APPROACHES TO THE AUTOMATIC GENERATION AND CONTROL OF FINITE ELEMENT MESHES

Mark S. Shephard
Center for Interactive Computer Graphics
Rensselaer Polytechnic Institute
Troy, NY 12180-3590 USA

ABSTRACT

This review paper discusses the algorithmic approaches being taken to the development of finite element mesh generators capable of automatically discretizing general domains without the need for user intervention. The paper demonstrates that because of the modeling demands placed on a automatic mesh generator, all the approaches taken to date produce unstructured meshes. Consideration is also given to both a priori and a posteriori mesh control devices for automatic mesh generators as well as their integration with geometric modeling and adaptive analysis procedures.

INTRODUCTION

The generation of finite element models has historically been one of the drawbacks to the widespread use of the analysis technique. Over the past fifteen years, code developers have addressed this deficiency by producing stand alone finite element preprocessing systems for the generation of finite element models. These systems typically employ a number of mesh generation techniques in an interactive graphic framework that allows the user to define the domain and mesh for the problem at hand. During that same period of time, other developers were constructing interactive graphics-based geometric modeling systems. The early versions of these systems simply computerized the standard drafting processes and were used almost exclusively for making engineering drawings for the shop floor. It was quickly realized that there is a large potential for directly employing the information available in a geometric modeling system for a variety of

applications such as machining and engineering analysis. However, the early systems that simply computerized the drafting process did not contain all the geometric information needed to allow applications to operate automatically. Therefore, the more recent geometric modeling systems, commonly referred to as solid modelers [1-3], employ complete and unique geometric representations. These systems contain all the geometric information needed to allow any geometrically controlled operation to be automated.

Since the generation of a finite element mesh is a geometrically controlled process, it is possible to automate the mesh generation process when the geometry of the object is defined in a solid modeling system. There are three reasons why such capabilities are not yet commonly available. The first is the lack of mesh generators capable of discretizing general domains without the need for extensive user interaction to partition the domain into meshable regions. The second is the lack of the geometric modeling support capabilities needed by automatic mesh generators to interrogate and, for some algorithms, to modify the geometric representation of the solid. These modeling capabilities typically exist within the modeling system itself, but are not available in a form that they can be easily separated from the modeler and used by an applications procedure such as a mesh generator. The third reason is the inability of finite element analysis programs to automatically modify the finite element discretization so that the analysis results yield a prescribed level of accuracy. This necessitates the need for current users to specify mesh control information to yield the type of element distribution that, based on their knowledge and experience, should yield the desired accuracy.

The purpose of this paper is to discuss the progress that has been made in addressing these three needs. The majority of the paper is devoted to the algorithmic approaches to automatic mesh generation that are currently under development, and the techniques available to control the distributions of elements throughout the domain of the object. As discussed in the third section,the integration with

geometric modeling systems is much more than the simple passing of geometric information, it also includes the geometric modeling functionality needed for the automatic mesh generators to operate. Consideration is also given to the use of these procedures in adaptive finite element analysis. Adaptive analysis procedures promise to provide the analysis functionality needed to assess and control finite element discretizations to provide the level of accuracy prescribed.

ALGORITHMIC APPROACHES TO AUTOMATIC MESH GENERATION

In recent years, the generation of finite element meshes has been dominated by the application of mapped mesh generators that produce what are commonly referred to as structured meshes. These mesh generators [4-7] have the advantage of being able to produce well controlled meshes within the individual 'patches' passed to the mesh generator. They have the disadvantage of requiring the domain to be meshed be partitioned into a set of mappable regions which will yield the type of mesh control desired. Since the majority of finite element models constructed in the past were produced independently of any computerized geometric model, it was convenient to define the object in a bottom-up fashion in terms of mappable mesh patches. However, the complexity of reducing the complex three-dimensional domains available from geometric modeling systems into a set of mappable regions has lead to an increased interest in the development of mesh generators capable of automatically meshing the entire domain. For the purpose of this discussion, an automatic mesh generator is an algorithmic procedure capable of producing a valid finite element mesh in a domain of arbitrary complexity given no input past the computerized geometric representation of the domain to be meshed.

Before discussing the specific algorithmic approaches to automatic mesh generation, it is important to emphasize the fundamental operational difference between mapped meshing procedures and the automatic mesh generation techniques that have been considered to date. When mapped mesh generators are used, the geometry of the object is constructed by gluing together the individual, fixed topology,

mesh patches. Therefore, the geometric representation is explicitly defined in terms of those mesh patches. The mapping operators used to define the mesh within each of the mesh patches employ, in either an explicit or implicit form, a set geometric representation for each mesh patch defined in terms of the information available on the boundary of the mesh patch. The user is responsible for defining a valid set of mesh patches, which implicitly define the geometric representation and explicitly provide the geometry necessary for meshing to occur. The mesh generators are, therefore, not concerned with the actual geometry of the object. This is, however, not the case for an automatic mesh generator which is given a complete geometric representation of the domain of interest and is responsible for decomposing, without a priori information of the shape of the domain, it into a valid set of elements. Since an automatic mesh generator must determine the limits of the domain to be meshed, the most computationally intensive portion of these procedures are the carrying out of geometric interrogations for this purpose. Since mapped mesh generators need not carry out these interrogations, it is not surprising to find they are much more computationally efficient, however, at the expense of user productivity.

Another important difference between these two approaches is that all of the current automatic mesh generators produce unstructured meshes and are best suited to producing simplex element topologies. This means triangular elements in two dimensions and tetrahedral elements in three dimensions. Although a number of algorithm developers have successfully implemented two-dimensional algorithms to produce acceptable quadrilateral meshes, it is not likely that procedures to create acceptable all hexahedronal meshes for general three-dimensional domains will be easy to produce. (There is a simple subdivision procedure to convert a tetrahedral mesh into an all hexahedral mesh [8], but the shape of the elements tend not to be satisfactory.) Although some effort is under way to develop all hexahedral meshes automatically, there are good reasons to assume they are not going to be overly successful. It is because hexahedral elements are reasonably sensitive to element shape and any automatic

mesh generator producing them is unlikely to be able to control the shape adequately. The other possibility is to generate a mesh with a mixture of element types with as many hexahedronal elements as possible. However, the need to match the faces of elements to insure inter-element continuity means that a number of element shapes would have to be used including a pyramid element and that the percentage of hexahedron that would be produced in general geometries may not be high.

For some classes of problems analyzed by the finite element method, the use of various polynomial order tetrahedron is considered quite acceptable. However, in other problem classes, particularly stress analysis, users have a strong bias against these elements. The major reason for this concern is that the majority of tetrahedral elements in analysis packages were linear displacement, and thus constant stress, elements which are well known to perform poorly in these classes of problems. Recently, due primarily to the push for the availability of automatic mesh generators, code developers have been adding higher order tetrahedron elements to their element libraries. Although not yet heavily tested, initial experience indicates that the use of second order tetrahedron elements in conjunction with automatic mesh generators will provide a cost effective means of performing stress analyses of general geometries. Additional development of tetrahedronal element types will be needed to fully address the use of these elements for other analysis classes. For example, the use of displacement-based tetrahedral elements for incompressible problems leads to the application of too many constraint equations often yielding a severely over constrained system of equations.

The automatic mesh generating procedures considered in this section are fully three-dimensional or the extension from the existing two-dimensional procedure to a three-dimensional procedure appear possible. Therefore, no attempt is made to provide a complete bibliography of papers on automatic mesh generation, most of which are two-dimensional. Instead effort is concentrated on those papers that consider three-dimensional techniques, making reference to selective

early papers that are relevant. For purposes of this discussion, the algorithms that have been developed will be classified as being based on one of the following algorithmic approaches;

1. point placement followed by triangulation,
2. removal of individual subdomains,
3. recursive subdivision of the domain, and
4. spatial decomposition followed by subdomain meshing.

Although specific automatic meshing algorithms may overlap two of the approaches listed, or may be implemented in specific steps where separate steps use different approaches to carry out the appropriate operations, the above classification provides a reasonably fundamental separation of algorithmic approaches.

Point Placement Followed by Domain Triangulation

In this approach, the generation of the element mesh is carried out in two distinct steps. The first step is to place points throughout the domain of interest in a manner such that during the second step, the triangulation of the points into an element mesh, the desired mesh gradations and representation of the domain is obtained. As done in the early survey on mesh generation [9], any mesh generation process can be viewed as carrying out these two steps. However, this subsection is only concerned with algorithmic approaches that contain them as two distinct operational steps.

The first attempts to develop mesh generation procedures using these approaches concentrated on the automation of the second step on two-dimension domains [10]. Even in today's three dimensional procedures [11], this is the better understood of the two steps. The early two-dimensional procedures [10,12] employed ad-hoc rules to determine how to connect points together to create triangular elements. A properly constructed set of rules is capable of producing a well controlled mesh within a set of points, but the majority of the early procedures required extensive searching and a large number of

checks, many more than needed in an optimal triangulation algorithm. In addition, it was difficult to develop a set of triangulation rules that would insure the elements generated satisfy a given shape criteria. This would indicate that the extension to three-dimensions could be difficult and likely to be computationally intensive. One three-dimensional rule-based procedure [13], which is an extension of the point surrounding concept presented in [10], has been developed. In this approach, the concept of surrounding a given point with triangular elements is replaced with surrounding a line between two points with elements and then to move on to another line until the mesh is complete. Given a line connecting two points this procedure will find a near-by point to form a triangular plane. This triangular plane serves as a face of a tetrahedron of the first element which is defined by another near-by point selected to complete it. One of the two triangular faces of the tetrahedron that use that edge is selected as the base triangle for the next tetrahedron. This process is continued until the line is surrounded at which time a new line is selected for surrounding.

Most of the recent effort in the development of procedures to produce elements given a set of points employ the properties of the geometric constructs of Dirichlet tessellation and, more importantly for mesh generation, the dual Delaunay triangulation of a given set of node points. Cavendish, et al. [11] gives an interesting account of the history of these procedures in the mathematics literature and their more recent use for the purposes of finite element mesh generation. The basic property of a Delaunay triangulation in two dimensions that makes it appropriate for use in mesh generation is the resulting set of triangles is as close to equilateral as possible [14]. More specifically, the basic property of a Delaunay triangulation is that there are no points inside the circum-circle defined by the three corners of the triangles in two dimensions and no points inside the circum-sphere defined by the four corners of the tetrahedron in three dimensions. This distinction is of critical importance since this property does correspond to well shaped, as compared to an equilateral triangle, elements in two dimensions, but does not insure well shaped

elements in three dimensions, as compared to an equilateral tetrahedron. As indicated below, this does have an important impact on the development of a Delaunay based three-dimensional mesh generator.

There are a number of algorithmic approaches to the construction of a Delaunay triangulation. A currently popular approach is a version of an algorithm proposed by Watson [15] based on the property that in a Delaunay triangulation there are no node points on the interior of the circle defined by the three nodes of any of the triangles. The mesh generation algorithm of Cavendish, et al. [11] uses this property directly by constructing the mesh by a node insertion procedure. Given a Delaunay triangulation for a subset of the total set of nodes, one of the remaining nodes is considered. The circum-circles of the existing triangles are tested to see which contain the new node. These triangles are flagged for deleting from the mesh (Fig. 1a) which creates a unfilled polygon with a single internal node. It can be shown that the Delaunay triangulation including the new node is simply constructed by connecting all the vertices of the unfilled polygon to the new node (Fig.1b) . This process is continued until the mesh is complete.

It is important to note that the triangulation produced by a Delaunay process represents the convex hull of the points used. This means specific consideration must be given when the domain to be meshed in not convex. This concern is easily addressed by rejecting elements that are not within the domain of interest if the original set of nodes are placed such that no element edges or faces are generated that pierce the boundary of the domain. It is possible to do this by the proper placement of points exterior to the domain when starting the triangulation process [11].

The development of algorithmic procedures for the placement of points such that the desired mesh gradations are created, and poorly shaped elements are not created because of poor point placement, is an important part of using a Delaunay procedure for finite element mesh generation. Cavendish [12] has presented a good two-dimensional scheme

that spreads points based on node point density factors which are specified in user defined regions. Another scheme for point placement based on the primitives in constructive solid modeling has been presented by Lee, et al. [16]. In this algorithm, the points are uniformly placed in each of the two-dimensional primitives used in the definition of the object. Since the shape of a primitive is well understood, this is a simple task. After the primitives are combined through the Boolean operations, a procedure to selectively eliminate selected points in the portions of the domain that overlap is applied to insure the creation of a mesh of the desired mesh density. Recently Lo [17] proposed the use of a simple ray firing technique in which points are placed along the rays when the ray is interior to the object and places nodes at the points where the rays enter and exit the domain. It is important to note that whatever technique is used to place points, it should properly consider the boundary of the domain, placing points so that the resulting finite element model properly represents the domain of the object.

Although the basic concept of Delaunay triangulation is directly extendible to three, and higher dimensional domains, its use for automatic three-dimensional mesh generation requires special consideration. This is because there is no guarantee that the resulting elements will have a satisfactory shape in terms of the ratio of volume to surface area. In fact it is possible to create zero volume tetrahedron [11,18,19] within a three-dimensional Delaunay triangulation. Dealing with the unacceptable element shapes, referred to as slivers [11,18,19], requires special considerations, taking a three-dimensional automatic meshing algorithm past that of basic Delaunay procedure. As an example of a Delaunay-based three-dimensional mesh generator that has considered these factors, a brief summary of the one such procedure [18,19] is:

1. Define a bounding box for the domain of interest and fill it with regular icosahedron following a specific procedure [18,19].
2. Discard all points belonging to that set of icosahedron

that fall outside the object to be meshed. The remaining set of points are referred to as the preliminary nodes.

