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

INTEGRATION OF FINITE ELEMENT MODELING WITH SOLID MODELING THROUGH A DYNAMIC INTERFACE

Abstract. Finite element modeling is dominated by geometric modeling type operations. Therefore, an effective interface to geometric modeling requires access to both the model and the modeling functionality used to created it. This paper discusses the use of a dynamic interface that addresses these needs through the use of boundary data structures and geometric operators.

Introduction. The generation of numerical analysis models, typically finite element models, is an important part of the computer-aided engineering (CAE) process. However, a disproportionately large percentage of the design/analysis process is required to carry out this task with the tools commonly available today. Over the past few years, substantial gains have been make in the development of the algorithmic procedures needed to make this a more automated process. To make effective use of these tools, specific consideration must be given to the proper integration of the component parts. This paper presents a general approach to performing the integration of the geometric modeling with advanced finite element modeling tools.

Three technical areas of importance to the eventual automation of the finite element modeling porcess are: geometric modeling, automatic mesh generation, and adaptive analysis techniques. There is no possibility of automating a geometrically-based procedure like finite element modeling if the geometric modeling procedures do not contain a complete and unique representation of the object to be analyzed. Therefore, the advances in geometric modeling based on solid modeling approaches is a prerequisite to automated finite element modeling. The second functionality needed is the ability to automatically discretize a geometric model into a finite element mesh. As is briefly reviewed in the next section, the recently developed algorithmic approaches to automatic mesh generation are addressing this need. The third area of development, adaptive analysis techniques, are not needed to be able to automatically perform an analysis, however, they are needed if robust automated finite element modeling procedures are to be developed. The goal of adaptive analysis techniques is to automatically improve a finite element discretization until the solution obtained yield results to a prescribed degree of accuracy. The next section also indicated the status of the development of these procedures.

The integration of geometric modeling systems with automated mesh generators is not completely addressed by the passing of a geometry file. Specific geometric modeling functionality is also needed to support the operations carried out by the geometric modeling system. The third section discusses an approach to the integration of geometric modeling and automatic mesh generation that supports these needs.

The fourth section discusses the question of controlling the process of going from the original geometric model to the finite element model. Central to this discussion is the form of data structure needed to support this process and the geometric modeling functionality needed. In particular, consideration is given data structures that will support the evolution of an original geometric model to the idealized geometric model that is to be discretized and then supporting the actual discretization process in a general manner.

Automated Finite Element Modeling Tools. Historically, 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. They have the disadvantage of requiring the domain to be meshed to be partitioned into a set of mappable regions which yields the desired distribution of elements. The complexity of reducing the complex three-dimensional domains available from today's 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 purposes 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.

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 it is to mesh, 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 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 tetrahedronal elements in three dimensions.

The three-dimensional automatic mesh generators that have been developed can be classified as being based on one of the following algorithmic approaches:

1. point placement followed by triangulation [CAVE85], [FEIL85], [FIEL86], [NGUY82].
2. removal of individual subdomains [WOO84], [WORD84].
3. recursive domain subdivision [SLUI82], and
4. spatial decomposition followed by subdomain meshing [SHEP86], [YERR84], [YERR85].

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. (See [SHEP87] for a more complete review of automatic mesh generation.)

A large number of two-dimensional mesh generators based on point placement followed by triangulation have been developed (see [CAVE74], [LEE84], [LO85] for example) using a variety of approaches to place points and triangulate them into elements. The three-dimensional procedures [CAVE85], [FEIL85], [FIEL86], [NGUY82] have followed a similar development path. In each of these algorithms, specific heuristics are employed to place points through the domain. The generation of the mesh using these points can either employ a set of triangulation heuristics, or can employ the mathematical properties of Delaunay triangulations [SIBS78], [WATS81] to develop the meshing algorithm. Although Delaunay properties are ideal for two-dimensional mesh generation, they are not fully satisfactory in the three-dimensional case. Therefore, three-dimensional mesh generators using Delaunay based procedures must be augmented with an appropriate set of heuristics to avoid possible problems [FIEL85], [FIEL86], [SHEP87].

Automatic mesh generators based on subdomain removal operate by removing individual pieces form the domain one at a time until the domain is reduced to one remaining acceptable piece. The majority of the algorithms of this type remove individual elements [SADE80], [SHEP86a], [WOO84], [WORD84], while others remove larger, but 'simple' portions of the domain and then triangulate them using a different procedure [BYKA76], [JOE86]. These procedures typically traverse the boundary of the object applying a set of heuristic operators to identify and then remove portions of the domain one at a time.

