15

Automatic Grid Generation Using Spatially Based Trees

Mark S. Shephard

Hugues L. de Cougny

Robert M. O'Bara

Mark W. Beall

15.1 Introduction

This chapter examines the use of spatially based trees defined by recursive subdivision methods in the automatic generation of numerical analysis grids. The application of recursive subdivision over a spatial domain begins with a regular shape that is subdivided, in some regular manner, into a number of similarly shaped pieces, to be referred to as tree cells. The subdivision process is recursively applied until the smallest individual cells satisfy a given criteria. This subdivision process leads naturally to the definition of a spatially based tree structure where the root node of the tree corresponds to the starting regular shape, and the nodes of the tree defined by its recursive subdivision correspond to a specific portion of the spatial domain. The terminal nodes represent the smallest cell defined for that portion of the domain.

Recursive subdivision provides a natural means to decompose a geometric domain into a set of terminal cells that can be related to the grids or elements used in a numerical analysis. The associated tree structure provides an effective means for supporting various operations common to grid generation and numerical analysis, including determining the cell covering a particular location in space and determining neighbors. If the shape of the geometric domain of the analysis corresponded directly to the regular shape of the root node, the process of automatic grid generation using recursive subdivision would be trivial. Since

0-8493-2687-7/99/$0.00+$.50
© 1999 by CRC Press LLC

the geometric domain of the analysis typically has a complex shape, specific consideration must be given to the interaction of the cells of the tree and the geometric domain of the analysis. Alternative methods for determining and representing those interactions have been devised for use in automatic grid generation. The method selected strongly influences all aspects of the grid generation process.

Determining the interactions of the cells of the tree with the analysis geometry and the decomposition of the cells into elements represents the most complex aspect of automatic grid generation using spatially based trees. In those cases where the tree cells are directly allowed to represent whatever portion of the analysis geometry included within them, the grid generation process is straightforward. The only technical issues relate to indicating the appropriate information to the analysis procedure for those cells containing some portion of the boundary of the domain on their interior. In those cases, where the elements defined in the tree cells have to conform to the geometry, the creation of elements in cells containing portions of the boundary of the domain is far more complex. In the worst case, the element creation procedures used in those cells represent complete automatic mesh generation procedures.

Section 15.2 outlines spatial subdivision techniques and associated trees that have been used in automatic mesh generation. Section 15.3 describes the basic issues that must be addressed in the use of spatial subdivision in automatic grid generation. Section 15.4 presents the techniques used in conjunction with automatic grid generation to construct the spatially based tree. Section 15.5 discusses the issues and approaches used to create elements within the cells of the tree. Finally, Section 15.6 indicates procedures that can be applied to improve the mesh after the basic mesh has been constructed.

15.2 Recursive Domain Subdivision to Define Spatially Based Trees

The application of recursive subdivision of a domain into subdomains, and the definition of an associated tree structure, has a long history (see Samet for a review of the area [3]) in a number of application areas including computer graphics, image processing, and computational geometry [9,10] (grid generation can be considered a computational geometry application). There are a variety of means in which the domains can be subdivided and the associated trees defined. For purposes of this discussion, emphasis will be placed on the quadtree structures for two-dimensional domains, and octree structures for three-dimensional domains, which have been most commonly used in grid generation (see also Section 3 of Chapter 14).

Considering the two-dimensional case, the first step in the generation of a quadtree for a given object is the definition of a rectangular-piped, typically a square, which covers the domain of the object. The rectangle is then subdivided into the four quadrants defined by bisecting each of the sides of the rectangle. Each quadrant is then examined to determine if it is to be subdivided based on given subdivision criteria. If they are to be subdivided, the process of creating the four quadrants for that rectangle is repeated. The process continues until the subdivision criteria are satisfied throughout the domain.

The process naturally defines a tree structure where the nodes in the tree correspond to rectangles at a particular point in the process. The tree is referred to as a quadtree, since four children are defined each time a node is subdivided an additional level in the tree. The original rectangle that encloses the object defines the root of the tree. The four quadrants defined by the subdivision of the root define the next level. These quadrants are each tested against the subdivision criteria. If they pass the criteria, they are marked as terminal quadrants. Any quadrant that does not pass the criteria is subdivided into its four quadrants, which form level two of the tree. This process is continued until all quadrants satisfy the given criteria, or the maximum tree level is reached. An octree for a three-dimensional object is defined in the same way, with the only difference being that the rectangular hexahedron, typically a cube, is subdivided into its eight octants such that each parent node in the domain has eight children.

A common cell (quadrant or octant) subdivision criteria used by many applications of spatial quadtrees and octrees is to refine the cell if it contains any of the boundary of the object, that is if the cell is neither fully within a single material region or exterior to the model. Figure 15.1 demonstrates the generation

ɔnsideration must be given
ılysis. Alternative methods
ɛ in automatic grid gener-
on process.
ry and the decomposition
generation using spatially
t whatever portion of the
ɪrward. The only technical
for those cells containing
ıere the elements defined
cells containing portions
ıent creation procedures

that have been used in
: addressed in the use of
ɪues used in conjunction
discusses the issues and
5.6 indicates procedures
ted.

atially

inition of an associated
ɔer of application areas
9,10] (grid generation
ɪf means in which the
; discussion, emphasis
e structures for three-
(see also Section 3 of

tree for a given object
ıin of the object. The
sides of the rectangle.
ı subdivision criteria.
ıngle is repeated. The
ı.

ɔond to rectangles at
children are defined
gle that encloses the
f the root define the
ass the criteria, they
subdivided into its
ɪuadrants satisfy the
ıal object is defined
typically a cube, is
t children.
of spatial quadtrees
if the cell is neither
ıtes the generation

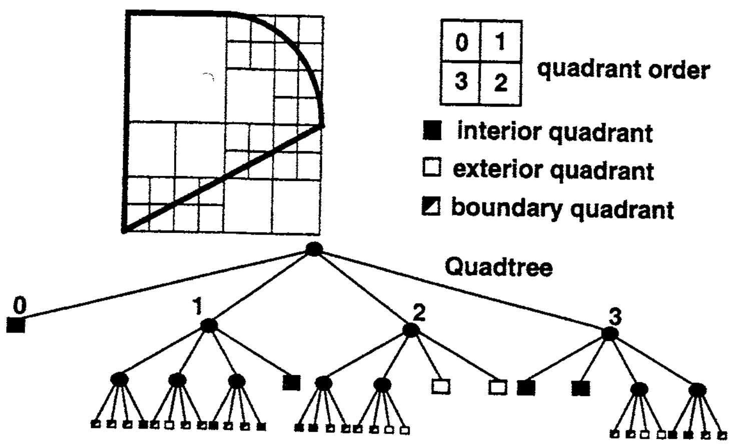


FIGURE 15.1 Quadtree example.

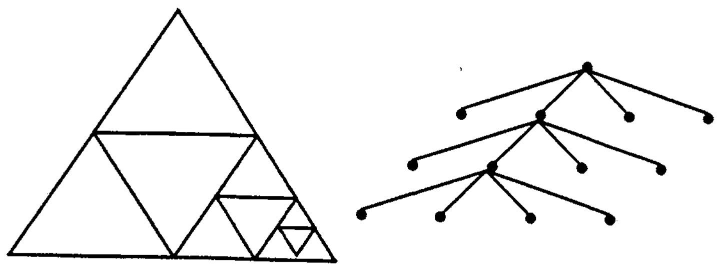


FIGURE 15.2 Quadtree defined by the subdivision of a triangle.

of a four-level quadtree for a simple two-dimensional domain bounded by three line segments and a circular arc. The object and tree quadrants resulting from three levels of subdivision of quadrants containing portions of the boundary are shown in the upper portion of Figure 15.1. The bottom portion of the figure shows the resulting tree structure. By the subdivision criteria used in this example, each parent quadrant contains a portion of the boundary and is neither fully inside or outside the object. The terminal quadrants are marked are either interior, exterior, or boundary depending on their relationship to the geometric domain.

There are alternative spatial decompositions and associated storage structures. One possibility is to consider the recursive subdivision of cells with alternative shapes. For example, in two dimensions, the root cell could be a single equilateral triangle and its four children defined by the bisection of the three sides forming four similar triangles, as shown in Figure 15.2. The extension of this procedure to three-dimensional simplices is not straightforward since the subdivision of a tetrahedron does not yield a set of similar tetrahedra (a regular tetrahedron does not close pack).

An alternative possibility to construct a spatially based structure is to consider anisotropic refinement of cells in which cells are only bisected in selected directions. Such subdivision processes do require the introduction of alternative structures for their definition. One example is a switching function representation [27,28] in which subdivision of cells can be limited to whichever coordinate direction is desired. Figure 15.3 shows the application of a switching function representation to the simple two-dimensional domain used earlier.

