1	1-2
2	2-3
3	3-4
4	4-5
5	5-6
6	6-7-10
7	7-8-10
8	8-9-10
9	9-6-10
10	6-5
11	5-1

(a)

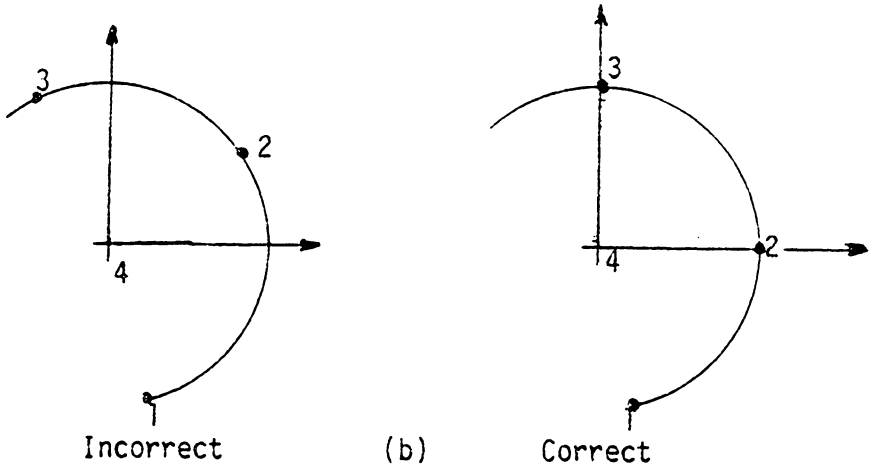


Figure 3-1
Examples of Segment Specification

about the model. The keypoints are used to define the beginning, end, and center of the segments. Although the segments may be entered in any order desired, the first segment entered must be defined in a counter-clockwise direction around the structure. In other words, as one moves from the beginning to the end of the first segment, the structure must be on the left. If there are any interior features, such as the hole in the structure of Figure 3-1a, the segment connecting the hole to the exterior boundary must be specified twice. The first specification will be going from the exterior boundary to the interior feature and the second will be going from the interior feature to the exterior boundary. This is illustrated in the list of the segment definitions adjacent to Figure 3-1a.

Entries D through H of Table 3-1 are used to describe the loading, boundary conditions, and desired material type for the structure. Loads may be applied in either of two ways as indicated in Table 3-1. If a pressure load is specified on a segment, the load is interpreted as acting perpendicular to the segment. A positive magnitude indicates a compressive load, a negative value a tensile load. Concentrated loads must be applied at the previously defined keypoints. They can be specified as acting in either the global X or Y direction. The sign (positive or negative) of the magnitude of the load indicates the direction of the load with respect to the global X or Y coordinate system.

Boundary conditions are used to describe the direction in which the nodal points located on the boundary are free to move. Either segments or keypoints may be specified to be fixed in either the X, the

Y, or both the X and the Y directions. To fix either a segment or a keypoint in both the X and Y directions, it is only necessary to enter the desired point or segment. To fix it in either the X or the Y directions, one must enter the desired point or segment and the direction in which one wishes to fix it.

The element type entry is used to indicate which element type is to be specified in the input file created for the finite element code. Currently, this program supports the MESS finite element code, a program in use at Virginia Polytechnic Institute and State University. The available two-dimensional elements allowed by this code are a four-noded axisymmetric and a four-noded plane-stress element. These elements are specified by entering a three or a four, respectively, at entry G of Table 3-1. Likewise, the MESS code requires the values of Young's modulus and Poisson's ratio for the material used in the structure. Entry H of Table 3-1 describes the material property entry.

After all of the input data has been read by the program, it begins to generate the boundary nodal points. These nodal points are generated along each segment using the specified grading values at the endpoints of the segment as a guide for the nodal point spacing. After nodal points have been generated along each segment, the segments are connected together to define a continuous counter-clockwise boundary around the structure. The boundary nodal points are numbered sequentially along the boundary.

Interior nodal point generation is done in layers around the structure. The location of the interior nodal points is based upon the spacing of nodal points on the boundary. The object is to make elements

which are as close as possible to either squares or equilateral triangles. For each interior nodal point generated, a number of trial points are located based upon the spacing of three or four previously defined nodal points in the same area of the structure. The locations of these trial points are averaged together to define the final location of the interior nodal point. After the final location is calculated, the newly generated nodal point is incorporated into an element which will include the previously defined nodal points in the vicinity of the new nodal point. The elements formed will be quadrilaterals unless a triangular element is needed for efficient transition of the mesh. As mentioned previously, these interior nodal points are generated in levels or layers around the structure. Each level of nodal points will define a new boundary around the portion of the structure which has not yet been divided into elements. Nodal point generation and element definition continues around the structure proceeding towards the center until the entire structure has been divided into elements.

Once the whole structure has been divided into elements, a bandwidth reduction routine is used to renumber the generated nodal points more efficiently. After the renumbering is complete, an output file is created that can be used as the input file to the program MESS. This file includes the listing of all of the nodal points, their coordinates and their specified boundary conditions, the listing of the elements, and the listing of the loads and material property data. In addition to this file which is formatted to be used with MESS, another output file is generated which tells the user how the program interprets his input data and gives messages about any problems which the program

encountered in trying to generate a mesh from the specified input.

This mesh generation program is currently limited by the following factors:

1. Maximum of 500 nodes, 300 elements
2. Maximum of 5 interior features such as holes
3. Boundary must be definable by straight lines and circular arcs.
4. Maximum of 100 nodes on any level, including the user-defined boundary.

These limitations do not cause any problem in the intended application of this code. This program was designed to be a pre-processor for the program MESS, whose post-processing routines can only accomodate 250 elements. MESS is used as an instructional finite element code for beginning users. The restrictions on the number of interior features, the type of boundary segments, and the number of nodes on any level do not cause any problems as the models developed for use with MESS are fairly small and geometrically simple.

4. Program Formulation

A. Input Data Interpretation

The required input for this mesh generator is listed in Table 3-1 (pg. 21). The program is set up to accept an input file that is assigned to logical unit 5. A copy of the input file must also be available and assigned to logical unit 7. The checking of all the input data is done using the copy of the input file (AUXIN) while the actual reading of data and assigning to variables is done with the file assigned to logical unit 5 (IN).

A line of data is read in from AUXIN as an 80 character long string. If the first character is a dollar sign, the card is interpreted as a comment card and the next line of data is read. Every character in the string is checked to make sure it is one of the allowable entries. These allowable entries include the numerals from 0 to 9, the letters X, Y, and E, the delimiters such as periods, commas, apostrophes, and blanks. If a string is found that contains a character that is not one of the allowable characters, both a message and the string are printed to the output file. If the string contains only valid entries, it is examined to see how many separate data entries are in the string. This is done with two logical flags that are reset when a blank or comma followed by a character is found.

If the string contains only blanks, it is a separator between sections of input and the program ignores it. However, the program does keep track of how many blank cards it has encountered so that it knows which section of input is currently being examined. A string of data

that contains non-blank characters is evaluated as to how many entries are on the line and to which section of the input file it belongs. Program control is sent to the appropriate section of the reading subroutine where the corresponding line is read from the IN file and assigned to the correct variables. If the combination of number of entries and input section does not match any of the allowable combinations, the string of data is printed out to the output file with an error message explaining the problem.

This method of checking, interpreting, and reading input data will catch any obviously incorrect entries before program execution has gone very far. The computer's interpretation of incorrect data is returned to the user along with an explanation of why the data is being rejected. If all of the data is interpreted as being correct and is read to the appropriate variables, the computer's interpretation of the input data is still printed to the output file as a reference for the user.

Once all of the data has been read in, several checks are performed. The first of these checks to see if any segments have been specified as arcs. Those segments that were defined by three points are listed in the output file with a message that they were interpreted as being arcs. Another check is made to see if the segments, as entered by the user, define a closed boundary around the structure in a counter-clockwise direction. One of the requirements for input is that the first segment, and the first segment only, must be entered such that as one moves from beginning to end, the structure lies to the left of the segment. The definition of the other segments are checked to see if

there is a tip-to-tail connection of segments around the structure in the correct direction. If this is not the case, the definition of the segments is reordered to describe a continuous counter-clockwise boundary around the structure.

As the segments are being checked, the program is also watching for any segments that are defined twice. This condition would indicate the presence of an internal feature such as a hole. Segments which are entered twice are stored in an array to be used later in boundary nodal point generation.

After the segments have been properly ordered and identified, the boundary keypoints are checked. If there are any arcs in the structure, the boundary keypoints are renumbered such that the arc centers are last in the list. This sets up a consecutive listing of the boundary keypoints that actually lie on the structure followed by the points that do not lie on the structure but are centers of arcs. The endpoints and the centers of all the segments are then renumbered to reflect the reordering of the boundary keypoints.

Once all of the data has been checked, interpreted, and assigned to the proper variables, the length of each of the segments is calculated. For segments defined as arcs, the radius and subtended angle are calculated and used to calculate the arc length. This information is needed in the next stage of the program, the generation of boundary nodal points.

B. Boundary Nodal Point Generation

This program generates boundary nodal points using the reference length, boundary keypoint data, and segment definitions. The grading value at a boundary keypoint is a measure of the relative size of the mesh in the region of that point. The reference length affects the overall size of the mesh. At a specific boundary keypoint, the element side lengths around the point will be as close as possible to the reference length multiplied by the grading value at the boundary keypoint.

Between endpoints of a segment, the grading value is assumed to be a linear function of the type

$$g(s) = ms + g_1 \qquad [4\text{-}1]$$

where s is the curvilinear coordinate system along the segment, m is the slope of the grading function along the segment, and g_1 is the grading value at the beginning of the segment where s_1 is zero. The slope, m, of the grading function is

$$m = \frac{g_n - g_1}{s'_n} \qquad [4\text{-}2]$$

where g_n is the grading value at the end of the segment and s'_n is the curvilinear coordinate of the endpoint of the segment.

As a result of the linear nature of the grading value function, the relationship between g(s) at different points on the segment may be written

$$\frac{g(s)_i}{s_{i+1} - s_i} = \frac{g(s)_{i+1}}{s_{i+2} - s_{i+1}} \qquad [4\text{-}3]$$

Substituting Eq.[4-1] into Eq. [4-3] and simplifying yields

$$s_{i+2} = \frac{m\, s_{i+1} - g_1 s_i + 2 g_1 s_{i+1}}{g_1 + m\, s_i} \qquad i=1,2,\ldots n-2 \qquad [4\text{-}4]$$

where n is the total number of nodal points on the segment.