Although they have been heavily published, the development of automatic mesh generators based on recursive domain subdivision is a popular approach under consideration by a number of CAD vendors. In these approaches the mesh is created by recursively splitting the domain [SLUI82], until the subdomains represent individual finite elements. A specific set of heuristics and geometric test are used to identify the 'splits' used to subdivide objects.

Mesh generators based on spatial decomposition employ some specific decomposition procedure to decompose, in a controlled manner, the domain into a set of simple cells and then to triangulate the individual cells in a manner such that a valid finite element mesh is generated. The procedures developed to date have relied on quadtree structures in two dimensions [BAEH87], [KELA86], [YERR83], and octree structures in three dimensions [SHEP86], [SHEP86a], [YERR84], [YERR85]. One of the key aspects of these procedures is the manner in which geometric information is associated with those cells containing portions of the boundary and how this information is used to generate the element mesh in those cells [BAEH87], [SHEP86a].

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 than 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 indicate an O(n log(log n)) , where n is the number of points, computational time possible. (In two dimensions, the number of elements is of the same order as the number of nodes [BOLS86].) Computational results of an implemented three-dimensional algorithm gave O(n**5/3) computer times [CAVE85]. (In the three-dimensional case, the number of elements can be from O(n) to O(n**2) [BOLS86]. 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). [BAEH87], [JOE86]. 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 [JOE86]. The finite quadtree mesh two-dimensional generator [BAEH87] also demonstrates a linear growth rate with the number of elements. It is also anticipated that the finite octree mesh generator can operate in linear time, however, neither the analysis or numerical studies needed to confirm this have been completed.

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 (see [BABU86] for a good overview of this area).

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 indicators to determine where and, in the ideal case, how to improve the finite element discretization, and
4. mesh enrichment 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 solutions to a sequence 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.

Substantial gains in the development of adaptive finite element analysis techniques have been made in the past few years. However. it will be some time before they appear in commercial systems. These procedures are critical to the future automation of finite element modeling since they must be used to insure that the results obtained are meaningful.

Geometric Modeling Support for Automatic Mesh Generation. As indicated in the previous section. automatic mesh generators are geometrically 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 the only information available to the mesh generator is an output file of the geometric representation [WILS87]. Assuming that a common format is used for this file. this approach has the disadvantage of requiring that all the geometric modeling functionality needed by the mesh generator be reproduced within the mesh generator. Assuming that this functionality already exists 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 [CAMI86]. as well as those needed specifically for mesh generation [SHEP85]. The discussion below assumes a dynamic interface between the automatic mesh generators and the geometric modeling system. See reference [SHEP85] for a more specific discussion of the geometric communication operators needed to support the various automatic mesh generation approaches.

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 is used 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. 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 is 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 process and the rest of the meshing process are the interrogation operators of point classification. the conversions between parametric and real coordinates. and the conversion from real to parametric coordinates.

Geometrically-Based Finite Element Modeling. The first key to the integration of geometric modeling and finite element modeling is the use of a general data structure that can properly house various geometric forms. As indicated above. the transfer of only geometric data into the finite element modeling system does not address the geometric modeling needs of finite element modeling. Therefore. the second key aspect of this integration is the use of a general set of operators to support the geometric modeling demands of the entire finite element modeling process.

Before discussing the data structures and geometric modeling functionality needed. it is necessary to understand the process of generating a finite element model. This process consist of the:

1. definition of the domain to be analyzed.
2. specification of the partial differential equations to be solved.
3. specification of the analysis attributes.
4. specification of the numerical analysis control information.
5. specification of the mesh control information. and
6. generation of the finite element mesh.

The first three steps are concerned with the specification of the problem to be analyzed and are entirely independent of the numerical analysis procedures used. The last three steps are concerned with the specification and generation of the numerical analysis model. There are a number of advantages that can be gained by separating the modeling process into these distinct steps. The most obvious is the increased levels of integration possible between geometric and finite element modeling procedures. Possibly the most important. but least obvious. is that increasing the level of automation of the finite element modeling process is only possible if there is a strict separation of these steps.