15.3 Quadtrees and Octrees for Automatic Mesh Generation

Octree and quadtree structures have been used to support the development of two- and three-dimensional mesh generators for a number of years [1,2,3,11,12,16–18,20,22,24,26,31,32]. Although each of the

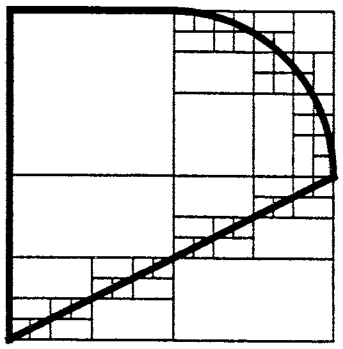


FIGURE 15.3 Switching function representation of a two-dimensional example.

quadtree- and octree-based mesh generators are different, there are specific basic aspects common to all the procedures: the mesh generation process is implemented as a two-step discretization process. The quadtree or octree is generated in the first step. The tree is then used to localize many of the element-generation processes, which constitute the second step. Those cells (quadrants or octants) containing portions of the object's boundary receive specific consideration to deal with the boundary of the object. The corners of the cells are used as nodes in the mesh. In specific procedures, additional nodes are defined by the interaction of the boundary of the object being meshed with the cells' boundaries. The mesh gradation is controlled by varying the level of the cells within the tree through the domain occupied by the object.

The specific algorithmic steps used within a quadtree- or octree-based mesh generator depend strongly on the assumptions made with respect to the representation of the boundary of the model, and on the form of interaction between the boundary of the model and a tree cell that is represented. Before discussing the alternative tree construction and element creation algorithms, these basic options for the representation of the model boundary and its interaction with the tree are discussed.

In general, the geometric domains to be meshed are curvilinear models defined within a geometric modeling system. The tree-based mesh generation procedures can attempt to interact directly with this curvilinear geometry, or require a polygonal approximation. The use of a polygonal approximation greatly simplifies the determination of the interactions of the geometric model with the tree cells. The polygonal approximation may be constructed through a process which is independent of the mesh generation process, or it may be the boundary triangulation that defines the surface mesh. These two polygonal forms are typically handled differently.

Factors that enter into the selection of the approach to account for the interactions of the model boundary with the boundary of the tree cells include (1) level of geometric approximation desired, (2) sensitivity of the element creation procedures to small features created by the model and cell boundary interactions, (3) importance of maintaining spatial associativity of the resulting tree cells. Figure 15.4 demonstrates three basic options (columns) for representing the interactions of the tree and model boundary (top row) and the potential influence on the resulting mesh (bottom row). The first option (left column) employs exact interactions of the model and tree as defined by the intersections of the model and cell boundaries. This option maintains the spatial associativity of the tree cells*, and does not introduce any geometric approximations. However, under the normal assumption in mesh generation that the trees cells are on the order of the size of the desired elements, this approach has the disadvantage of producing disproportionally small and distorted elements (see mesh in lower-left corner of Figure 15.4) when the model and cell interactions leave small portions of a cell in the model.

*Spatial associativity of the tree cells is maintained when the cells remain undistorted. When spatial associativity is maintained, the appropriate tree cell can be obtained by traversing down from the root using the coordinates of a point.

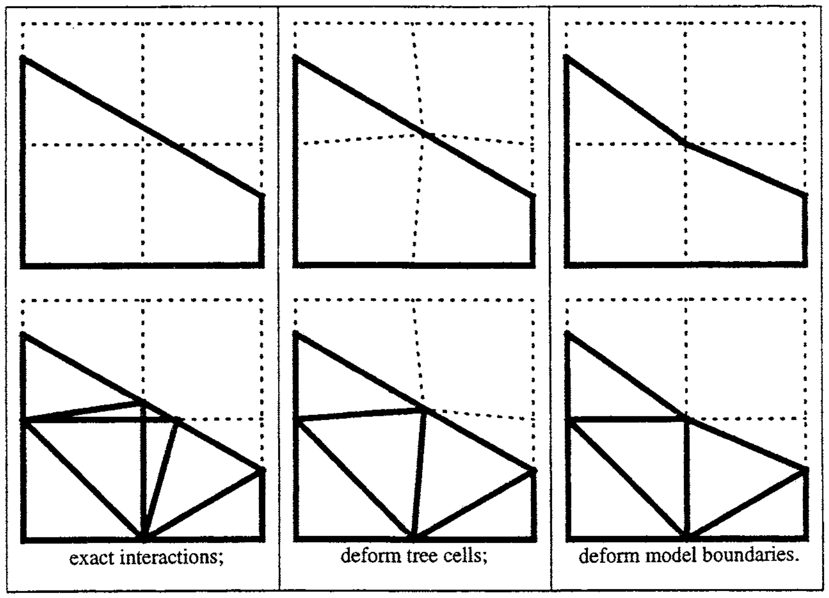


FIGURE 15.4 Options for the interactions of the model boundary with the boundary of the tree cells.

The other two options eliminate the influence on the mesh of these small portions of the cells by either distorting the necessary cells (center column, Figure 15.4), or distorting the geometry of the appropriate model entities (right column, Figure 15.4). Both approaches require the development of specific logic to determine when and how to perform the needed distortion. If tree cells are deformed (center column, Figure 15.4) they no longer can employ operations that rely on spatial associativity, while the deformation of the model can introduce undesirable geometric approximations into the process.

There are a variety of options available to create elements once the tree has been defined and the cells qualified with respect to the model. The tree cells that are interior to the domain of the object are typically meshed quickly employing procedures that take specific advantage of the simplicity of the cell's topology and shape, and use knowledge of the tree structure to determine the influence of neighboring cells on the mesh within the cell of interest.

Meshing of the cells that contain portions of the boundary of the object is a more complex process with the details being strongly influenced by the representation of the boundary of the model being used and the method used to represent the tree's boundary cells (see Section 15.5).

The input information required to generate the tree structure for use in mesh generation is the geometric model and information on the size of elements desired throughout the domain. For purposes of this discussion, no specific representation of the geometric domain is assumed. Instead, it is assumed that it exists and there is support for the interrogations of that representation needed to obtain the information required for the various operations performed during the tree construction and meshing processes. This approach allows a more uniform presentation of the tree building and element creation procedures, and provides a generalized method to link the meshing procedures to the domain geometry in a consistent manner [21,23].

In this discussion it is explicitly assumed that the size of the terminal cells throughout the domain of the geometric model is on the order of the element sizes required. Therefore, the information on desired element sizes will define the sizes of the terminal cells in the tree. Any spatially based mesh control functions can be easily represented using such an approach.

TABLE 15.1 Topological Entities for the Three Models

Model	Geometric	Octree	Mesh
Regions	G_i^3	O_i^3	M_i^3
Faces	G_i^2	O_i^2	M_i^2
Edges	G_i^1	O_i^1	M_i^1
Vertices	G_i^0	O_i^0	M_i^0

15.4 Tree Construction for Automatic Mesh Generation

15.4.1 Preliminaries

It is convenient to view the process of octree- and quadtree-based mesh generation as one of discretizing the geometric model into a model defined by the cells of the tree, and then discretizing the cells of the tree into the mesh model. Both of these steps require interactions with the geometric model. Irrespective of the algorithmic details used to carry out these steps, the key issue in ensuring the resulting mesh is valid is understanding the relationship of the mesh to the geometric model [19,23]. At the most basic level, the relationships between the models can be described in terms of the association of the topological entities defining the boundaries of the various model entities. In three dimensions the primary topological entities are regions, faces, edges and vertices which will be denoted for the geometric, octree, and mesh models, as indicated in Table 15.1.

To support the mesh generation requirements for the entire range of engineering analyses, the models must be non-manifold models [8,29], in which the entities and their adjacencies, in terms of which entities bound each other, are defined for general combinations of regions, faces, edges, and vertices.

The association of topological entities of the mesh with respect to the geometric model is referred to as classification, in which the mesh topological entities are classified with respect to the geometric model topological entities upon which they lie.

Definition: Classification *— The unique association of mesh topological entities of dimension* d_i, $M_i^{d_i}$ *to the topological entity of the geometric model of dimension* d_j, $G_j^{d_j}$ *where* $d_i \leq d_j$, *on which it lies is termed classification and is denoted* $M_i^{d_i} \sqsubset G_j^{d_j}$ *where the classification symbol,* $\sqsubset$, *indicates that the left-hand entity, or set, is classified on the right-hand entity.*

In specific implementations it is possible to employ the classification of the mesh entities against octree entities. Octree entities can cover portions of more than one model entity; therefore the use of classification of octant entities against model is not possible for all octant entities. However, understanding the relationship of the octant to the model is important to track during the tree and mesh construction processes. One device used to aid in the process of understanding the relationship of the closure* of the octant, $\overline{O}_j^3$, with respect to the geometric model is to assign each octant a type. The four octant types indicate if $\overline{O}_j^3$ is inside the geometric model region, $T(O_j^3) = in(G_i^3)$ where G_i^3 is the model region the octant and all its bounding entities are classified within, outside the domain, $T(O_j^3) = out$, contains a portion of the model boundary, $T(O_j^3) = bdry$, or its status is not yet determined $T(O_j^3) = unk$.