3. Use Watson's algorithm to construct a Delaunay triangulation of the preliminary nodes. Since the triangulation defines the convex hull of the points, discard all tetrahedron whose centroid is outside the domain of the object.
4. Eliminate the nodes, and associated tetrahedron, that are used to define any of the element face triangles that lie on the exterior of the triangulation produced in step three. (The exterior triangles are those that are used by only one element.)
5. Generate a set of nodes on the boundary of the original object. This includes nodes at model vertices, along model edges and on model faces.
6. Using Watson's algorithm, insert these nodes into the Delaunay triangulation. Again discard any tetrahedron whose centroid falls outside the domain of the object.
7. Calculate the shape measure for all elements within the triangulation. A good measure is the ratio of the radius of the inscribed sphere to circumscribed sphere, normalized to the ratio of a regular tetrahedron [19].
8. Collapse out the unacceptable surface tetrahedron, slivers, that can be eliminated.
9. Apply the sliver removal procedures described in [19] to eliminate all remaining sliver elements.

Mesh Generation Based on Sub-Domain Removal

Automatic mesh generation procedures in this group operate by removing individual pieces from the domain one at a time until the domain is reduced to one remaining acceptable piece. The majority of algorithms based on this approach remove individual elements one at a time [20-24] while others remove larger, but 'simple' portions of the domain and then triangulate these individual pieces using a different procedure [25-27].

Sub-domain removal meshing procedures typically employ a boundary representation of the domain and operate by searching for entities of specific topological type that satisfy a set of connectivity and geometric requirements. One of the set of entities that satisfy the given requirements is used as the base entity for a geometric removal operation that carves off a portion of the domain. The process of looking for and removing a new piece is then again applied to the domain remaining until it is reduced to a single acceptable piece. Mesh generators based on this approach often employ a number of operators, applied in a hierarchic manner, and attempt to consider the influence of a current choice on future removal operation selections.

As an example, consider the two basic element removal operators used by Woo and Thomasa [21] to mesh three-dimensional domains without voids. (A third operator is used if voids are present.) The first operator, VERTEX_REMOVAL is applied by searching the object for vertices with only three edges coming into it. Any such vertex that satisfies a set of geometric interference requirements is then removed from the object. The removal of a vertex carves a tetrahedron from the object (Fig. 2a). In cases where all vertices have more that three vertices, a second operator, EDGE_REMOVAL, is applied. In this case, a tetrahedron containing the selected edge is carved from the object (Fig. 2b). Since this operation reduces the number of edges connected to two of the vertices by one each, it eventually reduces the complexity of the object until the first operator can be applied again.

A topologically-based element by element removal procedure appears ideally suited for the construction of optimal h-p finite element meshes where coarse, exponentially graded meshes are desired [28,29]. A procedure under development for the generation of such meshes [24] employs four meshing operators to produce meshes in simply connected two-dimensional domains (Fig. 3). The first operator, SINGULARITY_REMOVAL, is used to isolate the locations of all possible singularities so that the proper set of elements can be placed around

the singularity. The remaining operators, VERTEX_REMOVAL, VERTEX_REMOVAL_WITH_EDGE_SPLIT, and EDGE_REMOVAL, are used to mesh the rest of the domain.

Since the amount of computation required for the application of each removal operation is high, these procedures are not computationally efficient for the creation of a fine mesh. However, the use of such procedures to remove large pieces of the object which, can then be quickly filled with with elements, can provide a computationally efficient method to produce meshes to any level of fineness. An example of such an approach is the algorithm of Joe and Simpson [26] which first reduces a two-dimensional domain into simply connected regions, and then reduces these to convex polygons. An optimal quasi-uniform triangulation of each convex region can then be quickly constructed.

The development of an algorithm that decomposes the domain into large chunks by removing them one at a time is an attractive way to consider the automation of the current methods of mesh generation where the user interactively decomposes the domain of interest into mappable regions and invokes a mapped mesh generator. The difficulty in developing such an approach is the identification and implementation of a set of rules that would examine a geometry to determine how to decompose it into mappable regions that will yield the type of mesh gradations desired as well as providing a satisfactory mesh topology. An example of such an approach for two-dimensional geometries is shown in figure 4. This procedure (an unpublished prototype program by the author) first invokes a set of 'rules' to identify the regions the should, based on the mesh control information and the geometry, be removed as a mappable region. It then applies another set of rules to decompose the remaining domain into acceptable shaped regions to be filled by a mapped mesh generator. (Only the second set of rules were used on the example in figure 4.) The main complexity in the development of such an approach is the development of a set of rules that can 'look' at the computerized representation of the entire geometry and decompose it in a manner simular to that a human produces

when they look at the geometry on a screen. It is interesting to note that in the development of the program used to generate the simple example shown in figure 4, several finite modeling experts were given example geometries and asked to define, without actually meshing them, the mesh regions they would define to mesh a given set of geometries. In most cases, they laid out substantially different regions. An attempt is currently under way [27] to develop a three-dimensional procedure taking a similar approach. The work is using the concepts of primitive identification and feature recognition as applied to geometric modeling based on constructive solid geometry (CSG) [1,2].

Mesh Generation by Recursive Subdivision

The recursive subdivision mesh generators [30,31] operate by the repeated splitting of a domain into simplier parts until the individual parts are single elements, or, possibly, simple regions in which elements can be quickly generated. As in the sub-domain removal procedures, this class of mesh generator typically operates off a boundary representation of the domain to be meshed, looking for candidate topological features meeting specific connectivity and geometric requirements, selecting a specific splitting operation, and updating the geometric and topological representations of the two sub-domains created by the split.

A simplified description in the steps involved in the generation of a three-dimensional finite element mesh by such an approach [30] is:

1. Reduce all the faces of the object to simply connected faces by the introduction of splitting curves from interior loops to the exterior loop. (Interior loops can connect to other interior loops so long as one in the chain of connected interior loops is then connected to the exterior loop.)
2. Place node points along the various edges in the model in a manner to reflect the mesh gradations desired. Topologically this operation is equivalent to introducing

vertices at various locations along edges and splitting the edges into multiple edges at those vertices.

3. Triangulate each of the surface patches into a set of surface triangles employing the nodes introduced in step 2. The surface triangulation is carried out by the recursive splitting of the face as follows;
   * a split line is introduced between two nodes on the boundary of the face that validly splits the face into two,
   * nodes are introduced along this split line based on the nodal spacing of the edges that it runs between,
   * the splitting of all sub-faces is continued until they are all reduced to individual triangles.
4. Using the element edges introduced on the faces, determine a splitting face that splits the object into two sub-objects.
5. Mesh the splitting face using Step 3.
6. Repeat Steps 4 and 5 until each of the remaining subdomains represents a single element.

Spatial Decomposition Followed by Subdomain Meshing

The basic idea behind these approaches is to use an efficient procedure to decompose, in a controlled manner, the domain of interest into a set of simple cells and to then mesh the individual cells in such a manner that the resulting mesh is valid. The one spatial decomposition approach that has been applied to mesh generation is the quadtree in two-dimensions [32-35] and the octree in three-dimensions [24,36-38]. In an octree representation, an object is represented as the union if a set of disjoint cubes of various size which are derived from the recursive subdivision of parent cubes into eight octants. The entire structure is stored in a hierarchic tree [39,40]. Since the size of octree cubes desired for use in finite element mesh generation are large with respect to the geometric details of the object, it is necessary to deal in a specific manner with those octree cubes that contain the boundary of the object and are neither fully

inside nor outside the object.

One approach to building a three-dimensional mesh generator using this basic tree representation is the finite octree, formerly modified-octree, technique [24,36-38]. (The paper by Baehmann, et al. [34], although limited to the two-dimensional finite quadtree, formerly modified-quadtree, gives the most complete description of the approach outlined below.) Since the proper representation of the topological features that define the boundary of the object is necessary to insure the validity of the mesh, the finite octree is defined by the insertion of topological entities hierarchically from the bottom. The vertices are first inserted into the tree being placed in the proper sized octants. Next the edges are inserted, in discrete form, into the proper sized octants. Edge insertion is carried out by traversing the edge starting from its first vertex, which already exist in its appropriate sized octant. The intersection where the edge leaves that octant is found and associated with that discrete segment as well as a pointer back to the original edge it came from. The intersection location where it exited the first octant is the starting point of the discrete segment of the second octant, the size of which is controlled by the mesh control information applied to the edge. The intersection where it exits that octant is found and the segment stored. This process is continued until the edge's second vertex is found. The faces of the object are then inserted in discrete form using the existing edge information and the intersections of the sides of octants with the surface patches making up the face. The definition of the octants containing the boundary of the object, referred to as cut octants, is completed by qualifying which side of the discrete boundary existing in the octant is inside the object. This operation requires a specific set of geometric checks. The interior octants within the finite octree are then quickly filled by a simple tree traversal process.

The finite element mesh is then generated within each of the octants using the tree to pass octant face mesh information required to insure a compatible mesh. The tetrahedronization scheme used for interior

octants need only deal with a shape that is topologically a cube with nodes at the corner and the possibility of mid-side and mid-face nodes if the neighboring octants are one level finer as is allowed in the mesh generator. The tetrahedronization of the boundary octants is more complex in that it employs the above information plus the discrete boundary information and specific geometric interrogations of the original description of those entities when needed. A nodal repositioning procedure to improve the shapes of the elements can also be invoked. Figure 5 shows an example mesh generated with this procedure.

Speed of Automatic Mesh Generators

The limited experience available to date indicates that the amount of computation needed to generate a mesh of a few thousand elements for a general three-dimensional geometry will be of the same order of magnitude as a linear analysis carried out on that system. Therefore, the computational efficiency of these procedures is of critical importance. The two measures of computational efficiency of importance are the time required by the given algorithms to generate comparable meshes and, even more importantly, the computational growth rate of the mesh generator. Tests run to date on complex two-dimensional geometries indicates that the implementation of various approaches yields speed differences that vary by more that an order of magnitude. (The test referred to are proprietary to the company that ran the test and can not be presented here.)

The various algorithmic approaches also demonstrate different growth rates. The approach with the greatest amount of theoretical results is Delaunay triangulation which, in the two-dimensional case [41], indicate an O(n log(log n)), where n is the number of points, computational time as being possible. (In two-dimensions the number of elements is of the same order as the number of nodes [42].) Computational results of an implemented three-dimensional algorithm gave O(n**5/3) computer times [11]. (In the three-dimensional case, the number of elements can be from O(n) to O(n**2) [42]. However, it

appears that in most practical cases the number of elements will be O(n).)

The best computational growth rate obtained thus far is linear, O(n), [26,34]. Joe and Simpson carried out a detailed study of the computational effort required for their two-dimensional algorithm and demonstrated times that were linear and asymptotic with one of the steps of the algorithm. The finite quadtree two-dimensional [34] and finite octree three-dimensional mesh generators also demonstrates a linear growth rate with the number of elements.

A Priori Control of Element Distributions

In addition to the ability to generate a valid mesh for any geometry, automatic mesh generators must permit the types of mesh gradations necessary to produce efficient finite element models. Ideally, the mesh control devices available allow for the convenient specification of both a priori and a posteriori mesh control information. A priori mesh control devices are used to specify the distribution of elements in the initial finite element model, while a posteriori mesh control devices are used during an adaptive analysis process [43] to improve the mesh as dictated by the results on the current mesh.

The devices available to control the distribution of elements throughout the domain of an object is at least partly a function of the mesh generation algorithm used. The ease with which particular forms of mesh control can be exercised is a function of both the mesh generation algorithm and its implementation.

Since the basic input to an automatic mesh generator is a geometric representation, any a priori mesh control device must be tied to the geometric representation. This means that a priori mesh control can also be a function of the particular geometric modeling approach used. For example, mesh control information could be tied to the individual primitives used is a constructive solid geometry modeling system and thus stored as attribute information tied to that primitive in the

binary tree used to store the primitives and Boolean operations carried out on them [1]. Although this may be a natural approach for use with constructive solid geometries and mesh generators designed to operate with such modelers [16], it is not general, and it most likely does not provide the type of mesh control that users of a priori mesh control devices would expect. A more general method to define mesh control information is to tie this information to the model through the topological entities in the boundary representation of the object. This method has the advantage of allowing for convenient specification of mesh gradations by assigning mesh control information to the individual vertices, edges, faces and regions that make up the domain to be meshed in such a manner that any type of mesh gradations that are desired and can be handled by the mesh generator will be produced. It is also a reasonably general approach since an object has a unique boundary representation which can be produced from any of the evaluated solid geometric modeling approaches [1,2,44,45]. In addition, most of the geometric modeling systems provide the ability to produce the boundary representation of the object no matter which solid modeling approach is used.

Automatic mesh generators that operate by removal of individual subdomains [20-26] and recursive subdivision [30,31] rely on boundary information and are well suited to employ mesh control information tied to the edges of the boundary. They are typically less suited for mesh control information defined in terms of the faces and regions that make up the domain of the object. However, it is possible with the appropriate implementation considerations to reflect that type of mesh control information in the mesh generation process.

The mesh control devices for automatic mesh generators that triangulate a set of points in space [10-14,16] are used to control the distribution of points in space. This has the advantage that any spatially-based procedure to place points in space can be used to control their distribution. The disadvantage is that, as indicated in the previous section, good procedures to define points throughout general three-dimensional domains are difficult to devise. It would be

desirable to construct procedures that are able to do this by specifying mesh control information to the various boundary entities of the object.

Mesh generators based on spatial decomposition also have the advantage of easily reflecting spatially-based mesh control so long as this information can be defined in such a manner that the decomposition can properly be reflected. The ease with which this can be carried out is a strong function of particular decomposition algorithm and its implementation. Since the finite quadtree [33,34] and finite octree [36,37] operate by inserting the boundary entities of the object into the tree following the hierarchy of topological entities, they are well suited for the specification of boundary-based a priori mesh control information [38]. Figure 6a shows a uniform finite quadtree mesh for an object when all the mesh control parameters for the vertices, edges and regions are the same, while Fig. 6b shows a mesh for the same object by simply changing the values of the mesh control parameters for some of the vertices and edges (Fig. 6c). Figure 7 shows two finite octree meshes for the same object with the only difference in mesh control parameters being the values along one edge.