When considering the development of integrated. geometrically-based finite element modeling procedures. it is important to realize that the geometric representation that is actually discretized into finite elements is often not the same as the original geometric description that defines the object. It is common in finite element analysis to ignore geometric details that are deemed unimportant to the analysis. Common geometric simplifications of this type include removing small fillets. and filling small holes and pockets. It is also common in finite element analysis to represent specific portions of the model with

reduced dimension entities. Common examples are to use only the 'mid-surface' of portions of the model that are 'small' in one direction compared to the other two. and to use only the 'center-line' of portions of the model which are 'small' in two directions. In these cases. the finite element discretization is of those reduced order entities where the eliminated dimensions are accounted for by the specification of 'section properties'.

There are two distinct steps in the finite element modeling process where these model domain differences can be specified. They can be done during the specification of the domain to be analyzed where the analyst would carry out the geometric modeling operations necessary to insure that the geometric representation used in the remainder of the finite element modeling process is that which is discretized into a finite element mesh. This is the approach commonly taken today.

The other step where the domain differences can be defined is during the specification of the numerical analysis attributes. In this case. those portions of the domain that are to be ignored or represented with reduced order elements are simply flagged with the appropriate attribute information defining how it is to be modeled in the numerical analysis model. It is then the responsibility of the finite element discretization procedures to perform the operations necessary to have the meshing procedures generate the mesh accounting for the domain differences. Although not commonly used procedures taking this approach can drastically reduce the amount of effort required for the generation of finite element models for some classes of problems [GREG87].

The previous section introduced the concept of geometric communication operators to support automatic mesh generators. In addition to the operators needed for this function [SHEP85]. sets of operators are needed to define both the analysis and numerical modeling attributes needed for the completion of the analysis model [SHEP85a]. [SHEP86b]. Efforts are currently under way to identify the mapping from the specific operators defined for finite element modeling [SHEP85], [SHEP85a] and those defined in the CAM-I Applications Interface Specification [CAMI86]. The advantage of this approach is obvious. it avoids the need to reproduce all the geometric modeling functionality of each geometry type within the finite element modeling system. This advantage is absolutely necessary if finite element modeling procedures are to be interfaced with the various geometric modeling systems.

The data structures used in a geometrically-based finite element modeling system play a critical role in the operation of the system. Since all geometrically complete representations can produce a boundary representation [RIQU82], and a boundary representation provides a level of abstraction that is independent of the specific geometric definition of the boundary of the domain [WEIL85]. WEIL86]. it is ideally suited for storing geometric representations for finite element modeling.

The combination of the topological information in a boundary representation and an appropriate set of geometric communication operators provides a generalized approach to the integration of finite element modeling capabilities with geometric modeling systems. The input to the finite element modeling software would be the topological representation of the object independent of the specific geometric definition of the topological entities. Although the topology contains no 'shape' information. it does contain a complete set of connectivity information and also indicates the dimensionality of the portions of the

object. The finite element modeling functions can be easily structured to be controlled by topological information calling the appropriate geometric communication operators to carry out the specific geometric calculations and modeling operations needed. The application of the geometric communication operators can also be keyed by topological information. Therefore, the finite element modeling software can carry out all its tasks without specific knowledge of the geometric representation.

There are a number of possible ways to group the finite element modeling data. The one given herein represents the minimal number of data sets that provide a logical separation of information needed for finite element modeling. The data sets include:

1. The MODEL data set
2. The ATTRIBUTE data set
3. The MESH data set

The MODEL data set contains the topological data, and points to the geometric information that defines the domain to be meshed. The ATTRIBUTE data set contains both the analysis attribute data (e.g., material properties, boundary conditions, etc.) and the analysis model control data. The MESH data set contains the finite element mesh generated for the model. The data structures are related through a well defined set of pointers which provide the mechanisms through which all non-MODEL data is tied to the MODEL and thus each other [SHEP86b].

The most fundamental data to the generation of a finite element model is the geometry. As indicated above, a boundary-based MODEL data structure provides a general framework for this data structure. There are a number of possible alternative boundary structures that can be considered [WEIL85],[WEIL86], with the choice to be made based on a trade-off between domain of geometries properly represented, storage, and need to search. The most critical of these questions is domain of geometries represented. Since finite element models commonly consist of combinations of three-dimensional (solid elements), two-dimensional (shell elements), and one-dimensional (beam elements) it is desirable to employ a MODEL representation that can house all three without the need for special cases. The commonly used boundary representations for solid modeling systems can only represent two-mainfold geometries which means that even a mesh of solid elements alone would require special consideration. However, the recently developed radial-edge data structure [WEIL86] can house combined solid, surface, and wireframe geometries in a consistent manner. Therefore, it is ideally suited for the representation of the finite element MODEL data structure [SHEP86b].