*The closure of an octant includes the octant, its 6 faces, 12 edges, and 8 vertices. Although it possibly can define an octant's relationship either with respect to the entity or its closure, the specific choice made influences the details of the various algorithms that carry out algorithmic steps based on the octant status.

15.4.2 Mesh Control and Octant Sizes

Since the edges of the terminal octants will become the edges of the elements in the grid, the size of the octants is dictated by the mesh control information applied. For a given root octant, the size of a terminal octant is controlled by its level in the octree; therefore, the sizes of the elements are controlled by specifying octant root size and levels throughout the object being meshed. Since the octree is, at least initially, spatially addressable, any mesh control function that can indicate the element size in a particular location in space can be used.

Although general functions to define element sizes as a function of position have application, alternative methods to specify mesh control tend to be easier to use. For *a priori* mesh size specification, users of automatic mesh generators find it advantageous to associate mesh size parameters with the topological entities of the model. For example, to indicate the maximum element size associated with an edge, vertex, face, or region. Users also like to be able to control the mesh size based on the local curvature of the model faces. *A posteriori* mesh size specification as defined by an adaptive procedure, which typically associates a desired element size with elements in the mesh of the previous steps. In an octree mesh generator, there is some advantage to associating this information directly with the octree octants to define the level variation.

In most octree mesh generators, the final octant size at a location is equal to or less than that indicated by the mesh control parameters. The octant size at a location can be forced to be less than requested by the mesh control parameters when the octant is subdivided to satisfy the commonly applied one-level difference rule. The one-level difference rule [31] (also known as the 2:1 rule [10]) is commonly used in octree-based meshing procedures to control mesh gradations and element aspect ratios. This rule forces octants that share an edge to have no more than a one-level difference. (This forces the maximum difference for octants that share only a corner to two levels.)

15.4.3 Definition of Octree

The first step in the construction of the octree is to define the size and position of the root octant, O_1^3, typically referred to as the universe. The object must be contained within the closure of the universe. If the domain has a polygonal representation, the minimum limits of the root can be easily defined in terms of the extreme coordinate components of the model vertices. However, if the model is curved, the extreme coordinate values have to be determined using more complex algorithms, which typically have some known degree of approximation error. In these cases, the conservative approach is to expand the coordinates defining the universe by some amount greater than the possible approximation error to ensure $\overline{O_1^3} \cup \overline{G} = \overline{G}$, where $\overline{G}$ is the closure of the model. Note that $T(O_1^3) = bdry$.

A number of alternative approaches have been proposed to decompose the root octant into the final octants that will be meshed [10,12,18,22]. Most of these rely on a recursive subdivision of a given parent octant into its children until the children are of the desired size as defined by the local mesh control information. Given a function that indicates the smallest element size desired within an octant, it is a simple process to examine the size of the current octant, and to subdivide it if it has not yet been refined to a sufficient level. The more critical issues of octant refinement are associated with determining, and representing, the interactions of octants with the portion of the geometric domain that are fully or partly contained within it, particularly in the case when these operations are performed directly with respect to the solid model representation.

The minimum information requirement during octant refinement is the octant type for each child. Since this understanding is gained by qualifying the interactions of the children octants with the model entities that interacted with the child's parent, the process of octree creation focuses on the most effective means to determine these interactions. In the case when the mesh control parameters are associated with the model's topological entities, determining which model entities interact with the octant is central to determining if a given octant is to be subdivided further. Octants can also be forced to subdivide simply due to the complexity of the portion of the geometric model within them because of limitations of specific octant triangulation procedures used to handle that level of complexity.

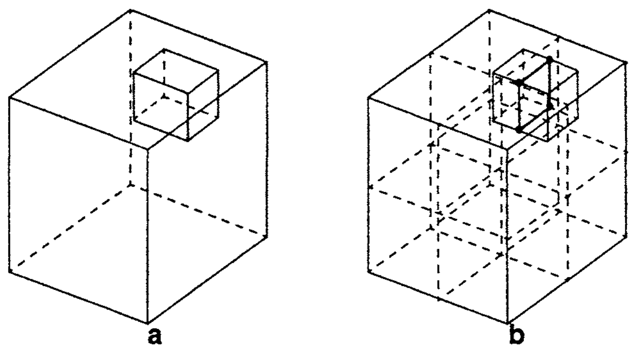


FIGURE 15.5 Octant subdivision and determination of model/octant interactions.

If an octant to be subdivided is inside, $S(O_j^3) = in(G_3^i)$, or outside, $S(O_j^3) = out$, each of the eight children receives the same octant type. The octants that contain portions of the model boundary, $S(O_j^3) = bdry$, or possibly contain portions of the model boundary, $S(O_j^3) = unk$, require execution of geometric operations to determine which of those entities are associated with each child octant, so that the octant's type can be properly set and the proper model entities associated with the children octants. In general the determination of the interactions of the model entities with an octant requires performance of intersections of octant boundary entities with model boundary entities, as well as operations to determine when model entities are entirely contained within an octant. Since these intersection operations can dominate the cost of an octree meshing process, their effective execution to determine the octant type and the specific intersection information needed for further tree refinement and later creation of elements is critical. The reader is referred to Kela [10] for details of a complete and effective procedure for this process.

As a demonstration of the type of operations that would be performed when the full set of interactions between the octant and model entities are desired, consider Figure 15.5a, which shows an octant with a rectangular prism in the upper rear portion of the octant. The model vertices, edges, and faces of this simple model are entirely inside the root octant. Key to determining the relationship of the model with the eight children created by subdivision of the root octant is determining the interactions of the three bisection planes shown in Figure 15.5b. The basic intersection operations performed to determine these interactions are the intersection of the model edges fully or partly within the parent octant with the three planes, and the intersections of the edges of the planes and the edges defined by the intersections of the three planes, with the model faces contained fully or partly in the original octant. In the particular example shown in Figure 15.5a, the result of these operations determines four intersections of the edges of the model with one bisection of the planes. The resulting intersection vertices, shown as darkened vertices in Figure 15.5b, are used in an edge and loop building algorithm to create the darkened edges that complete the qualification of the model information in the children octants and are used as edges and vertices in the finite element mesh. The result of subdividing the original boundary octant yields six children octants that are outside, and two that are boundary octants. Note that only performing intersections with the bisection planes is not sufficient to properly qualify the children octants in all cases. Information on portions of model entities associated with the original octant that do not interact with the bisection planes has to be transferred to the appropriate children. Information on the interactions of the model entities with octant entities, and model entity bounding boxes, allows this information to be determined quickly in most cases. When the results of these operations are inconclusive, more costly geometric operations are required [10].

In some octree-based mesh generators, the interaction that can be represented between the octant and model is more limited. For example, a procedure may allow interaction which can be adequately approximated by the diagonals between octant corners with only one model face cut per octant. If the model complexity at the requested octant level is too great to be properly approximated in the prescribed manner, the octant must be subdivided further until the number and complexity of model entities within the octant can be represented. This process does introduce refinement past what was requested. In addition, it is always possible to devise situations, particularly on nonmanifold models, where the topological complexity at the boundary of a particular model entity is such that no level of refinement will allow a topologically correct approximation of the situation when there are preset limits on the model topological complexity allowed within an octant.

15.4.4 Information Stored in the Tree

As the octree representation for a geometric domain is constructed, information about the interactions of the geometric model and the octants is associated with the octant in preparation for the creation of the elements in the next step. The amount of information stored is a strong function of the type of model/octree interaction information used to create the elements inside the octants.

Once an octant has been given the type *outside*, no additional information need be stored with it. In the case of octants inside a model region, the basic information stored with the octant is a pointer to the model region it is inside of, and information on the local element sizes, or at least the means to obtain that information through the region pointer.

Boundary octants carry additional information which aids in qualifying the interactions of the octant with the boundary of the domain. The specific model information stored is a function of what is needed to control octant subdivision and by the element creation procedures. In the simplest of cases where the analysis procedure will use the entire octant geometry and only account for a volume fraction correction, the information can be limited to a knowledge of the model boundary entities interacting with the octant, as is sufficient to calculate the volume fraction and control further octant subdivision.

Since there are no *a priori* limits on the number of model entities interacting with an octant, general octree mesh generation algorithms employ a more complete representation of the interactions of the model and the octant. The approach used to do this employs a localized boundary representation consisting of the entities defined by the intersection of the model and octant entities. As octants are subdivided, the octant level topological information is updated to indicate the information that is associated with the children octants and the new entities created by the intersection of the model entities with the new octant entities.

