SOLID/FEM INTEGRATION at SNLA *

Patrick F. Chavez
CAD Technology Division
Sandia National Laboratories
Albuquerque, New Mexico 87185

This presentation will describe the effort at Sandia National Laboratories Albuquerque with emphasis on the methodologies and techniques being used to generate strict hexahedral finite element meshes from a solid model. We utilize the functionality of the modeler to decompose the solid into a set of non-intersecting meshable finite element primitives. The description of the decomposition is exported, via a Boundary Representation format, to the meshing program which uses the information for complete finite element model specification. Particular features of the program will be discussed in some detail along with future plans for development which includes automation of the decomposition using artificial intelligence techniques.

*This work performed at Sandia National Laboratories was supported by the U.S. Department of Energy under contract DE-AC04-76DP00789.

Automatic Mesh Generation and Optimization from the Solids Model Database

SAND85-2822C, CAD/CAM 031

Patrick F. Chavez *

A proposed system to generate finite element models directly from the solids model database is presented. This system includes automatic error analysis with adaptive gridding for equilibration of the error estimator in use. The complete specification of the finite element model including boundary conditions and material identifiers is produced to a neutral output file. An illustrative example depicting the state of implementation of the proposed system is contained within. Current research is also briefly described.

Introduction

The advancing technology of computing hardware and software is well represented by the current Computer Aided Design (CAD) systems employing solids modeling. These solids modeling systems, under development by both universities and industry, have the obvious benefit for the realistic visualization of three-dimensional (3-D) objects. The most important benefits of solids modeling, however, do not lie in the solid model itself, but in the subsequent applications which utilize the valid and unambiguous geometric information available. In other words, the advantage of solids modeling is not as a stand alone application but as a means of creating a geometrical database to unify a number of applications. Indeed, users and vendors currently seem to be concentrating their efforts at integrating the solids model database in the areas of Finite Element Modeling (FEM) and Numerical Control (NC) Programming. Solids modeling does appear to have the potential for unifying the design, engineering, and manufacturing areas of industry.

At Sandia National Laboratories a unified geometric database is expected to reduce design time and yield added reliability and optimization of the designed systems. A joint effort between the Engineering Sciences and the Computer Aided Design Departments has been defined and is being pursued to integrate the Computer Aided Engineering (CAE) activities of the Engineering Sciences Department into the automated design and manufacturing process. The primary vehicle for this effort is the utilization of improved model generation capabilities with emphasis on advanced geometric definition and automatic mesh generation for FEM. In particular, the utilization of the CAD geometrical data and hence the elimination of the error prone reentry of such data is considered essential.

*Member of Technical Staff, CAD Technology Division 2814, Sandia National Laboratories, Albuquerque, NM, 87185.

This paper describes the effort underway at Sandia for integration of FEM Mesh generation utilizing PADL-2 [BROW82], the Constructive Solid Geometry (CSG) system produced at the University of Rochester. In general, because of the commercially available and locally developed finite element analysis codes in use at Sandia, a requirement for the use of hexahedral elements in 3-D FEM exists. This, coupled with the large number of nonlinear finite element analyses performed, prohibits us from considering the automatic tetrahedralization work [CAVE85] developed at General Motors Research Labs or the modified-octree work [YERR84] performed at Rensselaer Polytechnic Institute. The finite element mesh generation philosophy we are pursuing is divided into two primary phases; 1) initial mesh installation utilizing the available CAD geometric data base and 2) mesh optimization including mesh improvements based on geometrical aspects of the initial mesh and automatic error analysis coupled with node grading techniques to obtain uniformly reliable answers throughout the domain of analysis.

The following sections of this paper describe in some detail the relevant topics including 1) solids modeling, 2) application interface, 3) initial mesh generation, 4) mesh improvements, 5) and error analysis and adaptive gridding. An illustrative problem depicting the state of implementation of these topics is included.

Solids Modeling - A Geometrical Basis for Applications

The classical geometrical CAD database is the so-called "wireframe" format. To define wireframe, we introduce the notion of an edge. For us, how an edge is actually represented within some computer database is unimportant. Only the idea that an edge results from the intersection of two distinct surfaces matters. An edge is one-dimensional in a parametric sense. That is, although any point (x,y,z) on an edge is in Euclidean 3-space it can be derived through a system of equations depending on only one independent parameter of the form

$$x = X(s) \quad y = Y(s) \quad z = Z(s).$$

Here X, Y, and Z are functions of the independent parameter s which is bounded in the closed interval $[s_0, s_1]$. A wireframe representation then models a solid by simplistically specifying certain edges of the solid. Typically, those edges defined for a given solid correspond to the bounding edges of the domain being considered. Particular entity specification, referred to as instancing, is accomplished through a choice of a particular type of edge (say a line or circular segment) with a rigid motion and any other necessary parameters (say curvature) to complete the definition.

