

GOVERNMENT GRANT

5-437

# RENSSELAER POLYTECHNIC INSTITUTE

## CENTER FOR INTERACTIVE COMPUTER GRAPHICS/CAM-I

NA-5-437

10/1/87

(1+14)

IN-61-CIC

330 p

## WORKSHOP ON THE INTEGRATION OF FINITE ELEMENT MODELING WITH GEOMETRIC MODELING

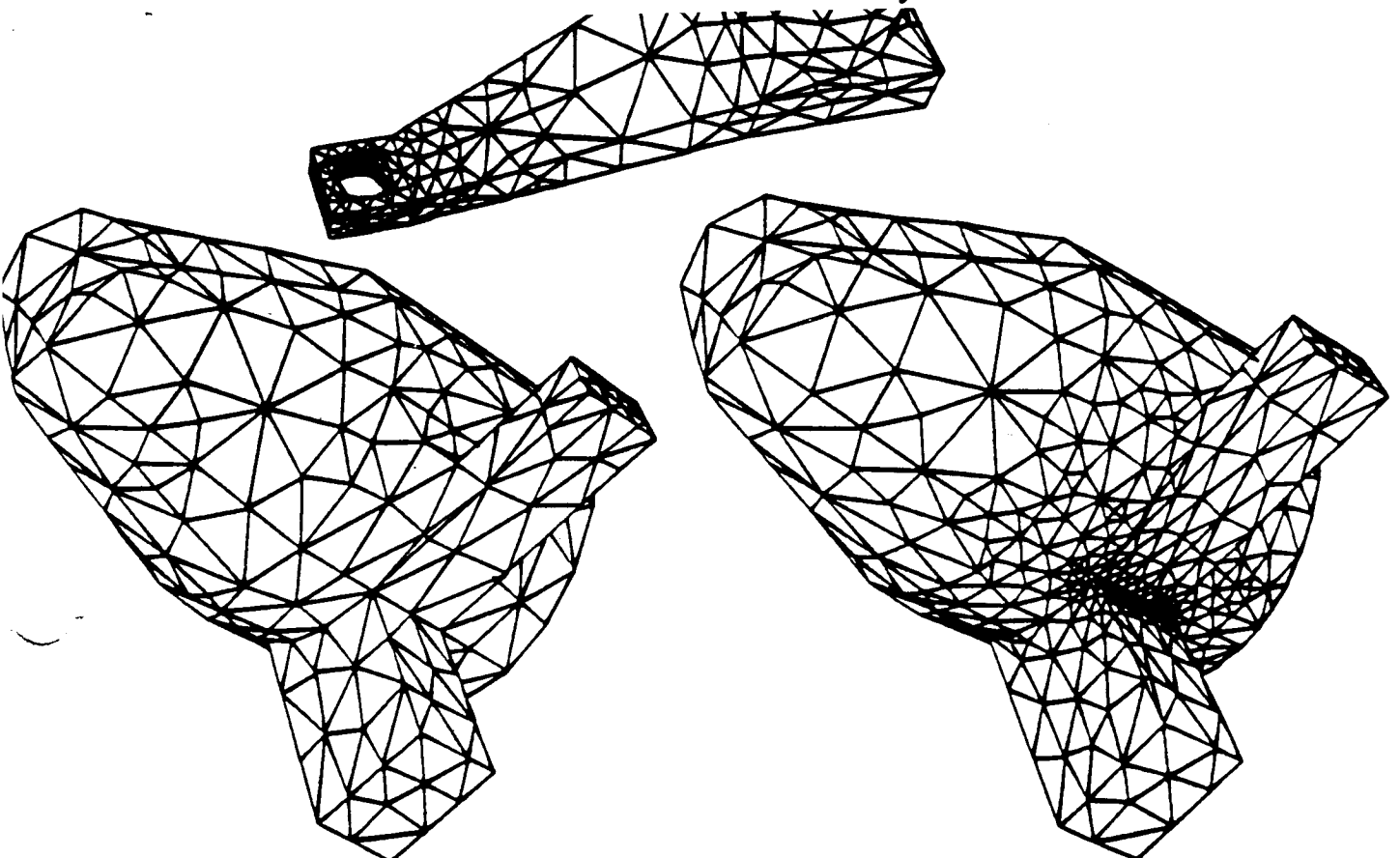
### MAY 12, 1987

(NASA-CR-182515) WORKSHOP ON THE  
INTEGRATION OF FINITE ELEMENT MODELING WITH  
GEOMETRIC MODELING (Rensselaer Polytechnic  
Inst.) 330 p

CSSL 09B

N88-19111  
--THRU--  
N88-19125  
Unclas  
0125786

G3/61



Center for Interactive Computer Graphics  
Rensselaer Polytechnic Institute  
Troy, New York 12180-3590

**WORKSHOP ON THE INTEGRATION OF FINITE ELEMENT  
MODELING WITH GEOMETRIC MODELING**

Room 4050: Center for Industrial Innovation (CII)

May 12, 1987

**AGENDA**

- 9:00 Introduction to the Integration of Geometric and Finite  
Element Modeling  
Mark S. Shephard, RPI
- 9:30 Integration of Geometric Modeling and Advanced Finite  
Element Programming  
Peter M. Finnigan, General Electric Corporate Research and  
Development
- 10:10 Break
- 10:30 Integration Architecture of SDRC Geometry and Finite  
Element Processors  
Robert L. Haubrock, Structural Dynamics Research  
Corporation
- 11:00 Finite Element Meshing of ANSYS Solid Models  
F. Stanley Kelly, Swanson Analysis Systems 54
- 11:30 Solid/FEM Integration at SNLA  
Patrick F. Chavez, Sandia National Laboratories 55

- 12:00 Lunch
- 1:00 Octree Based Automatic Meshing from CSG Models *56*  
Renato Perucchio, University of Rochester
- 1:30 Finite Octree Meshing Through Topologically Driven  
Geometric Operators *512*  
Kurt R. Grice, RPI
- 2:00 Design Modeling for Shape Optimization *511*  
Mark E. Botkin, General Motors Research Laboratories
- 2:30 Postprocessing Techniques for Three-dimensional Nonlinear  
Structures *513*  
Richard S. Gallagher, Hibbitt, Karlsson & Sorensen
- 3:00 Break
- 3:15 Geometric Versus Finite Element Modeling - Current and  
Future Trends at Northrop *514*  
Shiv Bajaj, Northrop Corporation
- 3:35 Building F.E. Applications Using Nonmanifold Boundary  
Operators, and the Generation of Idealized Models as Used  
in FEM  
Mark S. Shephard, RPI
- 3:55 Panel - Open discussion by the participants of the  
workshop
- 5:00 Cocktail Reception (Faculty/Staff Dining Hall)

01/17/72  
P1

**WORKSHOP ON THE INTEGRATION OF  
FINITE ELEMENT MODELING  
WITH GEOMETRIC MODELING**

**INTRODUCTION TO WORKSHOP**

Mark S. Shephard  
Rensselaer Polytechnic Institute

**Purpose of Workshop**

To discuss the geometric modeling requirements of the finite element modeling process and to better understand the technical aspects of the integration of these two areas.

**Workshop Agenda**

Introduction to the Integration of Geometric and Finite Element Modeling; Mark S. Shephard, RPI

Integration of Geometric Modeling and Advanced Finite Element Preprocessing; Peter M. Finnigan, General Electric Corporate Research and Development

Integration Architecture of SDRC Geometry and Finite Element Processors; Lee Robie, Structural Dynamics Research Corporation

Finite Element Meshing of ANSYS Solid Models; F. Stanley Kelly, Swanson Analysis Systems



**Solid/FEM Integration at SNLA; Patrick F. Chavez,  
Sandia National Laboratories**

**Octree Based Automatic Meshing from CSG Models;  
Renato Perucchio, University of Rochester**

**Finite Octree Meshing Through Topologically Driven  
Geometric Operators; Kurt R. Grice, RPI**

**Design Modeling for Shape Optimization; Mark E.  
Botkin, General Motors Research Laboratories**

**Postprocessing Techniques for Three-dimensional  
Nonlinear Structures; Richard S. Gallagher, Hibbitt,  
Karlsson & Sorensen**

**Geometric Versus Finite Element Modeling - Current  
and Future Trends at Northrop; Shiv Bajaj, Northrop  
Corporation**

**Building F.E. Applications Using Non-manifold  
Boundary Operators, and the Generation of Idealized  
Models as Used in FEM; Mark S. Shephard, RPI**

**Panel - Open discussion by the participants of the  
workshop**

## **TERMINOLOGY**

To address the integration of geometric and finite element modeling we must be familiar with some of the terminology from both areas.

Some of the terms to be commonly used during the workshop are:

**Geometric Modeling** - A collection of procedures and representations, assumed here to be computerized, for the construction and description of the shape and spatial relations of objects.

**Wireframe Modeling** - The collection of the curve definitions for the edges of that bound an object.

**Surface Modeling** - The collection of curve and surface definitions for the edges and faces that bound an object.

**Solid Modeling** - The complete and unambiguous representation of three-dimensional objects in a computerized representation. There are six known families of unambiguous schemes known.

**Constructive Solid Geometry (CSG)** - An approach to solid modeling in which the object of interest is constructed by applying Boolean set operators to simple, well understood, solids. The two common sets of simple objects are halfspaces and primitive shapes such as blocks, cylinders, spheres, etc.

**Boundary Representations (B-reps)** - An approach to solid modeling in which the boundary entities that enclose the object, along with sufficient associativity information to unambiguously define the object, are stored.

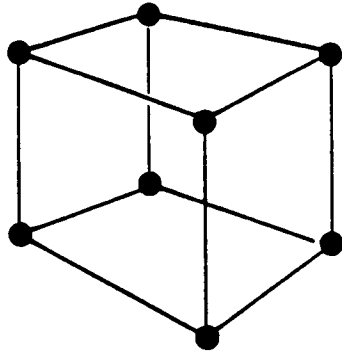
**Topology (with respect to geometric modeling)** - an abstract representation of an object that contains the associations of how they connect together. The topology of an object does not contain information on the shape of geometric entities.

**Manifold (2-manifold) solid representations** - every point on a surface has a neighborhood which is homeomorphic to a two-dimensional disk.

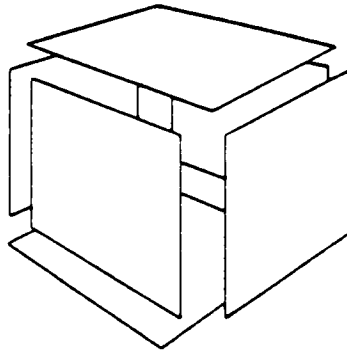
**Non-Manifold Geometric Modeling** - allows topological situations which are not 2-manifold. Neighborhood of a point on the surface need not be flat. Allows wire edges, dangling surfaces, and more than two faces to an edge.

**Static Geometric Interface** - A geometric interface in which a standardized file format is used to store the geometric representation of objects.

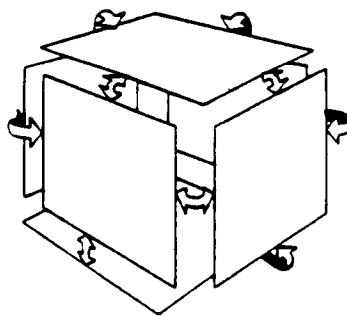
**Dynamic Geometric interface** - A geometric interface where both a description of the object as well as the functionality of the geometric modeling system used to define it are made available to application programs.



WIREFRAME



SURFACE



SOLID

Figure 3 - 1. Wireframe, surface, and solid modeling forms

---

FROM WEILER 1986

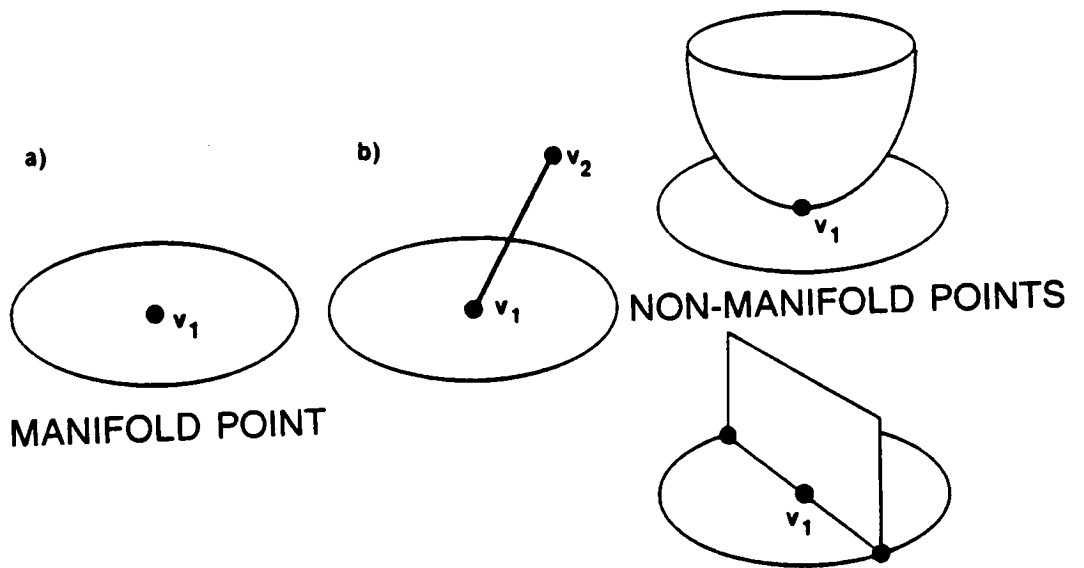


Figure 3 - 3. The 2-dimensional disk around points on a surface

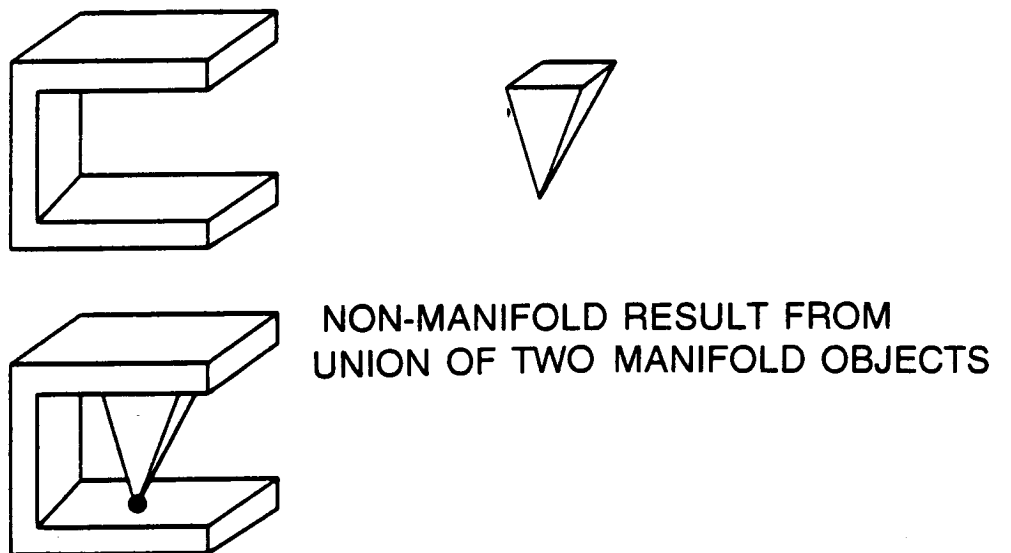


Figure 3 - 4. The Boolean union of two manifold objects yielding a non-manifold result

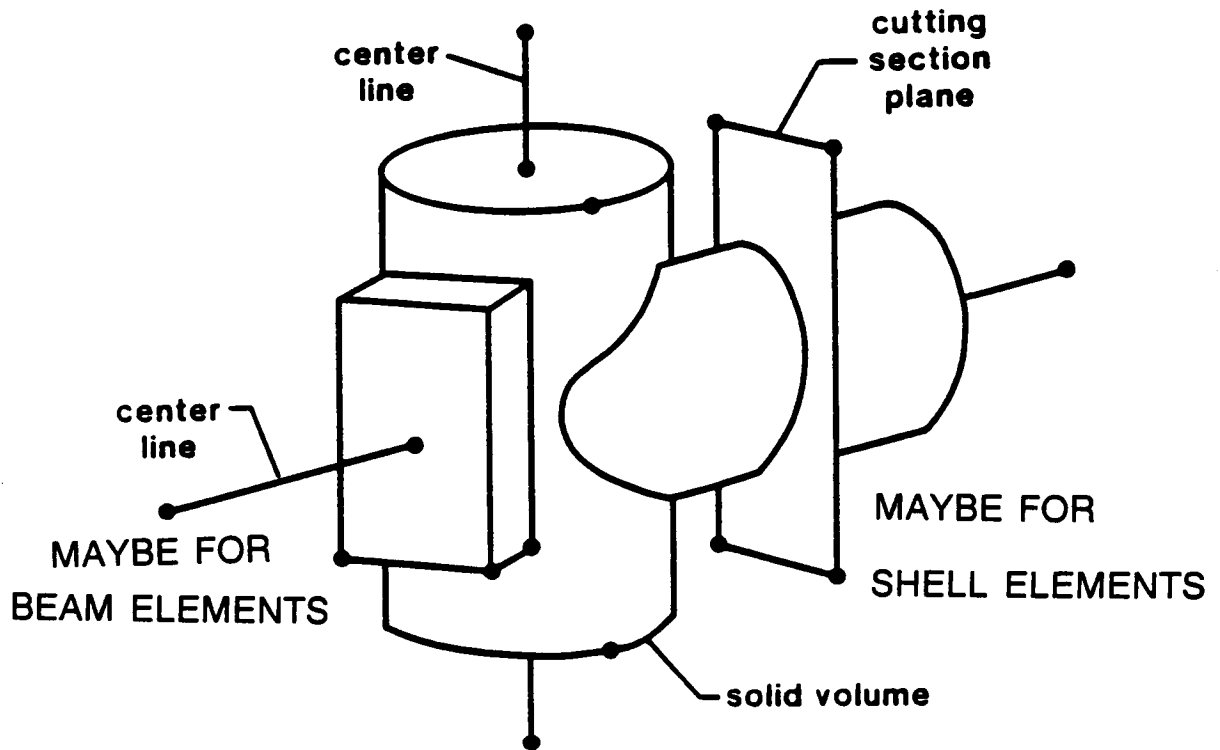


Figure 3 - 2. Example of a non-manifold geometric modeling form

**Geometric Operator** - A procedure which can be invoked by an applications program to have a geometric modeling function performed.

**Applications Interface Specification (CAM-I)** - A specification prepared by Computer Aided Manufacturing International containing an extensive list of geometric operators for use in the development of dynamic interfaces between geometric modeling systems and application programs. Test-bed versions of these operators have been implemented.

**Augmented Geometric Model** - A geometric representation plus associated data added that defines analysis attribute information needed for an application. For example the loads, material properties and boundary conditions needed for a finite element analysis.

**Idealized Model** - This is the model that is discretized into the finite element mesh that is then analyzed. This model is constructed from the augmented model by performing the geometric modifications desired to produce the model to be meshed.

**Analysis Attributes** - All information, past the base geometric model, needed to describe a physical problem in mathematical physics.

**Numerical Model Attributes** - All information past the augmented model specified needed to control the generation of the discrete numerical analysis model.

**Finite Element Modeling** - The process of going from an augmented geometric model to numerical solution results based on the use of finite element analysis procedures.

**Discretization** - The process of reducing a geometric object into a set of discrete entities as needed for a numerical analysis.

**Mesh Generators** - Procedures that can be used to discretize a geometric model into a finite element mesh.

**Adaptive Finite Element Procedures** - Techniques that employ solution results to determine where a finite element discretization needs to be altered to provide the desired degree of accuracy and the improvement of that discretization.

**A Posteriori Error Estimation** - The use of finite element solution results to estimate the discretization errors present in the current solution.

**Mesh Enrichment** - The improvement of a finite element model as dictated by the a posteriori error estimators and associated error indicators.

**Automated Finite Element Modeling** - A combination of algorithmic procedures capable of automatically performing the finite element modeling process, without used intervention, to provide solution results to a prespecified level of accuracy.



**CAM-I PROJECT ON DOCUMENTING  
THE GEOMETRIC MODELING REQUIREMENTS OF  
COUPLING GEOMETRIC MODELING SYSTEMS AND  
FINITE ELEMENT MODELING  
APPLICATIONS**

CAM-I is supporting RPI to perform this effort.

This workshop and the questionnaires you have been sent are being used as input to this effort

**QUESTIONNAIRE ON THE INTEGRATION  
OF GEOMETRIC MODELING AND  
FINITE ELEMENT MODELING**

Summary of questions

- Geometric modeling approaches
- Finite element meshing approaches
- Geometric modeling information and functionality
- Levels of integration
- Data used and saved
- Development of idealized models

Distribution of Responses to Date:

- 13 CAD/CAM Vendors
- 6 Research Groups
- 9 End User Groups

## **INITIAL QUESTIONNAIRE SUMMARY**

Most responses indicated that they utilized a boundary representation for modeling. Implicit and parametric storage schemes were both popular.

Future choices of model representation also favored the B-rep scheme. Some with current CSG approaches also expect to handle B-rep.

Most responses indicated that they had an integration between a finite element preprocessing package and the geometric modeler. In addition, most felt it was inadequate.

Transfer schemes varied from IGES, tight integration with the geometric modeler, and ad-hoc schemes developed within their own organizations.

The finite element preprocessors dependence on a topological data structure for the geometric modeler (vertex, edge, etc.) was nearly unanimous, even for those interfacing to CSG based model representations.

## **INITIAL QUESTIONNAIRE SUMMARY**

Commercial preprocessing capabilities, in general, are still dominated by the use of interactive mesh generators.

Future capabilities of preprocessors will utilize more automatic techniques for generating numerical analysis models.

The desire to link the finite element mesh back to the geometric model was also apparent. This may put additional demands on modelers or interfacing techniques. Typical reasons for this link were to allow for integrated optimization techniques and adaptive analysis.

Attributes such as loads, boundary conditions, and material conditions should be applied to the geometric model and later 'sent off' to the analysis with the resulting mesh.

## **INITIAL QUESTIONNAIRE SUMMARY**

One area of little response was the ability to define the idealized models. As a finite element analysis vendor indicated, the assumption is made that the model made was specifically for meshing and that no unnecessary details exist.

Current techniques to idealize the model are manual in nature. Either by developing the model in a restricted sense via some FEM commercial package, or by simply eliminating the detail to be ignored within the solid modeler.

Those that responded indicated a desire for some automatic means of developing the idealized model.

51-61  
105787  
17.8.

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 create 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 made 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 process 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 yields 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 led 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 tetrahedral 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 from 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-manifold 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. Bolssonat 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 Generation", *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 Computational 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.



52-61  
129733  
hjs

APPROACHES TO THE AUTOMATIC GENERATION  
AND CONTROL OF FINITE ELEMENT MESHES

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

**ABSTRACT**

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

**INTRODUCTION**

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

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

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

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

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

### ALGORITHMIC APPROACHES TO AUTOMATIC MESH GENERATION

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

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

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

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

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

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

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

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

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

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

#### Point Placement Followed by Domain Triangulation

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

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

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

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

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

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

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

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



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

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

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

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

### **Mesh Generation Based on Sub-Domain Removal**

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

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

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

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

the singularity. The remaining operators, VERTEX\_REMOVAL, VERTEX\_REMOVAL\_WITH\_EDGE\_SPLIT, and EDGE\_REMOVAL, are used to mesh the rest of the domain.

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

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

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

### **Mesh Generation by Recursive Subdivision**

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

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

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

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

### **Spatial Decomposition Followed by Subdomain Meshing**

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

inside nor outside the object.

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

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

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

### Speed of Automatic Mesh Generators

The limited experience available to date indicates that the amount of computation needed to generate a mesh of a few thousand elements for a general three-dimensional geometry will be of the same order of magnitude as a linear analysis carried out on that system. Therefore, the computational efficiency of these procedures is of critical importance. The two measures of computational efficiency of importance are the time required by the given algorithms to generate comparable meshes and, even more importantly, the computational growth rate of the mesh generator. Tests run to date on complex two-dimensional geometries indicates that the implementation of various approaches yields speed differences that vary by more 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 [41], indicate an  $O(n \log(\log n))$ , where  $n$  is the number of points, computational time as being possible. (In two-dimensions the number of elements is of the same order as the number of nodes [42].) Computational results of an implemented three-dimensional algorithm gave  $O(n^{5/3})$  computer times [11]. (In the three-dimensional case, the number of elements can be from  $O(n)$  to  $O(n^2)$  [42]. However, it



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

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

### **A Priori Control of Element Distributions**

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

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

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

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

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

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

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

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

#### **INTEGRATION OF AUTOMATIC MESH GENERATORS WITH GEOMETRIC MODELERS**

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

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

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

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

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

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

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

Although the algorithmic approaches to automatic mesh generation and

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

#### **ADAPTIVE ANALYSIS AND A POSTERIORI MESH CONTROL FOR AUTOMATIC MESH GENERATORS**

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

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

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

indicators.

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

#### **Overview of A Posteriori Error Estimation**

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

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

$$\theta_{(k_j)} = \frac{\|E_j\|}{\|u-u_j\|} \quad (1)$$

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

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

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

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



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

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

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

subject to

$$u(x,y) = 0 \quad (x,y) \in \partial\Omega_1 \quad (3b)$$

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

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

where

$\Omega$  is a bounded region in  $R^2$

$\partial\Omega$  is the boundary of  $\Omega$

$\eta$  is the unit outward normal to  $\partial\Omega$

$a(x,y)$ ,  $b(x,y)$  and  $f(x,y)$  are given functions meeting the necessary smoothness criteria subject to  $a(x,y) > 0$  and  $b(x,y) \geq 0 \quad \forall (x,y) \in \Omega$

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

$$A(u,v) = (f,v) + \langle q,v \rangle_{\partial\Omega_2} \quad \forall v \in H_0^1 \quad (4a)$$

where

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

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

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

and  $H_0^1$  is the set of all functions contained in  $H^1$  which are zero on  $\partial\Omega_1$ . Recall that the space  $H^1$  contains all functions for which the

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

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

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

where the basis function selected for  $U$  and  $V$  are piecewise polynomials defined over individual elements of the triangulation of the domain  $\Omega$  such that  $C^0$  inter-element continuity [50,56] is maintained. This allows the integrals in equation (5) to be carried out over the individual finite elements,  $\Omega_i$ , and then properly summed.

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

$$A(e, v) = (f, v) + \langle q, v \rangle_{\partial \Omega_2} - A(U, v) \quad \forall v \in H_0^1 \quad (6)$$

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

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

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

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

(8)

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

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

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

$$A(e,v)_{\Omega_i} = (f,v)_{\Omega_i} + \langle q,v \rangle_{\partial\Omega_{2i}} - A(U,v)_{\Omega_i} + \langle au_{\eta}, v \rangle_{\partial\Omega_i} \quad \forall v \in H_0^1 \quad (9)$$

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

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

$$\langle au_{\eta}, v \rangle_{\partial\Omega_i} = \int_{\partial\Omega_i} au_{\eta} v \, ds \quad (9d)$$

where

$\Omega_i$  is the domain of the element,  $\partial\Omega_i$  is its boundary,  $u_{\eta}$  is the normal derivative of  $u$  on the element boundary.

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

$$A(E, v^*)_{\Omega_i} = (f, v^*)_{\Omega_i} + \langle q, v^* \rangle_{\partial\Omega_i} - A(U, v^*)_{\Omega_i} + 1/2 \langle a(U_\eta^+ + U_\eta^-), v^* \rangle_{\partial\Omega_i} \\ \forall v^* \in S_t^* \quad (10)$$

where  $U_\eta^+$  and  $U_\eta^-$  denote the value of the normal derivatives on either side of the edge.

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

$$A(e, v)_{\Omega_i} = A_r(U, v)_{\Omega_i} + \langle q, v \rangle_{\partial\Omega_{2i}} - \langle aU_\eta, v \rangle_{\partial\Omega_i} + \langle au_\eta, v \rangle_{\partial\Omega_i} \\ \forall v \in H_0^1 \quad (11a)$$

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

$$A_r(U, v)_{\Omega_i} = \int_{\Omega_i} (\nabla(a\nabla U)v - bUv + fv) \, d\Omega \quad \forall v \in H_0^1 \quad (11b)$$

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

$$\langle \Delta aU_\eta, v \rangle_{\partial\Omega} = \left[ \begin{array}{l} \langle a(u_\eta - u_\eta), v \rangle_{\partial\Omega_i, \partial\Omega_i \& \partial\Omega_2} \\ \langle (q - aU_\eta), v \rangle_{\partial\Omega_i, \partial\Omega_i \& \partial\Omega_2} \end{array} \right] \quad (12)$$

where

$$\begin{aligned} \langle a(u_\eta - U_\eta), v \rangle_{\partial\Omega_i} &= \int_{\partial\Omega_i} a(u_\eta - U_\eta) v \, ds \\ \langle (q - aU_\eta), v \rangle_{\partial\Omega_i} &= \int_{\partial\Omega_i} (f - aU_\eta) v \, ds \end{aligned}$$

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

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

where

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

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

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

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

### Mesh Improvement in Adaptive Analysis

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

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

$$E^\alpha = \sum \eta_i^\alpha \tag{14}$$

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

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

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

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

procedures [66].

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

#### **Automatic Mesh Generators and A Posteriori Mesh Control**

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

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

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

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

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

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

process can lead to substantial computational savings).

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

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

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

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

#### CONCLUDING REMARKS

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

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

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

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

#### ACKNOWLEDGEMENTS

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

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

#### REFERENCES

1. A.A.G. Riquicha and H.B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment", IEEE Computer Graphics and Applications, Vol. 3, No. 2, 1982, pp. 9-24.
2. A.A.G. Riquicha and H.B. Voelcker, "Solid Modeling: Current Status and Research Directions", IEEE Computer Graphics and Applications, Vol. 3, No. 7, October 1983, pp. 25-37.
3. M.S. Pickett and J.W. Boyse, Eds., Solid Modeling by Computers: From Theory to Applications, Plenum Press, 1984.
4. O.C. Zienkiewicz and D.V. Phillips, "An Automatic Mesh Generation Scheme for Plane and Curved Surfaces by Isoparametric Co-ordinates", Int. J. Num. Meth. Engng., Vol. 3,



- No. 4, 1971, pp. 519-528.
5. W.J. Gordon and C.A. Hall, "Construction of Curvilinear Co-ordinates Systems and Applications to Mesh Generation", Int. J. Num. Meth. Engng., Vol. 7, 1973, pp. 461-477.
  6. J.E. Thompson, Numerical Grid Generation, North-Holland, 1982.
  7. W.A. Cook, "Body Oriented (Natural) Co-ordinates for Generating Three Dimensional Meshes", Int. J. Num. Meth. Engng., Vol. 8, 1974, pp. 27-43.
  8. R.B. Haber, M.S. Shephard, J.F. Abel, R.H. Gallagher, and D.P. Greenberg, "A Generalized Two-Dimensional Graphical Finite Element Preprocessor Utilizing Discrete Transfinite Mappings", Int. J. Num. Meth. Engng., Vol. 17, 1981, pp. 1015-1044.
  9. W.R. Buell and B.A. Bush, "Mesh Generation - A Survey", Trans. ASME J. of Engng. for Industry, Vol. 95, pp. 332-338, 1973.
  10. C.O. Frederick, Y.C. Wong and F.W. Edge, "Two-Dimensional Automatic Mesh Generation for Structural Analysis", Int. J. Num. Meth. Engng., Vol. 2, 1970, pp. 113-144..
  11. J.C. Cavendish, D.A. Field and W.H. Frey, "An Approach to Automatic Three-Dimensional Mesh Generation", Int. J. Num. Meth. Engng., Vol. 21, 1985, pp. 329-347.
  12. J.C. Cavendish, "Automatic Triangulation of Arbitrary Planar Domains for the Finite Element Method", Int. J. Num. Meth. Engng., Vol. 8, 1974, pp. 679-697.
  13. Nguyen-Van-Phai, "Automatic Mesh Generation with Tetrahedron Elements", Int. J. Num. Meth. Engng., Vol. 18, 1982, pp. 273-289.
  14. R. Sibson, "Locally Equiangular Triangulations", The Computer Journal, Vol. 21, No. 3, 1978, pp. 243-245.
  15. D. F. Watson, "Computing the n-Dimensional Delaunay Tessellation with Applications to Voronoi Polytypes", The Computer Journal, Vol. 24, No. 2, 1981.
  16. Y.T. Lee, A. de Pennington and N.K. Shaw, "Automatic Finite Element Mesh Generation from Geometric Models - A Point-Based Approach", ACM Transactions on Graphics, Vol. 3, 1984, pp. 287-311.
  17. S.H. Lo, "A New Mesh Generation Scheme for Arbitrary Planar Domains", Int. J. Num. Meth. Engng., Vol. 21, 1985, pp. 219-249.
  18. D.A. Field, "Implementing Watson's Algorithm in Three Dimensions", Proc. Second Annual ACM Symposium on Computational

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

19. D.A. Field and W.H. Frey, "Automation of Tetrahedral Mesh Generation", Research Publication GMR-4967, General Motors Research Laboratories, Warren, MI, 1985.
20. B. Wordenweber, "Finite Element Mesh Generation", Computer-Aided Design, Vol. 16, 1984, pp. 285-291.
21. T.C. Woo and T. Thomasa, "An Algorithm for Generating Solid Elements in Objects with Holes", Computers and Structures, Vol. 18, No. 2, 1984, pp. 333-342.
22. E.A. Sadek, "A Scheme for the Automatic Generation of Triangular Finite Elements", Int. J. Num. Meth. Engng., Vol. 15, 1980, pp. 1813-1822.
23. F.T. Tracy, "Graphical Pre- and Post-Processors for Two-Dimensional Finite Element Programs", Computer Graphics, Transactions of ACM, Vol. 13, 1977, pp. 8-12.
24. M.S. Shephard, K.R. Grice and M.K. Georges, "Some Recent Advances in Automatic Mesh Generation", Modern Methods for Automating Finite Element Mesh Generation, K. Baldwin, Ed., ASCE, NY, 1986, 1-18.
25. A. Bykat, "Automatic Generation of Triangular Grid: I - Subdivision of General Convex Subregions, II - Triangulation of Convex Polygons", Int. J. Num. Meth. Engng., Vol. 10, 1976, pp. 1329-1342.
26. B. Joe and R.B. Simpson, "Triangular Meshes for Regions on Complicated Shapes", Int. J. Num. Meth. Engng., Vol. 23, 1986, pp. 751-778.
27. P.F. Charvez, "Automatic Mesh Generation and Optimization from the Solid Model Database", Modern Methods for Automating Finite Element Mesh Generation, K. Baldwin, Ed., ASCE, NY, 1986, pp. 29-42.
28. I. Babuska and E. Rank, "An Expert-System-Like Approach in the hp-Version of the Finite Element Method", Institute for Physical Science and Technology Lab. for Num. Analysis, TN BN-1048., University of Maryland, 1986.
29. B.A. Szabo, "Mesh Design for the p-Version of the Finite Element Method", Computer Meth. in App. Mech. and Engng., Vol. 55, 1986, pp. 181-197.
30. M.L.C. Sluiter and D.L. Hansen, "A General Purpose Two- and Three-Dimensional Mesh Generator", Computers in Engineering, Vol. 3, L.E. Hulbert, Ed., Book No. G00217, ASME, 1982, pp. 29-34.

31. A.J.C. Schoofs, L.H.Th.M. Van Beukering and M.L.C. Sluiter, "A General Purpose Two-Dimensional Mesh Generator", Advances in Engineering Software, Vol. 1, No. 3, 1979, pp. 131-136.
32. M.A. Yerry and M.S. Shephard, "Finite Element Mesh Generation Based on a Modified-Quadtree Approach", IEEE Computer Graphics and Applications, Vol. 3, No. 1, 1983, pp. 36-46.
33. M.S. Shephard and M.A. Yerry, "Approaching the Automatic Generation of Finite Element Meshes", Computers in Mech. Engng., Vol. 1, No. 4, 1983, pp. 49-56.
34. P.L. Baehmann, S.L. Wittchen, M.S. Shephard, K.R. Grice and M.A. Yerry", Robust Geometrically-Based Automatic Two-Dimensional Mesh Generation", TR-86007, Center for Interactive Computer Graphics, RPI, Troy, NY, 1986, to appear, Int. J. Num. Meth. Engng..
35. A. Kela, R. Perucchio and H.B. Voelcker, "Towards Automatic Finite Element Analysis", Computers in Mech. Engng., July 1986, pp. 57-71.
36. M.A. Yerry and M.S. Shephard, "Automatic Three-Dimensional Mesh Generation for Three-Dimensional Solids", Computers and Structures, Vol. 20, 1985, pp. 173-180.
37. M.A. Yerry and M.S. Shephard, "Automatic Three-Dimensional Mesh Generation by the Modified-Octree Technique", Int. J. Num. Meth. Engng., Vol. 22, 1984, pp. 1965-1990.
38. M.S. Shephard, M.A. Yerry and P.L. Baehmann, "Automatic Mesh Generation Allowing for Efficient A Priori and A Posteriori Mesh Refinements", Computer Meth. in Appl. Mech. Engng., Vol. 55, 1986, pp. 161-180.
39. C.L. Jackins and S.L. Tanimoto, "Octrees and Their Use in the Representation of Three-Dimensional Objects", Compt. Graph. Image Process., Vol. 14, 1980, pp. 249-270.
40. D. Meagher, "Geometric Modeling Using Octree Encoding", Computer. Graph. Image Process., Vol. 19, 1982, pp. 129-147.
41. R.A. Dwyer, "A Simple Divide-and-Conquer Algorithm for Constructing Delaunay Triangulations in  $O(n \log \log n)$  Expected Time", Proc. Second Annual ACM Symposium on Computational Geometry, ACM 0-89791-194/6/86/0600/276, 1986, pp. 276-284.
42. J.D. Bolsonnat and M. Tellaud, "A Hierarchical Representation of Objects: The Delaunay Tree", Proc. Second Annual Symposium on Computational Geometry, ACM 0-89791-194- 6/86/0600/260, 1986, pp. 260-268.
43. I. Babuska, O.C. Zienkiewicz, J. Gago and E.R. De A. Oliveria,

- Accuracy Estimates and Adaptive Refinements in Finite Element Computations, John Wiley and Sons, Chichester, 1986.
44. K.J. Weiler, "Edge Based Data Structures for Solid Modeling in Curved-Surface Environments", IEEE Computer Graphics and Applications, Vol. 5, No. 1, January 1985, pp. 21-40.
  45. K.J. Weiler, "Topological Structures for Geometric Modeling", PhD Thesis, Center for Interactive Computer Graphics, TR-86032, Rensselaer Polytechnic Institute, Troy, NY, 1986.
  46. P.R. Wilson, "Data Transfer and Solid Modeling", Geometric Modeling for CAD Applications, M.J. Wozny, H.W. McLaughlin and J.L. Encarnacao, Eds., North Holland, to appear.
  47. "Applications Interface Specification (Restructured Version)", CAM-I Report R-86-GM-01, January 1986.
  48. M.S. Shephard, "Finite Element Modeling within an Integrated Geometric Modeling Environment: Part I - Mesh Generation", Engineering with Computers, Vol. 1, 1985, pp. 61-71.
  49. M.S. Shephard and P.M. Finnigan, "Integration of Geometric Modeling and Advanced Finite Element Preprocessing", Proc 4th Chautauqua on Productivity in Engineering and Design...The Quest for Quality, H. Shaeffer, Ed., PDA Eng., Los Angeles, CA, 1986, pp. 231-233.
  50. G. Strang and G. Fix, An Analysis of the Finite Element Method, Prentice Hall, 1973.
  51. I. Babuska and W.C. Rheinboldt, "A Posteriori Error Estimate for the Finite Element Method", Int. J. Num. Meth. Engng., Vol. 12, 1978, pp. 1597-1615.
  52. I. Babuska, W.C. Rheinboldt, "Error Estimates for Adaptive Finite Element Computations", SIAM J. Numer. Anal., Vol. 15, No. 4, 1978, pp. 736-754.
  53. I. Babuska, "A Posteriori Error Estimates and Adaptive Approaches for Finite Element Modeling", Finite Element Workshop 1980, Technical Note BN-940, I. Babuska, Ed., Laboratory for Numerical Analysis, U. of Maryland, May, 1980.
  54. Proceedings of Int. Conf. on Accuracy Estimates and Adaptive Refinements in Finite Element Computations, Vol. 1 and 2, Int. Association of Computational Mechanics, 1984.
  55. S. Adjerid and J.E. Flaherty, "Local Refinement Finite Element Methods on Stationary and Moving Meshes for One-Dimensional Parabolic Systems", to appear.
  56. J.N. Reddy, An Introduction to the Finite Element Method, McGraw-Hill Book Co., NY, 1984.

57. I. Babuska and A. Miller, "A Posteriori Error Estimates and Adaptive Techniques for the Finite Element Method", Institute for Physical Science and Technology, Laboratory for Numerical Analysis, Tech. Note BN-968, University of Maryland, 1981.
58. S. Adjerid and J.E. Flaherty, "A Moving Finite Element Method for Time Dependent Partial Differential Equations with Error Estimation and Refinement", SAMI Numer. Anal., Vol. 23, pp. 778-796, 1985.
59. S. Adjerid and J.E. Flaherty, "A Moving Mesh Finite Element Method with Local Refinement for Parabolic Partial Differential Equations", Comp. Meths. Appl. Mech. Engng., 1986, pp. 3-26.
60. R.E. Bank, "Analysis of a Local A Posteriori Error Estimate for Elliptic Equations", Accuracy Estimates and Adaptive Refinements in Finite Element Computations, I. Babuska, O.C. Zienkiewicz, J. Gago and E.R. de A. Oliveria, Eds., 1986, pp. 119-128.
61. O.C. Zienkiewicz, J. Gago and D.W. Kelly, "The Hierarchical Concept in Finite Element Analysis", Computers and Structures, Vol. 16, 1983, pp. 53-65.
62. I. Babuska and D. Yu, "Asymptotically Exact A Posteriori Error Estimates and Adaptive Approaches for Biquadratic Elements", Tech. Note BN-1050, Institute for Physical Science and Technology, U. of Maryland, 1986.
63. I. Babuska and D. Yu, "A Posteriori Error Estimation for Biquadratic Elements and Adaptive Approaches", to appear.
64. S. Adjerid and J.E. Flaherty, "Second-Order Finite element Approximations and A Posteriori Error Estimation for Two-Dimensional Parabolic Systems", Report No. 87-1, Department of Computer Science, Rensselaer Polytechnic Institute, 1987.
65. I. Babuska and W.C. Reinboldt, "Analysis of Optimal Finite Element Meshes in  $R^1$ ", Math. of Comp., Vol. 33, 1979, pp. 431-463.
66. O.C. Zienkiewicz and A. Craig, "Adaptive Refinement, Error Estimates, Multigrid Solution and Hierarchic Finite Element Method Concepts", Accuracy Estimates and Adaptive Refinements in Finite Element Computations, I. Babuska, O.C. Zienkiewicz, J. Gago and E.R. de A. Oliveria, Eds., 1986, pp. 25-59.
67. D.J. Turcke and G.M. McNeice, "Guidelines for Selecting Finite Element Grids Based on an Optimization Study", Computers and Structures, Vol. 4, 1974, pp. 449-519.
68. C.A. Fillipa, "Optimization of Finite Element Grids by Direct Energy Search", Appl. Math. Modeling, Vol. 1, September 1976,

pp. 93-96.

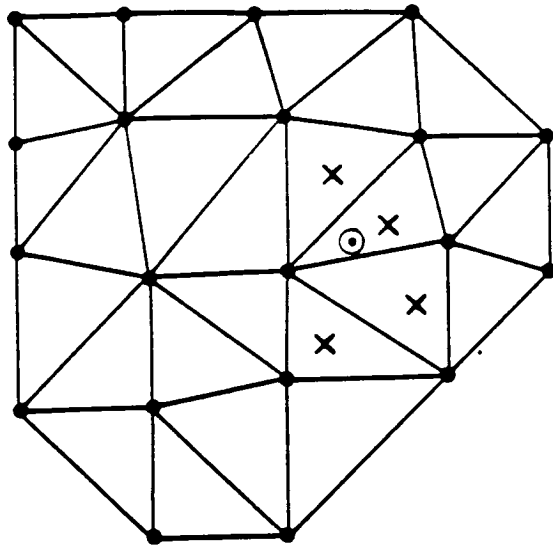
69. K. Miller, "Recent Results on Finite Element Methods with Moving Nodes", Accuracy Estimates and Adaptive Refinements in Finite Element Computations, I. Babuska, O.C. Zienkiewicz, J. Gago and E.R. de A. Oliveria, Eds., 1986, pp. 225-338.
70. R.J. Melosh and P.V. Marcal, "An Energy Basis for Mesh Refinement of Structural Continua", Int. J. Num. Meth. Engng., Vol. 11, No. 7, 1977, pp. 1083-1092.
71. W.C. Rheinboldt and C.K. Mesztenyi, "On a Data Structure for Adaptive Finite Element Mesh Refinements", ACM Transaction on Maths. Software, Vol. 6, No. 2, June 1980, pp. 166-187.
72. W.C. Rheinboldt, "Adaptive Mesh Refinement Processes for Finite Element Solutions", Int. J. Num. Meth. Engng., Vol. 17, 1981, pp. 649-662.
73. R.E. Ewing, "Adaptive Mesh Refinements in Large-Scale Fluid Flow Simulation", Accuracy Estimates and Adaptive Refinements in Finite Element Computations, I. Babuska, O.C. Zienkiewicz, J. Gago and E.R. de A. Oliveria, Eds., 1986, pp. 229-314.
74. R.E. Bank, A.H. Sherman and A. Weiser, "Refinement Algorithms and Data Structures for Regular Local Refinement", Scientific Computing: Applications of Mathematics and Computing to the Physical Sciences, R.S. Stepleman, Ed., North Holland, 1983, pp. 3-17.
75. M.C. Rivara, "Algorithms for Refining Triangular Grids Suitable for Adaptive and Multigrid Techniques", Int. J. Num. Meth. Engng., Vol. 20, 1984, pp. 745-756.
76. M.S. Shephard, J.E. Flaherty and P.L. Baehmann, "Adaptive Analysis for Automated Finite Element Modeling", to appear, Proc. of The Mathematics of Finite Elements and Application, 1987.
77. A.G. Peano, "Hierarchies of Conforming Finite Elements for Plane Elasticity and Plate Bending", Comp. Math. with Appl., Vol. 2, 1976.
78. A. Peano, R. Riccioni, A. Pasini and L. Sardella, "Adaptive Approximations in Finite Element Structural Analysis", Computers and Structures, Vol. 10, 1979, pp. 333-342.
79. I. Babuska and B.A. Szabo, "On the Rates of Convergence of the Finite Element Method", Int. J. Num. Meth. Engng., Vol. 18, 1982, 323-341.
80. I. Babuska and M. Dorr, "Error Estimates for the Combined h and p Versions of the Finite Element Method", Numer. Math., Vol. 25, 1981, pp. 257-277.

81. M.S. Shephard, R.H. Gallagher and J.F. Abel, "Synthesis of Near-Optimum Finite Element Grids with Interactive Computer Graphics", Int. J. Num. Meth. Eng., Vol. 15, 1980, pp. 1021-1039.
82. J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz, "Adaptive Remeshing of Compressible Flow Computations", Institute for Numerical Methods in Engineering, CR/R/544/86, U. College Swansea, Swansea, Wales, 1986.
83. B. Guo and I. Babuska, "The h-p Version of the Finite Element Method - Part 1: The Basic Approximation Results; Part II - General Results and Applications", to appear, Computational Mechanics.
84. W.H. Frey, "Selective Refinement: A New Strategy for Automatic Node Placement in Graded Triangular Meshes", GM Research Publication, GMR-5432, 1986.
85. J. Penman and M.D. Grive, "An Approach to Self-Adaptive Mesh Generation", IEEE Transactions on Magnetics, Vol. MAG-21, No. 6, 1985, pp. 2567-2570.

## FIGURE CAPTIONS

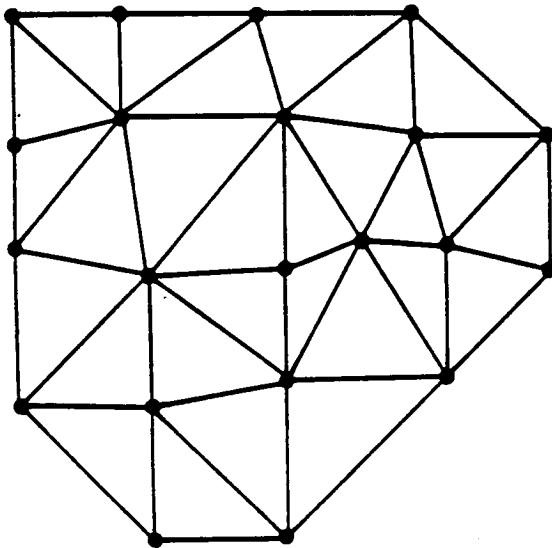
1. Watson's Algorithm for inserting a point into a Delaunay triangulation
2. Basic three-dimensional element removal operators
3. h-p mesh generated by element removal
4. Subdomain removal to decompose object into mappable regions
5. Finite octree mesh example
6. Finite quadtree mesh control
  - a) uniform mesh
  - b) graded mesh
  - c) mesh control parameters for graded mesh
7. Finite octree mesh control
  - a) uniform mesh
  - b) graded mesh
8. h-refinement by element bisection
  - a) quadrilateral element
  - b) triangular element
9. Finite quadtree mesh refinement by local remeshing
  - a) initial mesh
  - b) affected portion of mesh removed
  - c) refined quadrants
  - d) resulting refined mesh





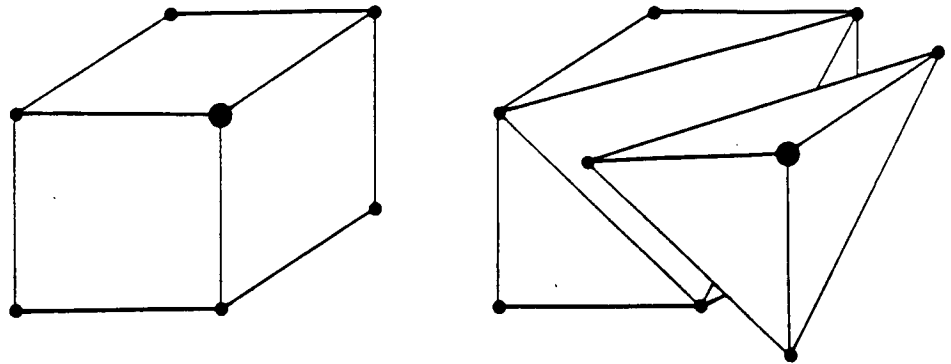
- - node
- × - element flagged for deletion
- ⊙ - new node being inserted

a) original mesh with new node inserted

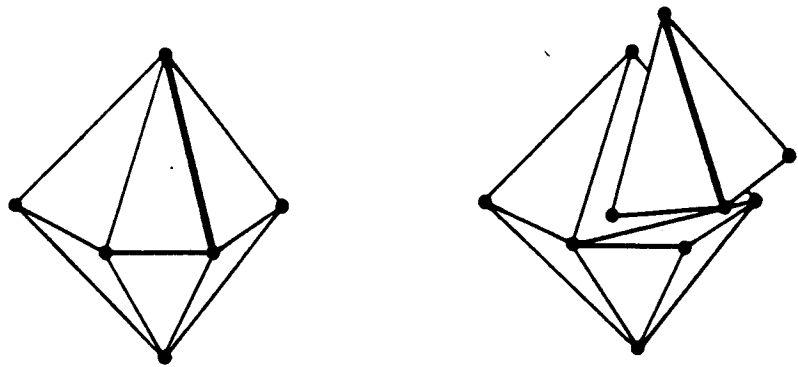


b) resulting Delaunay triangulation

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



a) vertex removal



b) edge removal

FIG. 2. Basic three-dimensional element removal operators

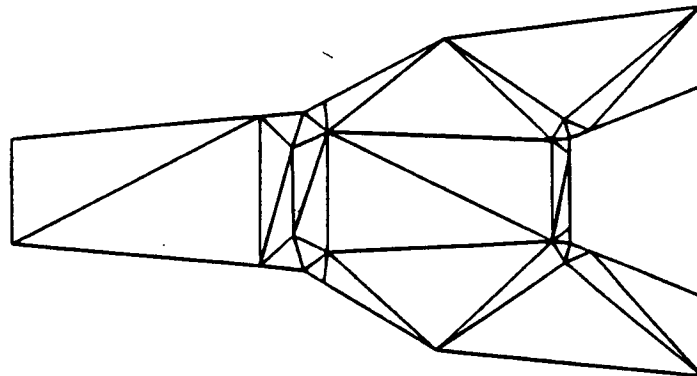
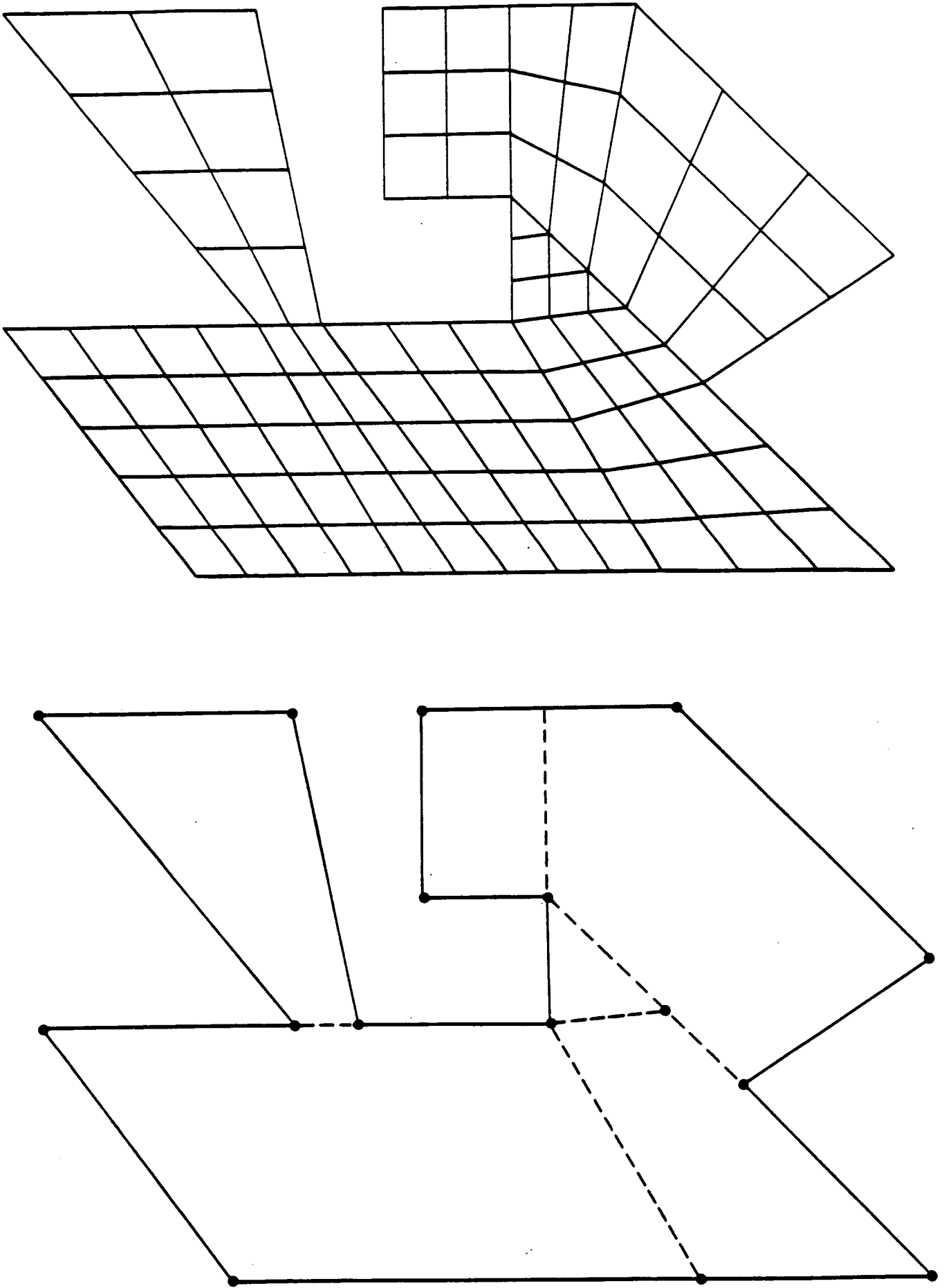


FIG. 3. h-p mesh generated by element removal



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

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

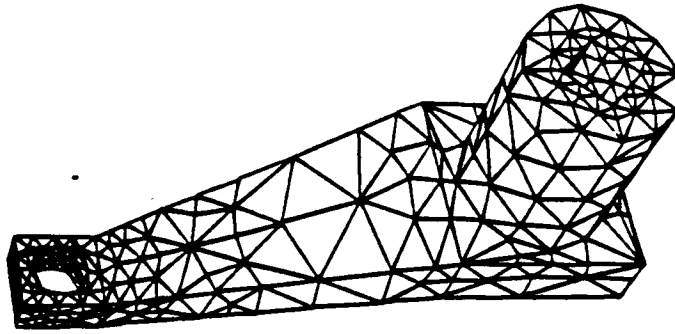
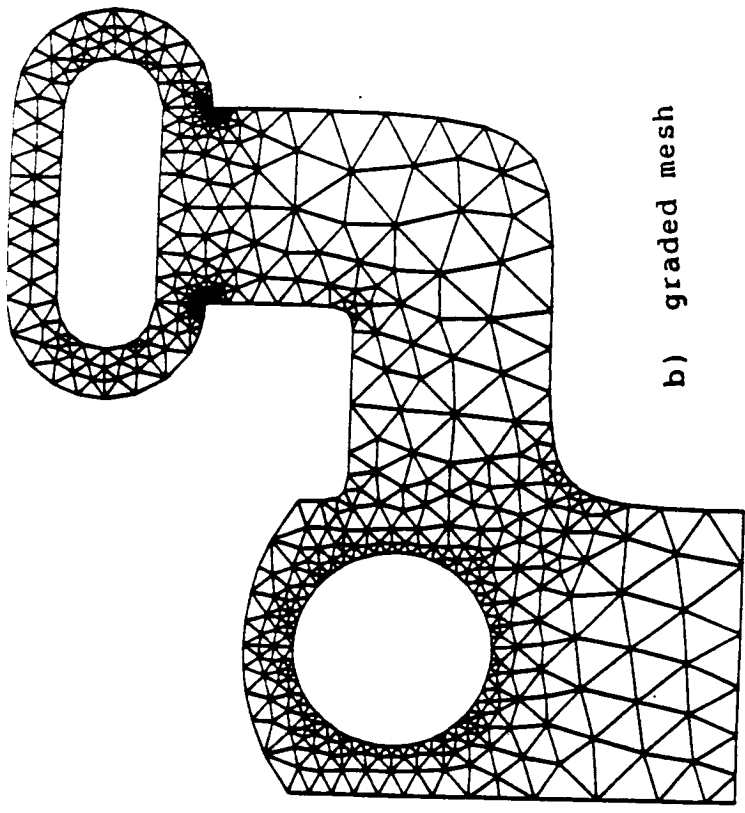
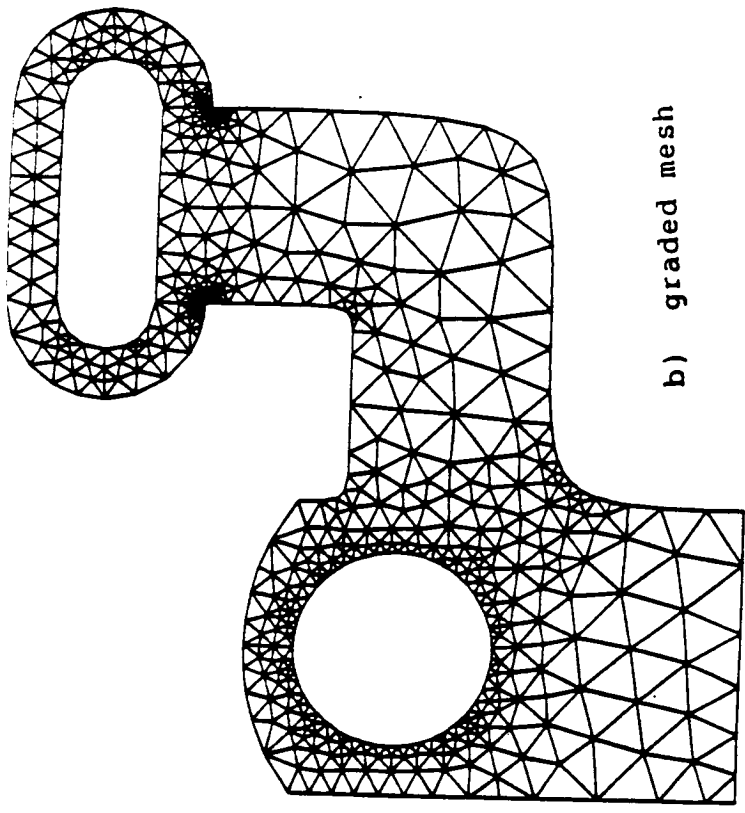


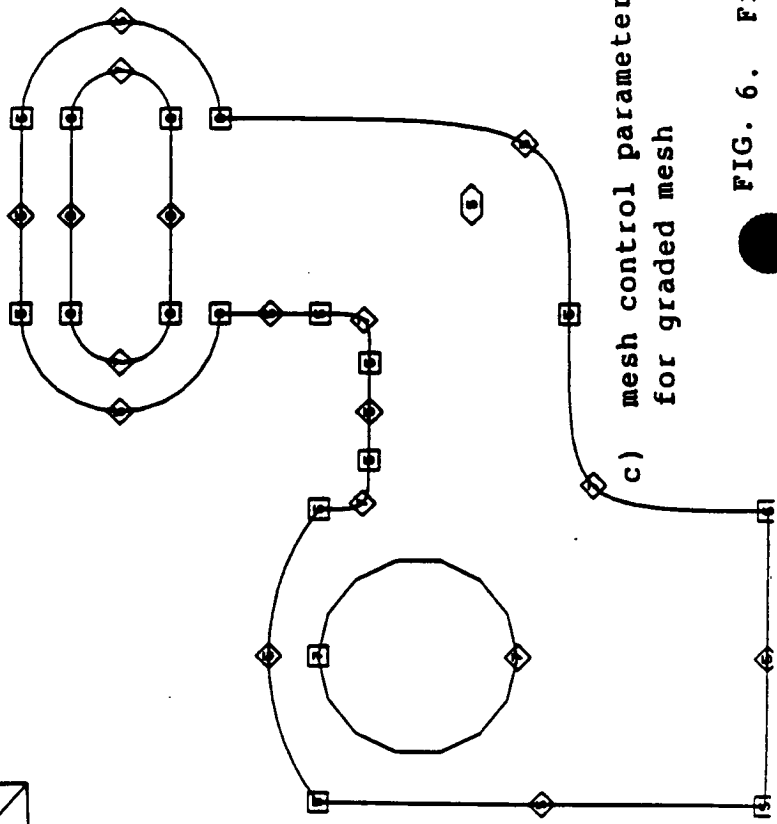
FIG. 5. Finite octree mesh example



a) uniform mesh

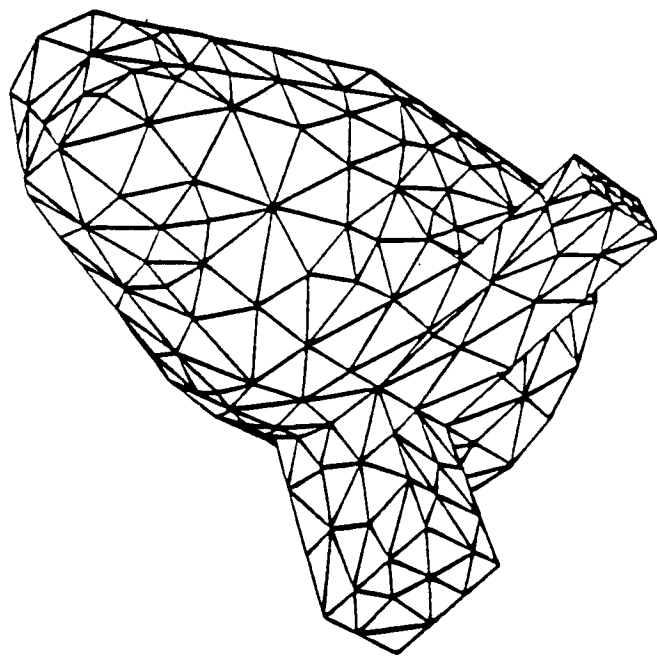


b) graded mesh

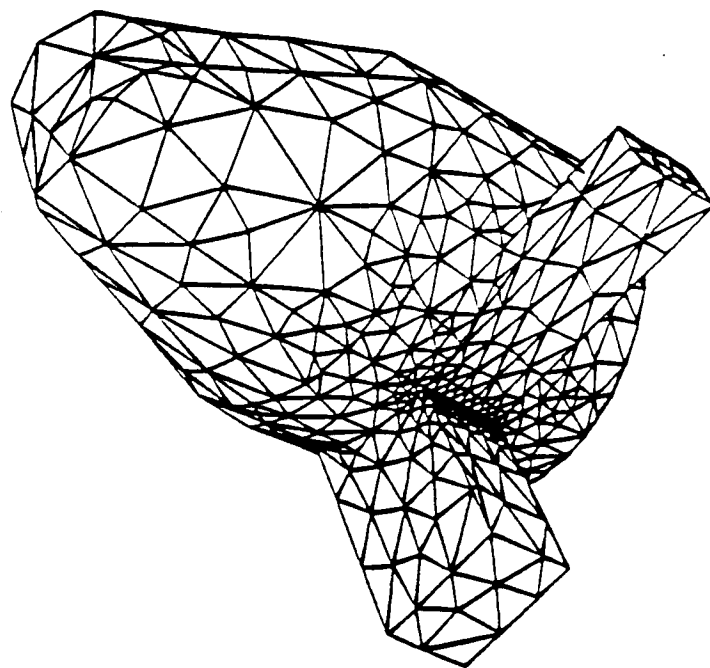


c) mesh control parameters for graded mesh

FIG. 6. Finite quadtree mesh control

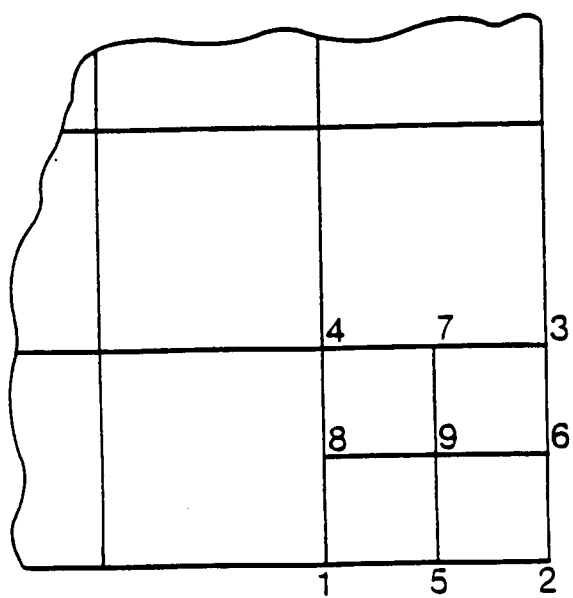


a) uniform mesh

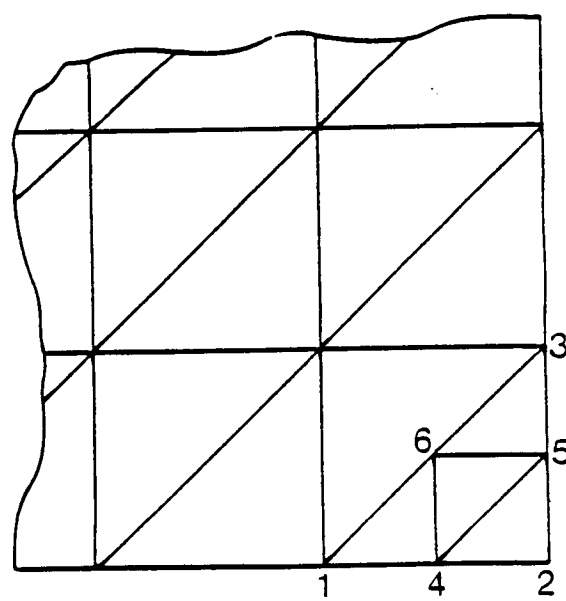


b) graded mesh

FIG. 7. Finite octree mesh control

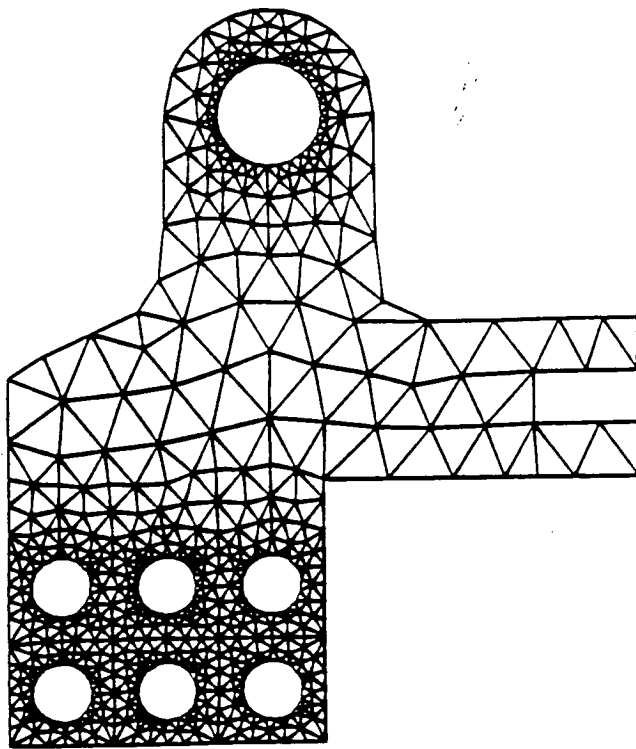


a) quadrilateral element

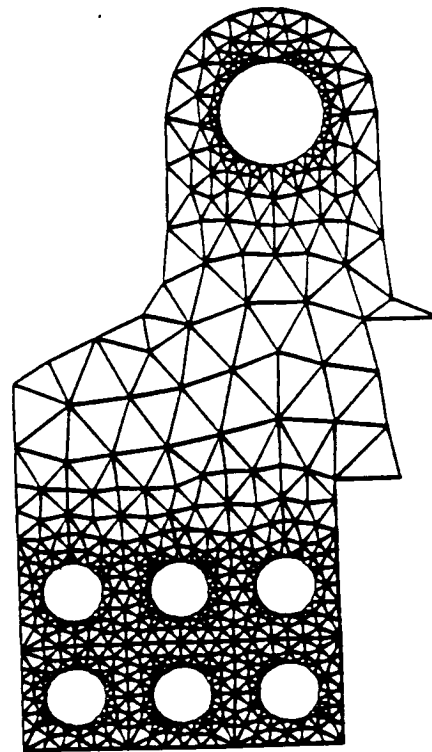


b) traingular element

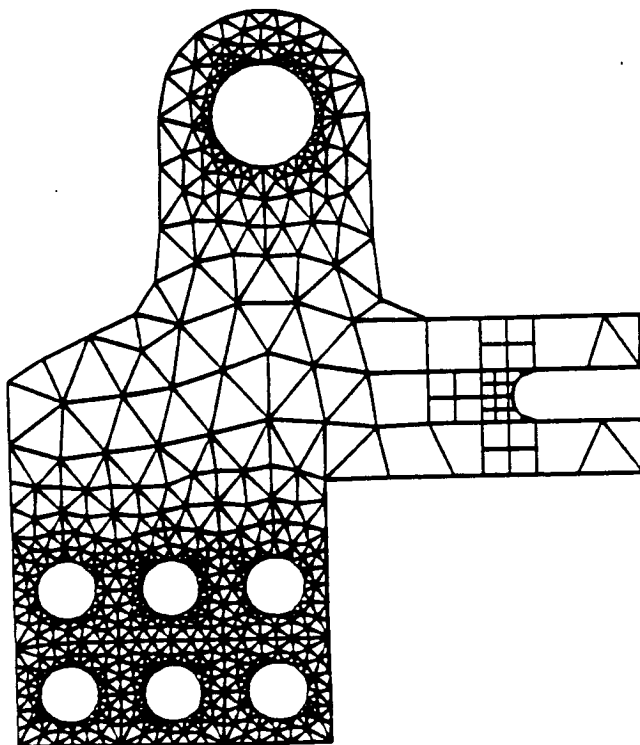
FIG. 8. h-refinement by element bisection



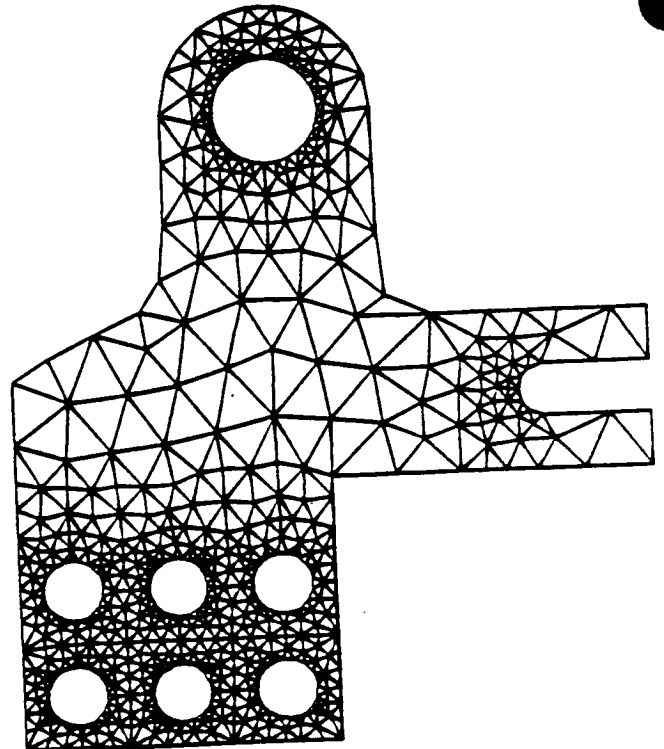
a) initial mesh



b) affected portion of mesh removed



c) refined quadrants



d) resulting refined mesh

FIG. 9. Finite quadtree mesh refinement by local remeshing

N88-19114 i

S3-61  
125789  
248

INTEGRATION OF GEOMETRIC MODELING AND  
ADVANCED FINITE ELEMENT PREPROCESSING

By

Mark S. Shephard  
Rensselaer Polytechnic Institute

and

Peter M. Finnigan  
General Electric Corporate Research and Development Center



## ABSTRACT

The structure to a geometry-based finite element preprocessing system is presented. The key features of the system are the use of geometric operators to support all geometric calculations required for analysis model generation, and the use of a hierarchic boundary-based data structure for the major data sets within the system. The approach presented can support the finite element modeling (FEM) procedures used today as well as the fully automated procedures under development.