INTEGRATION OF AUTOMATIC MESH GENERATORS WITH GEOMETRIC MODELERS

As indicated in the previous section, automatic mesh generators are geometrically very demanding. In particular, they require a large number of geometric interrogations; and, depending on the meshing algorithm, a large number of geometric model modifications to operate. Therefore, they are not well suited to a static interface with geometric modeling systems in which all that is available to the mesh generator from the geometric modeling system is an output file of the geometric representation [46]. Assuming that a common format is used for this file, this approach has the disadvantage of requiring all the geometric modeling functionality needed by the mesh generator be reproduced within the mesh generator. Assuming that this functionality already exist within the geometric modeling system, which is typically the case, the development of that capability in the mesh generator is

a redundant effort that has to be repeated for each new geometry form to which the mesh generator is interfaced.

An alternative approach is to employ a dynamic interface in which the mesh generation algorithms can interact directly with a geometric modeling system through a set of procedures, to be referred to as geometric communication operators, that can perform specific geometric interrogations and modifications. The definition of geometric communication operators is being considered for geometrically-based applications [47], as well as those needed specifically for mesh generation [48]. One approach to effectively employing geometric communication operators in a finite element modeling system is to have the input information used directly by the finite element modeling software be the topological description of the object. Topology represents an abstraction that is independent of the specifics of the geometric definition, but does contain the connectivity information necessary to control finite element modeling software which operates through a set of geometric communication operators. One topological representation well suited to this application is Weiler's non-manifold radial edge data structure [45]. A high level design of such a system is contained in [49].

The discussion below assumes a dynamic interface between the automatic mesh generators and the geometric modeling system. See reference [48] for a more specific discussion of the geometric communication operators needed to support the various automatic mesh generation approaches.

The integration of an automatic mesh generator with a geometric modeling system requires a substantially different set of geometric communication operators than is needed for interactive finite element model generation. The complexity of the interface of an automatic mesh generator with a solid modeler is a function of the algorithmic approach underlying the mesh generator. Mesh generation algorithms that operate through geometric interrogation only require a simpler set of geometric communication operators than needed by mesh

generators that must both interrogate and modify the geometric representation during the mesh generation process. In general, the majority of computational effort required for automatic mesh generation is spent in carrying out geometric communication operations. Since geometric interrogations typically require much less computation than geometric modifications, mesh generators requiring geometric interrogation are typically more efficient, on a per element basis.

Two of the four algorithmic approaches to automatic mesh generation discussed above require geometric interrogation only. They are point placement followed by triangulation, and spatial decomposition followed by subdomain meshing. The other two, removal of individual subdomains and recursive subdivision, require both geometric interrogation and modification. To better see this differentiation, consider the comparison of the interactions with a geometric representation for both an element-by-element removal algorithm and the finite octree approach. In the element-by-element removal process, topological and geometric interrogations are used to look for a candidate feature to be carved off; geometric interrogations are used to see if that removal is valid; and finally the feature is removed. Since the next element removal must consider the geometry as it stands after the current element was removed, the geometric model must be updated by the use of geometric modification operators to reflect this removal. In contrast, the primary geometry-related task in the finite octree mesh generator is to determine how the boundary of the object interacts with the appropriate sized octants in the tree. This information is obtained through geometric interrogation only by intersecting the boundary entities of the object with the appropriate boundary features of the octants. The only other geometric communication operators needed for this and the rest of the meshing process are the interrogation operators of point classification, the conversion from parametric and real coordinates, and the conversion from real to parametric coordinates.

Although the algorithmic approaches to automatic mesh generation and

the geometric modeling procedures are available, the sets of geometric communication operators needed to properly integrate them are not readily available. Since the vast majority of these operators represent operations that the geometric modeler must already support, there is no major technical hurdles to be overcome to provide this functionality for finite element modeling.

ADAPTIVE ANALYSIS AND A POSTERIORI MESH CONTROL FOR AUTOMATIC MESH GENERATORS

As the finite element technique becomes more heavily used by designers who do not possess extensive expertise in numerical analysis, there is not only a need to improve the speed and robustness of the model generation procedures, but a need to insure that the analysis results produced are of sufficient accuracy to be meaningful. As in the case of the model generation process, increasing the robustness of the analysis to produce a prespecified degree of accuracy is best obtained through the development of automated procedures for that purpose. This is the goal of efforts on the development of adaptive finite element analysis procedures.

In an adaptive finite element analysis procedure, the solution results on a given mesh, in combination with a knowledge of that mesh, are used to both estimate the accuracy of that solution as well as how to best improve the mesh to efficiently obtain the level of accuracy desired. The major components of such a system include;

1. finite element equation formulation and evaluation algorithms,
2. a posteriori error estimation techniques to estimate the discretization errors in the current solution,
3. error indication, or alternatively, correction indication to determine where and, in the ideal case, how to improve the finite element discretization, and
4. mesh improvement schemes to improve the finite element discretization as indicated by the error or correction

indicators.

Since adaptive finite element analysis employs a feedback procedure which requires a number of solutions to sets of related finite element equations, the techniques used for each of the component portions of the system must be able to operate in an efficient manner. In addition to being able to efficiently solve related sets of finite element equations, the development of these systems must consider the most appropriate mesh generation and update procedures to be used with the various adaptive analysis approaches. Since this paper is primarily concerned with the automatic generation and control of finite element meshes, this section is concerned with the use of various automatic mesh generators and mesh update procedures appropriate for use with them. It first introduces some of the basic concepts and terminology of a posteriori error estimation to place the remainder of the section into context. The reader interested in more detail on error estimation, as well as the efficient solution of the evolving sets of algebraic equations arising in such systems, should begin by consulting [43] and the appropriate references sighted in the remainder of this section.

Overview of A Posteriori Error Estimation

A critical aspect of an adaptive analysis process is the estimation of the discretization errors present in a given solution as well as determination of how to most efficiently improve the finite element model to obtain the level of accuracy desired. Since a priori finite element error estimates can only indicate the convergence rate [50], useful error estimates must employ a posteriori techniques which use the analysis results to estimate the overall discretization error in one or more solution norms. The concepts and techniques used to calculate a posteriori error estimates and to determine how to most efficiently improve a finite element discretization have begun to mature since the early pioneering works of Babuska and his co-workers (see [51-53] for example).

Investigators in the area of adaptive finite element techniques [43,54] agree that the primary function of a useful a posteriori error estimator, E, is to provide a convergent and accurate measure of the discretization error, e, of a given finite element solution. The commonly used measure of the accuracy of an error estimator is the effectivity index, Θ, which is defined for the jth mesh in a convergent sequence of meshes, K, as:

$$\Theta_{(k_j)} = \frac{||E_i||}{||u-u_j||} \qquad (1)$$

where u is the exact solution and u_j is the finite element solution on mesh j. One required property of a useful a posteriori error estimator is

$$|\Theta_{(k_j)} - 1.0| \rightarrow 0 \text{ as } j \rightarrow \infty \qquad (2)$$

The practical measurement of the usefulness of an a posteriori estimator is to apply it to a set of problems with known solutions (either analytic or very accurate numerical solution) and to calculate the effectivity indices for a sequence of adaptively refined meshes.

In addition to the necessary requirement that the effectivity index for an a posteriori error estimator be close to one, there are two additional desirable properties. The first is that the computations of the error estimate, E, be an accurate approximate to the true error, e, on as local a basis as pointwise evaluations. This allows the estimate to be used to measure errors in any of a number of norms as opposed to only integrated norms. The second property is that the estimates, both local and global, be inexpensive to evaluate relative to the effort required to calculate the finite element solution. These two properties tend to work against each other. Estimates that are computationally efficient, with a computational cost on the order of n, where n is the number of unknowns in the finite element model, are often accurate only for specific global norms defined in terms of integrals over domain. On the other hand expensive estimates that require the same order of computation as the original solution

(typically $O(n^{\alpha})$, $1.5 \leq \alpha \leq 2$) are more likely to give useful estimates for any norm.

To demonstrate some of the basic concepts of error estimation consider a model elliptic [55] problem defined in two dimensions as

$$-\nabla a \nabla u + bu \equiv -\frac{\partial}{\partial x}\left(a \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(a \frac{\partial u}{\partial y}\right) + bu = f(x,y), \quad (x,y) \in \Omega \tag{3a}$$

subject to

$$u(x,y) = 0 \qquad (x,y) \in \partial\Omega_1 \tag{3b}$$

$$\frac{\partial u}{\partial \eta} = q \qquad (x,y) \in \partial\Omega_2 \tag{3c}$$

$$\partial\Omega = \partial\Omega_1 \cup \partial\Omega_2$$

where

Ω is a bounded region in R^2
$\partial\Omega$ is the boundary of Ω
η is the unit outward normal to $\partial\Omega$
$a(x,y)$, $b(x,y)$ and $f(x,y)$ are given functions meeting the necessary smoothness criteria subject to $a(x,y)>0$ and $b(x,y)\geq 0$ $\forall (x,y)\in\Omega$

The weak form of solution to this problem is to find $u \in H^1$ such that

$$A(u,v) = (f,v) + \langle q,v\rangle_{\partial\Omega_2} \quad \forall\, v \in H^1_o \tag{4a}$$

where

$$A(u,v) = \int_\Omega [a\nabla u.\nabla v + buv]\, d\Omega \tag{4b}$$

$$(f,v) = \int_\Omega fv\, d\Omega \tag{4c}$$

$$\langle q,v\rangle_{\partial\Omega_2} = \int_{\partial\Omega_2} qv\, ds \tag{4d}$$

and H^1_o is the set of all functions contained in H^1 which are zero on $\partial\Omega_1$. Recall that the space H^1 contains all functions for which the

function and its first partials are square integrable over the domain.

A finite element approximation, $U \varepsilon S_t \subset H^1$ to u is obtained by solving

$$A(U,V) = (f,V) + \langle q,V \rangle_{\partial \Omega_2} \quad \forall\ V \varepsilon S_t \qquad (5)$$

where the basis function selected for U and V are piecewise polynominals defined over individual elements of the triangulation of the domain Ω such that C^o inter-element continuity [50,56] is maintained. This allows the integrals in equation (5) to be carried out over the individual finite elements, Ω_i, and then properly summed.

After the system is solved for U, the goal to obtain a useful approximation, E, to the actual error, $e = u - U$, measured in an approximate norm. The most direct means to do this is to replace u by $U + e$ in equation (4) yielding

$$A(e,v) = (f,v) + \langle q,v \rangle_{\partial \Omega 2} - A(U,v) \quad \forall\ v \varepsilon H_o^1 \qquad (6)$$

and to replace e and v by piecewise polynomial basis functions, $E \varepsilon S_t^* \subset H^1$, to yield the error estimate

$$A(E,V^*) = (f,V^*) + \langle q,V^* \rangle_{\partial \Omega_2} - A(u,V^*) \quad \forall\ V^* \varepsilon S_t^* \qquad (7)$$

It is important to note that the space spanned by the basis function S_t^* can not be just any set within H^1. For example, assume that $S_t^* = S_t$ in which case

$$A(E,V^*) = (f,V^*) + \langle q,V^* \rangle_{\partial \Omega_2} - A(U,V^*) = (f,V) + \langle q,V \rangle_{\partial \Omega_2} - A(U,V) \quad \forall\ V \varepsilon S_t \qquad (8)$$

which is zero by equation (5). To provide useful estimates of the errors S_t^* must be a richer space than S_t. One possible choice is to use polynomials of one order higher for S_t^* which is the approach taken by a number of investigators including Babuska and Miller [57]

who used piecewise biquadric functions for E and V^* when the finite element solution, U and V employed bilinear functions. As stated in equation (8), the computational effort required to solve the error estimation equations is on the same order as the original finite element analysis. To reduce this computational cost additional approximations are necessary. For example, Adjerid and Flaherty [58,59] employed nodal superconvergence by neglecting the errors at the nodes relative to that within the element to reduce the solution of the error equations to the solution of a number of local Dirichlet problems associated with the nodes.

Another approach to the derivation of the error equation is to replace u by U + e in the equation (3) substituting this into the weighted residual form and applying the divergence theorem which yields the elemental level error expression [55,60]

$$A(e,v)_{\Omega_i} = (f,v)_{\Omega_i} + \langle q,v\rangle_{\partial\Omega_{2i}} - A(U,v)_{\Omega_i} + \langle au_\eta ,v\rangle_{\partial\Omega_i} \quad \forall\ v\varepsilon H_o^1 \tag{9}$$

$$A(u,v)_{\Omega_i} = \int_{\Omega_i} [a\nabla u\nabla v + buv]d\Omega \tag{9b}$$

$$(f,v)_{\Omega_i} = \int_{\Omega_i} fv\ d\Omega_i \tag{9c}$$

$$\langle aU_\eta ,v\rangle_{\partial\Omega_i} = \int_{\partial\Omega_i} au_\eta v\ ds \tag{9d}$$

where

Ω_i is the domain of the element, $\partial\Omega_i$ is its boundary, u_η is the normal derivative of u on the element boundary.

A key to the application of (9a) is the evaluation of the third term on the right hand side since it contains the only unknown, u_η. A possible approximation for measuring this term is to use the average value obtained from the two elements sharing the boundary which, when applied with a specific set of weight functions, V^*, yields

$$A(E,V^*)_{\Omega_i} = (f,V^*)_{\Omega_i} + \langle q,V^*\rangle_{\partial\Omega_i} - A(U,V^*)_{\Omega_i} + 1/2\ \langle a(U^+_\eta + U^-_\eta),\ V^*\rangle_{\partial\Omega_i} \qquad \forall\ V^* \varepsilon S^*_t \tag{10}$$

where U^+_η and U^-_η denote the value of the normal derivatives on either side of the edge.