When i is equal to one, the value of s_1 is zero and the value of s_2 can be approximated as g_1 multiplied by the reference length. The value of m can be calculated from Eq. [4-2], thereby allowing one to solve for s_3. This process can be repeated by using s_2 and s_3 to solve for s_4, s_3 and s_4 to solve for s_5, etc.

Curvilinear coordinates of the points on the segments are calculated until two values of s are found which bracket the established value of s'_n, the length of the segment. The number of nodal points on the segment is equal to the subscript of the calculated value of s which is closest to s'_n. A correction factor, c, is calculated from

$$c = \frac{s'_n - s_n}{n-1} \qquad [4\text{-}5]$$

If there are n nodal points on the segment, there are n-1 element sides on the segment. Therefore, the dimensions of c are units of length per number of element sides. The value of c is added to the value of s_2 used in Eq.[4-4]to calculate s_3. New values of s_3 through s_n are calculated using Eq. [4-4].

Iteration on the values of s_3 through s_n continues until one of three possible conditions occurs. First, if the value of c is less than 0.1% of the segment length, iteration stops and the value of s_n is set equal to the value of s'_n, the actual coordinate of the segment endpoint. To maintain the structural dimensions specified by the user, the first and last nodal points on a segment must have the same location as the endpoints of the segment. The nodal points located on the segment are numbered with two indices, one indicating the segment number and the other the sequential number of the node between one and n. Then, the nodal points are located on another of the segments defining the boundary. This continues until nodal points have been located on all of the segments.

Another possible condition is that c starts to increase. If this happens, the current values of s_3 through s_n are dropped. The current value of s_2 is used in Eq. [4-4] and new values of s_3 through s_n are calculated. This will also result in a new value of n. Iteration begins again using the new values of n and s_3 through s_n.

The last possiblity is that after twenty iterations c has neither increased nor converged to the specified limit. If this is the case, execution ends and an error message is printed in the output file. An example of a condition in which this situation might occur is if the user specified grading values at the endpoints of 2 and 0.5, with a reference length of one, and a segment length of 1.5 inches. Since the reference length multiplied by the grading value at the beginning point is greater than the length of the segment, the program can not locate nodes under this condition and will stop trying to meet the convergence

conditions after twenty iterations.

Once the nodal points have been located on all of the segments, the program calculates the loads to be applied at the generated nodes. Also, during this stage of the program, the user-specified boundary conditions are applied to the nodal points.

Two types of user-specified loads are allowed, pressure loads on a segment and concentrated loads on the user-defined boundary keypoints. If a pressure load, p_j , has been specified along segment j , the load at a nodal point i on segment j is calculated from

$$f(j,i)= \frac{p_j}{2} [s(j,i) - s(j,i-1)] + \frac{p_j}{2} [s(j,i+1) - s(j,i)]$$

Concentrated loads are specified in accordance with the current numbering of the user-defined boundary keypoints. For example, a concentrated load to be applied at boundary keypoint 5, which defines the beginning of segment 3, would be specified as being applied to node (3,1). The 3 refers to the segment number and the 1 refers to the node's position on the segment.

There are also two types of boundary conditions allowed. Both segments and user-defined boundary keypoints can be fixed in either the X, the Y, or both directions. The boundary conditions that have been specified are related to the nodal points located on the segments, using a zero to represent a free degree of freedom and a one to represent a fixed degree of freedom.

The final step in defining the boundary of the structure is to calculate the global X and Y coordinates of the boundary nodal points

and to renumber the points consecutively around the boundary beginning with one. The points are still indicated with two indices indicating the level number and the location on that level. The boundary is given a level number of one. During this stage of the program operation, any duplicate nodes that have been generated on a segment connecting an interior features to the boundary are redefined. The duplicate nodes are redefined so that they are in the same location on the segment. The X and Y components of the pressure loads are calculated and added to the concentrated loads at the boundary nodes. The load data and boundary condition data are referenced to the new numbering system and all of the arrays set up to begin internal nodal point generation.

C. Interior Nodal Point Generation and Element Definition

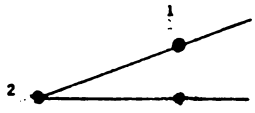
Once all of the boundary points have been located, the program begins preparing to generate the interior nodal points. Interior nodal points are generated in layers or levels around the structure. Each layer of nodal points defines a new boundary around the portion of the structure that has not been divided into elements. The information required to begin generating interior nodes is the location of all of the nodal points on the current level. The array containing the lengths of all the segments between nodal points on the current level must also be available.

The coordinates and segment lengths are used to calculate the interior angle at each free node on the current level. If the interior angle at node j is to be calculated, vectors are defined from point j to point i and from point j to point k, where points i and k are the nodal points on each side of point j. The dot product of these two vectors will be the cosine of the angle between the two vectors. The angle enclosed by these two vectors will be either the inverse cosine of this result or 2π minus the inverse cosine of this result. For the case of three consecutive nodes i, j, and k, the equation for the interior angle at node j would be

$$\theta(j) = \cos^{-1}\left[\frac{(x_i - x_j)(x_k - x_j) + (y_i - y_j)(y_k - y_j)}{(\ell_{ij})(\ell_{jk})}\right]$$

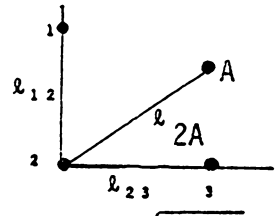
The calculated angle θ is then checked to find out if it is the actual interior angle or if the interior angle is 2π minus θ.

Once the interior angle at all of the nodes has been calculated, the program decides how many trial nodal points can be generated in the interior relative to each nodal point on the current level. Even though the program is set up to generate predominantly quadrilateral elements, the trial nodal point locations are found using relationships derived from equilateral triangles. To begin, the program calculates how many sixty degree pieces can be cut out of the interior angle at each node on the current level. If this value is less than 0.5, it is rounded to 1. If the value is between 0.5 and 2.5, it is rounded to 2. By rounding all values between 0.5 and 2.5 to 2, at least one trial point will be generated for all cases except those where the angle involved is less than 30 degrees. An angle less than 30 degrees is too small to be used in a quadrilateral element so no trial points are generated and a triangular element is formed. Values between 2.5 and 3.5 are rounded to 3, and so forth. This value is given the name NUME. The number of trial points that can be generated at each nodal point on the current level is one less than the value of NUME for the nodal point. The equations used to locate the trial points are derived from relationships equating the ratio of the sides of triangles which incorporate the trial points. These equations and the situations to which they apply are illustrated in Figure 4-1a,b,c,d,e, and f. For example, if one has a node 2 with NUME(2) equal to two, one trial point, A, can be generated, as shown in Figure 4-1b. If node 2 lies between nodes 1 and 3, the triangles that can be formed are defined by 1-2-A and A-2-3. If these triangles were equilateral triangles, the following relation would hold



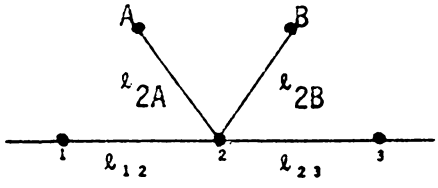
No trial points generated

(a)



$$\ell_{2A} = \sqrt{\ell_{12}\ell_{23}}$$

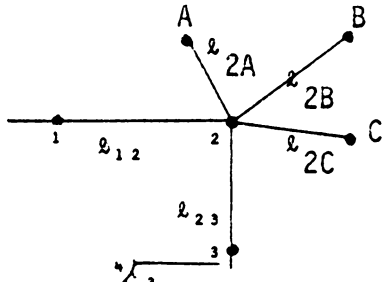
(b)



$$\ell_{2A} = \sqrt[3]{\ell_{12}^2\ell_{23}}$$

$$\ell_{2B} = \sqrt[3]{\ell_{12}\ell_{23}^2}$$

(c)

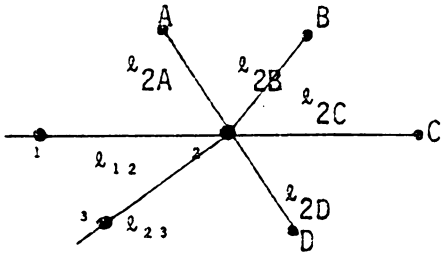


$$\ell_{2A} = \sqrt[4]{\ell_{12}^3\ell_{23}}$$

$$\ell_{2B} = \sqrt{\ell_{12}\ell_{23}}$$

$$\ell_{2C} = \sqrt[4]{\ell_{12}\ell_{23}^3}$$

(d)



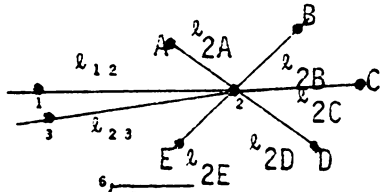
$$\ell_{2A} = \sqrt[5]{\ell_{12}^4\ell_{23}}$$

$$\ell_{2B} = \sqrt[5]{\ell_{12}^3\ell_{23}^2}$$

$$\ell_{2C} = \sqrt[5]{\ell_{12}^2\ell_{23}^3}$$

$$\ell_{2D} = \sqrt[5]{\ell_{12}\ell_{23}^4}$$

(e)



$$\ell_{2A} = \sqrt[6]{\ell_{12}^5\ell_{23}}$$

$$\ell_{2B} = \sqrt[6]{\ell_{12}^4\ell_{23}^2}$$

$$\ell_{2C} = \sqrt{\ell_{12}\ell_{23}}$$

$$\ell_{2D} = \sqrt[6]{\ell_{12}^2\ell_{23}^4}$$

$$\ell_{2E} = \sqrt[6]{\ell_{12}\ell_{23}^5}$$

(f)

Figure 4-1
Relationships to Locate Trial Points

$$\frac{\ell_{12}}{\ell_{2A}} = \frac{\ell_{2A}}{\ell_{23}}$$

This can be simplified to

$$\ell_{2A} = \sqrt{\ell_{12}\,\ell_{23}}$$

The other relationships listed in Figure 4-1 are derived similarly.

