CHAPTER-3

AUTOMATIC MESH GENERATION

3.1 Introduction

To solve the electro-magnetic field problems by the finite element method, the domain of interest is first subdivided into a number of discrete subdomains or elements. Variational, Galerkin’s or other techniques are used to obtain an integral formulation, which is usually referred to as ‘weak’ form. After evaluation of the integrals a set of algebraic equations, in matrix form, is obtained for each element. Once such equations are formulated for all the elements, the global matrix is then assembled and final solution obtained.

The size and placement of these elements largely determines the accuracy with which the problem can be solved. Reducing the element size and thereby increasing the number of nodal points usually yields a more accurate solution but at the expense of increased CPU time and memory requirement. The key to efficient and economic solution of the problems is not merely the number of points and elements but also their placement. Regions with larger gradients will need a high mesh density, with quiescent regions requiring a comparatively coarser mesh. The engineering judgment comes from the behavior of the system and an estimate of the computational cost involved in the mesh refinement. Therefore the production of the ‘good’ finite element meshes remains very much related to the ‘art’ of the user where experience and familiarity plays a large part. The inexperienced user will very often produce very inefficient mesh models, which lead to inaccurate solutions and wastage of computer resources.

The finite element technique and the subsequent result retrieval can be divided into three sections:
Preprocessor
Processor
Postprocessor

In principle, these amounts to setting up the problem for solution,
carrying out the mathematical work, and evaluating the answers to produce some useful results.

**Preprocessor**

Preprocessor [61,62] involves the preparation of data, such as nodal coordinates, connectivity, boundary conditions and loading material information. A major step in preprocessing consists of describing the geometric shapes and size of the object or situation to be analyzed and stating how the object is to be discretized for purpose of analysis. These two activities are sharply separated in some system, while in other they may merge almost imperceptibly into single activity. The discretization has to be performed automatically or nearly so, with minimal user intervention. In some cases user ought to have full control of all approximation involved and ought therefore be allowed as active role as he wishes in any working step.

In system based on FEM, two distinct approaches are commonly used for generating and storing geometric information about the object to be analyzed. In one approach, a numeric representations of the object itself is produced. In the other, the shape of object is implied by the finite-element mesh which is used in mathematical analysis.

In FEM, every geometric object of interest is viewed as composite structure made up of more or less standardized parts or elements. The finite elements most commonly used in current FEA packages are triangular. It is generally recognized that triangular elements can best adapt the boundaries of domain, and very often only triangular elements are able to fill those domains with vary irregular boundaries. However, the quadrilateral elements are numerically superior than triangular.

**Processor**

Processors are really problem-defining programs, which arrange for all necessary physical data to be assembled. Processors are programs for constructing the systems of algebraic equations which model the physical situation mathematically and produce solutions. [63] Processors are very frequently used for batch processes, requiring little if any interactive control by the designer. These can therefore be implemented on large remote computers as an alternative to the
local workstation. In most magnetic problems, and in nearly all currently available packaged systems, and in the problems analysed in this work the end product of the processor is a set of potentials describing the magnetic field in the entire model.

**Post Processor**

The potential solution produced by post processor provides the raw information about the boundary conditions of the device, subject to the approximations made in the modelling and solution phases. These data are not necessarily, indeed not usually, what is required by the designer, whose need are much more likely to include derived quantities like impedance, power densities, or mechanical forces. However, the required result may be extracted from the potential solution by further mathematical manipulation. The post processing facilities therefore must provide two main functional ingredients: mathematical manipulations for treating the available solution data and graphical tools for displaying the result thus obtained.

The ensemble of activities by which the engineering significance of mathematical field solution is evaluated is generally termed post processing. It includes the derivation of specific numerical result as well as their graphical presentation. A postprocessor is a major part of any design system since it allows relevant data to be extracted from the solution and presented in a way that has meaning to the user.