As a more explicit example of the information that may be stored in an octant [22], consider the boundary octant shown in Figure 15.6, where most of the octant is interior to a model region and one corner is exterior to the domain due to a reentrant corner in the geometric model. Since the octant level information stored will be used to drive the octant meshing process, the specific entities defined at the octant level will consist of mesh vertices, mesh edges, and octant level loops which are classified against the original model. Figure 15.6 shows the visible mesh vertices and mesh edges for our example.

Visible mesh vertices M_i^0 through M_6^0 are classified on octant vertices, $M_i^0 \sqsubset O_j^0$ and interior to a model region, $M_i^0 \sqsubset G_k^3$. M_8^0, M_{10}^0, and M_{12}^0 are classified on octant edges, $M_i^0 \sqsubset O_k^1$, and model faces, $M_i^0 \sqsubset G_k^2$. M_7^0, M_9^0 and M_{11}^0 are classified on octant faces, $M_i^0 \sqsubset O_k^2$, and model edges, $M_i^0 \sqsubset G_k^1$. M_{13}^0 is classified in the octant interior, $M_{13}^0 \sqsubset O_k^3$, and a model vertex $M_{13}^0 \sqsubset G_k^0$. The one invisible mesh vertex is classified on an octant vertex, $M_i^0 \sqsubset O_j^0$, and interior to a model region, $M_i^0 \sqsubset G_k^3$.

Visible mesh edges M_1^1 through M_6^1 are classified on octant edges, which they span, and interior to a model region $M_i^1 \sqsubset G_k^3$. M_7^1 through M_9^1 are classified on the octant edges, which they partly span, and interior to a model region $M_i^1 \sqsubset G_k^3$. M_{10}^1 through M_{15}^1 are classified on octant faces, and on model face $M_i^1 \sqsubset G_k^2$. M_{16}^1 through M_{18}^1 are classified in an octant region, and on model edge $M_i^1 \sqsubset G_k^1$. There are three invisible mesh edges which are classified on octant edges, which they span, and interior to a model region $M_i^1 \sqsubset G_k^3$.

There are six visible loops of mesh edges in the example octant. The mesh edge loops $M_4^1 - M_5^1 - M_9^1 - M_{14}^1 - M_{15}^1 - M_5^1$, $M_1^1 - M_7^1 - M_{10}^1 - M_{11}^1 - M_8^1 - M_2^1$ and $M_3^1 - M_8^1 - M_{12}^1 - M_{13} - M_9 - M_6$ are classified on octant faces, $M_i^0 \sqsubset O_k^2$, with four of the edges interior to a model region $M_i^1 \sqsubset G_k^3$, and two on model faces. The mesh edge loops $M_{14}^1 - M_{13}^1 - M_{18}^1 - M_{16}^1$, $M_{11}^1 - M_{17}^1 - M_{18}^1 - M_{12}^1$ and

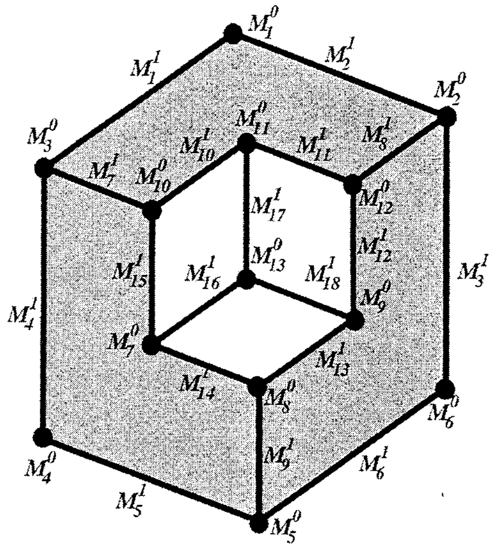


FIGURE 15.6 Information stored at the octant level.

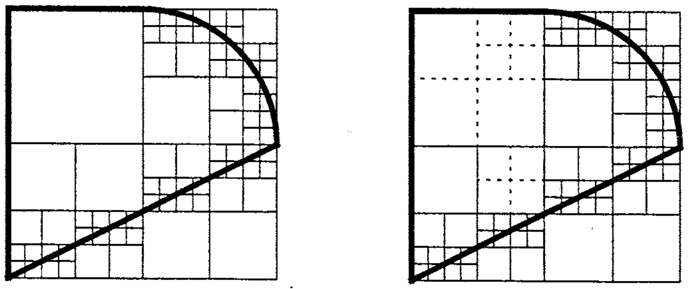


FIGURE 15.7 Quadtree example before (left) and after (right) one-level difference enforcement.

$M_{10}^1 - M_{15}^1 - M_{16}^1 - M_{17}^1$ are classified in the octant interior, $M_{13}^0 \sqsubset O_k^3$, and on model faces, $M_i^1 \sqsubset G_k^2$. The three invisible loops each have four mesh edges that correspond to the four octant edges that bound the octant face. They are classified interior to a model region $M_i^1 \sqsubset G_k^3$.

As a last step before generating the mesh within the octants, most octree-based mesh generators will enforce a one-level difference between octants sharing edges and neighbors. This process helps control element gradations and shapes, and makes the meshing of interior octants easier. Figure 15.7 demonstrates this for a two-dimensional quadtree case. The left image shows a tree before the application of a one-level difference operation, while the right image shows the tree with the additional quadrant refinements (dashed lines) required for one-level difference between edge neighbors. The determination of the tree cells needing refinement is easily determined using tree traversal [31]. It should be noted that when this process forces boundary cells to be refined, the process of determining the appropriate boundary interaction must be carried out with respect to the refined cells.

15.5 Mesh Generation Within the Tree Cells

15.5.1 Meshing Interior Cells

It is common to take specific advantage of the simple geometric shape of the interior cells when creating the elements within those cells. In some cases, the interior octants are treated as individual hexahedral elements. If the tree level through the domain is uniform, the use of one hexahedron per interior octant is possible without further consideration. In the case where there are level differences between neighboring octants, it becomes necessary to account for the fact that the faces of neighboring hexahedra across level differences will not be conforming. For example, in the case of a one-level difference, the one face of the hexahedron will be covered by four quadrilateral faces of the lower level neighbors. These situations can be addressed by the imposition of appropriate multipoint constraint equations. The tree structure can be effectively used to determine the neighboring information needed to construct these constraints.

It is possible to construct conforming meshes that will account for the level differences when tetrahedral elements are used. Again, the tree structure is used to determine the required neighboring information. In some implementations, template structures have been devised to mesh most or all of the internal octants. The simple six-pyramid procedure [31] is easy to implement, but yields more than the desired number of elements in the cases of level differences. More elaborate schemes that maintain the minimum number of elements are possible [18].

Template procedures for interior octants which produce conforming Delaunay meshes have also been developed [18]. By using a slightly reduced circumsphere concept the Delaunay triangulation for an octant, which has all eight vertices on the same circumsphere, becomes uniquely defined by the order in which points are inserted during octant Delaunay point insertion. Combining the ability to control the triangulation, by the order of point insertion, coupled with the knowledge of the octants neighbors available from the tree structure, allows the automatic construction of octant template codes for the interior octants. These procedures can account for neighbors with a level difference. Note that interior octants neighboring boundary octants with non-corner mesh vertices near the interior octants will require the overriding of the template defining the interior octant triangulations to regain a globally Delaunay triangulation. The triangulation process in this case must consider information from neighboring octants.

15.5.2 Meshing Boundary Cells

The process of meshing the boundary cells is a strong function of the level of geometric complexity supported by the mesh generator. In cases where there is only a limited amount of geometric complexity allowed per octant, simple templates are possible. When there is no specific limitation on the level of geometric complexity allowed within the octant, the process of meshing the boundary octant requires all the functionality of an automatic mesh generator applied to the local region [12,19,22].

To demonstrate the issues and options associated with meshing boundary octants, the basics of four different approaches will be considered for the creation of elements in the boundary octants. The first two create tetrahedral elements assuming that the surface has not been pre-triangulated. The first of these approaches applies an element removal procedure starting from a basic octant level boundary representation as outlined in the previous section. The second approach develops a Delaunay triangulation based on the mesh vertices of the octant level boundary representation, which is then followed by an assurance algorithm that insures the resulting surface triangulation is topologically compatible and geometrically similar. Since the first two procedures operate strictly accounting for the intersections of the model and octant boundary entities, they are susceptible to the small, poorly shaped elements caused by boundary octants nicking the model boundary.

The third procedure creates tetrahedral elements from a given surface triangulation using an element removal procedure. The last boundary octant meshing procedure considers the creation of hexahedral elements to fill the region between the interior octants and the model boundary. These two procedures create the elements in the regions between the model boundary and interior octants without strict adherence to the boundary octant's boundary. Therefore, they are not susceptible to the creation of poorly shaped elements caused by the boundary octants nicking the model boundary.