New geometrical modeling technologies are becoming popular. The two most popular technologies are CSG and Boundary Representation (B-Rep). CSG systems define solids as Boolean operations (union, difference, intersection) of simpler primitive solids (blocks, spheres, wedges, cones) instanced by size and location. The B-Rep, on the other hand, is a heirarchical extension of the wireframe format. In the B-Rep, solids are described as a collection of instanced (by type, size, and location) faces, each of which in turn are composed by a

number of edges. Explicit mathematical descriptions of both the faces and edges are usually available. The user interface for the CSG and B-Rep systems appear to be unifying with each other borrowing from the others successes. Primitive instancing, once strictly a tool of the CSG modeler, is found in several B-Rep modelers. Similarly, a sweeping formulation of edges to create faces and the sweeping of faces to form volumes have begun to appear in some CSG formulations.

We are, of course, interested in utilizing an unambiguous and valid description of a solid. The adjectives "unambiguous" and "valid" are similar to the terms "one-to-one" and "onto" as applied to invertible functions. When we say an unambiguous solid representation we imply that for a given representation it should correspond to one and only one solid. We do not have strict one-to-oneness since there is no unique representation for a given solid, but only a unique solid for every representation. Indeed, in any of the currently available geometrical modeling systems, there is no unique representation for a solid. There are as many definitions of a solid as there are users. As for the term "valid", we imply that for any representation we derive, it describes a solid although it need not be realizable from a manufacturing point of view.

It is easy to imagine that wireframe representations are neither unambiguous nor valid. Indeed, there are a myriad number of counter-examples testifying to this. On the other hand, both CSG and B-Rep systems have the ability to produce unambiguous and valid descriptions. Our work in automatic finite element analysis has been based on the unambiguous and valid geometric description available within the CSG modeler PADL-2. The choice of PADL-2 has been more a matter of convenience, since the source code and expertise are available at Sandia, than a matter of preference of CSG over B-Rep. In fact, it may be argued that the B-Rep facilitates certain applications, for example FEM and NC programming, that are primarily surface oriented.

For our implementation of the mesh generation we use the B-Rep, as supplied through a conversion routine available in PADL-2. These conversion routines are generally well understood and details of the PADL-2 implementation can be found in [HART81]. Our development thus is considered generic in the sense that any solid modeler capable of ultimately delivering a B-Rep, independent of its own internal representation, would be able to utilize the capabilities we are developing.

We have realized the benefits of using a valid and unambiguous solid model as neither the geometry nor the topology has to be supplemented. For wireframe applications it is quite typical that either additional topology or geometry has to be supplied before applications are undertaken.

Application Interface - The Link Between Geometry and Applications

The idea for using arbitrary solid modelers in conjunction with various applications is known under the broader category of "application interface". An application interface has been likened to a "software bus" enabling applications to communicate directly to

solids modelers for the purpose of interrogating or modifying the solid model. To date no standard application interface exists for the available solids modelers although efforts [CAMI86] to this end have been underway for some time. Still, some solids modelers make available to some extent the modeling operations required by locally developed applications. Application interfaces can be thought of as part of the solids modeler which make the internals of the modeler transparent to the application.

We have been able to use a number of the available routines within PADL-2 to facilitate the interface to the finite element mesh generator. These include routines for identifying and utilizing the geometric entities within the representation. For example, routines pertaining to the storage management structure, the rigid motion facility, and the computational geometry package have been used to discretize the body for mesh generation. Other utilities necessary for linking our application to PADL-2 have had to be defined and developed. These include routines that format the B-Rep available in PADL-2 for export to applications and the corresponding routines to read the representation into the mesh generator. The mesh generator also requires contiguous lists of edges and faces, called loops, which PADL-2 does not require. These have been developed. Redundant edges and faces are either necessary or add to the robustness of a solids modeler, but are detrimental to mesh generation. Algorithms to identify and eliminate redundant faces and edges have been implemented. Finally, although PADL-2 contains routines usable to discretize edges, none existed to discretize surfaces.