## 1. INTRODUCTION

The generation of numerical analysis models is one of the major steps in the computer-aided engineering (CAE) process. Of primary concern is the disproportionate amount of the entire design/analysis process that is currently dedicated to this task. If significant productivity gains are to be achieved in CAE, this bottleneck must be reduced. In the long term, this means the automation of the entire finite element process, which would include such things as adaptive analysis and optimization techniques. In the short term, this means improving the basic model generation tools and developing preprocessing systems that employ advanced geometric modeling and more powerful data structures. This paper presents the overall design of a geometry-based FEM system that will address today's needs, as well as provide a foundation for the fully automated systems of tomorrow.

The most obvious reason for a better integration of finite element and geometric modeling is to avoid the need to redefine the geometry during finite element modeling. The second is to make more direct use of the functionality present in advanced geometric modeling systems. The third reason is the creation of a more unified design/analysis environment that employs a geometry-based (object) problem description. This includes geometry-based analysis attribute specification, which is not only a necessary part of an object-based problem definition, but is the most efficient method of prescribing this information. Finally, it is only with the close integration of geometric and finite element modeling procedures, that FEM can be fully automated.

In addition to the requirements placed on a preprocessing system by the above needs, it must give the analyst all the model generation functions needed to efficiently create controlled element meshes. At this time, that requires the system to support bottom-up, mapped, and fully automatic mesh generators such that they can be used in various combinations in a consistent manner. This also means that flexible, geometry-based mesh control specification techniques

be used.

The majority of the current finite element preprocessing systems have been developed in an evolutionary manner, independent of geometric modeling systems. Interfacing, not integration, results between the two systems. With the recent advances in geometric modeling procedures, there is a desire to make more direct use of the geometry available for the generation of the finite element model. To date, there has been limited success in integrating geometric and finite element modeling. The major factors deterring this integration are:

1. Generally, finite element preprocessing systems are designed to construct a finite element model by directly building the object's description in terms of topologically simple shapes (i.e., triangles, quadrilaterals, tetrahedrons, hexahedrons, etc.). This approach is not well suited for general geometric manipulation.
2. The data structures within a finite element preprocessor are designed to house little more than the most basic of mesh construction information and the mesh data (i.e., node point coordinates and element definitions). They do not possess a general geometric data structure to house the original geometric definition of the object, nor do they maintain relationship information which explicitly couples the finite elements themselves to the geometry from which they came.
3. The geometric modeling systems do not make their intrinsic geometric manipulation capabilities readily available to other applications on the system which require such functionality.

The majority of effort that has been expended on improving the level of integration between finite element and geometric modeling has been aimed at particular modeling systems. The finite element preprocessing developers have added their own geometric modeling capabilities, or the geometric modeling developers have extended their systems to include finite element model generation. Although these approaches represent improvements, they tend to lack generality and represent a large duplication of software development effort. In addition, they typically lead to systems that have lopsided strengths; such as geometric modeling systems that are well suited for developing finite element models but poor for other applications. The position taken in this paper is that the developers of geometric modeling systems should concentrate on providing advanced geometric modeling functionality, FEM developers should concentrate on the advancement of finite element modeling, and that these two groups work together to integrate their respective capabilities in a cohesive manner.

A general integration of geometric and finite element modeling requires not only the transfer of data but also the transfer of functionality from the geometric modeling system to the finite element modeler. This paper presents a methodology that addresses this need. It must be noted that the implementation of these methodologies requires a major expansion of the data structures underlying the finite element modeling system, the strict adherence to prespecified operators to interact with a geometric modeling system, and the construction of those operators. The successful implementation of such a

system depends on:

1. Finite element modeling developers recognizing the need for change.
2. Finite element modeling developers working closely with geometric modeling developers to better understand each others requirements and limitations.
3. Geometric modeling developers providing the requisite modeling functionality for FEM.

Section 2 indicates an approach to the modular integration of geometric and finite element modeling. Section 3 indicates the type of data structures required in a geometry-based finite element modeling system. Section 4 gives a more specific indication of how the various finite element model generation procedures would operate within such a system. Section 5 addresses the remaining open questions of the development of a geometry-based finite element modeling system.

## 2. APPROACH TO MODULAR INTEGRATION WITH GEOMETRIC MODELING

The first key to the development of a finite element modeling system that can be efficiently integrated with various geometric modeling systems is the use of a general data structure that can support various geometric forms. The second key aspect of this integration is the use of a general set of geometric communication operators [1]. A geometric communication operator is a procedure designed to perform a geometric function, given specific information about the operation and the geometry involved. The operator would return the result of the operation and/or modify the geometric representation to reflect the invocation of the operation.

The information passed directly to the geometric communication operator has a general structure. Any data, specific to a particular geometric modeling system, would be extracted directly from its geometric database by a geometric communication operator. Therefore, it is only necessary to provide a set of geometric communication operators for each new geometric modeling system that the finite element modeler is to be interfaced. No changes within the finite element modeling system need be applied. The approach discussed in this section is consistent with the CAM-I work on an applications interface for geometric modeling [2,3]. The advantage of this approach is obvious; it avoids the need to reproduce all the geometric modeling functionality within the finite element modeling system. This advantage is absolutely necessary if finite element modeling systems are to be interfaced with the various forms of geometric modelers being developed. Once a set of geometric communication operators are agreed on, the operators can be constructed by the developers of a geometric modeling system. Hopefully, the majority of them can be extracted directly from the modeling capabilities already present in the system.

The geometric communication operators needed for finite element modeling can be grouped into the following five categories:

1. BASIC QUERY - A request for geometric information that is intrinsically a part of the geometric modeling database.
2. DERIVED DATA QUERY - A request for geometric information not directly stored in its database. The determination of the requested data requires the performance of geometric calculations. A DERIVED DATA QUERY will not alter the contents of the geometric modeler's database.
3. GEOMETRIC MODELING OPERATION - A request is made that invokes one or more geometric modeling operations such that the geometric model is altered in the process.
4. ATTRIBUTE SPECIFICATION - The geometry-based specification, modification, or deletion of the model's attributes.
5. GENERAL UTILITY - These would contain the operators that request general, geometry independent information on a model.

A large number of geometric communication operators are needed for finite element modeling. They will most likely be built in two levels. The low level operators will represent atomic geometric operations such as Euler operators [4] or specific Boolean operations [5]. The higher level operators, those oriented toward finite element modeling, would be constructed primarily from the low level operators. This approach has the advantage of insulating the FEM system from changes in the geometric modeling system because only the internals of the higher level layer would be affected. Thus, the FEM system could be interfaced to a new geometric modeling system with relative ease assuming it provided, in some form, the low level operators. It should be noted, that since the type and amount of information stored in a geometric modeler's database is a function of modeling approach and implementation, an operator that is a BASIC QUERY in one system may be DERIVED DATA QUERY in another. A few example operators are listed below:

#### BASIC QUERY

1. RETURN\_GEOMETRY\_COEFFICIENTS - return the coefficients used in the definition of a geometric entity.
2. RETURN\_TOPOLOGICAL\_ENTITY - return the definition of the requested topological entity.
3. RETURN\_ENTITY\_ASSOCIATIVITY - return a specific set of associativities for a topological entity.

#### DERIVED DATA QUERY

1. DETERMINE\_DISTANCE\_BETWEEN - calculate the minimum distance between two geometric entities.
2. POINT\_CLASSIFY - determine if a given point is inside the object, outside the object, or on the surface of the object.

3. DETERMINE\_INTERSECTIONS - calculate the intersections between two geometric entities.

#### GEOMETRIC MODELING OPERATION

1. ADD\_ENTITY - add a given entity to the geometric model.
2. SPLIT\_ENTITY - break a given entity into multiple entities in a prescribed manner.
3. PERFORM\_BOOLEAN - carry out a Boolean operation between two specified entities.

#### ATTRIBUTE SPECIFICATION

1. PLACE\_ATTRIBUTE - apply an attribute to a given entity.
2. MODIFY\_ATTRIBUTE - modify a given attribute on a given entity.

#### GENERAL UTILITY

1. GET\_MODEL - retrieve a given model form the database.
2. SAVE\_MODEL - store the current model in the database.

Although the concept of geometric communication operators represents the most general method to tie the functionality of geometric modeling systems to geometry-based applications, the geometric modeling systems available today do not fully support this concept. This is expected to change over the next few years. The move to more open architectures, the increased pressures from applications, a maturing of geometric modeling systems, and specific research on the creation of such operators are contributing to this change. Developers of application software should expect the availability of specific sets of low level operators in the near future.

### 3. DATA STRUCTURES IN A GEOMETRY-BASED PREPROCESSOR

This section first provides an overview of the data structures required in a truly geometry-based preprocessor before discussing the details of a particular implementation. It is important to understand, in very general terms, the data structures themselves and how they interact.

There are a number of possible ways to group the requisite preprocessing data. The one listed below was selected because it uses the minimal number of data sets that provide a logical separation of information needed for finite element modeling. The data sets include:

- The MODEL data set
- The ATTRIBUTE data set
- The MESH data set

The MODEL data set contains the geometric and topological data 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 mesh control data. The MESH data set contains the finite element mesh generated for the model. Each data set has its own structure tailored to meet its special requirements. 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 data. The information content of these data structures and their relationships are described in more detail in the sections which follow.

### 3.1. Model Data Structure

The most fundamental data to the preprocessor is the geometry. The work described in this paper uses the concept of a non-manifold geometric modeling topology representation [4]. In a manifold representation, the area surrounding any point on a surface (in the limit) is "flat." In a non-manifold representation, the "flatness" criterion is not a requirement.

Historically, solid modeling systems have employed manifold representations. However, this has caused problems when non-manifold results would occur as a natural part of the model building process. One major benefit of a non-manifold representation is that it permits wire frame, surface, and solid models to coexist in the same system concurrently. Relative to finite element modeling, it appears that the developers have not appreciated the importance of a well defined topological model and that the topology which does exist in these systems has been evolutionary. Not unexpectedly, they are inadequate to support major advances in automation.

There is a close parallelism between finite element modeling and geometric modeling with the three representations. That is, because of the abstractions associated with FEM, all three geometric representations may be necessary simultaneously. For example, many "real world" models are typically comprised of a combination of solid, shell, and beam elements. Although not exclusively, these element types lend themselves to being modeled with the "corresponding" geometric representations. That is, the shell portion of the model with a surface representation, the beam portions with a wire frame representation, and of course, the solid elements with a solid representation. The logical conclusion is that a non-manifold data structure that can support the three forms of geometric modeling, can also support the geometric aspects of finite element modeling. A single representation opens the possibility for the finite element modeling system to make direct use of geometric operators developed in support of the geometric modeling system.

The non-manifold geometry/topology hierarchical model used is depicted in Figure 1. In addition, the relationships between geometry and topology are shown.

#### 3.1.1. Model Data Definitions

This section provides a set of working definitions for the various geometric and topological entities used for geometric modeling. In addition to the basic definition being stored in the data structure, its origin or purpose will also be stored. That is, if an entity originates from the geometric

modeler, or if an entity is added for the purpose of controlling the mesh or applying a boundary condition, it will be so identified.

#### .1.1.1. Geometric Entities

There are four geometric entities which the system will support as defined below:

- POINT -- A point is a geometric entity specified by a triple of numbers representing its position in space.
- CURVE -- A curve is a geometric entity representing the trajectory of a point through space. Curves can be infinite in extent.
- SURFACE -- A surface is a geometric entity representing a two-dimensional locus of points. Surfaces can be infinite in extent.
- VOLUME -- A volume is a geometric entity representing a three-dimensional locus of points.

It should be noted that in most geometric modeling systems, volumes in space are defined in terms of a set of surfaces that enclose it. It is, however, desirable to support the possibility of a specific volume geometric entity which adds the ability to house volumes with internal definitions in the system [10,11].

#### 3.1.1.2. Geometric Modeling Topological Entities

The topological entities form a hierarchy which, when coupled with geometry, provide a complete definition of the part. A brief description of each of the geometric modeling topological entities to be used in the model data is given below.

- VERTEX -- A vertex is the topological equivalent of a three-dimensional point in space. It is typically used to bound an edge. There is always a vertex at the joining of edges. Vertices may also be used as a boundary of a face or shell.
- EDGE -- An edge is the topological equivalent of a geometric curve (i.e., line, arc, spline). It is bounded by two vertices. An edge may be closed, in which case the starting and ending vertices are the same.
- LOOP -- A loop consists of an ordered, closed, connected, set of edges. A loop bounds a face.
- FACE -- A face consists of a portion of a shell. A face is bounded by at least one loop, and may be internally bounded by further interior loops (i.e., holes).
- SHELL -- A shell consists of a set of faces which bound a region. A shell may consist of a connected set of faces which form a closed volume or may be an open set of adjacent faces, a wire frame, a combination of these, or even a single point. In the case of a solid model,

one shell is required to define the external boundary and additional shells are required to define voids within the solid.

- REGION -- A region is a volume of space. A region has one exterior shell and one interior shell for each void contained within it.
- MODEL -- A model is a collection of regions. Regions within a model may be distinct because of physical separation in space, or simply because a user wishes to keep them logically distinct.

### 3.2. Attribute Data Set

Any form of numerical analysis, requires the following information:

1. A complete specification of the physics of the problem to be analyzed.
2. Specification of the desired level of domain discretization.
3. The specification of the required analysis control parameters.

In general terms, this information is referred to as the attribute data for the model.

The attribute data structure will contain all of the information, past the geometric definition of the object, that is needed to complete the description of the problem. Attribute data includes both geometric and non-geometric information. Data which is geometric in nature must be tied to the original geometric definition of the object.

A number of different modeling attribute types are needed for finite element analysis. A partial list of these includes:

1. Analysis program control data.
2. Case information.
3. Finite element type declaration information.
4. Nodal (skewed) coordinate system data.
5. Material property data.
6. Physical property data.
7. Mesh control data
8. Essential boundary condition data
  - e.g., displacements in a stress problem or temperatures in a heat conduction problem.
9. Natural boundary condition data
  - e.g., pressures in a stress problem or fluxes in a heat conduction problem.
10. Initial condition data.

#### 3.2.1. Classes of Attributes



Finite element modeling attributes can be categorized by class, depending on how they interact with geometry. Three distinct classes of attribute data have been identified, and are described below:

CLASS #1:

Attribute data in class #1 is characterized by data which is required for the analysis to be performed but which is totally independent of the geometric definition of the model.

CLASS #2:

Attribute data in class #2 is characterized as data which is attached directly to geometric entity data, and which can be described completely in terms of that geometric entity.

CLASS #3:

Attribute data in class #3 is characterized by data which is attached directly to geometric entity data but which needs auxiliary geometric data (henceforth known as attribute specification geometry) to help define the attribute. That is, the attribute may not be conveniently described in terms of the geometric entity to which it is attached, and thus two pieces of geometric entity data are required; a piece of geometry data and a piece of attribute specification geometry data.

The basic distinction between class #2 and class #3 data is that class #2 data needs a single geometric entity to define both (a) the associativity of the attribute with the model and (b) the attribute's definition. Class #3 data, on the other hand, requires a piece of geometric data to define its associativity with the model, and in addition, requires attribute specification geometry to define the attribute.

As an aid to understanding class #3 data, consider the case of an arbitrarily complex flat plate in the xy-plane. Suppose that the structure was subjected to a normal pressure load which was linearly varying as a function of y. If one attempted to describe this pressure solely in terms of pressure values along the edges, one would not have a uniquely defined pressure surface, and thus the pressures on individual elements could be evaluated incorrectly. Alternatively, the pressure could very simply be specified by defining an auxiliary piece of geometry; in this case a rectangular face which covered the entire 2-D domain and four pressure values, one at each of the corners. Pressure on individual elements could then be evaluated in a totally unambiguous manner.

### 3.2.2. Attribute Specification Geometry

Attribute specification geometry is simply geometry plus topology which is used to help define the physics of certain attributes. It can be thought of as being auxiliary to the part definition. It has no direct links to the part's geometric data structure. The attribute specification geometry will be an intrinsic part of the definition of the attributes. It is stored in the MODEL data structure but is referenced through the attributes. The attribute

specification geometry maintains the same hierarchic structure as the models geometry/topology.

The purpose of attribute specification geometry is twofold. First, it provides a mechanism for allowing simpler and more efficient specification of attribute data. Secondly, it provides a means for evaluating attribute data directly and unambiguously.

### 3.2.3. Attribute Data Structure

The relationships between attribute data and geometry are as follows. The model's topological entities point to attributes. An attribute contains its definition along with two pointers. One pointer points back to the geometric entity to which it is "tied." The other pointer points to the attribute specification geometry which is used to help define the attribute. If this geometry is, in fact, the same as the model's geometry, then the attribute specification geometry pointer is a null pointer. Figure 2 shows the general data structure for a generic attribute data type. It indicates how attributes are specified and associated with geometry/topology. The dimensionality of the attribute topology can be the same or lower order as the model's topology to which it points; it can never be of higher order. That is, an edge is a one-dimensional entity. The permissible associated attribute's topological entities are "vertex" and "edge."

### 3.3. Mesh Data Structures

In addition to the hierarchy of geometric modeling entities discussed earlier, there will also be a hierarchy of finite element entities, the MESH data structure (Figure 3), which will be 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).

The finite element entities have two types of data associated with them. The first is the modeling topology data, and the second is the finite element attribute data (i.e., material properties, physical properties, etc.). Each finite element entity points to the lowest order modeling topology entity which it is inherently a part. 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. It is this relationship that permits attribute data which is tied to the geometry to be evaluated on an element by element or node by node basis.

Pointers from the geometry to individual element components (i.e., fe-nodes, fe-edges, fe-faces, etc.) are also desirable for efficient postprocessing activities. The storage penalties for maintaining these relationships are easily offset by the performance gains which can be achieved.

#### 3.3.1. Finite Element Entity Definitions

What follows is a set of working definitions for the various finite element entities.

- FE-NODE -- A fe-node is a three-dimensional point in space. Typically, it is used to define a fe-edge; however, it can also be used to help define a finite element. For example, the mid-face node used in a Lagrange parabolic element is not associated with a fe-edge, but rather with the fe-face itself. In addition, a reference node used to define the center of curvature or the plane in which a beam element lies would point to the element rather than the edge of that element. Finally, a fe-node could be used to define the element explicitly such as the concentrated mass element. A fe-node may lie on a vertex, an edge, a face, or be completely contained within a region.
- FE-EDGE -- A fe-edge is a combination of topology plus implied geometry. That is, the nodes used in the definition of an edge are used to bound the geometric curve, and at the same time can be used to define the geometry (i.e., straight line, parabola, or cubic) for the element itself. A fe-edge can be used to define a fe-face, as with planar or solid elements, or can be used to define the finite element directly, as is the case with truss and beam elements. It may lie on an edge, a face, or be completely contained within a region.
- FE-FACE -- A fe-face is bounded by fe-edges. It is either used to define the surface of a "planar" finite element or is used in combination with other fe-faces to bound a solid finite element. Topologically, a fe-face will be either triangular or quadrilateral in nature. It may lie on a face, or be completely contained within a region.
- FINITE ELEMENT -- Depending on the type, a finite element can be a fe-node, a fe-edge, or a collection of fe-faces. It may be completely contained within a region. However, it is true only for solid finite elements that the entity "finite element" can point to a region because other finite element types will have lower order entities which point to various geometric entities.

### 3.3.2. Advantages of a Finite Element Entity Hierarchy

At first glance, the MESH data structure, with its hierarchy of finite element entities, may seem too elaborate, perhaps even wasteful of valuable storage. However, on closer inspection some distinct advantages emerge. The most powerful advantages come from the links to the other data structures. These relationships are discussed in the next section. Independent of the benefits which accrue due to these links, a number of other benefits surface as enumerated below:

- It provides an organization for handling any type of finite element in a uniform manner.
- 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.
- It makes it possible to interrogate the finite element model using a geometric entity as a key word for searching.

- It provides a mechanism which supports mesh generation on the basis of topologically simple cells (i.e., quadrilaterals, triangles, hexahedrons, etc.) which corresponds to linear finite elements, providing a direct path to upgrade to higher 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.

### 3.4. Relationships of the Three Data Structures

The power of the implementation is derived from two sources; the data structures themselves and the relationships between the structures. Figure 4 shows the relationships which exist amongst the three data structures. It is, in fact, these links that provide the necessary structure for claiming to be a geometry-based system. These links provide a bond between the data structures which permit the system to respond in a cohesive manner.

The links are automatically established during the model generation process. The natural progression of events is something like this:

1. The part is generated via a geometric modeling session. The geometric entities are loaded into the MODEL data structure.
2. Model attributes are defined and loaded into the attribute data structure. Since the attributes are associated with the model's geometry, two-way pointers are established between the MODEL data and the ATTRIBUTE data. In addition, any necessary attribute specification geometry is generated and stored in the MODEL data structure. Links are also established between the ATTRIBUTE data structure and the attribute specification geometry.
3. One of the attributes is mesh control data. Having this information, mesh generation can proceed, and the resulting node and element data is stored in the MESH data structure. During the mesh generation process, the associations which exist between the finite element mesh data and the part definition are known, and thus pointers between the MESH data structure and the MODEL data structure can be generated.
4. Since both the MESH data and the ATTRIBUTE data point to the MODEL data, the attribute data can then be evaluated on a node by node or element by element basis. The links between the MESH data and the ATTRIBUTE data structures are established at this point.

This completes the model building process. It accomplishes what it was intended for; to use a completely geometry-based approach to produce an analysis model.

## 4. DESIGN OF A GEOMETRY-BASED PREPROCESSOR

The approaches and data structures outlined above form the basis on which an advanced geometry-based preprocessing system can be built. The remaining capabilities needed are the actual finite element model definition procedures and the user interface.

The best form of user interface for this system is an interactive graphics front end. This is obviously the most convenient form of interface for the specification of geometry and geometry-based information. Even for those cases where the geometry to be meshed is identical to that obtained from the geometric modeler, and an automatic mesh generator is used, there is still the need for the specification of the analysis attribute information in terms of the geometric model. Until fully automatic, adaptive procedures are available, the system must support the entire range of finite element mesh generation procedures. These are most efficiently operated in an interactive graphics mode.

Geometric operations within the system will be carried out making heavy use of the capabilities of the geometric modeling systems to which it is interfaced. It is important that geometric modeling functions be presented in a form appropriate for finite element modeling. This may be different than the way they are presented in the geometric modeling system. In addition, it must be recognized that different geometric modeling systems will not provide the same sets of geometric modeling functions. The two level approach to the implementation of the geometric communication operators provides a method to deal with both of these concerns. The high level geometric communication operators for finite element modeling would be designed to fit directly into the modules of the preprocessing system. Since they are constructed by the combination of the low level geometric communication operators, which represent the actual tie to the geometric modeler, they need not necessarily be altered when a new modeler is interfaced to the system. If a particular geometric modeler does not provide specific low level operators used by a high level operator, it may be possible to reconstruct the high level finite element operators by a different combination of low level operators.

The geometric modeling functions needed by a complete finite element preprocessing system are extensive. They include a full set of high level operators, such as the Boolean operators, for the construction and modification of geometry, as well as for use by automatic mesh generators to decompose the geometry into a valid finite element mesh. A full range of geometric interrogation operators will be required for use by the mesh generation algorithms, the mesh checking procedures, the geometric construction operations, and the attribute specification procedures. Finally, a full range of bottom-up geometric modeling functions are needed to allow the analyst to define all or part of a geometry.

The attribute specification procedures in a geometry-based preprocessor must give the analyst a high level of flexibility in the specification of the various types of finite element attributes. The geometric specification procedures for defining analysis attributes, such as distributed loads, must allow for the convenient description of the distribution of the loads, as well as for defining the portions of the object on which they act.

Flexible procedures must be available to group attributes of the same type into sets for simple manipulation during the specification of the actual load cases to be analyzed. The reason for allowing the grouping of attributes is partly to provide convenience to the analyst, but is mainly for the purpose of allowing a greater degree of automatic validity checking in the system. By only allowing the combination of attributes of one type into sets, automatic

validity checks on attribute combinations, which are based on attribute type, can easily be done. The combination of attribute sets into analysis cases allows the application of an additional set of checks which can only be made after the analysis process control information has been indicated. Thus, the user maintains a high degree of flexibility while affording the system a means of performing validity checks at the appropriate levels.

A difficulty in the implementation of the mesh control attributes in a system that allows a variety of mesh generation approaches is devising a procedure that can operate from a single internal representation of mesh control information. Since all attributes, including mesh control information, are directly tied to entities in the geometric model, the most direct method of dealing with this specification is to tie mesh control parameters to each of the topological entities that define the object. The analyst can be given a set of procedures that allow for geometry-based specification of the mesh control information and have it properly associated with the topological entities. Since it is possible to introduce geometric entities for the sole purpose of attribute specification, this approach maintains the desired level of flexibility. The remaining question is the selection of mesh control parameters for the various topological entities that can always be meaningfully converted to the specific parameters needed by the various mesh generators. The simplest solution is to assign a single element size parameter to all entities. Although seeming simplistic, this tends to be acceptable for all entities except the edge. The reason for this is simply that most mesh generators base all their mesh control on edge information, and those that use additional parameters, typically use a single parameter per entity. The mesh control information appropriate for edges should allow for the specification of the number of elements along the edge, as well as biasing parameters to grade the size of elements in a flexible manner.

As indicated above, the preprocessor should house a variety of mesh generation procedures ranging from simple bottom-up meshing procedures through fully automatic meshing procedures. It is anticipated that as automatic mesh generators become more robust, and as the geometric modeling capabilities needed to support them continue to improve, they will tend to become the main mesh generation tool. However, until fully automated finite element modeling systems become available, there will be a continued need to support the other mesh generation approaches.

Bottom-up mesh generation will tend to be used for the quick construction of both mesh and geometry (in terms of the finite elements) for very simple objects, and for adding simple finite element entities to an object that does not contain all the geometric entities in a form convenient for generation of that portion of the finite element model. Although, these procedures will not represent the major mesh generation workhorse, their presence in the system is necessary.

Mapped mesh generators are the most popular mesh generation procedures currently available. To some extent, they are more difficult to provide the needed geometric communication operators than some of the automatic mesh generation approaches [1]. The system must contain procedures that allow the analyst to easily define the supplementary geometry needed to define the boundaries of mesh patches and to be able to select the geometric entities that

define the specific mesh patches. The process of defining these mesh patches in a geometry-based preprocessor that accepts a general geometric model as input is substantially different from preprocessors where the geometry is built in a bottom-up fashion in terms of mesh patches. The user tools needed to efficiently decompose a general geometry into a set of valid mesh patches are different from those that are efficient for defining a geometry in terms of a set of mesh patches. The preprocessing system discussed here should support both sets of capabilities.

The selection of fully automatic mesh generation procedures to be included in such a system must consider the following factors:

1. The ability of the mesh generator to produce the desired types of meshes.
2. The ease of integration of the meshing procedure with geometry through a set of geometric communication operators.
3. The computational efficiency of the mesh generator.

Since the level of complexity of geometric operators needed to integrate an automatic mesh generator with a geometric representation varies greatly [1], as does their computational efficiency, it is likely that these two factors will dictate the selection of automatic mesh generators to be included in the preprocessor.

## 5. OPEN QUESTIONS IN THE DESIGN OF A GEOMETRY-BASED PREPROCESSOR

The procedures presented in the previous sections address the close coupling of the geometric representation of an object and the finite element mesh used to analyze it when there is an obvious correspondence of the finite elements generated and the geometric entities in the model. The type of finite element models that yield this correspondence are those where the finite elements are dimensionally the same as the geometric entities, and when the domains spanned by the geometric and finite element model are the same. However, it becomes much less clear how to account for the coupling between the geometric and finite element models when the geometric model is simplified for purposes of finite element analysis or when the finite element mesh contains a mix of element types of different geometric order representing various portions of a solid model. Common examples of these cases include ignoring specific geometric features deemed unimportant, and the use of shell or beam elements when one or two of the geometric dimensions of specific portions of a geometric model are small compared to the others. Element types of a dimensional order less than the geometric entity they represent account for the small dimensions in terms of element parameters such as thickness and moment of inertia. Elements of this type will subsequently be referred to as indirect elements.

Historically, the concern for the proper representation of the differences between the geometric model and the finite element model have not existed. This is because the two modeling processes were carried out independently. However, the desire to make direct use of the information in the original geometric model, to maintain complete links for making model revisions easier,

and to maintain a clear history of the analysis modeling procedures used, makes it necessary to address the question of how to define and account for these differences.

A major portion of the answer to these questions lies in the data structures to be used and the procedures employed to reflect the differences between the models in the database. However, this is not the appropriate place to begin to address these questions. One of the major factors that makes this a complex question is the lack of analytic procedures, or even an agreed upon set of rules for determining when and how these modeling differences should be used. If this information were available, it would be possible to devise algorithmic approaches to carry out these processes and it would become more obvious as to the best method to account for the results of those processes. Lacking such information requires that the approach taken to address these questions be somewhat open ended, thus allowing users to carry out the operations associated with geometric simplification and the generation of indirect element types in a flexible manner.

As an example of the range of possible approaches to geometric simplification, three different approaches to account for domain differences are considered. In all cases, the finite element analyst begins with the complete specification of the geometric model. In approach one, the analyst generates the mesh interactively with a mapped mesh generator. In this case, the finite element model generation process consists of the analyst simplifying the geometric model by performing specific geometric modeling operations during the construction of the various mesh patches. With currently available finite element modeling procedures, this is an appropriate method. However, this approach does not readily lend itself to account for the specific geometric simplifications made to the model before mesh generation. Even if the analyst was required to make the geometric simplifications, independent of the definition of the mesh patches, accounting for the simplification would require the explicit storage of both models or storing the list of modeling operations carried out during the simplification, neither of which is convenient.

The second and third approaches require the availability of a fully automatic mesh generation procedure that can ignore geometric features during the meshing process. With such a capability, the mesh generator can be passed the entire geometric model. Geometric features to be ignored are flagged appropriately. Accounting for the differences between the geometric and finite element models consist of simply examining the geometric model to see which geometric features are flagged. The second approach would consist of the analyst flagging the geometric details to be ignored while the third approach would rely on adaptive analysis procedures to determine the features to be ignored. Although the finite element modeling capabilities needed to support these two approaches are not fully available, components of them are currently being investigated. For example, the quadtree [6] and octree [7] mesh generators operate on the basis of hierarchic insertion of the geometric entities within an object's boundary file into a tree structure. Therefore, it is possible to simply identify those entities associated with the geometric features to be ignored so they are represented in an approximate manner. Although this approach may not be able to account for all desired forms of geometric simplification, it should be able to easily handle a majority of them. Efforts are currently underway to develop and test these capabilities.



The development of adaptive analysis procedures to automatically identify geometric details to be ignored, is a much more complex issue. One possible approach is to combine a set of rules employing analytic stress concentration factors with the results from an initial analysis that ignored features in order to estimate their influence [8] and to determine if they should be included.

The controlled generation of, and accounting for, the use of indirect element types is an even more complex process. The computerization of this process could make effective use of artificial intelligence techniques to help convert geometric representations to numerical analysis representations [9].

## 6. CLOSING REMARKS

There are a number of areas that must be addressed before fully automated finite element modeling becomes a reliable analysis tool that is an integral part of the computer-aided engineering process. This paper has addressed one of those areas which is the framework of a preprocessing system that allows the complete integration of finite element modeling with geometric modeling. The two key aspects of the approach are the use of geometric communication operators and the use of advanced data structures required to store the various data sets needed in finite element modeling. The key to the data structures is the use of a single hierarchic boundary-based geometric representation for both the geometric model and the finite element model. To this, auxiliary data structures (e.g., the attribute data structure) can be linked. A boundary-based representation was selected because:

1. It is the most general form of geometric representation to which other geometric forms can be converted.
2. It is a convenient framework on which new geometric and finite element types can be quickly added.
3. It is the most natural form, since finite element modeling is dominated by boundary information.

The major penalty for the added capability of this approach is the large amount of data storage. This is unavoidable if the goal of a general, geometry-based system is to be achieved. The only way to reduce the amount of information required is to reduce the level of integration with general geometric modeling systems or to limit the number of finite element modeling procedures that can be supported.

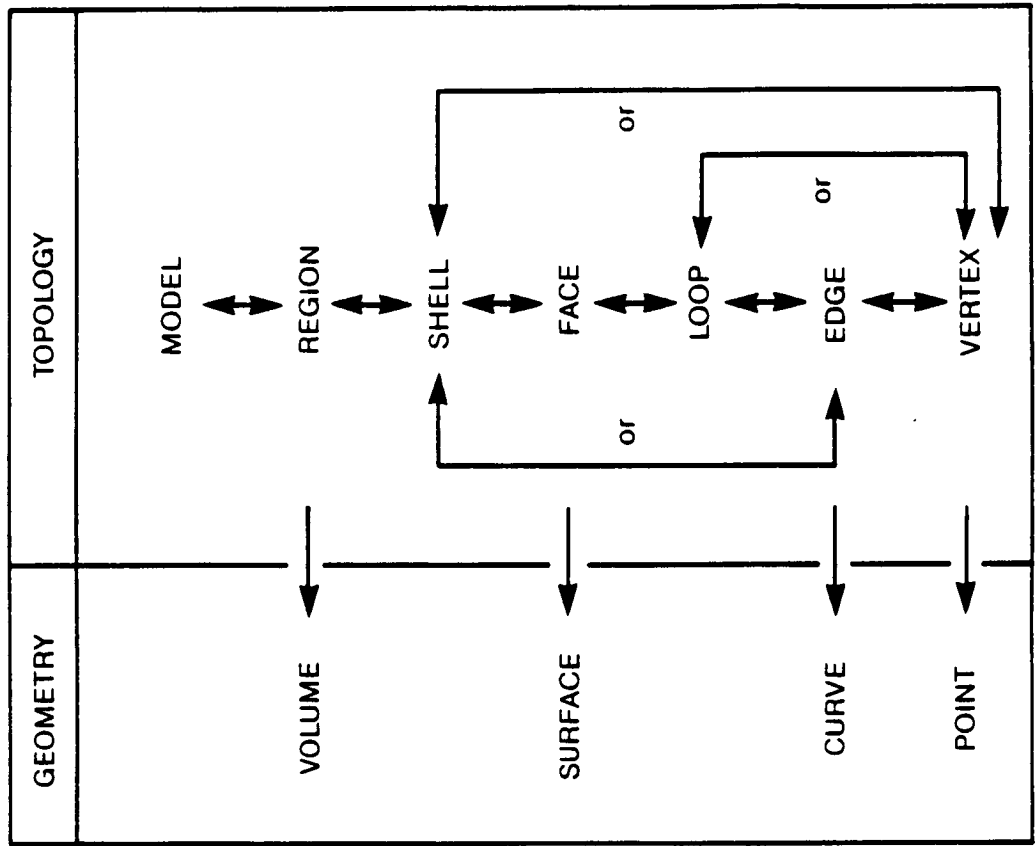
A final advantage of the approach presented here is that it can fully support today's finite element modeling procedures while allowing the introduction of ever increasing levels of automation as fully automatic mesh generators and adaptive analysis procedures evolve. This is very important since current preprocessing systems cannot support full automation and it is only through automation of these procedures that finite element techniques can be made a reliable tool for designers and not just the experts.

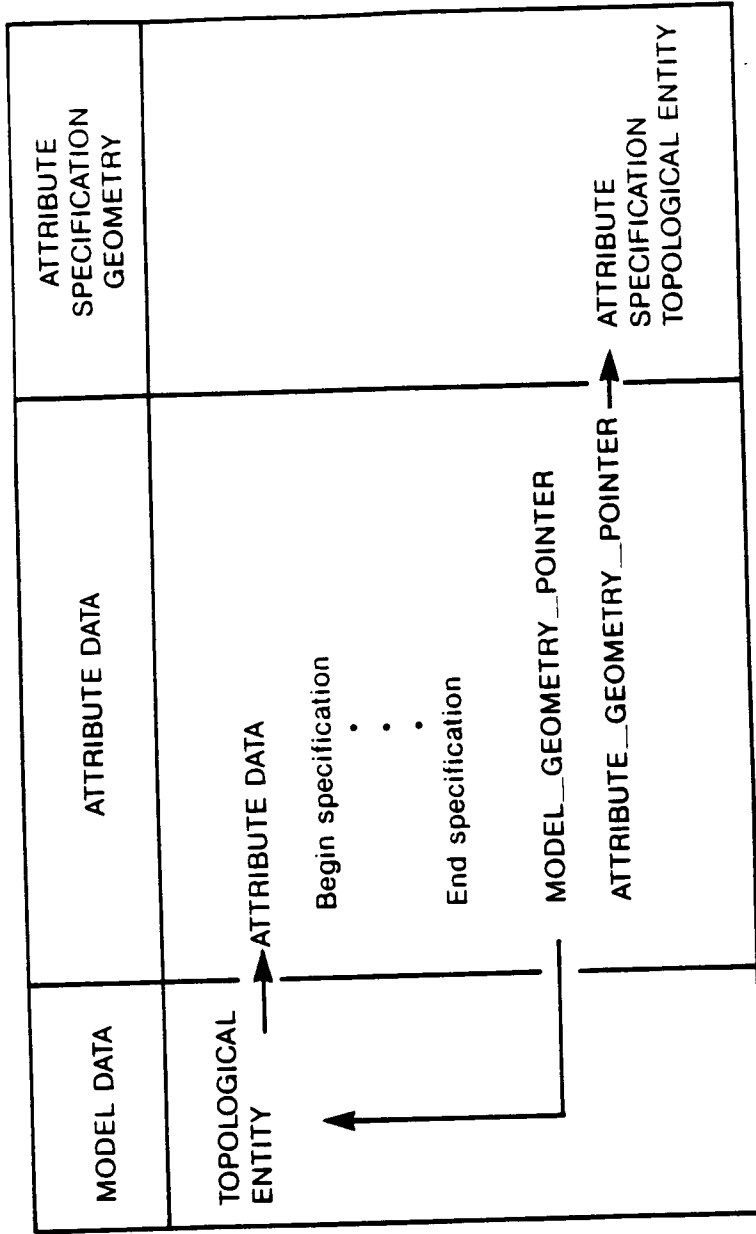
## REFERENCES

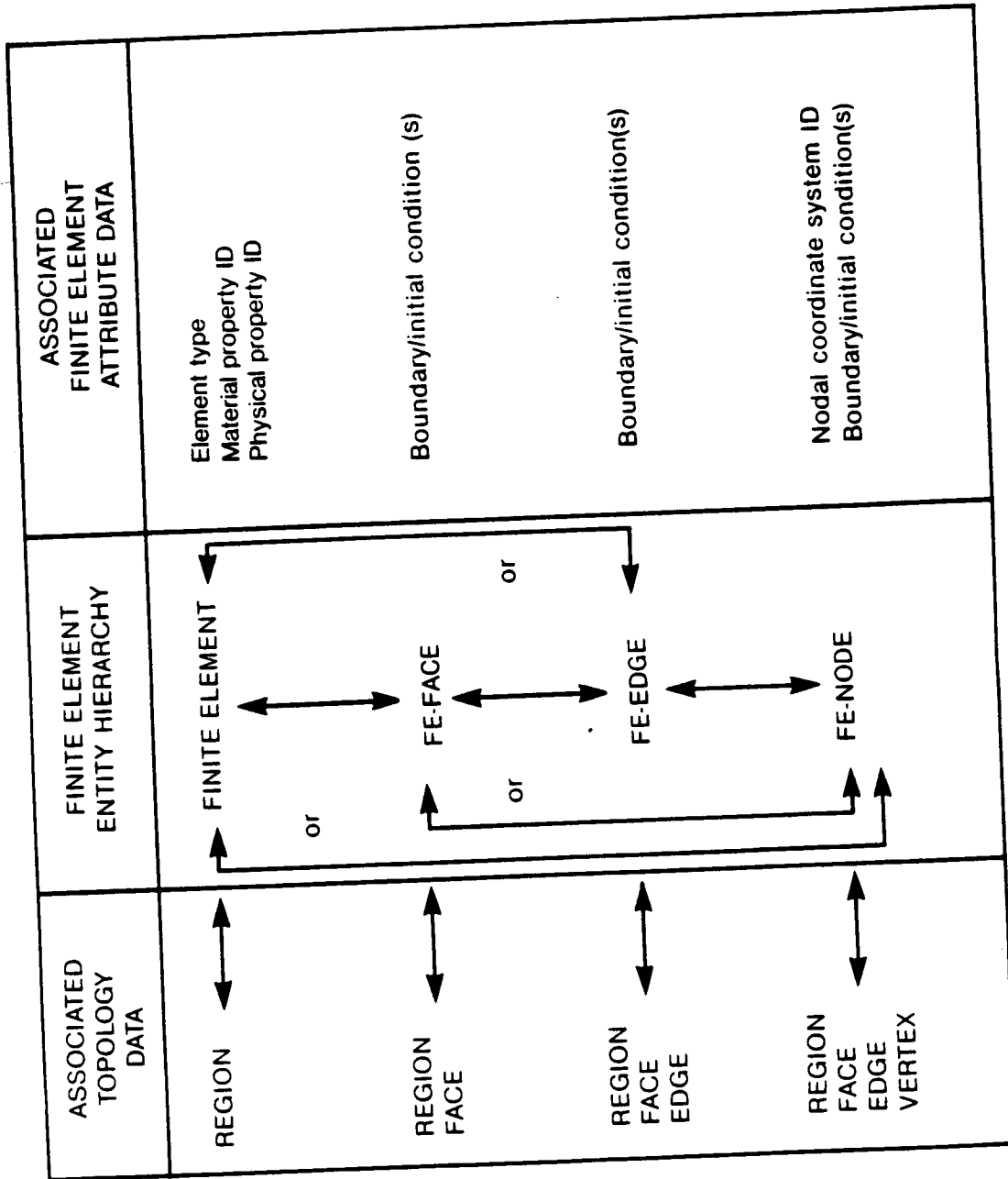
1. Shephard, M. S., "Finite Element Modeling Within an Integrated Geometric Modeling Environment: Part I - Mesh Generation, Part II - Attribute Specification, Domain Differences, and Indirect Element Types," Engineering With Computer, Vol. 1, 1985, pp. 61-85.
2. "CAM-I Geometric Modeling Project Boundary File Design (XBF-2)." CAM-I Report R-81-GM-02. 1, October 1982.
3. Wilson, P. R., I. D. Faux, M. C. Ostrowski, and K. G. Pasquill, "Interfaces for Data Transfer Between Solid Modeling Systems," IEEE Computer Graphics and Applications, Vol. 5, No. 1, 1985, pp. 41-51.
4. Weiler, K., "Topological Structures for Geometry Modeling," PhD Thesis, Rensselaer Polytechnic Institute, Troy, New York, 1986.
5. Requicha, A. A. G. and H. B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment," IEEE Computer Graphics and Applications, Vol. 3, 1982, pp. 9-24.
6. Baehmann, P. L., 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, Rensselaer Polytechnic Institute, Troy, New York, 1986.
7. Yerry, M. A. and M. S. Shephard, "Automatic Mesh Generation for Three-Dimensional Solids," Comput. Struct., Vol. 20, 1985, pp. 31-39.
8. Shephard, M. S. and M. A. Yerry, "Toward Automatic Finite Element Modeling for the Unification of Engineering Design and Analysis," Finite Elements in Analysis and Design, Vol. 2, 1986, pp. 143-160.
9. Gregory, B. L. and M. S. Shephard, "Design of a Knowledge Based System to Convert Airframe Geometric Models to Structural Models," Expert Systems in Civil Engineering, ASCE, New York, New York, 1986, pp. 133-144.
10. Casale, M. S. and E. L. Stanton, "An Overview of Analytic Solid Modeling," IEEE Computer Graphics and Applications, Vol. 5, No. 2, February 1985, pp. 45-56.
11. Farouki, R. T. and J. K. Hinds, "A Hierarchy of Geometric Forms," IEEE Computer Graphics and Applications, Vol. 5, No. 5, May 1985, pp. 51-78.

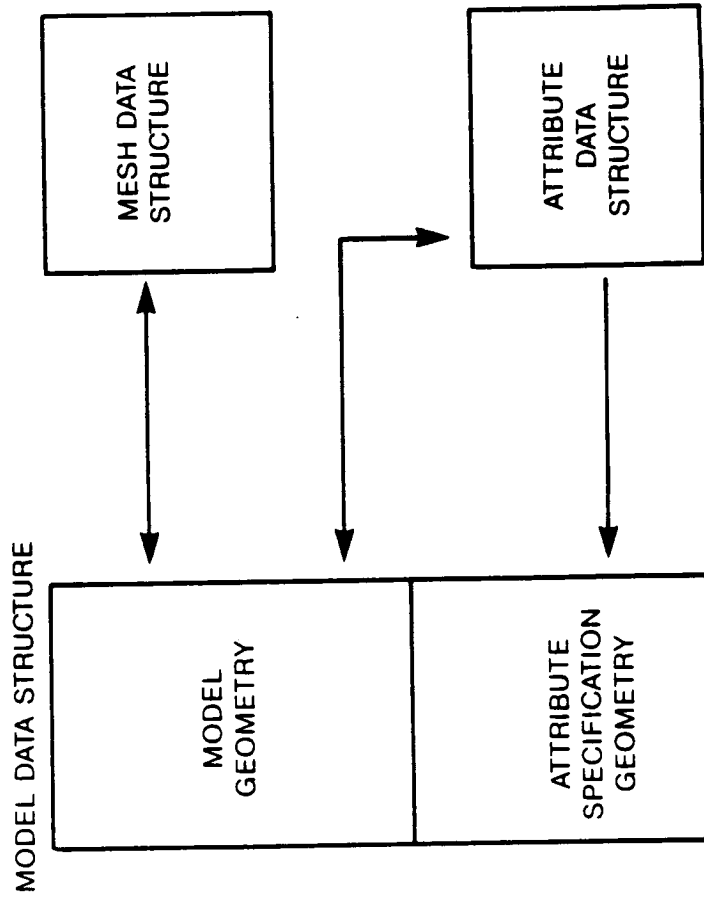
LIST OF FIGURES

1. A NON-MANIFOLD GEOMETRY REPRESENTATION FOR FINITE ELEMENT MODELING
2. GENERIC ATTRIBUTE DATA STRUCTURE
3. MESH DATA STRUCTURE (A hierarchy of finite element entities)
4. RELATIONSHIPS OF THE DATA STRUCTURES









*CMIT*

# **Geometric Modeling in Transition**

**May 12, 1987**

Lee Robie  
SDRC  
Milford, Ohio



# Geometric Modeling in Transition

## • Application

- Specific Tool - Limited Percent of the Total Job
- Fatigue Analysis
- NC Programming
- Mechanism Simulation

## • Utility

- General Support Tool
- Data Management
- User Interface
- Graphics

## • Transition Indicators

- Need: General Applicability
- Age
- Consensus of Approach

"It is clear that solid modeling is increasingly being viewed as a tool for creating a central database upon which most applications may run"

-CAD/CIM Alert - October 31, 1986

# Mechanical Product Definition

- Complete Product Definition Data
  - Object Representation (Geometry + Topology)
  - Features
  - Dimension and Tolerance
  - History and Heritage
  - Associated and Administrative Data
  
- Where is the Application - Utility Dividing Line?
  - Varies with time
  - Object Representation
    - Precise Boundary Representation
    - NURBS
  - Features
  - Associated Data
    - Application Specific
    - Generic

# Features

## • Definition

- An aspect of an object that is significant in some context
- A closed volume with an implied boolean

## • Contents

- Administrative Data
- Geometric Association
- Geometric Parameters
- Application Data
- Modification Rules
- Recreation Procedure

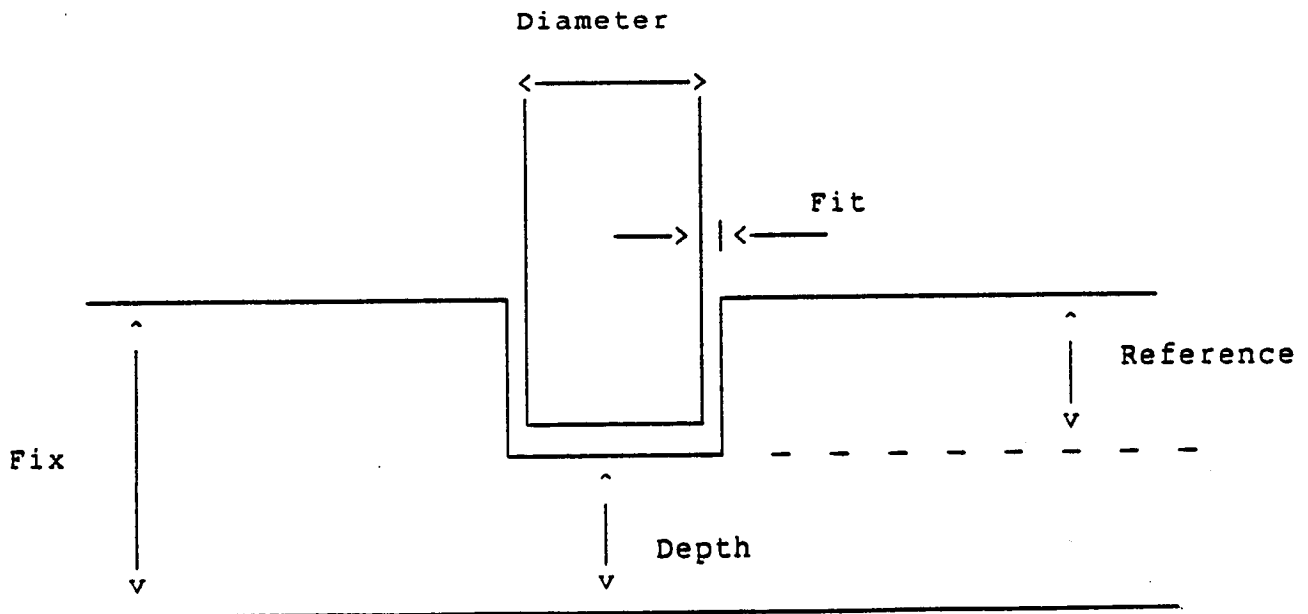
## • Uses

- Higher level of Information Content
- User Interface
- Design Rules
- Geometric Abstraction
- Shape Optimization
- Downstream Functions

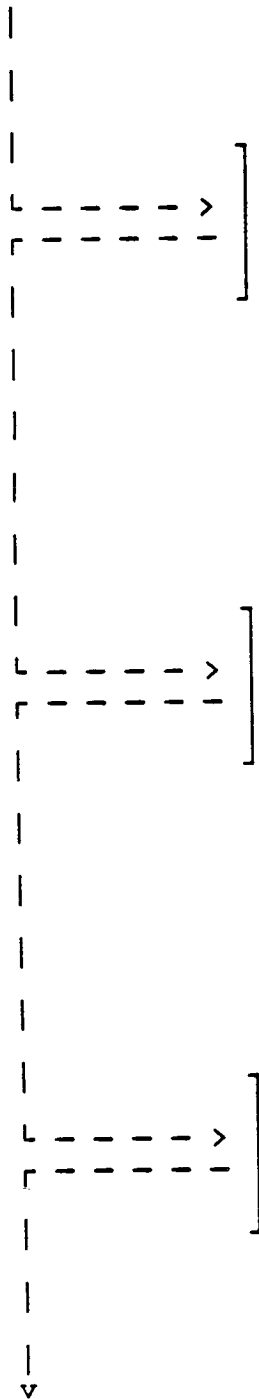
# Variational Geometry

- Developed at MIT in late '70's
- Relationships:

Angle	Parallel	Tangent
Distance	Perpendicular	Fit
- Captures Design Intent - Improves Modification
- Embryonic
- Futures: Sculptured Geometry



# Product Model Data Flow



- Design

- Geometry
- Features
- Constraints
- Associated Data

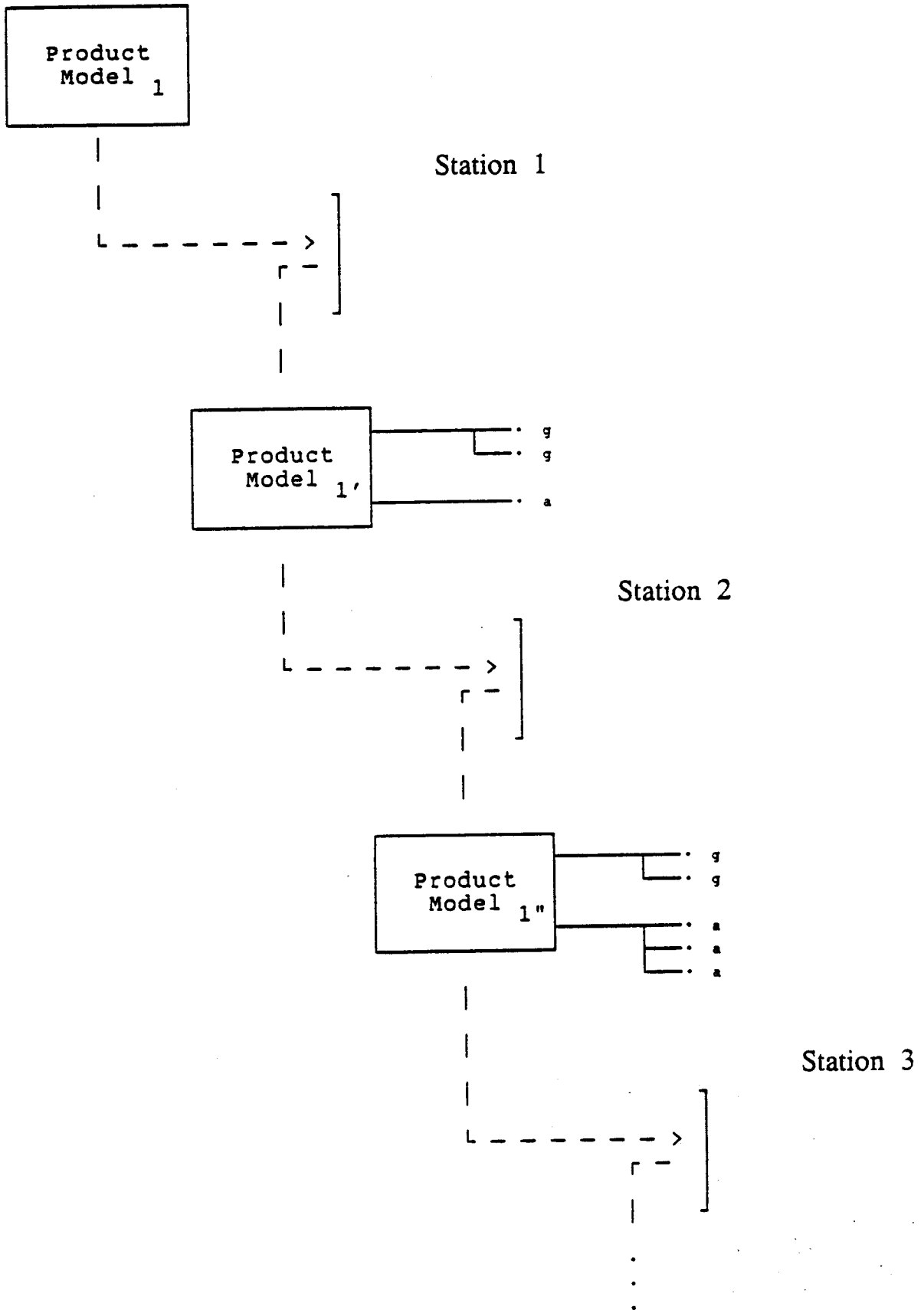
- Structural Verification

- Select Material
- Define Loads
- Mesh and Analyze
- Modify

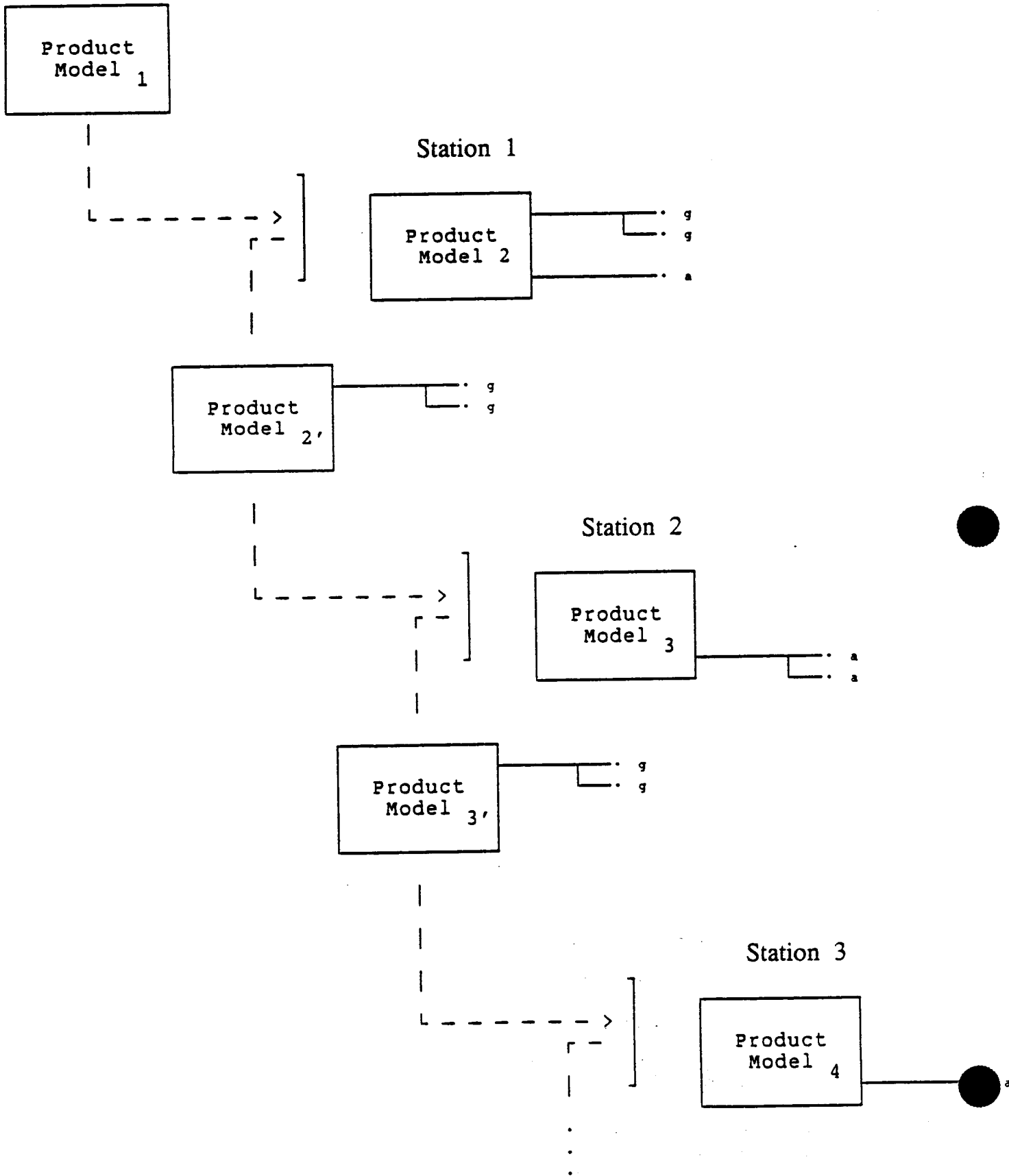
- Manufacturing Engineering

- Material Cost
- Manufacturing Process and Cost
- Modify

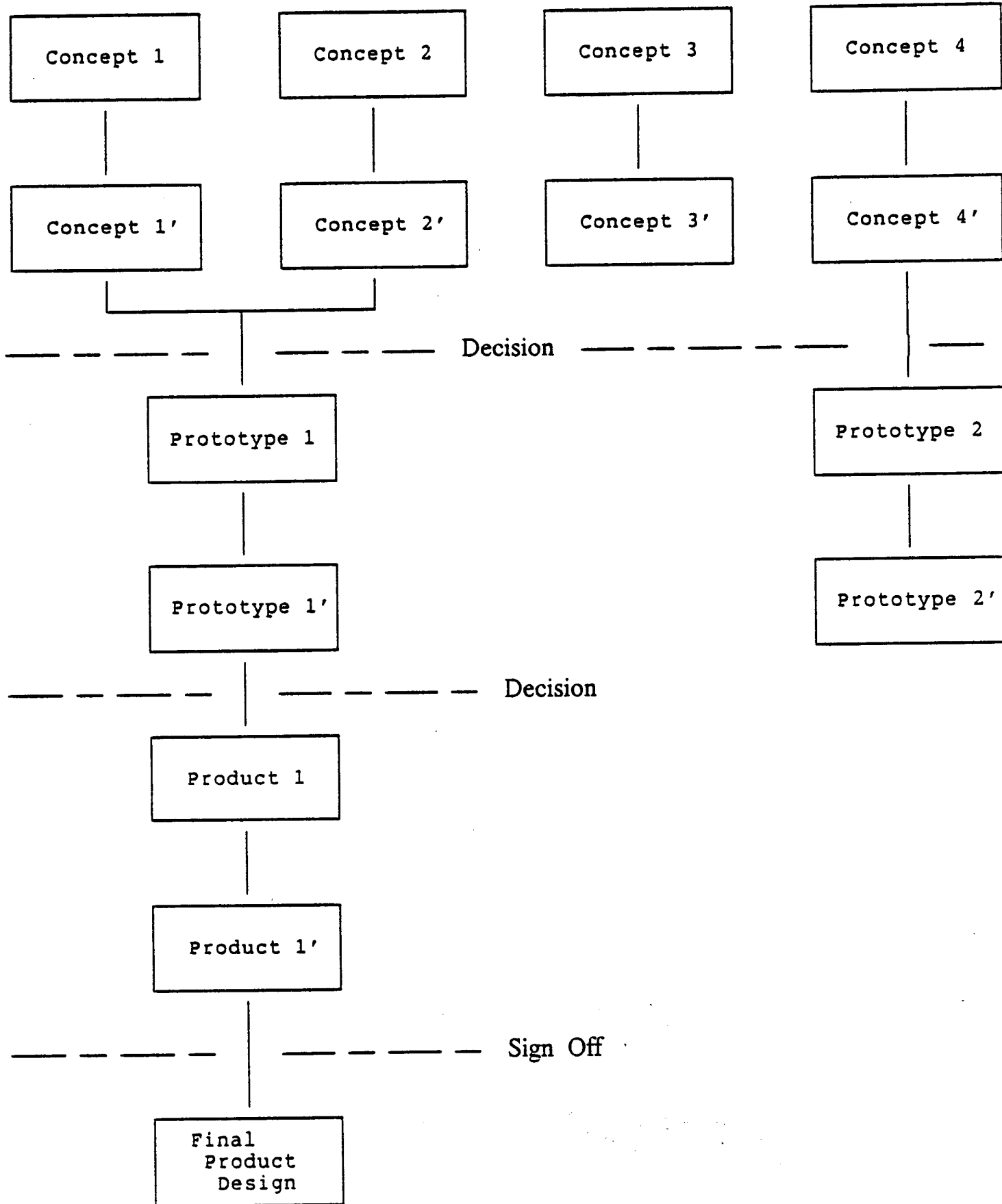
# Single Copy Product Database



# Multiple Copy Product Database

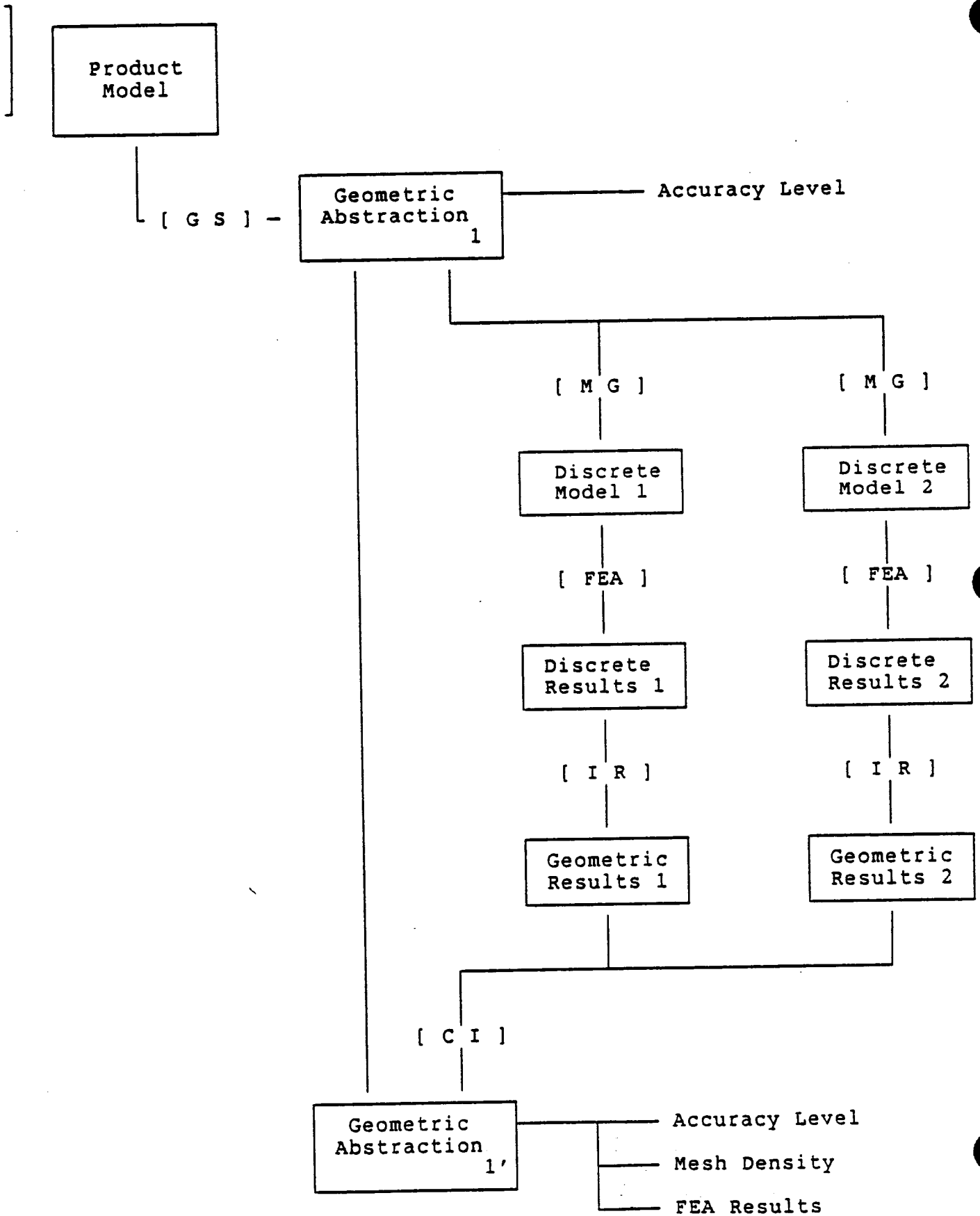


# Product Development Process

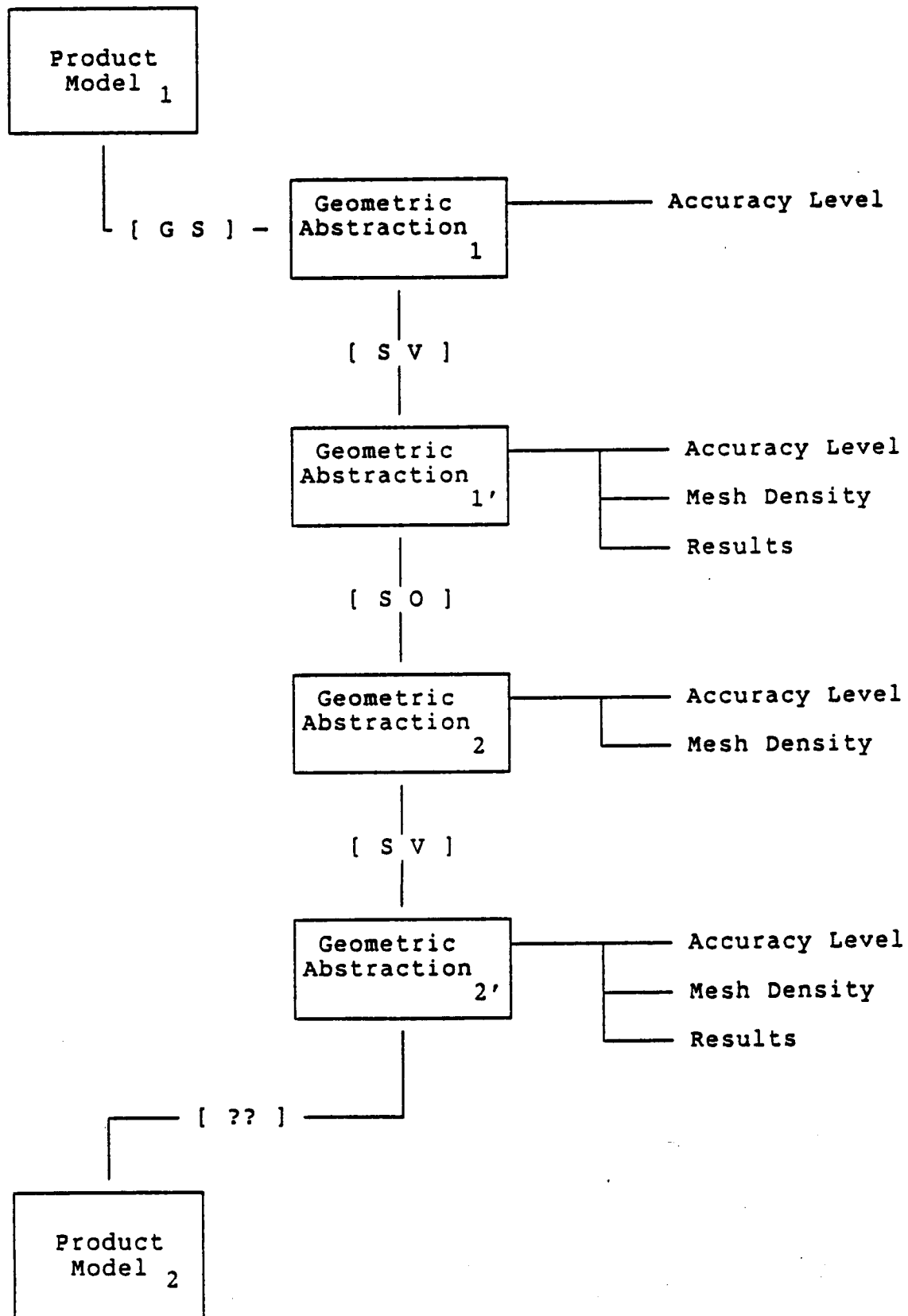




# Structural Verification Process



# Structural Optimization Process



# Geometric Modeling in Transition

- SDRC Direction
  - Geometry Data and Procedures as Utility to the MCAE System
  
- Mechanical Product Definition
  - Increasingly Application Independent over time
  - Common Geometry
  
- Product Development Process
  - Flexible Database
  
- Structural Analysis
  - Use Geometric Model not the Finite Element Model
    - Geometric Abstraction
    - Mesh Generation
    - Result Interpolation

N88-19115

54-61

125790

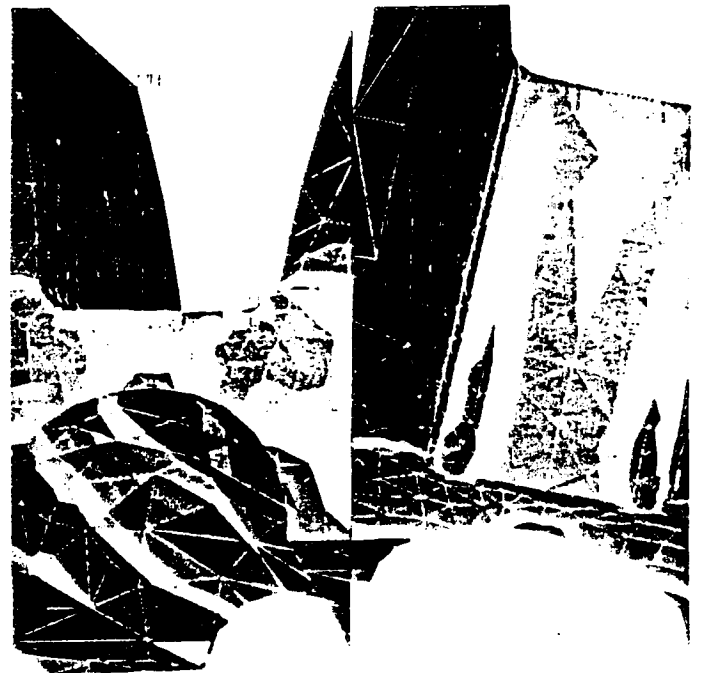
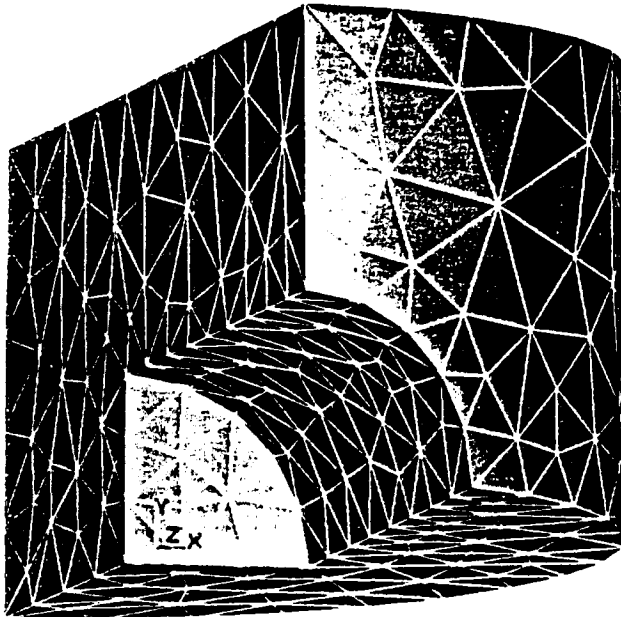
178

# Finite Element Meshing of ANSYS® Solid Models

F. S. Kelley, Supervisor, General Development  
Swanson Analysis Systems, Inc.  
P. O. Box 65, Johnson Road  
Houston, Pennsylvania 15342  
Telephone: (412) 746-3304

ORIGINAL PAGE IS  
OF POOR QUALITY

Presentation at the Workshop on the Integration of  
Finite Element Modeling with Geometric Modeling  
Rensselaer Polytechnic Institute  
May 1987



(All images are from ANSYS Revision 4.3)

## 1. INTRODUCTION - ANSYS AND SASI HISTORY

Swanson Analysis Systems, Inc. (SASI) was founded in 1970 by Dr. John A. Swanson to develop, support, and market ANSYS, a large scale, general purpose finite element computer program. ANSYS and the recently introduced ANSYS-PC products remain the only reasons for SASI's existence. There is no engineering consulting practice to distract attention away from the software business. SASI currently employs over 100 people at its office near Pittsburgh, Pennsylvania, and there are thirty regional support distributors marketing and supporting ANSYS worldwide.