In postprocessing, the various mathematical operations e.g. differentiation, integration, multiplications, and whatever else may be required cannot be performed on the exact fields themselves, but must use the finite element approximations computed by processing programs instead.

**3.2 Automatic Mesh Generation**

A key step of the finite element method for numerical computation is mesh generation. One is given a domain (such as a polygon or a more realistic versions of the problem allow curved domain boundaries) and must partition it into simple "elements" meeting in well-defined ways [64-66]. All elements should be "well shaped" (which means different things in different situations, but generally involves bounds on the angles or aspect ratio of the
elements). One distinguishes "structured" and "unstructured" meshes by the way the elements meet; a Structured mesh is one in which the elements have the topology of a regular grid. Structured meshes are typically easier to compute with (saving a constant factor in runtime) but may require more elements or worse-shaped elements. Unstructured meshes are often computed using Delaunay Triangulations of point sets; however there are quite varied approaches for selecting the points to be triangulated.

There has now been considerable theoretical work in the geometry community on these problems, complementing and building on practical work in the numerical community. There is also beginning to be a convergence of these communities, in which theoretical work is fed back into practical mesh generation applications. Theoretically, the preferred type of mesh is the triangulation, but in mesh generation practice quadrilaterals or higher dimensional cubical element shapes are preferred (both because fewer are typically needed and because they have better numerical properties). Remaining problem areas in the theory of meshing include triangulations in dimensions higher than two, meshes with cubical elements, mesh smoothing, mesh decimation and multigrid methods, and data structures for efficient implementation of meshing algorithms. There has also been some theoretical work on using geometry to partition meshes by finding small separators of their underlying graphs.

Automatic mesh generation could be a part of the refinement algorithm. In automatic mesh generation the user supplies the input data plus an accuracy parameter and the algorithm generate adaptively an optimal mesh that achieves the desired accuracy in the solution process. It is also possible to add points to the mesh recursively and also to refine mesh anywhere without producing elements of distorted shape. The technique is called Adaptive Finite Element Method. It, therefore, is based on coupling of two different aspects of the finite element method, viz mesh generation and error analysis. If this is done automatically without user intervention, the process is called Self-Adaptive. This has been taken up in more details in Chapter-4.
To start, the problem to be analyzed is described by a very crude finite elements mesh, and a rough solution is produced. Error estimates are made for this solution and a new, more refined mesh is produced. Error estimates are made for this solution and a new, more refined mesh is produced. A solution is computed on the finer mesh, an error estimate is made, then another solution and so on until the solution has desired accuracy. The initial finite element mesh to be used for self adaptive finite element analysis has to be generated either manually or automatically. Using the input data, the procedure for automatic mesh generation has been developed and consists of following main steps:

(i) Node generation
(ii) Formation of elements
(iii) Mesh optimization. This has been taken up in section 3.5
(iv) Mesh refinement. This has been coupled with error estimation techniques and has been taken up in Chapter-4.

3.2.1. Node Generation

In continuum problem of any dimension, the field variable possesses infinitely many values because it is a function of each generic point in the body or subregion. Consequently the problem is one with an infinite number of unknowns. The approximating functions to estimate unknown are defined in term of value of field variables at specified points called nodal points. Nodes usually lie on the element boundaries where adjacent elements are considered to be connected. In addition to the boundary nodes an element may also have a few interior nodes. The nodal values of field variable and the interpolation functions for the elements completely define the behaviour of the field variables within the elements. For the finite elements representations of a problem the nodal values of field variable become the unknowns. Once these unknown are found the interpolation function define the field variable throughout the assemblage of elements. Node points should be placed in region wherever values of variables are sought or specified.
Nodes should be more closely spaced in region where the variable parameters is expected to vary rapidly. Next, we begin connecting nodes points with straight lines to form a network of quadrilateral through the region. When working with bodies composed of different materials, the parting lines of materials should be represented by the boundaries of quadrilateral.