The above development has allowed the mesh generator to directly create finite element models from a solids description while guaranteeing that all nodes defined for the mesh either lie in the body or are exactly on the surface. This group of routines are necessary within another solids modeler for our implementation of the finite element mesh generator.

Mesh Generation - An Application for Solids Modeling

In this section the philosophy for generating hexahedral finite element meshes from a solids model database is presented. We proceed by briefly describing one technique for generating hexahedral meshes that is representative of the classical methods used. The method we are pursuing for mesh generation is an extension of these ideas imbedded in new technologies, namely solids modeling and feature recognition.

A hexahedral mesh can be constructed through a coordinate transformation in conjunction with higher order approximating functions. More specifically, the geometry of the body is constructed using hexahedral subregions each having six well defined faces and twelve edges. The description of each hexahedron requires the coordinates of eight corner points and one interior point of each edge for a total of twenty points. During the construction of a particular hexahedron, faces which are coincident with previously defined faces are identified. Thus, coincident nodes on coincident faces are assigned the same node number. Finally, a consistent number of divisions along

three "mutually orthogonal" directions for each hexahedron is specified.

A mesh of hexahedral elements can then be installed in each hexahedral region in the following manner. The twenty points given on each hexahedral subregion are considered the images of the unit cube S, where $S = [(r,s,t): 0 \leq r,s,t \leq 1]$, via maps given by $x = X(r,s,t)$, $y = Y(r,s,t)$, and $z = Z(r,s,t)$. Here the usually polynomial functions X, Y, and Z are of total degree three in each variable. The unit square is then subdivided into the specified number of divisions and the grid so formed is transformed via the above maps to the physical domain. If the interior points defined along the edges of the hexahedral are placed closer to a corner point, a higher density of elements is obtained in that portion of the subregion.

The mapping technique described above, usually referred to as an isoparametric mapping, necessarily matches the body at only the twenty interpolation points defined. A different mapping technique has been utilized in our work. Our mapping technique is related to the transfinite mapping work of Haber et. al. [HABE81,HABE82], in that a non-denumerable set of points on the surface of the body can exactly be matched. This mapping is derived by utilizing the paramatric representation of the surfaces available in PADL-2 to locate the mesh points on two "opposite" faces. The interior points of the mesh are then generated through a lofting of the meshes on these faces. For the simple subregions implemeted to date, these interior points are guaranteed to lie interior to the subregion. As more geometrically complicated subregions are added, validity of the location of the interior points will be checked through point classification, a capabilitiy of the solids modeler PADL-2.

In the discussion of a classical hexahedron mesh generator, we described the geometric definition of the body as an assemblage of large hexahedra. This definition of a solid is overly restrictive. This construction is unnatural and inefficient when using general solids modelers. Even for systems explicitly designed for this purpose, this construction can be overly time consuming for all but the simplest cases.

For our work, no such restrictions on the geometry creation is assumed. The full power of the solids modeler is utilized. Our philosophy for subregion definition is that all the capabilities of the solids modeler are used to decompose the body into a set of regions within each of which a hexahedral finite element mesh can be installed. We term these subregions "finite element primitives". That is, the solid model is decomposed using the primitives and Boolean operations of the solids modeler into a set of finite element primitives. The resulting set of finite element primitives need not coincide with the geometric primitives of the solids modeler. The finite element primitves to be supported include all the geometric primitives plus all the topolologically equivalent entities. For example, any volume defined by one surface, topologically equivalent to a sphere, will be able to be meshed.

Allowing more general finite element primitives either necessitates the definition of new mapping techniques or a decomposition of each of the finite element primitives into a collection of hexahedra. This last alternative is easily accomplished. Figure 1 shows the decomposition of the standard geometric primitives. This decomposition of the finite element primitives will be automatic in the solid modeler and transparent to the user.

The mesh generation is only automatic in each finite element primitive. Presently human interaction is required for the primitive decomposition. Work is beginning in the area of feature recognition, as applied to recognizing the finite element primitives, to automate this process.