ANSYS was developed solely for the commercial market, with no government or university funding. It has more than 1000 installations to date, including universities, but not PC's.

## 2. PURPOSE OF ANSYS SOLID MODELING

ANSYS was perhaps the first commercially available program to offer truly interactive finite element model generation. (In the late 1970's, there was confusion about what constitutes "interactive" processing. Some programs would simply prompt users for a fixed sequence of commands.) ANSYS Revision 3, released in August 1978, contained PREP7. This processor allowed a user to create, display, and modify a finite element mesh in whatever order desired.

ANSYS Revision 3 also contained a powerful 3-dimensional automatic mesh generator called PREP5. Based on keypoints, lines, areas, and volumes, this processor created brick models with relative ease. It was also capable of automatic application of boundary conditions. PREP5 was never as popular for model creation as PREP7. However, some users were upset when we removed PREP5 at Revision 4.0, released in 1982.

ANSYS Revision 4.0 (1982) introduced the PREP7 Mesh module, with powerful automatic quadrilateral and brick meshing capabilities. The 4.0 MESH module was widely used for model generation, but it could not handle irregular regions.

The ANSYS PREP7 MESH module was rewritten as a solid modeler for Revision 4.2 (1985), and enhanced in Revision 4.3 (to be released in June 1987). This was done solely to aid ANSYS users in the creation of finite element analysis models. SASI did not have to patch finite element meshing into the ANSYS solid modeler as an afterthought. It was designed in from the beginning.

From SASI's point of view, any other benefits which may be derived from the creation of a solid model in ANSYS (such as pretty pictures) are bonuses rather than primary objectives.

## 3. ANSYS REVISION 4.3 SOLID MODELS

ANSYS solid models are internally stored in several forms. The first of these has been well documented in textbooks and papers. Lines, surfaces, and volumetric regions are defined by Hermite cubic splines as shown below. The parameters r, s, and t vary from 0.0 to 1.0. See figures 1 - 3.

$$\begin{aligned} X &= C1 + C2 \cdot r + C3 \cdot r^2 + C4 \cdot r^3 && \text{(line)} \\ Y &= C5 + C6 \cdot r + C7 \cdot r^2 + C8 \cdot r^3 \\ Z &= C9 + C10 \cdot r + C11 \cdot r^2 + C12 \cdot r^3 \end{aligned}$$

$$\begin{aligned} X &= C1 + C2 \cdot r + C3 \cdot r^2 + C4 \cdot r^3 && \text{(surface)} \\ &+ (C5 + C6 \cdot r + C7 \cdot r^2 + C8 \cdot r^3) (s) \\ &+ (C9 + C10 \cdot r + C11 \cdot r^2 + C12 \cdot r^3) (s^2) \\ &+ (C13 + C14 \cdot r + C15 \cdot r^2 + C16 \cdot r^3) (s^3) \\ Y &= C17 + C18 \cdot r + C19 \cdot r^2 + C20 \cdot r^3 \\ &+ (C21 + C22 \cdot r + C23 \cdot r^2 + C24 \cdot r^3) (s) \\ &+ (C25 + C26 \cdot r + C27 \cdot r^2 + C28 \cdot r^3) (s^2) \\ &+ (C29 + C30 \cdot r + C31 \cdot r^2 + C32 \cdot r^3) (s^3) \\ Z &= C33 + C34 \cdot r + C35 \cdot r^2 + C36 \cdot r^3 \\ &+ (C37 + C38 \cdot r + C39 \cdot r^2 + C40 \cdot r^3) (s) \\ &+ (C41 + C42 \cdot r + C43 \cdot r^2 + C44 \cdot r^3) (s^2) \\ &+ (C45 + C46 \cdot r + C47 \cdot r^2 + C48 \cdot r^3) (s^3) \end{aligned}$$

$$\begin{aligned} X &= C1 + C2 \cdot r + C3 \cdot r^2 + C4 \cdot r^3 && \text{(volumetric region)} \\ &+ (C5 + C6 \cdot r + C7 \cdot r^2 + C8 \cdot r^3) (s) \\ &+ (C9 + C10 \cdot r + C11 \cdot r^2 + C12 \cdot r^3) (s^2) \end{aligned}$$

$$\begin{aligned}
& + (C13 + C14 \cdot r + C15 \cdot r^2 + C16 r^3) (s^3) \\
& + [C17 + C18 \cdot r + C19 \cdot r^2 + C20 r^3 \\
& + (C21 + C22 \cdot r + C23 \cdot r^2 + C24 r^3) (s) \\
& + (C25 + C26 \cdot r + C27 \cdot r^2 + C28 r^3) (s^2) \\
& + (C29 + C30 \cdot r + C31 \cdot r^2 + C32 r^3) (s^3) ] [t] \\
& + [C33 + C34 \cdot r + C35 \cdot r^2 + C36 r^3 \\
& + (C37 + C38 \cdot r + C39 \cdot r^2 + C40 r^3) (s) \\
& + (C41 + C42 \cdot r + C43 \cdot r^2 + C44 r^3) (s^2) \\
& + (C45 + C46 \cdot r + C47 \cdot r^2 + C48 r^3) (s^3) ] [t^2] \\
& + [C49 + C50 \cdot r + C51 \cdot r^2 + C52 r^3 \\
& + (C53 + C54 \cdot r + C55 \cdot r^2 + C56 r^3) (s) \\
& + (C57 + C58 \cdot r + C59 \cdot r^2 + C60 r^3) (s^2) \\
& + (C61 + C62 \cdot r + C63 \cdot r^2 + C64 r^3) (s^3) ] [t^3]
\end{aligned}$$

(The equations for Y and Z are similar, using C65 through C192.)



Figure 1 Hermite Spline Defining a Line

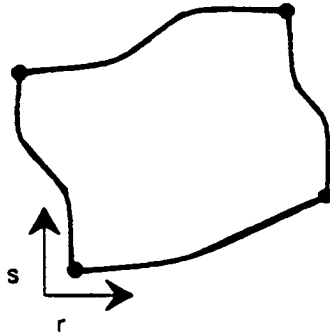


Figure 2 Bicubic Hermite Spline Defining a Surface Region

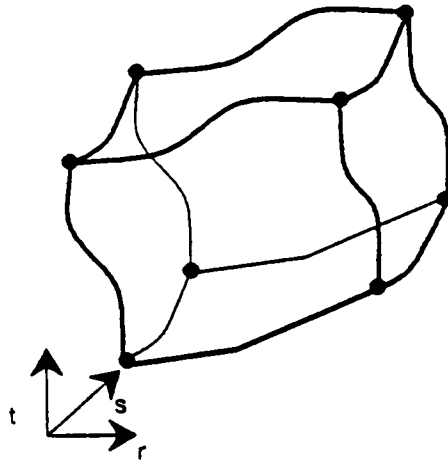


Figure 3 Tricubic Hermite Spline Defining a Volumetric Region

ANSYS also allows the definition of degenerate Hermite regions (figures 4 and 5). This is very important. There is no assurance that an arbitrary surface can be mapped by quadrilateral regions, and even less assurance that an arbitrary 3-dimensional object can be mapped by hexahedral regions. The degenerate forms give ANSYS a far more general modeling capability than would be provided by the standard regions.

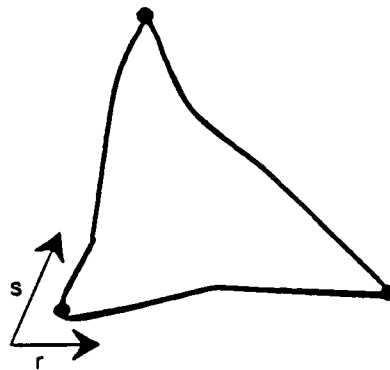


Figure 4 Degenerate Surface Region

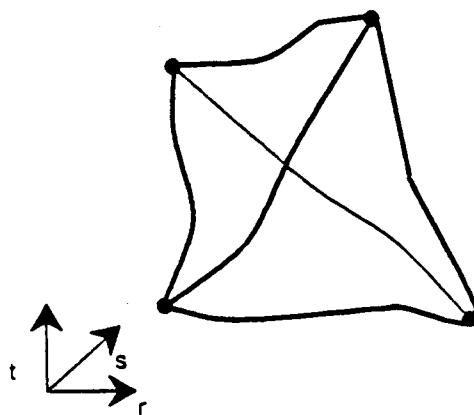


Figure 5 Degenerate Volumetric Region

New to ANSYS Revision 4.3 is the ability to define surface regions by a list of up to 200 cubic line segments, and volumetric regions by a list of up to 200 bicubic surface regions (figures 6 and 7). These alternate region types allow great flexibility in the modeling of complex structures. They also make it difficult to classify the ANSYS solid modeler into one category, such as "B-rep" or "CSG". Perhaps "hybrid CSG" is the best term to apply.

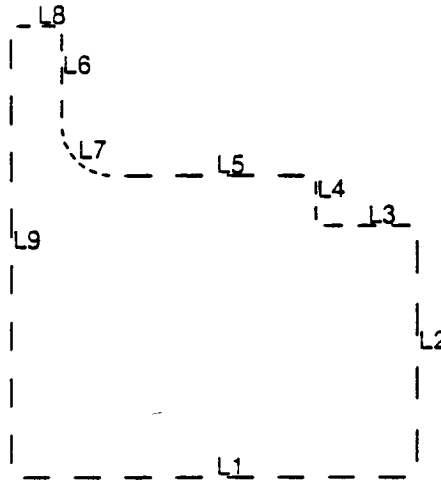


Figure 6 Surface Region Defined by a List of Lines

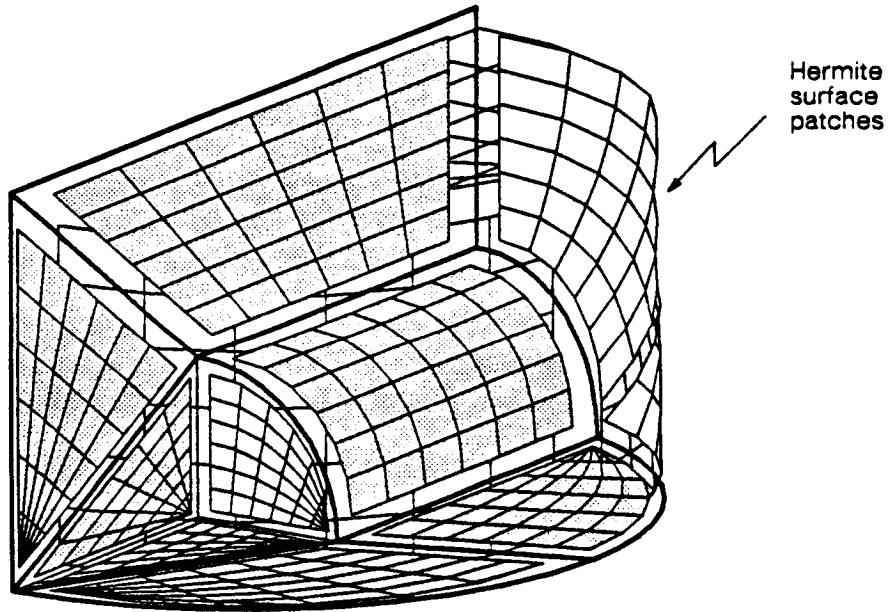


Figure 7 Volumetric Region Defined by a List of Surface Patches

Definition of solid models in ANSYS begins with the input of several "keypoints". "Line segments", "areas", and "volumes" may be defined by connecting keypoints. Lower order entities are generated automatically as needed. Lines and areas follow the curvature of the "currently active coordinate system" (figure 8). Translation, rotation, and symmetry reflection operations are available. Line segments may be rotated about an axis or dragged along a path to produce areas (figure 9). Areas may be rotated about an axis or dragged along a path to produce volumes



(figure 10). ANSYS is command driven, with complete documentation available on-line via a menu system. Cross hair and digitizing tablet input is also possible.

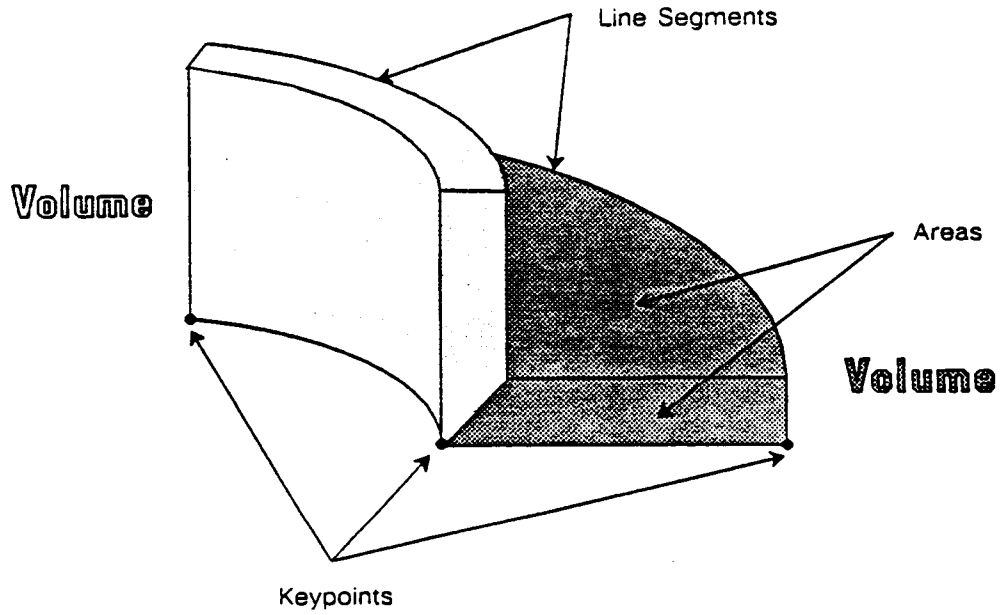


Figure 8 ANSYS Keypoints, Line Segments, Areas, and Volumes

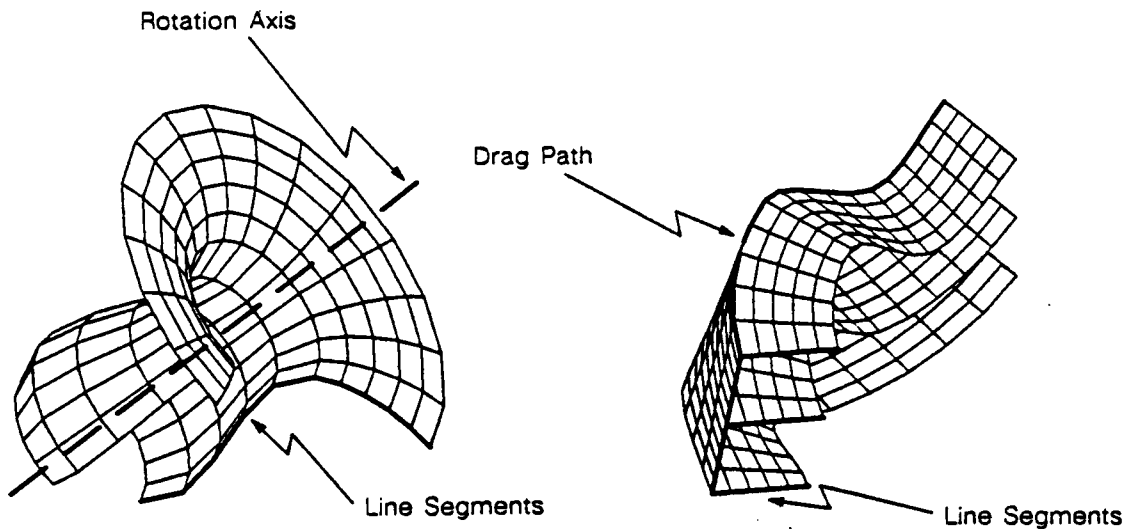


Figure 9 Rotation and Dragging of Line Segments to Create Areas

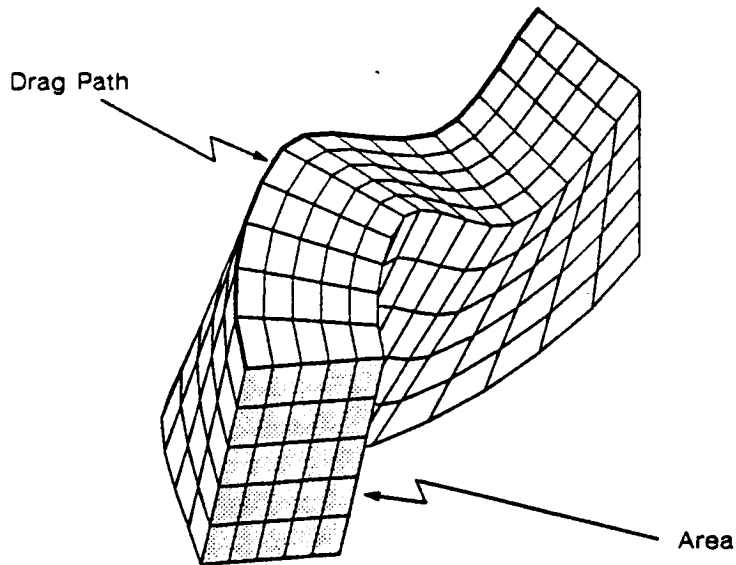


Figure 10 Dragging of an Area to Create Volumes

#### 4. SURFACE ACCURACY OF ANSYS SOLID MODELS

Accuracy of curved surfaces in cubic spline based solid modelers can be of concern. Circular arcs and intersection lines cannot be represented exactly by Hermite splines. A circular arc of 90 degrees has a radius error of 0.03 % (figure 11). For an arbitrary region extending 90 degrees on the surface of a cylinder, the radius error can be as much as 0.2 % (figure 12). Lines resulting from the intersection of arbitrary 90 degree regions can have a radius error of 0.4 % (figure 13). Figure 14 shows the effect of a  $\pm 0.5$  % local perturbation of radius on the results of the analysis of a flat plate with a hole. The maximum corner stress decreased by 0.8 % as a result of the perturbation. For solid elements, the stress error appears to be of the same order of magnitude as the geometric error. Figure 15 shows the effect of a  $\pm 0.5$  % local perturbation of radius on the results of the analysis of a pressurized spherical shell. The stress error introduced was approximately 8 %. For shell elements, the stress error appears to be an order of magnitude higher than the geometric error. The radius error in the ANSYS solid modeler is drastically reduced if the line segments and areas are limited to spans of 45 degrees or less. Typical radius errors are then 0.0005 % for line segments, 0.005 % for areas, and 0.03 % for intersection lines.

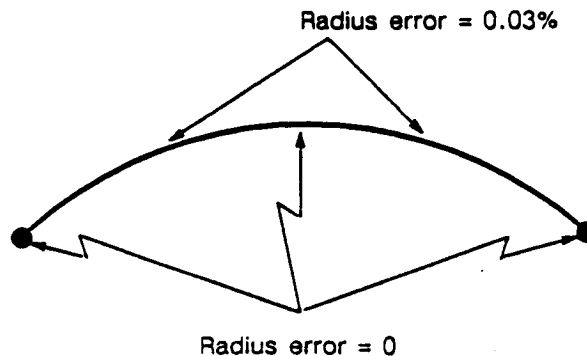


Figure 11 Radius Error for a 90 Degree Line Segment

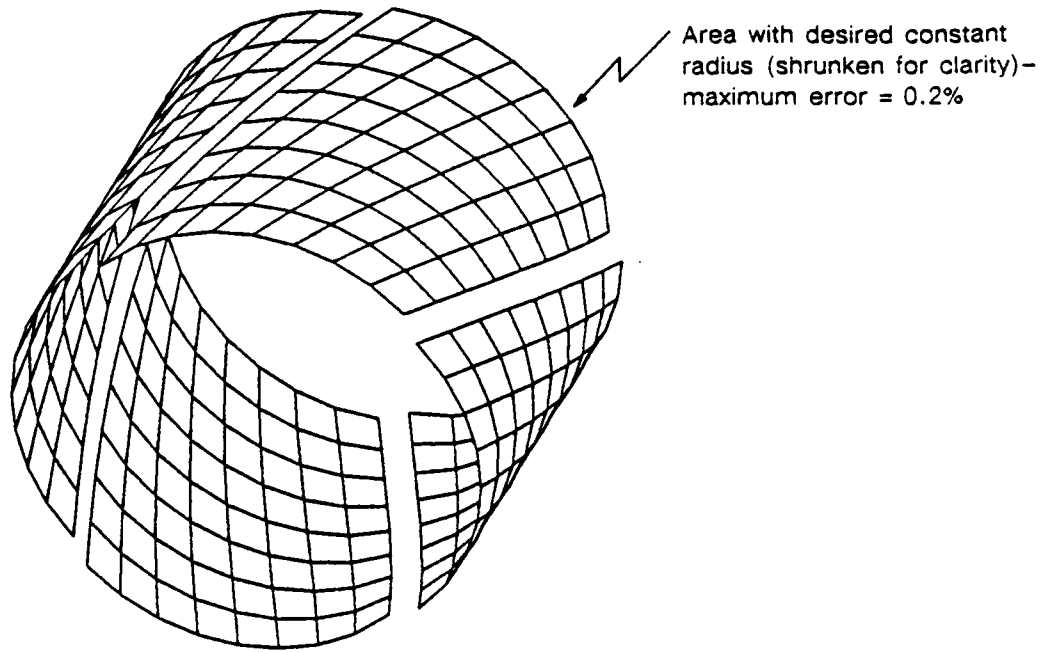


Figure 12 Radius Error for Arbitrary 90 Degree Area

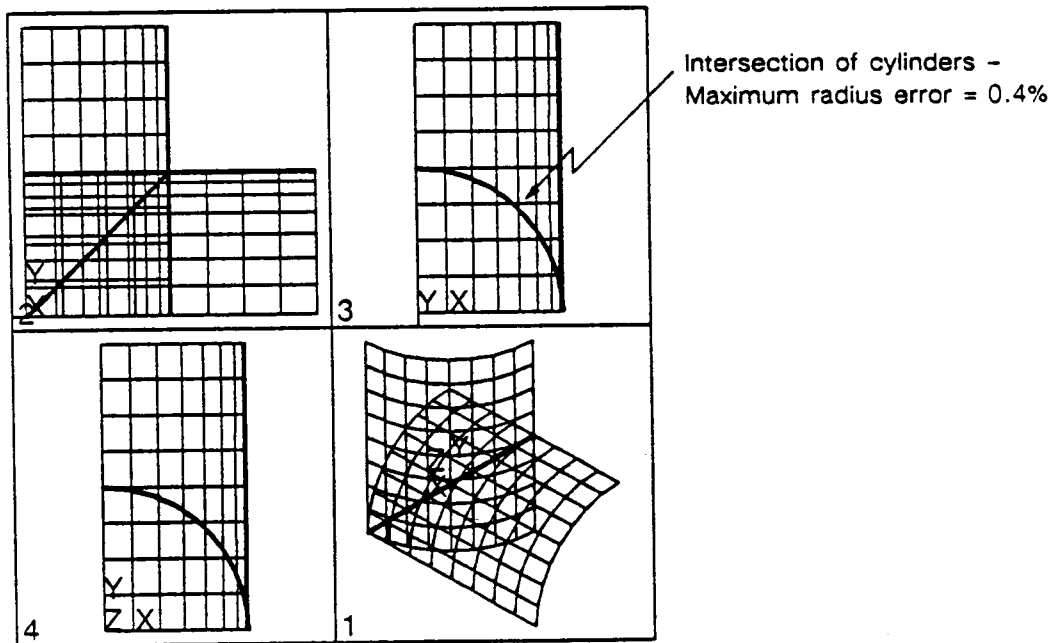


Figure 13 Radius Error for Intersection of 2 Arbitrary 90 Degree Areas

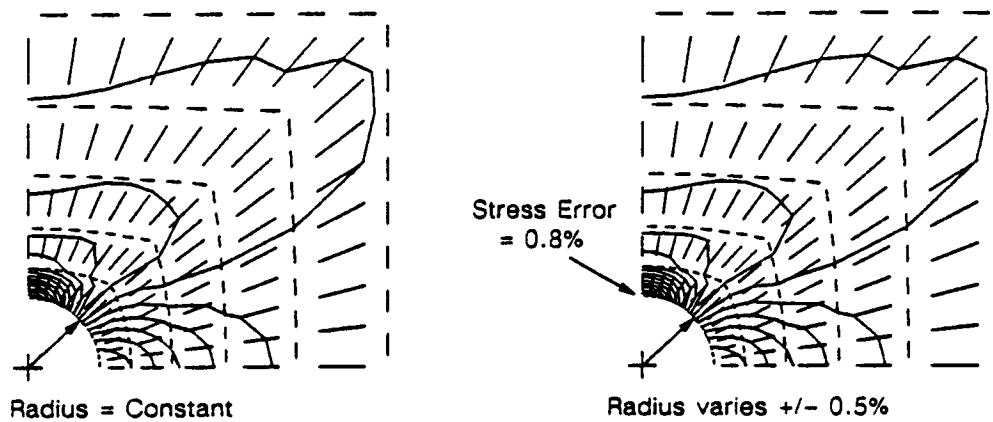


Figure 14 Effect of Radius Error on Plane Stress Solution

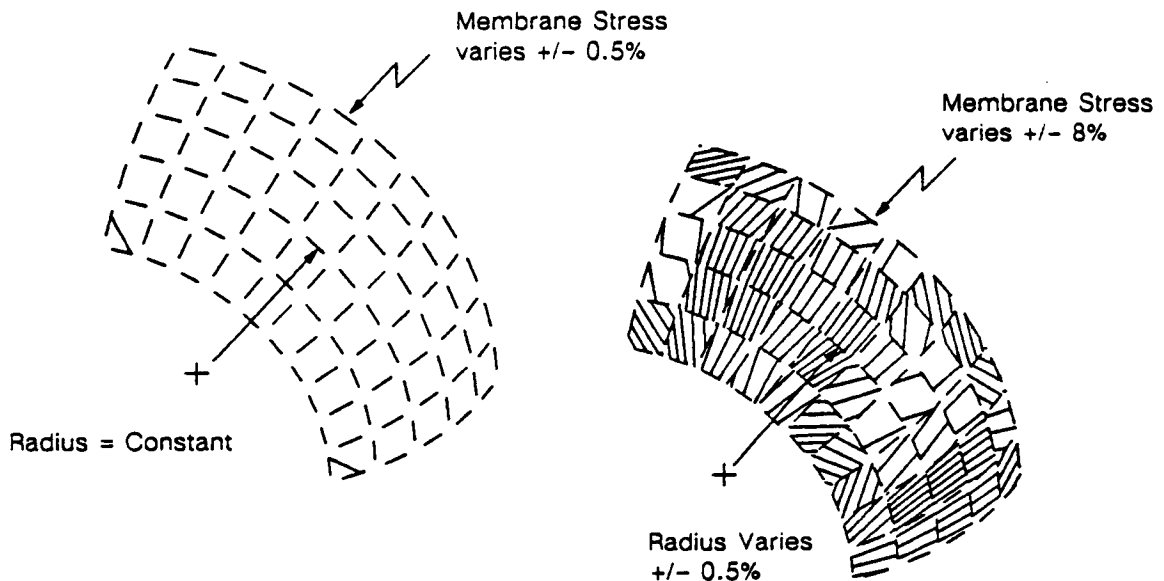


Figure 15 Effect of Radius Error on Shell Stress Solution

## 5. FINITE ELEMENT MESHING OF AN ANSYS SOLID MODEL

The first step in meshing of an ANSYS solid model is to establish the mesh density. This is accomplished by assigning a number of element divisions and a spacing ratio to every line segment attached to the areas or volumes (figure 16). Commands are available for making the assignments line segment by line segment or to a group of line segments at once. Divisions can be computed based on line segment length and a desired element size and assigned automatically. Spacing ratios can also be computed automatically for smooth mesh transitioning (figure 17).

Divisions Established  
for All Line Segments

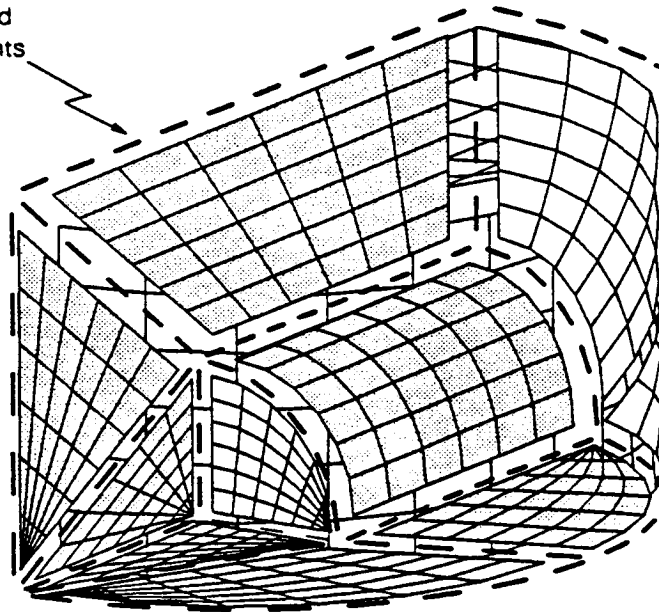


Figure 16 Establishing Finite Element Mesh Density

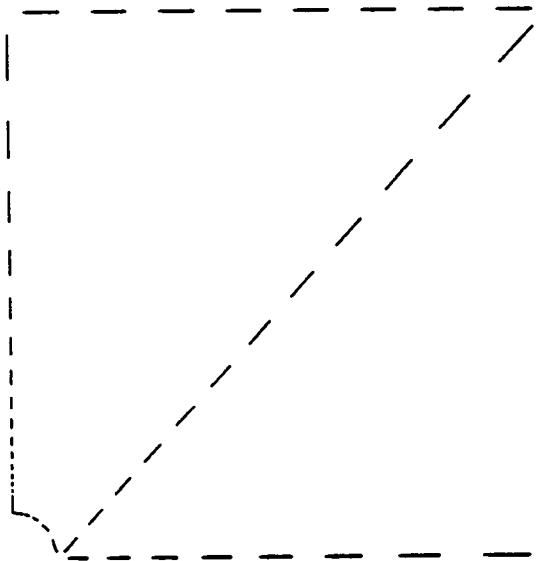


Figure 17 Automatically Adjusted Line Divisions and Spacing Ratios

Meshing of areas with quadrilateral elements and volumes with brick elements is available in certain cases. The most limiting restriction is that only the standard region shapes (four keypoints on areas, eight keypoints on volumes) are allowed. Further, the number of element divisions requested must match on opposing sides of areas (figure 18). The area corner angles must be reasonable for quadrilateral or brick elements. The mesh is mapped onto the natural coordinates of the areas and volumes (figures 19 and 20).

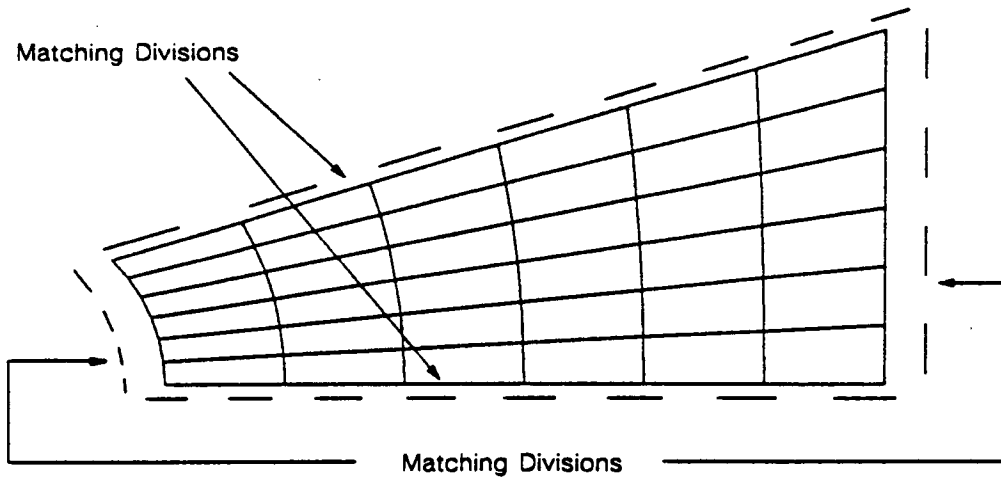


Figure 18 Matching Divisions on Areas Required for Quadrilateral or Brick Meshing

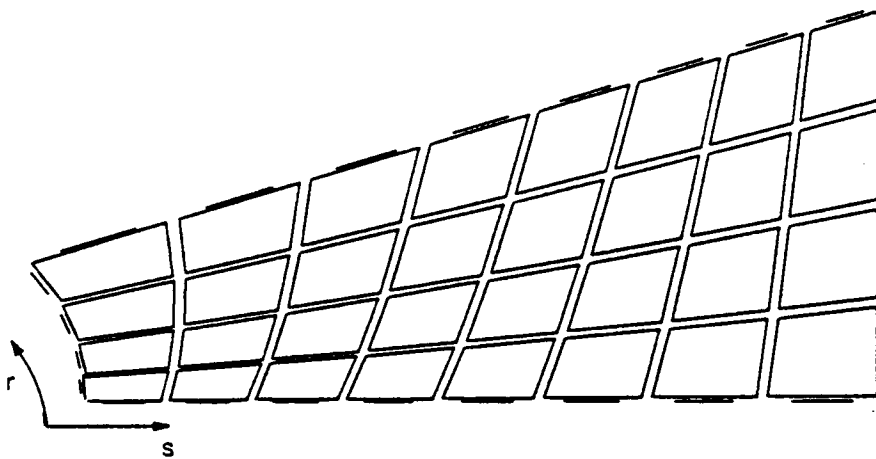


Figure 19 Quadrilateral Element Meshing of an Area

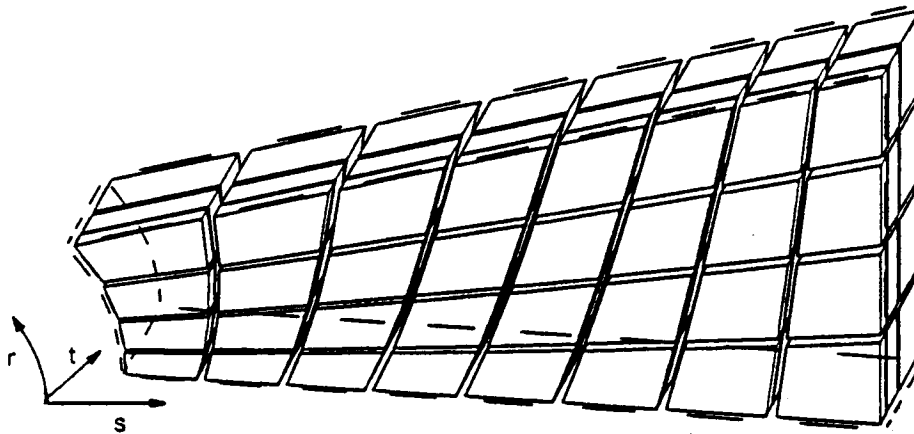


Figure 20 Brick Element Meshing of a Volume

Meshing with triangles is available for all areas, regular or not. Meshing with tetrahedra is available for all volumes, regular or not. The elements of choice are the 6-noded triangular solid or shell (figure 21) and the 10-noded tetrahedral solid (figure 22). ANSYS has these elements available for stress, thermal, electro-magnetic, or multi-field analysis. For planar, axisymmetric, or shell applications, 6-noded triangles are good performers, giving results of equal or superior quality for equal edge divisions when compared to 4-noded or 8-noded quadrilaterals. For 3-dimensional solid applications, 10-noded tetrahedra perform well. (This element is a theoretically consistent, completely conforming element which passes the patch test. Because tetrahedral meshes are rarely symmetric, however, this element can develop localized spurious deformation modes. For this reason, some theoreticians have refused to bless this element for general use. This is really bad news if their fear is justified, since tetrahedral meshing is the only reasonable approach to automated meshing of arbitrary 3-dimensional shapes. No conclusions can be reached, however, until the stress analysis community has had ample opportunity to gain experience with tetrahedra.) ANSYS uses the same algorithm for triangular meshing of areas and tetrahedral meshing of volumes: an initial mesh is formed without regard to region shape and is then repeatedly improved by operations which divide or combine elements, until all elements are nicely shaped or until no operations available will improve the situation. This iterative scheme is computationally intensive, but is highly reliable and produces well distributed meshes of well shaped elements.

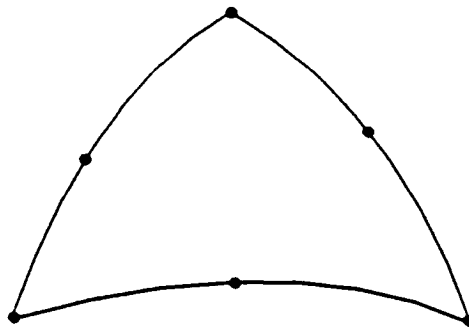


Figure 21 6-Noded Triangle Element

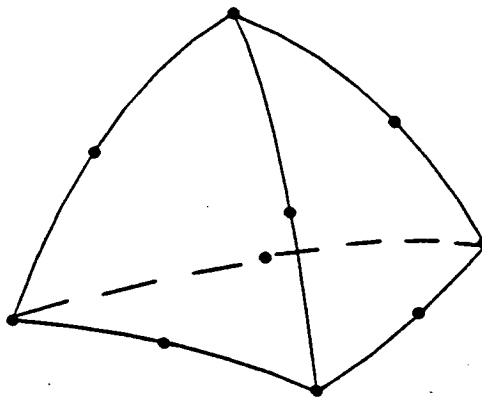


Figure 22 10-Noded Tetrahedron Element

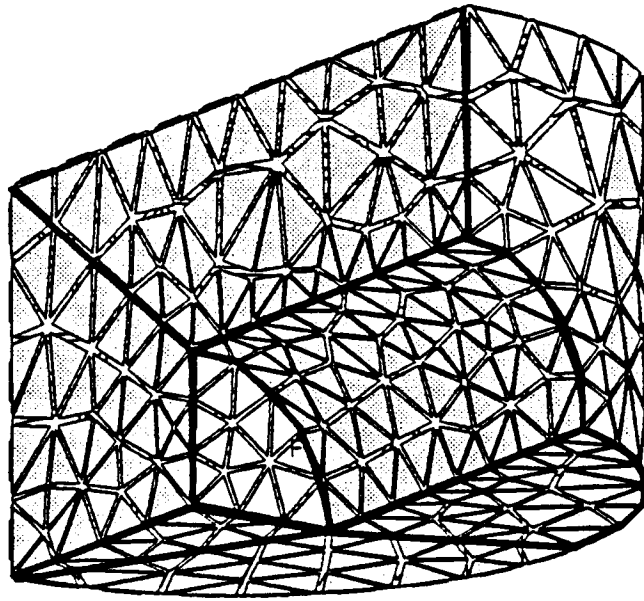


Figure 23 Triangular Mesh on the Exterior of a Volume Prior to Tetrahedron Meshing

We at SASI have been asked on several occasions why we do not mesh areas with mixtures of quadrilaterals and triangles, or mesh volumes with mixtures of bricks, wedges, and tetrahedra. First, there is little evidence to suggest that such mixed meshes are likely to perform any better than meshes consisting entirely of triangles or tetrahedra. Second, the algorithms to produce such meshes appear to be at least as complex and compute intensive as the triangle and tetrahedron algorithms, if they are to check element shape as thoroughly as they should. Finally, connecting brick elements to wedges and tetrahedra is not a straightforward process if one wishes to avoid displacement incompatibilities.

Meshing of adjacent areas or adjacent volumes in ANSYS will always produce compatible and properly interconnected finite element meshes. This is possible because the triangle and tetrahedron meshing algorithms used do not have the "authority" to alter the exterior of the mesh of a region. As shown in figure 23, the tetrahedral meshing of a volume starts with a fixed exterior triangular mesh, which cannot change.

## 6. BOUNDARY CONDITIONS

The following boundary conditions may be defined directly on an ANSYS solid model.

imposed displacements at keypoints	(stress analysis)
imposed temperatures at keypoints	(thermal analysis)
imposed voltage at keypoints	(electrical analysis)
imposed magnetic potential at keypoints	(magnetic analysis)

(constraints can be interpolated over attached line segments, areas, and volumes)

temperatures at keypoints	(stress analysis)
heat generation rates at keypoints	(thermal analysis)

(can be interpolated over attached line segments, areas and volumes)



applied forces at keypoints	(stress analysis)
applied heat input at keypoints	(thermal analysis)
applied current flow at keypoints	(electrical analysis)
applied magnetic flux at keypoints	(magnetic analysis)
pressures on line segments	(2-D stress analysis)
convections on line segments	(2-D thermal analysis)
symmetry / antisymmetry on line segments	(2-D solid or 3-D shell stress analysis)
pressures on areas	(3-D stress analysis)
convections on areas	(3-D thermal analysis)
symmetry / antisymmetry on areas	(3-D stress analysis)

Boundary conditions may be defined before or after finite element meshing, and can be displayed on the solid model. They will be transferred to the finite element model automatically when needed. (The transfer can be forced earlier if the user wishes to display them on the finite element model.)

Even if boundary conditions are not applied directly to an ANSYS solid model, they can be conveniently applied to a finite element model created by the solid modeler. Nodes and elements associated with various mesh entities can be activated or deactivated as desired, making it easy to specify where constraints or loadings belong.

## 7. ANSYS INTERFACE WITH CAD SYSTEMS

ANSYS accepts keypoint and line segment information from a number of other solid modeling systems (see Table 1). A user can use this data to create areas and/or volumes and a finite element model.

Table 1

Translations within ANSYS	Other Systems Having Some Interface with ANSYS		
IGES	ADAMS-DRAM	CIS-MEDUSA	PATRAN
MEDUSA	ADVANTAGE	DIAD Solid Modeler	PDGS
FEMVIEW	ANVIL	ENGINEER WORKS	PROF. CADAM
FEMGEN	APPLICON	EUCLID	ROMULUS-D
NASTRAN	AUTOCAD	FEMAS	SOLUTION 3000
SUPERTAB	AXXYZ	GRAFTEK	UNIGRAPHICS
	CADAM	ICAD	VERSACAD
	CADKEY	INTERGRAPH	
	CATIA	ME30	
	CDS-4000	M.E. Workbench	
	CIMLINK	MOLDFLOW	

Even though ANSYS can accept nodes and elements created by other systems, we believe that in most cases the user will be better off doing the finite element meshing step within ANSYS. First, we have seen evidence (finite element models from other systems) that not all developers of meshing software know what constitutes a good analysis model. Badly shaped elements may give poor quality analysis results. (It is far better to inform the user that meshing is not possible with

the data supplied than to produce an unacceptable mesh.) Second, if the user has meshed in another system, he or she may be reluctant to make any alterations to the model which may be indicated by initial analysis results. Third, the required mesh may be load dependent. Finally, nodes and elements brought into ANSYS from another system will not be associated with the solid model, and boundary condition manipulation will be difficult.

## 8. FUTURE DEVELOPMENT PLANS

In the short term (Revision 4.4, 1988), we plan improvements in the command structure for defining ANSYS solid models. We hope to improve the speed and reliability of our meshing algorithms. We plan to allow definition of contact surfaces. We plan to improve our interfaces with other software packages.

In the long term (Revision 5, 1990), we want to address some or all of the following.

- mapping analysis results back onto the solid model
- adaptive mesh refinement
- improved curved surface accuracy
- improved user interface

## 9. EXAMPLES

Figures 24 through 31 show several examples of ANSYS solid models and resulting finite element meshes. Table 2 shows the various statistics for these models.

**Table 2**  
**Example Statistics**

<u>Model</u>	<u>Number of Commands</u>	<u>Number of Elements</u>	<u>Elapsed time* for creation of solid model &amp; finite element model</u>	<u>Computer</u>
Block with two holes	159	3047 tetrahedra	183 minutes	VAX 11/780
Helix	62	2856 tetrahedra	160 minutes	Prime 9950
Pawn	60	819 triangles	19 minutes	MicroVax
Knight	351	2427 tetrahedra	318 minutes	MicroVax
Gear	279	1136 tetrahedra	31 minutes	Prime 9950
Gear Submodel	186	2684 tetrahedra	138 minutes	Prime 9950
Turbine Spacer	403	1224 tetrahedra	75 minutes	Prime 9950

\*CP times are nearly identical

ORIGINAL PAGE IS  
OF POOR QUALITY.

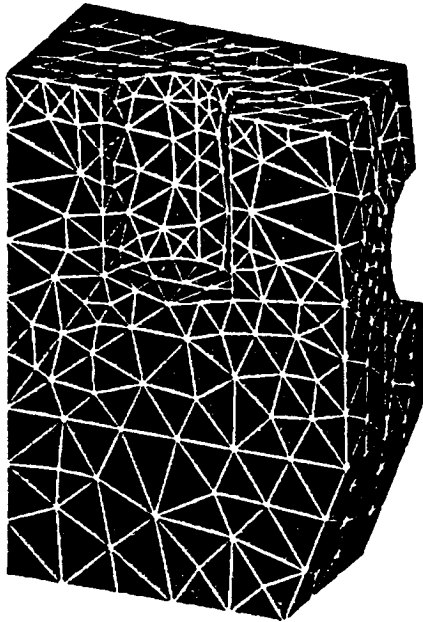


Figure 24 Tetrahedron Model of Block with Two Holes

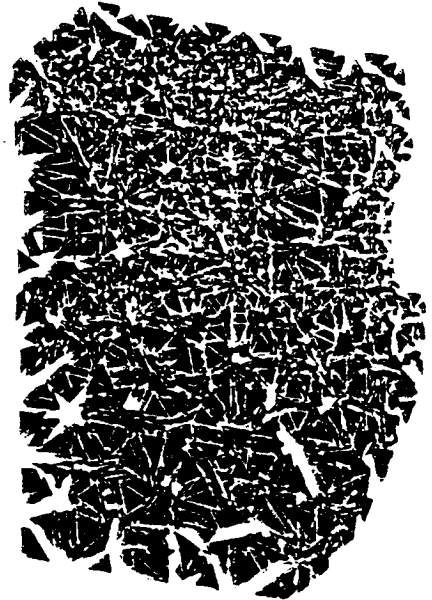


Figure 25 Detail of Tetrahedron Model - Block with Two Holes

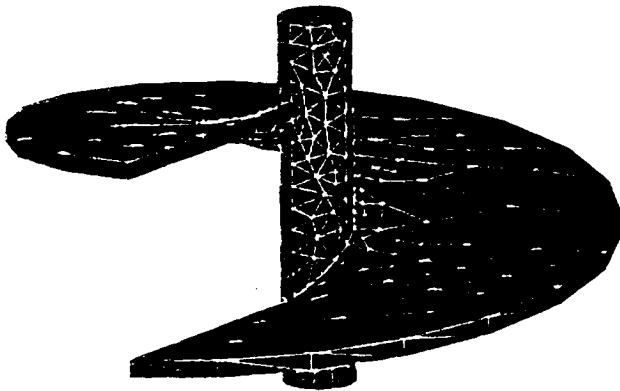


Figure 26 Tetrahedron Model of Helix

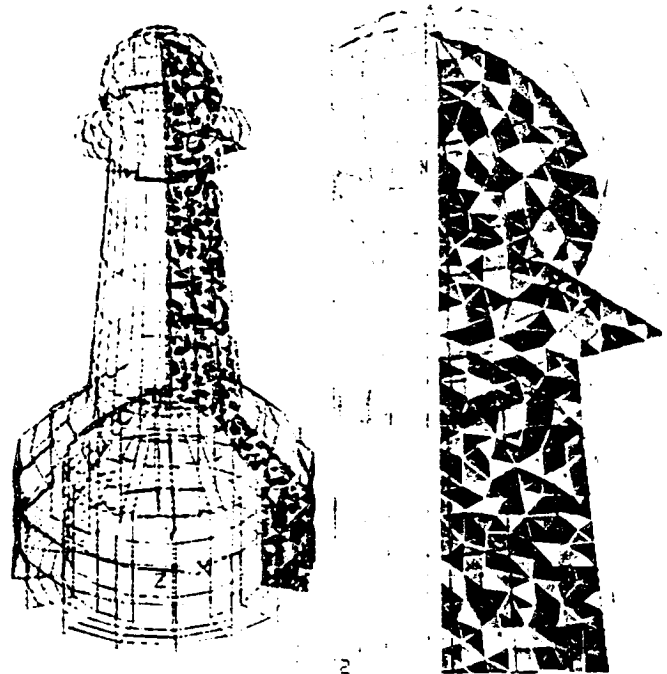


Figure 27 Triangle Model of Pawn

ORIGINAL PAGE IS  
OF POOR QUALITY

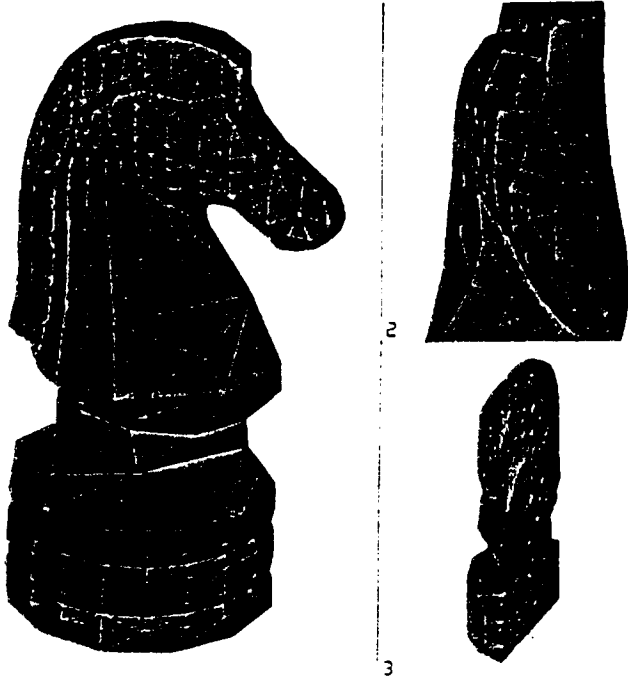


Figure 28 Tetrahedron Model of Knight

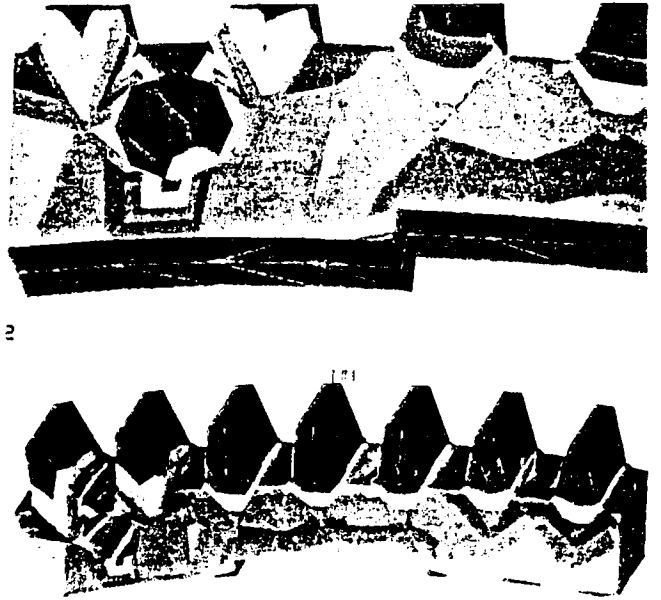


Figure 29 Stress Contours on Tetrahedron Model of Gear

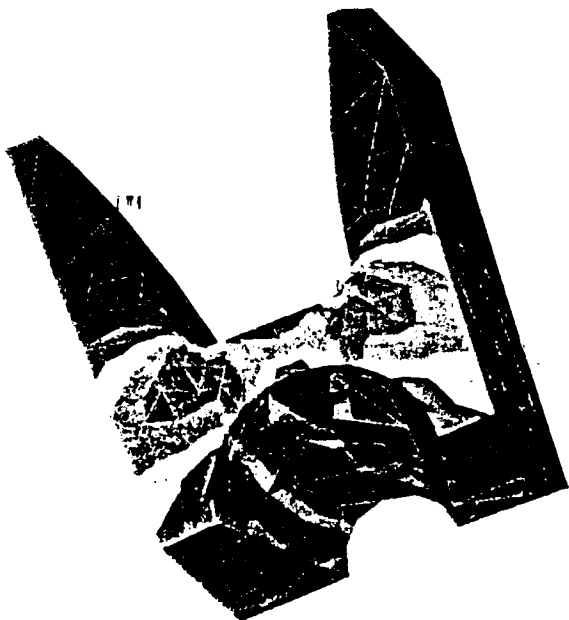


Figure 30 Stress Contours on Tetrahedron Submodel of Gear

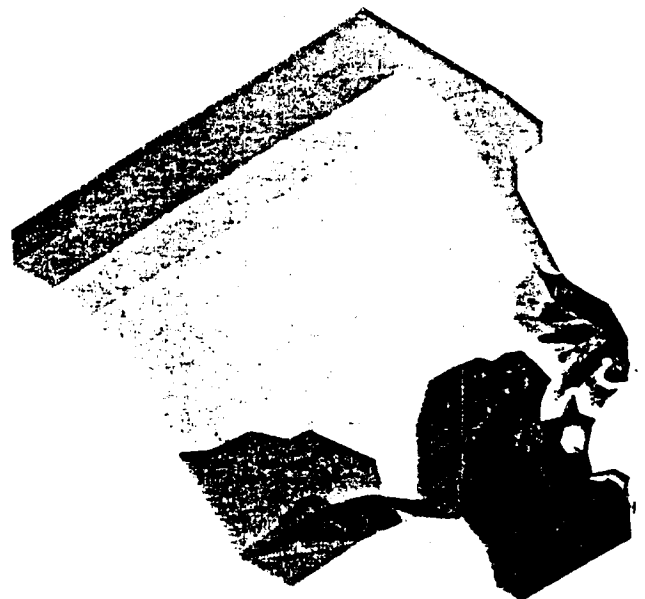


Figure 31 Stress Contours on Tetrahedron Model of Turbine Spacer

## N88-19116

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 hierarchical 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 parametric 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 implemented 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 capability 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 primitives to be supported include all the geometric primitives plus all the topologically 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.

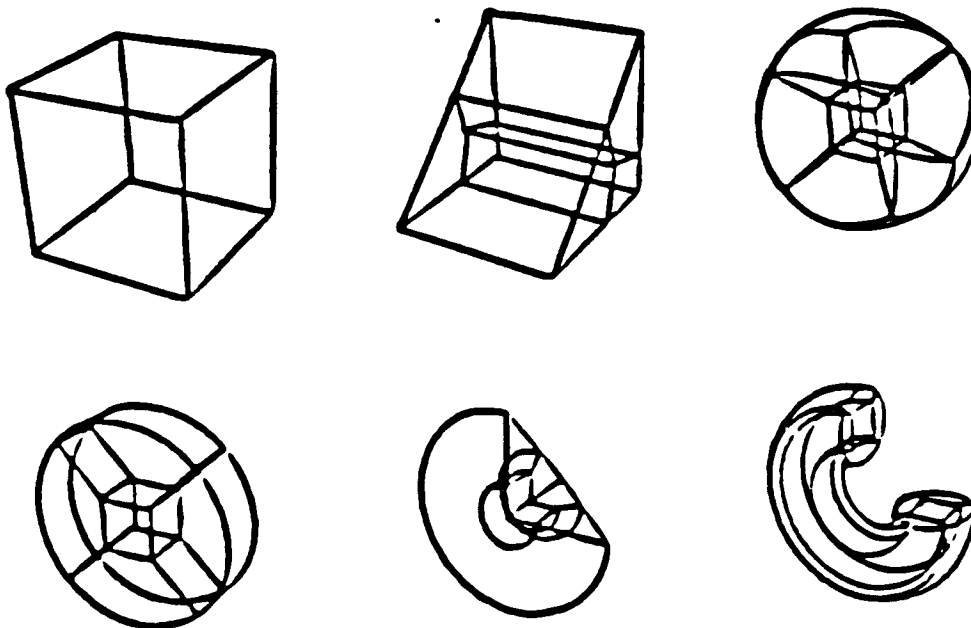


Figure 1. Decomposition of the Standard Geometric Primitives into Hexahedron.

#### 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 incorporated 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

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

where  $V_A$  is the "Area-Pull" movement vector (corresponding to  $V_L$  in the node equidistribution) applied to the node in question. The  $A_{if}$  and  $A_{ib}$  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  $\|*\|_{L^\infty}$  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  $D^2 u$  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  $\nabla u$  by  $\nabla u_h$ . For quadratic elements the second inequality can be used replacing  $D^2 u$  with  $D^2 u_h$ . 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 analysis. 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



ORIGINAL PAGE IS  
OF POOR QUALITY

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.

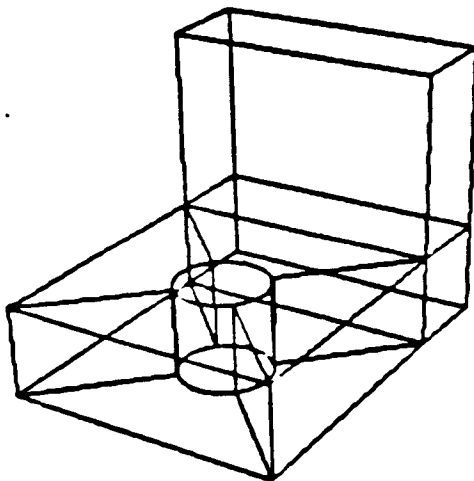


Figure 2. The Solid Model of the Primitive Decomposition

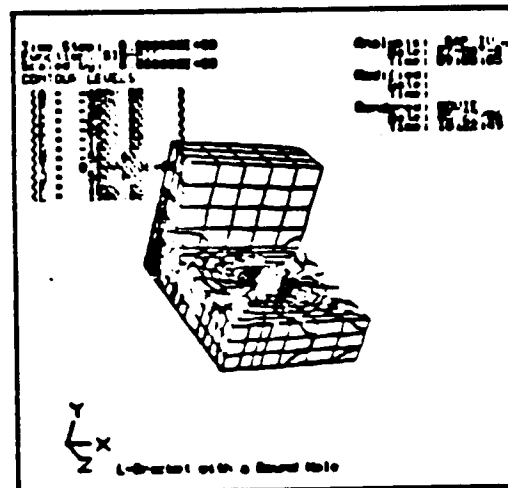


Figure 3. The Minimum Principal Stress Contours on the Hexahedral Mesh Generated.

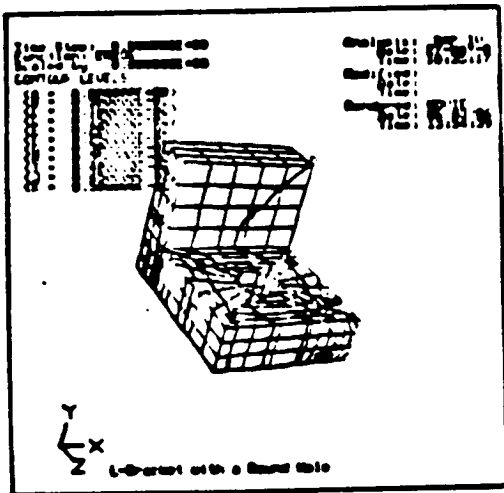


Figure 4. Error Contours from the Automatic Error Analysis

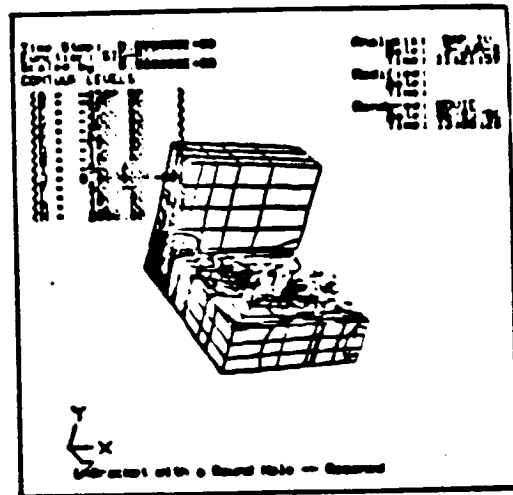


Figure 5. The Minimum Principal Stress Contours After Adaptive Gridding

#### 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

N88-19117

56-61

ABS. ONLY

125792  
10

## OCTREE BASED AUTOMATIC MESHING FROM CSG MODELS

Renato Perucchio, Assistant Professor  
Department of Mechanical Engineering  
University of Rochester  
Rochester, NY 14627

### Abstract

Finite element meshes derived automatically from solid models through recursive spatial subdivision schemes (octrees) can be made to inherit the hierarchical structure and the spatial addressability intrinsic to the underlying grid. These two properties, together with the geometric regularity that can also be built into the mesh, make octree based meshes ideally suited for efficient analysis and self-adaptive remeshing and reanalysis. Our presentation is focussed on the element decomposition of the octal cells that intersect the boundary of the domain. The problem, central to octree based meshing, is solved by combining template mapping and element extraction into a procedure that utilizes both constructive solid geometry and boundary representation techniques. Boundary cells that are not intersected by the edge of the domain boundary are easily mapped to predefined element topology. Cells containing edges (and vertices) are first transformed into a planar polyhedron and then triangulated via element extractors. We also analyze the modelling environments required for the derivation of planar polyhedra and for element extraction.

~~QMIT~~

**PRODUCTION AUTOMATION PROJECT**

College of Engineering & Applied Science  
The University of Rochester  
Rochester, New York 14627

**GEOMETRICAL AND TOPOLOGICAL ISSUES  
IN OCTREE BASED AUTOMATIC MESHING**

by

Mukul Saxena and Renato Perucchio

to be presented at NAFEMS International Conference  
on Quality Assurance and Standards in Finite Element Analysis

to be held

May 13-15, 1987

in

Brighton, England

The work described in this paper is supported by the companies in the P.A.P.'s Industrial Associates Program. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the opinions of the industrial sponsors or the University of Rochester.

**GEOMETRICAL AND TOPOLOGICAL ISSUES IN OCTREE  
BASED AUTOMATIC MESHING**

Mukul Saxena and Renato Perucchio\*  
Production Automation Project  
and Department of Mechanical Engineering  
University of Rochester  
Rochester, N.Y. 14627, U.S.A.

**SUMMARY**

Finite element meshes derived automatically from solid models through recursive spatial subdivision schemes (octrees) can be made to inherit the hierarchical structure and the spatial addressability intrinsic to the underlying grid. These two properties, together with the geometric regularity that can also be built into the mesh, make octree based meshes ideally suited for efficient analysis and self-adaptive remeshing and reanalysis. This paper discusses the element decomposition of the octal cells that intersect the boundary of the domain. The problem, central to octree based meshing, is solved by combining template mapping and element extraction into a procedure that utilizes both constructive solid geometry and boundary representation techniques. Boundary cells that are not intersected by the edge of the domain boundary are easily mapped to predefined element topology. Cells containing edges (and vertices) are first transformed into a planar polyhedron and then triangulated via element extractors. This paper also analyzes the modelling environments required for the derivation of planar polyhedra and for element extraction.