15.5.2.1 Element Removal to Mesh Boundary Octants

One approach to generate meshes in the boundary octants is to apply a general set of element removal operations to the local octant boundary representation developed during the octree creation process. In this approach the only interaction with neighboring octants which must be taken into account is to copy the surface triangulations of any common neighboring interior or boundary octant's faces that already have been triangulated.

The most general procedure for the creation of elements in the boundary octants is to apply the three-element removal operators of vertex removal, edge removal, and face removal [22,30], working from the boundary representation defined in terms of octant face loops. These removal operators are capable of creating the surface triangulation on octant face loops that have not yet been triangulated, while matching existing triangulation for those that have been previously triangulated. Preference is given to the application of the vertex removal and edge removal operations since they do not create any new mesh vertices. However, situations can arise where face removal must be applied.

To demonstrate the application of element removal on the boundary octant, the process of meshing the boundary octant of Figure 15.6 with the octant faces already triangulated (Figure 15.8, upper-left image) is considered. The first three tetrahedral elements are created by the removal of mesh vertices M_{10}^0, M_8^0, and M_{12}^0. The upper-right image of Figure 15.8 shows the octant after the three vertex removals. The next three elements are created by edge removal of edges $M_7^0 - M_{11}^0$, $M_{11}^0 - M_9^0$, and $M_9^0 - M_7^0$. The lower-left image of Figure 15.8 shows the octant after the application of the three edge removals. The next six elements are created by the application of three edge removals and three vertex removals. For example, the three edge removals could be $M_7^0 - M_3^0$, $M_5^0 - M_9^0$, and $M_2^0 - M_{11}^0$. This process creates edges $M_4^0 - M_{13}^0$, $M_6^0 - M_{13}^0$, and $M_1^0 - M_{13}^0$, thus allowing the application of vertex removal at vertices M_7^0, M_9^0, and M_{11}^0. The lower-right image of Figure 15.8 shows the octant after the removal of these six elements. The last six elements are created by one edge removal and five vertex removals. For example, if edge $M_4^0 - M_5^0$ is removed, edge $M_6^0 - M_{13}^0$ is created, thus allowing vertex removal at vertex M_5^0. The last four vertex removals are then applied to vertices M_6^0, M_2^0, M_1^0, and M_3^0 in order.

15.5.2.2 Delaunay Point Insertion to Mesh Boundary Octants

An alternative approach to meshing boundary octants has been used in an octree-Delaunay mesh generation procedure [18]. In this procedure each complete boundary octant is first meshed without consideration of the model boundary, using the same procedure that produces compatible triangulations for the interior octants. Assuming that the surface has not already been pre-triangulated, the remaining steps in meshing the boundary octant in this procedure include the following:

1. Insert the mesh vertices necessary to account for the interaction of the model boundary with the octant.
2. Perform topological compatibility and geometrically similarities of the octant level mesh edges and faces classified in the model's boundary to ensure a valid geometric triangulation of the octant [19,23].
3. Eliminate all tetrahedra exterior to the model.

The vertices inserted in the first step are defined by (1) model vertices within the octant, (2) the intersection of model edges with the octant faces, and (3) the intersection of the octant edges with the model faces. The creation of a globally Delaunay triangulation as these points are inserted requires consideration of the triangulation of, at a minimum, those octants the mesh vertex being inserted bounds. In addition, when the mesh vertices are close to other octants, their triangulation may also need to be considered during the vertex insertion process. Specific methods to know which octants must be considered have been developed [18].

A generalized topological compatibility and geometric similarity algorithm [19,23] must be applied after the points have been inserted. In some cases it is not possible with the given set of points to recover a valid geometric triangulation which satisfies the Delaunay empty circumsphere requirement. In these cases, additional points can be generated using octree subdivision or specific point insertion processes [14,18]. After a valid boundary triangulation has been constructed, it is a simple task to complete the boundary octant triangulation process by deleting those elements outside the domain of the object.

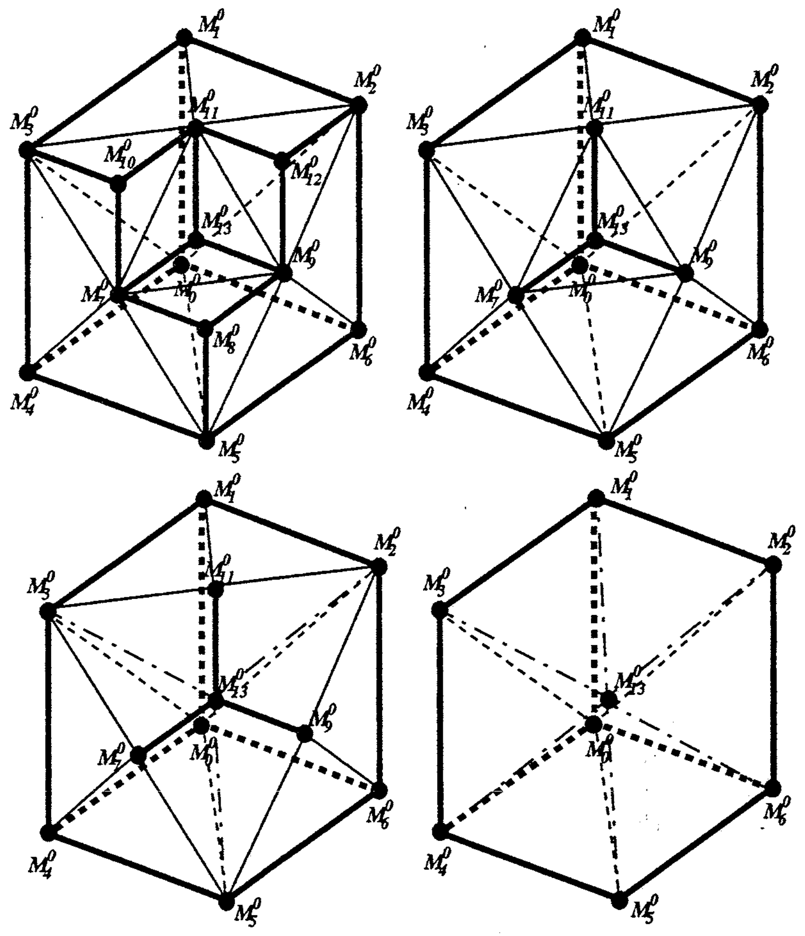


FIGURE 15.8 Mesh generation in a boundary octant by element removal.

15.5.2.3 Element Removal from a Pre-Triangulated Surface to Interior Octants

In this octree mesh generator the tetrahedral mesh is created from a pre-triangulated surface mesh [6]. The octree for this procedure is created such that the octants containing the surface triangles are sized to have edge lengths equivalent to that of the edges of the surface triangulation that is partly or completely interior to them. The interior octants are created such that they satisfy the one-level difference rule. To avoid the poorly shaped elements caused by close interaction of surface triangles and interior octants, the additional cell type of boundary-like interior cells is introduced. These are interior cells that are closer than some fraction of the surface triangle edge length to the surface triangulation. Using one half an edge length of the near-by surface triangle as the distance criterion works well for this purpose.

Figure 15.9 demonstrates the application of this process to a simple two-dimensional domain. The left image shows the set of domain boundary segments and the quadrants generated based on them. The image shows the boundary quadrants that contain portions of the boundary segments, the interior quadrants that are more than half an edge length from the boundary segments, and the boundary-like interior quadrants that are interior octants within one half an edge length of the boundary segments. The interior cells are meshed using templates and are indicated by the shaded triangles in the right image of Figure 15.9.

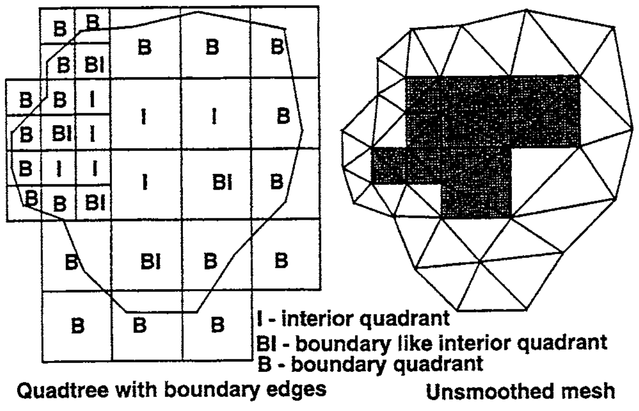

B
BI
I
I - interior quadrant
BI - boundary like interior quadrant
B - boundary quadrant
Quadtree with boundary edges
Unsmoothed mesh

FIGURE 15.9 Mesh generation given a discretized boundary

After the interior octants are meshed, the remaining portion of the domain to be meshed is that region lying between the outer faces of the meshed interior octants and the surface triangulation. This region is meshed employing element removal operations in a manner similar to that used in the current advancing front mesh generators.