An alternative form of the elemental error equation can be obtained by integrating the third term on the right hand side of equation (9) by parts to give

$$A(e,v)_{\Omega_i} = A_r(U,v)_{\Omega_i} + \langle q,v\rangle_{\partial\Omega_{2i}} - \langle aU_\eta\ ,v\rangle_{\partial\Omega i} + \langle au_\eta\ ,v\rangle_{\partial\Omega i} \qquad \forall\ v\varepsilon H^1_o \tag{11a}$$

where the first term on the right hand side of equation (11a) is the weighted integral of the residual over the element of the finite element solution defined as

$$A_r(U,v)_{\Omega_i} = \int_{\Omega_i} (\nabla(a\nabla U)v - bUv + fv)\ d\Omega \qquad \forall\ v\varepsilon H^1_o \tag{11b}$$

Again, a key aspect of working with this form, referred to as the residual form, of the error estimate is dealing with the last term on the right hand side of equation (11a) which is a function of the unknown solution u. A more appropriate method to account for this term in the residual form of the error estimate is to combine it with the other boundary terms in equation (11a) producing the so called jump term, $\langle \Delta aU_\eta, v\rangle_{\partial\Omega_i}$ defined as

$$\langle \Delta aU_\eta\ ,v\rangle_{\partial\Omega} = \begin{cases} \langle a(u_\eta - u_\eta),v\rangle_{\partial\Omega_i}, & \partial\Omega_i \notin \partial\Omega_2 \\ \langle (q - aU_\eta),v\rangle_{\partial\Omega_i}, & \partial\Omega_i\ \varepsilon\ \partial\Omega_2 \end{cases} \tag{12}$$

where

$$\langle a(u_\eta - U_\eta, v\rangle_{\partial\Omega_i} = \int_{\partial\Omega_i} a(u_\eta - U_\eta)\, v\, ds$$

$$\langle (q - aU_n), v\rangle_{\partial\Omega_i} = \int_{\partial\Omega_i} (f - aU_\eta)\, v\, ds$$

Thus the jump term represents a weighted integral of the difference between the normal derivatives of the exact and finite element solutions for those portions of the element boundary upon which the normal derivatives have not been defined, and a weighted residual of the difference between the prescribed normal derivatives and the normal derivatives of the finite element solution on the portions of the element boundary upon which the normal derivatives are prescribed. This form leads to a natural selection for an approximation to the jump term when an estimate to the error is to be obtained. Selecting a set of weighting function, V^*, an approximation to the error is obtained as

$$A(E,V^*)_{\Omega_i} = A_r(U,V^*)_{\Omega_i} + \langle \Delta a U_\eta, V^*\rangle^*_{\partial\Omega_i} \quad \forall\, V^* \in S_t^* \tag{13a}$$

where

$$\langle \Delta a U_\eta, V^*\rangle_{\partial\Omega_i} = \begin{cases} 1/2\langle a(U_\eta^+ - U_\eta^-), V^*\rangle_{\partial\Omega_i}, & \partial\Omega_i \notin \partial\Omega_2 \\ \langle (q - aU_\eta), V^*\rangle^*_{\partial\Omega_i}, & \partial\Omega_1 \in \partial\Omega_2 \end{cases} \tag{13b}$$

The term $(U_\eta^+ - U_\eta^-)$ represents the jump in normal derivatives between two elements.

A number of investigators [51,52,60-64] have used equation (13) with various selections of finite subspace(s), S_t^*, for the functions, V^*, outlined above. It is interesting to note that in the application of equation (13) with linear or bilinear finite elements the jump term tends to dominate the a posteriori error estimator. This observation

has recently been confirmed by Babuska and Yu [62,63] who proved that the discretization error for odd-order elements is primarily due to the jump terms. They have also shown [62,63] that when even order elements are used, the interior residual, $A_r(U,V^*)$, dominates the discretization error estimate. This allows one to neglect the jump terms in these cases which means the error estimation process requires only element integrals which can greatly reduce the programming complexity of adaptive analysis procedures [64] by avoiding the need to track and calculate the interelement boundary integrals.

Mesh Improvement in Adaptive Analysis

After an estimate to the total error is obtained, the next step is to determine how to improve the finite element model such that the desired level of accuracy is obtained. One method to do this is by the uniform improvement of the entire mesh by either subdividing each element into a number of new elements of the same type (h-refinement) or increasing the polynomial order of all elements (p-refinement). Although convergent, such an approach is unsatisfactory from the viewpoint of computational efficiency. It also turns out to be unsatisfactory for use with many of the error estimation procedures since the accuracy of the estimate often depends on the mesh having a near optimum mesh distribution. Therefore, it is important to devise a means to improve the finite element discretization is an optimal, or near optimal, manner.

One approach to generating a near optimum mesh that yields the requested degree of accuracy is to directly employ the information generated during the error estimation process. This is a fairly straight forward process since the majority of the error estimation procedures calculate elemental level contributions to the overall error estimate equations, equations (10) or (13) for example, and sum them in an appropriate manner to obtain the global error estimate. That is

$$E^{\alpha} = \Sigma \eta_i^{\alpha} \qquad (14)$$

where η_i is the contribution from element i and is referred to an the elemental error indicator, and the exponent α is set so that the summation is proper, for example $\alpha=2$ if the error is measured in the energy norm. A simple strategy to the development of a near optimal mesh is to improve the discretization within individual elements when

$$\eta_i \geq \lambda \max_j \eta_j \qquad 0 \leq \lambda \leq 1 \qquad (15)$$

Although simple, such an approach develops meshes in which the η_i's are nearly equal in each element. It has been proven that the optimum mesh for one dimensional elliptic problems is one in which the error indicators are equal, in an asymptotic sense, for all elements [65]. It has also been demonstrated numerically that equilibrating the error indicators in meshes in higher dimensions is a near optimal strategy for elliptic problems. This property, although often used and seemingly reasonable, is not likely to be optimal for parabolic or hyperbolic problems where the influence of time must be considered.

If more that a single procedure for mesh enrichment is available, such as selecting between element subdivision or increasing the polynomial order of selected elements for example, the error indicators η_i can not tell which would be more effective for a selected element. Although the error indicators will properly dictate mesh improvement in the asymptotic sense, they may not lead to the best selections in a practical sense. For example, assume the mesh improvement is carried out by adding higher order polynomial shape functions and that the error existing in the solution is orthogonal to that new term. In this case, the addition of that term will not reduce the solution error. To address this, the concept of a correction indicator, γ_i, has been introduced [66]. The function of a correction indicator is to estimate the amount of solution improvement that will be gained by the application of a particular mesh enrichment procedure. By evaluating several possibilities, one can select that which will yield the greatest improvements. (It should be noted that most error indicators are correction indicators for one particular enrichment method.) This concept appears well suited for use with hierarchic mesh enrichment

procedures [66].

Once the portions of the mesh requiring improvement are determined, the finite element discretization in that area must be improved. There are a number of techniques available to carry out these improvements including;

1. relocating the positions of nodes within a given finite element mesh topology (r-refinement),
2. subdividing selected elements into smaller elements of the same type (h-refinement),
3. increasing the polynomial order of selected elements (p-refinement),
4. defining an entirely new mesh topology with an improved distribution of elements,
5. various combinations of two or more of the above techniques.

Each approach has its advantages and disadvantages with the most efficient approach being dependent of the class of equation being solved, smoothness of the solution, dimension of the domain of the solution, and the overall modeling and computing environment available.

The earliest methods for adaptively improving finite element meshes considered the positions of the node points of a given mesh as unknowns in the energy functionality governing the system [67,68]. The resulting minimization problem, with appropriate constraints to insure the domain and mesh topology remained unaltered, was then solved to provide both the positions of the nodes and the values of the primary unknowns at those nodes. Although the use of this approach, coupled with a standard minimization procedure for nonlinear merit function and constraints, is not commonly used for the solution of elliptic equations, r-refinement techniques based on more direct node moving criteria are being successfully used for the solutions to nonlinear parabolic and hyperbolic problems. In these cases, the original

partial differential equations are reduced to a set of ordinary differential equations (ODE's) in time by the introduction of the finite element discretization into an appropriately defined functional which has the amplitudes of the functions at the nodes and the velocity of the nodal positions as unknowns [69]. The functionality used contains a specific penalty term to insure the mesh remains valid. These problem types require time marching, and in the nonlinear case, iteration. Therefore, the extra computation required to calculate improved positions for the nodes can be more than compensated for by the fact that a much coarser overall mesh can be used. In fact, it has been found [69] that very accurate results can be obtained for some classes of problems by using r-refinement methods on coarse meshes. A drawback of r-refinement methods is that since these methods do not introduce new degrees of freedom into the system, there is a limit on the solution accuracy possible which is dependent of the number of elements and initial mesh topology. These methods also require special care to maintain the validity and numerical stability of meshes as the nodes move. The complexity of dealing with the mesh validity and numerical stability increases drastically as one increases the dimensionality of the problem.

One of the most commonly used methods to increase the numbers of degrees of freedom in a finite element mesh is to introduce more elements of the same type into the mesh. In a feedback procedure, this is typically done by subdividing selected elements into a new set of elements of the same type, thus decreasing the size of the elements in that area. This approach is referred to as h-refinement because the mesh improvements are carried out by reducing the size of elements which is typically measures in terms of a length parameter h.

There are a number of methods possible to subdividing selected elements into new ones, however, care must be exercised in the selection and application of procedures. An important consideration is the control of the shape of the element, particularly if several levels of refinement are applied is which case a refinement procedure that causes deterioration in element shapes can lead to elements with

numerical conditioning problems. This concern leads to the use of element bisection methods in which the subelements formed are similar in shape to the parent element [70-73]. Figure 8 demonstrates the application of element bisection of a single element in both a quadrilateral and trianqular mesh. In each scheme, a subdivided element is replaced by four subelements with nodes introduced at the midpoint of each of the original element sides. If these new nodes lie along the edge of an element that is not subdivided, such as nodes 7 and 8 in the quadrilateral mesh and node 6 in the triangular mesh, constraint equations must be written to maintain the continuity requirements along that edge. The handling of the constraints, as well as the efficient solution of the sequence of meshes defined as the process continues, can be addressed by the careful selection of data structures and solution algorithms [35,71-75].

H-refinement procedures for triangles have been devised in which the need for constraint equations are avoided [74-76]. This is done by allowing elements neighboring subdivided elements to be split in a manner that constraints are not needed to maintain continuity. This splitting does reduce the shape quality of the element, however, it is only applied for one level; and, in a temporary manner such that if those elements are to be subdivided, the subdivision is applied to the original element.

An advantage of the p-refinement method is that improvements in solution accuracy are obtained by increasing the polynomial order of selected elements without the need to alter the mesh topology. This process is made even more effective by the use of hierarchic finite element elements where the shape functions for an element of order p is a subset of those for the element of order p+1 [61,66,77] which means the stiffness equations for an enriched mesh can be efficiently generated by simply adding new terms to the previous stiffness matrix. It is also possible to employ these shape functions in a manner that avoids the need to write constraint equations to maintain inter-element continuity when elements of different polynomial orders neighbor each other. Another benefit of p-refinement procedures is

that the rate of convergence, in the energy norm when defined in terms of the number of unknowns, is better in elliptic problems with singularities [79,80]. For these reasons, these approaches are receiving considerable attention in the adaptive analysis literature.

Another feedback approach to the development of improved finite element meshes is to use the results on the current mesh to guide the generation of an entirely new mesh. Simplistically, this approach could be considered a combination of r- and h-refinement which need not suffer from the basic restrictions of either. That is, it can be structured to allow the equivalent of node movement, but without the restrictions of maintaining a fixed mesh topology, and it allows the number of elements to be increased without the need to consider constraint equations. The two questions that must be addressed in the application of such an approach are; the information to dictate the element distributions and how a new mesh will be generated based on that information. One approach that has been developed plotted contours of a specific solution parameter that gave the analyst an indication of how the mesh should be distributed and then allowed him/her to then interactively generate a new mesh that followed those contours [81]. A more recent approach defines a mesh density function over the domain of interest that is then used by an automatic mesh generator to generate a new mesh that has an appropriate element distribution to efficiently calculate a solution of the required level of accuracy [82].

In addition to the individual application of the above mesh enrichment schemes, it is possible to apply them in various combinations. For some classes of problems, the proper combination of two techniques appears quite appropriate. The first is the combination of r- and h-refinement techniques for the solution of parabolic or hyperbolic equations. In these problem types, it is often possible to obtain greatly improved solutions with only a given amount of mesh motion. However, since r-refinement methods do not allow for an increase in the number of unknowns, it may not be possible to obtain the required degree of accuracy with them alone. Therefore, the addition of

h-refinement where needed can supply the additional unknowns needed.

In the case of elliptic problems with singularities present, it has been shown [28,29,80,83] that the proper combination of h- and p-refinement can be an extremely efficient combination. In particular, is has been shown that optimal hp-refinement procedures can give exponential rates of convergence in the energy norm in terms of the number of degrees of freedom.

Automatic Mesh Generators and A Posteriori Mesh Control

The various mesh enrichment schemes indicated above can be combined with automatic mesh generators to provide the mesh generation and control needed for the development of automated finite element analysis systems. One aspect of combining the mesh enrichment procedure directly with the functionality of an automatic mesh generator is that the mesh refinement can be carried out such that the mesh's approximation to the domain being analyzed is improved as the mesh is improved. For example, consider the use of h-refinement where the boundaries of the domain are curved, but the initial, coarse mesh consist of straight sided elements. If the element refinement is carried out based on the element information only, the meshes approximation to the boundary is never improved over that defined by the initial mesh. However, if a close link back to the original geometry is maintained through the mesh generator, the refinement process can use the capabilities of the automatic mesh generator to place new boundary nodes on the boundary of the object.