**1 INTRODUCTION**

In this paper, we describe an approach for resolving the geometrical and topological issues that arise when a recursive spatial subdivision scheme (octree) is used to generate automatically a FEM mesh from a solid model. Amongst the various schemes that have been proposed for automatic mesh generation from solid models [WOO84, WÖRD84, CAVE85, SHEP85, YERR85] recursive spatial subdivision schemes have been found to offer an efficient avenue for automatic mesh generation as well as for self-adaptive remeshing and reanalysis because of two intrinsic properties: hierarchical structure and spatial addressability [KELA86]. To understand the importance of these two properties consider the subdivision rule and the associated tree structure illustrated - for a 2-D example - in Figure 1.1. The recursive subdivision rule can be concisely described as follows: (i) the solid domain is "boxed" and the box is decomposed into quadrants (octants in 3-D); (ii) quadrants are classified with respect to the domain: when a quadrant is totally inside or outside of the object, the decomposition ceases; when a quadrant is neither wholly inside nor outside, it is further subdivided into quadrants; (iii) the process continues until some minimal resolution level is reached. The resulting quaternary or octal tree can be thought of as a hierarchical cataloging structure for data describing particular regions of space. The lowest level of the

\* Research Assistant and Assistant Professor, respectively

tree (the "resolution" level) contains the smallest spatial regions and the ordinary finite elements. At higher levels the regions become larger and the finite elements become substructures ("superelements") with associated assembled stiffness matrices. As shown in [KELA86], such a hierarchical organization is ideally suited for self-adaptive mesh refinement and incremental analysis. Furthermore, if the quadrant or octant cells are numbered systematically, then the index of any cell in the hierarchical tree can be quickly computed from its size and position, and conversely the size and position of a cell can be directly derived from its index. This property, called spatial addressability, permits direct access to pertinent geometrical and analytical data during both global mesh generation and localized mesh refinement.

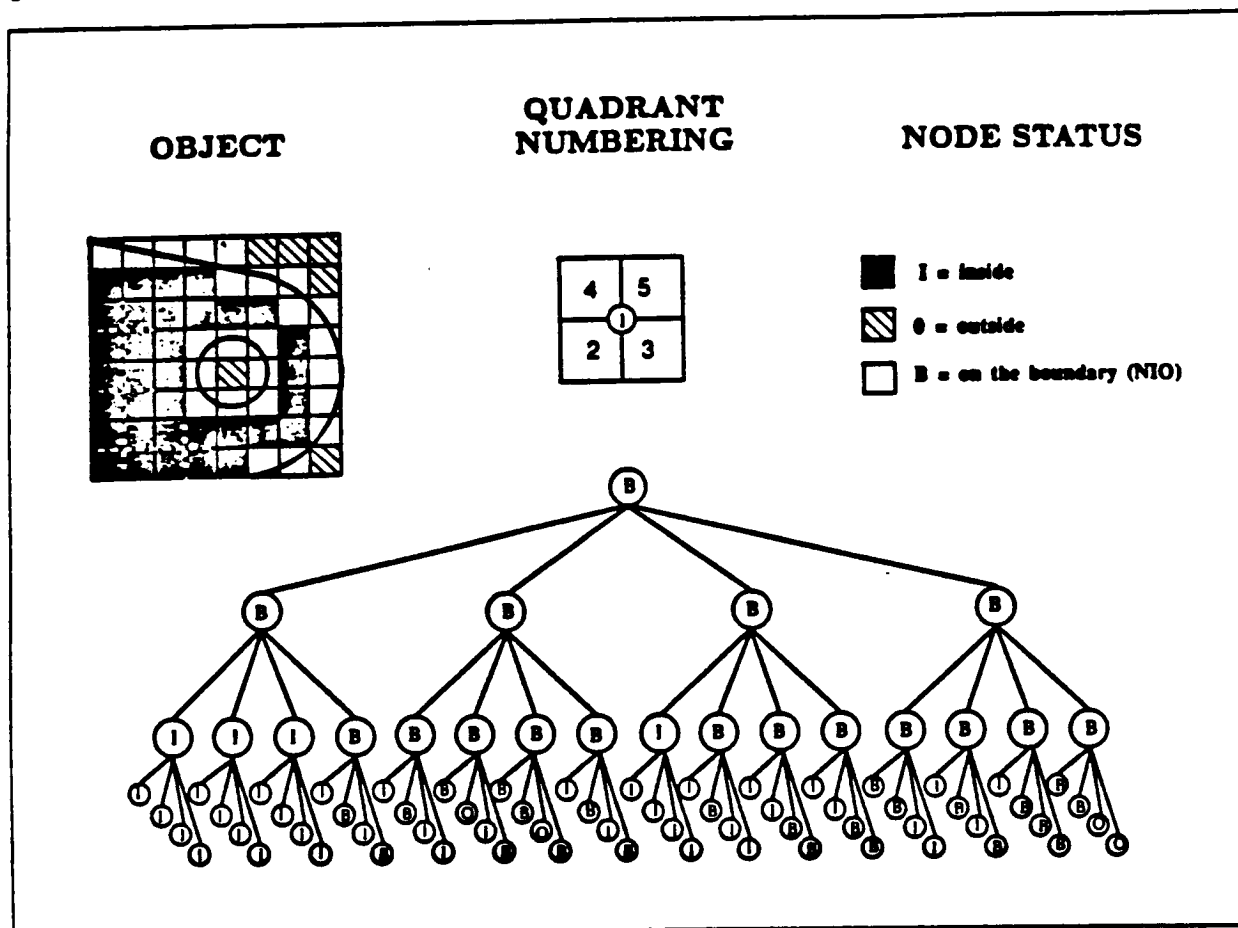


FIGURE 1.1: Quadrant numbering scheme for a 2-D decomposition.

The octree-based scheme presented here is a direct extension of the work in 2-D meshing and adaptive analysis reported in [KELA86]. The scheme involves two stages of meshing. In the first stage the interior of the domain is meshed with a geometrically regular grid of hexahedral elements that includes all the inside octree cells. In the second stage the mesh is extended to the boundary of the domain by inserting finite element topologies in the 3-manifolds formed by the intersection of the octree cell with the bounding surface of the solid. While for 2-D problems the manifolds are cut quadrants that can be easily decomposed into quadrilateral and triangular elements via template matching, the 3-manifolds associated to boundary intersecting octree cells are far more complex structures that cannot be handled by templates alone. Furthermore, to enforce continuity of the field variable and to maintain the geometrical regularity of the interior mesh, the interface between each 3-manifold and neighboring hexahedral elements must be a square. Since tetrahedral elements are essential for the decomposition of the cells intersected by



the boundary, the interface requirement can be satisfied only by introducing pentahedral ("pyramid") elements to provide the transition between triangular and square faces.

In essence, the crucial problem of octree based meshing is to decompose the cells on the boundary into valid element assemblies while maintaining the hierarchical structure, the spatial addressability, and the geometrical regularity associated with the underlying octree grid. In the following sections we discuss the meshing scheme in general terms and then we focus on the decomposition of 3-D boundary cells.

## 2 AN OCTREE BASED MESHING SCHEME

We begin by describing the object in a modelling system based on Constructive Solid Geometry<sup>1</sup> which provides all the basic geometric operators for spatial decomposition and meshing. The idea underlying the octal decomposition scheme is to approximate the object to be meshed with a union of disjoint, variably sized cells (cubes) [JACK80]. However, such an approximation cannot be directly mapped onto a finite element mesh for two crucial reasons: (i) adjoining elements corresponding to octal cells of different size violate the connectivity rules between finite elements, and (ii) the union of orthogonal surfaces that approximates the boundary of the solid contains re-entrant vertices and edges which introduce artificial singularities in the FEM model. We modify the octal decomposition scheme to yield a valid FEM discretization according to the following two-stage strategy.

### First stage of Meshing

The object  $S$  is enclosed in a "box" and the box is recursively decomposed into octal cells which are classified as being "IN"  $S$ , "OUT" of  $S$  or neither in nor outside ("NIO"). For IN cells subdivision ceases and the octant is mapped directly on to a finite element substructure. OUT cells are discarded and NIO cells are further subdivided and classified until a pre-specified level of subdivision (the "resolution" level) is reached and no cell contains more than one connected boundary segment of  $S$ . IN cells at resolution level are mapped onto finite elements. The collection of IN cells forms the interior octree of the solid.

Figure 2.1 shows the interior octree for a solid part - a bracket modelled in the PADL-2 domain [HART83]. The classification procedure used in this stage of meshing is described in [LEE82].

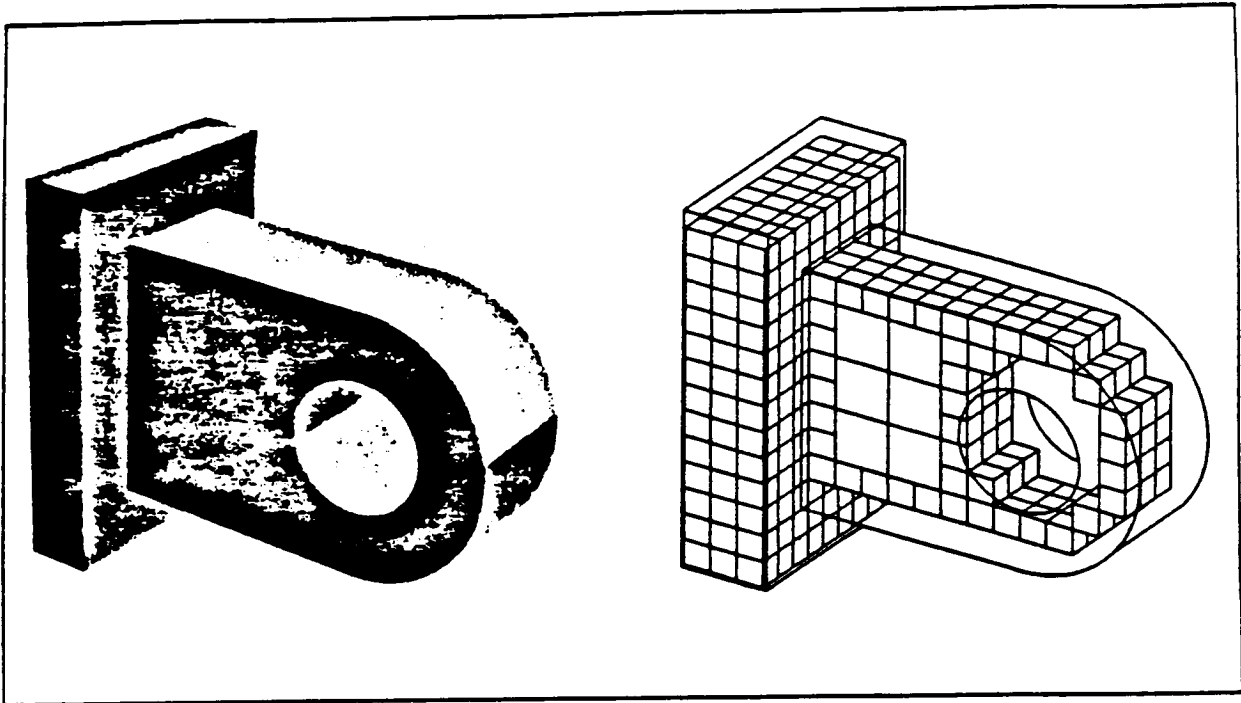
### Second stage of Meshing

During the second stage the interior octree is extended to the boundary of  $S$ ,  $bS$ . This requires associating each of the NIO cells (more precisely the intersection of the solid  $S$  and the octant) to valid finite element topologies. The NIO cells that do not contain edges of  $bS$  are classified as Simple ("SNIO") and their finite element topologies are easily derived through template association.

For the NIO cells that contain edges and vertices of  $bS$ , decomposition through templates is not feasible because of the large number of possible configurations for the edge-cell intersection. These cells, labelled "Complex" NIO (CNIO), are decomposed through a set of element extractors that operate recursively on the topological and geometrical description of the cell. The starting point for recursive element extraction is a valid boundary representation of the polyhedron  $\mathfrak{R}_c$ , formed by the intersection of the CNIO cell and the cutting planes on  $bS$ . These operators are discussed in detail in the following section.

---

<sup>1</sup> Constructive Solid Geometry (CSG) exploits the notion of "adding" and "subtracting" simple building blocks ("primitives") via set-union and set-difference operations.



**FIGURE 2.1 :** A bracket and its interior octree.

The finite element mesh is complete at the end of second stage. The interior of the mesh consists of identical hexahedral elements and substructures associated with IN cells at resolution and higher levels, respectively. Also, the mesh inherits the hierarchical structure and the spatial addressability of the underlying octree decomposition.

As shown in [KELA86, 87], the regularity of the interior mesh together with the spatial addressability of the entire model provides the basis for a very powerful procedure for doing analysis as well as remeshing and reanalysis. Briefly, stiffness matrices are built and stored for all the non-OUT cells in the hierarchical tree (for identical interior elements and substructures they are copied into storage from precomputed values). This is done from the bottom up by assembling son matrices and condensing-out the interior degrees of freedom to build parent matrices at each level. A preliminary study on a 2-D implementation reported in [KELA87] suggests that this substructuring procedure is asymptotically more efficient than direct Gaussian reduction. For adaptive remeshing and reanalysis, spatial addressability allows efficient localized mesh modification. The reanalysis proceeds incrementally: the new stiffness matrices are inserted in the appropriate tree location and are combined with the stiffness of the unmodified elements and substructures.

In conclusion, the strict adherence of the FE mesh to the underlying octree structure offers some unique advantages for the analysis and, as such, is worth preserving. Therefore, stage 2 of the meshing procedure is designed in such a way as to leave intact the interior octree and the spatial addressability of the mesh.

### 2.1 Decomposition of 2-D NIO cells

The approach to 2-D NIO cells decomposition described in [KELA86, 87] is based on deriving finite element topologies exclusively through templates. In this case the number of required templates is small because of the following constraints imposed on the topology of the 2-D NIO cells:

(1) each NIO cell can be traversed by  $bS$  only once, such that

$$\text{NIO} \cap bS = \pi^1 \quad (\text{1-D simply connected polyhedra}); \quad (1)$$

(2) each NIO cell can contain at most one "vertex" node of  $bS$ ;

(3) each edge of the NIO cell can have at most one intersection with  $bS$ .

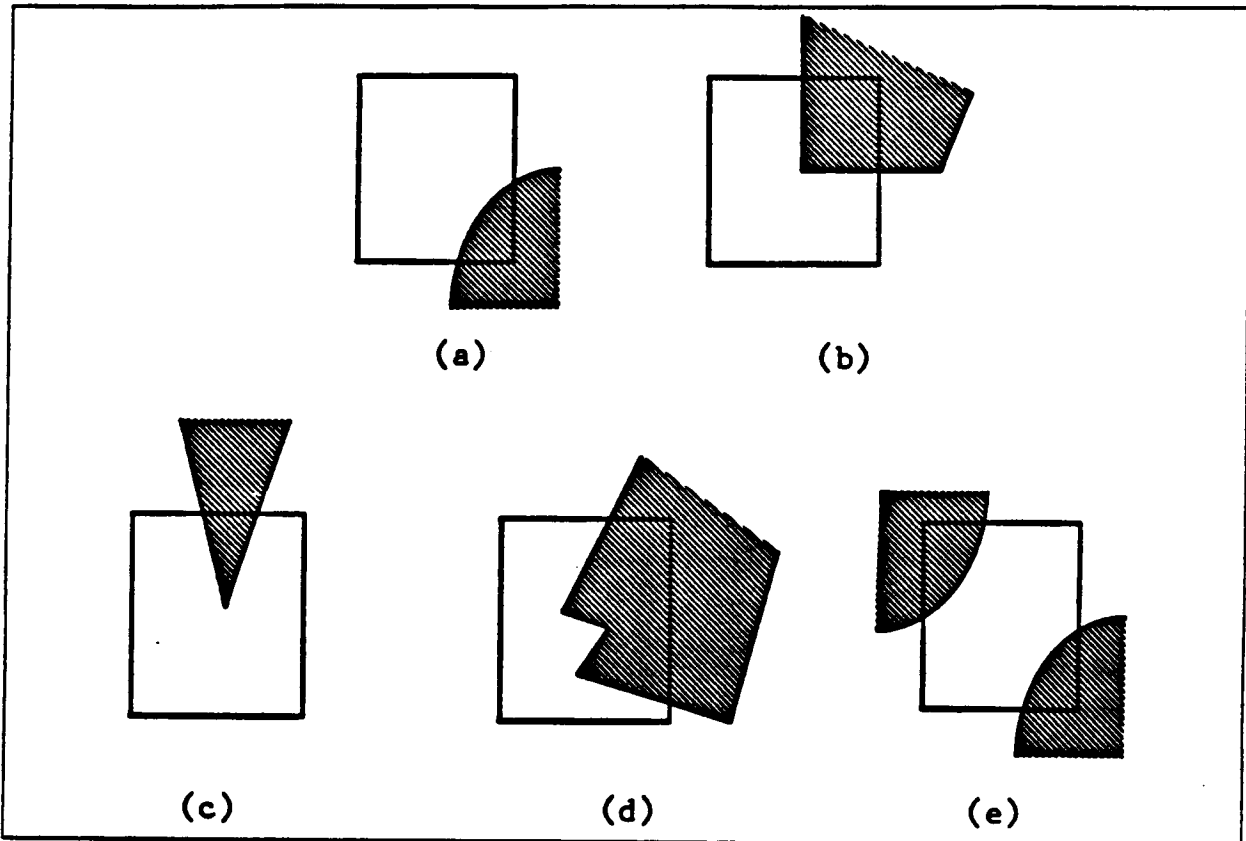


FIGURE 2.2: Valid (a,b) and invalid (c,d,e) 2-D NIO cells.

Valid and invalid 2-D NIO cells are shown in Figure 2.2. As shown in Figure 2.3, the derivation of element topologies, based on the above constraints, is simple. When the NIO cell does not contain any vertex, the element topology may be derived by traversing the boundary of the quadrant and counting the intersections with the object boundary. If the intersections are encountered on alternate edges, a quadrilateral element is mapped on to this cell. If the intersection takes place on adjacent edges, triangular elements are generated by connecting the intersection points to the appropriate cell node classified as IN.

For the case of NIO cells containing a vertex of  $bS$ , the vertex becomes a finite element node and triangles are generated by connecting this node to all the intersection points and the cell nodes that are inside the domain.

This simple decomposition approach – and the topological constraints on which it is based – cannot be extended to 3-D NIO cells because a 3-D  $bS$  contains edges. In this case, unless one imposes overly

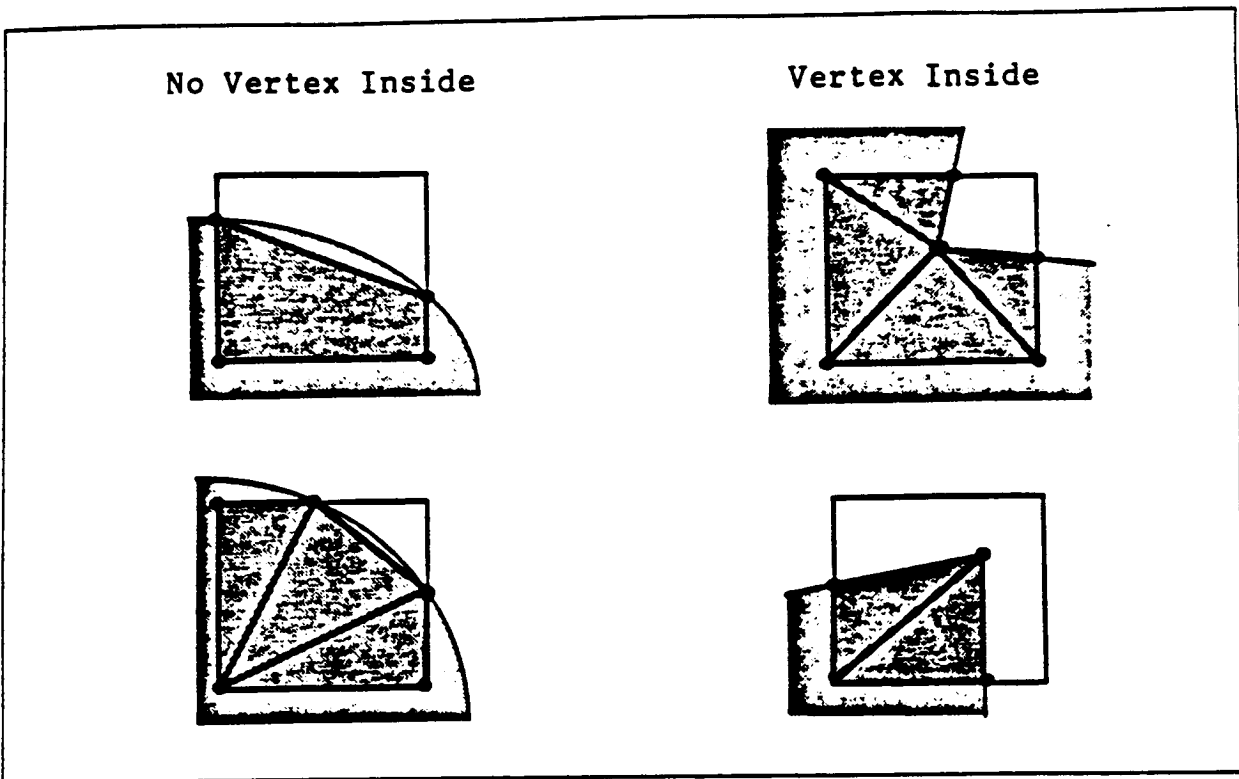


FIGURE 2.3 : Derivation of finite element topologies for 2-D NIO cells.

restrictive conditions on the way a  $bS$  edge is permitted to intersect a NIO cell, decomposition solely via template matching is infeasible.

An important property of the decomposition procedure described above is that each 2-D NIO cell contains all the topological information necessary to associate a finite element structure to the cell such that the continuity of the field variable across the cell boundary is ensured. Thus each 2-D NIO cell can be meshed independently from neighboring cells.

To prove this property we note that the interface between neighboring 2-D elements is an edge (1-D polyhedron). Therefore, to ensure continuity across the interface, the edge shared must be topologically identical, i.e., the edges must have the same bounding vertices (nodes) in both elements. Along the boundary of the NIO cell FE nodes are inserted only at the intersection points and at the cell vertices classified as IN. Because of this condition, any finite element topology introduced in the NIO cell contains only elements that have the correct interface with neighboring elements associated to either IN or NIO cells. Also, the insertion of triangular elements in a NIO cell does not disrupt the regularity of the mesh of square elements associated with the interior quadtree.

For 3-D problems, neighboring elements have a face in common (a 2-D polyhedron) and continuity requires that the shared face have the same set of bounding edges in both elements. In this case, the insertion of nodes on the NIO cell boundary only at the intersection points and at the cell vertices is not sufficient to ensure that 3-D NIO cell meshed independently will satisfy continuity across the interface. We shall expand on this problem later.

### 3 DECOMPOSITION OF 3-D NIO CELLS

The NIO cells are classified as SNIO or CNIO, based on the topological description of the associated polyhedron,  $\mathfrak{R}$ , defined as

$$\mathfrak{R} = \text{NIO} \cap^* S. \quad (2)$$

Here  $\cap^*$  denotes a regularized intersection [REQU85]. If  $\mathfrak{R}$  does not contain any vertex or edge of  $bS$ , the cell is classified as SNIO. In this case,  $\mathfrak{R}_s$ , the polyhedron associated with the SNIO cell, can be simply described as an octal cell in which a number of vertices have been shaved off by a single cutting surface, i.e.,

$$\mathfrak{R}_s = \text{Octant} \oplus H_1. \quad (3)$$

where octant indicates an octal cell at resolution level,  $\oplus$  a regularized boolean operation and  $H_1$  is the cutting surface. Figure 3.1 shows a typical SNIO cell and its corresponding location in the solid part. Note that the associated polyhedron  $\mathfrak{R}$ , is a cube with four corners shaved off.

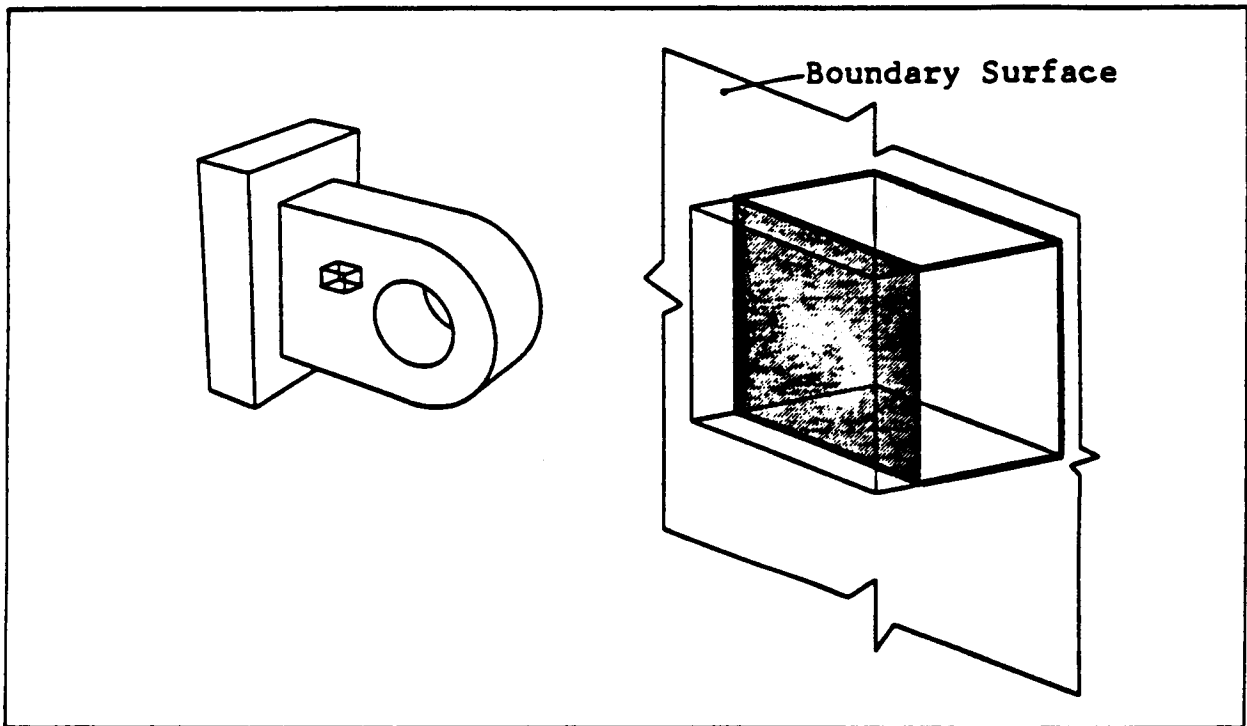


FIGURE 3.1: A SNIO cell and its location on the solid part.

If  $\mathfrak{R}$  contains vertices or edges of  $bS$ , the cell is classified as CNIO. Since a vertex is always defined by three or more intersecting surfaces and an edge by exactly two, the associated polyhedron can be represented as

$$\mathfrak{R}_c = \text{Octant} \oplus H_1 \oplus H_2 \dots \oplus H_n \quad (4)$$

where  $H_1, H_2, \dots, H_n$  are cutting surfaces. Figure 3.2 shows a CNIO cell which contains three edges and a vertex of  $bS$ .

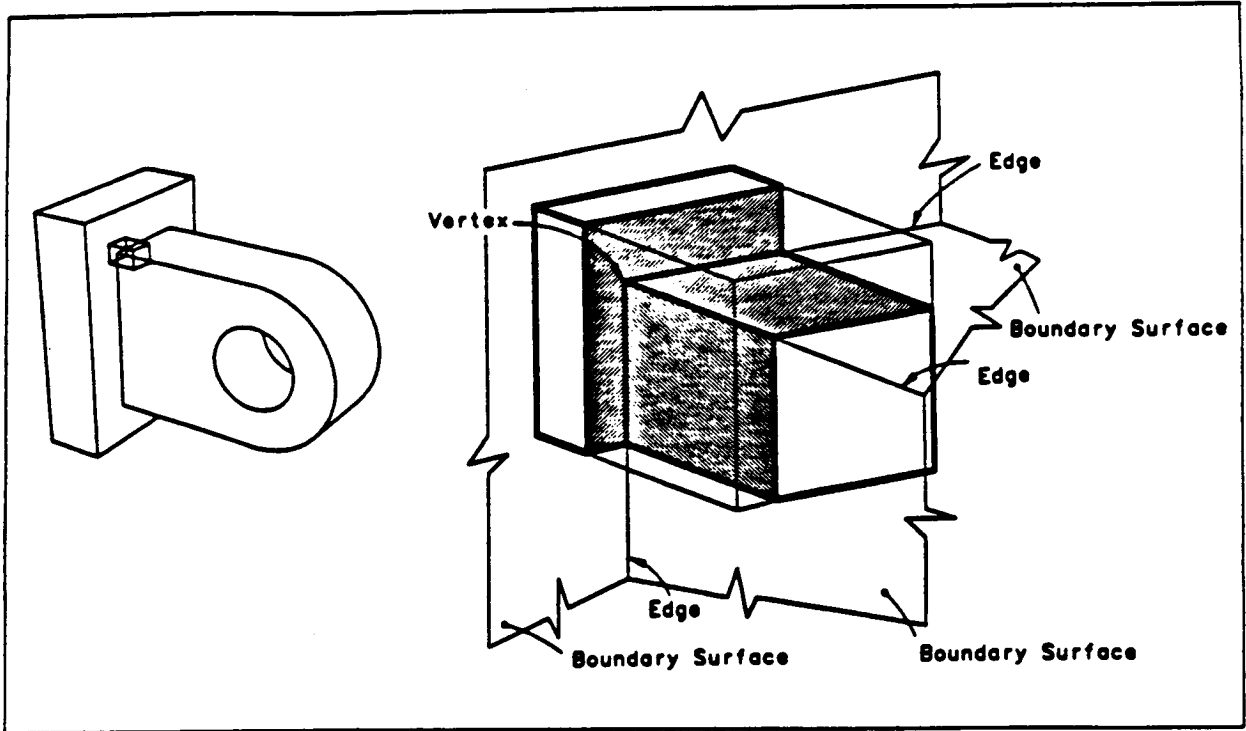


FIGURE 3.2 : A CNIO cell and its location on the solid part.

### 3.1 Decomposition of SNIO cells

Since only seven different FE topologies are required for all possible SNIO cell configurations, the cell is decomposed by first selecting the appropriate template and then mapping the mesh from the template onto  $\mathcal{R}_c$ . The template is chosen by counting the number of vertices shaved off by the cutting surface.

Figure 3.3 shows four cases of SNIO cells and the associated template derived meshes. The remaining three cases of possible SNIO cells, not illustrated in this figure, are the complements of (a), (b) and (c). The templates shown are based on linear hexahedral, pentahedral, wedge and tetrahedral elements<sup>2</sup>. The quadrilateral faces of the pentahedral and wedge elements mapped on to the square sides of the NIO cell ensure continuity along the interface between the NIO cell and the interior mesh. The union of the finite elements represents a planar approximation of the actual geometry with all the non-planar segments of  $bS$  intersected by the NIO cell replaced by triangular and quadrilateral bilinear patches. Finally, we note that most of the NIO cells are classified as SNIO cells and their decomposition through template matching is computationally inexpensive.

### 3.2 Decomposition of CNIO cells

The element "extractors", shown in Figure 3.4, are a modified version of the operators presented in [WOO84]. The operators  $\tau_1$  and  $\tau_2$  produce tetrahedral elements while  $\tau_3$  extracts pentahedra based on the square faces of  $\mathcal{R}_c$  that correspond to the original faces of CNIO cell. These operators work as follows.

<sup>2</sup> Linear pentahedral, wedge and tetrahedral elements can be generated by collapsing a standard isoparametric brick element [BATH82].

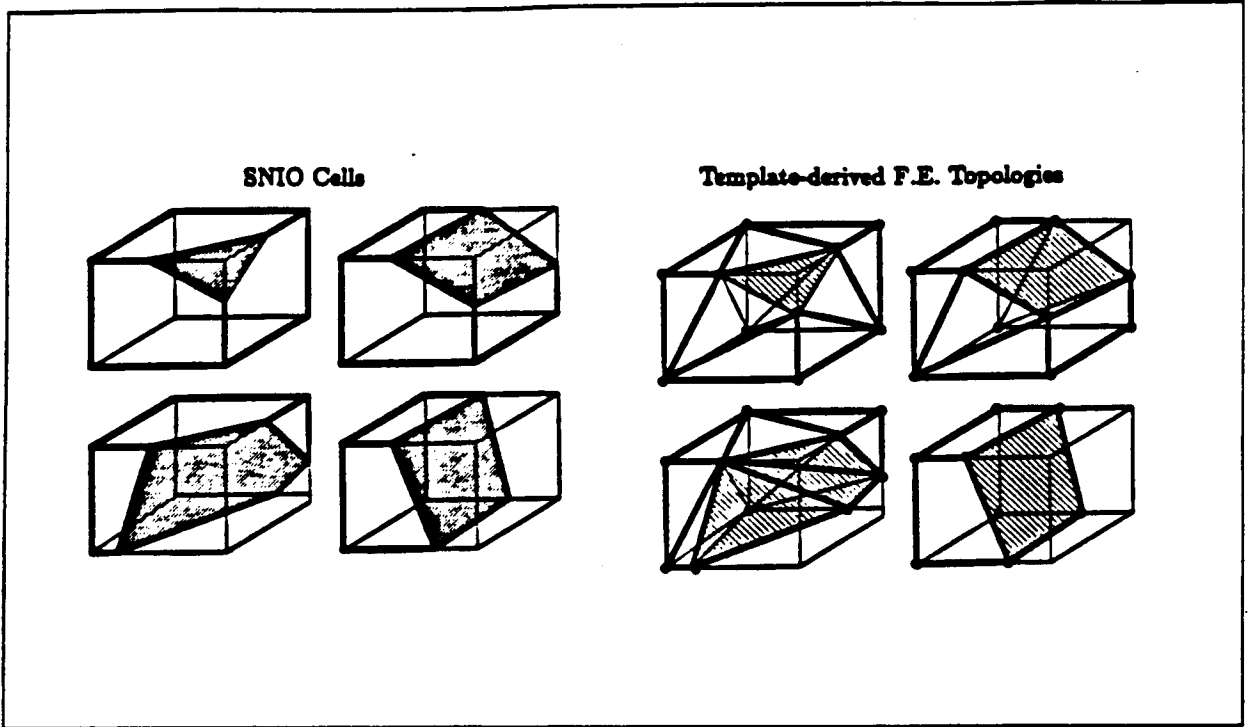


FIGURE 3.3: SNIO cells and associated templates.

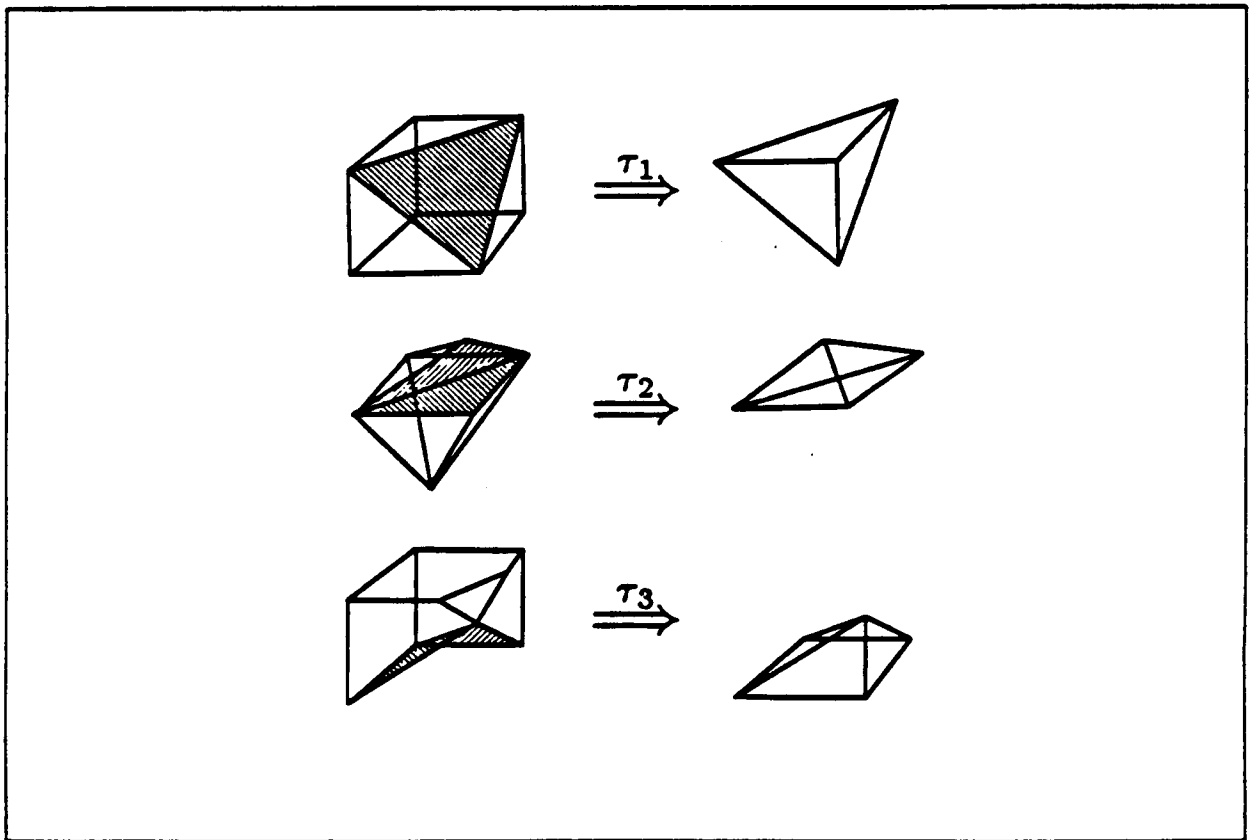


FIGURE 3.4: Element extractors for CNIO cells.

- 1)  $\tau_1$  scans the boundary representation of  $\mathcal{R}_c$  searching for convex trivalent vertices. When such a vertex is found  $\tau_1$  extracts a tetrahedron by introducing a single cut in the domain (this corresponds to the “slicing” operation in [WOO84]).
- 2)  $\tau_2$  is applied when all the convex trivalent vertices are exhausted. This operator uses a convex edge to extract a tetrahedron by introducing two cuts in the domain – referred to as “digging” into the domain in [WOO84].
- 3)  $\tau_3$  looks for faces of  $\mathcal{R}_c$  that correspond to original cell faces and extracts a pentahedron by introducing multiple cuts that vary according to the location of the apex vertex. The choice of the apex vertex is based on interference considerations. The operator  $\tau_3$  is applied before  $\tau_1$  and  $\tau_2$  in order to preserve all the original cell faces contained in  $\mathcal{R}_c$ .

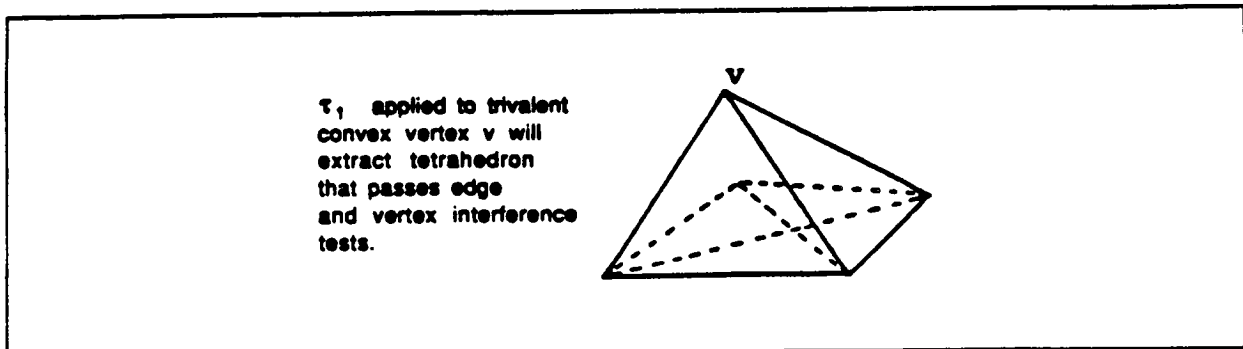


FIGURE 3.5 : Pathological case for the interference test in [WOO84].

Before each extraction the validity of the candidate tetrahedron or pentahedron is checked through a series of tests. As in [WOO84], the vertices and edges of  $\mathcal{R}_c$  are checked for interference with the faces of the candidate element. More precisely, the interference test ensures that: (i) no vertex of  $\mathcal{R}_c$  lies on any of the faces of the candidate element, and (ii) no edge of  $\mathcal{R}_c$  intersects any of the faces of the candidate element. This test is not enough, however, to ensure the validity of the element – see the exemplary pathological case illustrated in Figure 3.5. To remove the ambiguity, an additional check is performed by classifying the centroid of each face of the candidate element against  $\mathcal{R}_c$ . If all centroids are classified as ON or IN the element is valid.

The implementation of the element extractors and the geometric checks described above requires a point-membership classifier (PMC) — a function that returns the classification of a point  $p$  with respect to the polyhedron  $\mathcal{R}_c$  as

$$PMC(p, \mathcal{R}_c) = (In, On, Out). \quad (5)$$

The PMC developed for the present work operates on planar polyhedra and is based on ray casting algorithms [KALA82].

The boundary representation (BRep) structure used for maintaining and updating the topology of  $\mathcal{R}_c$  has two graphs imbedded in it: (i) Face  $\rightarrow$  Edge  $\rightarrow$  Vertex and (ii) Vertex  $\rightarrow$  Edge  $\rightarrow$  Face. This double structure provides greater flexibility while manipulating the BRep for the polyhedron, because it reduces the number of scans required to retrieve the necessary information about the boundary. The PMC permits the classification of the edges and the vertices in the BRep as convex or concave. This piece of information is crucial for element extraction and must be updated after each element removal.



All the operators used for element extraction preserve the differential form of the Euler-Poincare formula [BAUM72]

$$\nabla F - \nabla E + \nabla V = 0 \quad (6)$$

where  $F$  is the number of faces,  $E$  is the number of edges and  $V$  the number of vertices in the polyhedron. Provided that the initial polyhedron is valid, the satisfaction of equation (6) ensures that the validity is maintained at each stage of the extraction.

Figure 3.6 shows different stages of the element extraction on the  $\mathcal{R}_c$  associated with the CNIO cell in Fig. 3.2. The operator  $\tau_3$  is applied to extract a pentahedral element whose base is the original cell face. This is followed by the recursive application of the operator  $\tau_1$  until all trivalent convex vertices are exhausted. Operator  $\tau_2$  takes over until one or more trivalent convex vertices become available and  $\tau_1$  can be applied again. This progressively reduces the domain to a single tetrahedron.

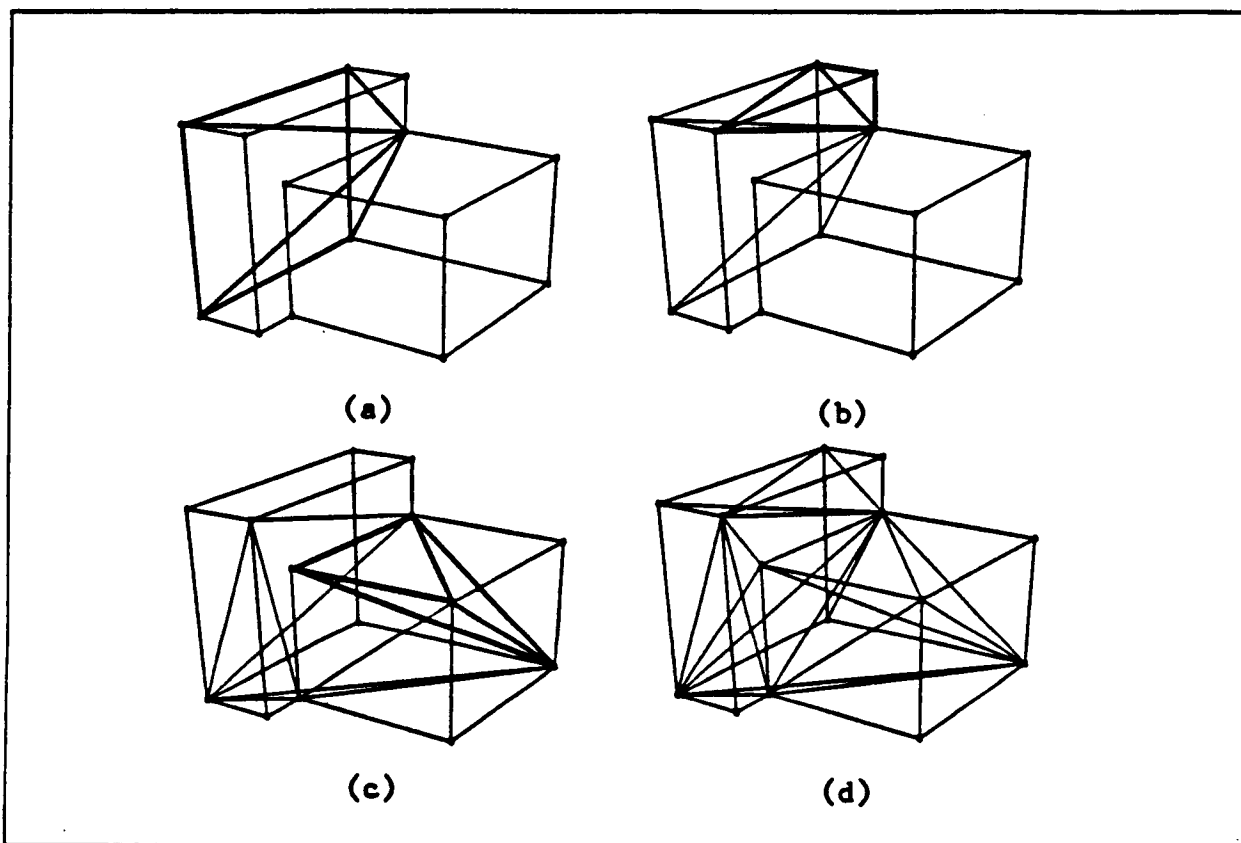


FIGURE 3.6 : Element extraction on a CNIO cell:  $\tau_3$  extraction of a pentahedron (a), a tetrahedron extracted via  $\tau_1$  (b) and  $\tau_2$  (c), the complete element topology.

The computational cost for deriving the BRep of  $\mathcal{R}_c$  and decomposing it through element extraction is considerably higher than that associated with SNIO cell decomposition. We note, however, that the number of CNIO cells is relatively small.

#### 4 BOUNDARY EVALUATION FOR CNIO CELLS

In the preceding section we indicated that a boundary representation of the polyhedron  $\mathcal{R}_c$  is needed for the element extraction. The standard approach to derive the boundary for solids described in a CSG environment is to intersect the faces of all the primitives that constitute the CSG definition of the object and classify the resulting edges against the combinatorial tree (this operation is called boundary merging [REQU85]). Since this merging process involves all the primitives, boundary evaluation for a CSG described solid is in general a computationally expensive procedure<sup>3</sup>.

The polyhedron  $\mathcal{R}_c$  is formed by the intersection of the CNIO octant and the original solid, i.e.,

$$\mathcal{R}_c = \text{CNIO} \cap S \quad (7)$$

and therefore its boundary evaluation appears to require the boundary merging of the complete solid  $S$  and the CNIO cell. We note, however, that generally the cell under consideration is spatially localized, i.e., each CNIO cell intersects only a limited number of primitives. In this case the boundary of  $\mathcal{R}_c$  can be obtained by merging only the boundary of the primitives which interfere with the cell.

The primitive incidence information required to generate the necessary set of tentative edges is produced the following way. At the last level of the octree decomposition every NIO cell is classified against each of the primitives in the CSG tree. When the cell is classified "ON" a primitive, the primitive is added to the incidence information carried with the cell. Hence, at the end of the classification, each NIO cell points to the subset of the CSG primitives that "interact" spatially with the cell. As indicated in [TILO81], primitive incidence leads to "pruning" of the CSG tree and, consequently, reduces the computational cost of boundary merging.

The boundary of  $\mathcal{R}_c$  is necessarily contained in the boundary of the CNIO cell and of the primitives incident upon the cell. Therefore the tentative set of edges generated merging the incident primitives and the cell suffices for building the boundary representation of  $\mathcal{R}_c$ . Since several CNIO cells may be incident upon a small number of primitives, exploiting primitive incidence may save considerable computational time during mesh generation. Fig. 4.1 illustrates tree pruning for the CNIO cell shown in Fig. 3.2.

#### 5 DISCUSSION

As indicated in the previous sections, by adhering to the underlying octal cell decomposition the mesh acquires hierarchical structure, spatial addressability and interior geometrical regularity. We consider these three properties central to the automation of finite element analysis. Therefore our approach to octree based automatic meshing is focussed on preserving a tight correspondence between the finite element and the cell structure. In particular, this requires embedding a finite element topology in the NIO cells without disrupting the global structure and addressability of the mesh as well as the regularity of the interior octree.

We treat NIO cells in a selective way: element extractors are used only for those - relatively few - cases for which template matching is infeasible. Template controlled decomposition is appealing because it is

<sup>3</sup> The asymptotic time complexity for boundary evaluation ranges between  $O(n^3 \log n)$  and  $O(n^4)$ , where  $n$  is the number of primitives [TILO81].

cant these same 12 concepts be used to do better files in better than  $n^3 \log n$

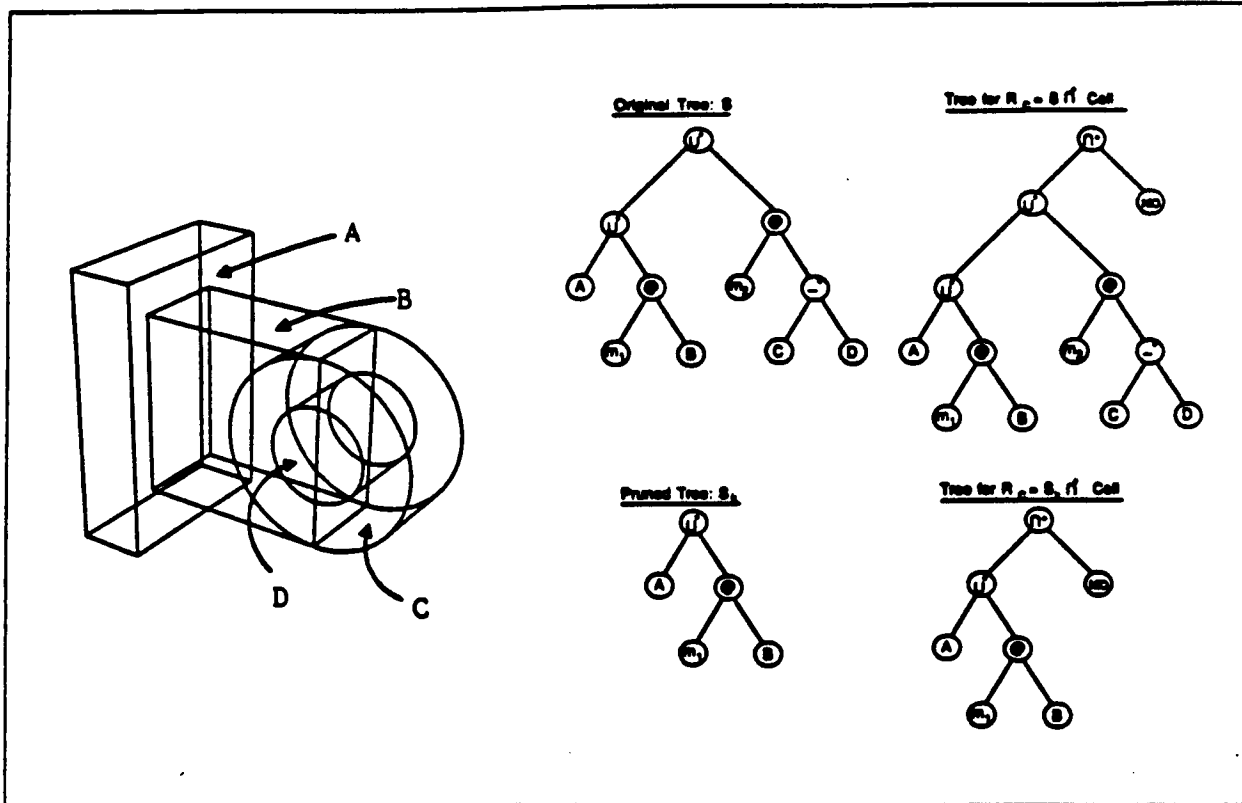


FIGURE 4.1 : Tree pruning for a CNIO cell. (A, B, C, D are primitives;  $m_1, m_2$  are motions;  $\cup, \cap, -$  represent regularized Boolean operations;  $\odot$  indicates the application of a motion.

computationally inexpensive and allows a good degree of control on the elements which are inserted in the NIO cell. Conversely, element extractors require building and maintaining a sophisticated data structure and provide a very limited amount of control on the mesh. The exclusive use of mapping or element extraction on all the boundary cells – as proposed in [YERR84] and [YERR85], respectively – is either too limited for handling complex geometries (the former) or computationally too demanding for practical implementation (the latter). The selective use of the two algorithms based on the preliminary NIO cell classification described in this paper results in a flexible approach designed to exploit the advantages of both types of decomposition.

The algorithms discussed here are currently being implemented in an experimental code built on the PADL-2 modelling system. PADL-2 provides the utilities for modelling the solid and extracting the octree. Also, the geometric routines contained in the modeller are used extensively to perform the operations required for the SNIO/CNIO cell classification, the SNIO template mapping and the derivation of the CNIO boundary representation. In particular, boundary evaluation is done by first pruning the CSG tree and then using the PADL-2 incremental boundary evaluator [HART85]. The CNIO cell decomposition is carried out in an independent modelling environment based on the BRep structure described in Section 3. The implementation of the element extractors is built on a specialized point-membership classifier that operates on planar polyhedra.

To complete the implementation of our meshing algorithm we have to resolve some specific issues related to interfacing CNIO cells with IN and SNIO cells. Our plans are the following. The CNIO/IN interface is generally taken care of by using pentahedral elements. Whenever that is not possible, the propagation of triangular faces is contained within the adjacent IN cell with a two-step procedure: (i) decompose

the IN cell into 6 identical pyramids with the apex at the cell centroid and (ii) split the pyramid on the interface into two tetrahedral elements. The interface between SNIO/CNIO cells can be modelled by either embedding the edges on the SNIO face into the BRep associated with the CNIO cell, or by modifying locally the SNIO mapped mesh to reflect the entities on the CNIO face. Note that the task of identifying the two cells sharing a face is considerably simplified because of the spatial addressability of the cell (and element) structure.

In conclusion, the approach presented here resolves efficiently the geometrical and topological issues related to octree based automatic meshing and - in analogy with the quadtree structures described in [KELA86] - opens a promising avenue for self-adaptive analysis.

## ACKNOWLEDGEMENTS

Herbert Voelcker of Cornell University, Ajay Kela of General Electric Company, and Aristides Requicha of the University of Southern California contributed to this research. The figures were produced on equipment donated by Tektronix, Inc. Other Industrial Associate companies of the Production Automation Project provided sustaining support. The findings and opinions expressed here do not reflect the views of the sponsors.

## REFERENCES

- [BATH82] K. J. Bathe, *Finite Element Procedures in Engineering Analysis*. New Jersey: Prentice-Hall, 1982.
- [BAUM72] B. G. Baumgart, "Winged edge polyhedron representation", STAN-CS-320, Stanford Artificial Intelligence Project, Stanford University, October 1972.
- [CAVE85] J. C. Cavendish, D. A. Field and W. H. Frey, "An approach to automatic three-dimensional finite element mesh generation", *International Journal for Numerical Methods in Engineering*, vol. 21, pp. 329-347, 1985.
- [HART83] E. E. Hartquist, "Public PADL-2", *IEEE Computer Graphics and Applications*, vol. 3, no. 7, pp. 30-31, October 1983.
- [HART85] E. E. Hartquist, "PP2/2.N Boundary Evaluator", Incremental Boundary Evaluator Doc. No. 4, Production Automation Project, University of Rochester, December 1985.
- [JACK80] C. L. Jackins and S. L. Tanimoto, "Oct-trees and their use in representing three-dimensional objects", *Computer Graphics & Image Processing*, vol. 4, no. 3, pp. 249-270, November 1980.
- [KALA82] Y. E. Kalay, "Determining the spatial containment of a point in general polyhedra", *Computer Graphics & Image Processing*, vol. 19, no. 4, pp. 303-334, August 1982.
- [KELA86] A. Kela, R. Perucchio and H. B. Voelcker, "Toward automatic finite element analysis", *ASME Computers in Mechanical Engineering*, vol. 5, no. 1, pp. 57-71, July 1986.

- [KELA87] A. Kela, "Automatic finite element mesh generation and self-adaptive incremental analysis through solid modeling", Ph.D. Dissertation, Dept. of Mechanical Engineering, University of Rochester, January 1987.
- [LEE82] Y. T. Lee and A. A. G. Requicha, "Algorithms for computing the volume and other integral properties of solids: Part II - A family of algorithms based on representation conversion and cellular approximation", *Communications of the ACM*, vol. 25, no. 9, pp. 642-650, September 1982.
- [REQU85] A. A. G. Requicha and H. B. Voelcker, "Boolean operations in solid modelling: Boundary evaluation and merging algorithms", *Proceedings of the IEEE*, vol. 3, no. 1, pp. 30-44, January 1985.
- [SHEP85] M. S. Shephard, "Finite element modeling within an integrated geometric modeling environment: Part I - Mesh generation", *Engineering with Computers*, vol. 1, pp. 61-71, 1985.
- [TILO81] R. B. Tilove, "Line/polygon classification: A study of the complexity of geometric computation", *IEEE Computer Graphics and Applications*, vol. 1, no. 2, pp. 75-88, April 1981.
- [WOO84] T. C. Woo and T. Thomsma, "An algorithm for generating solid elements in objects with holes", *Computers & Structures*, vol. 18, no. 2, pp. 333-342, 1984.
- [WÖRD84] B. Wördenweber, "Finite-element analysis for the naive user", in M. S. Pickett and J. W. Boyse, Eds., *Solid Modelling by Computers*. New York: Plenum Press, 1984, pp. 81-102.
- [YERR84] M. A. Yerry and M. S. Shephard, "Automatic three-dimensional mesh generation by the modified-octree technique", *International Journal for Numerical Methods in Engineering*, vol. 20, pp. 1965-1990, 1984.
- [YERR85] M. A. Yerry and M. S. Shephard, "Trends in engineering software and hardware - Automatic mesh generation for three-dimensional solids", *Computers & Structures*, vol. 20, no. 1-3, pp. 31-39, 1985.

N88 - 19119

5-8-61  
125794  
348

Production Automation Project  
College of Engineering & Applied Science  
The University of Rochester  
Rochester, New York 14627

**A HIERARCHICAL STRUCTURE FOR AUTOMATIC  
MESHING AND ADAPTIVE FEM ANALYSIS**

by

Ajay Kela, Mukul Saxena and Renato Perucchio

(November 1986)

submitted for publication in  
a special issue of  
*Engineering Computations*

The work described in this paper was supported by the National Science Foundation under Grants ECS-8104646 and DMC-8403882 and by companies in the P.A.P.'s Industrial Associates Program. The findings and opinions expressed here are those of the authors and do not necessarily reflect the views of the various sponsors.

# A HIERARCHICAL STRUCTURE FOR AUTOMATIC MESHING AND ADAPTIVE FEM ANALYSIS

Ajay Kela \*

Corporate Research and Development  
General Electric Company  
Schenectady, N.Y. 12301

&

Mukul Saxena and Renato Perucchio \*\*  
Production Automation Project  
and Department of Mechanical Engineering  
University of Rochester  
Rochester, N.Y. 14627, U.S.A.

## SUMMARY

*This paper deals initially with a new algorithm for generating automatically, from solid models of mechanical parts, finite element meshes that are organized as spatially addressable quaternary trees (for 2-D work) or octal trees (for 3-D work). Because such meshes are inherently hierarchical as well as spatially addressable, they permit efficient substructuring techniques to be used for both global analysis and incremental re-meshing and re-analysis. The paper summarizes the global and incremental techniques, and presents some results from an experimental closed loop 2-D system in which meshing, analysis, error evaluation, and re-meshing and re-analysis are done automatically and adaptively. The paper concludes with a progress report on a 3-D implementation.*

---

\* former Research Assistant, Production Automation Project  
\*\* Research Assistant and Director, respectively

## 1 INTRODUCTION

Interactive computer graphics has reduced the cost of using the Finite Element Method (FEM) to analyze mechanical parts and structures [PERU82]. However, interactive mesh generation still requires the guidance and ingenuity of an expert analyst to produce a valid FEM model, to interpret computed results and to modify the model when results are questionable. Thus analysing a fixed design is usually an iterative process; moreover as design itself is iterative, the current use of the FEM requires continued human guidance within a doubly iterative process. It is obvious that automatic mesh generation, followed by adaptive mesh refinement would dramatically reduce the cost of the design process. Two newly available tools – solid modelling systems [REQU83] and algorithms for a posteriori error analysis [BABU78, PEAN79, KELL83, GAGO83] – make this goal reachable.

< Figure 1 >

Figure 1 illustrates the architecture of an automatic analysis system. The user defines an initial geometrical domain in the Solid Modelling System (SMS) together with such attributes as boundary conditions, loads, material properties, and analysis related parameters. The mesh generator produces a discretized model – the FEM mesh – from the geometric definition and the attribute specification (attributes may determine, for example, the locations of some nodes). The FEM analysis processor computes primary and secondary field variables (in general, the displacements vector at nodal points and the stress tensor within the elements) from the initial FEM model. The error evaluator compares global error estimates derived from the analysis output with pre-specified error-tolerances to either accept the results or request a new analysis based on a modified mesh. In the feed-back loop, the analysis control process indicates regions of refinement in the current model for the next cycle of mesh generation and analysis. In case of reanalysis, mesh generation and mesh analysis proceed through localized mesh refinement and incremental re-analysis, i.e. the use of previous unaltered regions of the mesh as well as intermediate analysis computations to derive new results. This approach to automatic FEM analysis is embodied in an experimental 2-D system whose underlying principles are explained below. All meshes and analytical results that appear in later sections were produced with the experimental system.



The next section opens with a discussion of automatic mesh generation focussed mainly on a particular approach – hierarchical grids – that fosters spatial addressability (an important property explained below). Later sections discuss algorithms for (1) generating hierarchical grid-based meshes, then (2) analyzing such meshes, (3) refining and re-analyzing, and finally (4) extending meshing and analysis to 3-D work. The paper concludes with a short discussion of the strengths and weaknesses of the approach.

## 2 AUTOMATIC MESH GENERATION

Most “automatic” meshing facilities in contemporary CAD systems operate from wireframe descriptions of objects, via mapping algorithms. The user must partition the domain, which is represented by a collection of edges, into a set of topologically simple subdomains in which meshes can be generated automatically. This approach is unsuitable for a fully automatic meshing procedure because it depends on human judgement both to guide meshing per se and to resolve ambiguities in the wireframe representation.

Genuinely automatic mesh generation must start from an unambiguous representation of the object to be analyzed, and thus some form of solid modelling system (SMS) is a primary utility. Nearly all current SMS's are based internally on either a Constructive Solid Geometry (CSG) Representation or a Boundary Representation, or both [REQU83]. CSG exploits the notion of “adding” and “subtracting” (via set-union and set-difference operations) simple solid building blocks. Boundary schemes describe solids indirectly, via sets of faces which are represented by sets of edges that bound finite regions of surfaces.

The various schemes that have been proposed for automatic mesh generation may be catalogued for present purposes into three families: triangulation, element extraction and recursive spatial subdivision (quadtree and octree) schemes. We shall discuss the first two family briefly and then focus on the third.

Originally limited to 2-D problems, triangulation algorithms require some level of interactive user control to generate irregular assemblies of triangular elements [SUHA72]. Recently, however, Cavendish and co-workers [CAVE85] have developed a two-stage approach to automatic triangulation of solid domains. In the Cavendish method, points are injected into the domain, and then a solid triangulation is induced in which the points

become nodes of tetrahedral elements. The main working tool of the second-stage triangulation is a Delaunay algorithm that generates valid meshes of tetrahedral elements within convex hulls of node points. Automatic algorithms are still being sought for (a) inserting points in the procedure's first stage, (b) removing elements that are generated outside the domain, and (c) representing the domain's boundary correctly.

Meshing schemes based on element extraction also result in decomposing the domain into a irregular collection of tetrahedral elements [WOO84, WÖRD84]. Elements are extracted by recursively applying a set of operators that work on the topological and geometrical description of the domain. The tetrahedral meshes that result are coarse and usually contain distorted elements that must be refined for analytical use. Also, existing operators for element extraction are not robust as required for a truly automatic implementation.

In both of these family of approaches, mesh refinement is done by splitting existing elements. Because refinement is driven from a FEM mesh rather than from the original solid model, refinement does not improve the geometric approximation of the original solid. Also, the meshes are not spatially addressable.

The idea underlying recursive spatial subdivision schemes is to approximate the object to be meshed with a union of disjoint, variably sized rectangles (in 2-D) or blocks (in 3-D); these are generated by subdividing recursively a spatial region enclosing the object, rather than the object itself. Figure 2.1 provides a 2-D example. The object – a bracket with a hole – is “boxed” to establish a convenient minimal spatial region, and then the box is decomposed into quadrants. When a quadrant can be classified as wholly inside or outside of the object, subdivision ceases; when a quadrant cannot be so classified, it is subdivided into quadrants and this process continues until some minimal resolution level is reached. (In 3-D, the decomposition proceeds by octants.) Approximations produced in this manner can be represented by logical trees whose nodes have four or eight sons (see Figure 2), hence the popular names quadtree and octree [JACK80].

< Figure 2 >

Inside cells of a spatial decomposition can be converted easily into FEM elements or substructures, but Boundary cells require further processing to produce valid elements that approximate closely the object's boundary.

Recursive spatial decompositions have two intrinsic properties – hierarchical structure and spatial addressability – that are central to the mesh refinement and incremental

analysis techniques described later. These intrinsic properties, briefly presented here, are fully discussed in [KELA87].

The tree structure in Figure 2 can be regarded as an organizing or cataloging structure for data describing particular regions of space. At the lowest level of the tree one finds the smallest spatial regions and simplest finite elements. As one ascends the tree, the regions become larger (encompassing multiples of four or eight elemental regions) and the finite elements become super-elements with associated ("assembled") stiffness matrices, collected constraints, and so forth. As we shall see later, such an organization is ideally suited to mesh refinement by subdivision and incremental mesh analysis.

The diagram in Figure 2 suggests the classical approach to accessing the data structure associated with the tree: represent a tree with a linked-list in which nodes are addressed indirectly through downward pointers to sons and perhaps lateral pointers to siblings. Thus one accesses data by following pointers downward from the root of the tree. Alternatively, a recursive spatial decomposition can be viewed as a directly addressable hierarchical grid in which the number of cells in each linear dimension is an integer power of two. The key notion here is a systematic scheme for numbering all possible nodes of the underlying tree. In Figure 2, "1" represents the enclosing box, "2" - "5" represent specific quadrants of "1", "6" - "9" would represent quadrants of "2", and so on. Thus to access the spatial data for a particular node in the underlying tree, one merely calculates an array index through a simple formula and follows the single pointer stored there. This is usually much faster than the pointer-following method noted above, but it carries a storage penalty [KELA87].

Suppose finally that we know the geometric size and spatial position of the "1" cell (the overall box) in Figure 2. We can compute quickly the index of any cell in the hierarchy from its size and position, and conversely from an index we can compute quickly the size and position of the associated spatial cell. We have already seen that cell indices allow access through a single pointer to data associated with the cell, and thus we can associate, without searching, spatial regions with stored data, and stored data with spatial regions. This is what is meant by spatial addressability.

### 3 AN AUTOMATIC MESHING PROCEDURE

The procedure described below produces a spatially addressable FEM mesh embedded in the lowest level of a hierarchical grid. Higher levels of the grid are used during construction of the mesh and, as explained later, when the mesh is analyzed, refined, and incrementally re-analyzed. The procedure starts with a representation in a Solid Modelling System of the object to be meshed, and operates in two stages. The first stage meshes the interior of the object by spatial subdivision, and the second extends the mesh to the object's boundary. Each stage is described and illustrated below.<sup>1</sup>

We wish to note that the use of quadtree/octree methods for automatic mesh generation was pioneered by Shephard & Yerry [YERR83,YERR84]. Our work is similar to theirs, but the differences are real and important.

STAGE 1: Interior Meshing See Figure 3. The object  $S$  is enclosed in a box which is recursively subdivided into a grid whose smallest cell size determines the element size (or element density) of the initial FEM mesh; this minimal size is determined by subdividing cells until no cell contains more than one connected boundary segment of  $S$ . As the subdivision proceeds the cells are classified as being In  $S$  ("IN"), Out of  $S$  ("OUT"), or Neither In nor Out ("NIO"). Cells classified as IN at higher levels in the hierarchy are subdivided to the final grid size without further classification. The collection of IN cells constitutes the interior mesh of  $S$ .

< Figure 3 >

The main computational utility used for cell classification is the modified cell classification procedure

$$\text{ModClassCell}(\text{cell}, \text{solid}) = (\text{"IN"}, \text{"OUT"}, \text{"?"}),$$

which is described fully in [LEE82].

#### STAGE 2: Boundary-Region Meshing

The task here is to fill the region between the boundary of the interior mesh (denoted  $bIS$  - see Figure 4a) and the boundary  $bS$  of the solid  $S$ . Observe that

$$bS \subset (\cup \text{NIO cells}) \cup bIS$$

<sup>1</sup> The discussion here and in the next several sections is cast in 2-D; 3-D extensions are discussed in section 6.

Thus bS usually is contained in the NIO cells and special element-building operations are required, but sometimes segments of bS coincide with bIS (as at the top of Figure 4a) and no special processing is needed. Thus we can mesh the inter-boundary region by visiting each NIO cell and creating elements that link the bS segment passing through it to the interior of the solid.

< Figure 4 >

There are three main technical issues involved in this process: devising a systematic way to insure that all NIO cells are visited, creating nodes on bS, and associating bS-nodes with existing bIS-nodes to form valid elements. We shall discuss each of these issues briefly.

All NIO cells can be visited by an exhaustive scan of the lowest-level grid, or by tree traversal, or by traversing bS. Since no single approach seems to offer substantial advantages over the others, we use grid-scan for generating the initial mesh and, because operations tend to be more localized, tree-traversal for re-meshing and re-analysis.