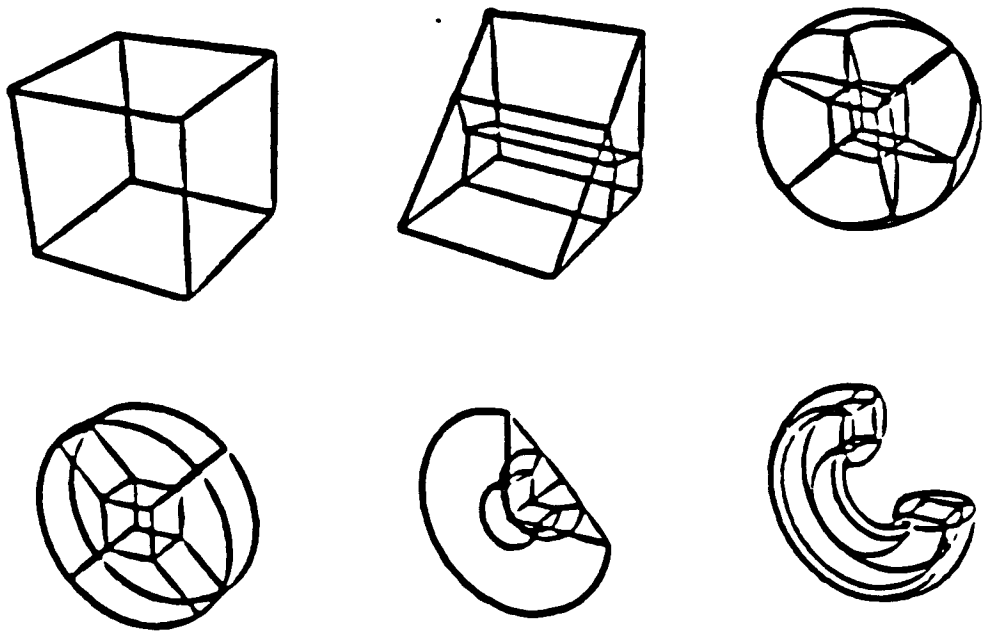

Figure 1. Decomposition of the Standard Geometric Primitives into Hexahedron.

Mesh Improvements - Assuring Geometrically Good Meshes

In the previous discussion of the mesh generator we did not enumerate the characteristics of a "good" mesh. We do so now. Some characteristics of a good mesh are 1) gradually changing element sizes, 2) gradually changing element shapes, and 3) as nearly rectangular (even cubical) elements as possible. These characteristics have important numerical consequences. For example, the third condition assures us in practice of a well defined (one-to-one and onto) coordinate transformation during the stiffness matrix formulation. In addition, all the above characteristics attempt to maintain the condition numbers of the stiffness matrices generated for two nearby elements to be similar.

Our approach to generating good meshes is an extension of the ideas incoporated in the two-dimensional (2-D) mesh generator QMESH [JONE74]. Only the necessary details of these developments will be given in this section. A more complete description of the 2-D implementation can be found in the QMESH documentation.

In our mesh generator, like QMESH, the initial mesh is evaluated and improved through a series of processors working in tandem. The processors have the capabilities to automatically reposition nodes, delete elements, and rearrange the topology in an attempt to improve the element geometry. The mesh improvements, as we now discuss, are only concerned with the geometrical aspects of the mesh. The sufficiency of the mesh with regard to accuracy is discussed in the next section of this paper. The general concepts of the algorithms for node smoothing, topology restructuring, and element deletion are now described.

The node repositioner, or smoother, consists of attempting to have the nodes equidistant and the elements having equal volumes. Requiring the nodes to be equidistant is tantamount to requiring that every node is at the average location of all its neighbors. Symbolically we have

$$(x,y,z) = \frac{1}{n}(\sum x_i, \sum y_i, \sum z_i)$$

where n is the number of neighboring points each with coordinate (x_i, y_i, z_i). This formula is the one applied in the smoothing code but in a slightly altered fashion. The expression is rewritten as

$$(x,y,z) = (x_0,y_0,z_0) + \frac{1}{n}(\sum(x_i-x_0), \sum(y_i-y_0), \sum(z_i-z_0))$$

or more succinctly

$$(x,y,z) = (x_0,y_0,z_0) + \frac{1}{n}\sum l_i.$$

Here (x_0,y_0,z_0) and (x,y,z) are respectively the old and updated positions of the node being moved and $V_L = \frac{1}{n}\sum l_i$ is the "Laplacian" movement vector.

Only a related form of the volume equilibration has been considered to date. Instead of requiring the volumes for all elements to be the same, we impose that each of the areas of the faces on all the elements are equal. This has been proposed to more fully utilize the capabilities already developed in QMESH. This requirement is represented in the formula

$$V_A = \sum \frac{A_{if}-A_{ib}}{A_{if}+A_{ib}} V_i$$

where V_A is the "Area-Pull" movement vector (corresponding to V_L in the node equidistribution) applied to the node in question. The $A_{i,f}$ and $A_{i,b}$ refer to the areas of the face "in front" and "in back" of the node. Again, more complete description of the formulation in the two-dimensional setting can be found in [JONE74]. The Laplacian and Area-Pull moment vector for a node are incorporated through a convex

combination of the two. That is the moment vector for a given node is taken as

$$V = \alpha V_A + (1-\alpha)V_L$$

with $\alpha \in [0,1]$ a user selectable parameter.