In general, there are specific combinations of algorithmic approaches to automatic mesh generation and mesh refinement that are appropriate for three-dimensional geometries. Mesh generation algorithms based on Delaunay triangulation are well suited for use with h-refinement schemes that avoid the need to apply constraint equations. This can be done by using the error indicators to place additional points in those portions of the mesh that are not fine enough. Then Watson's algorithm [15] can be used to determine the affected elements to be removed,

thus creating new elements using the added node inside the element. Approaches of this type have been developed for two-dimensional domains [84-85] in which minor alterations to the strict adherence to a Delaunay mesh properties have been made. Since the basic Delaunay mesh properties cause complications in the three-dimensional case, similar modifications are likely.

The application of h-refinement in combination with mesh generators based on spatial decomposition is an attractive combination since the tree structure used to store the decomposition of the domain can be used effectively in the adaptive process [35,38,76]. In this approach, the mesh refinement would be carried out by the appropriate refinement of the cells of the decomposition based on the values of the error indicators of the elements inside the cell. Since the tree used to define the spatial decomposition can maintain pointers back to the geometric entities located within it [24,38,76], the enrichment of the mesh in that cell can efficiently account for any geometry approximation improvements. This is an important feature in the three-dimensional case since the amount of computation required for the mesh generation process is high and any localization of the process possible leads to substantial computational savings.

Approaches have been developed that combine h-refinement and spatial decomposition mesh generators that do [35] and do not [38,76] require the application of constraint equations. In both cases, the tree structure plays a critical role.

In the case where mesh refinement is carried out by cell bisection only [35], it is necessary to apply constraints on the cell boundaries when there is a level (cell size) difference. However, by the appropriate use of the information in the tree structure, not only can the need to apply constraints be quickly determined, but, with the right combination of solution procedures, the finite element solution, including constraints, can be efficiently calculated [35]. (Since an adaptive analysis process requires a number of analyses, the advantageous use of this tree structure to control the entire solution

process can lead to substantial computational savings).

The need to apply constraint equations can be avoided by directly employing all the features of the automatic mesh generator. For example, procedures have been developed for the finite quadtree [34] and finite octree mesh generators [24,37] that use the tree structure to determine the cells that are affected by mesh refinement to re-mesh only those cells, at their new levels, using the facilities of the mesh generator [76]. This process is depicted graphically in Figure 9 for a finite quadtree example. The mesh before refinement is shown in Fig. 9a, while Fig. 9b shows the area that is affected by the refinement removed. The cells at their new levels are then defined, Fig. 9c, and the mesh topology is created in those cells thus creating the refined mesh shown in Fig. 9d. Figure 9d also demonstrated the automatic improvement of geometry approximation gained by doing the refinement through the functionality of the mesh generator. The process is identical in the three-dimensional case. The same concepts can be used to perform de-refinement in portions of the model where the error indicators say the mesh is finer than needed. Such a capability is particularly useful in time dependent problems where the critical regions of the model change with time.

The generation of entirely new meshes based on the error indicators is also possible with automatic mesh generators based on spatial decomposition. In this case, all that is needed is information that dictates the levels of the tree, and thus the cell sizes, for all the cells. This process is in fact much the same as the local remeshing procedures indicated above, except the entire mesh is redone instead of refining and/or de-refining only portions of the model.

The use of automatic mesh generators for hp-refinement is another possibility. Since the basic form of the mesh can be indicated in an a priori manner based on the geometry and analysis attributes (loads, material properties and boundary conditions) [24,28,29], the initial mesh can be generated using the proper basic mesh topology. The adaptive mesh updates then consists of only some minor mesh

enhancements in local regions and increasing the polynomial order of selected elements. As indicated previously, not all automatic meshing approaches can produce the coarse exponentially graded meshes needed for these cases. However, a properly constructed element removal mesh generator can produce the meshes needed. This mesh generator would generate the initial mesh [24] by first invoking an operator that isolates and removes all singularities. The remaining operators then create the coarsest possible mesh in the rest or the domain (see Fig. 3 for such an example). An initial analysis can be carried out and the results used to determine the number of layers of elements needed around the singularity [28]. These can then be easily inserted and adaptive analysis using p-refinement continued.

CONCLUDING REMARKS

This paper has reviewed the algorithmic approaches currently available for the truly automatic generation of finite element meshes. Although these approaches have been under consideration for a number of years for two-dimensional domains, it is the recent efforts on three-dimensional techniques, coupled with the geometric modeling procedures needed to support them, that is making them an important capability needed to improve the general usefulness of the finite element method in engineering design. Mesh generators of the type discussed in this paper are beginning to become available to the finite element user community. By their nature, they will require substantially more computational effort than other techniques. However, the amount of user input required to use them will reduce the amount of user time needed to generate a valid finite element mesh to a small fraction of what is required using other techniques.

To be used most effectively, these mesh generation procedures must be coupled with adaptive analysis procedures that can insure that the final mesh yields the requested degree of accuracy. Without adaptive analysis procedures based on reliable a posteriori error estimators, the analyst will need to use a priori mesh control techniques to generate the desired element distributions. However, more importantly,

the analyst will not know if the results produced insure the desired level of accuracy. Although adaptive techniques to completely control errors in any norm of interest are not yet available, the currently available techniques do represent an important capability that can be effectively used to produce much more reliable finite element results.

Increasing the level of automation and reliability in the finite element modeling process is necessary if finite element analysis is to be a common part of engineering design. Ultimately, consideration need be given to the complete automation of the finite element modeling process.

ACKNOWLEDGEMENTS

The author would like to acknowledge the input of Professor Joseph E. Flaherty for his review of the section on a-posteriori error estimation and the efforts of Peggy L. Baehmann, Kurt R. Grice and Marcel K. Georges for developing the programs used to generate most of the finite element models shown in the figures.

The support of the National Science Foundation, under Grant MSM83-05950 and DCM-8603025 and the Industrial Associates of the RPI Center for Interactive Computer Graphics. Any opinions expressed in this paper are those of the author and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

1. A.A.G. Riquicha and H.B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment", IEEE Computer Graphics and Applications, Vol. 3, No. 2, 1982, pp. 9-24.

2. A.A.G. Riquicha and H.B. Voelcker, "Solid Modeling: Current Status and Research Directions", IEEE Computer Graphics and Applications, Vol. 3, No. 7, October 1983, pp. 25-37.

3. M.S. Pickett and J.W. Boyse, Eds., Solid Modeling by Computers: From Theory to Applications, Plenum Press, 1984.

4. O.C. Zienkiewicz and D.V. Phillips, "An Automatic Mesh Generation Scheme for Plane and Curved Surfaces by Isoparametric Co-ordinates", Int. J. Num. Meth. Engng., Vol. 3,

No. 4, 1971, pp. 519-528.

5. W.J. Gordon and C.A. Hall, "Construction of Curvilinear Co-ordinates Systems and Applications to Mesh Generation", Int. J. Num. Meth. Engng., Vol. 7, 1973, pp. 461-477.

6. J.E. Thompson, Numerical Grid Generation, North-Holland, 1982.

7. W.A. Cook, "Body Oriented (Natural) Co-ordinates for Generating Three Dimensional Meshes", Int. J. Num. Meth. Engng., Vol. 8, 1974, pp. 27-43.

8. R.B. Haber, M.S. Shephard, J.F. Abel, R.H. Gallagher, and D.P. Greenberg, "A Generalized Two-Dimensional Graphical Finite Element Preprocessor Utilizing Discrete Transfinite Mappings", Int. J. Num. Meth. Engng., Vol. 17, 1981, pp. 1015-1044.

9. W.R. Buell and B.A. Bush, "Mesh Generation - A Survey", Trans. ASME J. of Engng. for Industry, Vol. 95, pp. 332-338, 1973.

10. C.O. Frederick, Y.C. Wong and F.W. Edge, "Two-Dimensional Automatic Mesh Generation for Structural Analysis", Int. J. Num. Meth. Engng., Vol. 2, 1970, pp. 113-144..

11. J.C. Cavendish, D.A. Field and W.H. Frey, "An Approach to Automatic Three-Dimensional Mesh Generation", Int. J. Num. Meth. Engng., Vol. 21, 1985, pp. 329-347.

12. J.C. Cavendish, "Automatic Triangulation of Arbitrary Planar Domains for the Finite Element Method", Int. J. Num. Meth. Engng., Vol. 8, 1974, pp. 679-697.

13. Nguyen-Van-Phai, "Automatic Mesh Generation with Tetrahedron Elements", Int. J. Num. Meth. Engng., Vol. 18, 1982, pp. 273-289.

14. R. Sibson, "Locally Equiangular Triangulations", The Computer Journal, Vol. 21, No. 3, 1978, pp. 243-245.

15. D. F. Watson, "Computing the n-Dimensional Delaunay Tessellation with Applications to Voronoi Polytypes", The Computer Journal, Vol. 24, No. 2, 1981.

16. Y.T. Lee, A. de Pennington and N.K. Shaw, "Automatic Finite Element Mesh Generation from Geometric Models - A Point-Based Approach", ACM Transactions on Graphics, Vol. 3, 1984, pp. 287-311.

17. S.H. Lo, "A New Mesh Generation Scheme for Arbitrary Planar Domains", Int. J. Num. Meth. Engng., Vol. 21, 1985, pp. 219-249.

18. D.A. Field, "Implementing Watson's Algorithm in Three Dimensions", Proc. Second Annual ACM Symposium on Computational

Geometry, ACM 0-89791-194-6/86/0600/246, 1986, pp. 246-259.

19. D.A. Field and W.H. Frey, "Automation of Tetrahedral Mesh Generation", Research Publication GMR-4967, General Motors Research Laboratories, Warren, MI, 1985.

20. B. Wordenweber, "Finite Element Mesh Generation", Computer-Aided Design, Vol. 16, 1984, pp. 285-291.

21. T.C. Woo and T. Thomasa, "An Algorithm for Generating Solid Elements in Objects with Holes", Computers and Structures, Vol. 18, No. 2, 1984, pp. 333-342.

22. E.A. Sadek, "A Scheme for the Automatic Generation of Triangular Finite Elements", Int. J. Num. Meth. Engng., Vol. 15, 1980, pp. 1813-1822.

23. F.T. Tracy, "Graphical Pre- and Post-Processors for Two-Dimensional Finite Element Programs", Computer Graphics, Transactions of ACM, Vol. 13, 1977, pp. 8-12.

24. M.S. Shephard, K.R. Grice and M.K. Georges, "Some Recent Advances in Automatic Mesh Generation", Modern Methods for Automating Finite Element Mesh Generation, K. Baldwin, Ed., ASCE, NY, 1986, 1-18.

25. A. Bykat, "Automatic Generation of Triangular Grid: I - Subdivision of General Convex Subregions, II - Triangulation of Convex Polygons", Int. J. Num. Meth. Engng., Vol. 10, 1976, pp. 1329-1342.

26. B. Joe and R.B Simpson, "Triangular Meshes for Regions on Complicated Shapes", Int. J. Num. Meth. Engng., Vol. 23, 1986, pp. 751-778.

27. P.F. Charvez, "Automatic Mesh Generation and Optimization from the Solid Model Database", Modern Methods for Automating Finite Element Mesh Generation, K. Baldwin, Ed., ASCE, NY, 1986, pp. 29-42.

28. I. Babuska and E. Rank, "An Expert-System-Like Approach in the hp-Version of the Finite Element Method", Institute for Physical Science and Technology Lab. for Num. Analysis, TN BN-1048., University of Maryland, 1986.

29. B.A. Szabo, "Mesh Design for the p-Version of the Finite Element Method", Computer Meth. in App. Mech. and Engng., Vol. 55, 1986, pp. 181-197.

30. M.L.C. Sluiter and D.L. Hansen, "A General Purpose Two- and Three-Dimensional Mesh Generator", Computers in Engineering, Vol. 3, L.E. Hulbert, Ed., Book No. G00217, ASME, 1982, pp. 29-34.

31. A.J.C. Schoofs, L.H.Th.M. Van Beukering and M.L.C. Sluiter, "A General Purpose Two-Dimensional Mesh Generator", Advances in Engineering Software, Vol. 1, No. 3, 1979, pp. 131-136.

32. M.A. Yerry and M.S. Shephard, "Finite Element Mesh Generation Based on a Modified-Quadtree Approach", IEEE Computer Graphics and Applications, Vol. 3, No. 1, 1983, pp. 36-46.

33. M.S. Shephard and M.A. Yerry, "Approaching the Automatic Generation of Finite Element Meshes", Computers in Mech. Engng., Vol. 1, No. 4, 1983, pp. 49-56.

34. P.L. Baehmann, S.L. Wittchen, M.S. Shephard, K.R. Grice and M.A. Yerry", Robust Geometrically-Based Automatic Two-Dimensional Mesh Generation", TR-86007, Center for Interactive Computer Graphics, RPI, Troy, NY, 1986, to appear, Int. J. Num. Meth. Engng..

35. A. Kela, R. Perucchio and H.B. Voelcker, "Towards Automatic Finite Element Analysis", Computers in Mech. Engng., July 1986, pp. 57-71.

36. M.A. Yerry and M.S. Shephard, "Automatic Three-Dimensional Mesh Generation for Three-Dimensional Solids", Computers and Structures, Vol. 20, 1985, pp. 173-180.

37. M.A. Yerry and M.S. Shephard, "Automatic Three-Dimensional Mesh Generation by the Modified-Octree Technique", Int. J. Num. Meth. Engng., Vol. 22, 1984, pp. 1965-1990.

38. M.S. Shephard, M.A. Yerry and P.L. Baehmann, "Automatic Mesh Generation Allowing for Efficient A Priori and A Posteriori Mesh Refinements", Computer Meth. in Appl. Mech. Engng., Vol. 55, 1986, pp. 161-180.

39. C.L. Jackins and S.L. Tanimoto, "Octrees and Their Use in the Representation of Three-Dimensional Objects", Compt. Graph. Image Process., Vol. 14, 1980, pp. 249-270.