Figure 4a shows exemplary bS nodes (P1, P2, P3 in Figure 4a) that are created as follows.

- Vertices of bS within each NIO cell (e.g. P2 in Figure 4a) are tagged as such and are always used as finite element nodes.
- Additional bS nodes are created by intersecting bS with the boundaries of the NIO cells (P1 and P3 in Figure 4a).

The generation of valid elements within an NIO cell is straightforward if the cell does not contain bS-vertices (corner-nodes): nodes on bS and bIS belonging to the same NIO cell are simply linked to form quadrilateral and triangular elements (see the lower left portion of Figure 4b). When a corner is present, the corner node is linked to bS and bIS nodes within the cell and templates are used to form a web of triangular elements - see Figure 4b. To avoid generating elements with poor aspect ratios, the distances between nodes are checked by using a node-neighborhood test, and closely spaced nodes are merged into single nodes on bS. Figure 5 provides an example of this process.

< Figure 5 >

The FEM mesh is complete at the end of Stage 2. A regular mesh of quadrilateral elements in the interior results from a direct mapping of IN cells. On the boundary, NIO cells are associated with quadrilateral and triangular elements. It is important to note that, the

FEM mesh inherits the spatial addressability and structure of the hierarchical grid because elements and substructures are associated with the quadrants of the original decomposition. Figure 6 shows an example of a mesh generated by our automatic procedure.

< Figure 6 >

The Shephard-Yerry (S-Y) boundary-region meshing algorithm performs in/out tests on the mid-points and quarter-points of the edges of NIO cells, and then maps each NIO cell into one of a finite number of cut-quadrant forms; each cut-quadrant is then meshed. (We avoid such geometric approximations by computing exact points of intersection on bS.) The final stages of the S-Y algorithm move nodes in NIO cells to the boundary, and then eliminate ill-formed elements by using a Lagrangian relaxation procedure to smooth a triangulated version of the entire mesh. This last operation destroys the uniform quadrilateral interior mesh and also spatial addressability - because elements are not constrained to remain in their original cells.

#### 4 ANALYSIS OF HIERARCHICAL MESHES

This section summarizes a FEM analysis procedure that exploits the properties of the hierarchical, spatially addressable meshes described above. Recall that data specifying the finite elements in the initial mesh are accessed through the lowest level of the hierarchical grid.

One analytical simplification is immediately obvious: because the interior mesh elements are uniform, their stiffness matrices are identical if the material properties are homogeneous and thus only one stiffness matrix need be computed for all of the interior elements. Other, more important analytical simplifications accrue during both assembly and solution of the system of equations, because the hierarchical grid - which has provided spatial substructuring for meshing - can serve also as a multi-level analytical substructuring mechanism.

##### Assembly Procedure

Most FEM analysis procedures build a single stiffness matrix to cover the whole domain. Our Assembler builds and stores stiffness matrices for every non-OUT cell in the hierarchical grid. This is done bottom-up - see Figure 7 - by assembling son-matrices and condensing-out interior d.o.f.'s to build parent-matrices at each level. The parent nodes of

the interior mesh with identical sons (uniform) yield identical substructures, hence need be assembled only once. (The mesh generator tags identical interior-mesh nodes at all levels of the tree to facilitate this.)

< Figure 7 >

Figure 4.2 shows an initial mesh and substructures at various levels in the assembly process. Note in Figure 4.2 a that the initial mesh contains some higher-level substructures; these arise not from assembling lowest-level IN -elements, but from intermediate-level cells that were classified as IN and tagged as substructures during Stage-1 meshing. (The identical stiffness matrices for lowest-level IN -cells are needed in the assembly process only when IN -elements must be assembled with elements in NIO cells.)

< Figure 8 >

### Solution Procedure

Figure 9 illustrates various stages in the solution process. After loads and boundary conditions are attached to the root structure, the Solver computes the displacements of all nodal points on the boundary (i.e. the nodal points of the root substructure - see Figure 9a) and then traverses down the tree, recovering displacements of substructure nodes at each level. The displacements at all levels are saved in data records accessed through the hierarchical grid, and the lowest-level displacements are used to compute the stresses in the elements.<sup>2</sup>

< Figure 9 >

### Remarks on the Assembly and Solution Procedures

Our experience to date with this substructuring approach to analysis indicates the following.

- The hierarchical grid used for mesh generation has almost all of the data management facilities needed for analytical substructuring.
- The computing time and storage requirements for internal-element assembly are substantially reduced.
- We conjecture that our substructuring technique is asymptotically more efficient than the methods used in standard solvers. Preliminary result that support our conjecture will be reported in [KELA87].

---

<sup>2</sup> All analysis presented here are linear-static, based on linear isoparametric elements.

- Substructuring based on trees lends itself naturally to parallel (computer) processing.

More broadly our particular approach to substructuring seems promising for non-linear as well as linear analysis. In many practical problems (e.g. contact problems, fracture mechanics, localized plasticity), non-linear behavior occurs in isolated regions, and spatially localized analytical methods should prove to be efficient. (For example: during analysis regions that become non-linear can be tagged in the grid and handled specially.) In other types of problems one may want displacements and stresses only in small critical regions, and here again spatially localized methods seem very appropriate.

## 5 SELF-ADAPTIVE INCREMENTAL ANALYSIS

In this section we discuss first the techniques used for managing mesh refinement and incremental analysis, and then an error-driven algorithm for closing the feed-back loop in Figure 1.

### 5.1 Refinement and Re-Analysis

Assume that (1) a mesh has been constructed at the lowest level of the grid, (2) the mesh has been analyzed and the results stored in the grid and (3) evaluation of the results (discussed in the next subsection) has indicated that refinement is needed in a particular spatial region.

Two avenues for refinement are available: *h*-refinement and *p*-refinement. In *p*-refinement successively higher-order shape functions are assigned to the element formulation. To refine a particular element, the old stiffness matrix for the element is invalidated and a new matrix is computed from the new shape function. No new tree-nodes are generated, but the size of the stiffness matrix increases.

In *h*-refinement, existing elements are subdivided into smaller elements of the same type. To improve the geometric accuracy, localized *h*-refinement is done on the original geometric model rather than on the current finite element approximation. Thus to refine a particular element, one deletes the element, creates and classifies new vertices and nodes, and



inserts the smaller new elements into the grid. Discontinuities of displacements along edges where smaller elements abut on larger elements are avoided by using constraint equations.

< Figure 10 >

Figure 10 shows examples of localized refinement. Note that successive h-refinements improve the geometric approximation of the original solid. A maximum cross-element grading of 2:1 is maintained during refinement.

Storage for the new entities created by h-refinement could be provided by adding a whole new bottom layer to the grid, but this would be wasteful unless very extensive h-refinement is needed. If the h-refinements are sparse, small localized explicit schemes or linked-list methods are more efficient.

Assume now that the original mesh has been refined in a few regions using the methods just described, that the affected elements have been tagged, and that the refined mesh is to be re-analyzed. Clearly one wants to do incremental analysis, i.e. to use partial results from the earlier analysis insofar as possible. These results are available through the hierarchical grid; for example, a tree of K-matrices will exist - see Figure 7.

The incremental Assembler traverses the tree and by examining the sons of each parent node, detects new offspring and computes the appropriate stiffness matrices (Figure 11). Stiffnesses for unmodified elements are recovered from storage, and new and old stiffnesses are combined to form a modified substructure. If a node has no new offspring, the complete old substructure is reused. The incremental Solver works similarly, inspecting tags on data to distinguish valid and invalid old results and reusing the former whenever possible.

## 5.2 Self-Adaptive Algorithm

Our current algorithm for controlling self-adaptive incremental analysis operates as follows (see Figure 1). After a mesh (either initial or refined) has been analyzed, error indicators are computed for each element together with an estimate of the global error. If the global error exceeds a pre-specified limit, the systems calls for refinement and reanalysis in regions having large local errors. This process continues automatically until the global error estimate falls below the pre-specified limit.

Thus far we have done little research on errors per se, and our current error measures are crude. As in [KELL83], our element error-indicator ( $\epsilon_i$ ) is merely the average of the

stress jumps ( $J_s$  : normal and tangential) across each of the element's edges with dimension (h) and assuming linear isoparametric elements

$$\epsilon_i^2 = \frac{1-\nu}{E} \frac{h}{24} \int_{r_k} J_s^2 dr$$

normalized by the strain energy of the displaced model. Our global error estimator is simply the sum of the element error indicators. Figure 10c shows the computed values of the element error indicators for a sample problem. Note that, in the vicinity of the hole and around the re-entrant corner the data imply high stress gradients because the error indicators are high. Figure 10d shows an automatic refinement resulting from this set of error indicators.

An obvious improvement to the current algorithm: replace the single global error indicator with a hierarchical series of regional error indicators. These can be computed bottom-up in the tree, and should force selective refinement in cases where the overall (average) error is small but errors in small regions are high.

## 6 AUTOMATIC MESHING FOR 3-D PROBLEMS

In this section we present the algorithms that we are currently developing to extend to 3-D problems the automatic meshing procedure described in Section 3. Since our work is based on the octree generator built in the PADL-2 solid modelling system [HART83,KELA84], stage 1 of meshing - which includes (i) boxing the domain, (ii) subdividing the box into octal cells, (iii) classifying the cells as IN, OUT and NIO, and (iv) further subdividing and reclassifying NIO cells until a minimal level of subdivision is reached - is virtually completed. Figure 11 shows the interior octree for a PADL defined solid.

< Figure 11 >

Stage 2 involves associating each of the NIO cells (represented by the intersection of the solid with a grid-level octant) to a valid finite element topology. Before being decomposed into elements, NIO cells are classified as Simple (SNIO) or Complex (CNIO). SNIO cells, formed by the intersection of the grid-level octant with a single "cutting" surface, are topologically simple, as shown in Figure 12. CNIO cells, on the other hand, intersect the boundary surface and also contain vertices and edges coming from the solid's boundary. A typical CNIO cell is illustrated in Figure 13. Due to the differences in their geometry and topology, the decomposition of SNIO and CNIO cells proceeds along two different avenues.

&lt; Figure 12 &gt;

&lt; Figure 13 &gt;

Decomposition of SNIO cells

Since the number of possible configurations of SNIO cells is inherently limited, SNIO cells can be decomposed into finite elements by associating the cell to an appropriate template containing a mesh topology. Specifically, the number of possible cases is restricted to seven (the number of vertices of the original octant shaved off by the cutting surface identifies the appropriate template - Figure 14).

&lt; Figure 14 &gt;

The topologies embedded in the templates are not unique and include hexahedral, wedge, pyramid and tetrahedral isoparametric linear elements. However, as explained further on in this section - care has been taken in producing mesh topologies that, whenever possible, associate each uncut octant faces to a quadrilateral face of a hexahedral, wedge or pyramid element. We note, finally, that (a) in general, most of the NIO cells are classified as SNIO, and (b) SNIO decomposition is computationally inexpensive.

Decomposition of CNIO cells

The topological description of CNIO cells is not confined to a limited number of possible configurations. Hence, in this case mapping is of little use and the automatic decomposition of the cells can be done only by recursive element extraction. We are currently implementing a family of operators - based on the approach in [WOO84], Figure 15 - that works on the boundary representation (Brep) of the polyhedron associated with the CNIO cell. Because of the complexity of the operations involved - (i) scan the topological information contained in the Brep to identify a candidate element, (ii) verify the validity of the element, and (iii) extract the element and update the Brep - CNIO decomposition is considerably more expensive than template matching.

&lt; Figure 15 &gt;

Elements for 3-D analysis

The family of linear isoparametric elements used in the above decomposition schemes can be generated by collapsing a standard 8-node isoparametric brick element. Note that the use of pyramids is mandated by the necessity of preserving a regular interior mesh of hexahedral elements, whenever tetrahedral elements are introduced in the proximity of the

boundary. Pyramids allow interfacing triangular sides belonging to tetrahedral or wedge elements with quadrilateral faces of hexahedral elements without introducing discontinuities in the displacement field.

## 7 DISCUSSION

### Advantages

The main advantage we see is that mesh generation and mesh analysis are integrated and, in effect, collaborate under the control of the error evaluator. Thus the mesher only refines regions where refinement is needed, and the analyzer only computes "what's new" about a refined mesh. This type of efficient adaptive behavior is, in our opinion, the key to efficient automatic finite-element analysis.

Hierarchical substructuring is the driving principle in both the mesh generator and mesh analyzer.<sup>3</sup> It seems to be a very powerful principle of divide-and-conquer genre, in that it enables hard problems (object decomposition, equation-set solution) to be decomposed into smaller, tractable problems via spatial partitioning.

### Open Issues

We cite four sets of issues that will require extensive theoretical work.

1. Error measures and indicators: measures better than the ones we use currently are needed, especially for 3-D work.
2. Adaptive convergence: the convergence behavior of the self-adaptive process must be investigated (strong convergence properties are required for a truly automatic system).
3. Computational complexity: preliminary results let us conjecture that hierarchical substructuring techniques are asymptotically more efficient than the methods used

---

<sup>3</sup> The hierarchical tree might be viewed as a generalization of the structure described in [RHEI80]. However, the latter is applied in subdomains that are mapped to regular figures (squares and triangles), and Rheinboldt's tree addresses the element partitioning induced in the regular figures. By avoiding mapping we are able to use the same structure for both meshing and analysis; further, the regularity of our structure permits systematic cell numbering and, hence, data access through calculated addresses rather than through searching or table lookup.

in standard solvers, but an in-depth study is needed to prove/disprove our conjecture.

4. Non-linear analysis: our approach to substructuring appears promising for non-linear analysis.

While the issues above are certainly important in the long term, in the immediate future one other issue – completing the extension of our meshing and analysis system to 3-D problems – is more pressing. The current status of 3-D work is as follows :

- The 2-D spatial substructuring techniques for managing analysis and adaptive re-meshing and re-analysis extend gracefully to 3-D, and indeed most of the 2-D control code is directly usable in 3-D.
- The major open issues lie in Stage 2 of the automatic meshing procedure, specifically in decomposition of CNIO cells. A promising approach, based on a family of element extractors, is currently being implemented.

In summary , we believe that hierarchical substructuring as embedded in the experimental system described here represents an important contribution on the road to genuinely automatic finite element analysis.

#### ACKNOWLEDGEMENTS

Herb Voelcker, former director of the Production Automation Project and currently at Cornell University, contributed extensively to this research. Also we acknowledge the contribution and the encouragement of John Goldak, of Carleton University, and of Vic Genberg, of the Eastman Kodak Company. The computer-output displays were produced on equipment donated by Tektronix, Inc., and other Industrial Associate companies of the Production Automation Project provided both equipment and funds for the work. Sustaining support was provided by the National Science Foundation under Grant(s) ECS-8104646 & DMC-8403882. The findings and opinions expressed here those of the authors and do not necessarily reflect the views of the various sponsors.

#### **REFERENCES**

- [BABU78] I. Babuska and W. C. Rheinboldt, "A-posteriori error estimates for the finite element method", INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, vol. 112, pp. 1597-1615, 1978.

- [CAVE85] J. C. Cavendish, D. A. Field and W. H. Frey, "An approach to automatic three-dimensional finite element mesh generation", *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING*, vol. 21, pp. 329-347.
- [GAGO83] J. P. De S. R. Gago, D. W. Kelly, O. C. Zienkiewicz and I. Babuska, "A posteriori error analysis and adaptive processes in the finite element method: Part II - Adaptive mesh refinement", *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING*, vol. 19, pp. 1621-1656, 1983.
- [HART83] E. E. Hartquist, "Public PADL-2", *IEEE COMPUTER GRAPHICS & APPLICATIONS*, vol. 3, no. 7, pp. 30-31, October 1983.
- [JACK80] C. L. Jackins and S. L. Tanimoto, "Oct-trees and their use in representing three-dimensional objects", *COMPUTER GRAPHICS & IMAGE PROCESSING*, vol. 4, no. 3, pp. 249-270, November 1980.
- [KELA84] A. Kela, "Programmers guide to the PADL-2 octree processor output system", *INPUT/OUTPUT GROUP MEMO. No. 15; Production Automation Project, University of Rochester; January 1984.*
- [KELA87] A. Kela, "Automatic finite element mesh generation and self-adaptive incremental analysis through solid modeling", Ph. D. Dissertation, Production Automation Project, University of Rochester, 1987.
- [KELL83] D. W. Kelly, J. P. De S. R. Gago, O. C. Zienkiewicz and I. Babuska, "A posteriori error analysis and adaptive processes in the finite element method: Part I - Error analysis", *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING*, vol. 19, pp. 1593-1619, 1983.
- [LEE82] Y. T. Lee and A. A. G. Requicha, "Algorithms for computing the volume and other integral properties of solids: Part II - A family of algorithms based on representation conversion and cellular approximation", *COMMUNICATIONS OF THE ACM*, vol. 25, no. 9, pp. 642-650, September 1982.
- [PEAN79] A. G. Peano, A. Pasini, R. Riccioni and L. Sardella, "Adaptive approximation in finite element structural analysis", *COMPUTER & STRUCTURES*, vol. 10, pp. 332-342, 1979.

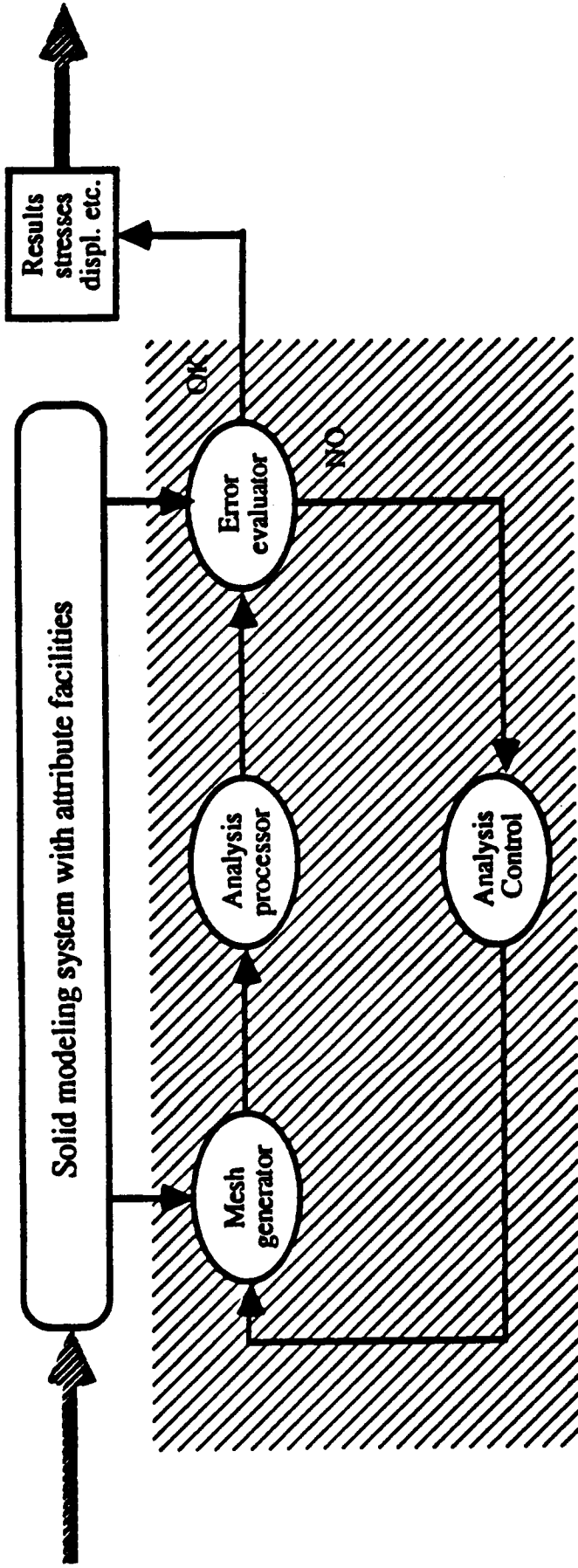
- [PERU82] R. Perucchio, A. R. Ingraffea and J. F. Abel, "Interactive computer graphic preprocessing for three-dimensional finite element analysis", *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING*, vol. 18, pp. 909-926, 1982.
- [REQU83] A. A. G. Requicha and H. B. Voelcker, "Solid modelling: Current status and research directions", *IEEE COMPUTER GRAPHICS & APPLICATIONS*, vol. 3, no. 7, pp. 25-37, October 1983.
- [RHEI80] W. O. Rheinboldt and C. K. Mesztenyi, "On a data structure for adaptive finite element mesh refinements", *ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE*, vol. 6, no. 2, pp. 166-187, June 1980.
- [SUHA74] J. Suhara and J. Fukuda, "Automatic mesh generation for finite element analysis", in *ADVANCES IN COMPUTATIONAL METHODS IN STRUCTURAL MECHANICS AND DESIGN*, J. T. Oden, R. W. Clough and Y. Yamadoto eds., Univ. of Alabama Press, pp. 607-624, 1974.
- [WOO84] T. C. Woo and T. Thomasma, "An algorithm for generating solid elements in objects with holes", *COMPUTERS & STRUCTURES*, vol. 18, no. 2, pp. 333-342, 1984.
- [WÖRD84] B. Wordenweber, "Finite element mesh generation", *COMPUTER-AIDED DESIGN*, vol. 16, no. 5, pp. 285-291, September 1984.
- [YERR83] M. A. Yerry and M. S. Shephard, "A modified quadtree approach to finite element mesh generation", *IEEE COMPUTER GRAPHICS & APPLICATIONS*, vol. 3, no. 1, pp. 39-46, January/February 1983.
- [YERR84] M. A. Yerry and M. S. Shephard, "Automatic three-dimensional mesh generation by the modified-octree technique", *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING*, vol. 20, pp. 1965-1990, 1984.

C-3

## LIST OF FIGURES

- Fig. 1 An automatic finite element analysis system.
- Fig. 2 A quadtree approximation
- Fig. 3 First stage of the automatic meshing algorithm.
- Fig. 4 Second stage of the meshing algorithm (a) generation of bS nodes and  
(b) linking bS and bIS nodes.
- Fig. 5 Node relocation to get well-formed elements.
- Fig. 6 Example of automatically generated 2-D FEM mesh.
- Fig. 7 Assembly via multi-level substructuring.
- Fig. 8 Substructures at various levels during assembly.
- Fig. 9 Nodal displacement at stages of the solution process.
- Fig. 10 Refinement driven by error indicators.
- Fig. 11 Solid domain with interior octree produced by PADL-2
- Fig. 12 Typical Simple NIO cell.
- Fig. 13 Typical Complex NIO cell.
- Fig. 14 Template driven decomposition of SNIO cells.
- Fig. 15 Element extractors for CNIO cells.





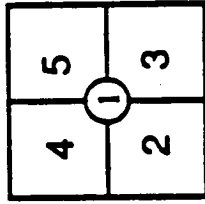
 Data Flow Requiring Human Action

 Automatic Process

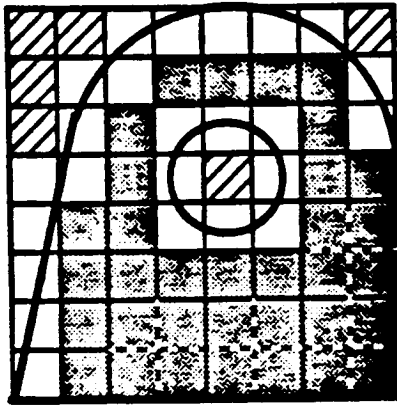
 Process Controlled Data Flow

Figure 1

**QUADRANT  
NUMBERING**



**OBJECT**



**NODE STATUS**

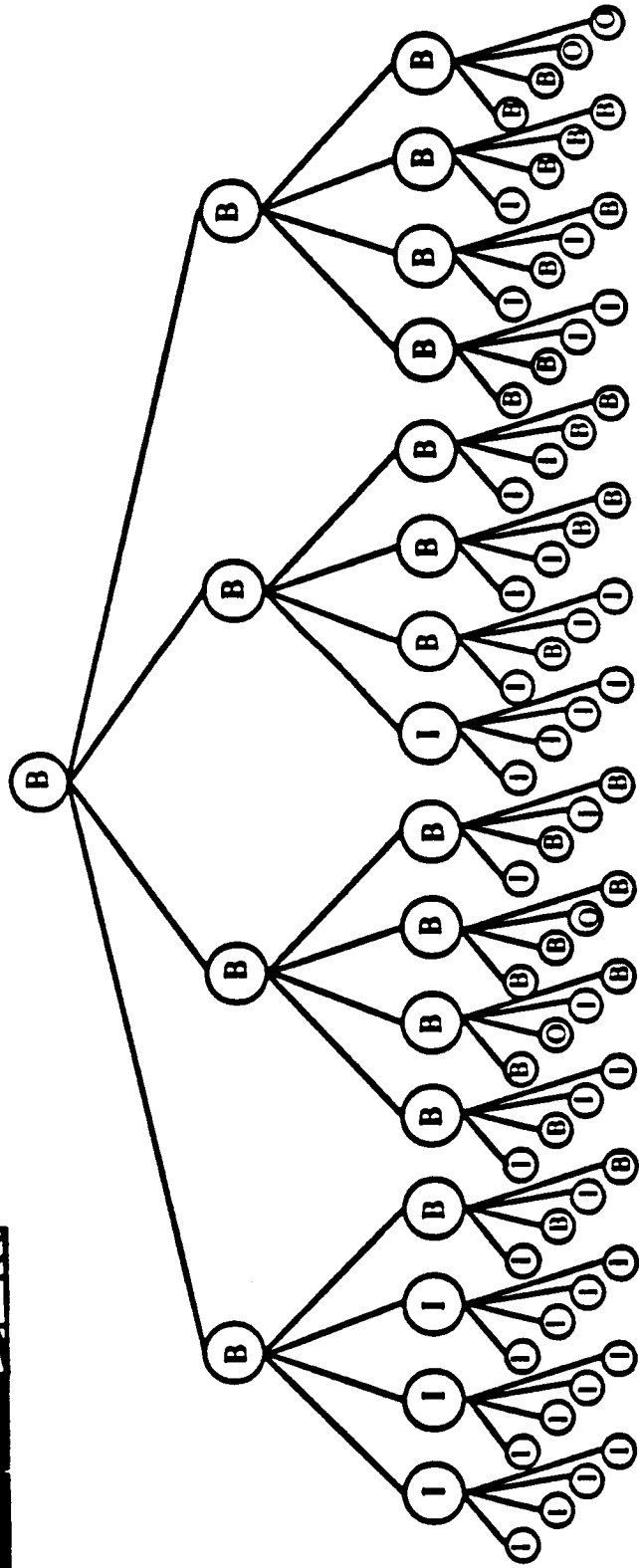
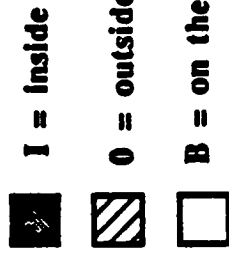
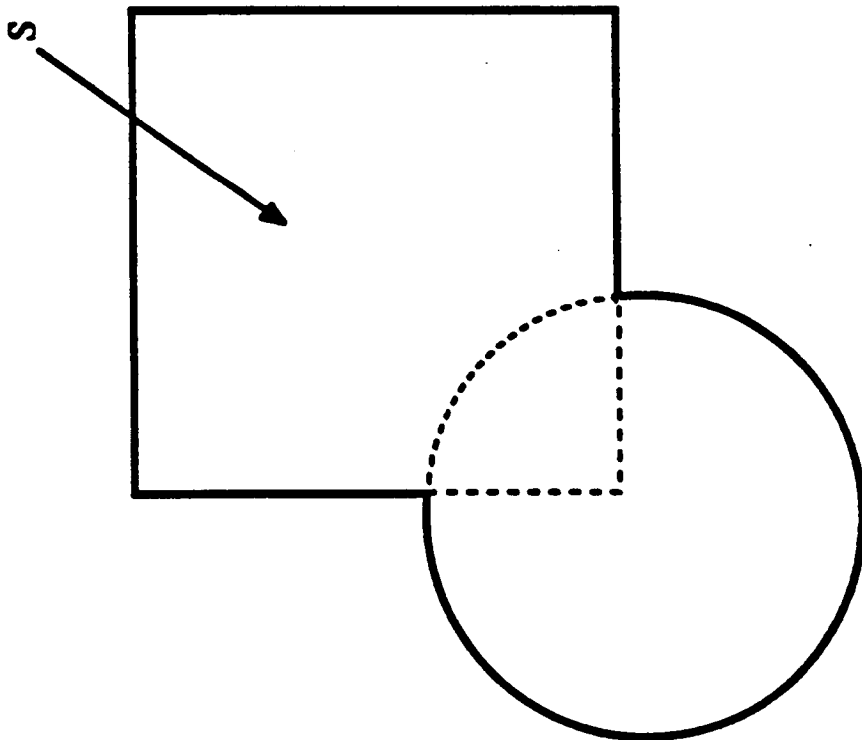
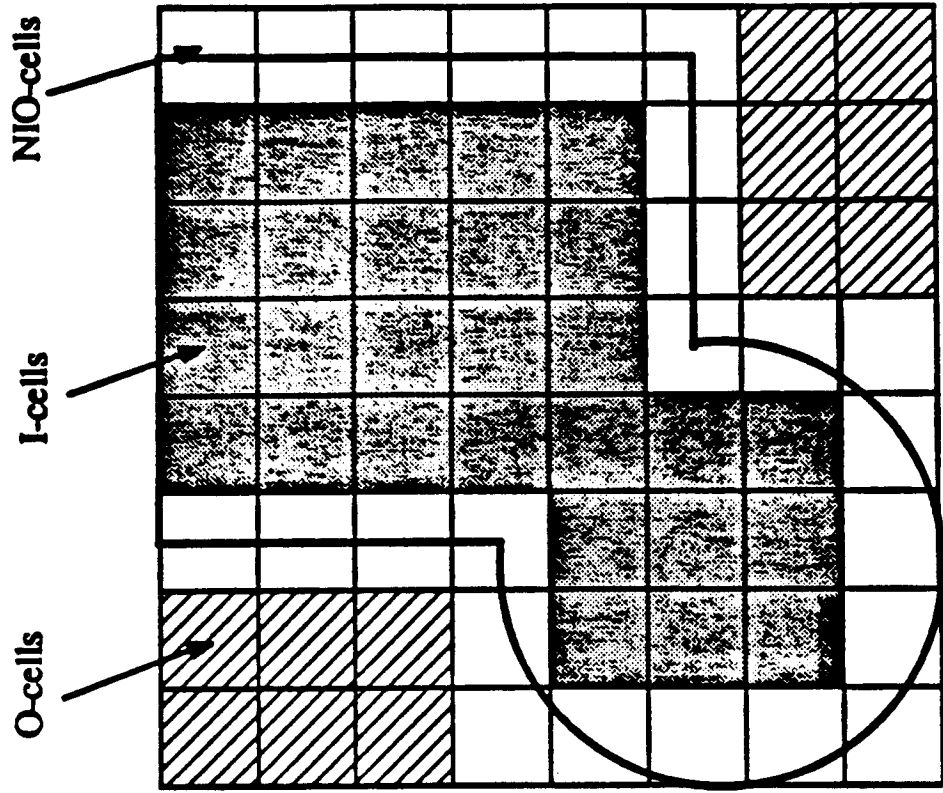


Figure 2

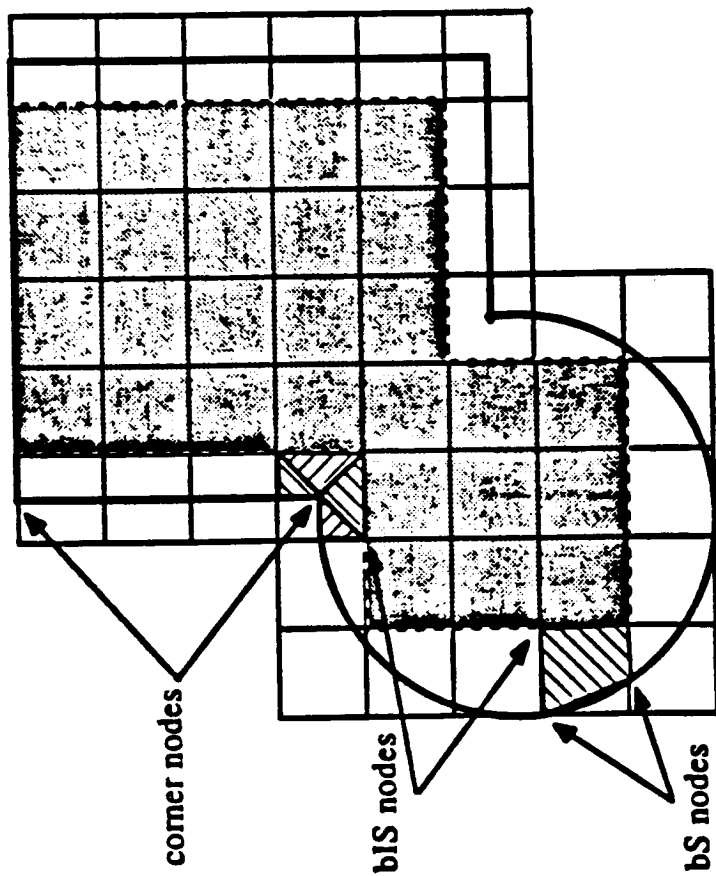
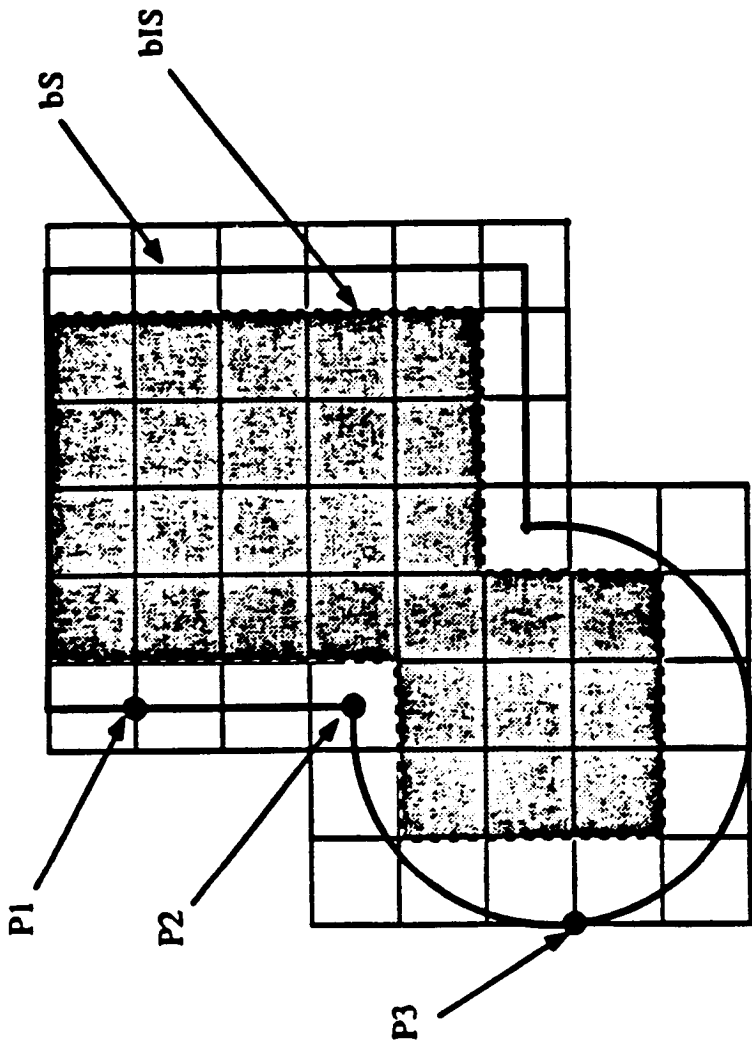
ORIGINAL PAGE IS  
OF POOR QUALITY



(b)

(a)

Figure 3



ORIGINAL PAGE IS  
OF POOR QUALITY

Figure 4

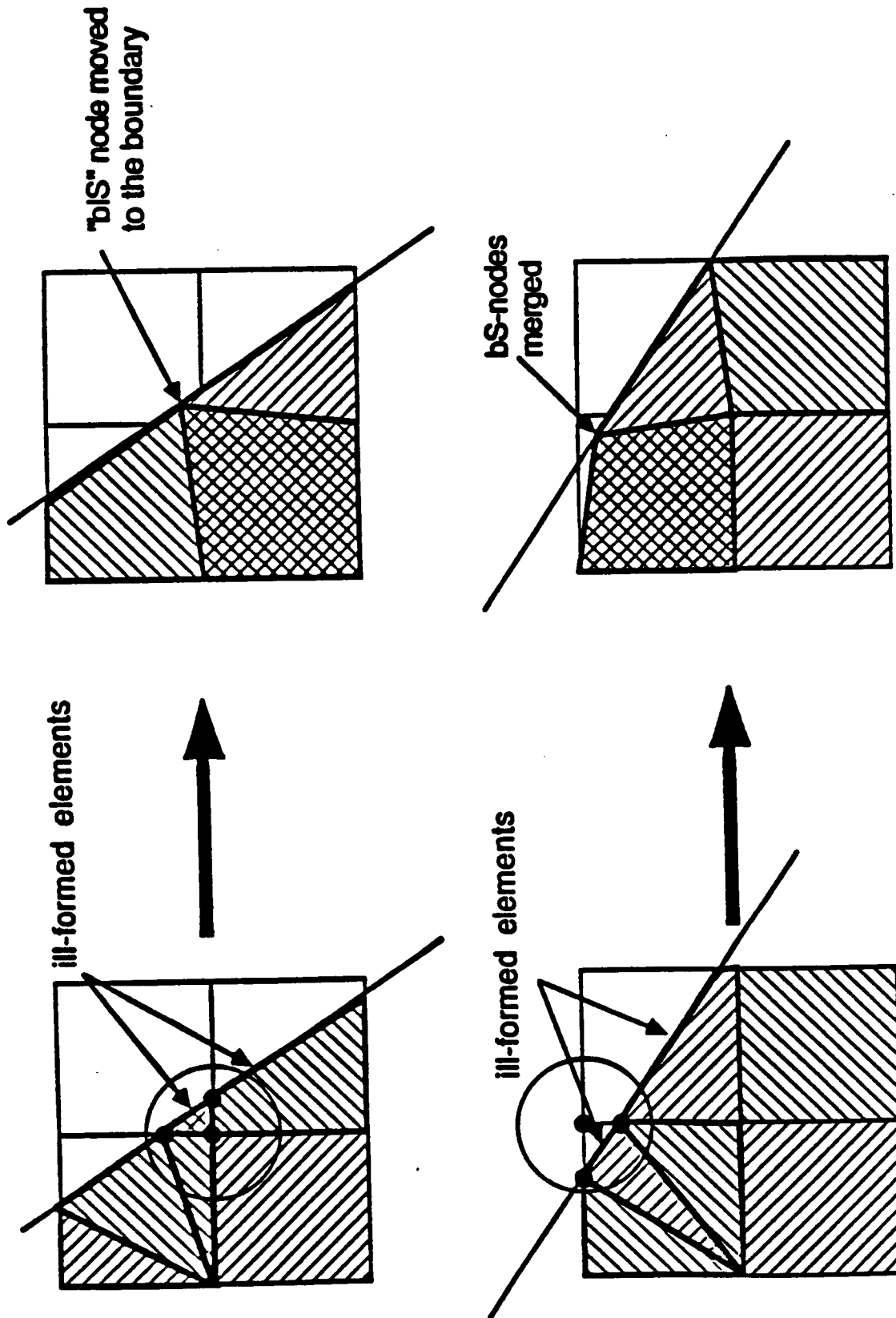


Figure 5

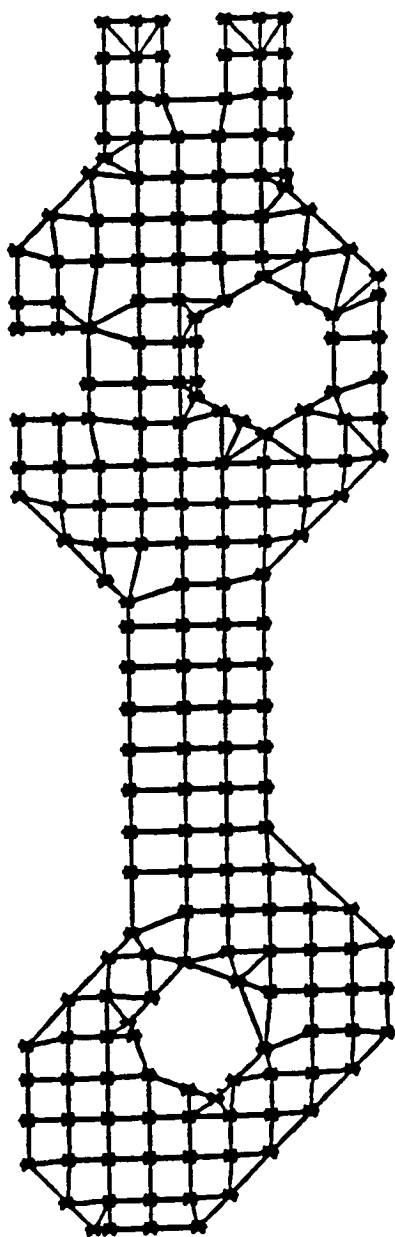


Figure 6

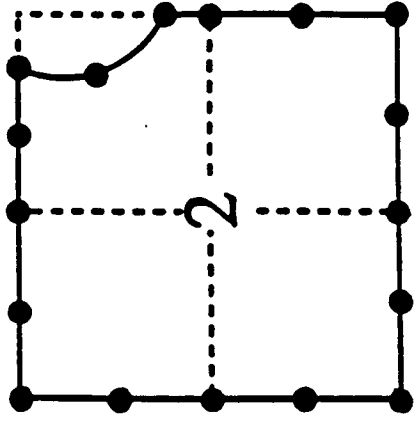
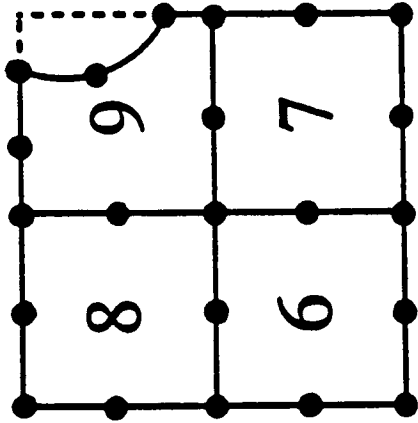
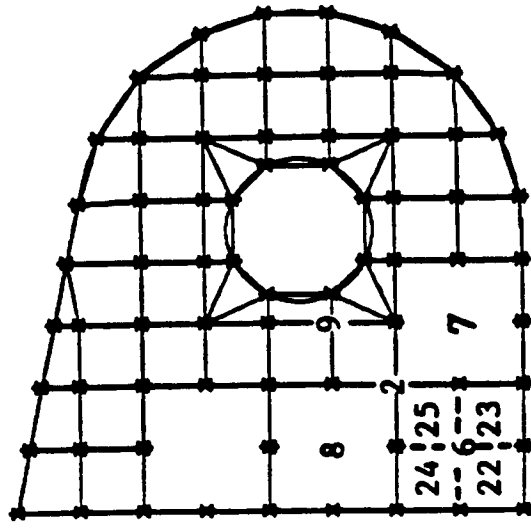
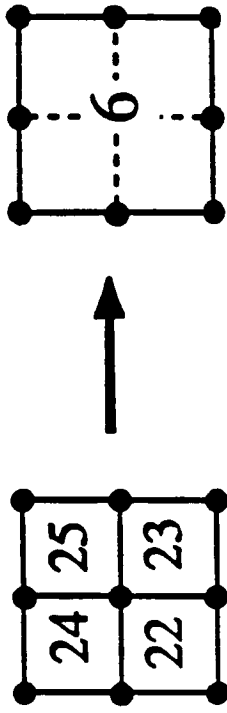
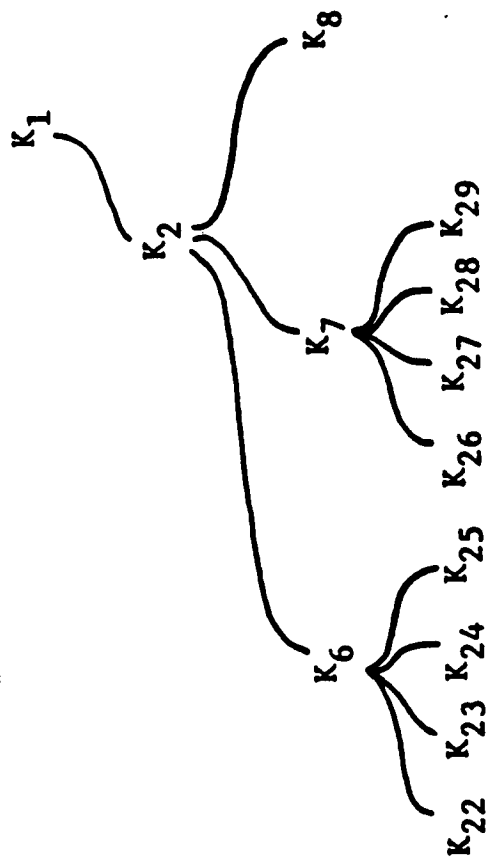
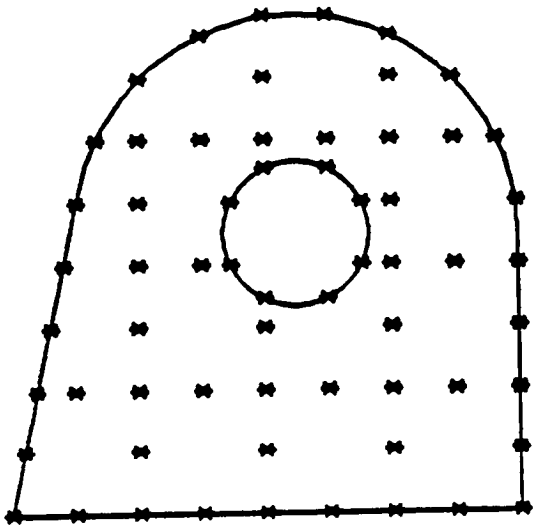
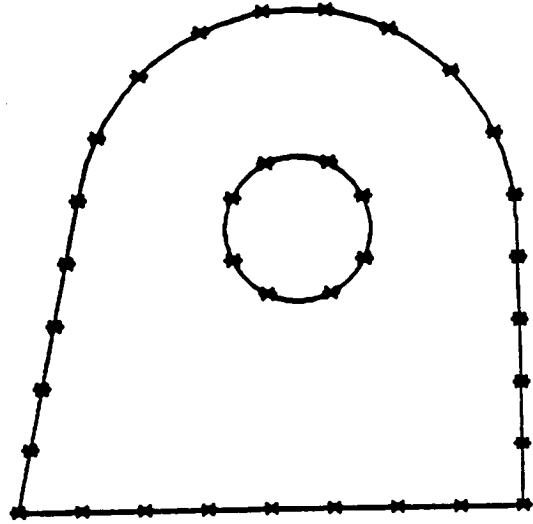


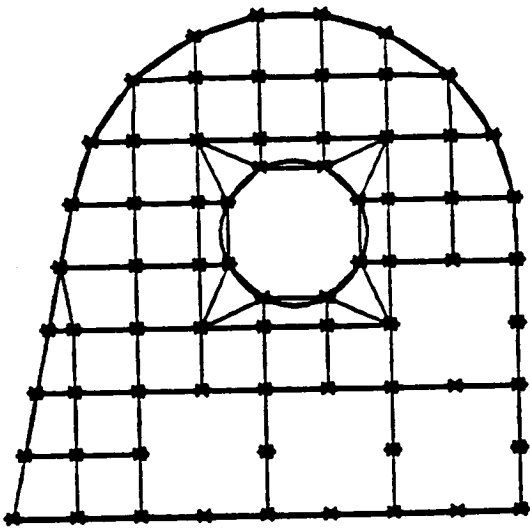
Figure 7



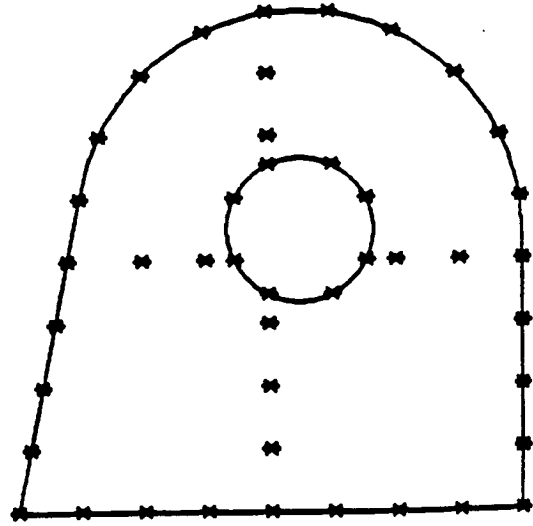
(b)



(d)



(a)



(c)



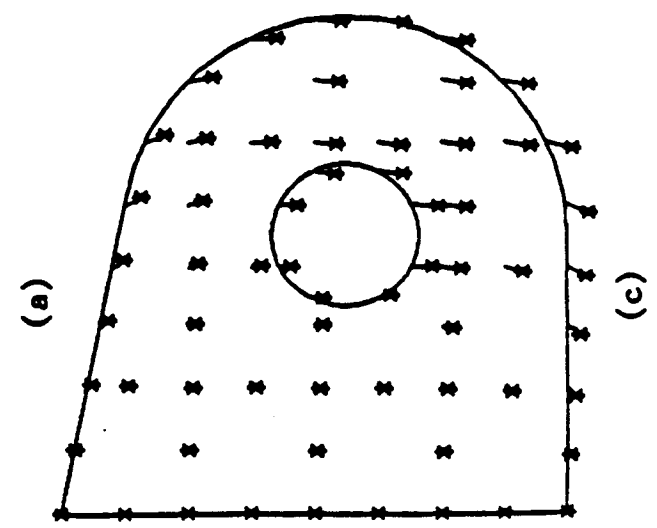
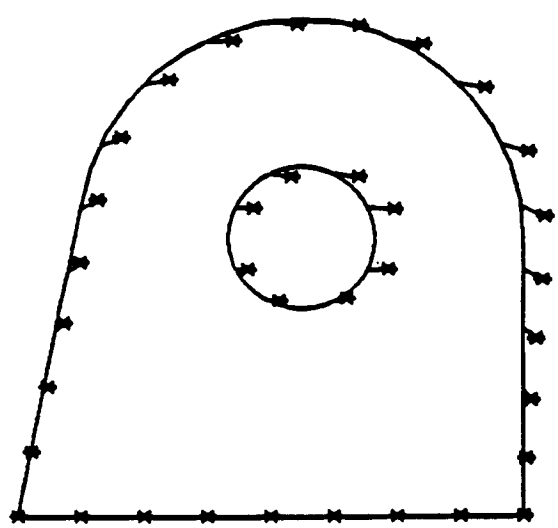
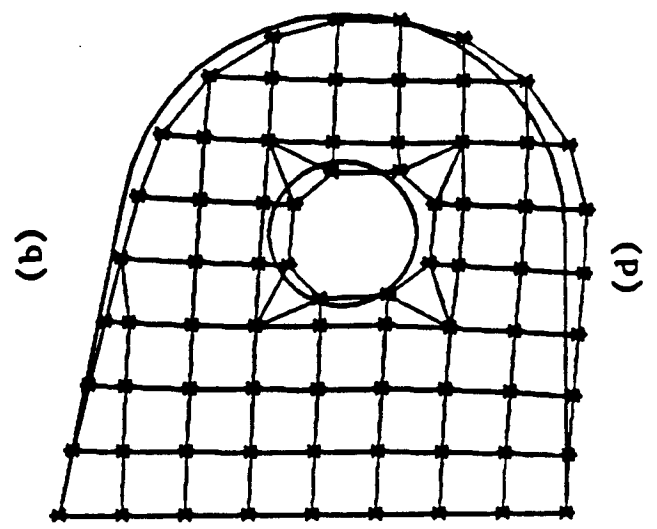
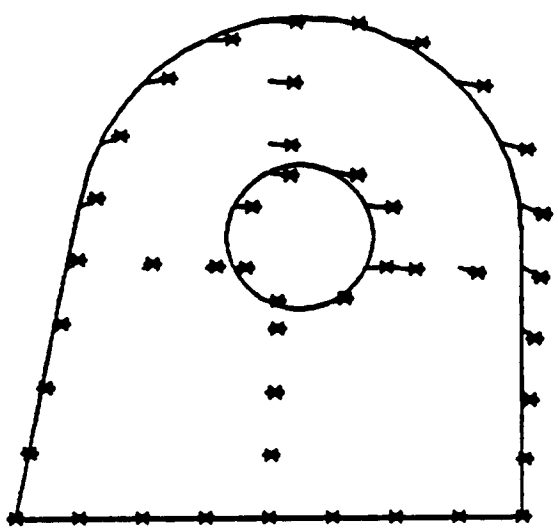
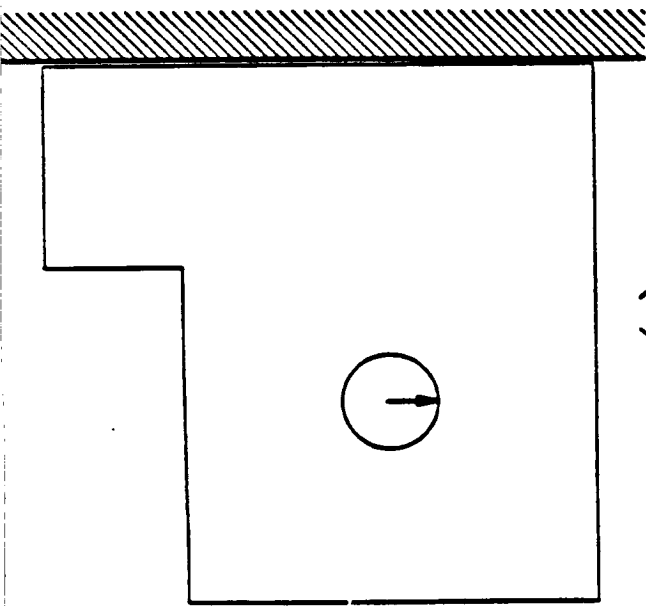
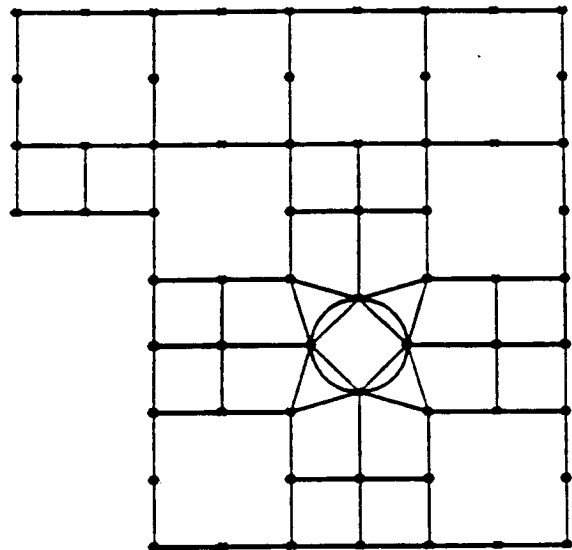


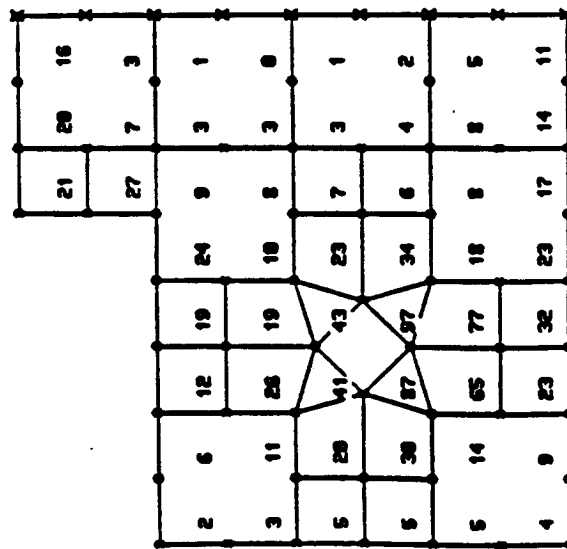
Figure 9



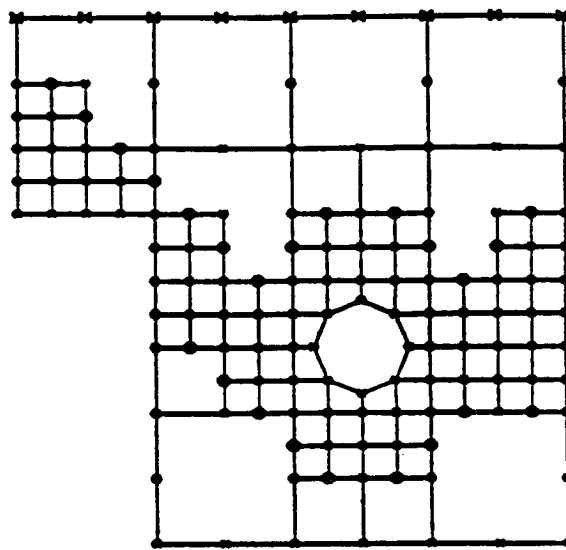
(a)



(b)



(c)



(d)

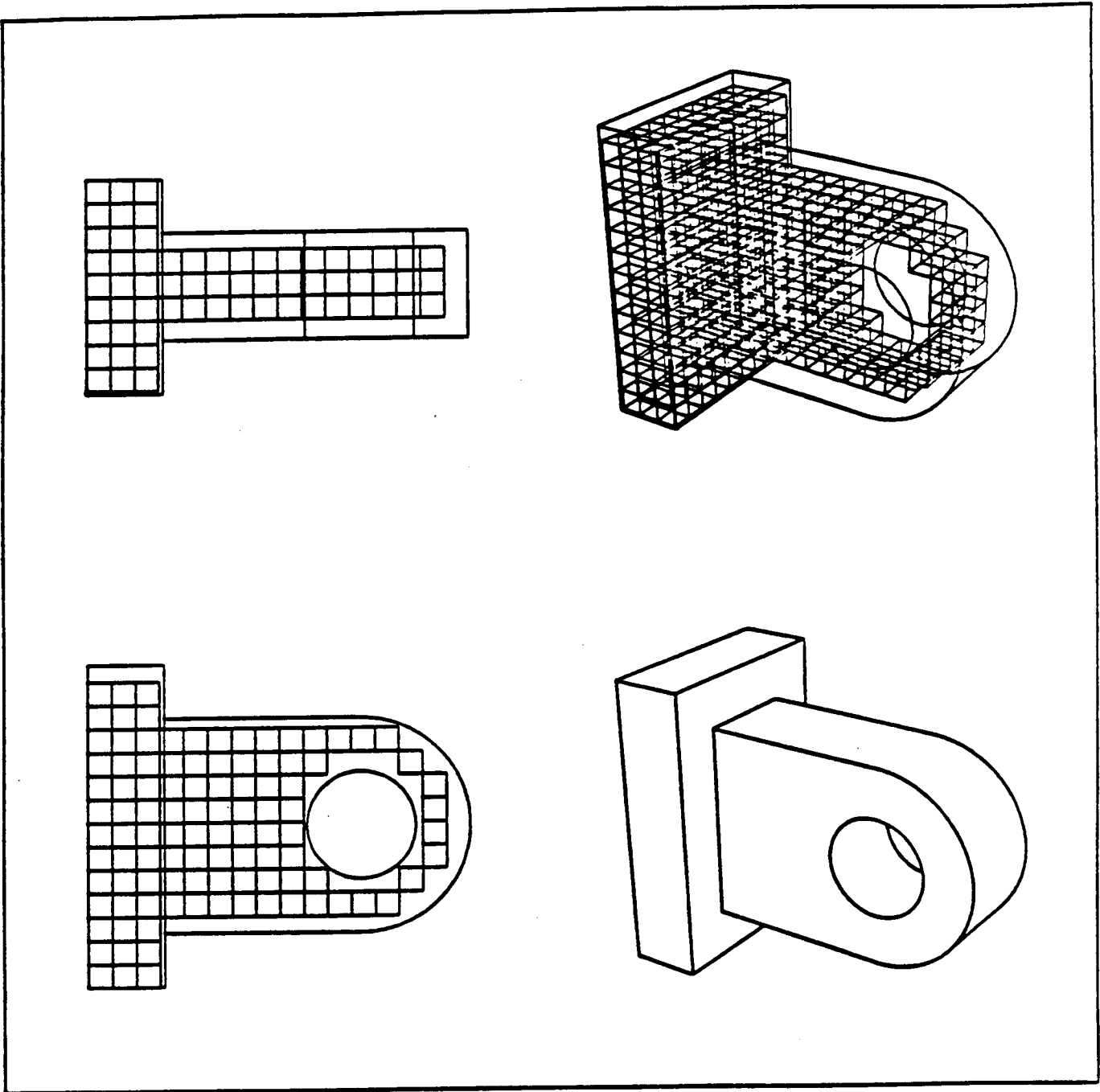


Figure 11

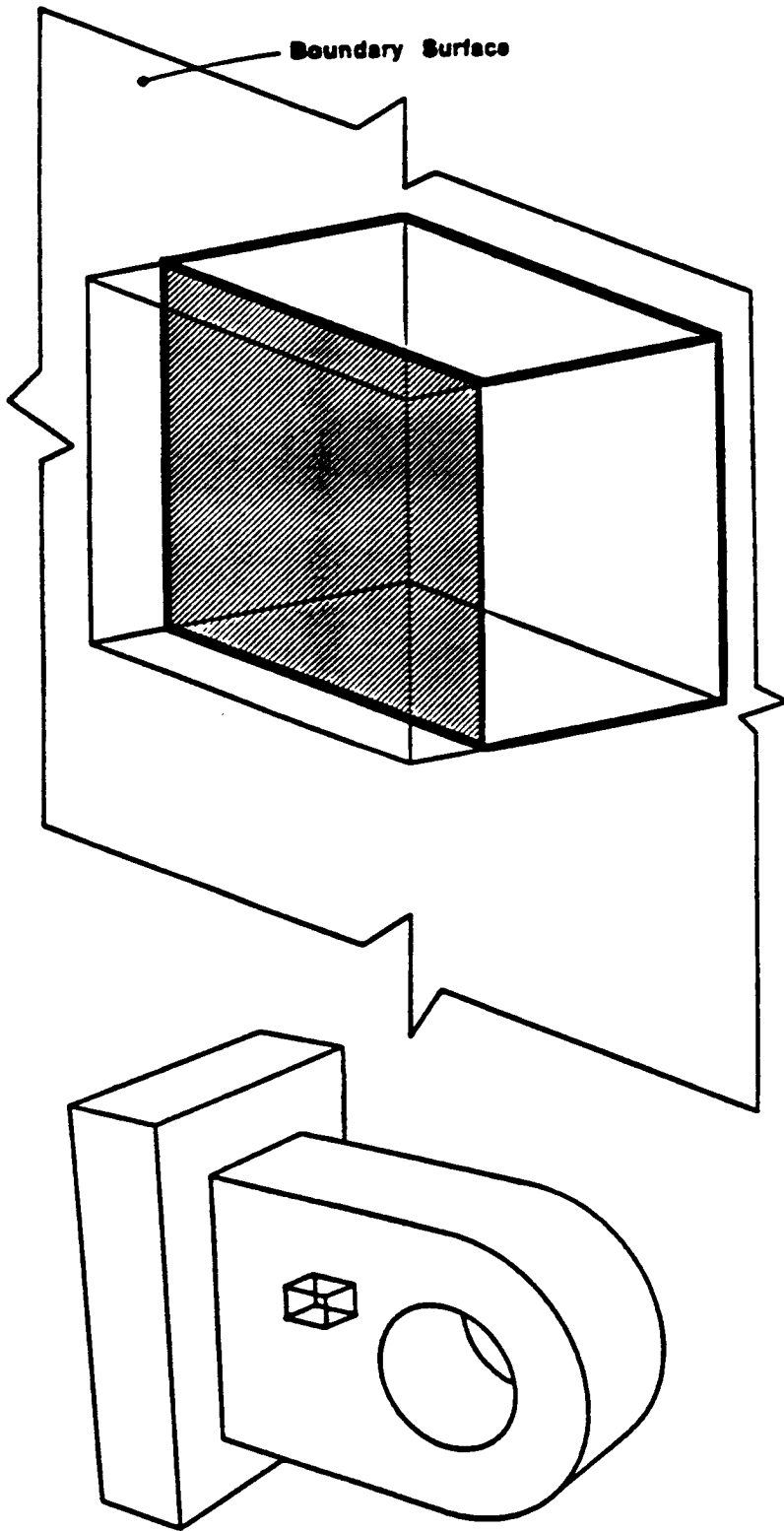


Figure 12

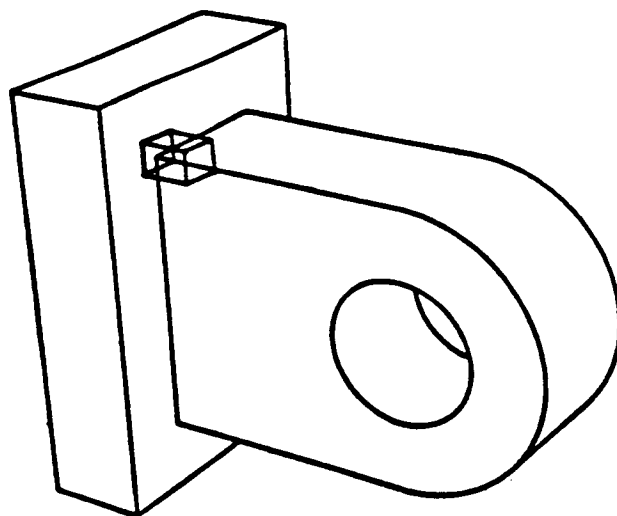
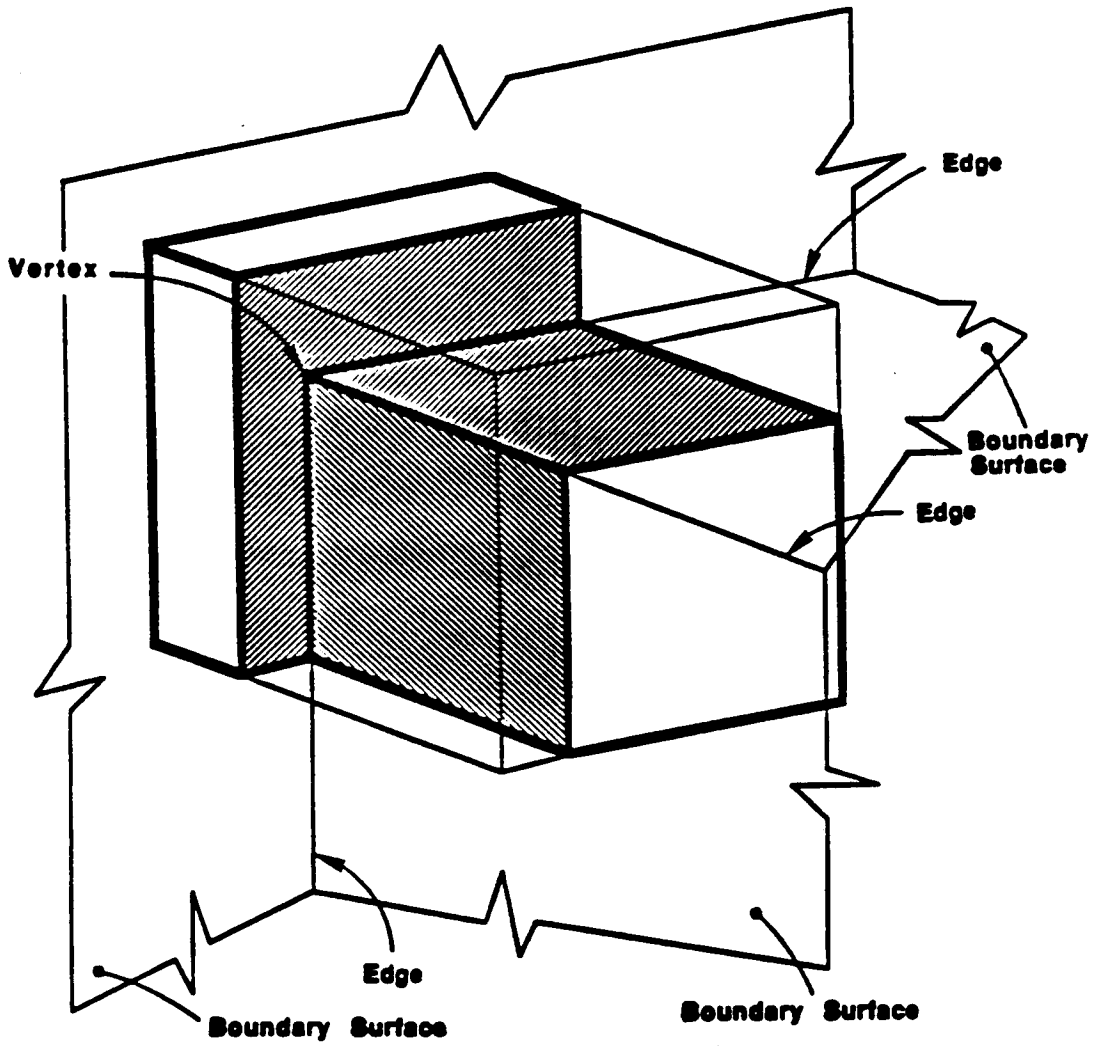
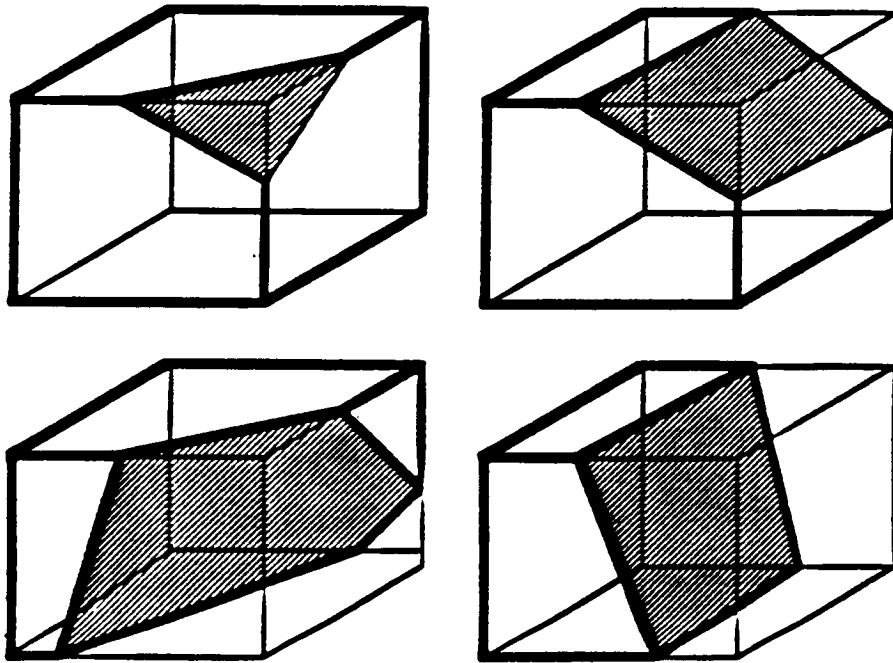


Figure 13

### SNIO Cells



### Template-derived F.E. Topologies

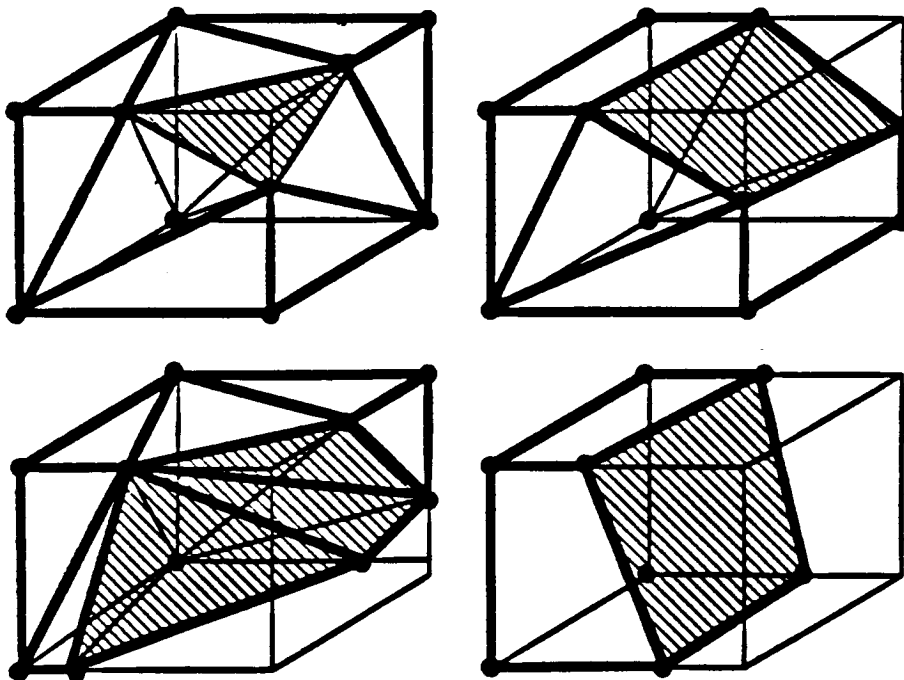
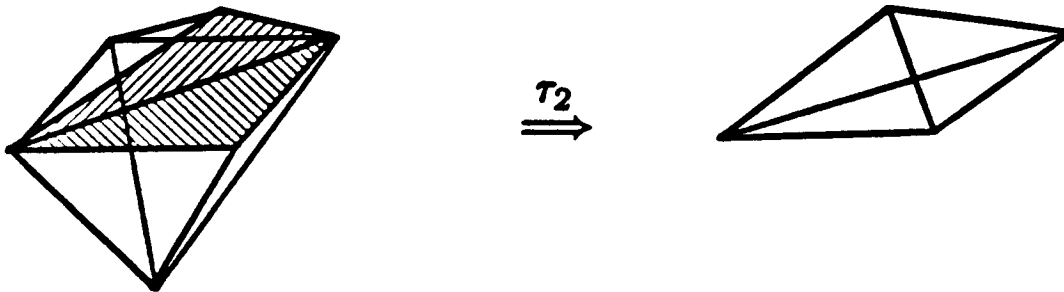
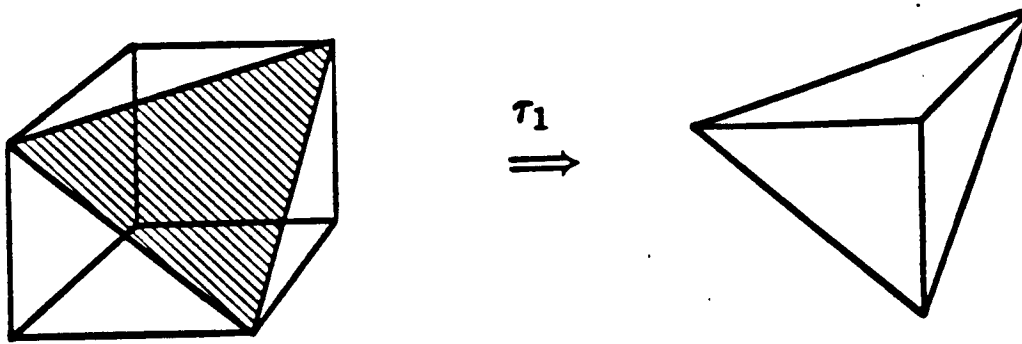


Figure 14

- Extraction of tetrahedra :  
Operators  $\tau_1$  and  $\tau_2$  (Woo & Thomasma, 1983).