The next capability is the restructuring of the element topology, i.e., the element connectivity. By this we mean the process of erasing an interface plane and drawing it differently to improve the geometrical shape of the neighboring elements to the plane. To assess the element shapes, three element evaluator functions referring to the angle condition, the aspect ratio, and the product of these two have been defined. These definitions are extensions to those developed by Jones for QMESH.

The operation of the restructuring process is then the following: the condition numbers for all the elements are evaluated and a list of the twenty-five worst (largest) is saved. The processor attempts to improve the worst element in the mesh. If no improvement is made in any of the first ten worst elements, the processor quits. If a restructure is accomplished, the list of worst elements is updated and the process is continued so long as a restructure is performed among the ten worst elements of the mesh.

The final processor contained is the element deleter. This processor attempts to improve the mesh by deleting elements. Element deletion is similar in nature to the restructuring processor. This processor sweeps through the mesh to make a list of the five worst "rhombic" elements. The measure of how rhombic an element, termed the R-number, is defined as the ratio of the length of the shorter diagonal to the length of the longer diagonal. If the R-number of an element is less than tan(V/2), where V is normally forty-five degrees, the element is placed in the candidate list for deletion in ascending order. The more rhombic an element is the smaller the R-number. The tolerance parameter tan(V/2) is the R-number of a parallelogram with opposite angles of V and is not simply a measure of how sharp an element is. The program then starting with the worst (smallest) R-number, attempts to eliminate the element. As soon as an element is deleted, control returns to the calling program.

The sequence that the processors operate on can effect the outcome of the mesh. A general method for specifying the sequence of the processors has been implemented. The entire sequence is iteratively performed until convergence (no more node smoothing, element restructuring, or element deletion) is attained.

The full capability of the mesh improvement has not been implemented to date. Only those capabilities corresponding to each of the lofting planes generated in the primitives are acted on. We can show that for certain limited cases this is a partial implementation of the entire algorithm for a true 3-D mesh improvement. We have observed that the primary processor functioning is the smoother, attributing to the initial quality of the meshes generated.

Error Analysis and Adaptive Gridding - Generating Computationally Optimum Meshes

The topics of error analysis and adaptive gridding have attracted considerable attention. For example, it is well known theoretically that inserting degrees of freedom into the finite element method can yield more accurate results. In practice, it has been verified that often the error can be reduced when the number of degrees of freedom are increased. In general, there are two ways of decreasing the error by increasing the number of degrees of freedom in the finite element space. The first method, usually referred to as the "p-method", involves increasing the degree of the polynomial used over each element while leaving the total number of elements in the domain fixed. This refinement, as usually implemented, has no effect on the approximation of the geometry and hence the initial mesh should be developed to include all the important geometrical aspects.

The second method of increasing the accuracy of the finite element method is termed the "h-method". In the h-method, the degree of the approximating functions over each element is maintained while the portion of the domain spanned by each element is decreased. In other words, additional elements (and thus nodal points) are placed in the region. Again, in practical applications of the h-method, no improvement of the geometrical model is attained. The theoretical aspects of the h-method are probably more developed than those pertaining to the p-method.

The difficulty with both the above approaches is that extensive modifications are required for the finite element analysis codes in use today to take advantage of these developments. A third alternative for adaptive gridding is possible. Errors in numerical methods are pointwise dependent. That is, errors in an analysis usually vary from one element to another. One reasonable goal to strive for is a uniformly reliable answer with the same error associated with every element. Thus, if elements are concentrated in the area where errors are large while decreasing the number of elements where errors are small, we can hope to produce such a result. In effect, we are trying to automate the technical expertise applied by computational stress analysts in achieving reasonable results.

The methodology currently being pursued is the latter approach primarily because of the large investment in conventional FEM techniques not involving the h- or p-methods of refinement. It may be quite some time before software is readily available for applying general h- or p-methods especially for the non-linear problems of interest at Sandia. The question now arises: HOW do we introduce the adaptive grading techniques in our calculations? To answer this we look at the composite parts of the problem. They are error evaluation and node distribution.