40. D. Meagher, "Geometric Modeling Using Octree Encoding", Computer. Graph. Image Process., Vol. 19, 1982, pp. 129-147.

41. R.A. Dwyer, "A Simple Divide-and-Conquer Algorithm for Constructing Delaunay Triangulations in O (n log log n) Expected Time", Proc. Second Annual ACM Symposium on Computational Geometry, ACM 0-89791-194/6/86/0600/276, 1986, pp. 276-284.

42. J.D. Bolssonnat and M. Tellaud, "A Hierarchical Representation of Objects: The Delaunay Tree", Proc. Second Annual Symposium on Computational Geometry, ACM 0-89791-194- 6/86/0600/260, 1986, pp. 260-268.

43. I. Babuska, O.C. Zienkiewicz, J. Gago and E.R. De A. Oliveria,

Accuracy Estimates and Adaptive Refinements in Finite Element Computations, John Wiley and Sons, Chichester, 1986.

44. K.J. Weiler, "Edge Based Data Structures for Solid Modeling in Curved-Surface Environments", IEEE Computer Graphics and Applications, Vol. 5, No. 1, January 1985, pp. 21-40.

45. K.J. Weiler, "Topological Structures for Geometric Modeling", PhD Thesis, Center for Interactive Computer Graphics, TR-86032, Rensselaer Polytechnic Institute, Troy, NY, 1986.

46. P.R. Wilson, "Data Transfer and Solid Modeling", Geometric Modeling for CAD Applications, M.J. Wozny, H.W. McLaughlin and J.L. Encarnacao, Eds., North Holland, to appear.

47. "Applications Interface Specification (Restructured Version)", CAM-I Report R-86-GM-01, January 1986.

48. M.S. Shephard, "Finite Element Modeling within an Integrated Geometric Modeling Enviroment: Part I - Mesh Generation", Engineering with Computers, Vol. 1, 1985, pp. 61-71.

49. M.S. Shephard and P.M. Finnigan, "Integration of Geometric Modeling and Advanced Finite Element Preprocessing", Proc 4th Chautauqua on Productivity in Engineering and Design...The Quest for Quality, H. Shaeffer, Ed., PDA Eng., Los Angeles, CA, 1986, pp. 231-233.

50. G. Strang and G. Fix, An Analysis of the Finite Element Method, Prentice Hall, 1973.

51. I. Babuska and W.C. Rheinboldt, "A Posteriori Error Estimate for the Finite Element Method", Int. J. Num. Meth. Engng., Vol. 12, 1978, pp. 1597-1615.

52. I. Babuska, W.C. Rheinboldt, "Error Estimates for Adaptive Finite Element Computations", SIAM J. Numer. Anal., Vol. 15, No. 4, 1978, pp. 736-754.

53. I. Babuska, "A Posteriori Error Estimates and Adaptive Approaches for Finite Element Modeling", Finite Element Workshop 1980, Technical Note BN-940, I. Babuska, Ed., Laboratory for Numerical Analysis, U. of Maryland, May, 1980.

54. Proceedings of Int. Conf. on Accuracy Estimates and Adaptive Refinements in Finite Element Computations, Vol. 1 and 2, Int. Association of Computational Mechanics, 1984.

55. S. Adjerid and J.E. Flaherty, "Local Refinement Finite Element Methods on Stationary and Moving Meshes for One-Dimensional Parabolic Sytstems", to appear.

56. J.N. Reddy, An Introduction to the Finite Element Method, McGraw-Hill Book Co., NY, 1984.

57. I. Babuska and A. Miller, "A Posteriori Error Estimates and Adaptive Techniques for the Finite Element Method", Institute for Physical Science and Technology, Laboratory for Numerical Analysis, Tech. Note BN-968, University of Maryland, 1981.

58. S. Adjerid and J.E. Flaherty, "A Moving Finite Element Method for Time Dependent Partial Differential Equations with Error Estimation and Refinement", SAMI Numer. Anal., Vol. 23, pp. 778-796, 1985.

59. S. Adjerid and J.E. Flaherty, "A Moving Mesh Finite Element Method with Local Refinement for Parabolic Partial Differential Equations", Comp. Meths. Appl. Mech. Engng., 1986, pp. 3-26.

60. R.E. Bank, "Analysis of a Local A Posterori Error Estimate for Elliptic Equations", Accuracy Estimates and Adaptive Refinements in Finite Element Computations, I. Babuska, O.C. Zienkiewicz, J. Gago and E.R. de A. Oliveria, Eds., 1986, pp. 119-128.

61. O.C. Zienkiewicz, J. Gago and D.W. Kelly, "The Hierarchical Concept in Finite Element Analysis", Computers and Strucutres, Vol. 16, 1983, pp. 53-65.

62. I. Babuska and D. Yu, "Asymptotically Exact A Posteriori Error Estimates and Adaptive Approaches for Biquadratic Elements", Tech. Note BN-1050, Institute for Physical Science and Technology, U. of Maryland, 1986.

63. I. Babuska and D. Yu, "A Posteriori Error Estimation for Biquadratic Elements and Adaptive Approaches", to appear.

64. S. Adjerid and J.E, Flaherty, "Second-Order Finite element Approximations and A Posteriori Error Estimation for Two-Dimensional Parabolic Systems", Report No. 87-1, Department of Computer Science, Rensselaer Polytechnic Institute, 1987.

65. I. Babuska and W.C. Reinboldt, "Analysis of Optimal Finite Element Meshes in R^1", Math. of Comp., Vol. 33, 1979, pp. 431-463.

66. O.C. Zienkiewicz and A. Craig, "Adaptive Refinement, Error Estimates, Multigrid Solution and Hierarchic Finite Element Method Concepts", Accuracy Estimates and Adaptive Refinements in Finite Element Computations, I. Babuska, O.C. Zienkiewicz, J. Gago and E.R. de A. Oliveria, Eds., 1986, pp. 25-59.

67. D.J. Turcke and G.M. McNeice, "Guidelines for Selecting Finite Element Grids Based on an Optimization Study", Computers and Structures, Vol. 4, 1974, pp. 449-519.

68. C.A. Fillipa, "Optimization of Finite Element Grids by Direct Energy Search", Appl. Math. Modeling, Vol. 1, September 1976,

pp. 93-96.

69. K. Miller, "Recent Results on Finite Element Methods with Moving Nodes", Accuracy Estimates and Adaptive Refinements in Finite Element Computations, I. Babuska, O.C. Zienkiewicz, J. Gago and E.R. de A. Oliveria, Eds., 1986, pp. 225-338.

70. R.J. Melosh and P.V. Marcal, "An Energy Basis for Mesh Refinement of Structural Continua", Int. J. Num. Meth. Engng., Vol. 11, No. 7, 1977, pp. 1083-1092.

71. W.C. Rheinboldt and C.K. Mesztenyi, "On a Data Structure for Adaptive Finite Element Mesh Refinements", ACM Transaction on Maths. Software, Vol. 6, No. 2, June 1980, pp. 166-187.

72. W.C. Rheinboldt, "Adaptive Mesh Refinement Processes for Finite Element Solutions", Int. J. Num. Meth. Engng., Vol. 17, 1981, pp. 649-662.

73. R.E. Ewing, "Adaptive Mesh Refinements in Large-Scale Fluid Flow Simulation", Accuracy Estimates and Adaptive Refinements in Finite Element Computations, I. Babuska, O.C. Zienkiewicz, J. Gago and E.R. de A. Oliveria, Eds., 1986, pp. 229-314.

74. R.E. Bank, A.H. Sherman and A. Weiser, "Refinement Algorithms and Data Structures for Regular Local Refinement", Scientific Computing: Applications of Mathematics and Computing to the Physical Sciences, R.S. Stepleman, Ed., North Holland, 1983, pp. 3-17.

75. M.C. Rivara, "Algorithms for Refining Triangular Grids Suitable for Adaptive and Multigrid Techniques", Int. J. Num. Meth. Engng., Vol. 20, 1984, pp. 745-756.

76. M.S. Shephard, J.E. Flaherty and P.L. Baehmann, "Adaptive Analysis for Automated Finite Element Modeling", to appear, Proc. of The Mathematics of Finite Elements and Application, 1987.

77. A.G. Peano, "Hierarchies of Conforming Finite Elements for Plane Elasticity and Plate Bending", Comp. Math. with Appl., Vol. 2, 1976.

78. A. Peano, R. Riccioni, A. Pasini and L. Sardella, "Adaptive Approximations in Finite Element Structural Analysis", Computers and Structures, Vol. 10, 1979, pp. 333-342.

79. I. Babuska and B.A. Szabo, "On the Rates of Convergence of the Finite Element Method", Int. J. Num. Meth. Engng., Vol. 18, 1982, 323-341.

80. I. Babuska and M. Dorr, "Error Estimates for the Combined h and p Versions of the Finite Element Method", Numer. Math., Vol. 25, 1981, pp. 257-277.

81. M.S. Shephard, R.H. Gallagher and J.F. Abel, "Synthesis of Near-Optimum Finite Element Grids with Interactive Computer Graphics", Int. J. Num. Meth. Eng., Vol. 15, 1980, pp. 1021-1039.

82. J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz, "Adaptive Remeshing of Compressable Flow Computations", Institute for Numerical Methods in Engineering, CR/R/544/86, U. College Swansea, Swansea, Wales, 1986.

83. B. Guo and I. Babuska, "The h-p Version of the Finite Element Method - Part 1: The Basic Approximation Results; Part II - General Results and Applications", to appear, Computational Mechanics.

84. W.H. Frey, "Selective Refinement: A New Strategy for Automatic Node Placement in Graded Triangular Meshes", GM Research Publication, GMR-5432, 1986.

85. J. Penman and M.D. Grive, "An Approach to Self-Adaptive Mesh Generation", IEEE Transactions on Magnetics, Vol. MAG-21, No. 6, 1985, pp. 2567-2570.

FIGURE CAPTIONS

1. Watson's Algorithm for inserting a point into a Delaunay triangulation

2. Basic three-dimensional element removal operators

3. h-p mesh generated by element removal

4. Subdomain removal to decompose object into mappable regions

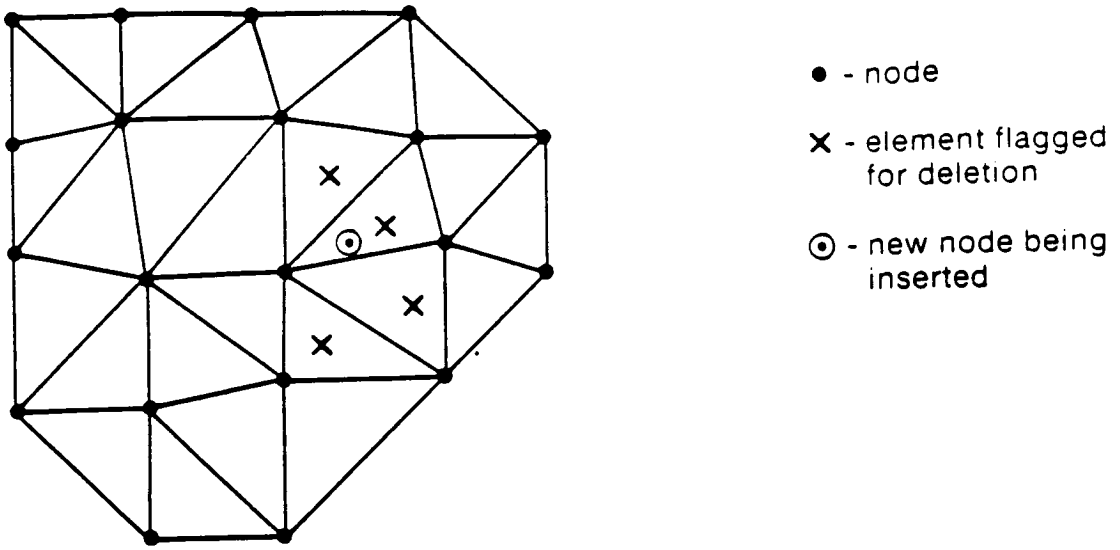
5. Finite octree mesh example

6. Finite quadtree mesh control
   a) uniform mesh
   b) graded mesh
   c) mesh control parameters for graded mesh

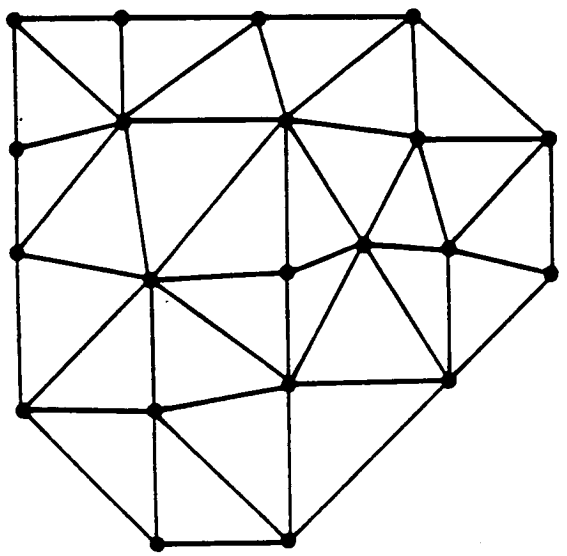
7. Finite octree mesh control
   a) uniform mesh
   b) graded mesh

8. h-refinement by element bisection
   a) quadrilateral element
   b) traingualar element

9. Finite quadtree mesh refinement by local remeshing
   a) initial mesh
   b) affected portion of mesh removed
   c) refined quadrants
   d) resulting refined mesh