Interior nodal points are located using three adjacent nodal points to calculate a number of trial points. The number of trial points that will actually be used to locate an interior point is dependent upon the values of NUME at each of the points used to generate the trial points. For example, if the three nodes used are designated by node(1), node(2), and node(3), and the number of elements that can be generated at each of these nodes is three, two, and three respectively, the situation shown in Figure 4-2a results. In this case, the points labeled B, C, and D would be averaged together to locate an interior nodal point. If the situation were as illustrated in Figure 4-2b where the value of NUME at all three nodes is three, two interior points would be generated. One would be the average of the points labeled B and C and the other would be the average of the points labeled D and E. A third possible case is shown in Figure 4-2c. Here, the value of NUME at node(1) and at node(3) is three and at node(2) is four. This would result in three interior nodal points being generated. One would be the average of the trial points B and C, another would be the trial point D, and the third would be the average of points E and F.

The averaging process used to locate the interior nodal points

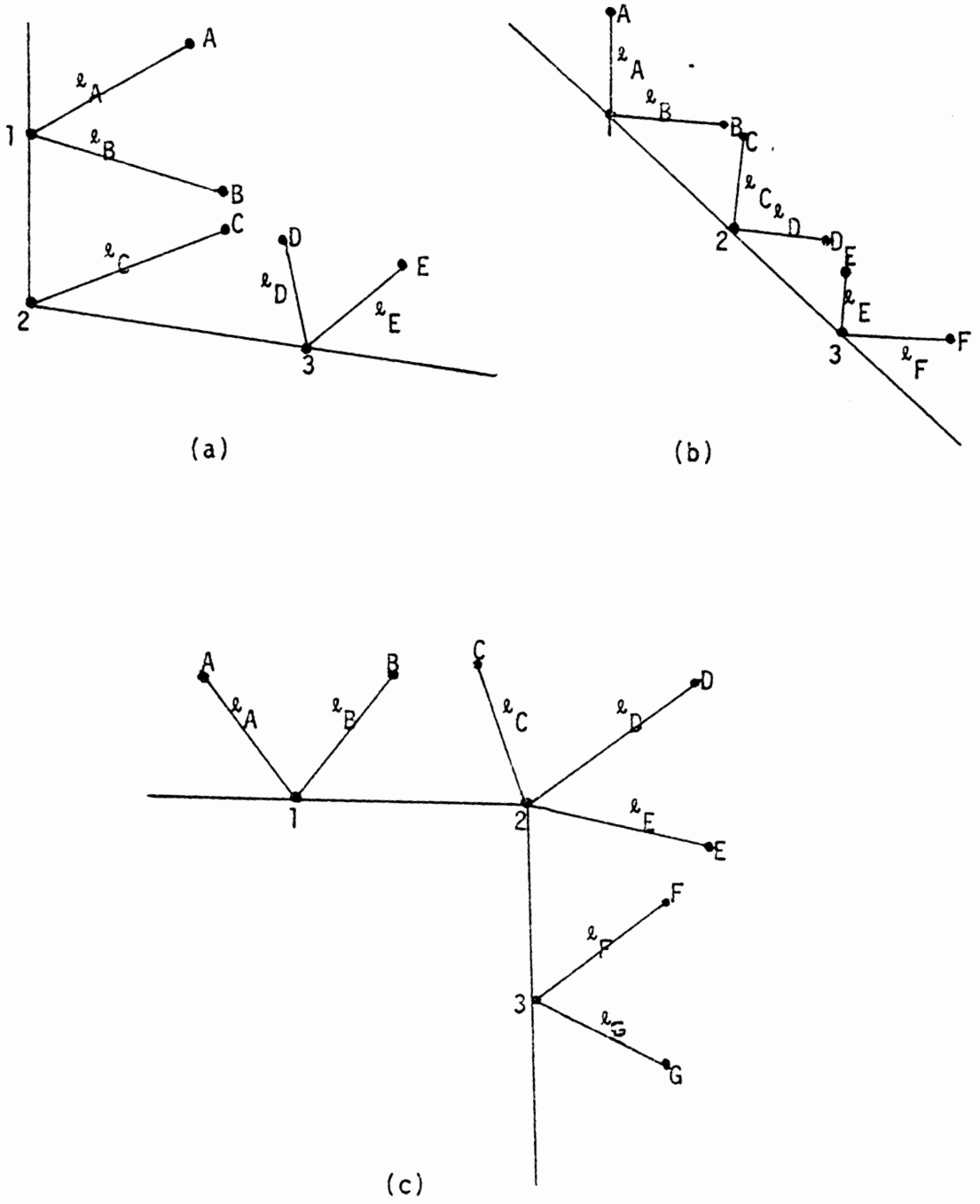


Figure 4-2
Examples of Trial Point Locations

involves the use of weighting factors. These weighting factors are based upon the lengths to the trial points averaged together to locate an interior nodal point. For instance, the weighting factors used for the situation shown in Figure 4-2a would be

$$w_B = (\ell_B + \ell_C + \ell_D)/\ell_B$$
$$w_C = (\ell_B + \ell_C + \ell_D)/\ell_C$$
$$w_D = (\ell_B + \ell_C + \ell_D)/\ell_D$$

The coordinates of the nodal point located using points B, C, and D would be

$$x = (w_B x_B + w_C x_C + w_D x_D)/(w_B + w_C + w_D)$$
$$y = (w_B y_B + w_C y_C + w_D y_D)/(w_B + w_C + w_D)$$

The coordinates of the interior nodal points generated in the situations illustrated in Figures 4-2b and 4-2c would be calculated in the same manner.

As mentioned previously, nodal point generation proceeds in layers around the structure until all of the structure has been divided into elements. The first nodal point generated on a new level will be found using the first defined corner node on the previous level and the two nodes adjacent to it. The next nodal point will be generated using the previously generated node, node(3) from the first case, and the node adjacent to the former node(3). For example, node 1 on level 2 (denoted node(2,1)) in Figure 4-3a would be located using nodes 1, 2, and 3 on level 1 (nodes (1,1), (1,2), and (1,3)). Node (2,2) would then be generated using nodes (2,1), (1,3), and (1,4). After node (2,2) is

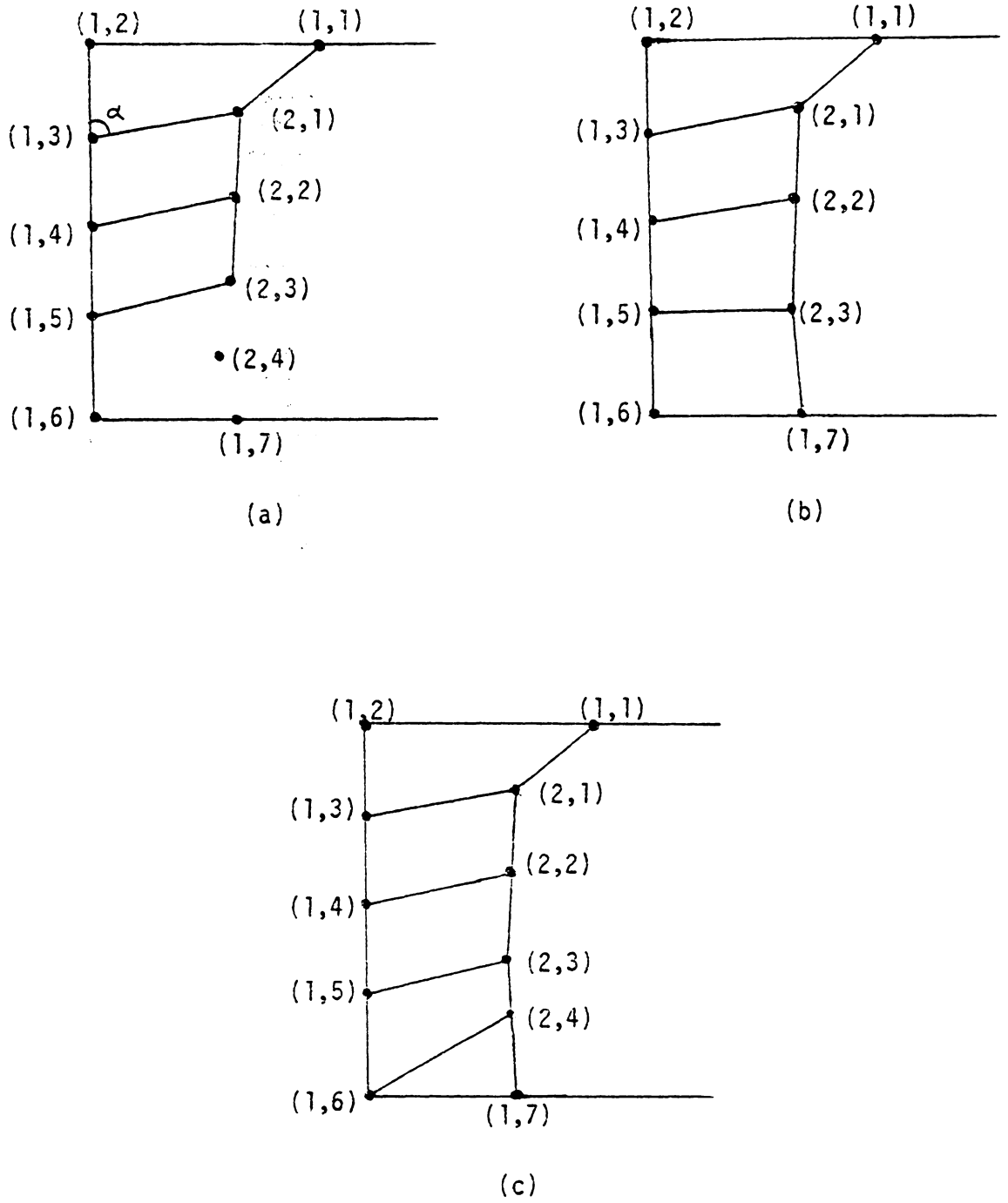


Figure 4-3
Examples of Element Generation

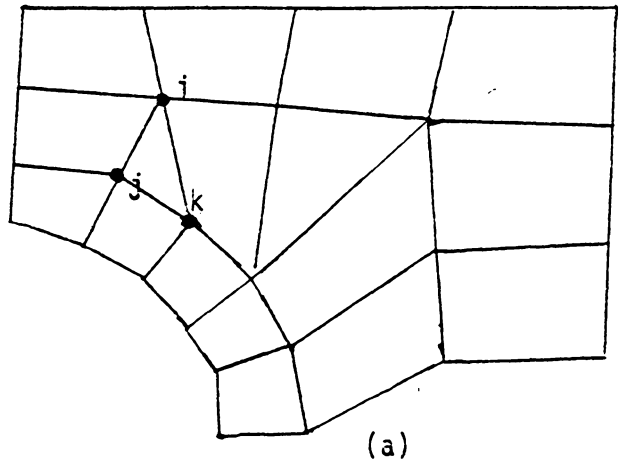
generated, the coordinates of node (2,1) might be recomputed to be the average of the coordinates of nodes (2,2) and the present (2,1). This would only be done if the new location of point (2,1) would make the angle α indicated in Figure 4-3a closer to ninety degrees. Likewise, the coordinates of node (2,2) might be redefined as the average of nodes (2,2) and (2,3).