3.2.2. Formation Of Elements

The discretization of the continuum or solution region into elements reduces the infinite number of unknown values to finite number of unknowns and by expressing the unknown field variable in terms of assumed approximating functions within each elements. The mesh is composed of quadrilateral finite elements. Several existing systems regard mesh as being built up by joining together submeshes each of which may in turn consists of one or more elements. At the elementary level of small submeshes, elements may be specified explicitly or implicitly through conventional rules for subdivisions of simple primitive shapes such as rectangles. Usually the submeshes are not given as part of data in problems, all that is available is finite set of distinct nodes at each of which the function and any relevant derivatives are evaluated.

The solution region can be discretized into different shapes such as rectangular, triangular or in quadrilateral elements. Once it has been properly discretized, we are ready to establish the system topology. This is done by numbering the nodes consecutively starting from 1. The elements must also be numbered consecutively starting from 1. Once the number have been assigned they must be fixed because the interpolation of final output relies on numbering schemes. The node numbers and element numbers should be stored for later reference. They will comprise part of input data to be used for processing. Most current systems build and maintain data base in which elements are explicitly specified. To specify a rectangular elements ambiguously, it suffices to give the locations of its nodal points and to
3.3 *Triangular Mesh Generation*

3.3.1 **Automatic Point Creation**

Points for connectivity by Delaunay algorithm can be derived in many ways. Two ways which can be used include Superposition Method and points generated from an independent technique. The former technique give rise to good quality grids in the interior of regions, but the grid quality can deteriorate where the triangles are constrained by the boundaries. The latter approach is restrictive for general geometries.

Hence a new method is used which is flexible, easy and efficient to implement, requires manual user input and provide good mesh.

3.3.2 **Point Creation Driven By The Boundary Point Distribution**

For mesh generation purposes the boundary of the domain is defined by the points and associated connectivities. It is assumed that the grid points on the surface reflect appropriate variations in surface slope and curvature. Ideally any method which automatically creates points should ensure that the boundary points distribution is extended into the domain in a spatially smooth manner.

Consider boundary line segment on which points have been distributed which encloses a domain. It is required to distribute points within region so as to construct a smooth distribution of points. For each point on boundary a typical length of scale for the point can be computed as the average of the two lengths of the connected edges.

No points should be placed within a distance comparable to the defined length scale since this would inevitably define badly formed triangles.
Hence for each point, it is appropriate to define a region $T(i)$ within which no interior point should be placed.

In a Delaunay Triangulation Algorithm the surface or boundary points are connected together to form an initial triangulation. Points can be placed anywhere within the interior but not inside any of region $T(i)$ already identified. Hence points are placed at the centroid of each of formed triangles and then a test is performed to determine if any of the points lie within any $T(i)$; it must be rejected if it lies within $T(i)$ and it can be included if it does not lie within $T(i)$ and connected using Delaunay triangulation algorithm. Once a point has been inserted, it too must have associated with it length scale which defines an effective region $T(i)$ for point exclusion. A newly inserted point takes a length scale from interpolation of the length scales from the nodes which form the triangle from which they are created. In this way a smooth transition between boundaries of interior points can be insured. This process of point insertion continues until no point can be added because the union of all $T(i)$ covers the entire interior domain.

The details of the implementation of this procedure in 2-D are:
1. Compute the point distribution function for each boundary point $r(o) = (x, y)$, that is for point $o$

$$d_p(o) = |r(i) - r(o)|$$

where $|$ is the Euclidean distance and it is assumed that the point $o$ is surrounded by $M$ points $i = 1, M$.
2. Generate the Delaunay triangulation of the boundary points.
3. Initialize the number of the interior field points created, $N = 0$.
4. For all triangle within the domain:

a) Define a prospective point, $Q$, to be at centroid of the triangle.

b) Derive the point distribution, $d_pQ$, for the point $Q$, by interpolating

the point distribution function from the nodes of the triangles, $m = 1, 2, 3$. 
c) Compute the distance $d_m, m=1,2,3$ from the prospective point $Q$ to each of three points of triangle.