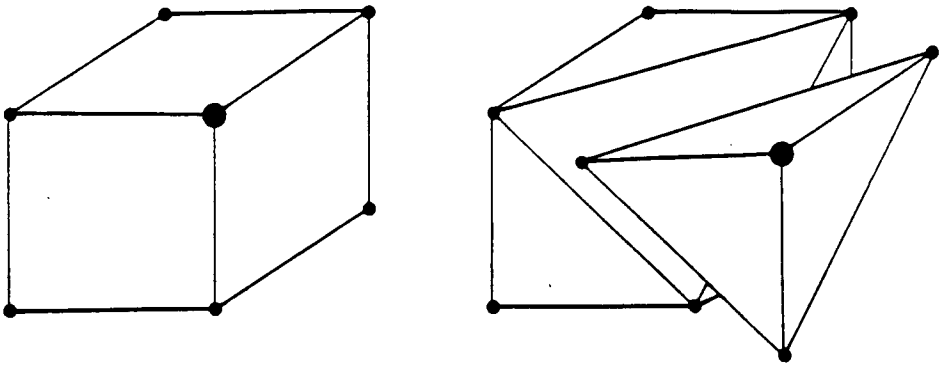


a) original mesh with new node inserted

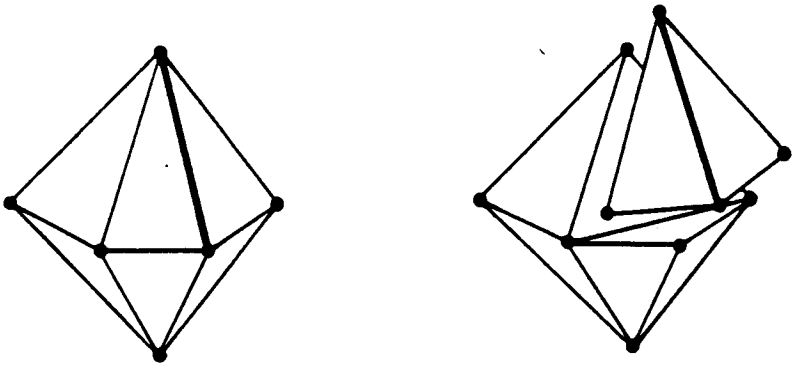


b) resulting Delaunay triangulation

FIG. 1. Watson's Algorithm for inserting a point into a Delaunay triangulation



a) vertex removal



b) edge removal

FIG. 2. Basic three-dimensional element removal operators

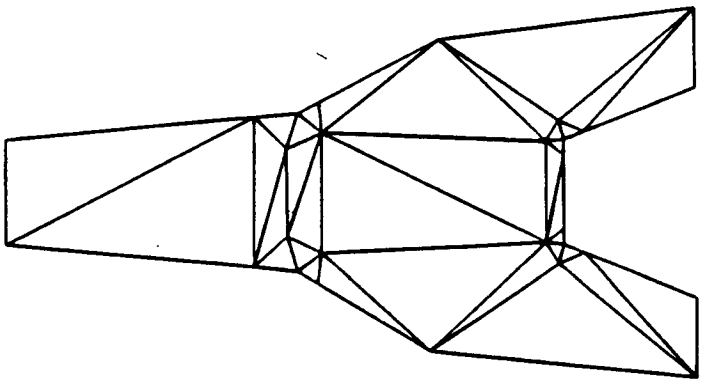
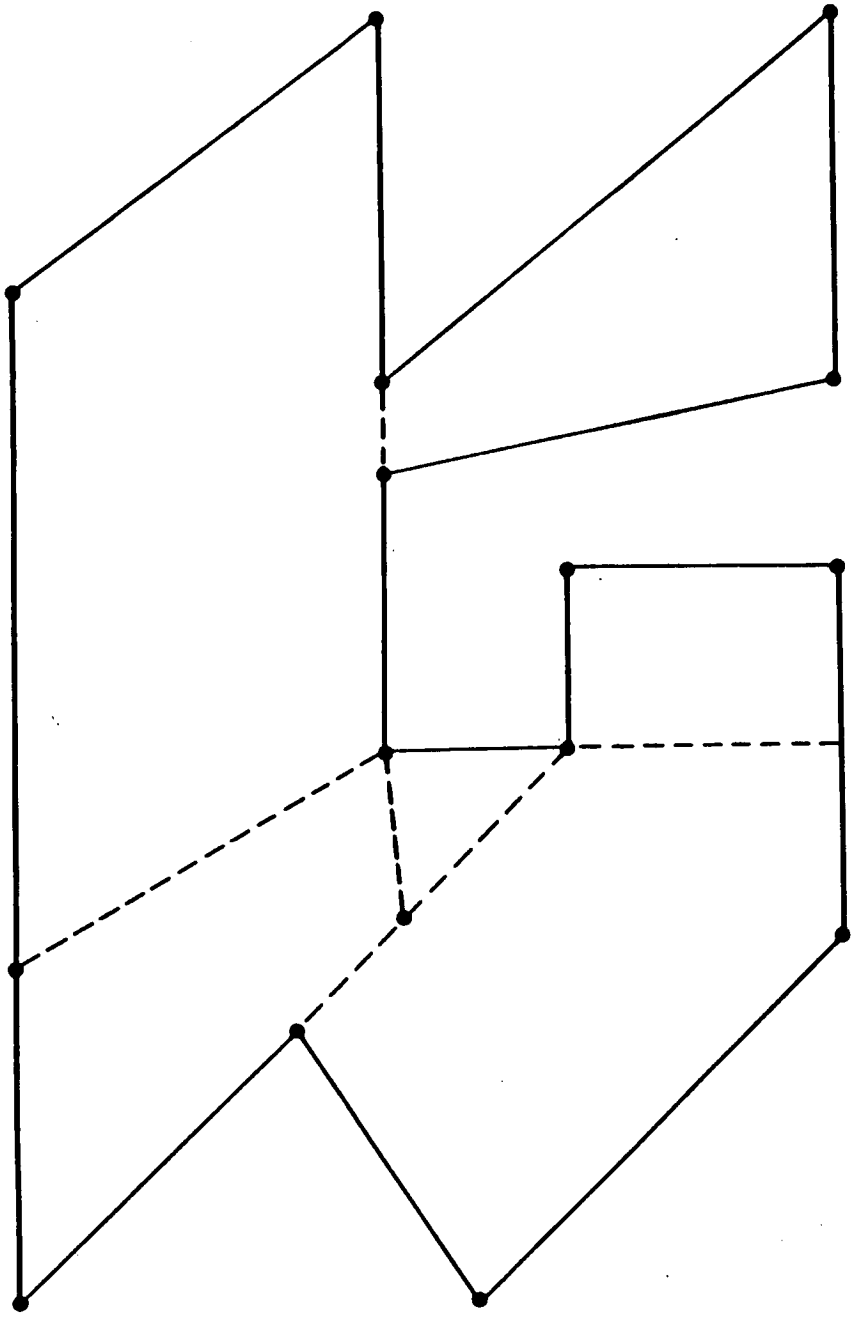
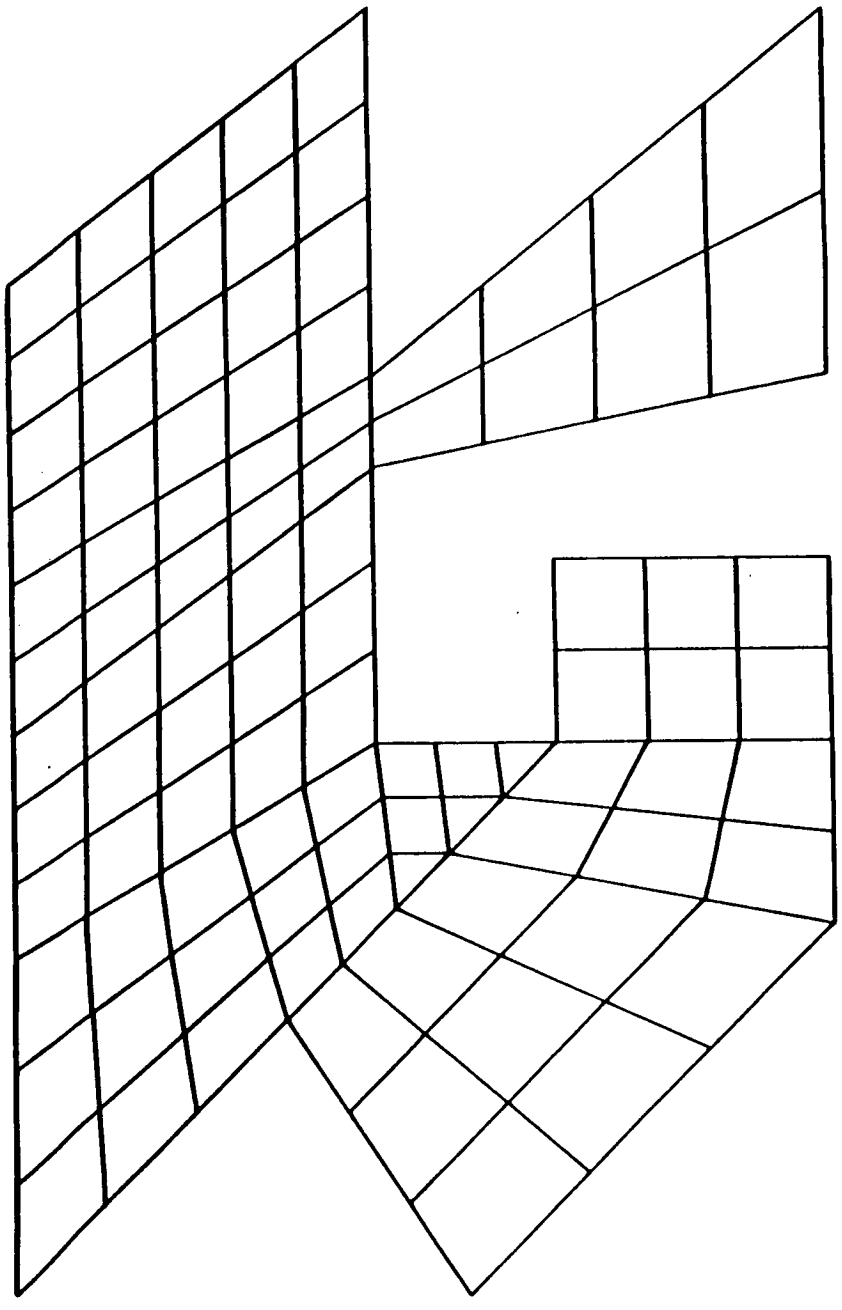


FIG. 3. h-p mesh generated by element removal



a) original geometry with general mesh batch boundaries (dashed lines)