- Extraction of pyramids : operator  $\tau_3$ .

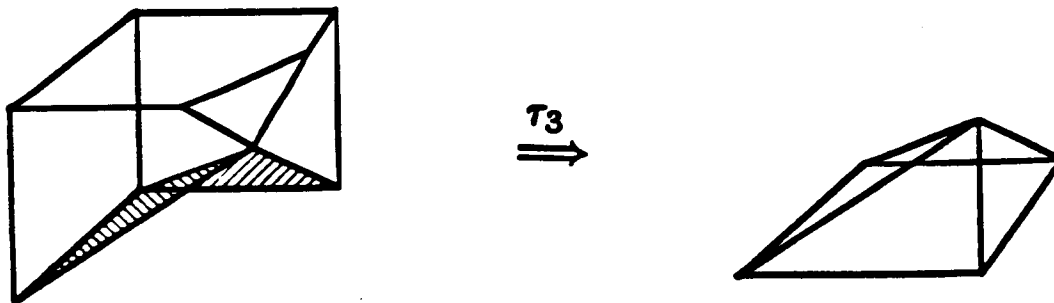


Figure 15

59-61  
125795

# Toward Automatic Finite Element Analysis

**Ajay Kola**

Research Assistant

**Ronato Porucchio**

Assistant Professor

Producer, Automation Project

Mechanical Engineering Department

University of Rochester

Rochester, NY

**Herbert Voelcke**

Graduate Fellow

Sibley School of Mechanical

and Aerospace Engineering

Cornell University

Ithaca, NY

Two problems must be solved if the finite element method is to become reliable and affordable "blackbox" engineering tool. FE meshes must be generated automatically from CAD databases, and mesh analysis must be made self-adaptive. The experimental system described here solves both problems in 2-D through spatial and analytical substructuring techniques that are now being extended into 3-D.

The essence of mechanical design is interplay between human creativity and incisive analysis. The procedure for designing a critical component or structure typically runs as:

1. Prepare a candidate design.
2. Analyze the design using the finite element (FE) method.
  - (a) Model the designed structure and its loading and constraints.
  - (b) Analyze the loaded model.
  - (c) Assess the validity of the analytical results.
  - (d) Repeat steps 2(a—c) until acceptable analytical results are obtained.
3. Assess the candidate design.
4. Repeat steps 1—3 until the design is acceptable.

Thus the design process is doubly iterative because current FE techniques are not single-shot blackbox tools with guaranteed reliability; they require human judgement and "tuning." It follows that the (in)efficiency of the inner analysis loop is a strong determinant of the quality of the final design when the cost of design matters, as is usually the case. If analysis can be made cheap, fast, and reliable, more alternatives can be considered and better designs will result.

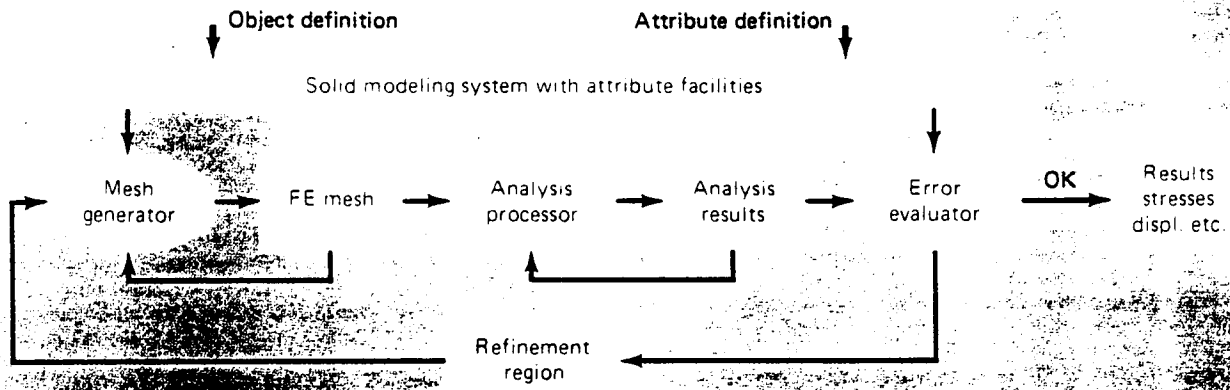
Let's look more closely at the analysis procedure. During step 2(a), the design is modeled as a properly connected mesh of suitably sized and shaped elements (triangles,

quads, etc.) from an element library. Its loading and constraints are modeled by assigning suitable constants (e.g. displacement and load values) to particular nodes of the mesh. The operative words here are "suitably sized and shaped" and "properly connected". If the elements are too large or have bad aspect ratios, or if the mesh as a whole does not obey the combinatorial sharing rules of FE mesh decompositions, inaccurate and inconsistent results will accrue because the mathematical conditions underlying the FE method will have been violated. In the early days of FE analysis, the analyst was wholly responsible for mesh and element integrity. Today, computer graphics preprocessors help ensure proper connectivity, but the selection, placement, and sizing of elements are still the user's responsibilities.

Step 2(b), analysis of the loaded model, is usually performed by using a standard code such as Nastran and Ansys. This step is largely automatic, and the popular codes are well debugged though sometimes expensive to run.

For step 2(c), assessing the validity of the results, there are no standard methods and the analyst's judgement plays a critical role. In the early days, when "results" were huge tables of numbers, assessment was largely a black art. Graphics postprocessors, which can display colored contour plots of stresses, temperatures, and so forth, enable experi-





**Fig. 1 An automatic finite element analysis system.**

enced analysts to identify trouble spots (such as regions with high cross-element gradients) quite effectively.

During step 2(d), the analyst refines the mesh by subdividing troublesome regions into smaller elements, and then reanalyzing the whole.

Obviously, automation of the whole process will make design more systematic and efficient by replacing the analyst's judgement with mathematical criteria. Two new tools make automation of the FE mesh feasible:

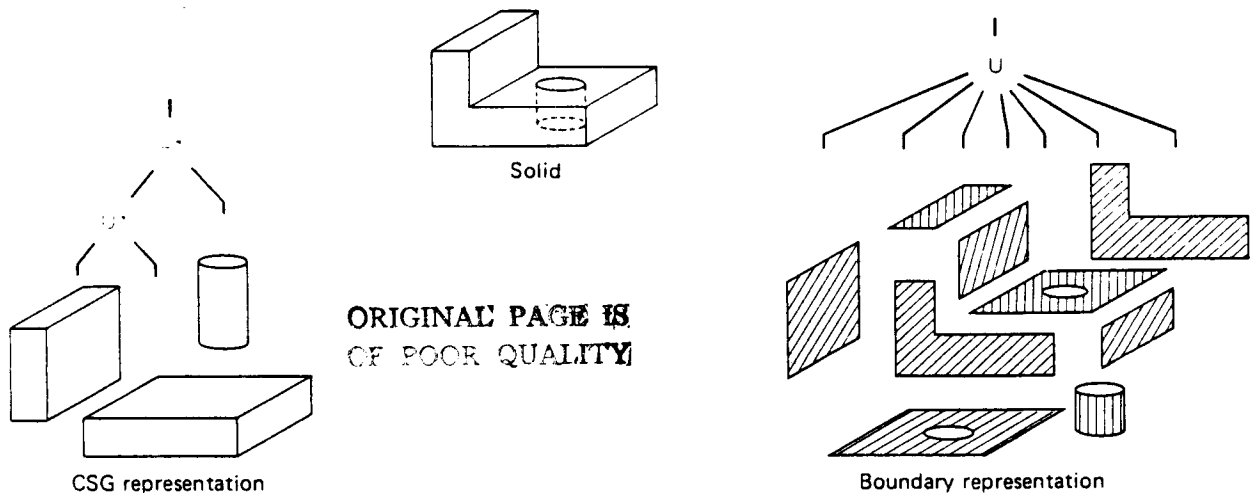
- Solid modeling technology [1, 2] enables designers to create and store in CAD systems informationally complete "master models" of mechanical parts and products. From there, one should be able to generate FE meshes automatically.
- New algorithms for analyzing errors in a finite element analysis [3-7] systematic means to automate the results assessments of step 2(c).

One more tool is needed: a good method for using error indicators to refine the FE mesh automatically. Another tool, while not essential, is also very desirable: a method for analyzing refined meshes selectively or incrementally so that results already computed for unmodified regions of a mesh

can be reused rather than recomputed.

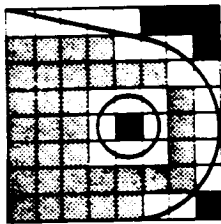
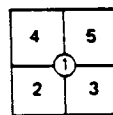
Figure 1 shows a design for an automatic analysis system. In this system, the user defines the structure to be analyzed in the Solid Modeling System (SMS) together with attributes such as boundary conditions, loads, material properties, and certain analytical parameters. The mesh generator produces a discretized model (the FE mesh) from the geometric definition and attribute specifications. (Attributes can determine, for example, the positions of some nodes.) The analysis processor performs FE analysis: it computes primary and secondary field variables (in general, the displacements vector at nodal points and the stress tensor within the elements) for the loaded and constrained FE mesh. Finally, the error evaluator compares error estimates derived from the analysis output with specified tolerances, and either accepts the results or requests a new analysis of a modified mesh. In the latter case, the error evaluator indicates the regions in the current model that require refinement. The inner mesh-generation loop and mesh-analysis loop in Figure 1 connote localized mesh refinement and incremental reanalysis.

This approach to automatic FE analysis has been embodied in an experimental 2-D system whose underlying princi-



**Fig. 2 Two unambiguous representation schemes for solids.**

Object

ORIGINAL PAGE IS  
OF POOR QUALITYQuadrant  
numbering

Node status

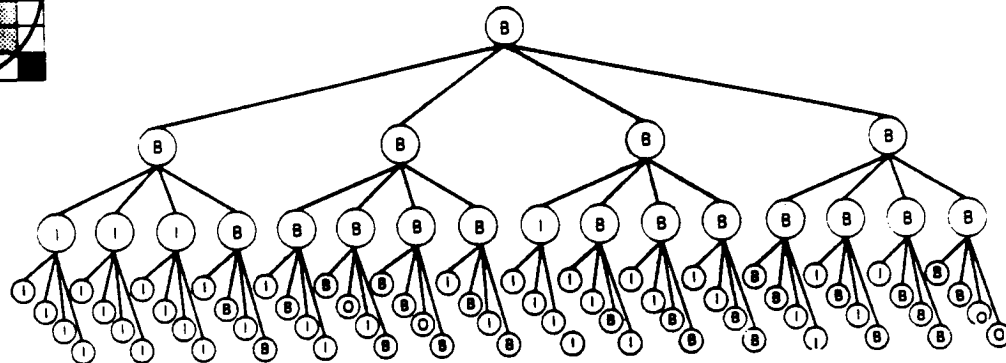
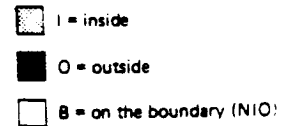


Fig. 3 A quadtree approximation.

ples will be explained. (Our actual implementation is somewhat different than Figure 1 for reasons of computational efficiency.) All meshes and analytical results that appear in this article were produced with this experimental system. This article summarizes a moderately complicated topic; for technical details, see [8].

### Automatic Mesh Generation

Most "automatic" meshing utilities in contemporary CAD systems actually operate from wireframe descriptions of objects via mapping algorithms. The user must partition the domain, which is represented by a collection of edges, into a set of topologically simple subdomains in which meshes can be generated automatically. This approach is unsuitable for a fully automatic meshing procedure because it depends on human judgement both to guide meshing and to resolve ambiguities in the wireframe representation.

Genuinely automatic mesh generation must start from an unambiguous representation of the object to be analyzed, and thus needs some form of SMS. Nearly all current SMS systems are based internally on one or both of the representation schemes illustrated in Figure 2 [1, 2]. Constructive Solid Geometry (CSG) exploits the notion of "adding" and "subtracting" simple solid building blocks (via set-union and set-difference operations). Boundary schemes describe solids indirectly via sets of faces which are represented by sets of edges that bound finite regions of surfaces. The various schemes that have been proposed for automatic mesh generation can be divided into two families: recursive spatial subdivision (quadtree and octree) schemes, and triangulation and other schemes. After a brief discussion of the second family, we will focus on the first.

### Triangulation and Other Schemes

Wordenweber [9] and Cavendish [10] have developed two different two-stage approaches to automatic triangulation of solid domains. Wordenweber's procedure first does surface

triangulation of the boundary of the solid, and then performs solid triangulation in the interior. The tetrahedral meshes that result are coarse and usually contain distorted elements that must be refined to be useful for analysis.

In the Cavendish method, points are injected into the solid, and then a solid triangulation is induced in which the points become nodes of tetrahedral elements. The main working tool of the second-stage triangulation is a Delaunay algorithm that generates valid meshes of tetrahedral elements within convex hulls of node points. Good methods are still being sought for inserting points automatically during the procedure's first stage.

In both of these approaches, mesh refinement is done by splitting existing elements. Because refinement is driven from an FE mesh rather than from the original solid model, refinement does not improve the geometric approximation of the original solid. Also, the meshes are not spatially addressable.

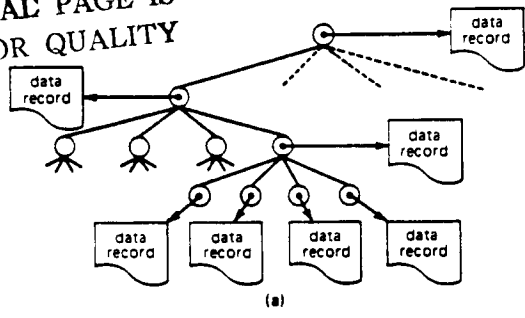
A few commercial CAD systems claim automatic meshing facilities that can involve triangulation but the principles are proprietary. Lee's method [11], which has been described publicly and implemented in 2-D, exploits the decomposition inherent in CSG representations rather than triangulation or spatial subdivision. Briefly, Lee generates "natural" distributions of points in each CSG primitive and then induces a uniform spatial distribution of points over the whole object by "thinning" points in regions where primitives overlap; a mesh of quadrilateral and triangular elements is then grown over the points in the object.

### Recursive Spatial Subdivision

We approximate the object to be meshed with a union of disjoint, variably sized rectangles (in 2-D) or blocks (in 3-D). These are generated by recursively subdividing a spatial region enclosing the object, rather than the object itself. Figure 3 shows a 2-D example.

The object (a rounded plate with a hole) is "boxed" to

ORIGINAL PAGE IS  
OF POOR QUALITY



ORIGINAL PAGE IS  
OF POOR QUALITY

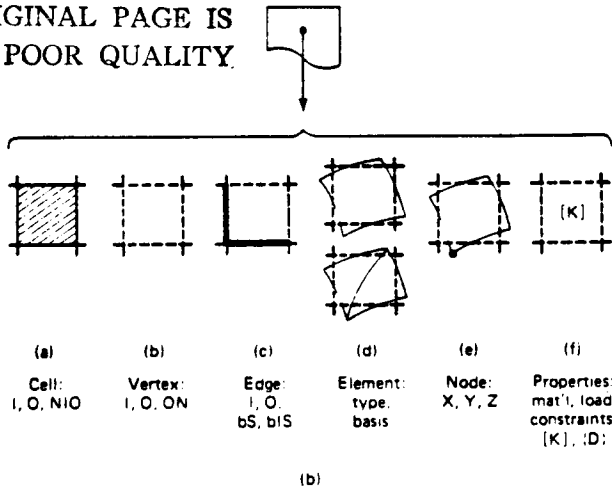


Fig. 4 Hierarchical structure for the FE model.

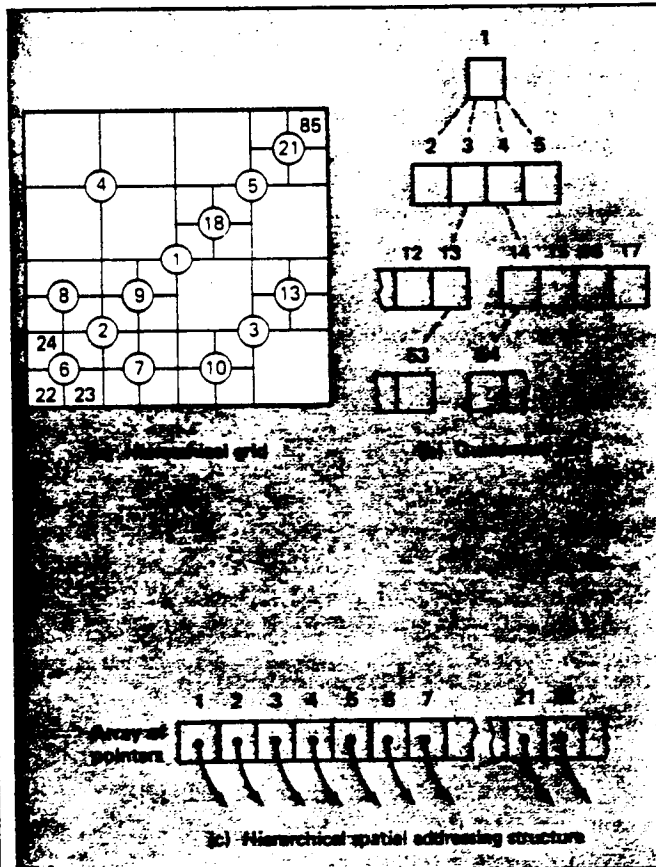


Fig. 5 Directly addressable hierarchical grid.

establish a convenient minimal spatial region, and then the box is decomposed into quadrants. When a quadrant can be classified as wholly inside or outside of the object, subdivision ceases; when a quadrant cannot be so classified, it is subdivided into quadrants. So this process continues until some minimal resolution level is reached. (In 3-D, the decomposition proceeds by octants.) Approximations produced this way can be represented by logical trees whose nodes have four or eight sons (see Figure 3), hence the popular names "quadtrees" and "octrees" [12].

As we will explain, inside cells of a spatial decomposition can be easily converted into "nice" mesh elements, but boundary cells require further processing lest their literal translations into mesh elements introduce bogus high-gradient stress regions in the analytical results. We'll deal with boundary-cell processing later; for the moment, assume that the "B" cells in Figure 3 are somehow reshaped into valid mesh elements that closely approximate the object's boundary.

Recursive spatial decompositions have two intrinsic properties, hierarchical structure and spatial addressability, that are central to the mesh refinement and incremental analysis techniques described later. These intrinsic properties, plus an extrinsic (engineered) property called logical addressability, warrant discussion.

**Hierarchical structure.** The tree structure in Figure 3 results from the subdivision rule used to produce the decomposition, and one can think of the tree as an organizing or cataloging structure for data describing particular regions of space.

Figure 4(a) illustrates this notion by showing a data record associated with each node of the tree; Figure 4(b) shows data pertinent to automatic mesh generation that might be stored within such a record. These include classification of the spatial region represented by the node as inside, outside, or on the boundary (Figure 3); shape functions for a few (typically one) finite elements associated with the region; and properties associated with the finite elements, such as one or more stiffness matrices, external constraints, and so forth.

At the lowest level of the tree one finds the smallest spatial regions and simplest finite elements. As one ascends the tree the regions become larger (encompassing multiples of four or eight elemental regions) and the finite elements become superelements with associated ("assembled") stiffness matrices, collected constraints, and so forth. Such an organization is ideally suited to mesh refinement by subdivision and incremental mesh analysis.

**Logical addressability.** Given the notion of a tree as an organizing structure for hierarchical spatial data, how should such a structure be mapped into computer storage as a data structure, and how does one gain access to it to store and retrieve data? The tree diagrams in Figures 3 and 4 suggest the classical approach: represent a tree with a linked list in which nodes are addressed indirectly through downward pointers to sons and perhaps lateral pointers to siblings. The data record associated with each node is addressed through a

special pointer stored with the node. Thus one has access to data by following pointers downward from the root of the tree.

Alternatively, a recursive spatial decomposition can be viewed as a directly addressable hierarchical grid (see Figure 5) in which the number of cells in each linear dimension is an integer power of two. The key here is a systematic scheme for numbering all possible nodes of the underlying tree. In Figure 5(a), "1" represents the enclosing box, 2—5 represent specific quadrants of "1," "6"—"9" represent quadrants of "2," and so on. The underlying relation, which can be applied recursively, is:

The four sons of a parent node  $P$  are  $[4 * P - 2, 4 * P - 1, 4 * P, 4 * P + 1]$ , and the parent of  $P$  is  $(P + 2) \text{ div } 4$ .

These numbers can be used as indices for a single array of pointers to data records, as shown in Figure 5(c). Thus, to access the spatial data for a particular node in the underlying tree, one merely calculates an array index through a simple formula and follows the single pointer stored there. This is usually much faster than the pointer-following method noted above but it carries a storage penalty. Specifically, the pointer array in Figure 5 (c) must be large enough to accommodate all possible nodes in the tree.

If the lowest-level grid in Figure 5 (a) requires  $N^*N^*K$  units of storage ( $N^*N^*N^*K$  in 3-D) for pointers and data records, one needs:

$$\frac{K * (2^D * (1 + \log_2 N) - 1)}{2^D - 1}$$

units of storage for the worst-case whole tree, where  $D$  is the dimension of the space and "log" is log-2. Thus a 2-D hierarchical grid requires at most about 33 percent more storage than the  $N^*N^*K$  units needed for its lowest level; in 3-D only about 14 percent more storage is needed.

**Spatial addressability.** Suppose that we know the geometric size and spatial position of the "1" cell (the overall box) in Figure 5(a). We can quickly compute the index of any cell in the hierarchy from its size and position, and conversely from an index we can quickly compute the size and position of the associated spatial cell (an example is in Table I). We have already seen that cell indices allow access through a single pointer to data associated with the cell, and thus we can associate, without searching, spatial regions with stored data and stored data with spatial regions. This is what is meant by spatial addressability.

In practical terms, if a particular region of an object proves troublesome either in mesh generation or mesh analysis, one has direct access to pertinent mesh and analytical data to take localized corrective measures.

### An Automatic Meshing Procedure Based On Spatial Subdivision

This procedure produces a spatially addressable FE mesh embedded in the lowest level of a hierarchical grid. Higher levels of the grid are used during construction of the mesh and when the mesh is analyzed, refined, and incrementally

reanalyzed. The procedure starts with a representation in an SMS of the object to be meshed, and operates in two stages. The first stage meshes the interior of the object by spatial subdivision and the second extends the mesh to the object's boundary. The following descriptions are in 2-D; 3-D extensions are in the final section.

The use of quadtree and octree methods for automatic mesh generation was pioneered by Shephard and Yerry [13, 14]. Our work is similar to theirs but important differences will be noted as we go along.

**Stage 1: interior meshing.** The object  $S$ , Figure 6(a), is enclosed in a box, Figure 6(b), which is recursively subdivided into a grid whose smallest cell size determines the element size (or element density) of the initial FE mesh. This minimal size is determined by subdividing cells until no cell contains more than one connected boundary segment of  $S$ . As the subdivision proceeds the cells are classified as being "IN"  $S$ , "OUT" of  $S$ , or neither in nor out ("NIO"). Cells

Table I

#### Finding a Spatial Position from a Cell Index

This procedure computes the center position and half-size of cell  $P$  when the enclosing box ( $P=1$ ) is square, has size (boxsz) and is centered at (bx, by). The procedure is invoked as:

*GetPosition (P, cellx, celly, cellsz)*

where cellx, celly are the x,y coordinates of the center of the cell and cellsz is the half-size of the cell; cellx, celly, and cellsz are initialized respectively as bx, by, and boxsz/2.0. The following algorithm is based on the cell numbering scheme shown in Figure 5.

procedure *GetPosition (P, cellx, celly, cellsz);*

{Traverse tree upwards to root-node, from cell (P) marking all ancestors of the cell.}

CellNo := P;

ParentLevel := 1;

while (CellNo < 1)

do begin

SonNo[ParentLevel] := (CellNo + 2) rem 4;

CellNo := (CellNo + 2) div 4;

ParentLevel := ParentLevel + 1;

end; { while }

{Move down the tree through the ancestors of the cell updating each ancestor location to terminate at cell location.}

for i := ParentLevel - 1 downto 1

do begin

cellsz := cellsz/2.0;

case SonNo[i]

of

0: cellx := cellx - cellsz; celly := celly - cellsz

1: cellx := cellx + cellsz; celly := celly - cellsz

2: cellx := cellx - cellsz; celly := celly + cellsz

3: cellx := cellx + cellsz; celly := celly + cellsz

end; { case }

end; { for }

end; { GetPosition }

ORIGINAL PAGE IS  
OF POOR QUALITY

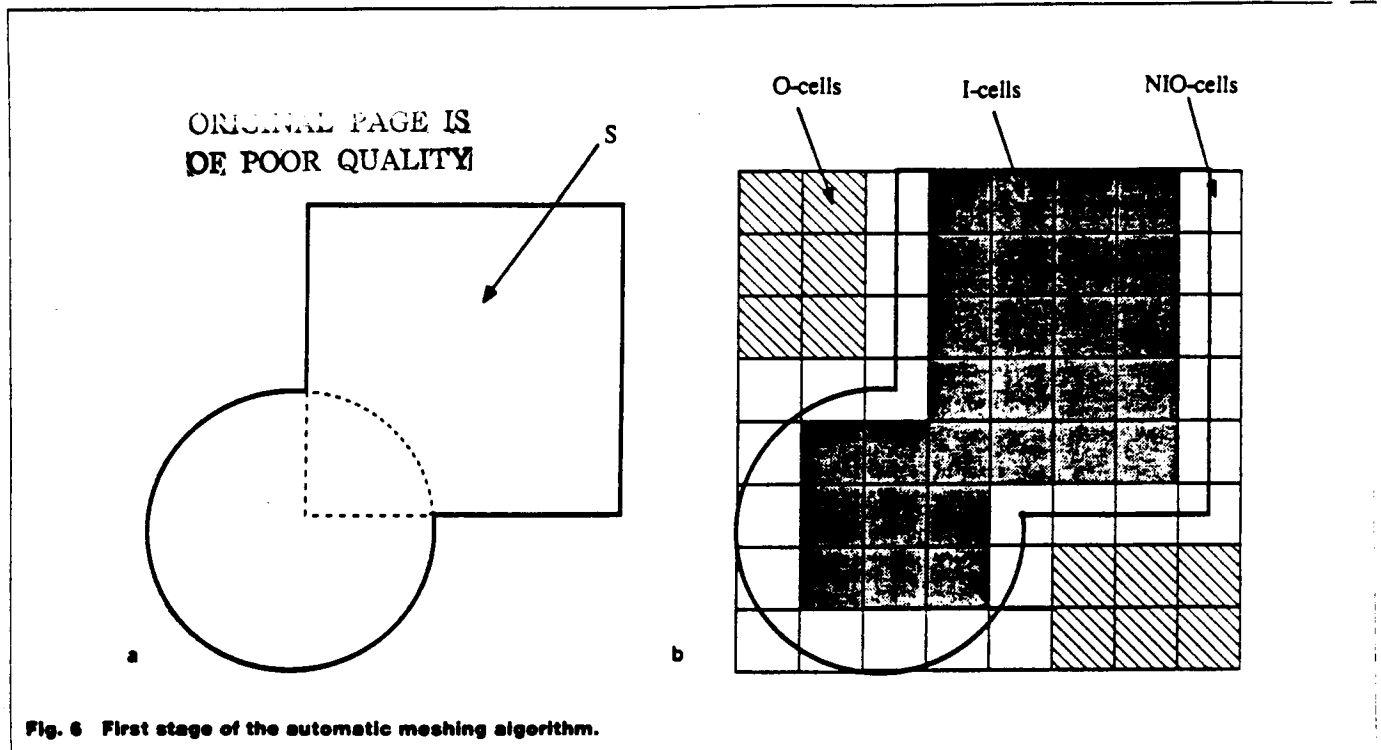


Fig. 6 First stage of the automatic meshing algorithm.

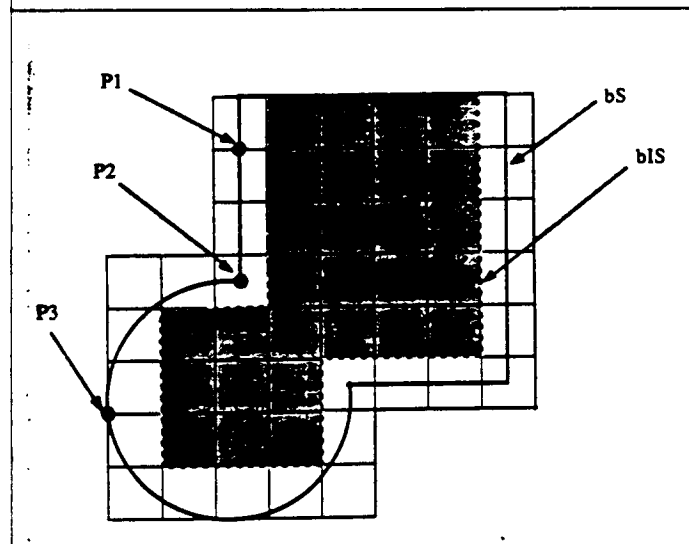


Fig. 7 Generation of  $bS$  nodes in stage 2 of the meshing algorithm.

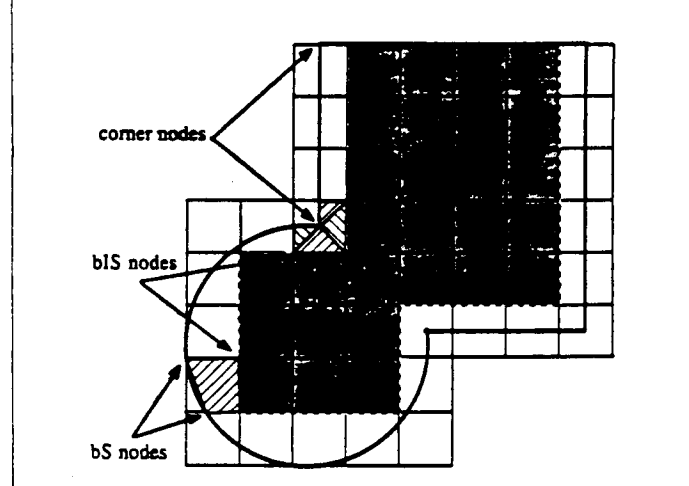


Fig. 8 Element generation via linking  $bS$  and  $bIS$  nodes.

classified as IN at higher levels in the hierarchy are subdivided to the final grid size without further classification. The collection of IN cells constitutes the interior mesh of  $S$ .

The main computational utility used for cell classification is the modified cell classification procedure:

$$ModClassCell(cell, solid) = ("IN", "OUT", "NIO")$$

which is described in [15].

$ModClassCell$  tests a cell to determine if it is entirely inside the solid, entirely outside, or undetermined. The "???" cells are further subdivided and tested. Stage 1 ends with special operations that reclassify final-sized "???" cells as IN, OUT, or NIO. (Some might think that "???" cells must always be NIO, but this is not true for Lee's efficient use of the classification procedure, which assumes a CSG representation of the solid  $S$  [15]. Although CSG implementations can be designed to insure that "???" cells are NIO, and the procedure can be used for solids represented in boundary format, both approaches are computationally expensive.)

Specifically, the vertices of each final "???" cell are classified; if one to three vertices are OUT, the cell is NIO. In cases where all four vertices have the same classification the cell is classified as:

$$\begin{aligned} &\text{if } (Cell \cap^* S = 0) \text{ then "OUT"} \\ &\text{else if } (Cell \cap^* S = Cell) \text{ then "IN"} \\ &\text{else "NIO"} \end{aligned}$$

where  $\cap^*$  is the regularized intersection operator [16]. Methods for performing the tests above are described in [8].

We note that the Shephard-Yerry cell classification procedure [13, 14] is based on in/out tests of cell vertices, with some special operations performed on vertices of cells having uniform vertex classifications. In/out tests on vertices are insufficient because cells containing holes or thin sections might be misclassified.

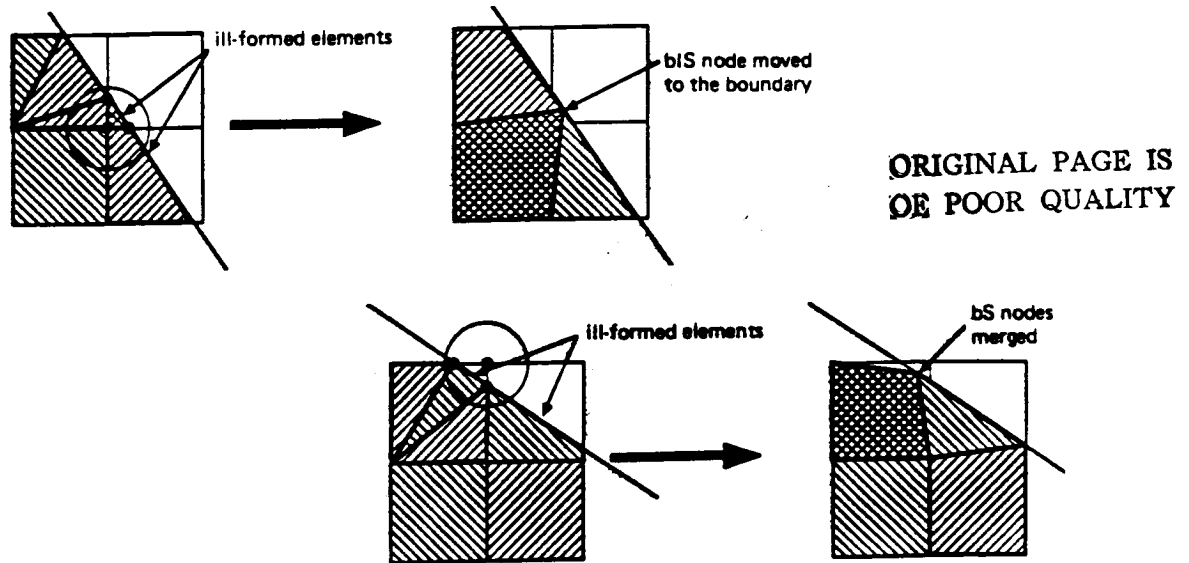
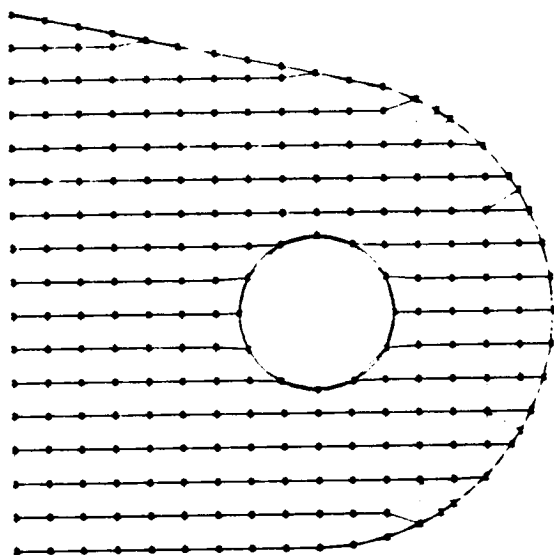


Fig. 9 Node relocation to get well-formed elements.



Stage 2: boundary-region meshing. The task here is to fill the region between the boundary of the interior mesh (denoted  $bIS$  in Figure 7) and the boundary  $bS$  of the solid  $S$ . Observe that:

$$bS \subset (\cup \text{"NIO" cells}) \cup bIS$$

Thus  $bS$  is usually contained in the NIO cells and special element-building operations are required, but sometimes segments of  $bS$  coincide with  $bIS$ , as at the top of Figure 6(b), and no special processing is needed. We can mesh the interboundary region by visiting each NIO cell and creating elements that link the  $bS$  segment passing through it to the interior of the solid.

There are three main issues in this process: to devise a systematic way to insure that all NIO cells are visited, to create nodes on  $bS$ , and to associate  $bS$  nodes with existing  $bIS$  nodes to form valid elements.

All NIO cells can be visited by an exhaustive scan of the

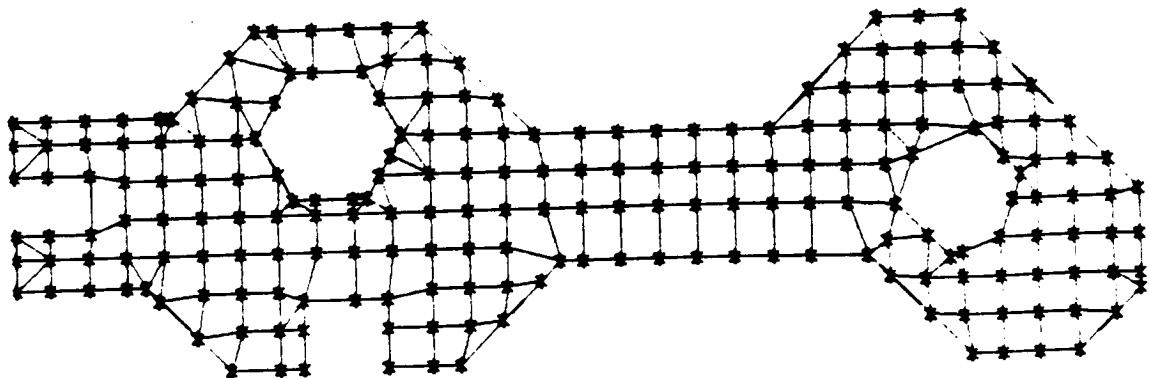


Fig. 10 Examples of automatically generated FE meshes.

ORIGINAL PAGE IS  
OF POOR QUALITY.

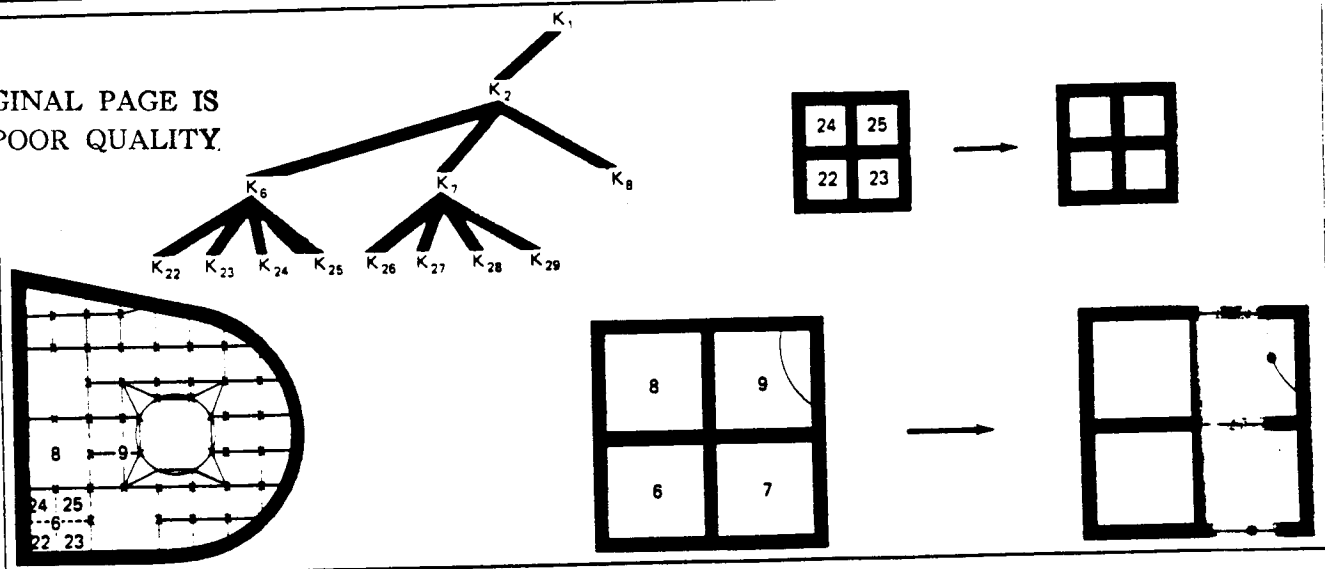


Fig. 11 Assembly via multi-level substructuring.

lowest-level grid, or by tree traversal, or by traversal of  $bS$ . Since no single approach seems to offer substantial advantages we use grid-scan for generating the initial mesh and, because operations tend to be more localized, tree-traversal for remeshing and reanalysis.

Figure 7 shows  $bS$  nodes  $P1$ ,  $P2$ ,  $P3$  that are created in the following manner. Vertices of  $bS$  within each NIO cell (e.g.  $P2$  in Figure 7) are tagged as such and are always used as finite element nodes. The vertices of  $bS$  are available explicitly if  $S$  is represented in boundary format. If only a CSG representation is available, as in our system, a limited form of boundary evaluation [17] must be performed. In 2-D, the CSG primitives that intersect an NIO cell are themselves

intersected to generate candidate  $bS$  vertices; the candidates are then classified to identify true  $bS$  vertices. The analogous 3-D operations amount to constructing a wireframe representation from a CSG representation. Additional  $bS$  nodes are created by intersecting  $bS$  with the boundaries of the NIO cells ( $P1$  and  $P3$  in Figure 7).

The generation of valid elements within an NIO cell is straightforward if the cell does not contain  $bS$  vertices (corner nodes): nodes on  $bS$  and  $bIS$  belonging to the same NIO cell are simply linked to form quadrilateral and triangular elements (see the lower left portion of Figure 8). The treatment is more involved when a corner is present: a detailed explanation is in [8]. Briefly, the corner node is linked to  $bS$  and  $bIS$  nodes within the cell to form a web of triangular elements (Figure 8). To avoid generating elements with poor aspect ratios, the distances between nodes are checked by using a node neighborhood test, and closely spaced nodes are merged into single nodes on  $bS$ . Figure 9 provides two examples of this process.

The FE mesh is complete at the end of stage 2 of the design procedure. A regular mesh of quadrilateral elements in the interior results from a direct mapping of IN cells. On the boundary, NIO cells are associated with quadrilateral and triangular elements. It is important to note that the FE mesh inherits the spatial addressability and structure of the hierarchical grid because elements and substructures are associated with the quadrants of the original decomposition. Figure 10 shows two examples of meshes generated by our automatic procedure.

The Shephard-Yerry (SY) boundary region meshing algorithm performs in/out tests on the midpoints and quarter-points of the edges of NIO cells, and then maps each NIO cell into one of a finite number of cut-quadrant forms: each cut quadrant is then meshed. (We avoid such geometric approximations by computing exact points of intersection on  $bS$ .) The final stages of the SY algorithm move nodes in NIO cells to the boundary, and then eliminate ill-formed elements by using a Lagrangian relaxation procedure to smooth a triangulated version of the entire mesh. This last operation

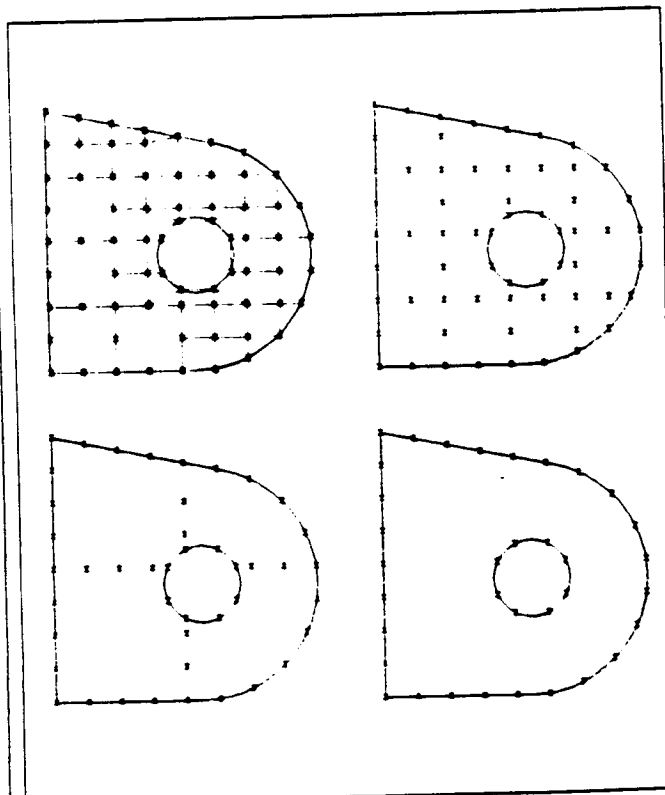


Fig. 12 Substructures at various levels during assembly.

destroys the uniform quadrilateral interior mesh and also spatial addressability, because elements are not constrained to remain in their original cells.

### Analysis Of Hierarchical Meshes

We will now summarize a mesh-analysis procedure that exploits the properties of the hierarchical, spatially addressable meshes already described. Recall that data specifying the finite elements in the initial mesh are accessed through the lowest level of the hierarchical grid; Figure 4(b) shows the types of data that are carried.

One analytical simplification is immediately obvious: because the interior mesh elements are uniform, their stiffness matrices are identical if the material properties are homogeneous and thus only one stiffness matrix need be computed for all of the interior elements. Other more important analytical simplifications accrue during both assembly and solution of the system of equations because the hierarchical grid, which so far has provided spatial substructuring for meshing, can serve also as a multilevel analytical substructuring mechanism.

**Assembly procedure.** Most FE analysis procedures build a single stiffness matrix to cover the whole domain. Our assembler builds and stores stiffness matrices for every non-OUT cell in the hierarchical grid. This is done from the bottom up (see Figure 11) by assembling son matrices and "condensing out" interior degrees-of-freedom to build parent matrices at each level. The parent nodes of the interior mesh with identical (uniform) sons to yield identical substructures and need be assembled only once. The mesh generator tags identical interior-mesh nodes at all levels of the tree to allow this.

Figure 12 shows an initial mesh and substructures at various levels in the assembly process. Note in Figure 12(a) that the initial mesh contains some higher-level substructures; these arise not from assembling lowest-level IN elements, but from intermediate-level cells that were classified as IN and tagged as substructures during stage 1 meshing. (The identical stiffness matrices for lowest-level IN cells are needed in the assembly process only when IN elements must be assembled with elements in NIO cells.)

**Solution.** Figure 13 illustrates various stages in the solution process. After loads and boundary conditions are attached to the root structure, the FE solver computes the displacements of all nodal points on the boundary, i.e., the nodal points of the root substructure as in Figure 13(a), and then traverses down the tree, recovering displacements of substructure nodes at each level.

The displacements at all levels are saved in data records accessed through the hierarchical grid, and the lowest-level displacements are used to compute the stresses in the elements. Figure 14 shows the displacements and average value per element of a stress component. The displacements in Figure 15 are exaggerated for clarity. All analyses here are linear-static, based on linear isoparametric elements. For nonlinear analysis, where displacements can be large, spatial

addressability is still maintained via a backward mapping that associates each displaced element to the original grid.

### Remarks

Our experience with this substructuring approach to analysis leads to some conclusions. The hierarchical grid used for mesh generation has almost all of the data management facilities needed for analytical substructuring. The computing time and storage requirements for internal-element assembly are substantially reduced. We have not yet compared the solution efficiency of our tree-traversal method with that

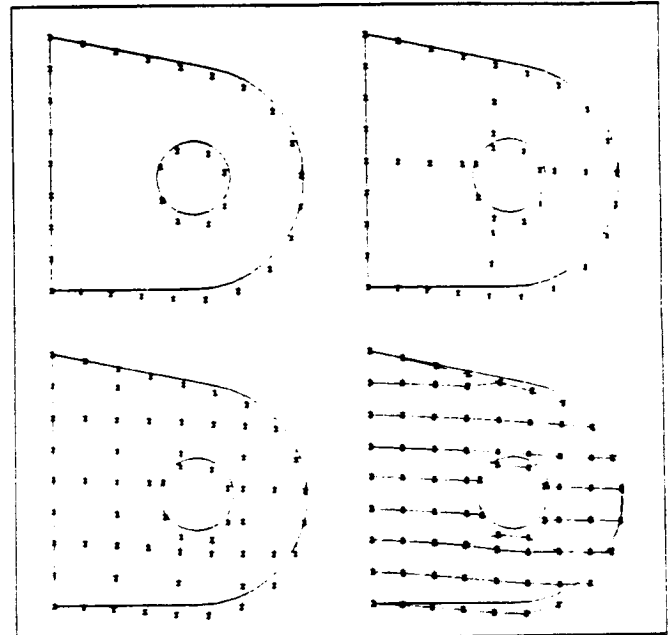


Fig. 13 Nodal displacements at stages of the solution process.

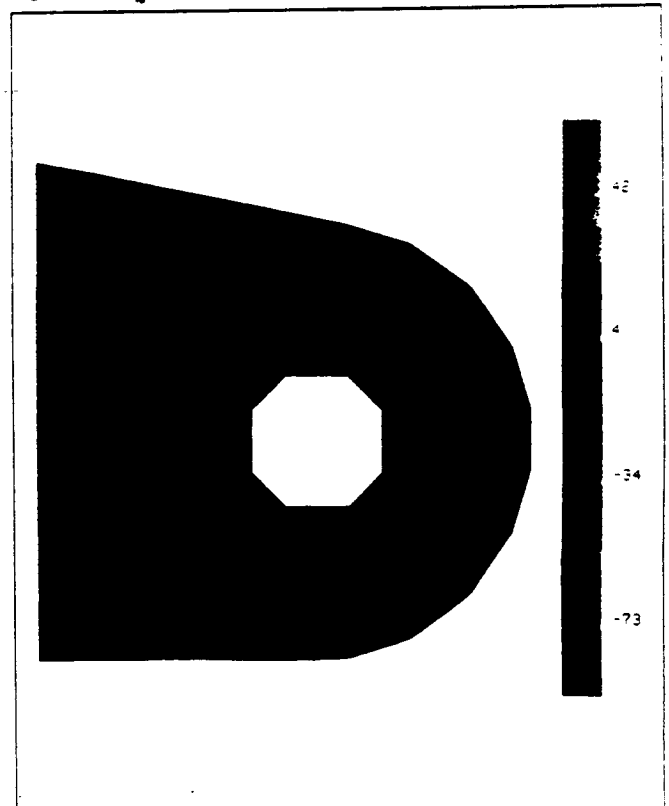
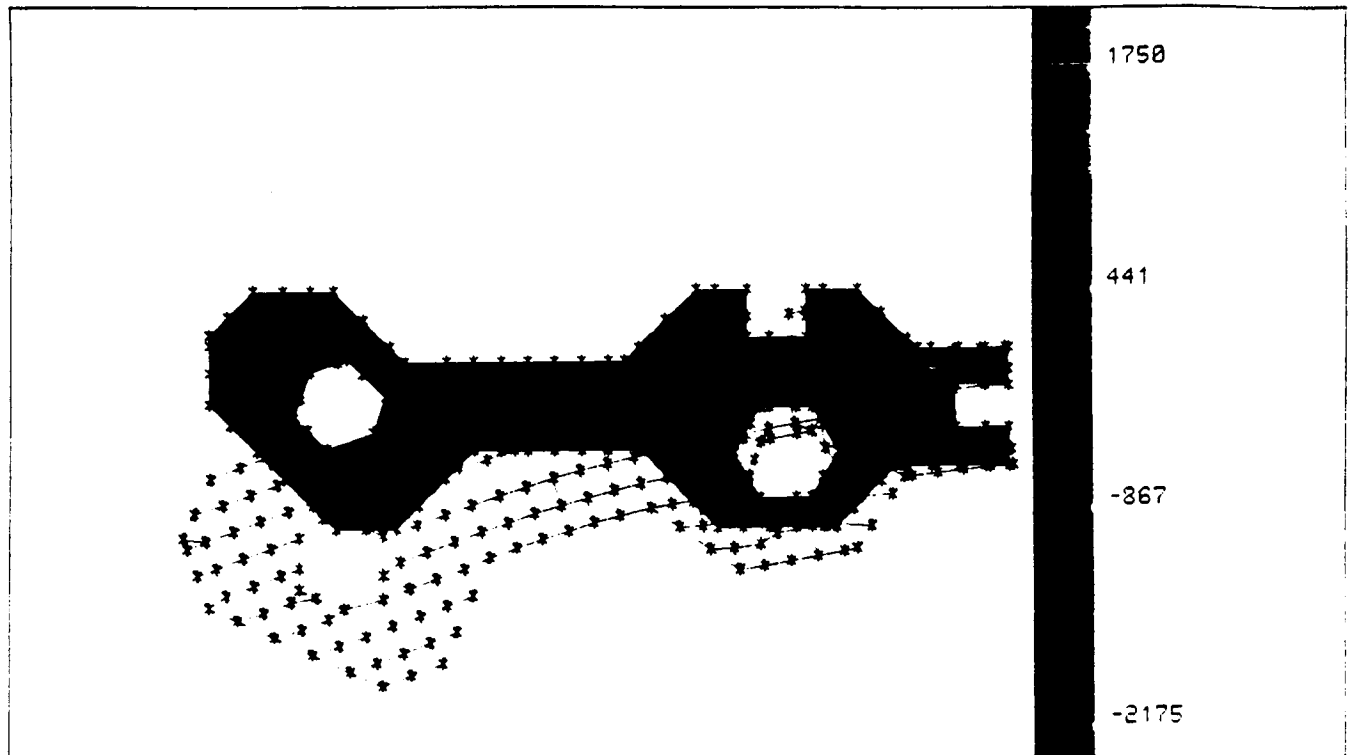


Fig. 14 Average value per element of a stress component.





**Fig. 15 A bicycle spanner in action.** The four nodes on the right-side notch are totally constrained to model engagement with a nut. The average value of a stress component is also shown

of standard solvers, in part because we have made no effort to optimize our code. However, the incremental reanalysis facilities described later clearly outclass standard solvers when it comes to adaptive analysis. Note that solution via tree traversal does not require the normally expensive global element- or node-numbering schemes used by standard solvers to minimize bandwidth or wavefront. Finally, substructuring based on trees lends itself naturally to parallel processing.

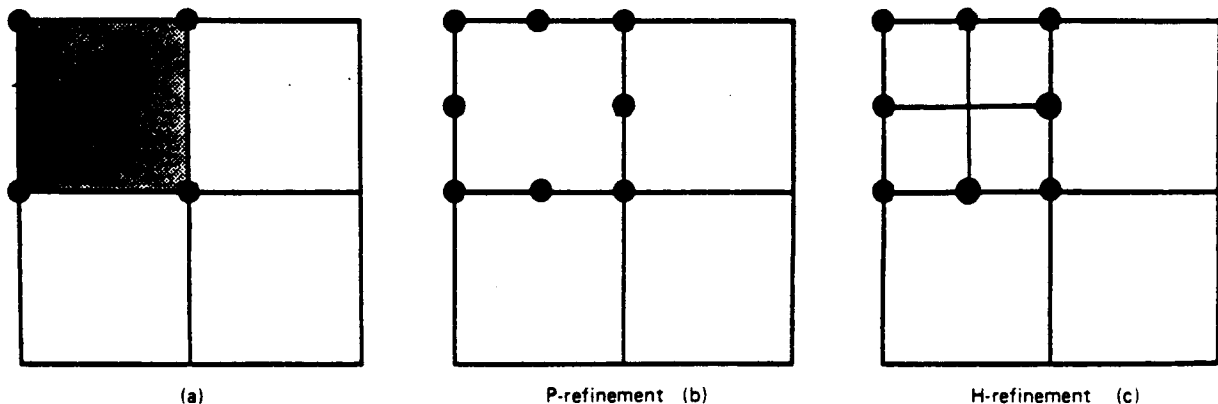
In general, substructuring has proven to be efficient [18] and our particular approach to substructuring seems promising for nonlinear as well as linear analysis. In many practical problems (e.g. contact problems, fracture mechanics, and localized plasticity), nonlinear behavior occurs in isolated regions, and spatially localized analytical methods should prove to be efficient. For example, during analysis, regions that become nonlinear can be tagged in the grid and specially handled. In other types of problems one might want dis-

placements and stresses only in small critical regions, and again spatially localized methods seem very appropriate.

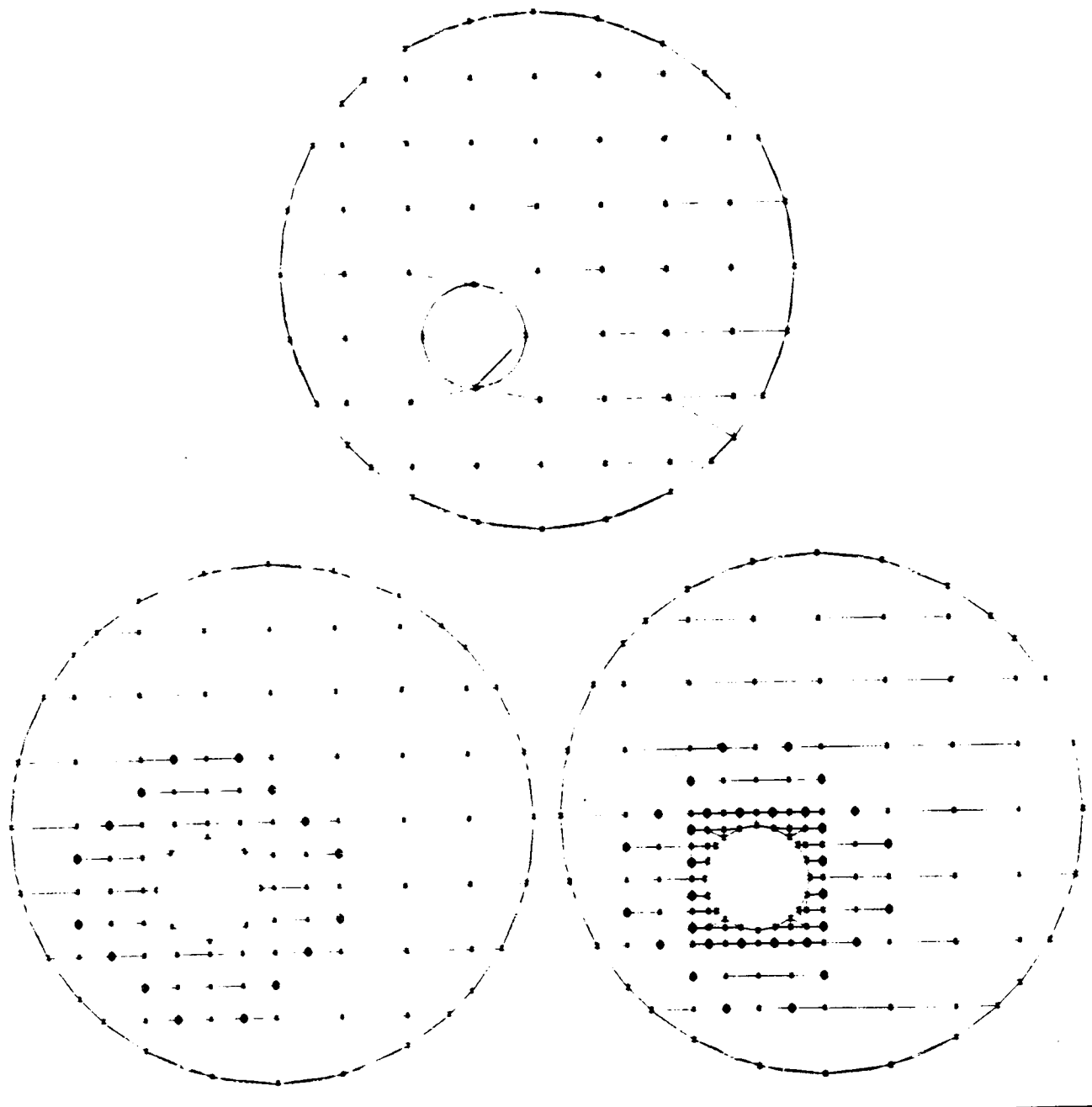
### Self-Adaptive Incremental Analysis

Assume that a mesh has been constructed at the lowest level of the grid; the mesh has been analyzed and the results stored in the grid (e.g. "f" in Figure 4); and evaluation of the results (discussed next) has indicated that refinement is needed in a particular spatial region, say that represented by the mesh fragment in Figure 16(a).

Two avenues for refinement are available, h-refinement and p-refinement. In p-refinement, illustrated in Figure 16(b), successively higher-order shape functions are assigned to the element formulation. To refine a particular element, the old stiffness matrix for the element is invalidated and a new matrix is computed from the new shape function. No new tree nodes are generated, but the size of the stiffness matrix increases.



**Fig. 16 Schemes for mesh refinement.**



**Fig. 17 Two stages of h-refinement.**

In h-refinement existing elements are subdivided into smaller elements of the same type, as in Figure 16(c). To improve the geometric accuracy, localized h-refinement is done on the original geometric model rather than on the current finite element approximation. Thus, to refine a particular element, one deletes the element, creates and classifies new vertices and nodes, and inserts the smaller new elements into the grid. Discontinuities of displacements along edges where smaller elements abut on larger elements are avoided by using constraint equations. These are indicated by the circled nodes in Figure 16(c).

Figure 17 shows examples of localized refinement. Note that successive h-refinements improve the geometric approximation of the original solid. A maximum cross element

grading ratio of 2:1 is maintained during refinement.

Storage for the new entities created by h-refinement could be provided by adding a whole new bottom layer to the grid, but this would be wasteful unless very extensive h-refinement is needed. If the h-refinements are sparse, small localized explicit schemes or linked-list methods are more efficient.

Now assume that the original mesh has been refined in a few regions using the methods just described, that the affected elements have been tagged, and that the refined mesh is to be reanalyzed. Clearly one wants to do incremental analysis, i.e., to use partial results from the earlier analysis as much as possible. These results are available through the hierarchical grid, for example, using a tree of K

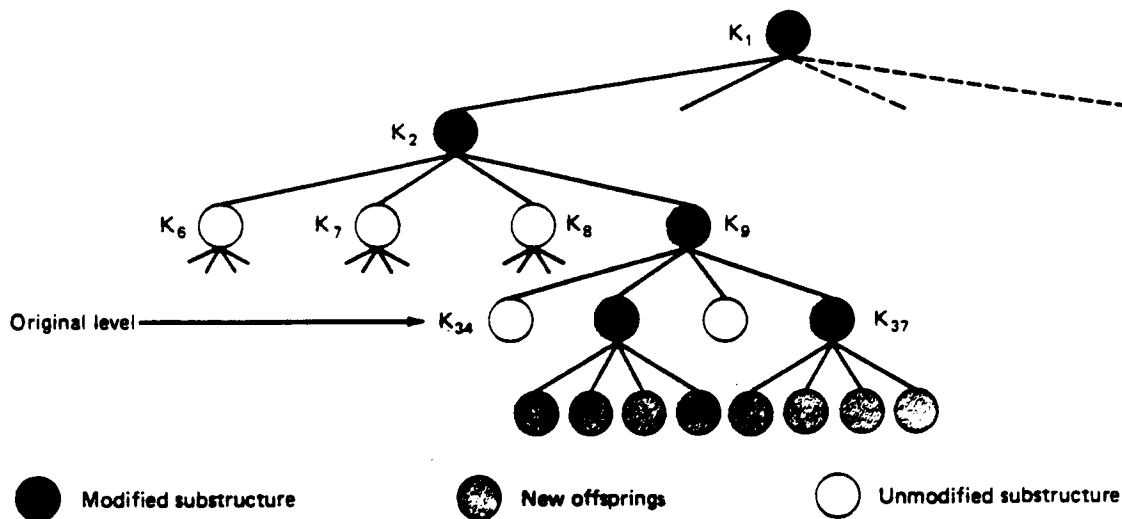


Fig. 18 Incremental reassembly.

matrices as in Figures 11 and 18.

The incremental FE assembler (Figure 1) traverses the tree and by examining the sons of each parent node, detects new offspring and computes the appropriate stiffness matrices (Figure 18). Stiffnesses for unmodified elements are recovered from storage, and new and old stiffnesses are combined to form a modified substructure. If a node has no new offspring, the complete old substructure is reused. The incremental solver (Figure 1) works similarly, inspecting tags on data to distinguish valid and invalid old results and reusing the former whenever possible.

**Self-adaptive algorithm.** Our current algorithm for controlling self-adaptive incremental analysis operates as follows (see Figure 10). After a mesh (either initial or refined) has been analyzed, error indicators are computed for each element together with an estimate of the global error. If the global error exceeds a specified limit, the system calls for refinement and reanalysis in regions having large local errors. This process continues automatically until the global error estimate falls below the specified limit. This rather simplistic control strategy seems to work in the cases we have tested, but it is crude and some needed improvements will be noted.

Considerable research has been conducted on the sources and nature of errors in FE analysis, and on their relationship to mesh refinement schemes [3-7]. Research pertinent to p-refinement has yielded significant results, whereas results on h-refinement have been based mainly on 1-D studies and are fairly primitive.

Thus far we have done little research on errors and our current error measures are crude. As in [5], our element error indicator ( $\epsilon_i$ ) is merely the average of the stress jumps ( $J_s$ , normal and tangential) across each element's edges with dimension ( $h$ ) and assuming linear isoparametric elements:

$$\epsilon_i^2 = \frac{1-\nu}{E} \frac{h}{24} \int_{\tau_i} J_s^2 d\tau$$

normalized by the strain energy of the displaced model. Our

global error estimator is simply the sum of the element error indicators. Figure 19 shows the computed values of the element error indicators for a sample problem (a plate with a hole under traction). Note that, in the vicinity of the hole, the data imply high stress gradients because the error indicators are high. Figure 19(b) shows an automatic refinement resulting from this set of error indicators.

An improvement of the current algorithm would be to replace the single global error indicator, which now serves as a simple refine/don't refine switch, with a hierarchical series of regional error indicators. These can be computed bottom-up in the tree, and should force selective refinement in cases where the overall average error is small but errors in small regions are high. Additional improvements can be expected as more is learned about the nature of errors in FE analysis. Such research should also generate the information needed to study the convergence properties of self-adaptive schemes.

### Advantages and Disadvantages

The main advantage of our approach is that mesh generation and mesh analysis are integrated and in effect collaborate under the control of the error evaluator. Thus, the mesher only refines regions where refinement is needed, and the analyzer only computes "what's new" about a refined mesh. This type of efficient adaptive behavior is, in our opinion, the key to efficient automatic FE analysis.