The description of the procedure to mesh the remaining portion of the domain focuses on the mesh faces defining the surface triangulation and those on the exterior of the interior octants. Since the completion of the meshing process requires connecting tetrahedra to one or, in multiregion problems, up to two sides of these faces, these faces are referred to as partly connected faces. The mesh generation process is complete when all mesh faces are fully connected.

The element creation process to connect partly-connected faces is not constrained to following cell boundaries. It is guided solely by the creation of elements to fill the region between partly connected faces. This is depicted in the right image of Figure 15.9 by the unshaded triangles created during this step. The tree structure is used during this process to efficiently locate neighbor information. Each partly connected face is associated with one or more octants, thus allowing the tree neighbor-finding procedure to be used to locate neighboring partly connected faces that a current face can be connected to.

Given a partly connected mesh face, the face removal consists of connecting it to a mesh vertex of a nearby partly connected face. Since the volume to be meshed consists of the region between the given surface triangulation and the interior octree, the vertex used is usually an existing one. In some situations it is desirable to create a new vertex. The choice of this vertex must be such that the created element is of good quality and its creation does not lead to poor (in terms of shape) subsequent face removals in that neighborhood.

Early element removal procedures had some difficulty in the process of determining the vertex to connect to, in that the criteria used emphasized the element being created with little consideration for the situation remaining for subsequent face removals. Consideration of the influence on subsequent face removals is a difficult process since one does not know about them until they arise. One possible solution is to make sure that any element creation does not make new mesh entities too close (relatively) to existing mesh entities. This process requires an exhaustive set of geometric checks against mesh entities in the neighborhood. Although it is possible to develop the appropriate set of checks, it is in general an expensive process since the number and complexity of checks required is quite high even when efficient procedures are used to provide a proper set of candidate mesh entities to consider. An alternative method is to use a more efficient criterion that indirectly accounts for the various situations that can arise. The Delaunay circumsphere criteria does provide a quality mesh when given a well-distributed set of points that avoids the creation of flat elements. The use of Delaunay criteria in general element removal mesh generation procedures has been shown to be an effective means to control this process.

One procedure [6] combines the use of the octree, Delaunay meshing criteria, and more exhaustive checks when a local Delaunay solution is not available. Starting with the mesh vertices closest to the face to be removed, the Delaunay circumsphere test is performed. Since the Delaunay mesh for a given set of points is unique to within the degeneracy of more than four points on the circumsphere, the first vertex which satisfies this criteria is used to create the element. If there are degeneracies, consideration must be given to the other points on the circumsphere to ensure a proper selection is made. If none of the candidate vertices satisfy the Delaunay criteria for that face, a more exhaustive checking procedure is undertaken which explicitly considers the shapes of the element created as well as shapes of future elements dictated by other nearby connections.

Face removals are performed in waves. A new partly connected mesh face resulting from some face removal is not processed until all other partly connected mesh faces existing at the beginning of that wave are processed. Also, partly connected mesh faces resulting from the meshing of interior terminal octants are never processed for face removals so long as other partly connected faces exist. This gives priority to partly connected mesh faces coming from the model boundary. The process of removing partly connected mesh faces ends when there are no more partly connected mesh faces.

15.5.2.4 Hexahedral Element Creation from the Interior Octants to the Model Boundary

A technique for the generation of hexahedral elements for the boundary octants has also been proposed [17]. The implementation discussed here requires the use of a uniform tree level throughout the domain. Octants more than one-half element length away from the model boundary are defined as interior octants and meshed with a single hexahedra. The region from those interior octants to the model surface is then meshed using a projection method.

The basic idea is to define one or more projection lines from each of the vertices on the outer surface of the interior octree to the outer surface of the model. The square faces on the outer surface of the interior octants and the projection lines are used to define hexahedra. The number and direction of projectors defined from a vertex on the surface of the interior octree depends on which of the eight octants the vertex bounds are interior octants.

In the case when the exterior of the object is a single smooth closed face, it is reasonably straightforward to use the default projectors and directions to define a set of hexahedra in the volume between the interior octants and boundary. The existence of model edges and vertices will force decisions to be made to alter the direction used for the projectors so that those edges and vertices are properly represented. In cases where they can be represented using the default numbers of projectors, it is possible to define elements that are topologically hexahedral in the volume from the interior octree to model surface. The problem that arises is that often some number of those hexahedra have unacceptable shapes in that some element angles are in the invalid range. Specific subdivision techniques can be used to produce a valid element at the cost of local increases in the number of elements. In addition to the geometric complexities introduced by the model edges and vertices, there can be configurations of edge and vertex interaction that will not always produce topological hexahedral polyhedra using the default projection edges.

15.6 Mesh Finalization Processes

Some analysis procedures take specific advantage of the regular shape, square in two dimensions and cube in three dimensions, of the tree cells. In other cases, particularly when confirming meshes of triangular and tetrahedral are generated, the analysis calculation does not require that the elements stay strictly aligned with the cell boundaries. In these cases, it is possible to apply procedures to improve the shapes and gradation of elements.

The most commonly, and easily, applied operation for such element shape improvements is to reposition node point locations. Although node point repositioning can lead to substantial improvements in element shape measures, there are many cases where the constraints of the neighboring elements are such that the element shape remains poor. The inclusion of various local mesh modifications can yield more dramatic improvements in the mesh. In addition, the application of a full set of mesh modification operators can be used to eliminate the adverse effects of the small elements caused when the model boundary and octant boundaries are close.

15.6.1 Node Point Repositioning

The application of node point repositioning within a quadtree or octree mesh generator can follow the normal process of iteratively repositioning one node at a time based on a specific repositioning criteria using the mesh connectivity information. It is also possible to use information based on the tree structure to define alternative connectivities.

Although any of the standard criteria for node point positioning can be applied, it is advisable to only apply criteria which ensures that the elements will remain valid. The application of Laplacian smoothing often yields good element shape improvements, but should only be applied in a constrained manner such that a node is allowed to move only if the shape of the worst shape element connected to it improves.

One approach that has worked well in an octree mesh generator is to employ a combination of two smoothing operators. The first operator applied is a constrained Laplacian operator where the standard average of all connected nodes is used as the target for the node point. If this location is found to yield improvement in the shape of the worst-shaped element connected to the node, the node is moved to that position. If that location does not yield improvement in the worst-shaped element, locations on the line from the current position to the centroid are checked and the first that yields improvement is selected.

The overall result of the Laplacian smoothing step is general improvement in the overall quality of the elements and reasonable improvements in the mesh gradation. However, a small number of the most poorly shaped elements are not improved, since the direction of motion defined by the centroid only degrades the shape of this element.

The second smoothing operator focuses its attention on the small number of poorly shaped elements. Several approaches that specifically focus on improving the shape of the worst shaped element connected to a node are possible. One reasonably efficient means to this is to employ a line search approach where the direction of motion is selected to ensure improvement in the shape of the worst element connected to the vertex. Given a node and the worst shaped element connected to it, it is possible to determine the position of the node which will optimize the shape of that element [5].

Moving the node all the way to the optimum position of the initially worst-shaped element can degenerate the shapes of other elements connected to the node being moved. Therefore, care must be taken to move the node to a location which does improve the shape of the current worst shape element, but limits the degradation of any other connected element such that its current shape and the shape of the starting connected worst-shaped-element are equivalent. Using the vector from the original model position to the optimum location for the worst shape element, it can be shown that the worst-shaped of any of the connected elements defines a precise function with a unique minimum that can be effectively determined using an efficient golden section search procedure [5].

A important aspect of node point smoothing in quadtree and octree mesh generators is to apply smoothing to all nodes classified on the boundary of the domain, as well as interior nodes near the model boundary. The procedures to smooth nodes classified on nodal edges and faces must consider the resulting shapes of all connected elements, while being constrained to stay on the model edge or face the node is classified on.

15.6.2 Elimination of Poorly Sized and Shaped Elements Caused by Interactions of the Object Boundary and the Tree

One undesirable feature of octree based mesh generators is the disproportionately small and poorly shaped elements that can arise when the boundary of the model comes close to the boundary of an octant. As indicated in Figure 15.4, for the two-dimensional case, it is possible to deform the octree cells so the model and octree boundaries yield elements of the desired size. In the two-dimensional case it is

a reasonably straig
interactions are de

The complexity
makes the immedi
An alternative app
process without oc
to eliminate the ap

To successfully m
must include delet
improve the eleme
two-dimensional m
short edge on the

15.6.3 Three-

The set of generali

1. Edge and fa
2. Edge, face, a
3. Edge collaps

The edge collapse o
elements caused b
characterized by th
by the local mesh c
these and their infl
situations for curve
invalid mesh in tha
will produce a mes
measures are impro

The application
elements must first
general mesh mod
improve the quality
mesh modifications
the maximum dihe

15.6.4 A Cou

The first example (
alignment of the o
the use of octree n
respect to the mesh
mesh generated. H
how the mesh is ge
mesh was generated
and tilted at 30° t
(Figures 15.10a an
octree cells. Altho
(Figure 15.10a) is s
of mesh quality in
greater than 179.9

a reasonably straightforward process to perform the quadrant distortion as the model boundary quadrant interactions are determined.