b) resulting mesh

FIG. 4. Subdomain removal to decompose object into mappable regions

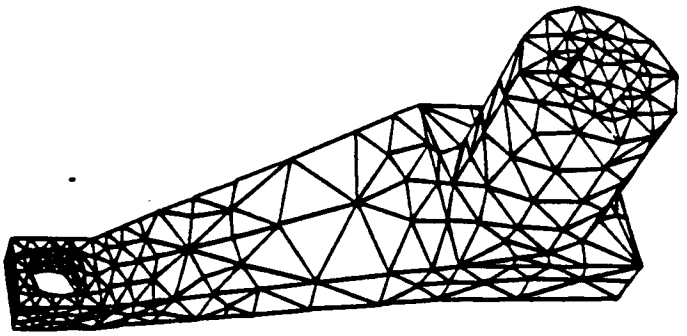


FIG. 5. Finite octree mesh example

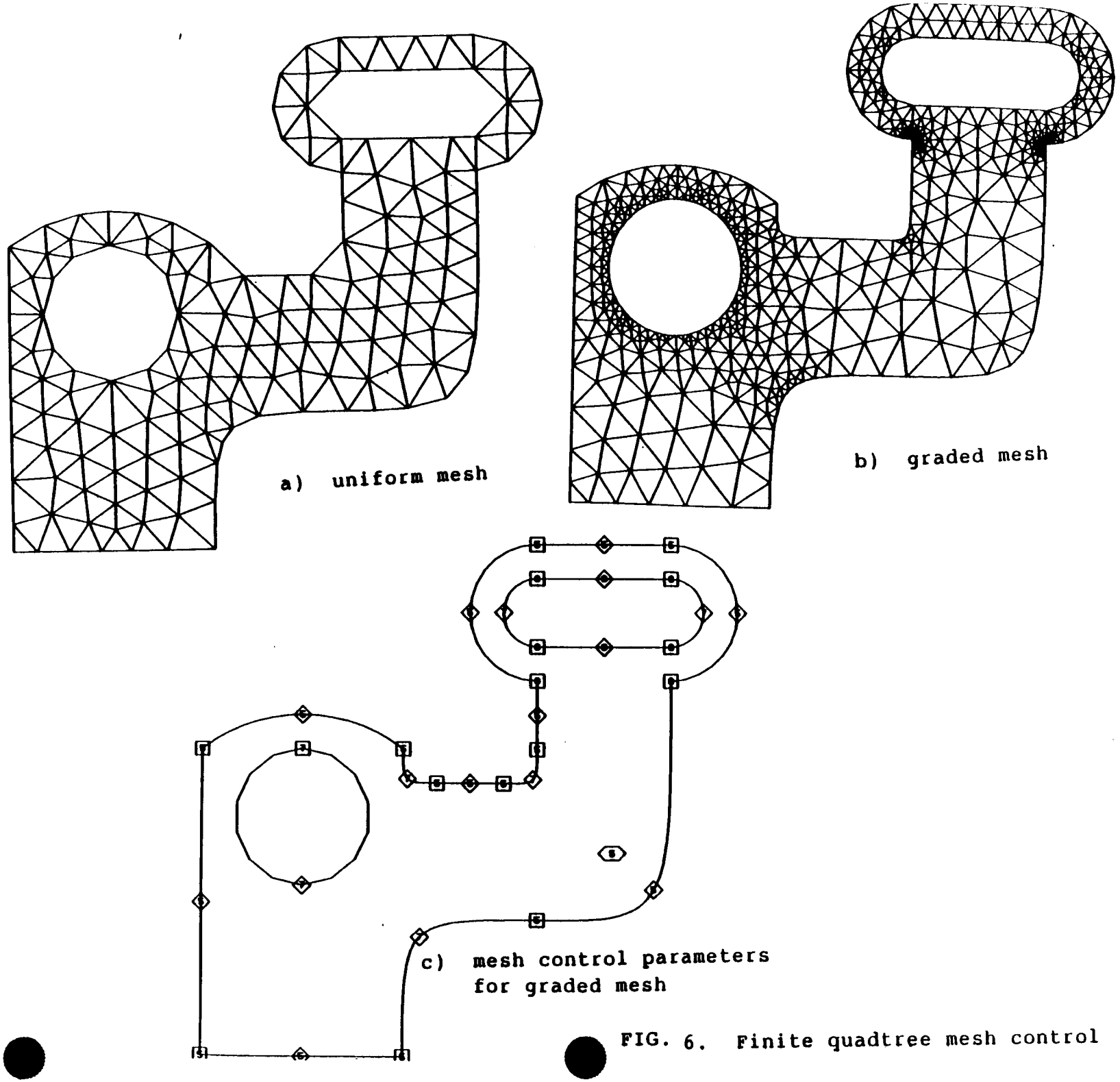
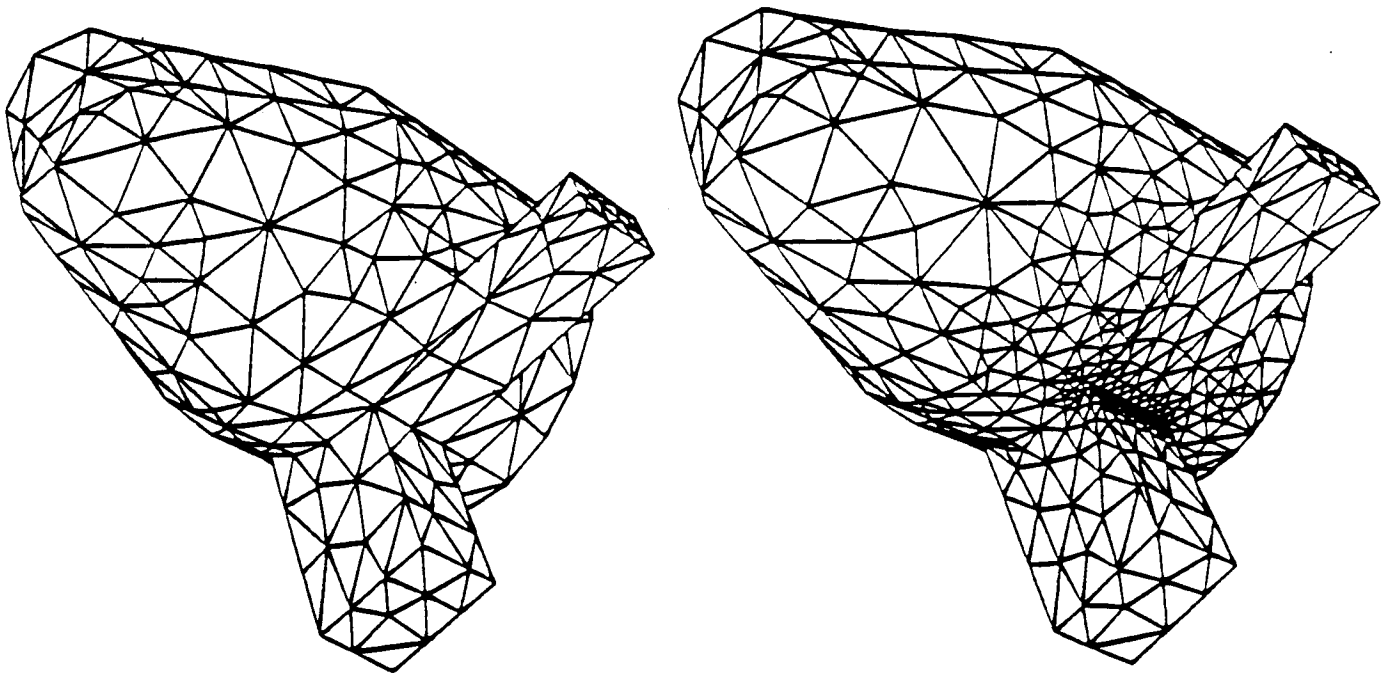


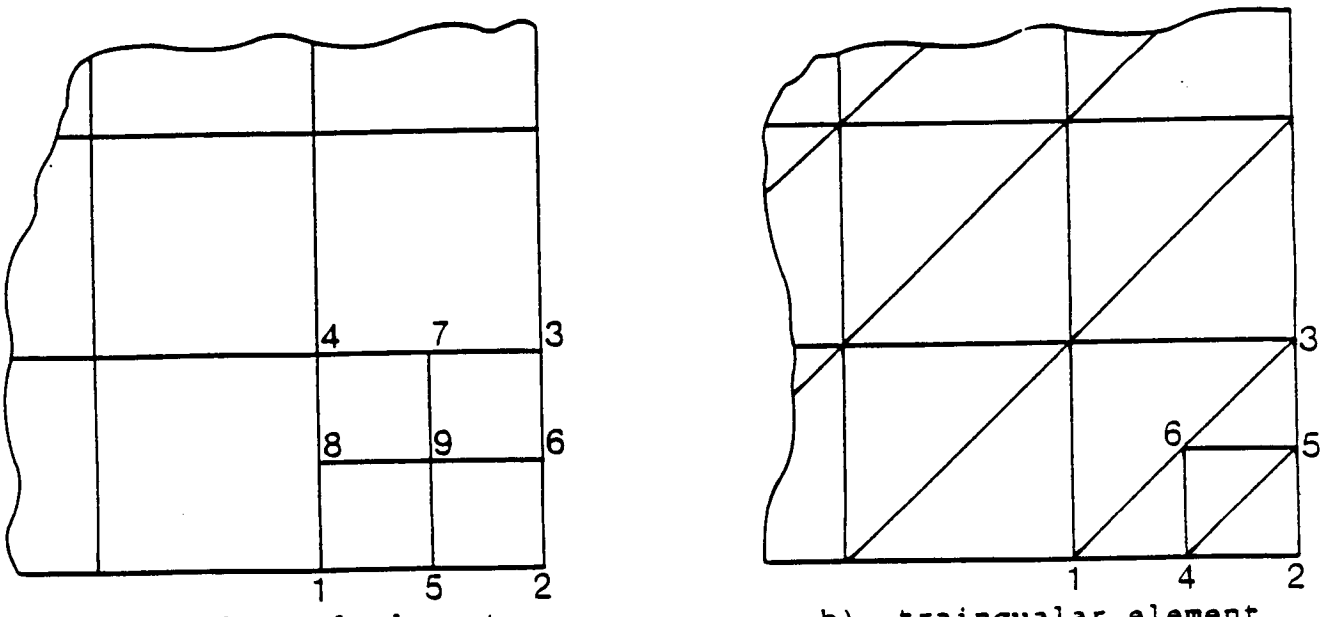
FIG. 6. Finite quadtree mesh control



a) uniform mesh

b) graded mesh

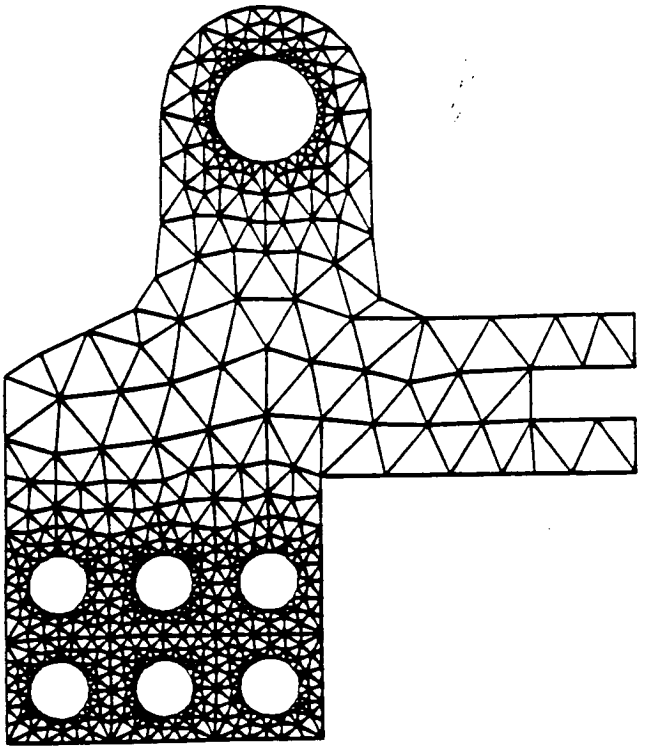
FIG.7. Finite octree mesh control



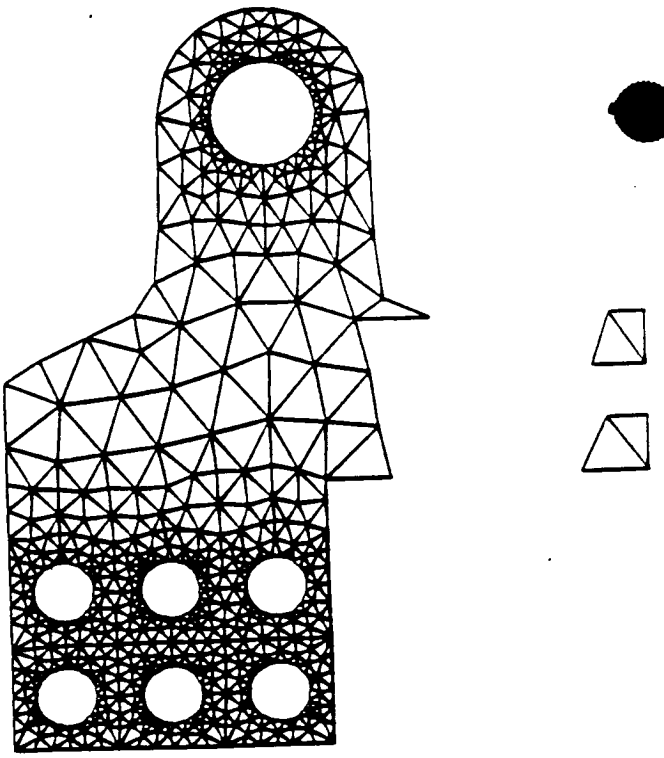
a) quadrilateral element

b) traingualar element

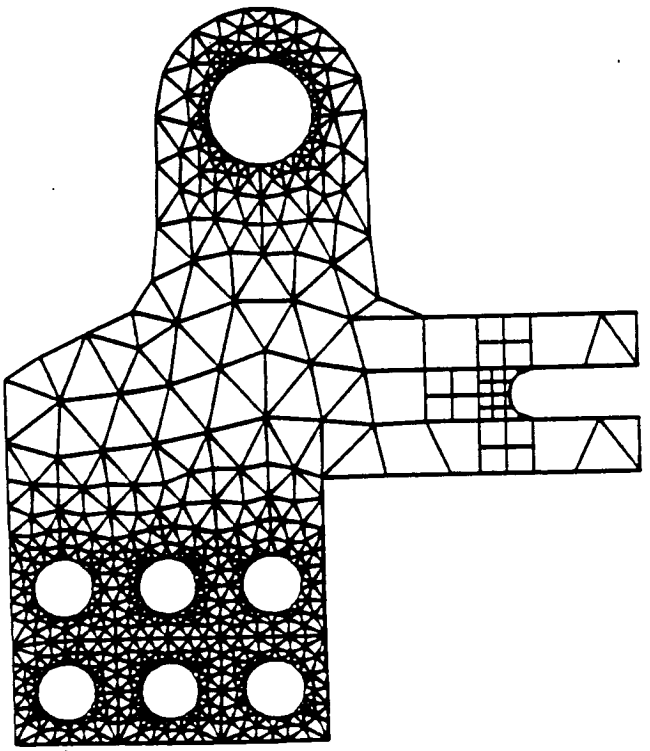
FIG. 8. h-refinement by element bisection



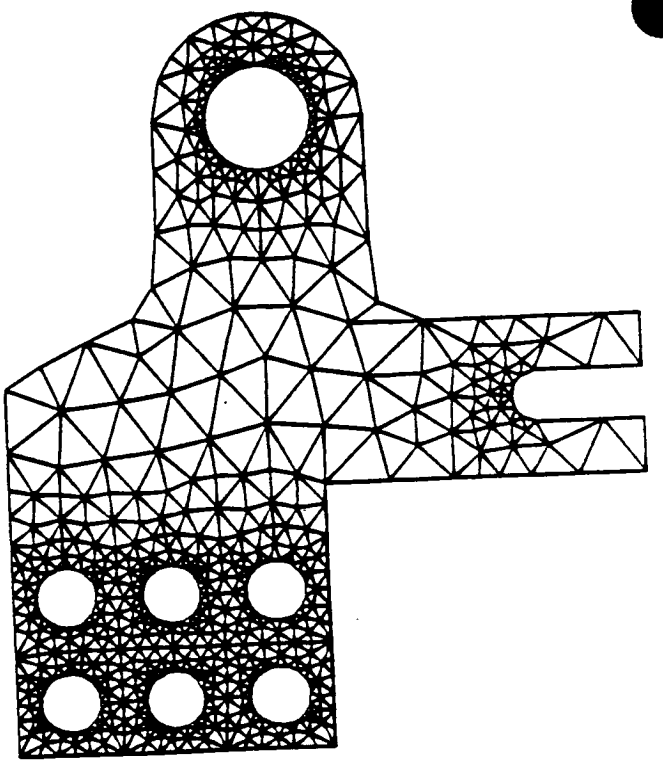
a) initial mesh



b) affected portion of mesh removed



c) refined quadrants



d) resulting refined mesh

FIG. 9. Finite quadtree mesh refinement by local remeshing