Another situation in which nodal point coordinates might be averaged together is when nodal points are being generated between two corners. This case is also illustrated in Figure 4-3a at the corner defined by nodes (1,5), (1,6), and (1,7). Two interior nodal points can be generated in approximately the same location. One would be generated using nodes (2,3), (1,5), and (1,6). The other would be generated using nodes (1,5), (1,6), and (1,7). Since the two nodes generated would be very close to each other, they would be averaged together to produce one node, (2,4), as shown in Figure 4-3a.

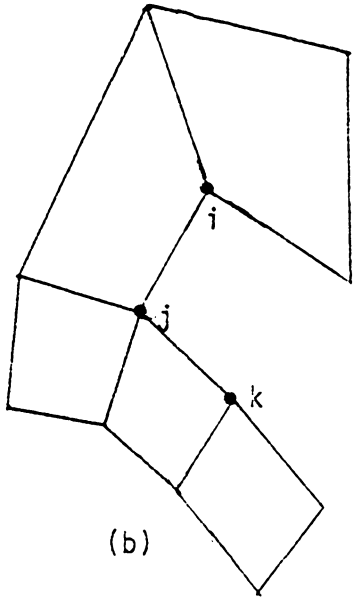
New elements are defined each time new interior nodal points are generated. Once an interior point is generated, its location is compared with the locations of all previously defined nodes to make sure no overlap occurs. If it is found to interfer with another node, the newly generated node is replaced with that node. New elements are then defined using the newly generated node or its replacement, the three nodes used to generate the new node, and the previously generated interior node. Referring again to Figure 4-3a, the element that would be defined after the generation of node (2,1) would be specified by the nodes (1,1), (1,2), (1,3), and (2,1). Node (2,2) would be incorporated into an element defined by nodes (2,1), (1,3), (1,4), and (2,2). The

situation at the next corner where node (2,4) was generated could be handled in one of two ways. The first method, illustrated in Figure 4-3b, would average the coordinates of nodes (2,3) and (2,4) together to produce one node, node (2,3). The corner element would then be defined by (2,3), (1,5), (1,6), and (1,7). Figure 4-3c shows the second approach. Here, node (2,4) would be left alone and two new elements defined. These would be specified by nodes (2,3), (1,5), (1,6), (2,4) and by nodes (2,4), (1,6), (1,7). The choice of one approach over another would depend upon which method produced the best shaped elements for the given situation.

The same subroutines that specify the new elements also pick the new values of node(1), node(2), and node(3) that will be used to generate the next interior nodal point. However, before these values are used for nodal point generation they are checked for possible problems. The first check makes sure that the nodes specified are actually free to be used for nodal point generation. A free node is one which is not completely surrounded by elements, i.e. is free to be used in another element. If the specified values of the three nodes do not pass this test, a searching routine is utilized that locates the next set of free nodes available on the current boundary. If the values do pass the test, they are then checked to make sure that they do not already define a triangular element or are connected to another point is such a way that a quadrilateral element could be formed. Situations of this type are illustrated in Figures 4-4a, 4-4b, and 4-4c. If any of these possiblities occur, a new element is defined and three new nodes are picked to be used as node(1), node(2), and node(3). The entire

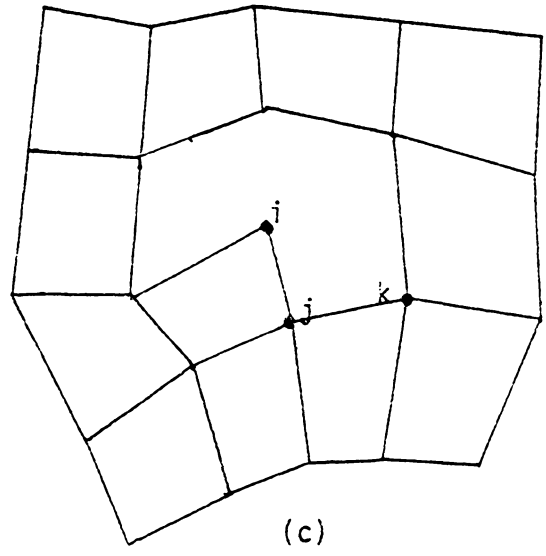


(a)



(b)

i = NODE(1)
j = NODE(2)
k = NODE(3)



(c)

Figure 4-4
Examples of Element Generation without
Generating New Nodes

checking process is then repeated with these three nodes.

Once three nodes are found which pass all of the checks, they are used to generate a new nodal point. The entire process described above is repeated again. This continues until an entire new boundary of nodes has been generated around the structure. After a new boundary is formed, the program examines the nodes on this boundary to find out how many of them are actually free to be used to generate more nodes. The interior angles and values of NUME are calculated at each of the free nodes. Nodal point generation and element definition then starts over again, working around the new boundary. This process is repeated as many times as necessary, ending only when the entire structure has been divided into elements. This state has been reached when the number of free nodes on the boundary is zero, meaning that all nodes in the structure are completely surrounded by elements.

When the entire structure has been divided into elements, there will be several layers of nodes throughout the structure. Each of the nodes will be indicated by two numbers, one indicating its level, the other its sequential position on that level. The next phase of program operation eliminates any duplicate nodes, renumbers all of the nodes consecutively beginning with one in a manner that produces the minimum bandwidth possible for the model, and produces the correctly formatted input file for the finite element analysis.

D. Bandwidth Reduction and Output File Creation

At the end of the third phase of the program, the generated nodes are numbered sequentially in layers around the structure. Each node is indicated by two indices, one indicating the level number and the other the sequential number on that level. Before the output file can be created, these nodes must be renumbered sequentially beginning with one. At the same time, they must be renumbered in a manner that minimizes the bandwidth of the structure. The first step in this process is to renumber the nodes sequentially around the structure, beginning with the first node on the first level and ending with the last node on the last level. During this process, any duplicate nodes that have been generated along segments connecting interior features to the exterior boundary must be eliminated.

As the nodes are being renumbered, any loads and boundary conditions must be respecified to reflect the new numbering of the nodes. The element specifications are also changed to reflect the new numbers and the elimination of duplicate nodes. The result is a completely defined mesh that has a large bandwidth due to the nodal numbering pattern. A second renumbering of the nodes is done using Collin's (8) method. This method, which is explained in Section 2, renumbers the nodes with a smaller bandwidth and creates two arrays, which can be used as cross reference arrays. One of these arrays uses the old node numbers as indices and gives the new node number. For example, if the array in NEWJT, NEWJT(1) is the new number for old node number 1. The other array, gives the old node number for the new node number. In this case, the array is JOINT and JOINT(1) is the old node

number for new node number 1. Once the nodal points have been renumbered more efficiently, these two arrays are used to reference the loads, boundary conditions and element definitions to the new node numbers

After the final numbering of the nodes is done the output file is created. Previously, one output file that was assigned to logical unit 6 was created. That file, given the same name as the input file with a file type of OUT, contains the interpretation of the input data and any messages about problems encountered in trying to create the mesh. If the program successfully creates the mesh, another output file is written. This file is assigned to logical unit 8 and is given a user-defined name. It will be used as the input file to the finite element code MESS. This file will contain the listing of the nodes, elements, loads, and material properties in the free format listing used as input to MESS. Assumptions made during the creation of this file is that there is only one load case, that all elements have the same material properties, and that there is only one type of element. Any of these assumptions can be changed by the user by editing this file and making the necessary changes. For the majority of the problems for which MESS is used, however, these assumptions are valid.

The last thing the program does before ending normal execution, is to display the created mesh on the screen. This is done by calls to the Tektronix PLOT 10 TERMINAL CONTROL SYSTEM graphics package. This package must be linked with the object files of the mesh generation program when the executable image.

E. Program Implementation

Currently this program is implemented upon a VAX 11/780 made by Digital Equipment Corporation and owned by the Mechanical Engineering Department, Virginia Polytechnic Institute and State University, Blacksburg, Virginia. The program logic is based on ANSI FORTRAN-66, but utilizes the capacity of ANSI FORTRAN-77 for variable and subroutine names that exceed six characters. A user's guide for the program is included in Appendix A. Appendix B contains a description of the subroutines that make up the program.

5. Examples

Several examples of meshes created using this program are shown in Figures 5-1 through 5-5. Also included are two examples of the input files used to create these meshes. Samples of the output file from the mesh generator and the file to be used as input to the finite element code MESS are in Figures 5-6 and Figure 5-7.

The first example structure is a square plate. By varying the reference length and the grading values, three different meshes were made. Figure 5-1a shows the input file used to create the first mesh. The keypoints defined in that file are the four corners of the structure and the segments listed are the four sides of the square. The mesh produced from this input should have the same number of elements on each side, as the grading value is the same at all four keypoints. This mesh is shown in Figure 5-1b. As predicted, there are eight elements on each side, a total of 64 elements in the structure. The elements are fairly close to squares, except near the center. There, because of the matching up of the last two nodes generated, the elements are slightly out of shape. In a situation like this, the user may want to edit the nodal point coordinate list and move the center nodes to make more regular shaped elements.