The complexity of the possible model boundary/octant boundaries in the three-dimensional case makes the immediate distortion of the octants as the interactions are determined much more difficult. An alternative approach to meet the same goal is to carry out the octree creation and mesh generator process without octant distortion and to then apply an appropriate set of mesh modification operators to eliminate the appropriate entities.

To successfully meet the goal of eliminating the adverse effects of the small features, the set of operators must include deletion and splitting operators in addition to the swapping operator commonly used to improve the element shapes. As a specific example of the usefulness of a deletion operator, consider the two-dimensional mesh shown in the lower part of Figure 15.4a. A collapse operator that eliminates the short edge on the boundary of the model yields the mesh shown in Figure 15.4b.

15.6.3 Three-Dimensional Mesh Modifications to Improve Mesh Quality

The set of generalized three-dimensional mesh modification operators includes

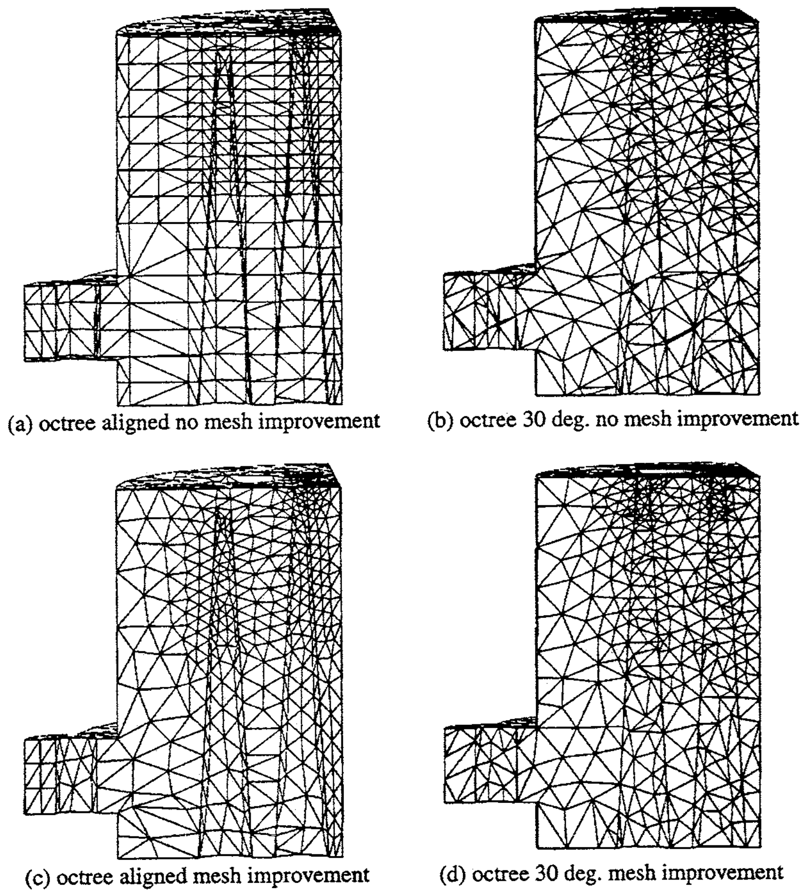
1. Edge and face swaps [4,7].
2. Edge, face, and region split operators [4].
3. Edge collapse operators [4,25].

The edge collapse operator is a key tool in the elimination of disproportionally small and poorly shaped elements caused by close model/octant boundary interactions. The majority of these situations are characterized by the existence of one or more mesh edges that are substantially shorter than that dictated by the local mesh control information. Once detected, edge collapse operations can be applied to eliminate these and their influence on the mesh. Although the majority of these edges can be collapsed, there are situations for curved geometric domains where the direct application of an edge collapse would yield an invalid mesh in that vicinity. In these cases the application of various swapping and splitting operations will produce a mesh configuration where an edge collapse can be applied, or the local mesh quality measures are improved.

The application of mesh modification operators with the goal of locally improving the quality of the elements must first decide the mesh quality measure to be concerned with. Since the application of the general mesh modification operators is reasonably expensive, these are often applied in an attempt to improve the quality of only elements that have shapes worse than some specific limit. In the case when mesh modifications are applied after disproportionately small mesh edges have been applied, reducing the maximum dihedral angles below some upper limit works well.

15.6.4 A Couple of Examples

The first example (Figure 15.10) demonstrates the influence of the mesh finalization procedures and the alignment of the octree with respect to the model. One concern often expressed in the literature about the use of octree mesh generation techniques is the influence of the alignment of the octree has with respect to the mesh generated for a model. In general, the alignment of the octree does affect the final mesh generated. However, the degree and importance of the influence depends strongly on details of how the mesh is generated within the octree and the level of mesh finalization used. In this example, the mesh was generated with the octree aligned with the vertical axis of the model (Figures 15.10a and 15.10c) and tilted at 30° to the vertical axis (Figures 15.10b and 15.10d) of the model. In the top images (Figures 15.10a and 15.10b) the mesh was generated strictly following the interactions of the model and octree cells. Although the meshes generated look very different, and one may feel the aligned mesh (Figure 15.10a) is superior to the one tilted at 30° (Figure 15.10b): both meshes lack acceptable control of mesh quality in that the maximum dihedral angle of the worst element in the mesh in both cases is greater than 179.9°. In contrast, the application of a full set of mesh finalization operations which


(a) octree aligned no mesh improvement (b) octree 30 deg. no mesh improvement

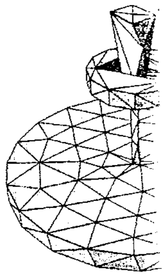
(c) octree aligned mesh improvement (d) octree 30 deg. mesh improvement

FIGURE 15.10 An example problem with different octree alignments, with and without mesh finalization applied.

reposition nodes, and modify the mesh to eliminate small segments caused by nicks and elements with large dihedral angles (Figures 15.10c and 15.10d), yields meshes that do not show any clear indication of the influence of the alignment of the octree with respect to the model. The quality of the element shapes produced is also similar, with the maximum dihedral angles of any elements in the meshes for the two cases being the same to three significant figures (145°) in this particular example.

The ability of the mesh finalization procedures to produce the control mesh entity shape quality is a strong function of the set of mesh finalization tools employed. If only the smoothing procedures described in Section 15.6.1 are applied to the meshes in Figures 15.10a and 15.10b, the resulting meshes would look much the same as those in Figures 15.10c and 15.10d. However, the quality of the worst-shaped elements would not be the same as when the full set of mesh finalization procedures are applied. In this example, smoothing would reduce the largest dihedral angle down to 169° and increase the smallest dihedral angle to only 1.23°. Including the mesh modifications to eliminate the small nicks does not eliminate the largest dihedral angle, but does increase the minimum angle to 6.7°. Finally, inclusion of the mesh modification operations to decrease the dihedral angle of elements with too large of a dihedral angle reduces the maximum dihedral angle to 145° and increases the smallest dihedral angle in the mesh to 11°.

Octree-based mesh generators easily support the application of any spatially based isotropic mesh size control functions. Figure 15.11 demonstrates the application of mesh controls associated with the entities of the geometric model. Figure 15.11a shows a uniform mesh in which the element sizes requested were the same throughout the model. In the mesh shown in Figure 15.11b, the element size requested for two


(a) unifor

cylindrical and conical fa
mesh shown in Figure 15.
faces based on the local c

15.7 Closing R

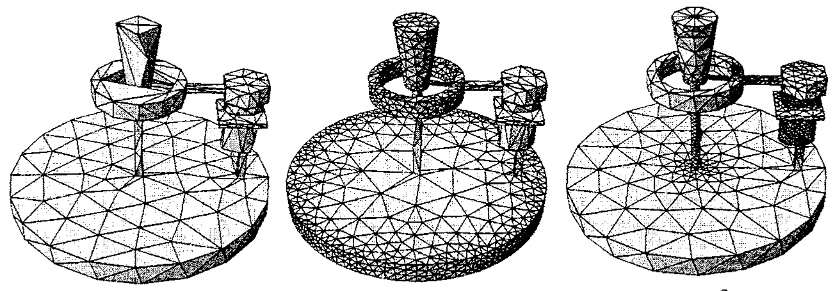
Spatially based tree struc
have been employed in t
the literature shows that
in the process of element
basic issues as other me
elements, and they must
meshed.