In addition to the hierarchy of geometric modeling entities, it is also desirable to employ a hierarchy of finite element entities in the MESH data structure. It is used to define the elements themselves. This is a departure from the way in which finite elements have historically been defined (i.e., an element of a specific type with a list of nodes which define the connectivity). In such a hierarchy each finite element entity points to the lowest order modeling topology entity which it is inherently a part [SHEP86b]. For example, a fe-edge which is on the surface of a region would point to the face on which it lies, rather than the region itself.

The MESH data structure. with its hierarchy of finite element entities. may seem too elaborate. perhaps even wasteful of storage. However. on closer inspection some distinct advantages emerge. The most powerful advantages come from the links to the other data structures. The major benefits for linking the finite element hierarchy to geometry is as follows:

1. It makes it possible to interrogate the finite element model using a geometric entity as a key word for searching.
2. It provides a mechanism which supports mesh generation on the basis of topologically simple cells (i.e.. quadrilaterals. triangles. hexahedrons. etc.) providing a direct procedure to represent all order elements without going back to the mesh generator. All higher order fe-nodes can easily be placed precisely on the appropriate associated geometric entity.
3. It provides an organization for handling any type of finite element in a uniform manner.
4. It provides direct access paths to higher order entities from lower order entities which make it very convenient to do such things as bandwidth minimization. postprocess the results of elements associated with a given set of nodes. etc.

Closing Remarks. The automated finite element modeling procedures currently under development place severe demands on the interface to geometric modeling. It is no longer satisfactory to simply pass a geometry file to the finite element modeling procedures. they require a full set of geometric modeling functions. These needs can only be addressed by the use of a dynamic interface of the type presented in the CAM-I Applications Interface Specification [CAMI86]. To support such an approach in a general and modular sense. future finite element modeling software should be driven by the topological information available from a boundary representation. Since finite element models are typically non-manifold. the boundary representation should be a complete non-manifold representation like the radial-edge structure [WEIL86].

References

[BABU 86]

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.

[BAEH 87]

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.*.

[BOLS 86]

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

[BYKA 76]

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

[CAMI 86]

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

[CAVE 74]

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.

[CAVE 85]

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

[DWYE 86]

R.A. Dwyer. "A Simple Divide-and-Conquer Algorithm for Constructing Delaunay Triangulation 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.

[FIEL 85]

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

[FIEL 86]

D.A. Field. "Implementing Watson's Algorithm in Three Dimensions". *Proc. Second Annual Symposium on Computa tional Geometry*. ACM 0-89791-194-6/86/0600/246. 1986, pp. 246-259.

[GREG 87]

B.L. Gregory and M.S. Shephard. "The Generation of Airframe Finite Element Models Using an Expert System". *Engineering with Computers*. to appear.

[JOE 86]

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

[KELA 86]

A. Kela. R. Perucchio and H.B. Voelcker. "Towards Automatic Finite Element Analysis". *Computers in Mech. Engng.*, July 1986. pp. 51-71.

[LEE 84]

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.

[LO 85]

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

[NGUY 82]

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

[RIQU 82]

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.

[SADE 80]

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

[SIBS 78]

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

[SLUI 82]

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.

[SHEP 85]

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

[SHEP 85a]

M.S. Shephard. "Finite Element Modeling within an Integrated Geometric Modeling Environment: Part II - Attribute Specification. Domain Differences. and Indirect Element Types". *Engineering with Computers*. Vol. 1. pp. 72-85. 1985.

[SHEP 86]

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

[SHEP 86a]

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. p. 1-18.

[SHEP 86b]

M.S. Shephard and P.M. Finnigan. "Integration of Geometric Modeling and Advanced Finite Element Preprocessing", to appear, *Finite Elements in Analysis and Design.*

[SHEP 87]

M.S. Shephard. "Approaches to the Automatic Generation and Control of Finite Element Mesh", TR-87005, CICG, RPI, Troy, NY, submitted to *Applied Mechanics Review.*

[WEIL 85]

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.

[WEIL 86]

K.J. Weiler, "Topological Structures for Geometric Modeling", PhD Thesis, CICG, TR-86032, Rensselaer Polytechnic Institute, Troy, NY, 1986.

[WATS 81]

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

[WILS 87]

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.

[WOO 87]

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

[WORD 84]

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

[YERR 83]

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.

[YERR 84]

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.

[YERR 85]

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.