In Figure 5-2 are examples of two more meshes created by changing values in the input file. The mesh in Figure 5-2a resulted from reducing the reference length for the whole mesh to 0.75 from 1.00. This produced smaller elements throughout the whole mesh. In Figure 5-2b, the grading value at keypoint 1 was changed to 0.25 from 1.00. The reference length for this mesh was 1.00.

```
$ INPUT FILE FOR SQUARE PLATE MESH
$ THIS IS THE REFERENCE LENGTH FOR THE ENTIRE MESH
1

$ THESE ARE THE DEFINITIONS OF THE FOUR BOUNDARY KEYPOINTS
1 -4 -4 1
2 4 -4 1
3 4 4 1
4 -4 4 1

$ THESE ARE THE DEFINITIONS OF THE FOUR SEGMENTS
1 1 2
2 2 3
3 3 4
4 4 1

$ THESE ARE THE LOADS--COMPRESSIVE PRESSURE LOADS
$ OF 10000 LBS. ON SEGMENTS 2 AND 4
2 10000
4 10000


$ THIS IS THE SPECIFIED BOUNDARY CONDITION--POINTS 1 AND 2 FIXED
$ IN BOTH THE X AND Y DIRECTIONS
1
2

4
30000000. 0.3
$
```

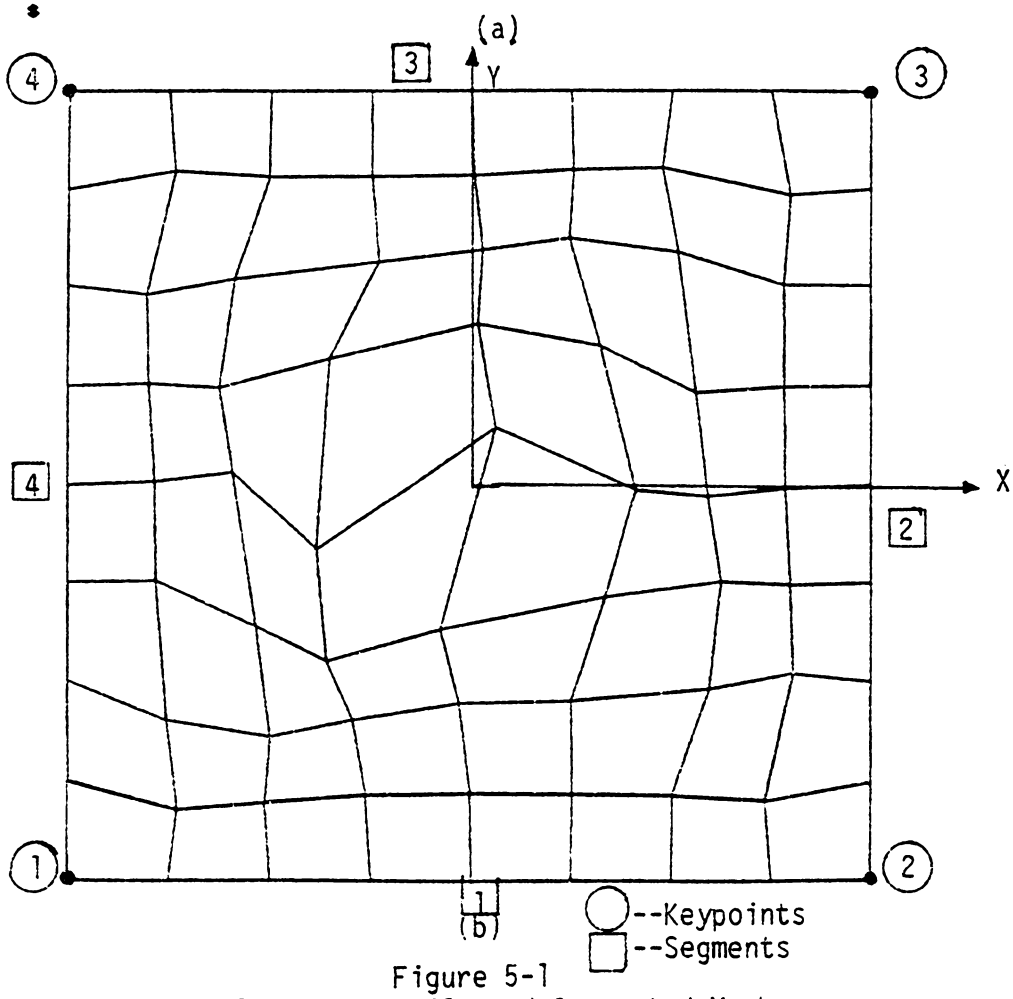
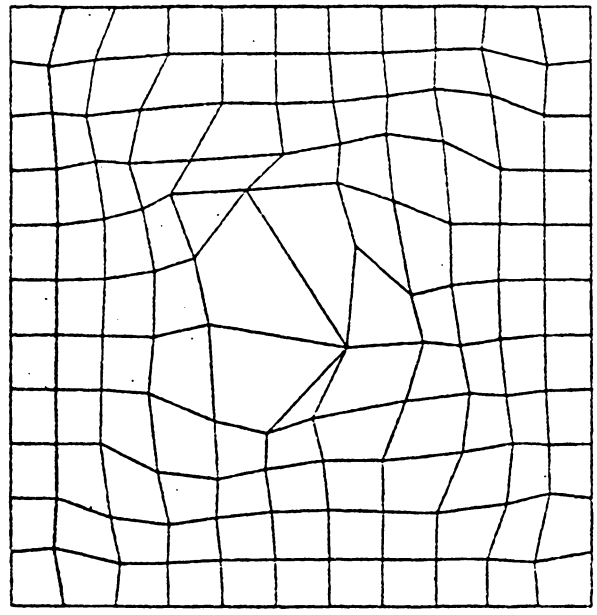
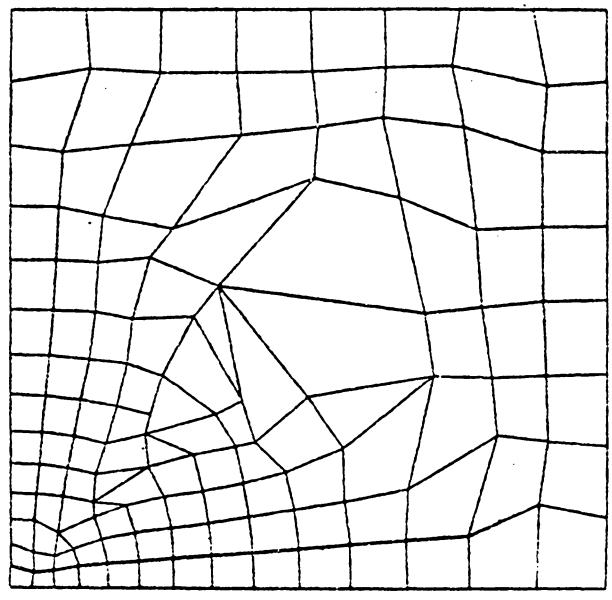


Figure 5-1
Square Plate Input File and Generated Mesh



(a)



(b)

Figure 5-2
Variations on Square Plate Mesh

Figure 5-3a is the input file describing a square plate with a hole in the center. This figure demonstrates the proper way to specify the segment joining an interior cutout feature to the exterior boundary. The mesh produced from this output file is shown in Figure 5-3b. Variations of this example are shown in Figures 5-4a and 5-4b. In Figure 5-4a, the grading value at the keypoint in the upper right hand corner was changed from 0.5 to 0.3. Figure 5-4b is the same as Figure 5-4a, except the reference length for the mesh was changed to 1.5 from 2.0. The effect of changing the grading value at one keypoint is to refine the mesh at that point without affecting the number of elements at other keypoints. Reducing the reference length reduces the size of the elements throughout the entire structure.

Figure 5-5a and Figure 5-5b show the mesh created for the case of an eighth of a square plate with a hole in the center. Because of symmetry, this is the smallest part of a square plate that can be modeled and correct results obtained from the finite element program. The reference length for both of these meshes is 0.75. The grading value in Figure 5-5a is 0.1 at each end of the arc. In Figure 5-5b, this value is 0.075 at each end of the arc. Changing the grading value at the endpoints of the arc produces a finer mesh around the hole without affecting the number of elements along the right edge.

This program produces two output files. One of these is a listing of the input data that the user has specified. The other is the file to be used as the input file for the program MESS. Examples of these files for the square plate example are shown in Figure 5-6 and Figure 5-7.

```
$ INPUT FOR PLATE WITH HOLE IN CENTER
$ THIS IS THE REFERENCE LENGTH FOR THE WHOLE MESH
2

$ THESE DESCRIBE THE BOUNDARY KEYPOINTS
$ THE GRADING VALUE IS OMITTED ON POINT 10 BECAUSE IT
$ IS THE CENTER OF THE HOLE
1 4 -4 .5
2 4 4 .5
3 -4 4 .5
4 -4 -4 .5
5 0 -4 .4
6 0 -2. .25
7 -2. 0 .25
8 0 2. .25
9 2. 0 .25
10 0 0

$ THESE DESCRIBE THE SEGMENTS. NOTE THAT SEGMENT 5 AND
$ SEGMENT 10 ARE THE SAME BUT SPECIFIED IN OPPOSITE
$ DIRECTIONS. THEY ARE THE SEGMENTS JOINING THE INTERIOR
$ HOLE TO THE BOUNDARY
1 1 2
2 2 3
3 3 4
4 4 5
5 5 6
6 6 7 10
7 7 8 10
8 8 9 10
9 9 6 10
10 6 5
11 5 1

$ THESE DESCRIBE THE LOADING--AN INTERNAL PRESSURE OF
$ 1000 PSI
6 1000
7 1200
8 1000
9 1000


$ THIS DESCRIBES THE BOUNDARY CONDITIONS--FIXED IN THE
$ X AND Y DIRECTIONS AT POINTS 1 AND 4
1
4

4
30000000 0.3
```