If $\{d_m < \alpha*dpq\}$ for any $m = 1,2,3$, then

Reject the point: Return to the beginning of step 4.

If $\{d_m > \alpha*dpq\}$ for all $m = 1,2,3$ then

Compute the distance $S_j, (j = 1, \ldots, N)$ from the prospective point $Q$ to other points to be inserted, $P_j, j = 1 \ldots, N$.

If $\{S_j < \beta*dpm\}$ then

Reject the point: Return to the beginning of the step 4.

If $\{S_j > \beta*dpm\}$ then

Accept the point $Q$ for insertion by Delaunay triangulation Algorithm and include $Q$ in the list $P_j, j = 1, \ldots, N$.

d) Assign the interpolated value of the point distribution function, $dpq$

To the new node, $P_n$.

e) Select next triangle.

5. If $N = 0$ go to step 7.

6. Perform the delaunay triangulation of the derived points, $P_j, j = 1, \ldots, N$

Go to step 3.

7. Smooth the mesh.

The coefficient $\alpha$ controls the grid point density by changing the allowable shape of all triangles, while $\beta$ has an influence on the triangulation by not allowing points within specified distance of each other to be inserted in the same sweep of the triangles within the field.
3.3.3. **Generation(Void)**

Initial triangulation having been done, generation function calls three functions:
1) point_dist();
2) centroid();
3) cent_dist();

**Point Distribution Function**

After the initial round of triangulation, each of the boundary points is associated with at least one triangle. We then proceed to find the Point Distribution Function for each boundary node, where the function is defined as:

$$M_{dpn} = \frac{1}{M} \sum_{i=1}^{M} |r(i) - r(n)| = 1$$

Here $N =$ total number of boundary nodes, $M = N - 1$, and $n =$ nth boundary node [10][11].

The basic principle is to summate the Euclidean distances from the node under consideration, to each of the other boundary nodes.

We begin by creating two [500] sized one dimensional integer arrays called temp_x and temp_y. Five integer variables used are i, j, k and $x_1, y_1$.

In order to calculate Euclidean distances from the node under consideration to each of the other nodes it is required to exclude this node from calculations. This can be done by dividing the range into two parts:

- Node number greater than the base node.
- Node number less than the base node.

Assuming that we are at nth node, two loops are executed which are identical except for their ranges. The first loop is from the 0th node to n-1th node and the second is from the n+1th node to the Nth node.
The Euclidean distances are found as the absolute differences of the x and y coordinates of the nodes which are found in separate data file. They are then summated and the result stored in a variable called sum. This is then divided by M which is one less than number of boundary points.

This procedure of calculating the p.d.f is repeated for all N points and the results stored in a new data file.

Centroid

Initial triangulation of boundary points, already having been done, the total no. of elements are stored in variable element. The input file consists of total no of elements and connecting nodes which can be read from the file count, node1, node2 and node3.

Another input file consists of x and y coordinates of all three nodes. A loop is set up starting from i = 0 to i = ele, and the centroid for each element is calculated using formula:

\[
\text{cent } x(i) = \frac{x_1 + x_2 + x_3}{3}; \\
\text{cent } y(i) = \frac{y_1 + y_2 + y_3}{3};
\]

After calculating, centroid co-ordinates for each element, the input file is rewound.

Cent_Dist

In this function, the centroid co-ordinates having been calculated, the point distribution function is determined by interpolation. A loop is formed from i = 0 to i = ele, input values, element no., node1, node2, node3 are obtained from input file.

The point distribution for each node is already calculated. Using that the point distribution function for each centroid can be calculated as shown below:

\[
\text{cent } \text{dest}[i] = \frac{\text{bdens}[\text{node1}] + \text{bdens}[\text{node2}] + \text{bdens}[\text{node3}]}{3}
\]

After this the input file is again rewound.
3.4 Quadrilateral Mesh Generation

Discretization of domain into quadrilateral mesh is based on facts that:

1. A Region can always be subdivided entirely into quadrilaterals if polygon which forms the boundary of the region has an even number of sides.