The problem of finite element error evaluation has been studied extensively. Sophisticated theoretical work has been done by Babuska and Rheinboldt [BABU78a, BABU78b, BABU80]; K. Miller and R. Miller [MILL81a]; K. Miller [MILL81b]; and Babuska and A. Miller [BABU81] on error evaluation. To paraphrase their work without extensive technical details, an estimate of the local error in an energy norm at a

given node is derived by considering the surrounding elements to that node. Here the energy norm for a function f is defined as

$$||f||_{E(D)} = \int_D |\nabla f|^2 dv.$$

The indicated integration is carried out over the domain covered by the neighboring elements of the node of interest and is notated by D. It is assumed that the finite element solution, denoted by u_h, to the desired partial differential equation (PDE) is available. A model problem, perhaps mimicking the actual PDE being solved, with boundary data corresponding to u_h is then solved on D and denoted by w. It is then reasonable to assume that the quantity $||u_h - w||_{E(D)}$ approximates the local energy error $||u_h - u||_{E(D)}$ where u is the exact (and usually unavailable) solution. Various refinements and extensions to this idea are the topics of the references cited.

The error indicator we are currently experimenting with is different from the one presented above. Our error indicator is simpler to evaluate and attempts to estimate the maximum pointwise error in an element rather than a local energy error. Briefly, under conditions which are usually satisfied, it can be shown that in the maximum norm the finite element solution is the optimal solution available from the span of the basis functions. Thus

$$||u-u_h||_{L_\infty} \leq C \min_{X \in S_h} ||u-X||_{L_\infty}$$

where $||*||_L$ denotes the maximum norm and S_h denotes the finite element subspace depending on the choice of the discretization h and the approximating polynomials S used. Then, on an element k approximation theory yields

$$||u-u_h||_{L_\infty}(K) \leq \begin{cases} Ch_K ||\nabla u||_{L_\infty}(K) \\ Ch_K^2 ||D^2 u||_{L_\infty}(K). \end{cases}$$

Here D^2u denotes the generic second derivative while h_k is the diameter of the element. In the case of linear elements we use the first inequality replacing ∇u by ∇u_h. For quadratic elements the second inequality can be used replacing D^2u with D^2u_h. The maximum norm is currently estimated at the quadrature points of the elements.

This error indicator is conservative for problems with smooth solutions. It does not take into account the full order of the polynomial approximation for such problems. It is known that for problems with smooth solutions, the error using linear elements would be of order h^2 and not h. Similarly for quadratic elements the error would be of order h^3 and not h^2. This error indicator may be more suitable for non-smooth problems such as in shock calculations. Again, the primary advantage of the proposed error indicator is its computability and it is a pointwise estimate.

We now consider the problem of how to distribute the nodes to equilibrate the error indicator and obtain a uniformly reliable solution throughout the domain of anyalysis. This problem is easily

addressed with the development of the geometrical smoothing as discussed earlier. As noted, the movement vector as considered in the Area-Pull and Laplacian smoother is given as

$$V = \alpha V_A + (1-\alpha) V_L.$$

It is reasonable to expect that the new node movement vector defined as

$$V = \alpha V_A e_A + (1-\alpha) V_L e_L$$

where e_A and e_L represent the errors associated with the Area-Pull and the Laplacian movement vector respectively, would cluster the elements where errors are large. For example, e_A could be defined as the maximum error computed for the element corresponding to v_i while e_L could be the average of the errors associated with all the elements surrounding l_i. Overall, the node distribution for adaptive grading is taken as an error weighting of the geometrical node smoothing.

An Example - Current Capabilities

As an illustration of the procedure for creating a finite element mesh and performing automatic error analysis and adaptive gridding, we consider a relatively simple but realistic problem. The problem involves an L-bracket with a cylindrical hole in the bottom slab. The part is constructed in the PADL-2 language through the union of two properly instanced blocks and the difference of a properly instanced cylinder. The solid is decomposed into six finite element primitives. The solids model of the decomposed L-bracket is shown in Figure 2. The four primitives surrounding the hole were created by first defining a coordinate system with respect to the axis of the hole. A properly instanced wedge with its apex parallel to the holes axis was defined. Finally, four intersections of the bracket with the wedge after appropriate rotations of the wedge about the holes axis yielded the indicated finite element primitives. The two remaining finite element primitives were obtained by intersecting appropriately instanced blocks with the L-bracket.