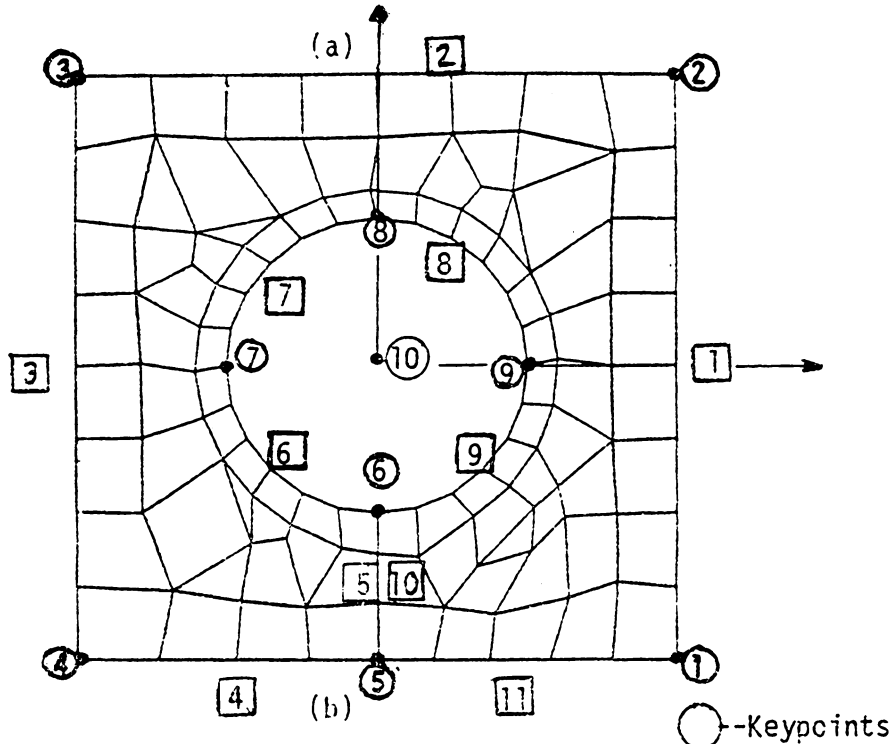
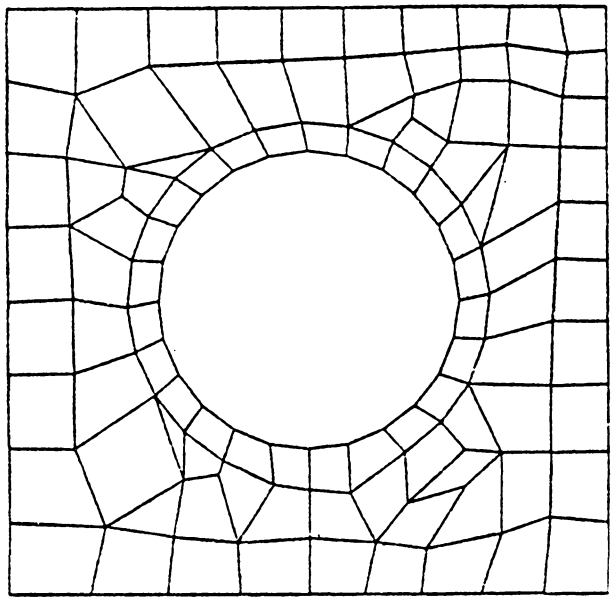
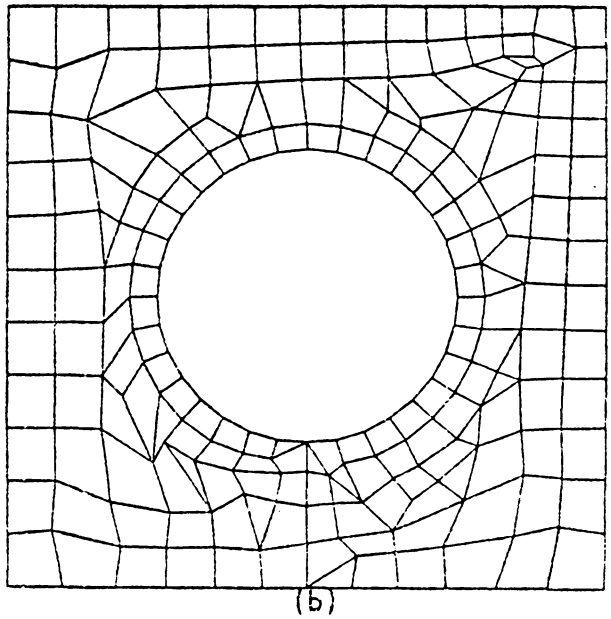


Figure 5-3.
Plate with Hole in Center
Input File and Generated Mesh

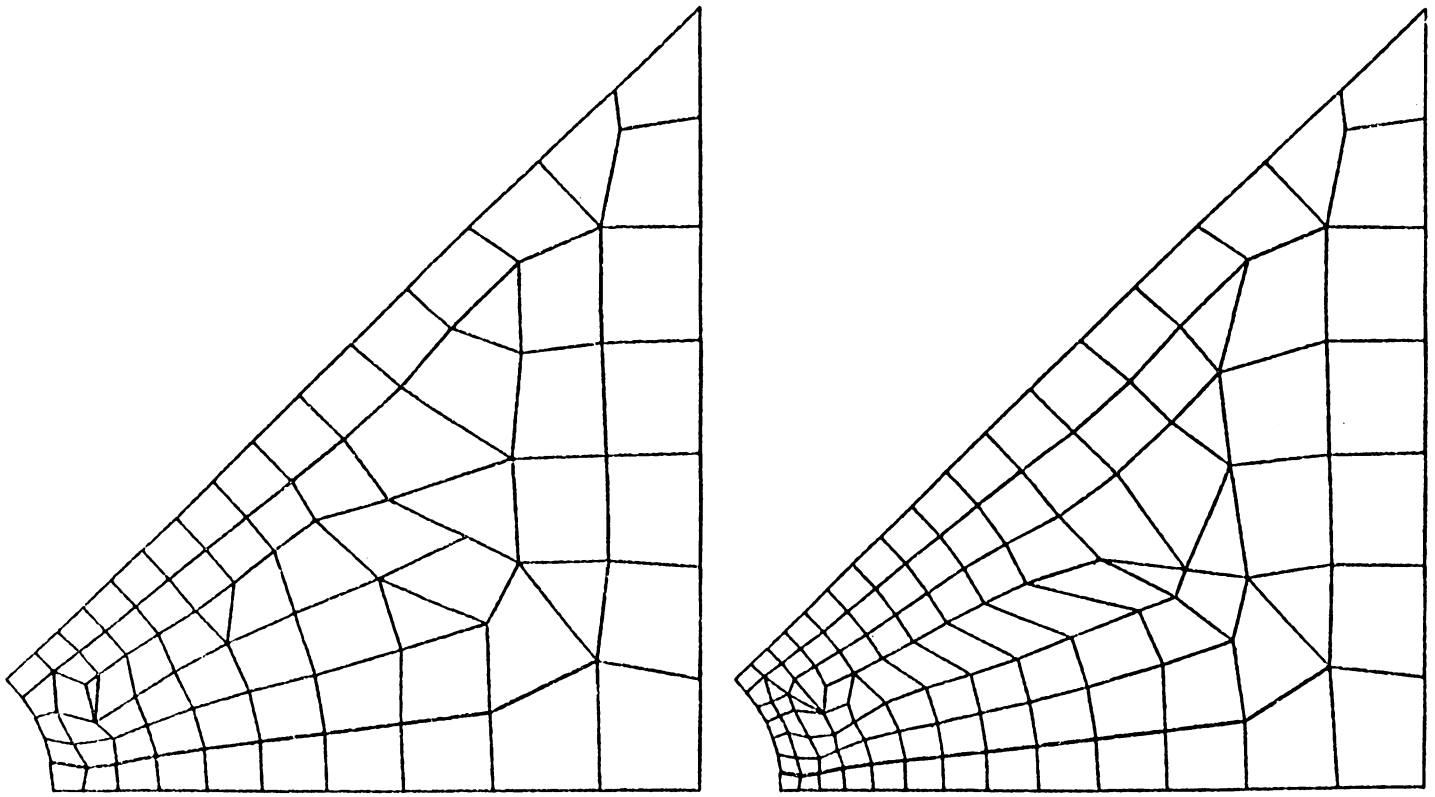


(a)



(b)

Figure 5-4
Variations on Plate with Hole



(a) (b)

Figure 5-5
Eighth of Plate with Hole in Center

```
        BOUNDARY KEYPOINT DATA

   I     X(I)       Y(I)       GRADE(I)

   1 -4.00000 -4.00000        1.00000
   2  4.00000 -4.00000        1.00000
   3  4.00000  4.00000        1.00000
   4 -4.00000  4.00000        1.00000

        SEGMENT DATA

  SEG      BEG(I)      END(I)      CENT(I)

    1           1           2           0
    2           2           3           0
    3           3           4           0
    4           4           1           0

BOUNDARY POINT      CONCENTRATED FORCE-X      CONCENTRATED FORCE-Y
             1              0.0000000000              0.0000000000
             2              0.0000000000              0.0000000000
             3              0.0000000000              0.0000000000
             4              0.0000000000              0.0000000000

SEGMENT      PRESSURE LOAD
      1            0.00000
      2        10000.00000
      3            0.00000
      4        10000.00000

BOUNDARY POINT     1  IS FIXED IN THE X AND Y DIRECTIONS

BOUNDARY POINT     2  IS FIXED IN THE X AND Y DIRECTIONS
```