2. A Quadrilateral can be formed by two triangles which share a common side. For quadrilateral mesh the number of boundary nodes must be even. If number of boundary nodes are odd then firstly make boundary nodes even by creating an extra node on the line joining the first node and the last node. Then the curvilinear boundary of the domain is transformed into a polygon with an even number of sides. The generation of an entire quadrilateral element mesh over the domain is therefore possible.

The main feature of quadrilateral generation process is that first two triangle which have common side are generated and then triangles are combined to form a quadrilateral. The process continues until the domain is fully covered by non-overlapping quadrilaterals.

The advantage of forming a quadrilateral by two triangles is that quadrilateral with internal angle equal to and greater than 80 are allowed to be generated. These types of elements play an important role in the generation of transition which are often necessary in a strongly graded mesh. This can be readily accomplished by almost any existing triangular mesh generator with some modifications. The procedure for quadrilateral mesh generation include:

Making The Boundary Points Even

First step in quadrilateral mesh generation is to make boundary nodes even. Hence for the figures having odd number of boundary nodes, an extra node is generated at the mid point of the line joining the
last node with first node at the time of node generation for triangular meshing.
This extra node now becomes last node, and accordingly the list containing the coordinates and the number of boundary nodes is appended.
After this is achieved, the algorithm for triangular mesh generation is invoked and following steps are performed.

2. Now consider those nodes, each one of which is connected to only one triangular elements. These will be boundary nodes such an element will have a connecting with only one another triangular element. The quadrilateral is thus formed by joining the two elements by omitting the line common to them both. Thus quadrilaterals can be generated by the following methods:
   1. One Node criterion
   2. Two Node criterion
   3. Three Node criterion

One Node Criterion
In one node method the node which is connected to only one triangular element is under consideration as shown in fig 3.1 node A is connected to only one triangular element 1 so elements 1 and 2 can be combined to form quadrilateral element ABCD, by omitting side BD.

Two Node Criterion
In this criterion, node which is connected to two triangular elements is under consideration as shown in fig 3.2 Node B which is connected to elements number 1 and 2 is under consideration and side BD is omitted to form quadrilateral element ABCD.

Three Node Criterion
In this criterion, node which is connected to three different triangular elements is under consideration as shown in fig 3.3 Node A is connected to
three different elements numbered 1, 2 and 3 and a straight line AD is drawn perpendicular to side BC and sides AB and AC are omitted to form two quadrilaterals ADBF and AECD.

3.5 Mesh Optimization

In interactive FEA system the creation and modification of geometric models and meshes become a matter of editing the underlying data structure i.e. of adding, removing or modifying its part until the data structure adequately describes the object to be analyzed [67].

The mesh quality is an important condition for the FEM mesh to ensure that a good FEM approximation to the sought solution is calculated.

The first essential condition is that the distortion $D$ of the elements measured by the ratio of the element size $H$ and the radius $R$ of the largest ball in the element:

$$D = \frac{H}{R}$$

The distortion should be smaller than 5, but values in the order of 10 are acceptable. Larger values of the distortion are not safe. Values with a distortion greater than 15 are not suitable for reliable FEM calculations. The following picture shows an element with a small and one with a large distortion:

In addition to the distortion of the element, the bending of the element edges has to be moderate. The distance of the node on the element edge to the center point of the edge vertices should be smaller than 15% of the distance of the vertices.

3.5.1. Internal Node Filtering

The nodes which have been generated at the centroid of each triangles are called Prospective Points, as it is not sure whether they will be accepted or rejected. So Dual Pass Method is used to filter out some of these prospective points and each method has a different criterion. Both these methods are explained below:
**Alpha Criterion**

This method involves comparing the values of three ‘node to centroid’ distances of each triangle with the value of $\alpha$ times the interpolated value of point distribution function at the centroid. Based on whether the criterion is met or not, the prospective point is accepted or rejected.