The B-Rep of the decomposed bracket was transferred to the mesh generator via several of the routines imbedded in the application interface. For analysis purposes, the part is assumed to be of aluminum construction with a load applied to the right front vertical edge. Discretization data supplied via the user interface in the mesh generator resulted in the mesh indicated in Figure 3. Appropriate material indicators and boundary condition flags were specified, again within the FEM user interface, before formatting the model into a neutral format for analysis. Automatic translation of the neutral formatted finite element model into SAP IV [BATH74] format for a linear elastic static analysis was accomplished. Typical results showing the minimum principal stress contours is shown in Figure 3.

Figure 4 shows the results of the automatic error analysis. Areas of large error are indicated as a high density of error contours and corresponds to locations of high stress. This is intuitively correct since the error measure is related to the calculation of the

strains and hence proportional to the stresses. Finally, Figure 5 shows the adapted mesh and the analysis results on that mesh. In general the error indicator has equilibrated to reasonable values and the answer is considered to be uniformly reliable in the domain of interest for the number of elements used.

Conclusions

We have presented in some detail the theory and development behind a three-dimensional hexahedral finite element mesh generator working directly from a solids model database. In conjunction with the mesh generator are co-developments in automatic error analysis and adaptive grading to produce uniformly reliable analyses.

Research and development pertaining to the overall system is continuing. Development of the mesh improvement schemes and inclusion of more topologically complex finite element primitives is proceeding. The mesh generation phase is only automatic in each finite element primitive. Full automation is impossible without automating the finite element primitive decomposition. Research is underway in the general area of feature recognition as applied to the process of primitive decomposition. In addition, work is continuing in the areas of automatic error analysis and adaptive grading. This work is primarily seen to remain in the area of adaptive grading because of the predominant nature of the commercially available finite element analysis codes.

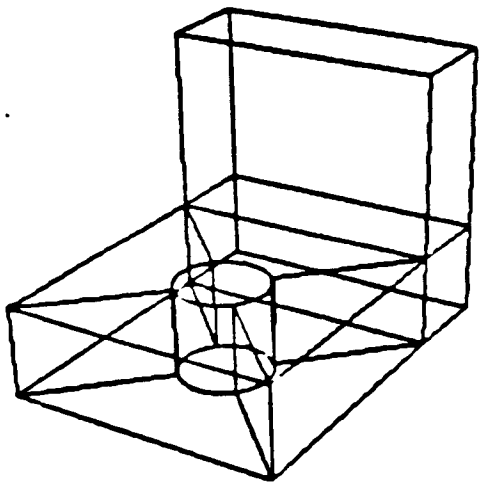

Figure 2. The Solid Model of the Primitive Decomposition

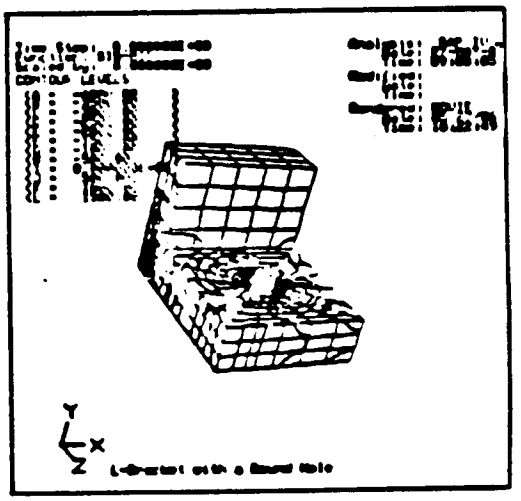

Figure 3. The Minimum Principal Stress Contours on the Hexahedral Mesh Generated.