Figure 5-6

Output File for the Example of Fig. 5-1

```
MESS INPUT FROM MESH
   4
  81
1
1
   1    0    0   -1.00000   -4.00000  0
   2    0    0   -2.00000   -4.00000  0
   3    0    0   -1.04973   -3.13255  0
   4    0    0   -2.05109   -3.21596  0
   5    0    0    0.00000   -4.00000  0
   6    0    0   -0.21165   -3.11224  0
   7    0    0   -3.00000   -4.00000  0
   8    0    0   -2.91956   -3.29213  0
   9    0    0   -1.17152   -2.38712  0
  10    0    0   -1.99975   -2.56853  0
  11    0    0   -0.11894   -2.21652  0
  12    0    0   -3.02071   -2.39564  0
  13    0    0    1.00000   -4.00000  0
  14    0    0    1.00645   -3.12363  0
  15    0    0    0.99668   -2.19641  0
  16    0    0   -4.00000   -3.00000  0
  17    1    1   -4.00000   -4.00000  0
  18    0    0   -4.00000   -2.00000  0
  19    0    0   -2.12781   -1.45598  0
  20    0    0   -1.43467   -1.79372  0
  21    0    0   -0.29953   -1.47845  0
  22    0    0   -3.12254   -0.98856  0
  23    0    0    1.33540   -1.14063  0
  24    0    0   -4.00000   -1.00000  0
  25    0    0    2.00000   -4.00000  0
  26    0    0    2.00607   -3.13419  0
  27    0    0    2.38275   -2.26877  0
  28    0    0    2.49390   -0.98893  0
  29    0    0   -3.14239    0.03126  0
  30    0    0   -2.35349    0.12469  0
  31    0    0   -1.53065   -0.65692  0
  32    0    0    0.25278    0.56940  0
  33    0    0    1.64627   -0.25057  0
  34    0    0   -4.00000    0.00000  0
  35    0    0    2.36029   -0.11485  0
  36    0    0    3.00000   -4.00000  0
  37    0    0    2.93557   -3.19321  0
  38    0    0    3.21877   -1.91627  0
  39    0    0    3.19458   -1.01590  0
  40    0    0    3.14251   -0.03296  0
  41    0    0   -4.00000    1.00000  0
  42    0    0   -3.19487    1.01657  0
  43    0    0   -2.49168    0.97681  0
  44    0    0   -1.39804    1.25930  0
  45    0    0    0.07234    1.60675  0
  46    0    0    1.28562    1.39532  0
  47    0    0    2.24244    0.92228  0
  48    0    0    3.12273    0.98854  0
  49    1    1    4.00000   -4.00000  2
  50    0    0    4.00000   -3.00000  0
  51    0    0    4.00000   -2.00000  0
  52    0    0    4.00000   -1.00000  0
  53    0    0    4.00000    0.00000  0
  54    0    0    4.00000    1.00000  0
  55    0    0   -4.00000    2.00000  0
  56    0    0   -3.00000    1.91652  0
  57    0    0   -2.34315    2.26381  0
  58    0    0   -0.91867    2.24540  0
  59    0    0    0.11102    2.36541  0
  60    0    0    0.97717    2.49197  0
  61    0    0    2.06344    2.34436  0
  62    0    0    3.12626    2.22200  0
  63    0    0    4.00000    2.00000  0
  64    0    0   -4.00000    3.00000  0
  65    0    0   -2.93839    3.18972  0
  66    0    0   -2.00209    3.12622  0
  67    0    0   -0.98856    3.12254  0
  68    0    0    0.03127    3.14239  0
  69    0    0    1.21657    3.19487  0
  70    0    0    1.91650    3.22812  0
  71    0    0    3.18972    2.93835  0
  72    0    0    4.00000    3.00000  0
  73    0    0   -3.00000    4.00000  0
  74    0    0   -4.00000    4.00000  0
  75    0    0   -2.00000    4.00000  0
  76    0    0   -1.00000    4.00000  0
  77    0    0    0.00000    4.00000  0
  78    0    0    1.00000    4.00000  0
  79    0    0    2.00000    4.00000  0
  80    0    0    3.00000    4.00000  0
  81    0    0    4.00000    4.00000  0
64
1
1  30000000.00000000000          0.32000000000
   1   16   17    7    8  1  1
   2    7    2    4    8  1  1
   3    2    1    3    4  1  1
   4    1    5    6    3  1  1
   5    5   13   14    6  1  1
   6   13   25   26   14  1  1
   7   25   36   37   26  1  1
   8   36   49   50   37  1  1
   9   50   51   38   37  1  1
  10   51   52   39   38  1  1
  11   52   53   40   39  1  1
  12   53   54   48   40  1  1
  13   54   63   62   40  1  1
  14   63   72   71   62  1  1
  15   72   81   80   71  1  1
  16   80   79   70   71  1  1
  17   79   78   69   70  1  1
  18   78   77   68   69  1  1
  19   77   76   67   68  1  1
  20   76   75   66   67  1  1
  21   75   73   65   66  1  1
  22   73   74   64   65  1  1
  23   64   55   56   65  1  1
  24   55   41   42   56  1  1
  25   41   34   29   42  1  1
  26   34   24   22   29  1  1
  27   24   18   12   22  1  1
  28   12   16   16    8  1  1
  29   12    8    4   10  1  1
  30    4    3    9   10  1  1
  31    3    6   11    9  1  1
  32    5   14   15   11  1  1
  33   14   26   27   15  1  1
  34   27   26   37   38  1  1
  35   38   39   28   27  1  1
  36   39   40   35   28  1  1
  37   40   48   47   35  1  1
  38   48   62   61   47  1  1
  39   61   62   71   70  1  1
  40   70   69   60   61  1  1
  41   69   68   59   60  1  1
  42   68   67   58   59  1  1
  43   67   66   57   58  1  1
  44   57   66   65   56  1  1
  45   56   42   43   57  1  1
  46   42   29   30   43  1  1
  47   29   22   19   30  1  1
  48   19   22   12   18  1  1
  49   19   18    9   20  1  1
  50    9   11   21   20  1  1
  51   11   15   23   21  1  1
  52   23   15   27   28  1  1
  53   28   35   33   23  1  1
  54   35   47   46   33  1  1
  55   46   47   61   60  1  1
  56   60   59   45   46  1  1
  57   59   58   44   45  1  1
  58   44   58   57   43  1  1
  59   43   32   31   44  1  1
  60   31   30   19   20  1  1
  61   31   20   21   32  1  1
  62   32   21   23   33  1  1
  63   32   33   46   45  1  1
  64   32   45   44   31  1  1
1
  18
  16
1
  10000.00000000000
  17
1
   5000.00000000000
  18
1
  10000.00000000000
  24
1
  10000.00000000000
  34
1
  10000.00000000000
  41
1
  10000.00000000000
  49
1
  -5000.00000000000
  50
1
 -10000.00000000000
  51
1
 -10000.00000000000
  52
1
 -10000.00000000000
  53
1
 -10000.00000000000
  54
1
 -10000.00000000000
  55
1
  10000.00000000000
  63
1
 -10000.00000000000
  64
1
  10000.00000000000
  72
1
 -10000.00000000000
  74
1
   5000.00000000000
  81
1
  -5000.00000000000
```

Figure 5-7
MESS Input File for the Example of Fig. 5-1

6. Conclusions

As shown by the examples of Section 5, this mesh generation program produces satisfactory meshes for a variety of structures. Occasionally, badly shaped elements are produced and the user may be required to reposition a few nodes to correct this condition. Even under these circumstances, the time and effort required to produce a mesh is much less than if other commonly available mesh generators were used or if all nodes and elements were specified manually.

The program as presented, is simple to use, requires little training, and provides the user with both graphical and tabulated output. It is useful for a variety of structures and conditions. The method of specifying both grading values and a reference length greatly simplifies the process of mesh refinement. The major advantage of the program is that the user needs to know only the location of a few selected nodes and relative element sizes in his mesh before beginning to model the structure. It takes much of the tediousness out of preparing a finite element model and eliminates much of the model error that results when the user must specify many nodal point locations and element definitions.

Additional work to be done on the program presented here includes the development of routines to make the program interactive, and improvement of the graphical output to include node and element numbers. It is also desirable to investigate the use of relationships for quadrilaterals, instead of equilateral triangles, to locate the interior nodal points. Future work should also include developing routines to close the structure in a more efficient manner. This would

encompass a method of locating nodes on the last layer such that well shaped elements are formed and the user would not have to relocate nodes to create a more pleasing mesh.

7. References

1. Buell, W.R. and Bush, B.A., "Mesh Generation--A Survey," Transactions of the ASME; Journal of Engineering for Industry, Feb. 1973, pp.332-338.

2. Sadek, E.A., "A Scheme for the Automatic Generation of Triangular Finite Elements," International Journal for Numerical Methods in Engineering, Vol. 15, 1980, pp. 1813-1822.

3. Schoofs, A.J.G.,van Beukering, L.H.Th.M., and Sluiter, M.L.C., "A General Purpose Two-Dimensional Mesh Generator," Engineering Software - Proceedings of the 1st International Conference, 1979, pp. 3-17.

4. Kleinstreuer, C., "A Triangular Finite Element Mesh Generator for Fluid Dynamic Systems of Arbitrary Geometry," International Journal for Numerical Methods in Engineering, Vol. 15, 1980, pp. 1325-1334.

5. Bykat, A., "Automatic Generation of Triangular Grid: I-Subdivision of a General Polygon into Convex Subregions. II-Triangulation of Convex Polygons," International Journal for Numerical Methods in Engineering, Vol. 10, 1976, pp. 1329-1342.

6. Harber, R., et. al., "A General Two-Dimensional Graphical Finite Element Preprocessor Utilizing Discrete Transfinite Mappings," International Journal for Numerical Methods in Engineering, Vol. 17, 1981, pp. 1015-1044.

7. Brown, P.R., "A Non-Interactive Method for the Automatic Generation of Finite Element Meshes using the Schwarz-Christoffel Transformation," Computer Methods in Applied Mechanics and Engineering, Vol. 25, 1981, pp.101-126.

8. Collins, R.J., "Bandwidth Reduction by Automatic Renumbering," International Journal for Numerical Methods in Engineering, Vol 6, 1973, pp. 345- 356.

9. Cuthill, E., McKee, J., "Reducing the Bandwidth of Sparse Symmetric Matrices," Proceedings of the 24th National Conference, ACM, San Francisco, 1969, pp. 157-172.

Appendix A

User's Guide

Mesh Generation Program

Input sections are separated by blank cards while comment cards must have a $ sign in column one. Allowable characters are numbers, periods, E, X, Y, plus sign, negative sign, commas, and single quotes. If one is not running the program interactively, X and Y must be enclosed by single quotes (') in the input file. Data may be separated by spaces or commas.

I. Reference Length

This value affects the overall size of the mesh. The smaller the value, the finer the mesh will be. It is suggested that one start with a value of one and adjust it after the first run to obtain the desired mesh size.

II. Boundary Keypoint Data

The entries in this section are the boundary keypoint number, the coordinates of the keypoint, and the grading value at the keypoint. These values are entered for each keypoint needed to describe the structure and/or loading points. The keypoints may be entered in any order but no numbers can be omitted. If a keypoint is the center of an arc and does not lie on the boundary of the structure, the grading value may be entered as zero for that point or may be omitted.

III. Segment Data

The entries in this section are the segment number, the beginning keypoint, the end keypoint, and, if the segment is a circular arc, the circle center keypoint. Segments may be entered in any order, but the

first segment entered must be specified as going in a counter-clockwise direction around the structure. The segments must specify a closed, singly connected boundary about the structure.