Variables used are: Count, node1, node2, node3 -- the values of which are stored in data file. $x1, y1, x2, y2, x3, y3$ the rectangular coordinates of the nodes. $d1, d2, d3$ the distances of nodes from the centroid.

If any of the three distances is less than $\alpha$ times the p.d.f then that point is rejected. If however, all the distances are greater than we can proceed to the second check using the beta criterion.

**Beta Criterion**

This is the second testing criterion for checking whether the prospective point should be accepted or rejected. It is used only after a particular centroid has already been passed the alpha criterion. It involves determining a set of values of $S_j$ where $j = 1$ to $N$.

These values represent the respective distances from the prospective point under consideration to each of the other prospective points to be inserted.

The criterion is checked by a single loop in which the value of $S_j$ is compared with that of the product of beta and the p.d.f at the prospective point. The logic used is:

- If $S_j$ is greater then $\beta d_{pm}$ - the point is accepted.
- If $S_j$ is smaller than $\beta d_{pm}$ - the point is rejected.
- If point is accepted, it is then included in the set of points of nodes data file and its distribution function is calculated in the next loop.
3.6 Mesh Smoothing

The quadrilateral elements produced by the generation procedure described above are not always well shaped, particularly in region where element size is varying sharply i.e. strongly graded mesh. Mesh smoothing and mesh modification techniques are employed to improve the quality of the quadrilateral elements.

In process of mesh smoothing, the connectivity of the elements and nodes are repositioned to produce the quadrilaterals with some what improved shapes. The technique repositions the nodes such that each internal node is at centroid of the polygon formed by its surrounding elements.

The new position of an internal node \( i \) is computed as:

\[
P(i) = \left( \frac{1}{4} \right) M \sum_{m=1}^{M} \left[ P_j + P_i + 2P_k \right]
\]

where \( M \) is the number of the elements around node \( i \). The mesh smoothing process consists of several loops of repositioning and each loop is made over the entire set of interior nodes. The technique has proved to be useful and effective and smoothes the mesh into one with better shaped elements.

3.7 Illustrative Problems

Using the algorithm described in section 3.3, the Flow-chart given in Fig. 3.5, the input domain (as shown in Fig 3.6(a)) is inputted. Initial triangulations using the boundary points as shown in Fig 3.6(b). Further the refinement of triangular mesh is performed by inserting more points without any user intervention as shown in Fig 3.6(c,d,e). Final optimal mesh is obtained as shown in Fig 3.6(f).

Figure 3.7 (a)-(f) shows the input domain, input triangulation of the given domain, triangulation of given domain after second run the next
triangulation of the domain, mesh optimization using circle check and the final optimal triangulated mesh respectively. Using the algorithm described in section 3.4, the input domain as shown in Fig. (3.8) is inputted for Quadrilateral mesh generation. Fig. (3.8 b) shows final quadrilateral mesh generated.

3.8 Conclusions

Automatic mesh generation software has been developed in this Chapter which discretize the given domain into either triangular elements or quadrilateral elements. The most elegant and general method for triangulation is Delaunay Method. Delaunay derived a simple procedure for triangulating an arbitrary set of points on a plane in such a way that the sum of minimum angle in each triangle would be maximized. Since finite element solutions are most accurate with equilateral triangle grids and since Delaunay triangulation procedure comes as close as possible to this, it is an excellent method to use with Finite Element method. Furthermore, it is possible to add, delete or modify a region configuration and grid without gross efforts. By using the Delaunay algorithm in all stages of the grid generation process, a triangular grid with optimal shape is produced. Procedure and algorithm for automatic points creation for generation of triangular mesh has been developed in Section 3.3. High quality meshes can be obtained using this procedure. However, for domains having very irregular boundaries it may be difficult to get the most reasonable distribution of points. In that case, the boundary domain is divided into a number of subregions. Each subregion is further processed independently to generate interior nodes. The distribution of the interior nodes depends on the boundary node density. This technique gives good quality meshes in the interior of the domain also and is flexible, easy and efficient to implement and requires minimal manual user input. It is capable of dealing efficiently with complex geometry and multiply connected regions generating optimal shaped finite element mesh models.