Some can argue that mesh generation and mesh analysis should not be integrated because integration precludes "mixing and matching", i.e. being able to analyze, through simple interface translators, a mesh from "any" CAD system or preprocessor using "any" popular analysis package. We believe that by the 1990s, however, the benefits of integration will outweigh those of mixing and matching.

Spatially localized substructuring is the driving principle in both the mesh generator and mesh analyzer. This principle derives from recursive spatial subdivision and is manifested in our hierarchical grid and its underlying tree. The tree

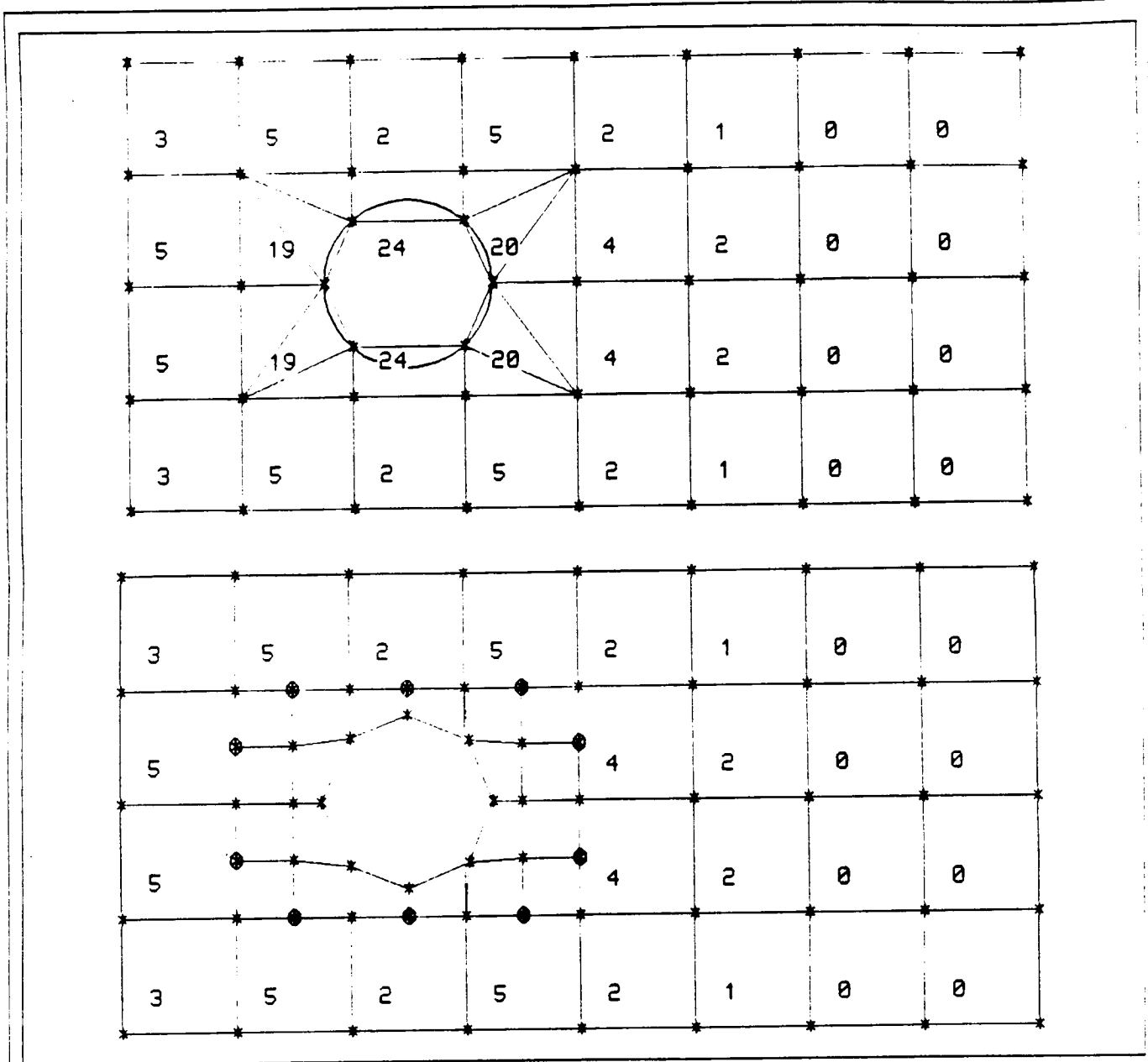


Fig. 19 Refinement driven by error indicator.

might be viewed as a generalization of the structure described in [19]. However, the latter is applied in subdomains that are mapped to regular figures (squares and triangles), and Rheinboldt's tree addresses the element partitioning induced in the regular figures. By avoiding mapping we are able to use the same structure for both meshing and analysis; further, the regularity of our structure permits systematic cell numbering and, hence, data access through calculated addresses rather than through searching or looking in tables.

This "divide-and-conquer" principle enables hard problems (such as object decomposition and equation-set solution) to be decomposed into smaller, tractable problems via spatial partitioning. We note that spatially localized substructuring, and spatial addressability in general, provide powerful mechanisms for coupling FE methods and results to other applications (e.g., manufacturing process modeling) through master data bases based on solid modeling.

Certain technical details already described, such as the

regularity of the interior mesh elements, are also advantages of this approach.

**Limitations.** The main limitation of spatial subdivision methods is that they produce meshes that are dependent on orientation and position if the initial enclosing box is not tight.

This is most easily seen in simple objects that have a single, natural orientation. As such objects are rotated in a fixed set of subdivision axes the induced meshes change, often dramatically. Figure 20 is an example with a simple object meshed in a nonstandard orientation. Skilled analysts call such meshes "unnatural," and note that they usually contain more elements than "hand-made" meshes.

Spatial subdivision can be applied in non-Cartesian domains. For example, predominantly circular 2-D objects can be meshed efficiently in polar coordinates by subdivision of  $(r, \theta)$ . The meshes so produced can be managed through the

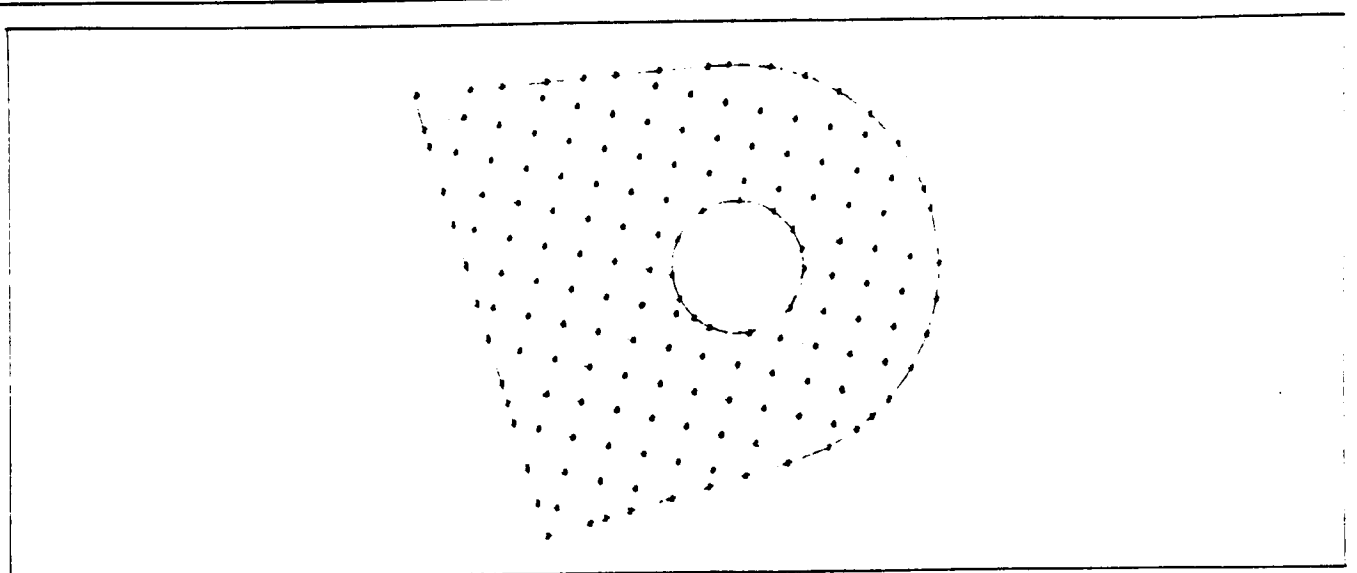


Fig. 20 Orientation and position dependence of meshes derived by spatial subdivision.

same hierarchical grid as is used for Cartesian subdivision [20]. Various schemes have been proposed for mixing subdivision strategies to cater to objects having both circular and rectilinear regions, but none seem promising [20].

The essential counter arguments are that "unnatural" meshes will produce valid results if the elements are valid, and that these results should converge under adaptive remeshing and reanalysis to a single set of (correct) results that is independent of position and orientation. Experimental evidence indicates that our approach exhibits such qualities.

### Still To Be Resolved

Over the long term, four areas will require extensive theoretical work to make truly automatic FE analysis possible:

- *Error measures and indicators.* Better measures than the ones we use currently are needed, but they need not be optimal if adaptive convergence can be guaranteed.
- *Adaptive convergence.* We have seen no experimental evidence of divergence in the self-adaptive process, but automatic analysis systems like ours will require human monitoring to guard against divergence until strong convergence properties can be guaranteed.
- *Computational complexity.* We think that spatial substructuring techniques are asymptotically more efficient than the methods used in current solvers, but we have no results to prove or disprove this. Complexity and convergence analyses, when coupled, should provide bounds on the inherent cost of finite element analysis.
- *Nonlinear analysis.* Thus far we have confined our efforts to linear analysis but our approach to substructuring appears promising for nonlinear analysis as well.

Two other issues are currently more pressing: extending the systems to 3-D problems and handling loads and constraints automatically.

We have done 3-D work in parallel with our 2-D work. An efficient publicly available interior mesher (octree generator)

has been created for solids describable in the PADL-2 solid modeling system [21, 22]. Figure 21 shows an example. The 2-D spatial substructuring techniques for managing analysis, adaptive remeshing, and reanalysis extend gracefully to 3-D, and indeed most of the 2-D control code is directly usable in 3-D. The major unresolved problems are in stage 2 of the automatic meshing procedure, i.e., in the handling of NIO cells. Promising methods for resolving these problems are being studied.

The handling of loads and constraints is the only aspect of 2-D linear FE analysis that we have not yet automated. At present, loads and constraints are applied manually when the assembler has completed its initial pass and the solver is about to begin its initial pass, i.e., at the transition between Figures 12(d) and 13(a). This raises two different questions.

First, there are no fundamental barriers to automating the application of loads and constraints at this stage of the solution procedure. The problems are strictly of an engineering nature. Essentially, what mechanisms should be provided in a solid modeler to support the declaration of loads and constraints (see Figure 1), and how should declarations be translated into mesh-node vector values? The translation problem is straightforward given a good solution to the declaration problem, and an experimental system with enough power to handle load and constraint declarations is already running under 3-D PADL-2 [23].

The second question is deeper. Should loads and constraints be applied at the outset, where they will influence construction of the initial mesh, rather than after an initial mesh has been built? This is certainly the case when meshes are constructed manually, and part of the analyst's skill is in knowing how fine a mesh should be in a loaded or constrained region. Should our mesher be modified to mimic this skill? The only possible gain we see is efficiency and this might be marginal because the current system already refines meshes automatically to reflect loads and constraints but only after it has passed from initial mesh analysis to adaptive remeshing and reanalysis.

In conclusion, we believe that the experimental system

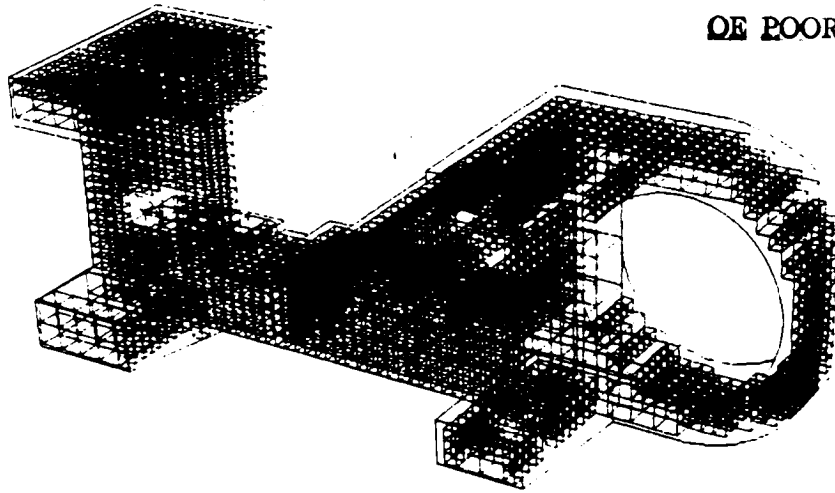


Fig. 21 Automatically derived octree decomposition of "Gehause" (a standard benchmark part for solid modeling systems). Here only the IN octree cells are displayed

described here and its underlying principles represent a milestone on the road to truly automatic finite element analysis. ■

### Acknowledgments

John Goldak of Carleton University contributed to this research and to the education of its authors. Victor Genberg of Eastman Kodak Company provided advice and encouragement. The plots were produced on equipment donated by Tektronix, Inc. Other industrial associate companies of the Production Automation Project provided both equipment and funds. Sustaining support was provided by the National Science Foundation under grants ECS-8104646 and DMC-8403882. The findings and opinions expressed here do not reflect the views of the sponsors.

### References

- 1 Requicha, A.A.G. and Voelcker, H.B., "Solid Modeling: A Historical Summary and Contemporary Assessment," *IEEE Computer Graphics and Applications*, Vol. 2, No. 2, pp. 9-24, March 1982.
- 2 Requicha, A.A.G. and Voelcker, H.B., "Solid Modeling: Current Status and Research Directions," *IEEE Computer Graphics and Applications*, Vol. 3, No. 7, pp. 25-37, Oct. 1983.
- 3 Babuska, I. and Rheinboldt, W.C., "A-posteriori Error Estimates for the Finite Element Method," *International Journal For Numerical Methods In Engineering*, Vol. 112, pp. 1597-1615, 1978.
- 4 Peano, A.G., Pasini, A., Riccioni, R., and Sardella, L., "Adaptive Approximation in Finite Element Structural Analysis," *Computers and Structures*, Vol. 10, pp. 332-342, 1979.
- 5 Kelly, D.W., Gago, J.P., Zienkiewicz, O.C., and Babuska, I., "A Posteriori Error Analysis and Adaptive Processes in the Finite Element Method: Part I. Error Analysis," *International Journal For Numerical Methods In Engineering*, Vol. 19, pp. 1593-1619, 1983.
- 6 Gago, J.P., Kelly, D.W., Zienkiewicz, O.C. and Babuska, I., "A Posteriori Error Analysis and Adaptive Processes in the Finite Element Method: Part II. Adaptive Mesh Refinement," *International Journal For Numerical Methods In Engineering*, Vol. 19, pp. 1621-1656, 1983.
- 7 Zienkiewicz, O.C., Gago, J.P., and Kelly, D.W., "The Hierarchical Concept in Finite Element Analysis," *Computers and Structures*, Vol. 16, No. 1-4, pp. 53-65, 1983.
- 8 Kela, A., "Automatic Finite Element Mesh Generation and Self-Adaptive Incremental Analysis Through Solid Modeling," Dissertation, Production Automation Project, University of Rochester, 1986 (in preparation).
- 9 Wordenweber, B., "Finite Element Mesh Generation," *Computer-Aided Design*, Vol. 16, No. 5, pp. 285-291, Sept. 1984.
- 10 Cavendish, J.C., Field, D.A., and Frey, W.H., "An Approach to Automatic Three-Dimensional Finite Element Mesh Generation," *International Journal For Numerical Methods In Engineering*, Vol. 21, pp. 329-347.
- 11 Lee, Y.T., "Automatic Finite Element Mesh Generation Based On Constructive Solid Geometry," Dissertation, Mechanical Engineering Dept., University of Leeds, England, April 1983.
- 12 Jackins, C.L. and Tanimoto, S. L., "Octrees and Their Use in Representing Three-Dimensional Objects," *Computer Graphics and Image Processing*, Vol. 4, No. 3, pp. 249-270, Nov. 1980.
- 13 Yerry, M. A. and Shephard, M. S., "A Modified Quadtree Approach to Finite Element Mesh Generation," *IEEE Computer Graphics and Applications*, Vol. 3, No. 1, pp. 39-46, Jan./Feb. 1983.
- 14 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.
- 15 Lee, Y.T. and Requicha, A.A.G., "Algorithms for Computing the Volume and Other Integral Properties of Solids: Part II. A Family of Algorithms Based On Representation Conversion and Cellular Approximation," *Communications of the ACM*, Vol. 25, No. 9, pp. 642-650, Sept. 1982.
- 16 Requicha, A.A.G., "Representations for Rigid Solids: Theory, Methods, and Systems," *ACM Computing Surveys*, Vol. 12, No. 4, Dec. 1980.
- 17 Requicha, A.A.G. and Voelcker, H.B., "Boolean Operations in Solid Modeling: Boundary Evaluation and Merging Algorithms," *Proceedings of the IEEE*, Vol. 73, No. 1, pp. 30-44, Jan. 1985.
- 18 Dodds Jr., R. H. and Lopez, L.A., "Substructuring in Linear and Nonlinear Analysis," *International Journal For Numerical Methods In Engineering*, Vol. 15, pp. 583-597, 1980.
- 19 Rheinboldt, W.O. and Mesztenyi, C.K., "On a Data Structure for Adaptive Finite Element Mesh Refinements," *ACM Transactions On Mathematical Software*, Vol. 6, No. 2, pp. 166-187, June 1980.
- 20 Kela, A., "Approaches to Automatic Finite Element Mesh Generation From CSG Representations of Solids," *ITM No. 43*, Production Automation Project, University of Rochester, July 1983.
- 21 Hartquist, E. E., "Public PADL-2," *IEEE Computer Graphics and Applications*, Vol. 3, No. 7, pp. 30-31, Oct. 1983.
- 22 Kela, A., "Programmer's Guide to the PADL-2 Octree Processor Output System," *Input/Output Group Memo No. 15*, Production Automation Project, University of Rochester, Jan. 1984.
- 23 Requicha, A. A. G. and Chan, S. C., "Representation of Geometric Features, Tolerances, and Attributes in Solid Modelers Based On Constructive Geometry," *ITM No. 48*, Production Automation Project, University of Rochester, Oct. 1985.

N88-19121 |

510-61

125796

268

WORKSHOP ON  
THE INTEGRATION OF FINITE ELEMENT MODELING  
WITH  
GEOMETRIC MODELING  
12 MAY 1987

**FINITE OCTREE MESHING  
THROUGH  
TOPOLOGICALLY DRIVEN  
GEOMETRIC OPERATORS**

Kurt R. Grice

Center for Interactive Computer Graphics  
Rensselaer Polytechnic Institute  
Troy, New York

# **OCTREE TECHNIQUE**

## **HIERARCHIC STRUCTURE**

- PROVIDES POWERFUL DATA STRUCTURE

## **SPATIALLY ADDRESSABLE**

- REGULAR HEXAHEDRA (PARALLELEPIPED)

## **FINITE INFORMATION**

- DISCRETE PORTION OF THE MODEL

# **FINITE OCTREE**



# **FINITE OCTREE - OVERVIEW**

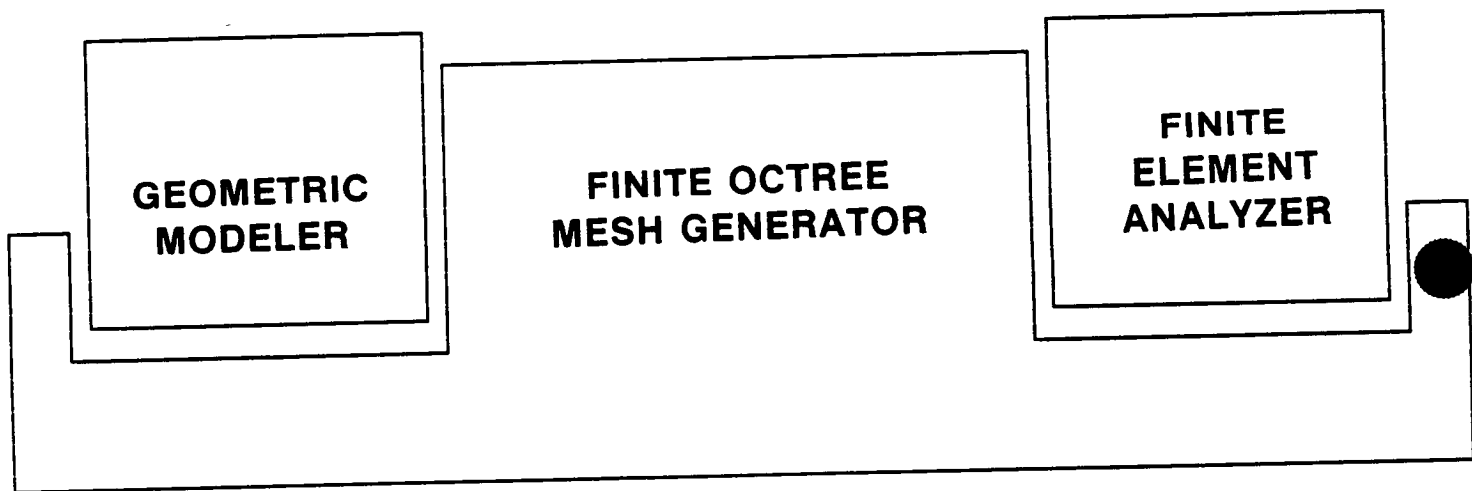
## **DISCRETIZATION OF SPACE**

- EACH TERMINAL CELL (OCTANT) CONTAINS SPECIFIC DISCRETE MODEL INFORMATION.
- THE DISCRETE INFORMATION IS TOPOLOGICALLY CORRECT, BUT GEOMETRICALLY INCOMPLETE.
- EACH DISCRETE ENTITY CONTAINS POINTERS BACK TO THE MODEL, SO ALL GEOMETRIC AMBIGUITIES CAN BE RESOLVED.

**THESE TERMINAL OCTANTS ARE FURTHER BROKEN UP INTO ELEMENTS.**

**THE ELEMENTS ARE THEN SUBMITTED TO AN ANALYSIS PACKAGE.**

**IF NEEDED, TERMINAL OCTANTS CAN BE EITHER RECOMBINED, OR FURTHER SUBDIVIDED IN AN ADAPTIVE TECHNIQUE.**



# FINITE ELEMENT SYSTEM

# **MODELER REQUIREMENTS**

## **BOUNDARY REPRESENTATION -**

- CONTAIN VERTEX, EDGE, FACE AND REGION ENTITIES ALONG WITH THE ADJACENCY INFORMATION.
- ALL COMPLETE AND UNIQUE GEOMETRIC REPRESENTATIONS CAN BE CONVERTED TO A B-REP.
- ANALYSIS ATTRIBUTES ARE DOMINATED BY INFORMATION ASSOCIATED WITH THE BOUNDARY.
- PROVIDES A GENERAL, ABSTRACT MEANS REPRESENTING NON-MANIFOLD STRUCTURE, ORIGINATING PERHAPS FROM AN IDEALIZATION OF THE MODEL

## **GEOMETRIC COMMUNICATION OPERATORS -**

- RESTRICTED SET OF QUERIES ON BOTH THE TOPOLOGICAL ADJACENCY AS WELL AS THE UNDERLYING GEOMETRIC DEFINITION.
- SIMILAR IN APPROACH TO THE CAM-I APPLICATION INTERFACE SPECIFICATION (AIS).
- PROVIDES MEANS OF INTERFACING TO VARIETY OF MODELERS.

# **MODELER REQUIREMENTS**

**EACH TOPOLOGIC ENTITY HAS A CORRESPONDING  
GEOMETRIC ENTITY ASSOCIATED WITH IT.**

- REGION TO VOLUME
- FACE TO SURFACE
- EDGE TO CURVE
- VERTEX TO POINT

**VOLUME, FACE AND EDGE ENTITIES CAN BE  
PARAMETERIZED**

**IDENTIFICATION OF EACH ENTITY IS UNIQUE**

# OCTREE DISCRETIZATION

ONE COULD VIEW THE COMPLETE DISCRETIZATION OF A MODEL AS POINT (OCTANT CORNERS) AND CELL (BOUNDARY INTERSECTIONS WITH OCTANTS) CLASSIFICATIONS.

THIS CLASSIFICATION AND THE ASSOCIATION WITH THE OCTANTS WILL PROVIDE THE DATA FOR GENERATING THE FINAL MESH.

POINT AND CELL CLASSIFICATION TECHNIQUES ARE EXTREMELY GEOMETRY INTENSIVE AND MAY REQUIRE EXTENSIVE QUERIES.

THESE CAPABILITIES MUST BE CAREFULLY IMPLEMENTED FOR USE IN A GENERAL MODELING ENVIRONMENT.

FROM A MODELING STAND POINT:

- IN NON-IMPLICIT REPRESENTATIONS, POINT CLASSIFICATION (IN/OUT/ON TESTING) IS NOT EFFICIENT.

FROM A FINITE OCTREE PERSPECTIVE:

- CLASSIFICATION OF AN 'ON' POINT IS MOST IMPORTANT (DETERMINATION OF A BOUNDARY).
- RESOLVE COMPLICATIONS OF THE MODEL AS EARLY AS POSSIBLE, INCLUDING CONTRIBUTIONS FROM ANALYSIS ATTRIBUTES.
- RESOLUTION OF NON-MANIFOLD REPRESENTATIONS COULD BE VERY COSTLY (ex: hanging faces).
- ONCE A DISCRETE REPRESENTATION OF THE BOUNDARY OF THE MODEL IS COMPLETE, IT IS A TRIVIAL MATTER TO IDENTIFY THE INTERIOR NODES.

# **OCTREE DISCRETIZATION**

## **GENERAL METHOD**

- INSERT TOPOLOGICAL ENTITIES OF THE MODEL FROM THE LOWEST ORDER UP
- VERTEX, EDGE, FACE, THEN INTERIOR (IF ANY)
- UTILIZE SPECIFIC GEOMETRIC COMMUNICATION OPERATORS, AVOID 'EXPENSIVE' OPERATIONS

## **ASSOCIATE THE DISCRETE ENTITIES BACK TO THE MODEL AND THE MODEL TO THE DISCRETE ENTITIES.**

- ALLOWS FOR RESOLUTION OF GEOMETRIC AMBIGUITIES
- ALLOWS FOR THE ASSIGNMENT OF GEOMETRICALLY ASSIGNED LOADS ON TO THE DISCRETE ENTITIES

# **GEOMETRIC COMMUNICATION OPERATORS**

**TWO TYPES CALLED BY THE FINITE OCTREE  
PROGRAM:**

- 8 EXPECT INFORMATION ON TOPOLOGICAL ADJACENCY OR ATTRIBUTES APPLIED TO THE TOPOLOGY.
- 10 EXPECT SPATIAL DATA AS A RESULT OF A COMPUTATION USING THE UNDERLYING GEOMETRY OF THE MODEL.
- ALL ARE TYPICALLY AVAILABLE IN GEOMETRIC MODELERS.

**GOES BEYOND THE STATIC FILE TRANSFER  
SCHEMES SUCH AS IGES, AND INTO A DYNAMIC  
INTERFACE WITH THE MODELER ITSELF.**

# **GEOMETRIC COMMUNICATION OPERATORS RETURNING TOPOLOGICAL ASSOCIATIVITY**

**GET A LIST OF MODEL ENTITIES, SUCH AS  
VERTICES, EDGES, OR FACES FOR INSERTION INTO  
THE TREE.**

**GET THE MESH CONTROL ATTRIBUTE ON THE  
MODEL ENTITIES.**

**GET LOWER ORDER ENTITIES ASSOCIATED WITH A  
SPECIFIED ENTITY. (ex: vertices of on edge)**

**GET HIGHER ORDER ENTITY ASSOCIATED WITH A  
SPECIFIED ENTITY. (ex: regions on either side of a  
face)**

**VERIFY WHETHER AN ENTITY IS ASSOCIATED WITH  
ANOTHER. (ex: edge in face)**

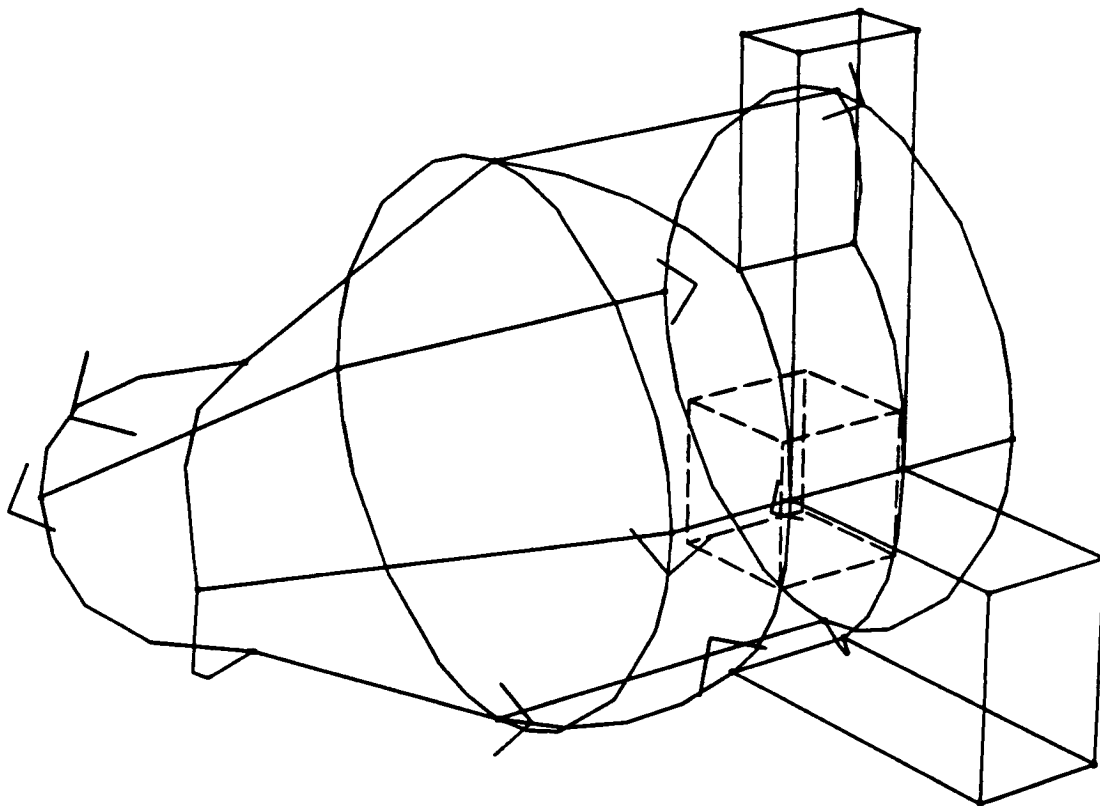


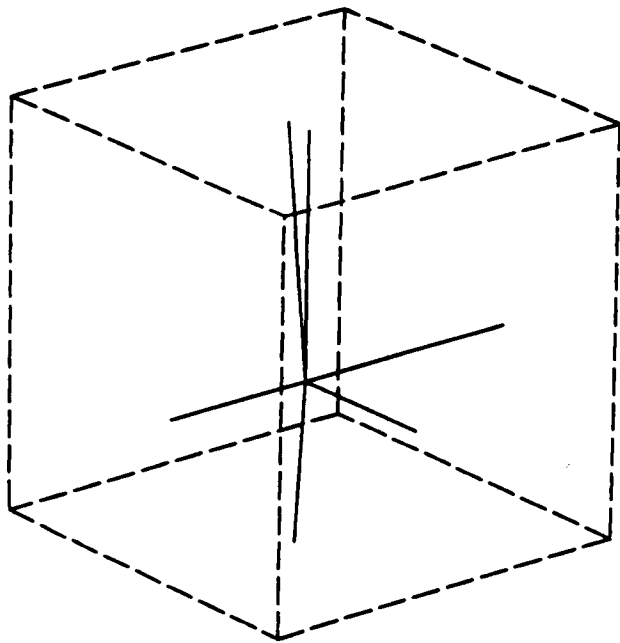
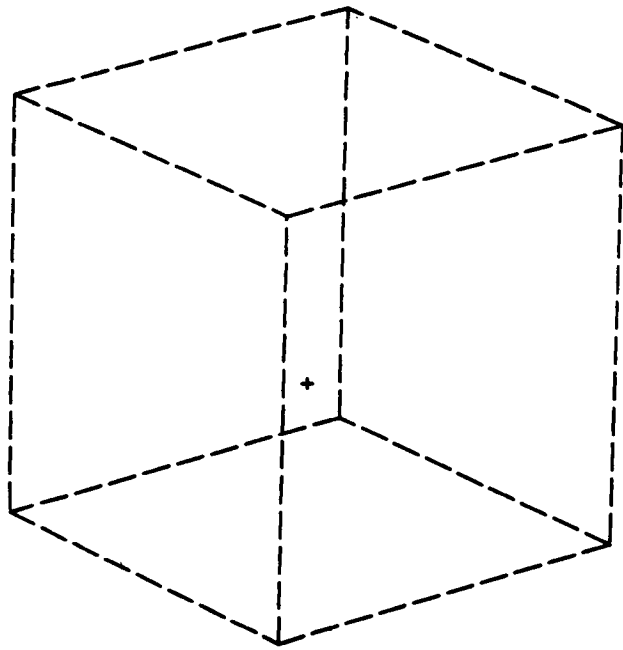
# **GEOMETRIC COMMUNICATION OPERATORS RETURNING SPATIAL DATA**

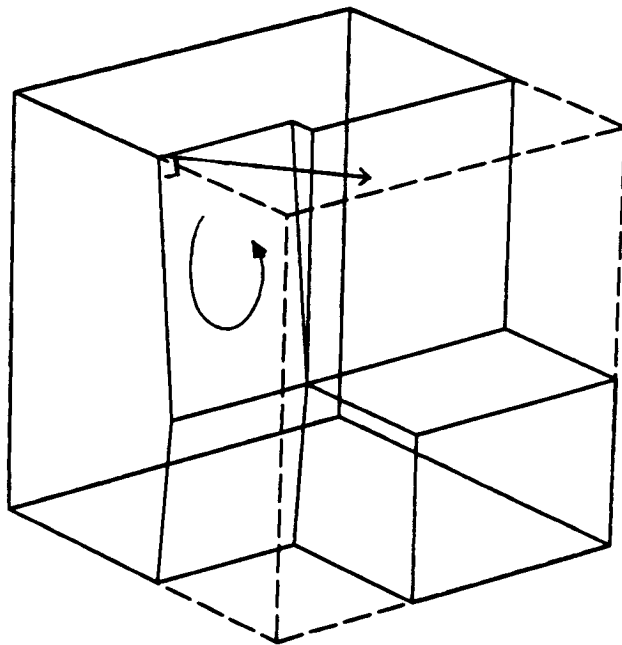
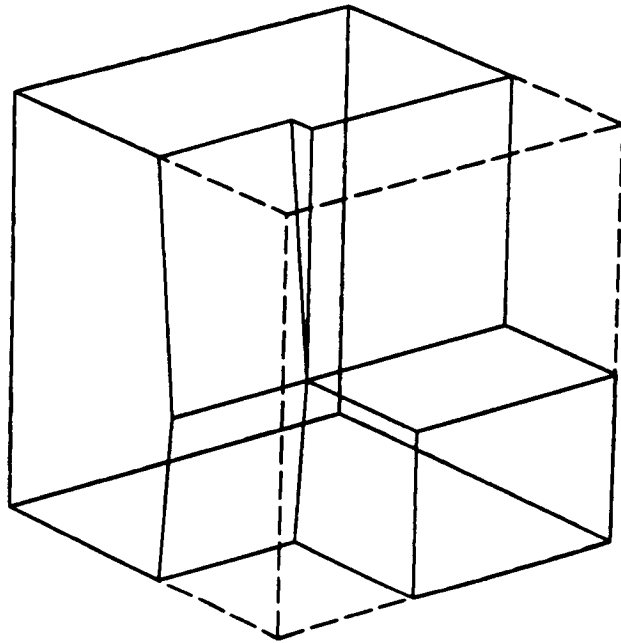
**RETURNED DATA IS ALWAYS BASED ON POINT  
INFORMATION: COORDINATES, PARAMETER  
VALUES, NORMALS, DISTANCES.**

## **EXAMPLES:**

- GET\_COORDINATE\_OF\_VERTEX
- INTERSECT\_PLANE\_WITH\_EDGE
- INTERSECT\_LINE\_WITH\_FACE
- GET\_NORMAL\_TO\_FACE





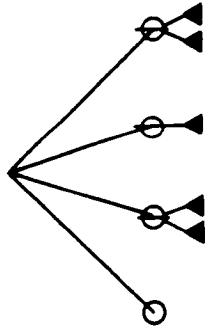
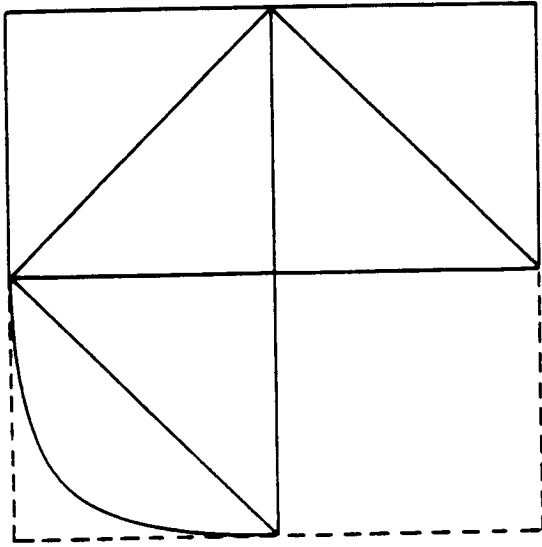


# **CAPABILITIES OF A FINITE OCTREE BASED MESHING PROCEDURE**

ADAPTIVE ANALYSIS TECHNIQUES WITH LOCAL  
REMESHING.

AUTOMATED METAL FORMING USING REMESHING  
CAPABILITIES.

# MODIFIED QUADTREE REFINEMENT

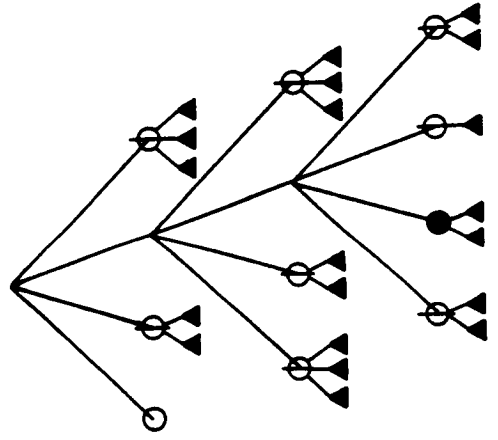
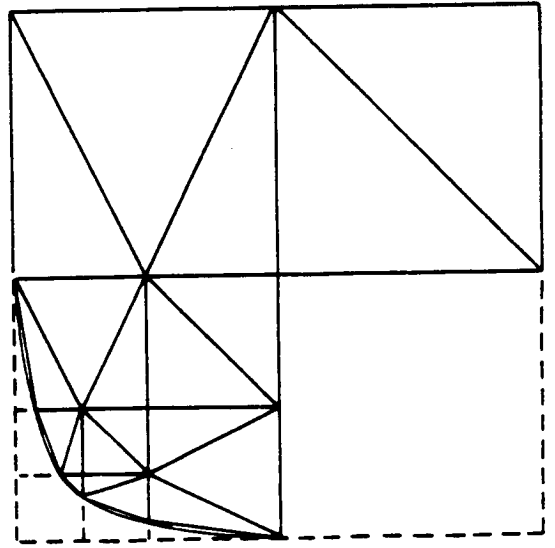


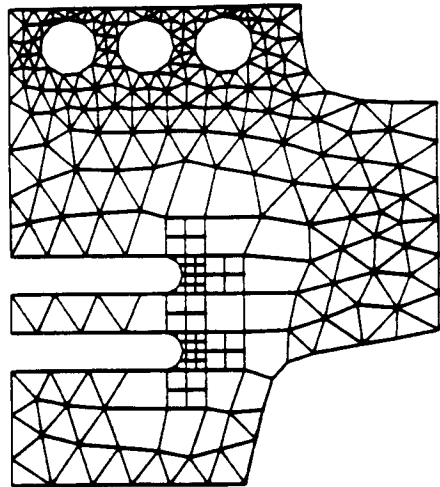
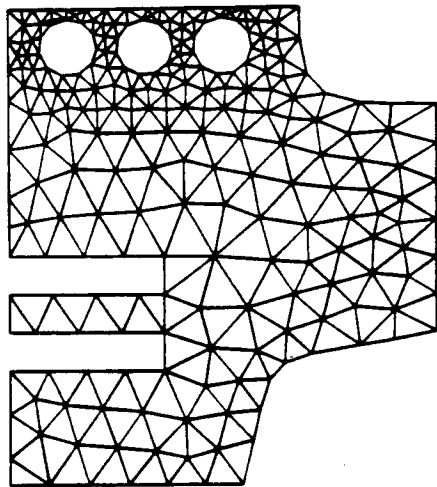
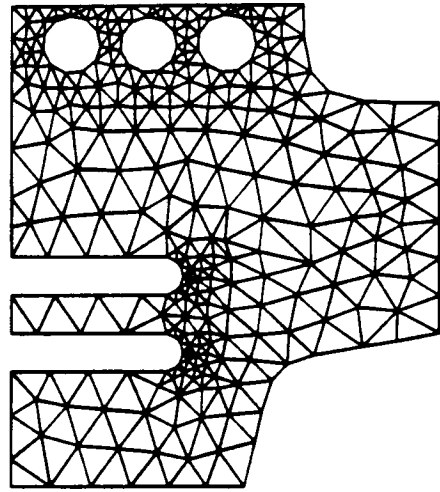
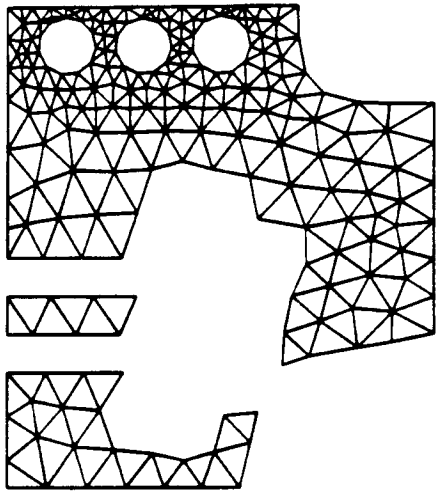
$\phi$  Boundary quadrant

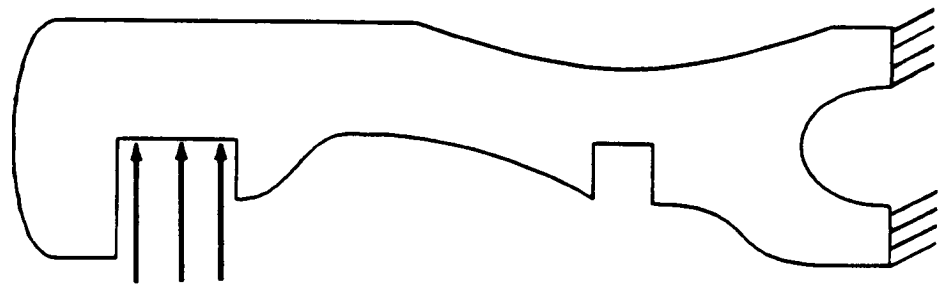
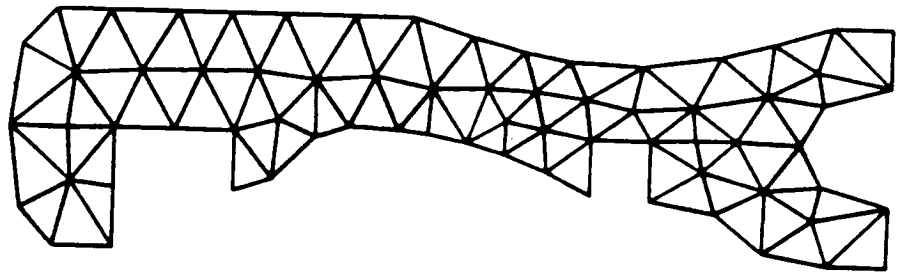
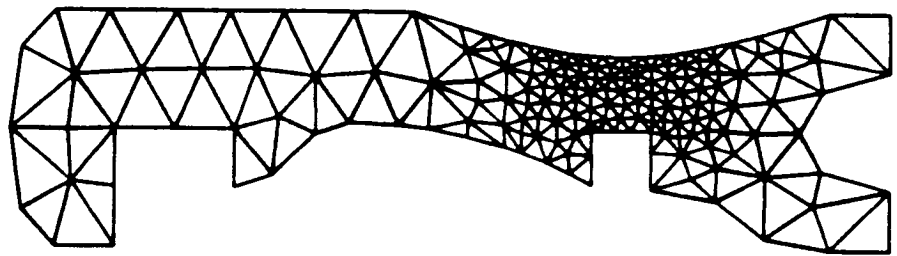
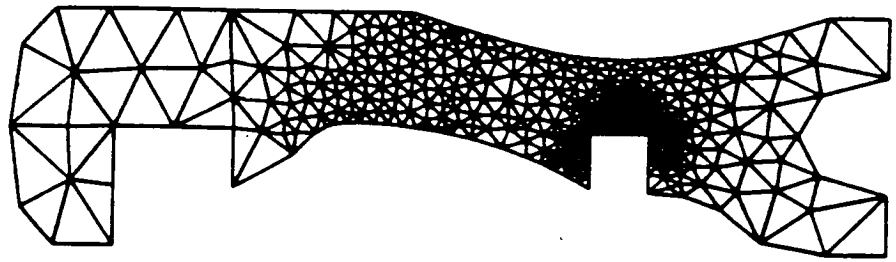
● Interior quadrant

○ Exterior quadrant

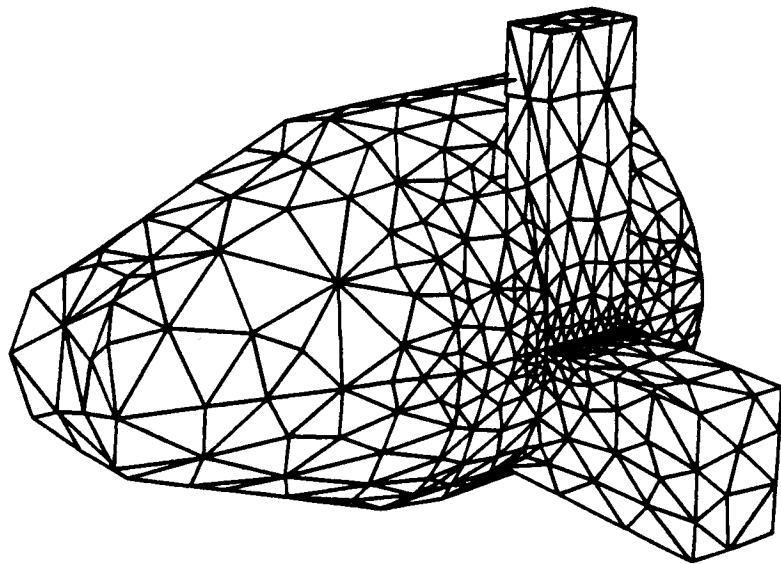
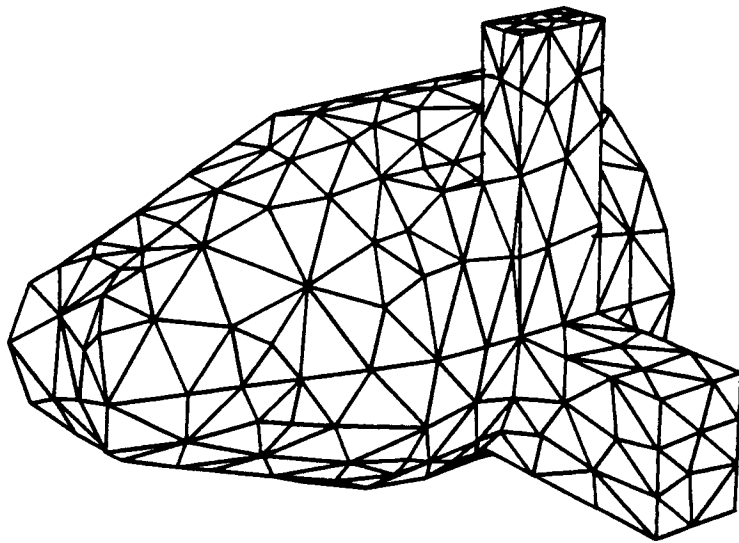
▲ Finite element

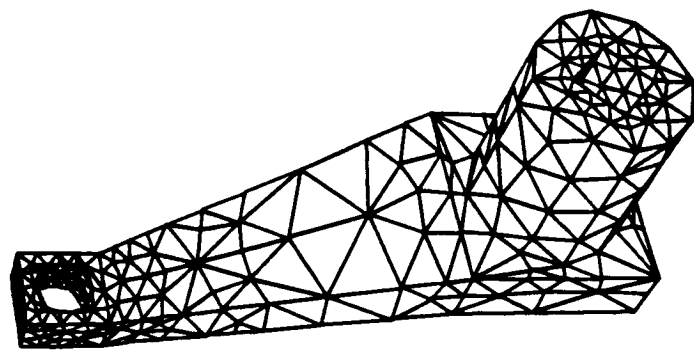
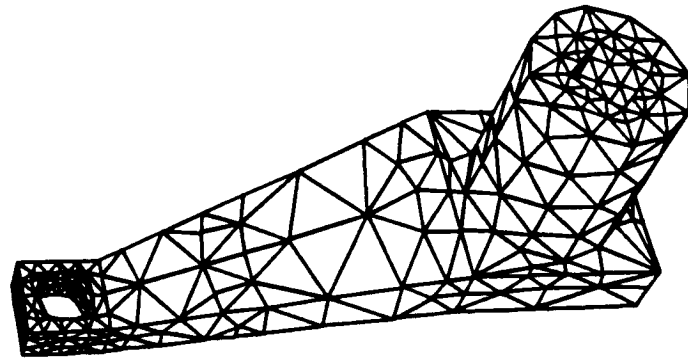












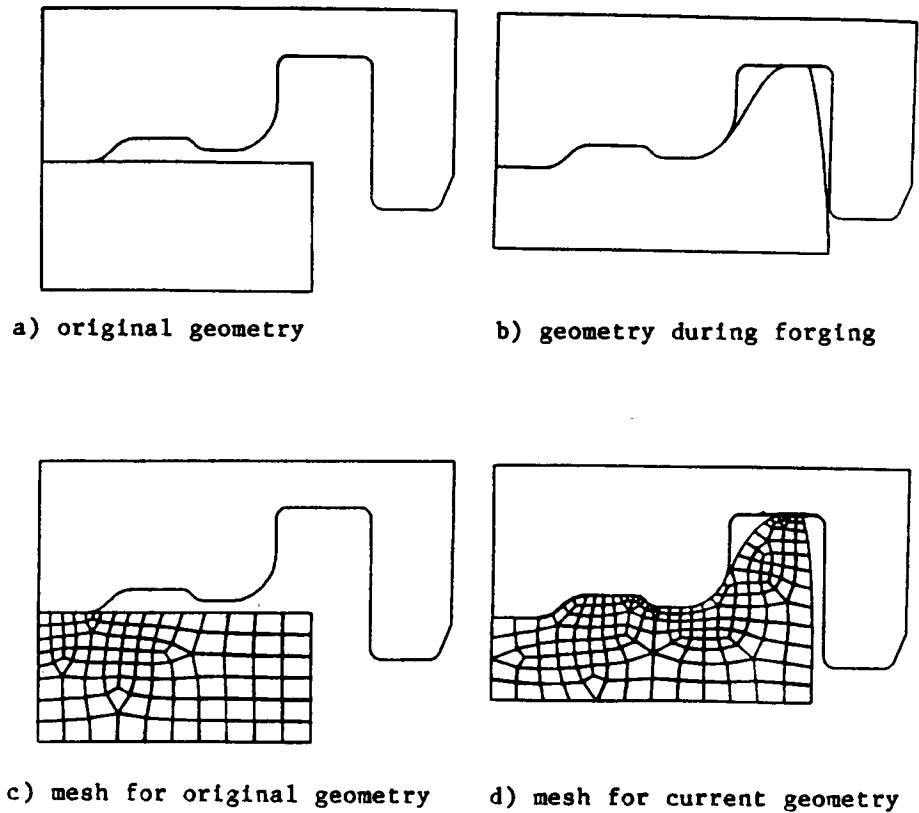


Figure 3. Modeling of forging process

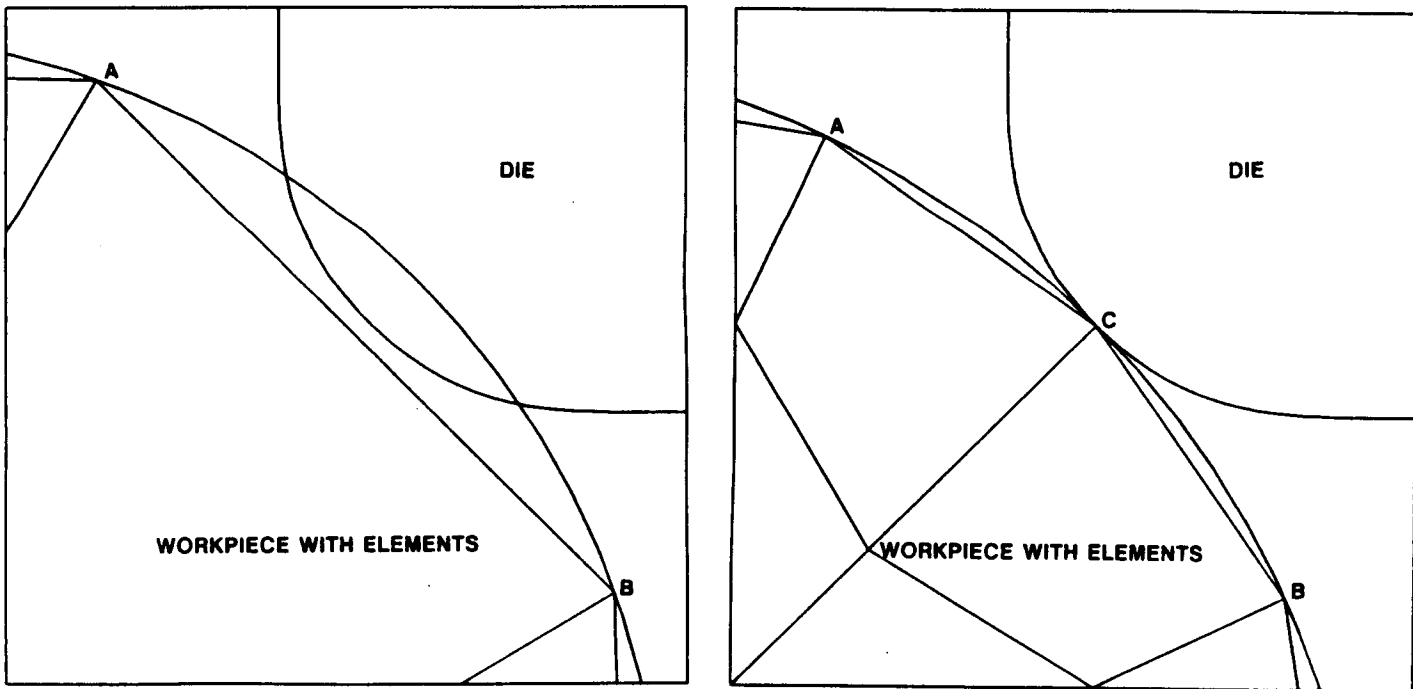


Figure 4. Volume control through geometric checks

# **SUMMARY**

**ADVANTAGES OF BOUNDARY REPRESENTATION**

**ADVANTAGES OF GEOMETRIC COMMUNICATION  
OPERATORS**

**IMPLEMENTATION PLAYS AN IMPORTANT ROLE IN  
THE INTEGRATION WITH A VARIETY OF GEOMETRIC  
MODELERS**

**CAPABILITIES OF CLOSED LOOP PROCESSES  
WITHIN A COMPLETE FINITE ELEMENT SYSTEM**

# **ELEMENT GENERATION**

## **PERFORMED ON AN OCTANT BY OCTANT BASIS**

- EACH OCTANT REPRESENTS ONE OR MORE DISCRETE REGIONS OF THE MODEL, EACH DISCRETE REGION BOUNDED BY DISCRETE FACES
- TOPOLOGICALLY CORRECT, BUT GEOMETRICALLY INCOMPLETE
- GEOMETRIC COMMUNICATION OPERATORS ARE STILL NECESSARY

**THE ELEMENTS ARE CREATED BY BREAKING THE DISCRETE REGION INTO A COLLECTION OF SIMPLEX ELEMENTS (TETRAHEDRONS)**

**CREATING THE ELEMENTS REQUIRES BOTH THE TRIANGULATION OF THE DISCRETE FACES AS WELL AS THE TETRAHEDRONIZATION OF THE DISCRETE REGIONS**

# **ELEMENT GENERATION**

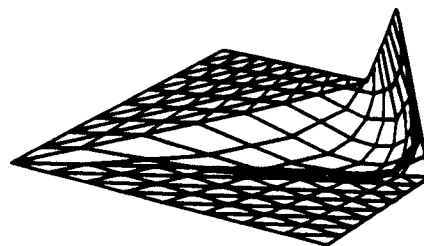
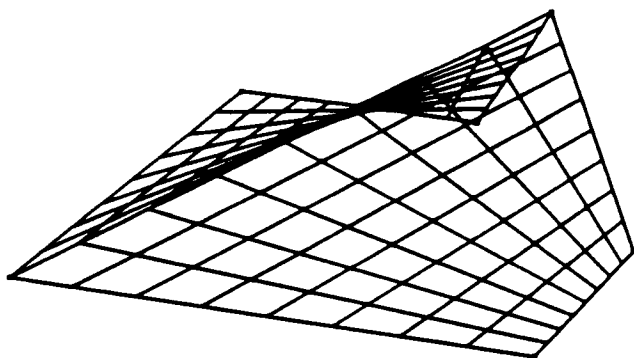
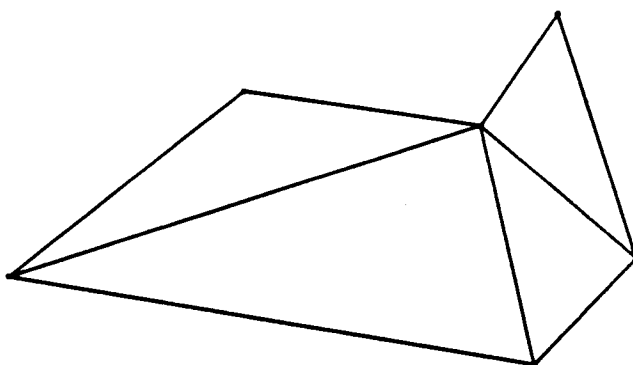
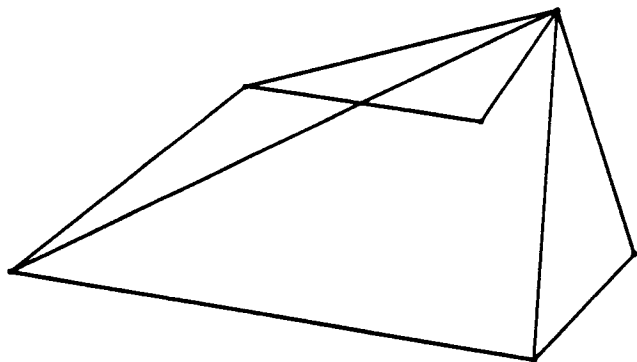
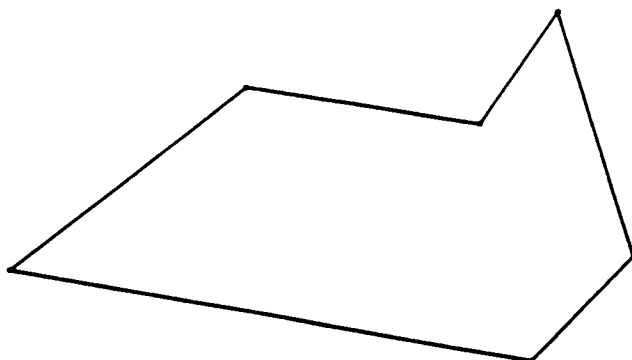
## **FACE TRIANGULATION**

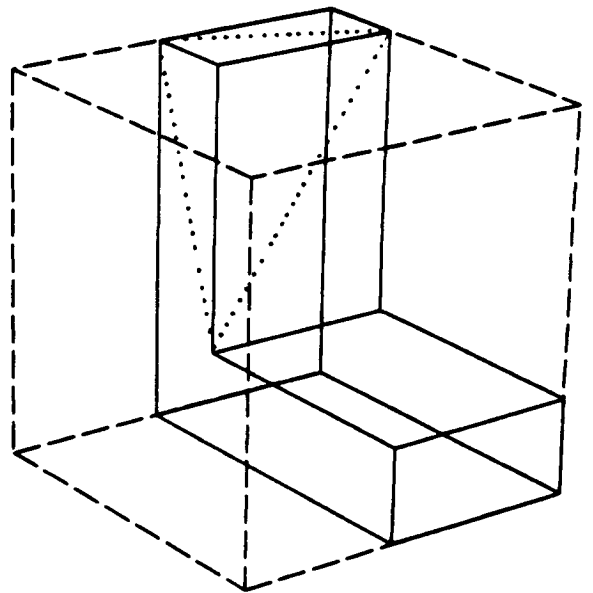
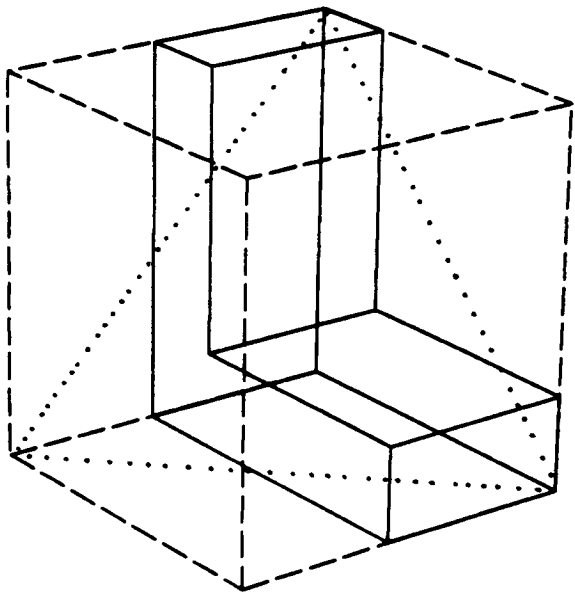
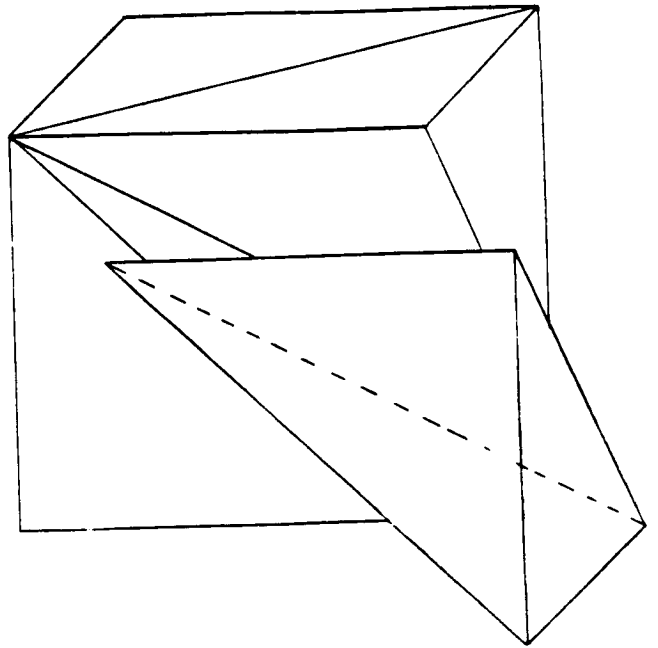
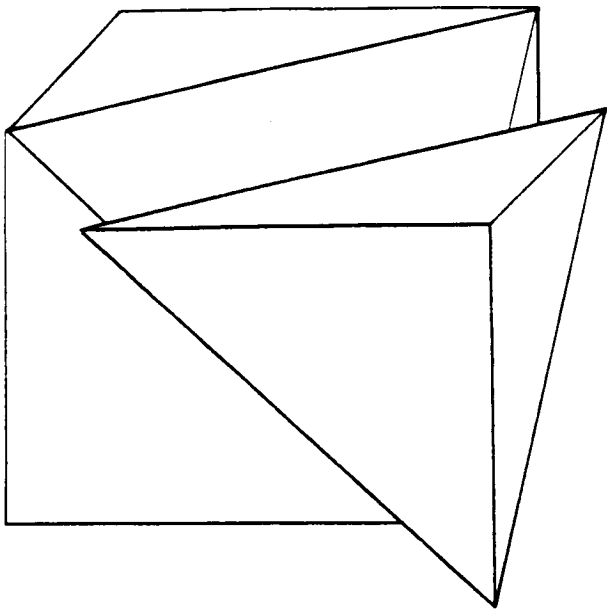
- SINGLE LOOP OF CONNECTED POINTS IN 3-SPACE IS BROKEN INTO A SET OF SIMPLEX ENTITIES (TRIANGLES)
- CRITERIA FOR TRIANGULATION BASED ON VALIDITY AND QUALITY
- NEITHER OF THESE CRITERIA CAN BE RESOLVED BASED ON THE TOPOLOGY OF THE LOOP, THE GEOMETRY OF THE MODEL MUST BE QUERIED

## **REGION TETRAHEDRONIZATION**

- BASED ON THE WORDENBER VOLUME TRIANGULATION TECHNIQUE
- OPERATIONS ARE EDGE REMOVAL AND VERTEX REMOVAL
- EACH REMOVAL MAY CREATE ADDITIONAL ENTITIES THAT MAY INTERFERE WITH THE GEOMETRIC MODEL, CAUSING INVALID ELEMENTS

**IN SHORT, THE TOPOLOGY SUPPLIED BY THE DISCRETE REPRESENTATION, IS SIMPLY NOT SUFFICIENT FOR TETRAHEDRONIZATION, GEOMETRIC QUERIES ASSURE A CORRECT AND APPROPRIATE MESH**







N88-19122 | 5/11-61

125797  
68.

DESIGN MODELING FOR SHAPE OPTIMIZATION

M.E. Botkin  
Engineering Mechanics Department  
General Motors Research Laboratories  
Warren, MI 48090-9057

ABSTRACT

Some important aspects of design modeling for shape optimization will be discussed for both stamped sheet metal components and cast solid components. For stamped components the basis for the modeling approach is a boundary design function. Design parameters control the shape of two-dimensional regions. For more complex, folded plate components, the two-dimensional regions can be assembled using translation and rotation operations. The analysis model is automatically created using a mesh generation procedure requiring only boundary data. For less complex solid components, it was found that this approach is not suitable. For these structures, the finite element models are typically created using very sophisticated graphical modeling systems. A new approach which overlays a parameterized surface design model on an existing analysis model is described. To summarize, the future needs for solid shape design will be described in terms of an extension of the previously described two-dimensional capability.

# Design Modeling for Large-Scale Three-Dimensional Shape Optimization Problems

R. J. Yang and M. J. Fiedler  
Engineering Mechanics Department  
General Motors Research Laboratories

## ABSTRACT

Modeling three-dimensional automotive components for shape optimization is described. Shape optimization differs from sizing optimization in the type of structure, type of design variable, and sensitivity analysis employed. The key element of the shape optimization design model is the parameterization of the geometry by which the optimizer controls the structure dimensions. Efficient generation of the design model is very critical in the design process. A quick generation of a good optimization model combined with an efficient optimization system will result in a drastic design time saving. In this paper, three approaches to generating the design model are discussed. Emphasis will be placed upon a special modeling technique which overlays the design model onto an already existing finite element model. This technique is incorporated in a modular three-dimensional shape optimization system which uses NASTRAN for analysis. A realistic automotive steering control arm is used as an example to demonstrate the use of the technique.

## INTRODUCTION

Optimization techniques have emerged as useful design tools in recent years. Structural optimization for sizing variables has been treated extensively in the literature. The problem of designing the shape of a structure for minimum mass constitutes another important class of optimization problems. Shape optimization differs from sizing optimization in several ways. First, sizing design variables are generally dimensions which do not affect the geometric configuration of the structure, such as cross-sectional dimensions of beam members (thickness, width, height, moment of inertia, etc.). Shape design variables define the geometry of two-dimensional plate and three-dimensional solid structures. As a result, shape design sensitivity analysis is much more complicated. In shape optimization, the boundary of the structure is variable, so parameterization of the geometry is the most important aspect of the shape design model. Modeling for shape optimization is more difficult because both the analysis and design mod-

els must completely describe the structure geometry. The design and analysis models for sizing optimization are inherently loosely coupled because there is little duplication of information. For an existing large analysis model whose surface is not parametrized, generating the design model is not trivial.

In the past, most effort has been put on shape design sensitivity analysis and most problems solved are limited to two-dimensional problems [1-4]. The importance of automatic creation of the design model was seldom found in the literature. Botkin et. al. [5] used computer graphics to generate shape design models for two and three-dimensional stamped structures. Only a limited amount of work has been accomplished in three-dimensional shape optimization using solid finite element analysis [5-7]. Refs. 6 and 7 generated design models manually and as a result, only simple geometries (cantilever beam, engine bearing cap, etc.) were optimized. Ref. 5 used an automatic mesh generator to create the design model and a more complicated engine connecting rod was optimized. However, connecting design variables to the geometry was still done manually. In the real world, three-dimensional problems are often complex and require large finite element analysis models. To be most effective in impacting the design process, the design model must be efficiently generated through an interface to a CAD system.

Many graphics oriented finite element preprocessors are available which can generate very complex finite element models. Unfortunately, these models cannot be used directly for optimization, since they offer no means of parameterizing the shape of the structure. Ideally, for shape optimization, the design and analysis models should be generated simultaneously using a CAD system. An alternative to this approach is an optimization system which generates the analysis model automatically from the design model description of the structure. The major disadvantage to both approaches is that the present state-of-the-art in mesh generation is not of a level where they would be robust enough to function in a real world design environment. However, finite element analysis is an accepted and established part of the design process. For im-

mediate impact, a shape optimization system should be able to take advantage of this fact. Hence, the third approach to design modeling which is presented in this paper is one which utilizes an already existing finite element model as the basis for the geometry description. A parameterization of key dimensions, edges, and surfaces is then overlaid on the finite element mesh.

In this paper, different design modeling approaches are first discussed. A new approach which can handle large-scale problems initially generated as analysis problems only is presented. A steering control arm is used as an example to demonstrate the use of the design modeling approach.