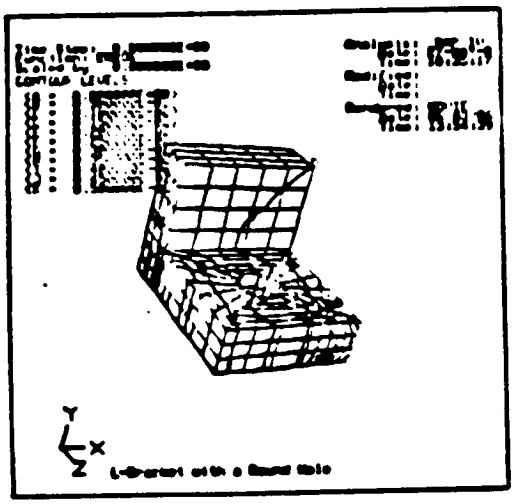

Figure 4. Error Contours from the Automatic Error Analysis

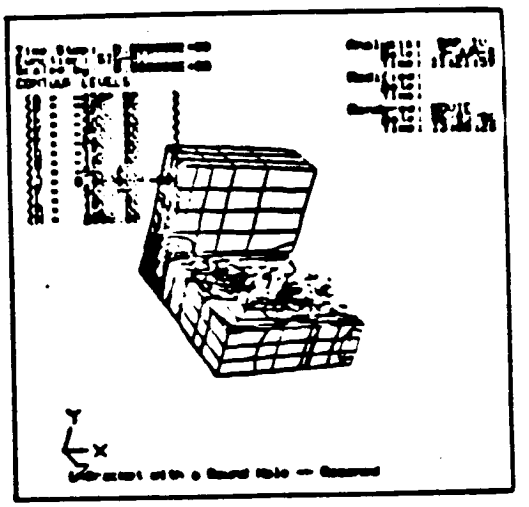

Figure 5. The Minimum Principal Stress Contours After Adaptive Gridding

References

BABU78a Babuska, I., and Rheinboldt, W.C., "Error Estimates for Adaptive Finite Element Computations," SIAM Journal of Numerical Analysis, Vol. 15, pp. 736-754, 1978.

BABU78b Babuska, I., and Rheinboldt, W.C., "A-Posteriori Error Estimates for the Finite Element Method," International Journal of Numerical Mechanical Engineering. Vol. 12, pp. 1597-1615, 1978.

BABU80 Babuska, I., and Rheinboldt, W.C., "Reliable Error Estimates and Mesh Adaptation for the Finite Element Method," Computational Methods in Nonlinear Mechanics, North Holland, Amsterdam, pp. 67-108, 1980.

BABU81 Babuska, I., and Miller, A., "A-Posteriori Estimates and Adaptive Techniques for the Finite Element Method," University of Maryland, Institute for Physical Science and Technology, Tech Note BN-968, 1981.

BATH74 Bathe,K.J., Wilson, E.L., and Peterson, F.E., "SAP IV - A Structural Analysis Program for Static and Dynamic Response of Linear Systems," Earthquake Engineering Research Center, College of Engineering, University of California, Berkeley, California, 1974.

BROW82 Brown, C.M., "PADL-2: A Technical Summary," IEEE Computer Graphics and Applications, Vol. 2, No. 2, pp. 69-84, March 1982.

CAMI86 Computer Aided Manufacturing-International, "Application Interface Specification-Volumes I and II", Cranfield Institute of Technology, January 1986.

CAVE85 Cavendish, J.C., Field, D.A., and Frey, W.H., "An Approach to Automatic Three-Dimensional Mesh Generation," International Journal for Numerical Methods in Engineering, Vol. 21, pp. 329-347, 1985.

HABE81 Haber, R., Shephard, M.S., Abel, J.F., Gallagher, R. H. and Greenberg, D.P., "A General Two-Dimensional, Graphical Finite Element Processor Utilizing Discrete Transfinite Mappings," International Journal for Numerical Methods in Engineering, Vol. 17, pp. 1015-1044, 1981.

HABE82 Haber, R. and Abel, "Discrete Transfinite Mappings for the Description and Meshing of Three-Dimensional Surfaces Using Interactive Computer Graphics," International Journal for Numerical Methods in Engineering, Vol. 18, pp. 41-66, 1982.

HART81 Hartquist, E.E., Peterson, D.P., and Voelker, H.B., "BFILE/2: A Boundary File for PADL-2," Technical Memo CGGM-20, University of Rochester, Rochester NY, March 1981.

JONE74 Jones, R.E., "QMESH: A Self-Organizing Mesh Generation Program," Publication N. SLA-73-1088, Sandia National Laboratories, Albuquerque, N.M., 1974.

MILL81a Miller, K., and Miller, R., "Moving Finite Elements-I," SIAM Journal of Numerical Analysis, Vol. 18, pp. 1019-1032, 1981.

MILL81b Miller, K., "Moving Finite Elements-II," SIAM Journal of Numerical Analysis, Vol. 18, pp. 1033-1057, 1981.

YERR84 Yerry, M.A., and Shephard, M.S., "Automatic Three-Dimensional Mesh Generation by the Modified-Octree Technique," International Journal for Numerical Methods in Engineering, Vol. 20, pp. 1965-1990, 1984.

Key Words

1. Solids Modeling
2. Application Interface
3. Finite Element Mesh Generation
4. Mesh Optimization
5. Error Analysis
6. Adaptive Gridding
7. Finite Element Analysis
8. Computer Aided Design
9. Computer Aided Engineering