Arcs must be specified by three keypoints and must be specified in ninety degree or less segments, relative to the global X,Y coordinate system through the center of the arc. Figure A-1 shows examples of correct and incorrect ways to specify an arc.

If there is an interior feature such as a hole, the segment specifying the connection of the interior feature to the exterior boundary must be specified twice. The first specification would be from the exterior boundary to the interior feature and the second would be from the interior feature to the exterior boundary.

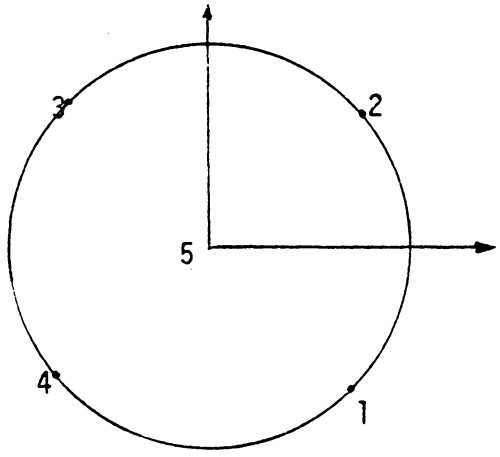
IV. Load Data

Two type of loads are allowed, concentrated loads at keypoints, and pressure loads along segments. To specify a concentrated load, one must enter the keypoint number, the direction of the load ('X' or 'Y'), and the magnitude of the load. If there is no sign with the magnitude of the load, it is assumed positive.

Pressure loads are specifed by entering the segment number and the load magnitude. Pressure loads are considered to act perpendicular to the segment and a positive magnitude is associated with a compressive pressure. A negative magnitude is interpreted as a tensile pressure on the segment.

V. Segment Boundary Conditions

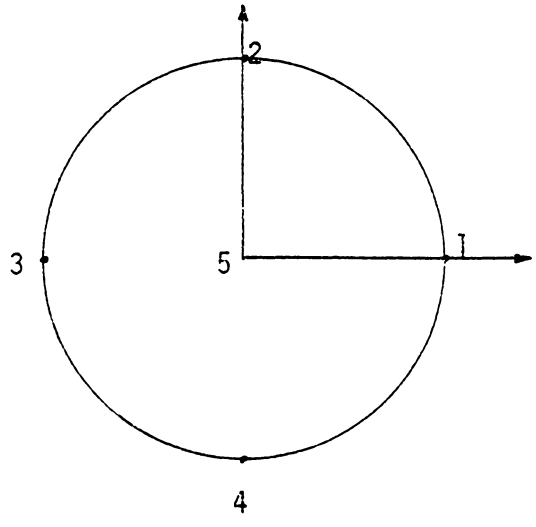
A segment may be fixed in the X, the Y, or both the X and the Y directions. To fix a segment in both X and Y directions, specify only



Segment	Points
1	1-2-5
2	2-3-5
3	3-4-5
4	4-1-5

(a)

Incorrect



Segment	Points
1	1-2-5
2	2-3-5
3	3-4-5
4	4-1-5

(b)

Correct

Figure A-1
Examples of Segment Specification

the segment number. To fix a segment in either the X or the Y directions, specify the segment number and the direction in which it is to be fixed ('X' or 'Y').

VI. Point Boundary Conditions

A point may be fixed in the X, the Y, or both the X and the Y directions. To fix a point in both the X and the Y directions, specify only the point number. To fix a point in either the X or the Y directions, specify the point number and the direction in which it is to be fixed ('X' or 'Y').

VII. Element Type and Material Properties

On the first line of this section, enter the element type to be used. Axisymmetric elements are indicated by entering a 3 while isoparametric elements are indicated by entering a 4.

On the second line of this section, enter the value of Young's modulus and Poisson's ratio.

VIII. Error Conditions

Several error messages may be printed out during the execution of the program. These are explained below.

1. THIS LINE CONTAINS CHARACTERS THAT CANNOT BE INTERPRETED

   The specified line contains characters that are not on the list of allowable characters. Check your input data for any stray characters ,including control characters, that may have been accidentally entered.

2. THIS LINE DOES NOT CONTAIN THE CORRECT NUMBER OF DATA ENTRIES

   The specified line either contains too many or too few entries for the section of data. Check the specifications for the

section above.

3. BOUNDARY POINT USED ONCE--STRUCTURE NOT CLOSED

   The specified boundary point has only been used once in the segment definition. This applies only to boundary points which actually lie on the structure, not those used as centers of arcs. For the specified boundary to be completely closed about the structure, each boundary point must have been used twice. Check your segment definitions.

4. SEGMENT NUMBER HAS BEEN INCORRECTLY SPECIFIED

   The specified segment has not been entered correctly. This usually refers to arcs which have not been specified in ninety degree segments. Refer to the section on segment specification above.

5. CONVERGENCE CANNOT BE ACHIEVED FOR SEGMENT

   This means that the program cannot locate nodes on the specified segment within the required accuracy. A common cause of this error is specifying the reference length and the grading value at the endpoints of the segment such that the product of the two is greater than the length of the segment. A solution is to change either the reference length or the grading value at the endpoints of the segment.

6. NUMBER OF ELEMENTS GENERATED HAS EXCEEDED ALLOWABLE

   This means that the program has generated 300 elements and is not through. The maximum number of elements in the entire structure is 300. A probable cause of this error is creating too fine a mesh. For the intended use of this program, 300

elements is ample. One should reexamine the input data and determine if the mesh must be as small as specified. A possible fix would be to increase the reference length.

7. NUMBER OF NODAL POINTS GENERATED ON LEVEL HAS EXCEEDED ALLOWABLE

   The program allows 100 nodal points on each level and a total of 500 nodal points for the whole structure. These limits should be practical for the type of problems for which this program is intended. This message means that on the specified level, 100 nodal points have been generated and the level is not complete. Reexamine the specified grading values and reference length to determine if the mesh has been over specified.

8. NUMBER OF NODAL POINTS IN STRUCTURE HAS EXCEEDED ALLOWABLE

   This means that the program has generated a total of 500 nodes and is not through. Reexamine the specified grading values and reference length to determine if the mesh has been overspecified.

9. THE FOLLOWING NODES DO NOT MEET ANY OF THE CONDITIONS

   This message indicates that the program has encountered a combination of nodes from which it cannot generate a new element. See the person responsible for maintaining the program if this message occurs.

Appendix B

Subroutine Description

Subroutine	Description
CHECK	Checks the assigned values of node(1), node(2), and node(3) to make sure they are free for nodal point generation. Checks the assigned values of node(1), node(2), and node(3) to see if they already form an element
CHECK1	Calculates the distance and angle between all nodes on the current level and the new nodes on the next level as they are generated.
CHECK2	Checks the location of the newly generated nodes to make sure they do not overlap or interfer with previously generated nodes.
ELEMGEN	Defines elements formed with newly generated nodal points
ELEM1	Defines elements when the angle at either node(1), node(2), or node(3) is less than 30 degrees. Assigns new values of node(1), node(2), and node(3).
ELEM2	Locates trial nodal points when the angle at node(1), node(2), or node(3) is between 30 and 150 degrees.
ELEM3	Locates trial nodal points when the angle at node(1), node(2), or node(3) is between 150 and 210 degrees.
ELEM4	Locates trial nodal points when the angle at node(1), node(2), or node(3) is between 210 and 270 degrees.
ELEM5	Locates trial nodal points when the angle at node(1), node(2), or node(3) is between 270 and 330 degrees.
ELEM6	Locates trial nodal points when the angle at node(1), node(2), or node(3) is between 330 and 360 degrees.
FIN3	Defines elements when there are only three free nodes on the new level

FIN4	Defines elements when there are only four free nodes on the new level
FIN5	Defines elements when there are only five free nodes on the new level
FIN6	Defines elements when there are only six free nodes on the new level
FIX1	Defines elements if there is interference between the generated nodes and the nodes on the current level
FIX2	Defines elements if there is interference between the newly generated nodes and the previously generated nodes on the new level
INITIAL	Initializes the arrays containing the distance and angle between nodes on the current level and between nodes on the level being generated
INTANG	Checks the angle calculated at a nodal point using the dot product to make sure it is the correct angle.
INTCHECK	Checks to see if there are any segments used twice indicating the presence of an internal feature such as a hole
LOADCALC	Calculates the load at each boundary node due to an applied pressure along the boundary.
MESH	Controls the generation of internal nodes and the definition of elements by calling all necessary subroutines.
MESH1	Calls all other subroutines--is the main routine
NPCHK	Locates free nodes adjacent to node(1), node(2), and node(3) for calculating angles
NPGEN	Generates boundary nodal points and sets up load arrays and boundary condition arrays.
NPLOC	Locates interior nodal point when averaging is not needed
OPTINUM	Does the renumbering of the nodes to minimize the bandwidth

OUTPUT	Eliminates any duplicate nodes and writes the file to be used as the input file for the finite element code
READ	Checks, interprets, and assigns the input data
REORDER	Redefines the segments so that they form a continuous, closed boundary. Reorders the keypoint numbers so that all points not on the structure are at the end of the list
RESTART	Calculates the angle at all free nodes on the newly generated level, defines a starting point on the new level, and starts the nodal point generation process over again.

The vita has been removed from the scanned document

A TWO-DIMENSIONAL FINITE ELEMENT MESH GENERATOR WITH AUTOMATIC TRANSITIONING CAPABILITY

by

CYNTHIA C. JARA-ALMONTE

(ABSTRACT)

A simple, user friendly, mesh generator designed as a pre-processor for two-dimensional finite element codes is presented. The program generates both triangular and quadrilateral two-dimensional elements and has automatic transitioning capability. The required input is the definition of the boundary of the structure, a measure of the relative size of the mesh at the key geometry points, and loading information.

Both boundary and interior nodal points are generated. The user is not required to develop the overall node and element plan and is not limited to quadrilateral regions with equal number of nodes on opposing sides. The generated mesh density as controlled by the user will automatically transition from low to high density areas.

A bandwidth reduction routine is included to renumber the generated nodes most efficiently. An output file is created of node, element, and nodal load definitions which is organized for input to a finite element program.