The output generated by the algorithm includes data files that contain information pertaining to nodal coordinates, element connectivities, and analysis.
specific data. In addition it includes options to create extended databases that reflect the original geometry to allow for remeshing and node repositioning during the course of a solution run.

A refined mesh is required in places where acute changes in geometry, boundary conditions, loading material properties, or solution occurs. However, the words coarse and fine are relative. In any given problem, one begins with a finite element mesh that is believed to be adequate (based on experience and engineering judgement). Then as a second choice, one selects a mesh that consists of a larger number of elements (and includes the first one as a subset) to solve the problem once again. If there is a significant difference between the two solutions, one sees the benefit of mesh refinement and further refinement may be warranted. If the difference is negligibly small, further mesh refinements are not necessary. In cases where computational costs is the prime concern, one must depend upon one’s judgment concerning what is a reasonably a good mesh. However, if one is concerned with the numerical accuracy of the solution, then a feel for the relative proportions and the directions of the errors introduced into the analysis helps to make a decision on when to stop further refining a mesh. Chapter 4 deals with discretization errors and adaptive refinement in detail.
Fig 3.1 ONE NODE CRITERION

Fig 3.2 TWO NODE CRITERION
FIG 3.3 THREE NODE CRITERION

FIG 3.4 MESH OPTIMIZATION

$D := \frac{H}{R}$
START
GET INPUT FROM DATA LINE
PERFORM INITIAL TRIANGULATION
CALCULATE POINT DISTRIBUTION VALUES FOR EACH BOUNDARY NODES AND EACH CENTROID
INITIALISE N = 0
PERFORM ALPHA-CHECK & BETA-CHECK TO SELECT CENTROID FOR FURTHER TRIANGULATION
INCREMENT N BY 1 FOR EACH SELECTED CENTROID
AGAIN PERFORM TRIANGULATION
PERFORM OPTIMISATION & CIRCLE CHECK
IS N = 0?
WRITE THE OUTPUT DATA TO FILE
STOP

FIG. 3.5 FLOW CHART FOR MESH GENERATION.
FIG 3.6 (a) INPUT DOMAIN FOR TRIANGULATION
FIG 3.6 (b) TRIANGULATION OF GIVEN DOMAIN AFTER FIRST RUN
G 3.6 (c) TRIANGULATION OF GIVEN DOMAIN AFTER SECOND RUN
FIG 3.6(d) TRIANGULATION OF GIVEN DOMAIN AFTER THIRD RUN
FIG 3.6 (c) TRIANGULATION OF GIVEN DOMAIN AFTER FOURTH RUN
FIG 3.6 (f)  FINAL TRIANGULATION OF GIVEN DOMAIN
FIG 3.7(a) INPUT DOMAIN FOR TRIANGULATION
FIG 3.7 (b) TRIANGULATION OF GIVEN DOMAIN AFTER FIRST RUN
FIG 3.7(c) TRIANGULATION OF GIVEN DOMAIN AFTER SECOND RUN
FIG 3.7 (d) TRIANGULATION OF GIVEN DOMAIN AFTER THIRD RUN
FIG 3.7(e) TRIANGULATION OF GIVEN DOMAIN AFTER FOURTH RUN
FIG 3.7 (f) FINAL TRIANGULATIONS OF GIVEN DOMAIN
FIG. 3.8 (a) INPUT DOMAIN FOR THE QUADRILATERAL MESH GENERATION
FIG. 3.8 (b)  FINAL QUADRILATERAL MESH FOR THE GIVEN DOMAIN