All octree and quadtr
neighbor information. M
efficiently create element

Some procedures crea
In most cases these proc
such procedures are com
the adverse influence of

References

1. **Baehmann, P. L.**
element analysis,
2. **Baehmann, P. L.,**
rically based, aut
pp. 1043–1078.
3. **Buratynski, E. K**
Int. J. Num. Met
4. **de Cougny, H. L**
Scientific Comp
NY, 12180-3590,
5. **de Cougny, H. L**
finite octree me
Polytechnic Inst
6. **de Cougny, H. L**
on distributed r



(a) uniform mesh (b) three faces refined (c) curvature refinement

FIGURE 15.11 Application of mesh size controls.

cylindrical and conical faces were set to be smaller than the rest of the model faces (Figure 15.11b). The mesh shown in Figure 15.11c employs a procedure that sets the element sizes associated with the model faces based on the local curvature of the face.

15.7 Closing Remarks

Spatially based tree structures, primarily octrees in three dimensions and quadtrees in two dimensions, have been employed in the development of automatic mesh generation algorithms. An investigation of the literature shows that the alternative procedures developed vary greatly in how the tree is employed in the process of element creation. In all cases, the mesh generation procedures must deal with the same basic issues as other mesh generators. That is, they must employ criteria and procedures to create the elements, and they must ensure that the resulting mesh is a valid representation of the domain being meshed.

All octree and quadtree mesh generators employ the tree structure in the efficient determination of neighbor information. Most procedures also take advantage of the regular shape of interior cells to efficiently create elements in those cells.

Some procedures create meshes which strictly adhere to the boundaries of the undeformed tree cells. In most cases these procedures are taking explicit advantage of this during the analysis process. When such procedures are combined with a general set of mesh finalization procedures, it is possible to eliminate the adverse influence of the shape and alignment of the tree cells on the final mesh.

References

1. **Baehmann, P. L. and Shephard, M. S.**, Adaptive multiple level h-refinement in automated finite element analysis, *Eng. with Computers*, 5(3/4), pp. 235–247, 1989.
2. **Baehmann, P. L., Witchen, S. L., Shephard, M. S., Grice, K.R., and Yerry, M.A.**, Robust, geometrically based, automatic two-dimensional mesh generation, *Int. J. Num. Meth. Eng.*, 1987, 24(6), pp. 1043–1078.
3. **Buratynski, E. K.**, A fully automatic three-dimensional mesh generator for complex geometries, *Int. J. Num. Meth. Eng.*, 30, pp. 931–952, 1990.
4. **de Cougny, H. L. and Shephard, M. S.**, Parallel mesh adaptation by local mesh modification, Scientific Computation Research Center, Report 21-1995, Rensselaer Polytechnic Institute, Troy, NY, 12180-3590, 1995.
5. **de Cougny, H. L., Shephard, M. S. and Georges M.K.**, Explicit node point smoothing within the finite octree mesh generator, Scientific Computation Research Center, Report 10-1990, Rensselaer Polytechnic Institute, Troy, NY, 12180-3590, 1990.
6. **de Cougny, H. L., Shephard, M.S., and Ozturan, C.**, Parallel three-dimensional mesh generation on distributed memory MIMD, *Eng. with Computers*, 12(2), pp. 94–106, 1996.

7. **de l'Isle, E. B. and George, P. L.**, Optimization of tetrahedral meshes, INRIA, Domaine de Voluceau, Rocquencourt, 1993, BP 105, 78153, Le Chesnay Cedex, France.
8. **Gursoz, E. L., Choi, Y., and Prinz, F. B.**, Vertex-based representation of non-manifold boundaries, *Geometric Modeling Product Engineering*, (Eds.) Wozny, M. J., Turner, J. U., Priess, K., North Holland, pp. 107–130, 1990.
9. **Jackins, C. L. and Tanimoto, S. L.**, Octrees and their use in the representation of three-dimensional objects, *Comput. Graphics and Image Processing*, 14, pp. 249–270, 1980.
10. **Kela, A.**, Hierarchical octree approximations for boundary representation-based geometric models, *Computer Aided Design*, 21, pp. 355–362, 1989.
11. **Kela, A., Perucchio, R., and Voelcker, H. B.**, Toward automatic finite element analysis, *Comput. Mech. Eng.*, pp. 57–71, July, 1986.
12. **Perucchio, R., Saxena, M., Kela, A.**, Automatic mesh generation from solid models based on recursive spatial decompositions. *Int. J. Num. Meth. Eng.*, 28, pp. 2469–2501, 1989.
13. **Samet, H.**, *Applications of Spatial Data Structures*, Addison-Wesley, Reading, MA, 1990.
14. **Sapidis, N. and Perucchio, R.**, Combining recursive spatial decompositions and domain delaunay triangulation for meshing arbitrary shaped curved solid models, *Comp. Meth. Appl. Mech. and Eng.*, 108, pp. 281–302, 1993.
15. **Saxena, M. and Perucchio, R.**, Parallel FEM algorithms based on recursive spatial decompositions - {i}.automatic mesh generation, *Computers and Structures*, 45, pp. 817–831, 1992.
16. **Saxena, M., Finnigan, P.M., Graichen, C.M., Hathaway, A.F., and Parthasarathy, V.N.**, Octree-Based Automatic Mesh Generation for Non-Manifold Domains, *Engineering with Computers*, 11, pp. 1–14, 1995.
17. **Schneiders, R. and Bunten, R.**, Automatic mesh generation of hexahedral finite element meshes, *Computer Aided Geometric Design*, 12, pp. 693–707, 1995.
18. **Schroeder, W.J. and Shephard, M. S.**, A combined octree/Delaunay method for fully automatic 3-D mesh generation, *Int. J. Num. Meth. Eng.*, 29, pp. 37–55, 1990.
19. **Schroeder, W. J. and Shephard, M. S.**, On rigorous conditions for automatically generated finite element meshes, *Product Modeling for Computer-Aided Design and Manufacturing*, Turner, J. U., Pegna, J., Wozny, M. J., (Ed.), North-Holland, Amsterdam, 1991, pp. 267–281.
20. **Shephard, M.S.**, Update to: Approaches to the automatic generation and control of finite element meshes, *Applied Mechanics Reviews*, 49(10-2), S5–S14, 1996.
21. **Shephard, M. S. and Finnigan, P. M.**, Toward automatic model generation, *State-of-the-Art Surveys on Computational Mechanics*, 3, A.K. Noor, A.K., Oden, J.T., (Eds.), ASME, pp. 335–366, 1989.
22. **Shephard, M. S. and Georges, M. K.**, Automatic three-dimensional mesh generation by the finite octree technique, *Int. J. Num. Meth. Eng.*, 32, pp. 709–739, 1991.
23. **Shephard, M. S. and Georges, M. K.**, Reliability of automatic 3-D mesh generation, *Comp. Meth. Appl. Mech. and Eng.*, 101, pp. 443–462, 1992.
24. **Shephard, M. S., Baehmann, P. L., and Grice, K. R.**, The versatility of automatic mesh generators based on tree structures and advanced geometric constructs, *Comm. Appl. Num. Meths.*, 4, pp. 145–155, 1988.
25. **Shephard, M. S., Flaherty, J. E., de Cougny, H. L., Bottasso, C. L., and Ozturan, C.**, Parallel automatic mesh generation and adaptive mesh control, *Solving Large Scale Problems in Mechanics: Parallel and Distributed Computer Applications*, Papadrakakis, M., (Ed.), John Wiley & Sons, Chichester, U.K., 1997, pp. 459–493.
26. **Shephard, M. S., Yerry, M. A. , and Baehmann, P. L.**, Automatic mesh generation allowing for efficient *a priori* and *a posteriori* mesh refinements, *Comp. Meth. Appl. Mech. and Eng.*, 55, pp. 161–180, 1986.
27. **Shpitalni, M.**, Switching function based representation — an alternative to quadtree encoding for manufacturing systems, *Annals of the CIRP*, 34, pp. 163–167, 1985.
28. **Shpitalni, M., Bar-Yoseph, P., and Krimberg, Y.**, Finite element mesh generation via switching function representation, *Finite Elements in Analysis and Design*, 5(2), pp. 119–130, 1989.

29. **Weiler, K. J.,** The radial-edge structure: A topological representation for non-manifold geometric boundary representations, *Geometric Modeling for CAD Applications,* Wozny, M.J., McLaughlin, H. W., Encarnacao, J. L., (Eds.), North Holland, pp. 3–36, 1988.
30. **Wordenweber, B.,** Automatic mesh generation, *Computer-Aided Design,* 16(5), pp. 285-291, 1984.
31. **Yerry, M. A. and Shephard, M. S.,** Automatic three-dimensional mesh generation by the modified-octree technique, *Int. J. Num. Meth. Eng.,* pp. 1965–1990, 1984.
32. **Yerry, M. A. and Shephard, M. S.,** Finite element mesh generation based on a modified-quadtree approach, *IEEE Computer Graphics and Applications,* 3(1), pp 36–46, 1983.