## DESIGN MODELING APPROACHES

When evaluating any modeling approach, the robustness of the technique and the difficulty of integrating the system into the design process are the two major criteria. A robust design model will be general enough to include every possible shape which will satisfy the design constraints. At the same time, the constraints must be flexible enough to eliminate the consideration of any impractical designs from a manufacturing standpoint. It is also important that the coupling between the design and analysis models be of a nature that maintains the integrity of the finite element analysis through the iterations of the optimization process.

Two design modeling approaches were found in the literature: a boundary design element concept, and a design element approach or a generic model approach. The present approach differs from both of these in that they both use some form of mesh generation while our approach uses mesh manipulation.

The boundary design element concept was first proposed by Bennett and Botkin [8] for two-dimensional plates and by Botkin and Bennett [9] for three-dimensional folded plate structures. The basic idea of this approach is to parameterize a boundary segment with several design variables, assemble all segments to form the whole part, and generate a finite element mesh within this boundary. The key to the success of this approach is the availability of a two-dimensional fully automatic mesh generator [10]. With this capability, a more advanced step which considers the accuracy of finite element analysis with mesh refinement was made possible [8]. This approach is probably the most robust and attractive as the creation of the finite element mesh is transparent to the designer. However, the boundary description format cannot be extended to three-dimensional solids because a fully automatic mesh generation [11] which relies on surface data is not developed to the point where it can be routinely used in an automated fashion.

The design element approach for two-dimensional elasticity problems was first used by Botkin [12] and also used by Braibant and Fleury [13], who employed Bezier and B-spline functions for boundary geometry. The design element or generic modeling scheme for three-dimensional shape optimization was used in Refs. 5-7. This approach can be thought of as a volume design element concept. In this approach, the geometry is described by design elements whose key dimensions are associated with the geometric design variables. The finite element mesh for analysis is then generated within each design element by an isoparametric mapping technique. The advantages of this approach are that no discontinuity exists at the

element interface, relatively few design variables are needed, and interior points are automatically adjusted when a boundary moves. The main disadvantage of this method is the relatively inflexible mesh generation scheme. Mesh gradation is completely controlled by the number of generic elements and the mapping technique used. Since the generation technique creates a very uniform mesh, refinement in a local region can only be accomplished by adding more design elements. For complex geometries which cannot be modeled with a coarse mesh, the density of the generic model quickly approaches that of the analysis model. In effect, the designer has to generate a full-scale finite element model anyway. With increased complexity of the mesh, the number of necessary design variables also increases. Although the finite element mesh generation is largely transparent in this approach, the quality of the mesh may not satisfy the designer who is used to generating finite element models with a graphics preprocessor. One other drawback to this method is that the designer will be restricted to using the finite element types permitted by the mesh generator.

## PRESENT APPROACH

In the present approach, the original finite element model is employed as the basis for the design model. There is a one-to-one correspondence between the finite element analysis and design model geometry descriptions. That is, the node numbers and locations for both models are identical. The design model attaches design variables to the node locations stored in the analysis model. As the optimizer changes the design, the analysis model is updated to reflect the change in node coordinates, and the design model is updated to reflect the change in the design variables.

All the additional data needed to describe the shape optimization model is stored in a single DESIGN file. The present model contains two key elements. The first is a list of design variables with upper and lower limits. When the optimization is performed, the design variable vector moves toward the optimal design. The second key element of this model is the type of geometric operators which give these numbers physical significance by relating them to actual part dimensions. This is done by manipulating the coordinates of the nodes which describe the finite element mesh. Three types of operators are included in the design model. LINK functions form the most direct relationship between the design variables and the part geometry. Each LINK function references a design variable or a linear combination of any number of the design variables as specified by the user. This dimension is then used to position a list of dependent nodes relative to some independent reference. The type of reference depends on the type of LINK function specified. For example, if a cylinder function is used, all the dependent nodes are positioned relative to an axis. Unlike LINK functions, POLY and GRID functions do not explicitly reference design variables. Therefore, they allow the designer to minimize the number of design variables necessary to completely describe a problem. Like LINK functions, both of these functions position nodes in a specified list relative to some independent nodes. POLY functions do this by putting a polynomial curve through the independent nodes and interpolating the dependent nodes onto it. GRID functions set chosen coordinates of the dependent nodes to a value determined by a linear combination of the independent node coordinates.

The most time-consuming and tedious part of this approach is locating and identifying the independent and dependent nodes used in the geometric functions. To expedite this process, an interface with a CAD system should be developed. With such a graphical system, the business of determining and attaching the node labels to the geometric functions would be transparent to the user. The designer would have to select the nodes graphically off the screen, while the computer internally stores the appropriate numbers and builds the DESIGN file. A key feature of the shape design modeler, which will be implemented in the future, is the ability to animate the geometric functions. This will allow the designer to instantly see the effect changing an individual design variable has on the part geometry.

The main advantage of this method is that it is applicable at any point in the design process. The designer does not have to sacrifice the time already invested in building the analysis model if he decides to run an optimization. Also, this method has been shown to work on real problems with technology that is currently available.

### MODULAR SYSTEM FOR SHAPE OPTIMIZATION

The design modeling technique described in the previous section is incorporated with a three-dimensional modular shape optimization system which uses MSC/NASTRAN for finite element analysis [5,14]. The system flow chart is shown in Figure 1. Each step is an independently executable module. CONMIN [15] is called as a subroutine from SENSTY. A fifth module (not shown) forms the link between STEP 4 and STEP 1. Termination is controlled by an iteration counter and can occur after STEP 2 or after STEP 4, as specified by the user. Steps 2 and 4 can be run independently to test the design model without running an analysis. In STEP 1, a NASTRAN static analysis is run using superelement formulation. The nodal coordinates, internal/external node label list, and displacements are written to an output file for use in the next steps. In STEP 2 (ADJLOD), stress and displacement constraints are evaluated. For those constraints which are active, adjoint loads are calculated. In STEP 3, each adjoint load is submitted as a separate load case in a restart on the analysis performed in the first step. Displacements from this analysis are written to an output file and used in the next step to calculate sensitivities. In STEP 4 (SENSTY), the gradients of the cost function and active constraints with respect to the design variables are evaluated. This information is fed to CONMIN, which forms Taylor series approximations of these functions and performs an optimization to arrive at the next design iteration. The grid coordinates are updated to reflect the new design variables. Then a Laplacian smoothing operation is carried out on all interior corner grids to minimize element distortion. Finally, the midside grids are linearly interpolated between their respective corner grids, except those on the boundary surfaces. The new coordinates and design variables are written to the NASTRAN and DESIGN files, respectively.

### STEERING CONTROL ARM

The forged steel steering control arm shown in Figure 2 was optimized. The arm is subjected to a single 9000 N steering load applied through a ball stud as shown. Constraints are applied around the strut tube on the upper and lower surfaces of the arm to simulate the welds. The NASTRAN model con-

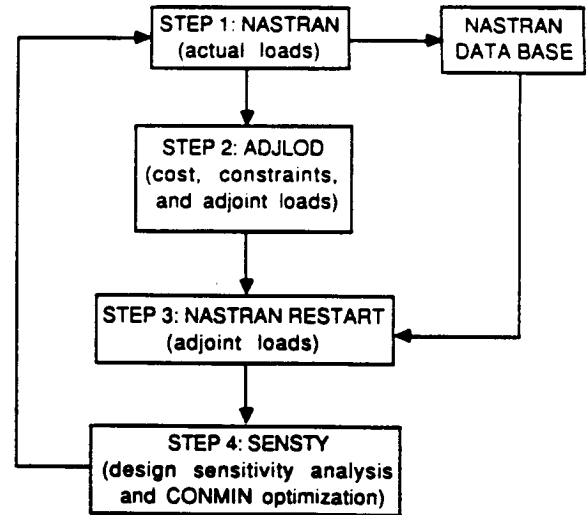


Figure 1. System flow chart

sists of 190 HEXA elements, 6 PENTA elements, and 30 BAR elements (used to model the ball stud). There are 1497 grids in the model which corresponds to roughly 4300 DOF. Young's modulus, Poisson's ratio, and the allowable octahedral shearing stress are  $2.07 \times 10^5 \text{ MPa}$ , 0.3, and  $250 \text{ MPa}$ , respectively. The optimization model shown in Figure 3 uses 12 design variables, 31 link functions, 15 polynomial interpolating functions, and 21 grid link functions. The numbered arrows represent the design variables. The lettered points are key node locations and the dashed lines are movable boundaries. The design variables are described in Table 1. Design variable 5 is actually fixed, but is needed to locate point F. Quadratic interpolation functions are used to generate curves KLM, BCD, and FGH. Cubic Hermite curves AB and DE form smooth transitions between BCD and the outside radii at the ball stud and the strut tube. Only half the model is shown in the XY-plane because it is symmetric about the X-axis. Figure 4 is a partial listing of the DESIGN file for this model. Three geometric constraints have been included to prohibit the inside wall boundary from crossing the outside wall boundary. The initial design is infeasible as the

Table 1. Design Variables Description

Design Variable	Description
1	Floor thickness
2	Strut tube (MN) thickness
3	Midsection (L) thickness
4	Width of inside wall at ball stud (F)
5	Radius of inside wall of ball stud (fixed)
6	Width of inside wall at midsection (G)
7	Radius of inside wall of strut tube
8	Radius of fillet (HJ)
9	Position of fillet radius center
10	Width of outside wall at ball stud (B)
11	Width of outside wall at midsection (C)
12	Width of outside wall at strut tube (D)

part had a high stress value near the inside fillet radius at the strut tube (Figure 2). The peak stress in the part violates the stress constraint by 87.5%. The initial mass is 615.4 g. After 1 design iterations, the stress constraints were met and an 8% weight savings was achieved (final mass of 566.5 g). Table 2 lists the initial and final values of the design variables as well as the limits placed on them. A comparison of the initial and final geometries is given in Figure 5. The design histories of the mass and maximum stress constraint are shown in Figure 6.

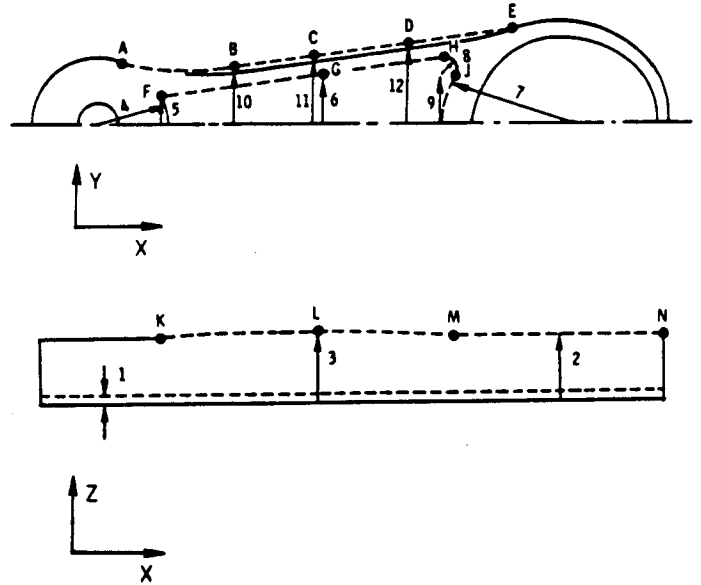


Figure 3. Design model for steering arm

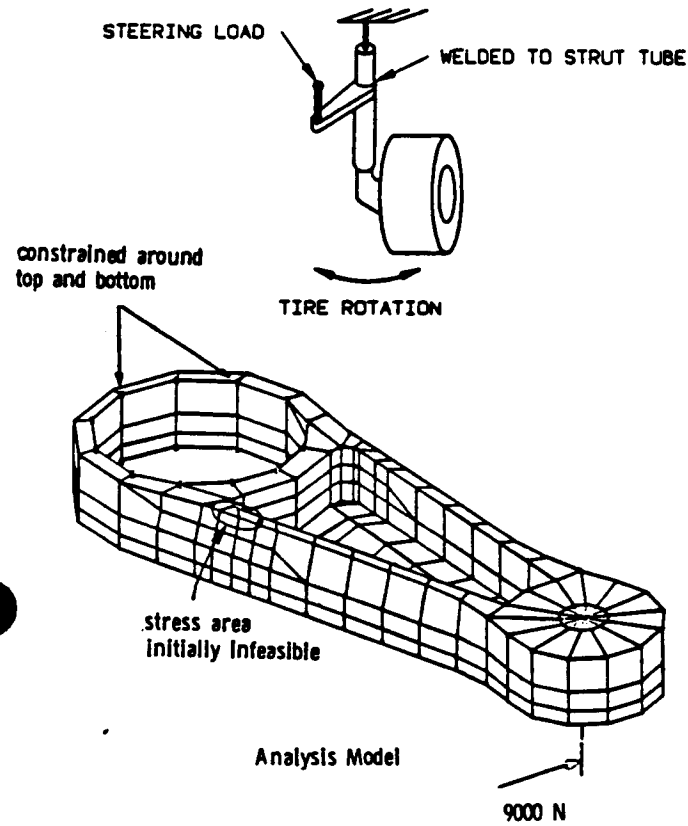


Figure 2. Steering control arm

Table 2. Design Variables for Steering Control Arm

No.	initial	final	lower bound	upper bound
1	4.20	2.50	2.50	10.00
2	20.00	20.00	20.00	40.00
3	20.00	22.12	11.00	40.00
4	8.97	8.64	2.20	17.00
5	20.75	20.75	20.75	20.75
6	15.86	17.95	2.20	26.00
7	34.30	30.46	30.40	40.00
8	4.00	5.72	1.00	10.00
9	15.29	11.30	2.20	18.00
10	18.61	19.00	4.20	19.00
11	21.98	22.26	4.20	30.00
12	26.21	25.46	4.20	29.00

unit: mm

```

$
PARAMETERS      ITER.MCOND / ICHECK.ISTOP.IDEBUG.ISMUTH.NSTR
1 0              1 1 0 1 7

$
OPTIMIZATION PARAM  NACH.IFLERY.ILINC.EPS.PER.BMINLK.THETA
-1 -1 -1 -1 0.10 0.10 0.05 -1.0

$
PROPERTY
2 0684E05 0.3
250

$
GEOMETRIC CONSTRAINTS
2 5
-1.10 1.0 4

$
GEOMETRIC CONSTRAINTS
2 4
-1.11 1.0 6

$
GEOMETRIC CONSTRAINTS
2 5
-1.12 1.0 9

$
STRESS CONSTRAINT REMOVAL
70
73 74 76 66 63 64 65 67 68 69 70 71 72 75
80 77 56 51 46 45 47 52 57 60 61 62 79 78
43 42 39 20 18 17 19 21 22 23 24 44 41 40
207 202 201 206 204 219 225 226 221 212 211 210 209 208
164 165 182 184 205 215 217 216 218 199 198 172 168

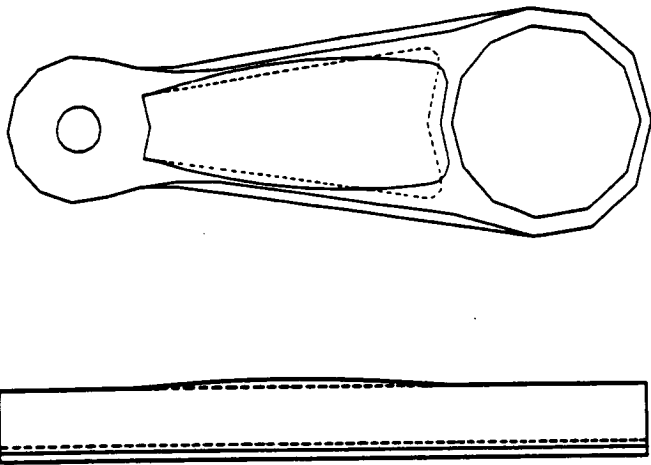
$
DESIGN VARIABLES  B4*6. B6*10. B8*10
12
1 2.500 4.200 4.200 10.000
2 20.000 20.000 20.000 40.000
3 11.000 20.000 20.000 40.000
4 2.200 8.968 8.968 17.000
5 20.750 20.750 20.750 20.750
6 2.200 15.858 15.858 26.000
7 30.400 34.300 34.300 40.000
8 1.000 4.000 4.000 10.000
9 2.200 15.285 15.285 18.000
10 4.200 18.610 18.610 19.000
11 4.200 21.976 21.976 30.000
12 4.200 26.214 26.214 29.000

$
LINK 3 113 1 0 0
1 0 1
0 0 1
76
62 49 124 182 269 369 379 378 346 339 218 155 95 61
63 50 125 183 270 297 316 315 290 234 219 156 96 64
440 515 590 679 779 903 936 947 1102
453 528 603 692 792 916 1034 1060 1100
458 533 608 697 797 921 1042
448 523 598 687 787 911 1037
427 502 577 666 766 897 975
411 486 561 650 757 839 970 986 1008
405 480 555 644 728 833 854 865 1010
1223 1300 1391 1429 1445 1444 1436 1410 1332 1267 1157 1140 1074 1206
1220 1298 1392 1430 1446 1447 1437 1411 1333 1265 1154 1142 1076 1208

$
LINK 3 28 1 0 0
1 0 2
0 0 1
76
1232 1309 1401 1462 1493 1492 1474 1424 1342 1276 1166 1151 1084 1217
1228 1306 1402 1463 1494 1495 1475 1425 1343 1273 1162 1150 1085 1216
$

```

Figure 4. DESIGN file for steering arm optimization



Initial dimensions shown as dashed lines

Figure 5. Initial and final designs of steering arm

## SUMMARY

Efficient creation of the the design model is crucial in three-dimensional shape optimization. In the ideal scheme, creation of the analysis model is completely integrated into the design model building process, thus eliminating any duplication of effort. At the same time, no compromise should be made with respect to mesh quality. For realistic three-dimensional parts, this technology is not available yet. In this paper, a design modeling approach was presented which takes advantage of the fully developed state of finite element analysis model building. In this method, the analysis model is the basis of the geometric description. Building the design model consists of overlaying a parameterization of the geometry onto the finite element mesh. This method is applicable with present technology. It has been used in a number of automotive component applications with success, one of which was presented here.

## REFERENCES

1. Ramakrishnan, C. V. and Francavilla, A., "Structural Shape Optimization Using Penalty Functions," *Journal of Structural Mechanics*, 3(4), 1974-1975, pp. 403-422.
2. Haug, E. J., Choi, K. K., Hou, J. W., and Yoo, Y. M., "A Variational Method for Shape Optimal Design of Elastic Structures," *New Directions in Optimum Structural Design*, Ed. E. Atrek et al., Wiley, New York, 1984.
3. Haug, E. J., Choi, K. K., and Komkov, V., *Design Sensitivity Analysis of Structural Systems*, Academic Press, 1986.
4. Yang, R. J., and Botkin, M. E., "Comparison Between the Variational and Implicit Differentiation Approaches to Shape Design Sensitivities," *AIAA*, Vol. 24, No. 6, 1986, pp. 1027-1032.

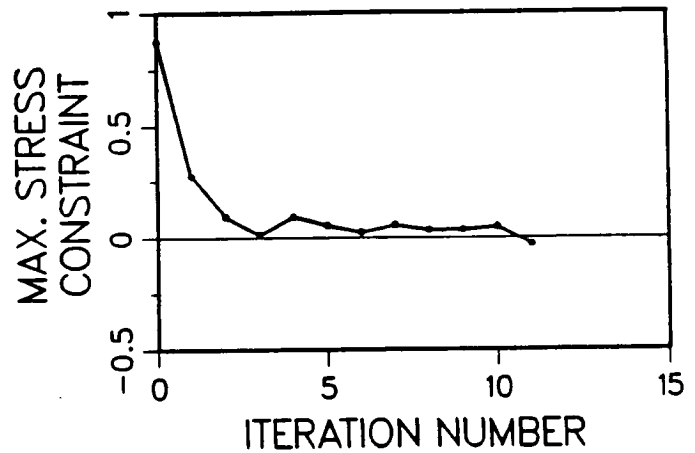
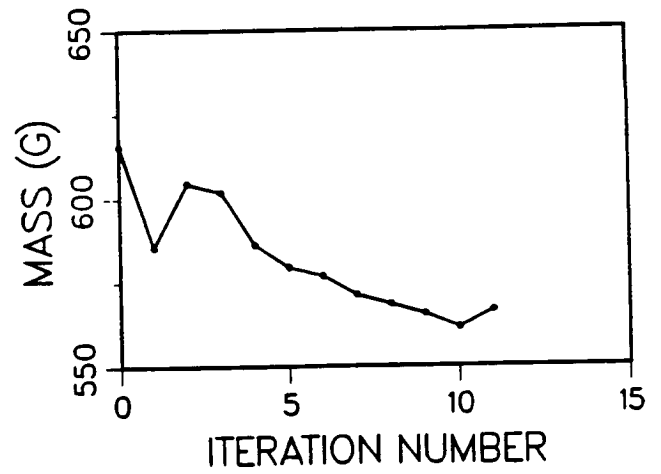


Figure 6. Design history of steering arm optimization

5. Botkin, M. E., Yang, R. J., and Bennett, J. A., "Shape Optimization of Three-Dimensional Stamped and Solid Automotive Components," *The Optimum Shape: Automated Structural Design*, Ed. J.A. Bennett and M.E. Botkin, 1986.
6. Imam, M. H., "Three-Dimensional Shape Optimization," *International Journal for Numerical Methods in Engineering*, Vol. 18, 1982, pp. 661-673.
7. Imam, M. H., "Minimum Weight Design of 3-D Solid Components", *ASME Computers in Engineering*, Vol. 3, 1982.
8. Bennett, J. A. and Botkin, M. E., "Structural Shape Optimization with Geometric Problem Description and Adaptive Mesh Refinement," *AIAA*, Vol. 23, No. 3, 1985, pp. 458-464.
9. Botkin, M. E. and Bennett, J. A., "Shape Optimization of Three-Dimensional Folded Plate Structures," *AIAA*, Vol. 23, No. 11, 1985, pp. 1804-1810.
10. Cavendish, J. C., "Automatic Triangulation of Arbitrary Planar Domains for the Finite Element Method," *International Journal for Numerical Methods in Engineering*, Vol. 8, No. 4, 1984, pp. 679-696.

11. Shephard, M. S. and Yerry, M. A., "Automatic Finite Element Modeling for Use with Three-dimensional Shape Optimization," The Optimum Shape: Automated Structural Design, Ed. J.A. Bennett and M.E. Botkin, 1986.
12. Botkin, M. E., "Shape Optimization of Plate and Shell Structures," AIAA, Vol. 20, No. 2, 1982, pp. 268-273.
13. Braibant, V. and Fleury, C., "Shape Optimal Design, A Performing C.A.D. Oriented Formulation," Proceedings of AIAA/ASME/ASCE/AHS SDM Conference, CP No. 84-0857, Palm Springs, CA, May 14-16, 1984.
14. Yang, R. J. and Botkin, M. E., "A Modular Approach for Three-Dimensional Shape Optimization of Structures," AIAA, Vol. 25, No. 3, 1987, pp. 492-497.
15. Vanderplaats, G., "CONMIN - A Fortran Program for Constrained Function Minimization User's Manual," NASA, TM X 62,282, 1973.

# RESEARCH PUBLICATION

GMR-5168

SHAPE OPTIMIZATION OF THREE-DIMENSIONAL  
STAMPED AND SOLID ALLOID VET COMPONENTS

W. B. BUCKLEY  
R. L. YAMASAKI  
J. A. BATHON

Engineering Mechanics Department

October 23, 1985

The GM logo consists of the letters "GM" in a bold, sans-serif font, with a horizontal line underneath the letters.

General Motors  
Research Laboratories  
Warren, Michigan 48090

ORIGINAL PAGE IS  
OF POOR QUALITY

MAY BE DISTRIBUTED OUTSIDE GENERAL MOTORS....., Not to be Reproduced Without Permission



N88-19123

S12-61

GMR-5168

125798  
398

SHAPE OPTIMIZATION OF THREE-DIMENSIONAL  
STAMPED AND SOLID AUTOMOTIVE COMPONENTS

M. E. Botkin, R.-J. Yang and J. A. Bennett  
Engineering Mechanics Department  
General Motors Research Laboratories  
Warren, MI 48090-9057

Presented at the  
1985 GMR Symposium

and

to be published in  
Symposium Proceedings

## Shape Optimization of Three-Dimensional Stamped And Solid Automotive Components

M.E.Botkin, R.J.Yang And J.A.Bennett  
Engineering Mechanics Department  
General Motors Research Laboratories  
Warren, MI 48090-9055

### ABSTRACT

The shape optimization of realistic, three-dimensional automotive components is discussed in this paper. The integration of the major parts of the total process: modeling, mesh generation, finite element and sensitivity analysis, and optimization is stressed. The paper will treat stamped components and solid components separately. For stamped parts a highly automated capability has been developed. The problem description is based upon a parameterized boundary design element concept for the definition of the geometry. Automatic triangulation and adaptive mesh refinement are used to provide an automated analysis capability which requires only boundary data and takes into account sensitivity of the solution accuracy to boundary shape. For solid components a general extension of the two-dimensional boundary design element concept has not been achieved. In this case the parameterized surface shape is provided using a generic modeling concept based upon iso-parametric mapping patches which also serves as the mesh generator. Emphasis is placed upon the coupling of optimization with a commercially available finite element program. To do this it is necessary to modularize the program architecture and obtain shape design sensitivities using the material derivative approach so that only boundary solution data is needed. Several realistic component designs will be shown to demonstrate the effectiveness of both capabilities.

### INTRODUCTION

Although structural optimization for sizing variables has been treated extensively in the literature for many

years[1,2] the problem of designing the shape of a structure for minimum mass is a comparatively new research topic[3,4,5]. Although earlier work[6,7,8] stressed the need for automatically modifying the mesh as the structural shape changes, limitations in the boundary representation and mesh generation aspects kept the capability from being truly automatic. Ultimately, one would like to merely describe the function of the structure to the computer in some convenient manner and then allow the program to automatically produce the optimum design[9]. The basic requirements necessary to do this are as follows: 1) the design model--this describes the shape of the structure, loads and constraints, and the design requirements; 2) the analysis model--the finite element mesh created using fully automatic mesh generation and improved using adaptive mesh refinement; and 3) the design modification--a numerical optimization process which iteratively improves the design until convergence to the optimum is obtained. Each of these topics and their implementation into the design program will be discussed.

Previous authors have not addressed the problem of handling the more general case of designing parts which are non-planar. Here the major difficulty is in modeling, in a parametric sense, all of the three-dimensional geometry. To do this it was necessary to extend the existing capability for flat parts using an assembly process of the two-dimensional segments. Furthermore, the ability to add curvature to planar segments was provided through the superposition of surface interpolation and transformation capabilities.

For solid components, very little research has been reported[7,8]. In this paper emphasis will be placed upon two major aspects which have not been previously treated. The first of these is the efficient calculation of the sensitivities of the displacement and stresses. Secondly, the idea of using one of the many commercially available finite element codes is attractive in order to alleviate the burden of software support of an analysis program sophisticated enough to handle solid models. Both of these issues have been addressed and will be discussed.

The integrated design processes described in this paper will stress the necessity for treating realistic, three-dimensional design problems typical of those found in automotive design. For this reason, the shape design element descriptions would be most suited for interfacing with the computer-aided drafting systems on which the geometry is initially created. Additionally, it is absolutely necessary

to have a capability which is as automatic as possible to free the engineer from the burden of finite element creation and modification and from the equally as great a burden of design modification.

## SHAPE OPTIMIZATION OF SHEET METAL PARTS

### Design Model Description

There are a significant number of structural components, such as the typical part shown in Fig. 1, that are produced from a single sheet of uniform thickness material. Using conventional optimization techniques in which element thicknesses are the design variables, little mass reduction can be achieved. To further reduce the mass, the shape of the part and the location of the cutouts must be represented by design variables. The resulting design model must provide the description of the boundary geometry as a function of the design variables and also the finite element structural model. To be most effective in impacting the design process, this information must be efficiently generated from conceptual sketches of the part or obtained through an interface to a computer-aided drafting (CAD) system. For that reason, the approach represented in Figs. 2 and 3 has been chosen. The part shown in Fig. 1 has been modeled in Fig. 2, using what will be referred to as boundary design elements. As well as associating the boundary with design variables, the boundary design elements are also used to define the stress constraints. Each boundary design element will be associated with at least one stress constraint which will be computed from the maximum stress of all the finite elements touching that boundary design element. The loads and structural boundary conditions are related to a set of reference nodes which are shown in Fig. 3 as key nodes. This information is in turn automatically transferred to the finite element model once it has been generated.

### Mesh Generation

Other work [6,10] has stressed the need for automatically modifying the mesh as the structure changes shape, but it was observed that the commonly used mesh generation techniques based upon coarse isoparametric or transformational mapping patches imposed limitations on the ability to treat large variations in shape. While these techniques do redistribute interior nodes as boundaries move, aspect ratios tend to get objectionably large as the shape becomes significantly different than the initial shape. Mesh grading and solution accuracy are difficult to control as well.

As an alternative to more traditional mesh generation methods, the use of fully automatic mesh generation based only upon boundary points coupled with adaptive refinement has been proposed[11]. This technique is capable of generating a nearly uniform initial mesh of triangular elements given a set of uniformly spaced boundary points. Thus, as the design changes, uniform triangular meshes can be recreated at any time.

After the design model has been created, the boundaries are automatically discretized into uniform segments called the characteristic length (CL) which is an input value. Automatic triangulation[12,13] is used to create a nearly uniform mesh from the set of boundary points and a set of points placed uniformly throughout the region's interior of approximately the same density as the boundary points. This process of creating the uniform mesh is repeated at each step in the design for which a new boundary description has been generated.

### Adaptive Mesh Refinement

Unlike the design of fixed configuration structures, it is not possible to assure the accuracy of the mesh as the shape changes, since the accuracy of various portions of the mesh will change. The ideas of adaptive mesh refinement can be incorporated to help resolve this difficulty[11].

The mesh refinement process is based upon the variation in strain energy density (SED) as a measure of the error in an element. Once SED variations have been determined for all elements, those elements which have undesirably high values must be selected for subdivision. Elements so selected define refinement regions which can be easily identified by graphical contouring. Since it is not practical from a computational standpoint to consider more than a two-step refinement process during the optimization (one initial and one refined analysis), a concept of multiple refinement regions has been implemented in an attempt to enhance convergence. As an example of the process Fig. 4(a) represents a uniform finite element mesh created using the triangulation technique described previously. Several refinement regions can be specified, as shown in Fig. 4(b), so that the resulting mesh, Fig. 4(c), will be more uniformly graded from coarse to fine. The elements in the region of highest SED variation, represented by the smallest dots in Fig. 4(b), are approximately one-fourth of the size of the initial grid. The region represented by the larger dots contains elements of approximately one-half of the

initial grid size. As many as six regions can be specified, uniformly graded down to one-eighth of the original grid size. The size of the regions can be varied depending upon the selection of an input parameter.

Obviously, the accuracy due to any refinement is unknown in advance. Although numerous papers have been written [14,15] on error estimates of total strain energy, this work has not been extended to stresses and displacements. It is desired, for the case of the iterative design process described in this paper, to have a conservative estimate of the converged finite element solution. This information may be obtained in an approximate manner using linear extrapolation, graphically represented in Fig. 5. This is a typical relationship, in the absence of a singularity, between a solution quantity and mesh size. Several steps of refinement are shown, with each step having reduced the element size in half. The solution will eventually converge to  $S_e$  and the slope of the curve reflects the rate of convergence. A conservative estimate of the converged solution, represented by points  $S_i$  and  $S_o$ , may be obtained by extrapolating data points produced by one unrefined analysis and one refined analysis. The extrapolated values will be used as stress constraints.

In order that more realistic three-dimensional plate structures can be analyzed, accurate refinements are necessary for finite elements with bending deformation. In general, refinement works best for conforming elements such as for the constant strain triangle already described. Meshes composed of these elements are always too stiff and solution convergence is predictable as shown in Fig. 5. On the other hand, meshes composed of nonconforming elements may switch from too stiff to too flexible as the refinement progresses. However, the triangular bending element used in this study [16] has been formulated in such a way as to reduce the degree of nonconformity, and convergence studies show that for uniformly refined meshes the element is always too stiff. Several examples have been presented in Ref. 11 which indicate that although the results are not as predictable as for the constant strain triangle, they are quite satisfactory.

### Extension To Nonplanar Parts

The design process which has been described has been extended in order to handle more realistic stamped sheet metal parts [17]. This was accomplished by treating the part as an assembly of the two-dimensional segments described

above. Each segment has one completely closed exterior boundary which may contain one or more interior cutouts. Segments may be joined along straight sides to form more complex assemblies. Furthermore, segments may be rotated along the joined edges to form three-dimensional geometry, as shown in Fig 6(a). Because each segment is represented by two-dimensional boundary information only, the addition of surface curvature to a planar segment for added stiffness must be addressed separately. Large curvature, such as a cylinder in Fig. 6(b), is accomplished through the definition of a cylindrical coordinate system for that segment alone. All nodes in that segment are transformed to the new surface. Small curvatures are treated by direct projection as shown in Fig. 6(c). The final assembly process can be seen in Fig. 7 in which all the three-dimensional geometry has been expressed in terms of a small number of parameters which can be treated as design variables.

### Interactive Graphics Geometrical Modeling

The need to model more complex geometries makes it obvious that some form of model preparation based upon graphics oriented preprocessing is necessary. Unfortunately, existing finite element preprocessors cannot be used directly, since they offer no means of parameterizing the shape of the model. Although some of the more recently developed modelers do include boundary functions, such as splines, there are no design parameters available externally for use with other programs. Furthermore, since the finite element mesh must change to reflect shape changes, loads and constraints must be associated with boundary functions instead of being directly applied to the finite element mesh, as in the typical modeling system. As a result, a special graphics preprocessor for shape optimization was developed [18], which allows a user to create a parameterized finite element model. A part is modeled as a collection of planar part segments, which are assembled to form a three-dimensional plate structure. Design variables define the shape of each part segment. Loads and constraints are applied to finite element nodes through boundary functions, instead of being applied directly to the nodes.

To begin model preparation, the user first selects the x and y dimensions of the part. Next, commands and cross-hairs are used to create the key nodes and boundary design elements that define the geometry of the part to be optimized. Figure 8(a) shows the six key nodes needed to define the boundary of a planar triangular bracket. Three exterior key nodes locate the perimeter of the part, while

three interior key nodes locate an interior cutout boundary. Associated with each key node is a radius, represented as a circle in Fig 8(a). The radius, as well as the x and y coordinates, are automatically designated as design variables.

Once the necessary key nodes have been created, the cross-hairs are used to connect the key nodes and create the boundary design elements, as shown in Figure 8(b). If the same key node is selected twice, a circular arc boundary design element is created. A circular arc element can be used to represent a round boundary, a fillet, or a circular hole. If two different key nodes are selected, the user can choose to connect the two key nodes with either a straight boundary design element or a double cubic boundary design element, as shown in Fig. 3. All design variables specified for a particular element type are automatically assigned when the element is created. Commands are available to link design variables, as required.

Other commands are available to be used for applying constraints or loads to a given boundary. The terminal cross-hairs are first used to select the boundary to be supported or loaded. The user is then prompted for a constraint type or a load magnitude and direction. The constrained boundaries are indicated by a letter 'C', while the loaded boundaries are indicated with a letter 'L', as shown in Fig. 8(b). At the time when loads are applied, optimization constraints on displacements can also be specified.

Most real production parts, however, have more complex geometries than these examples. For instance, a common manufacturing operation used to add stiffness to a planar part involves adding a lip, or flange, along the edge of the part. Modeling such a part with a conventional finite element preprocessor is relatively simple, but if the design of the part is to be automated, the geometric model of the part must fulfill the requirements already mentioned. Commands are available to create multiple part segments as shown in Fig. 7. An additional command can be used specifically for creating flanges, which automates some of the multiple-segment-creation steps.

Figure 9(a) shows six flanges added around the perimeter of the triangular bracket. A flange is added by using the cross-hairs to locate the portion of the boundary for which a flange is desired. The user is then prompted to specify the flange height at each end. The model is completed by



specifying the angle that each flange is rotated relative to the base part to form a three-dimensional model. This angle is normally ninety degrees. Each of the six flanges, as well as the base triangular bracket, is a separate part segment, on which a finite element mesh is generated. Figure 9(b) shows the assembled finite element model of the triangular bracket, generated from the boundary shape information created with the preprocessor.

### THREE-DIMENSIONAL SOLID COMPONENTS

Only a limited amount of work has been accomplished in three-dimensional shape optimization using solid finite element analysis[7,8]. Issues not treated previously will be emphasized in this paper[19]. Because a fully automatic mesh generation scheme which relies only on surface data[20] has yet to be developed, the boundary description format described for thin parts cannot be implemented for solid three-dimensional parts. Instead, it will be assumed that surface representation and mesh generation will be handled by a generic modeling scheme based upon isoparametric mapping patches described in Ref. 8 and shown for a typical part in Fig. 10.

The two topics which will be addressed are design sensitivities and program architecture. Work in both of these areas were largely driven by the desire to use a variety of structural analysis programs (NASTRAN, ANSYS, ADINA, etc.) to be used with a relatively small amount of additional program development. In this study, NASTRAN was used for analysis.

#### Design Sensitivity analysis

The variational design sensitivity theory uses the material derivative concept of continuum mechanics and an adjoint variable method to obtain computable expressions for the effect of shape variation on the functionals arising in the shape design problem. The resulting expressions provide analytical sensitivities of structural response.

The variation of displacement functional  $\psi$  with respect to shape change is derived by differentiating the variational equilibrium equation and employing the adjoint variable method, to obtain [21-23]

$$\partial\psi/\partial b = - \int_{\Gamma} \sigma^{ij}(z) \epsilon^{ij}(\lambda) n^T \partial r / \partial b \, d\Gamma \quad (1)$$

This equation is an integral along the perturbed boundary in which the required data for evaluation are the stresses from the actual load,  $\sigma^{ij}$ , the strains from the adjoint load,  $\epsilon^{ij}$ , the position vector,  $r$ , and the design variable vector,  $b$ . It should be pointed out that in Eq. 1 assumptions have been made in the derivation so that the kinematically constrained boundary and loaded boundary are assumed to be fixed, and the variation of the displacement functional is only affected by the normal movement of the boundary of the physical domain. Physically, the adjoint solution required in Eq. 1 is interpreted by applying a unit load at the point where the displacement is of interest.

To see the advantage of Eq. 1, a comparison should be made [24] with the well known expression for design sensitivities resulting from the implicit differentiation of the finite element equations

$$\partial z / \partial b = -K^{-1} \partial K / \partial b z \quad (2)$$

This equation evaluates the displacement derivative by computing derivatives of the terms of the stiffness matrix. There are two shortcomings to this approach. First, obtaining analytical expressions for the stiffness matrix derivatives is very difficult for boundary movements. These expressions are, in general, different for each element type, thereby requiring special computer code for each different element type. For this reason, a finite difference method is generally used to obtain stiffness derivatives. This usually requires a judicious choice of the step size to maintain accuracy. Finally, if it is desired to use a commercial finite program for analysis--for which the source code is not available--it is very difficult to manipulate the stiffness matrices to compute the needed derivatives. For these reasons, Eq. 1 is a more desirable expression for computing displacement sensitivities. The needed stresses and strains can be stored by most programs on files to be used by a post-processing routine to obtain the derivatives.

The stress variation also can be derived to obtain an expression similar to Eq. 1, except that the discontinuity of the stresses along the interelemental boundaries has to be properly handled. A characteristic function, which averages stress over a small region, is introduced to treat stress constraints in Refs. 24 and 25. This approach is similar to using the finite element center as the stress constraint point if the element is chosen as the small region and may lead to a misleading constraint value and may

result in an undesirable or inaccurate optimum shape if the finite element model is inadequate [25].

An alternative that avoids this problem is to obtain the stress sensitivity at a point, using the definition of stress computation in finite element analysis. The elemental stresses are computed by using the following equation

$$\sigma = D B z^e \quad (3)$$

where D is the elasticity matrix, B the strain recovery matrix, that contains the derivatives of shape functions, and  $z^e$  an elemental displacement vector. Differentiating Eq. 3 with respect to the design variables, b, one obtains

$$\sigma'_i = D(Bz_i^{e'} + B'_i z^e) \quad (4)$$

where the subscript i with a prime superscript indicates the derivative with respect to the ith design variable. Notice that the first term on the right side of Eq. 4 is only a combination of displacement gradients, and can be obtained by applying a combined adjoint load to the system and using the same formula of Eq. 1.

The primed matrix of the second term of Eq. 4 can be evaluated from the derivative of the nodal coordinates with respect to shape design parameters [26]. It can be computed analytically or by using a finite difference method. For a linear shape function element, such as constant stress triangular element, the matrix B' vanishes, while for a quadratic element, the B' matrix is constant. Therefore, the finite difference method is sufficient to evaluate the B' matrix, except when a higher order element is used. In this study, analytical derivatives are used for B' and the eight corner points of the solid element are chosen as the stress constraint points.

### Modularized Program Architecture

It was desired to have a system which uses a commercial finite element code as the analysis capability because of the generally widespread acceptance by the structural analysis community of such codes. A major drawback to achieving this goal is that most commercial finite element codes cannot be used as a subroutine. This problem was addressed by building a system of independently executable program modules in which the overall execution is controlled by job control language.

The modularized system is comprised of a mesh generator, the finite element code (NASTRAN), the adjoint load and constraints definition program, a design sensitivity analysis module, and an optimization module. Each of those is an independent program and is treated as a module. The flow chart of the system is shown in Fig 11. Initially, one has to generate a generic model for the structural component, and create a NASTRAN data deck for the NASTRAN run. The whole cycle of the system proceeds as follows: run the NASTRAN code for the actual load; calculate the cost function, constraints, and the adjoint loads using the NASTRAN output; rerun the NASTRAN code for the adjoint loads; and perform the design sensitivity analysis and optimization to obtain a new design. Finally, a new finite element mesh and NASTRAN data deck for the new design are generated.

The MSC/NASTRAN version 63 finite element code is employed for analysis. The new feature of the NASTRAN data base is used to save computing time for reanalysis of the adjoint loads. This data base, created by the first NASTRAN run, preserves the stiffness and boundary condition information and results in easier input data preparation and less computing time for the reanalysis. The displacements, stresses, and geometric information that are needed for design sensitivity calculation are obtained by using an ALTER feature in NASTRAN to write that information on a file for postprocessing.

The ADJLOD module (Fig. 11) is used to define the cost function and constraints for the design problem, and to calculate the adjoint loads for the constraints which are active or violated. The displacements, stresses, and geometric information from the NASTRAN output are first read to define the constraints for the structural component. A NASTRAN deck containing the adjoint loads is then created for reanalysis.

The SENSTY module (Fig. 11) performs the design sensitivity analysis for the cost and the active constraints, and then performs the optimization process by calling the optimizer (CONMIN[27]) as a subroutine. Before executing the module, the NASTRAN output files for the actual load and the adjoint loads should be available. The module then changes the design and creates new input data for the MESHGN module which will generate a new mesh and a new NASTRAN data file for the next design iteration, if necessary.

## DESIGN EXAMPLES

### Three-Dimensional Sheet-metal Part

Figure 12 shows the initial shape and dimensions of a realistic design example of a sheet metal part [17]. The model was initially created in two dimensions and then segments 2 and 4 were transformed into the third dimension. Structural boundary conditions were imposed around the holes labelled C and D. Loads P1 and P2 were applied at hole A in the y and z directions, respectively. Load P3 was applied at hole B in the y direction. The design criteria were a stress limit on all boundaries and a displacement limit at hole A. CL was chosen to be 0.80 cm for the initial mesh.

The current model is similar to an earlier part [17], except that flanges on the new model add seven flange design variables to the problem. The locations of these design variables are shown in Fig. 13. A total of nineteen design variables were used to parameterize the part's shape. Figure 14 shows the initial, unrefined finite element mesh.

This part was modeled to determine how the program would reduce the mass and tailor the flanges, subject to a displacement constraint. A displacement constraint was applied to the hole A, such that the displacement of the point was limited to 1 millimeter in the -z direction. Figure 15 shows the initial and final part designs. The program removed material from the interior cutouts on the base triangular part segment and the cylindrical part segment. A small amount of in-plane curvature was added along the edges of the triangular part segment to which flanges are attached. The flange heights were reduced to less than half the initial values everywhere except along the upper edge of the triangular part segment. The flange heights along this edge are controlled by flange design variables 3 and 4, as shown in Fig. 13. This edge serves as the primary load path for the structure, since it transfers the load from the tip of the triangular part segment to the support points. As a result, one would expect the flange along this edge to be the most important in maintaining the stiffness of the part. The flange design variable values for the initial and final designs are given in Table 1.

Figure 16 shows the design history for this part. A design variable move limit of five percent was used for the first ten steps, followed by a move limit of 2.5 percent for the last fourteen steps. The characteristic length was reduced from .8 to .6 in the last four steps to obtain more

accurate displacement values in the unrefined analyses. The reduction of the characteristic length eliminated design oscillations that emerged once the displacement constraint became active. The initial unrefined finite element mesh included 3000 degrees of freedom, while the initial refined mesh contained 4000 degrees of freedom.

Finally, some comments are in order concerning the results. First, the design history (Fig. 16) does not show traditional convergence behavior. The optimizer was turned off when it was felt that further mass reduction would require an excessive amount of computer time. Second, one might question the finite element accuracy in the flange areas. Constant strain elements were used, and only one or two elements were used to span the depth of each flange in the unrefined mesh. Bending of the flanges could result in stress variations that would not be picked up by so few constant strain elements. For this reason, the automatic mesh refinement technique described above was used to minimize this error.

Table 1. Design Variables for Transmission Bracket

No.	initial	final	lower bound	upperbound
1	2.12	0.91	0.5	3.0
2	1.50	0.68	0.5	3.0
3	1.50	0.89	0.5	3.0
4	1.50	0.89	0.5	3.0
5	2.12	0.96	0.5	3.0
6	1.50	0.68	0.5	3.0
7	1.50	0.67	0.5	3.0

Three-Dimensional Solid Part

An idealized engine connecting rod, which connects the crank shaft and piston pin of an engine and transmits an axial compressive load during firing and a tensile load during the intake cycle of the exhaust stroke, is employed as the example [19]. Shape optimization of similar components have been studied by Yoo et al. [28] and Yang et al. [29] assuming a plane stress state. However, a fully three-dimensional shape optimization for the connecting rod is still not available in the literature.

Figure 17 shows the generic model for the connecting rod. For simplicity, the right hole of the connecting rod

which connects the piston pin is fixed to eliminate rigid body motion; and the arbitrarily selected pressure of 3000 MPa is applied to the left hole, from 0 to 90 degrees, to simulate the firing forces. The von Mises stress constraint is imposed at each node in the finite element model of the connecting rod. The critical yield stress used for analysis is chosen as 3000 MPa. Young's modulus and Poisson's ratio are  $10.0 \times 10^6$  MPa and 0.3, respectively. The numerical data were selected to demonstrate the use of the system and are not representative of a specific production part.

Using the symmetrical conditions, only a quarter of the structure needs to be analyzed. The design variables are shown in Fig. 17. In this model, 8 design variables are chosen; 5 parameters define the shape of the shank and neck regions, 2 are the outer radii of the right and left holes, and 1 parameter defines the height of the web. The finite element model, as shown in Fig. 18, contains 105 solid(20 node) elements, 928 nodal points, and 2126 degrees-of-freedom.

The initial values of the design variables are shown in Table 2. Initially, the volume is 15686.7 cu mm with no stress violation. After 20 design iterations, it is reduced to 7217.8 cu mm with no stress violation. The final design variables and the final shape are shown in Table 2 and Fig. 17, respectively. Figures 19 and 20 show the design histories for the cost and the maximum constraint values, respectively, of the idealized connecting rod. In Fig. 19, one observes that the convergence rate is reasonably good. From design iterations 10 to 17, the optimizer tries to force the design into the feasible region. The slow correction for stress violation shown in Fig. 20 may result from Taylor's series expansion approximation for functions.

Table 2. Design Variables for Engine Connecting Rod

No.	initial	final	lower bound	upperbound
1	10.956	12.512	0.1	100.0
2	6.37	2.8478	0.1	100.0
3	3.9667	1.4220	0.1	100.0
4	3.0024	1.0964	0.1	100.0
5	3.2711	1.2733	0.1	100.0
6	6.8156	7.2219	0.1	100.0
7	31.271	26.461	24.0	100.0
8	17.553	13.300	13.3	100.0

## SUMMARY

An integrated approach to the shape design problem has been described for sheet-metal parts in which the problem description is stated in a simple format, the finite element mesh is generated automatically, and its accuracy is improved by adaptive mesh refinement. Non-planar structures can be treated using an assembly process of two-dimensional segments in such a way that all three-dimensional geometry is expressed in terms of a relatively small number of parameters. Surface curvature variations can be added to the planar sub-assemblies through the superposition of a variety of surface transformation and mapping options. All of the geometric problem description has been formulated in such a way that it is particularly suitable for interface to modern CAD systems.

It was found that for the design problem in which the boundaries of the part are moving, the accuracy of the finite element mesh must be continuously assessed and updated. Strain energy density variations within an element were used as a measure of error. Elements with errors greater than a specified value in an unrefined analysis were refined by adding nodes, and a new mesh was created using automatic triangulation. Results of the refined analysis were combined with the unrefined results to compute stress intensification factors which were used to approximate a refined solution for intermediate designs in which refinement did not take place.

The development of a modular computer program for the shape optimization of three-dimensional solid components is also discussed. The program uses NASTRAN for analysis and CONMIN for optimization. Since design sensitivities with respect to shape variables are not available in NASTRAN, a module had to be written to obtain these sensitivities which is based upon the material derivative concept applied to the variational state equation. Parameterized surface definitions and the finite element mesh were obtained from a module based upon generic modelling concepts. Each program module is a separately executable program but all modules can be executed sequentially using Job Control Language. A realistic design example has been provided to demonstrate the capabilities of the program.

In general, it has been shown that it is possible to automate the structural design process for determining the shape of quite complicated three-dimensional components



through the integration of a parameterized geometric description, automatic mesh generation, finite element analysis, design sensitivity analysis, and optimization. The resulting capabilities eliminate the need for tedious data transfer inherent in existing trial and error design approaches as well as eliminating many of the repetitive steps involved.

#### REFERENCES

1. Schmit, L. A., "Structural Synthesis by Systematic Synthesis", Proc. 2nd Conf. on Electronic Computation ASCE, New York, 105-122(1960).
2. Vanderplaats, G.N., "Structural Optimization - Past, Present, and Future" AIAA Journal, Vol. 20, No. 7, 992 - 1000(1982).
3. Zienkiewicz, O. C., and Campbell, J. S., "Shape Optimization and Sequential Linear Programming," Chap. 7 in OPTIMUM STRUCTURAL DESIGN, edited by R. H. Gallagher and O. C. Zienkiewicz, John Wiley & Sons, New York(1973).
4. Haug, E. J., Choi, K. K., Hou, Y. M., and Yoo, Y. M. "A Variational Method for Shape Optimal Design of Elastic Structures," OPTIMAL STRUCTURAL DESIGN II, (Ed. R. H. Gallagher), Wiley, New York(1983)
5. Haftka, R.T. and Gandhi, R.V., "Structural Shape Optimization-A Survey", The 26th AIAA SDM Conference, CP No. 85-0772, 617-628(1985).
6. Botkin, M. E., "Shape Optimization of Plate and Shell Structures," AIAA Journal, Vol. 20, No. 2, 268-273(1982).
7. Imam, M. H., "Three-Dimensional Shape Optimization," International Journal for Numerical Methods in Engineering, Vol. 18, 661-673(1982).
8. Imam, M. H., "Minimum Weight Design of 3-D Solid Components", Proceedings of the 2nd ASME Computers in Engineering Conference, Vol. 3, 119-126(1982).
9. Bennett, J. A., and Botkin, M. E., "Structural Shape Optimization with Geometric Problem Description and Adaptive Mesh Refinement", AIAA Journal, Vol. 23, No. 3, 458-464(1985).

10. Braibant, V. and Fleury, C., "Shape Optimal Design Using B-Splines", Computer Methods in Applied Mechanics and Engineering, Vol. 44, 247-267(1984).
11. Botkin, M. E., "An Adaptive Finite Element Technique for Plate Structures", Technical Note, AIAA Journal, Vol.23, No.5, 812-814(1985).
12. Cavendish, J. C., "Automatic Triangulation of Arbitrary Planar Domains for the Finite Element Method," International Journal for Numerical Methods in Engineering, Vol. 8, 679-696, 1974.
13. Frey, W.H. and Cavendish, J.C., "Fast Planar Mesh Generation Using the Delaunay Triangulation", Presented to the Society for Industrial and Applied Mathematics Meeting, Seattle, WA, July 16-20(1984).
14. Babuska, I., and Rheinbolt, W. D., "Adaptive Approaches and Reliability Estimates in Finite Element Analysis," Computer Methods in Applied Mechanics and Engineering, No. 17/18, 519-540(1979).
15. Shephard, M. S., "Finite Element Grid Optimization with interactive Computer Graphics," Ph.D Thesis, Department of Structural Engineering, Cornell University(1979).
16. Conner, J. J. and Will, G., "A Triangular Flat Plate Bending Element," M.I.T., Department of Civil Engineering, Report TR-68-3, Cambridge, MA(1968).
17. Botkin, M. E., and Bennett, J. A., "Shape Optimization Of Three-Dimensional Folded Plate Structures," 1984 AIAA SDM Conference, CP No. 84-0856, Palm Springs, CA., May 14-16, (1984).
18. Botkin, M.E. and Gressel, G.S., "Shape Optimization of Sheet Metal Components With Flanges", to be presented at the 6th SAE international Vehicle Structural Mechanics Conference, Detroit, Michigan, April 22-25(1986).
19. Yang, R. J., and Botkin, M. E., "A Modular Approach For Three-Dimensional Shape Optimization Of Structures", to be presented at the 27th AIAA SDM Conference, San Antonio, Texas, May 19-21(1986).
20. 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, No.11, 1965-1990(1984).

21. Haug, E. J., Choi, K. K., Hou, J. W., and Yoo, Y. M., "A Variational Method for Shape Optimal Design of Elastic Structures," New Directions in Optimum Structural Design, Ed. E. Atrek et al., Wiley, New York(1984).
22. Choi, K. K. and Haug, E. J., "Shape Design Sensitivity Analysis of Elastic Structures," Journal of Structural Mechanics, 11(2), 231-269(1983).
23. Haug, E. J., Choi, K. K., and Komkov, V., Design Sensitivity Analysis of Structural Systems, Academic Press, (1985).
24. Yang, R. J., and Botkin, M. E., "The Relationship Between the Variational Approach and the Implicit Differentiation Approach to Shape Design Sensitivities," presented at the 1985 AIAA SDM Conference, CP No. 85-0774, Orlando, Florida, April 15-17, (1985).
25. Yang, R. J., Choi, K. K., and Haug, E. J., "Numerical Considerations in Structural Component Shape Optimization," ASME Journal of Mechanisms, Transmissions, and Automation in Design, paper No. 84-DET-219(1984).
26. Ramakrishnan, C. V. and Francavilla, A., "Structural Shape Optimization Using Penalty Functions," Journal of Structural Mechanics, 3(4), 403-422(1974-1975).
27. Vanderplaats, G., "CONMIN - A Fortran Program for Constrained Function Minimization User's Manual," NASA, TM X 62,282(1973).
28. Yoo, Y. M., Haug, E. J., and Choi, K. K., "Shape Optimal Design of An Engine Connecting Rod," ASME Journal of Mechanism, transmissions, and Automation in Design, Vol. 106, 415-489(1984).
29. Yang, R. J., Choi, K. K., and Haug, E. J., "Finite Element Computation of Structural Design Sensitivity Analysis," Report CCAD No. 84-3, The University of Iowa, (1984).

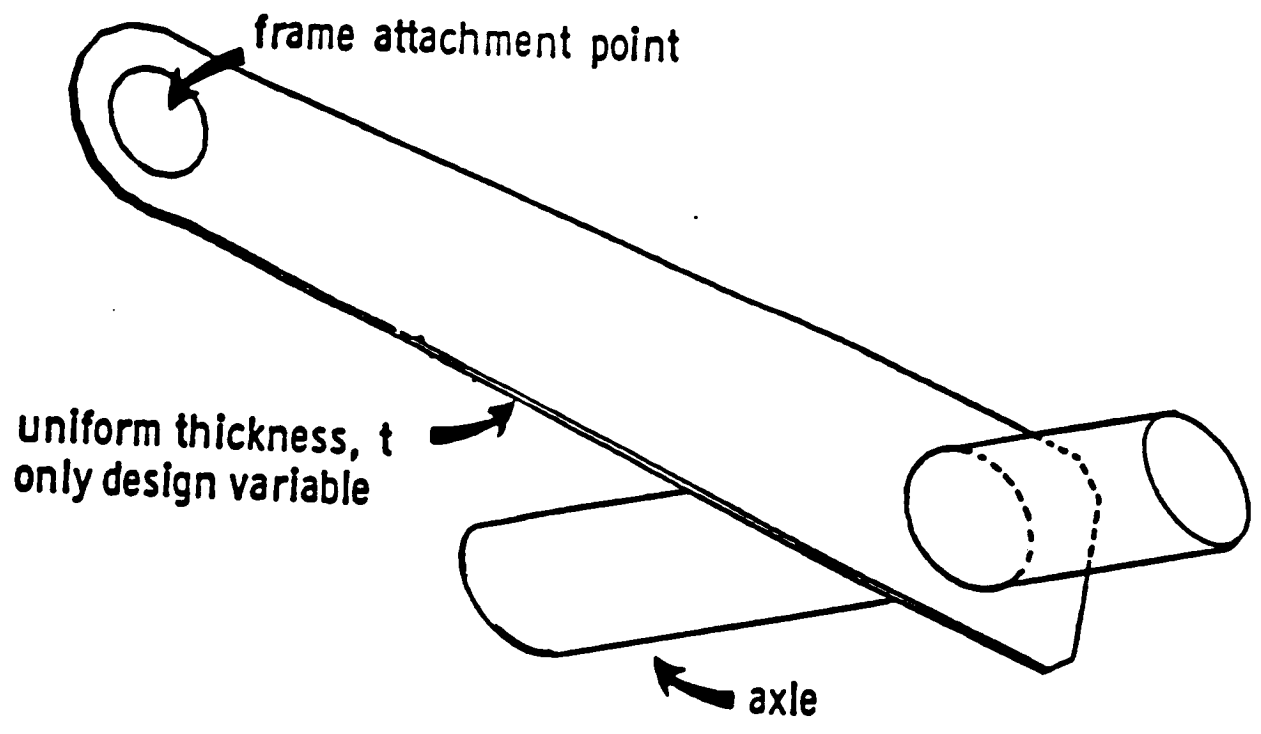
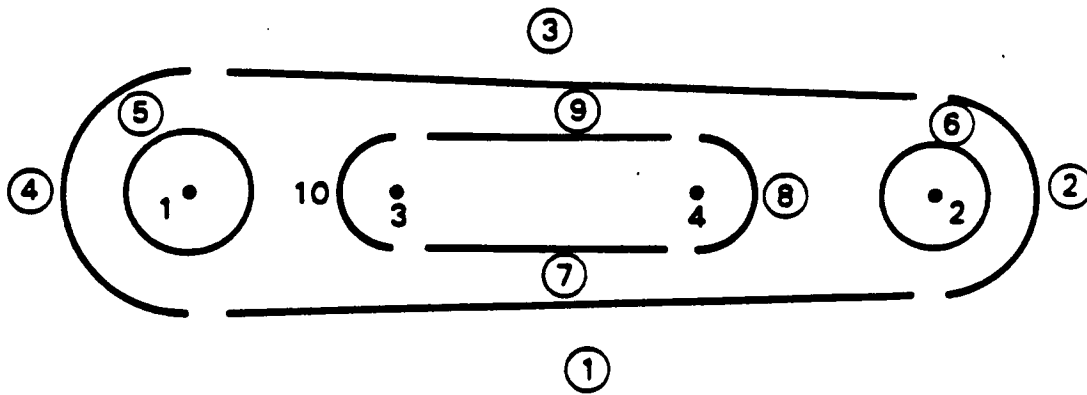


FIG. 1 Typical Part



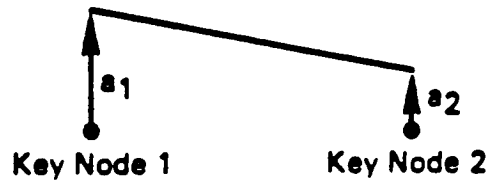
i Key Nodes

① Boundary Elements

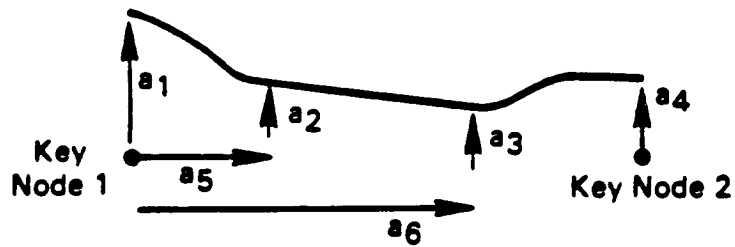
FIG. 2 Boundary Elements For Typical Part



a. Circular Arc

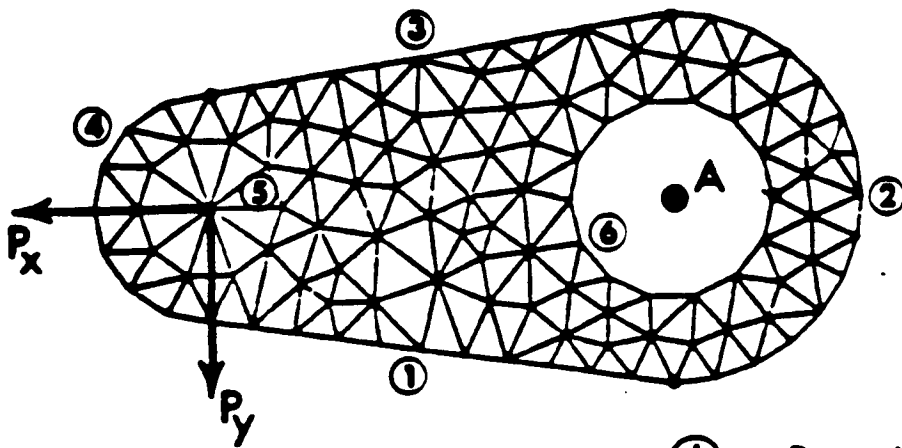


b. Straight Line Segment



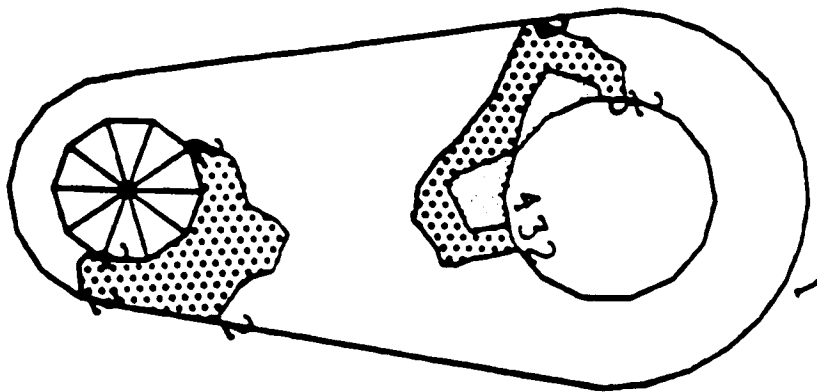
c. Double Cubic Connected by Straight Line.

FIG. 3 Boundary Design Elements

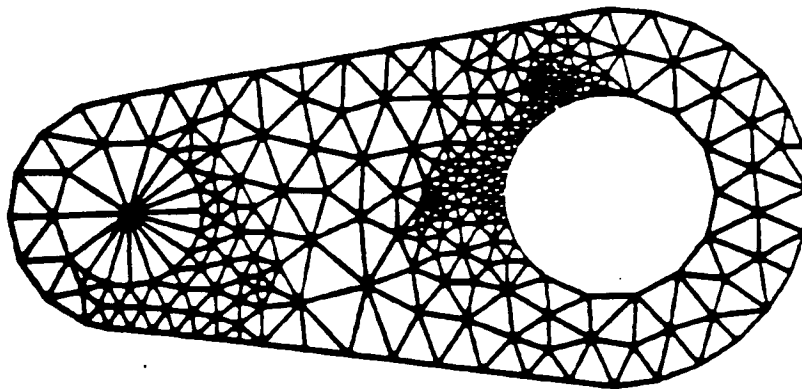


① - Boundary Element  $i$

a) Initial Uniform Mesh



b) SED DIFFERENCE CONTOURS



c) REFINED MESH

FIG. 4 Mesh Refinement

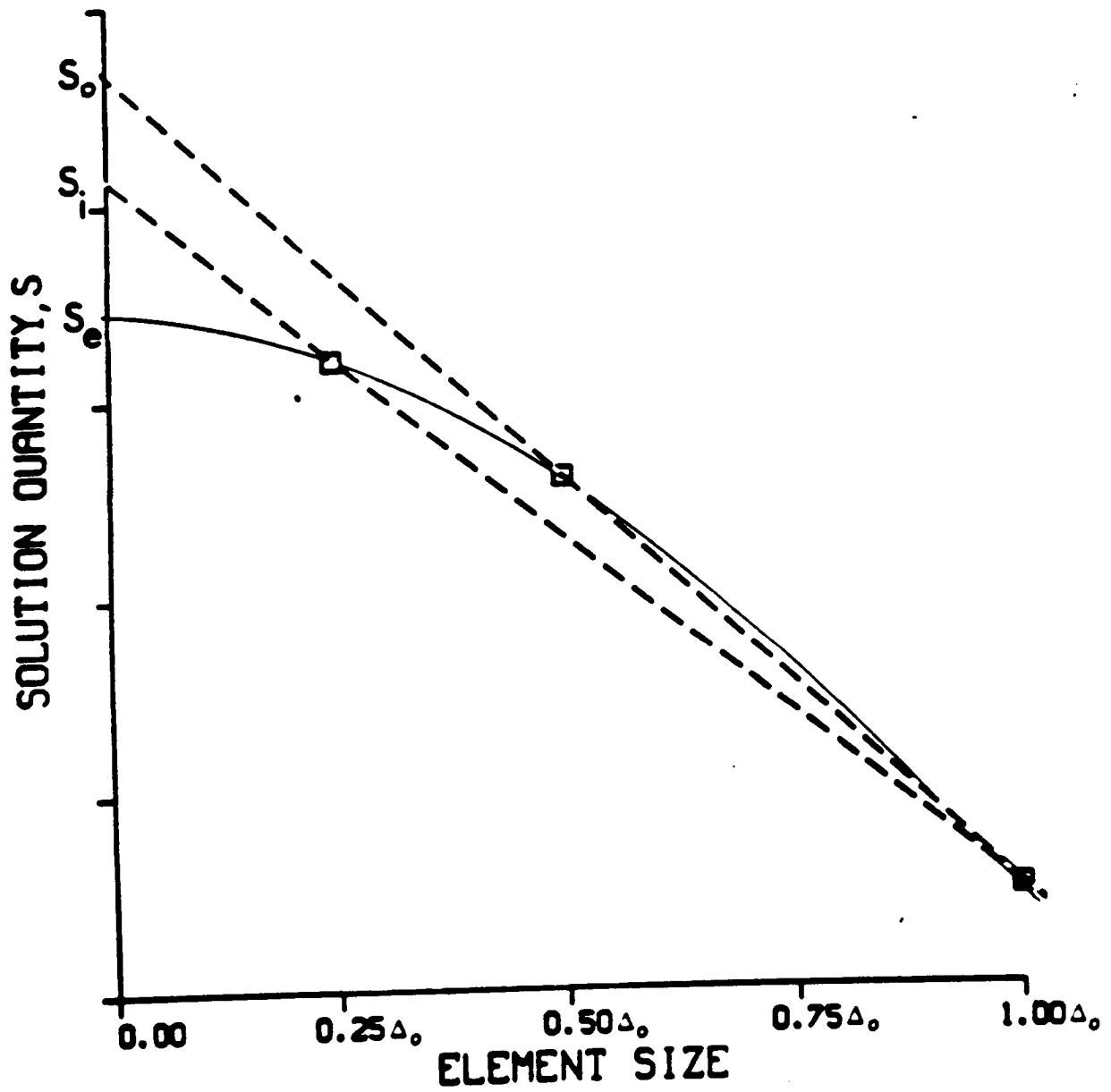
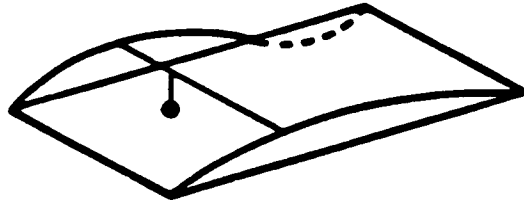


FIG. 5 TYPICAL SOLUTION CONVERGENCE



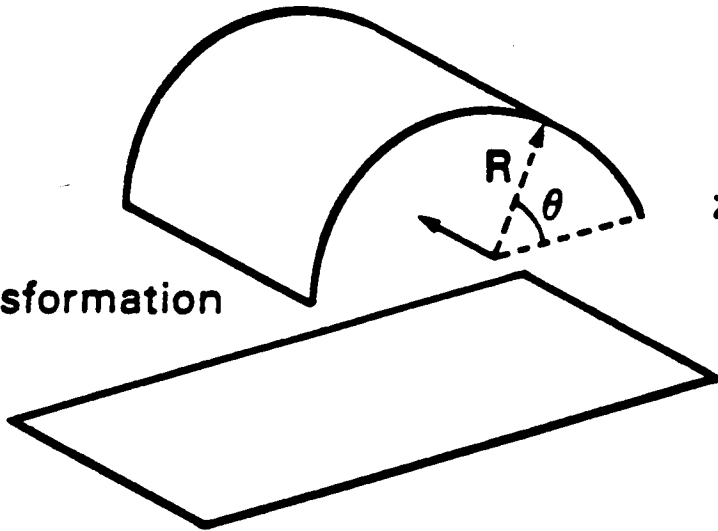
(c) Projection



$$z = Q(x,y)$$

Q = Interpolation  
Description  
of Surfaces

(b) Transformation



$$z = R \cos \theta$$

(a) Assembly of Segments

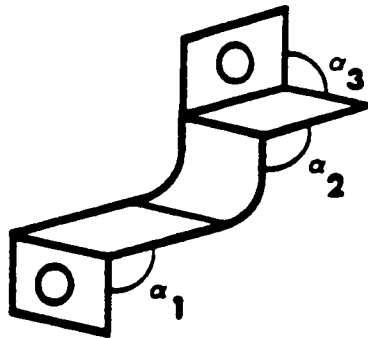


FIG. 6 THREE FORMS OF NON-PLANAR STRUCTURES

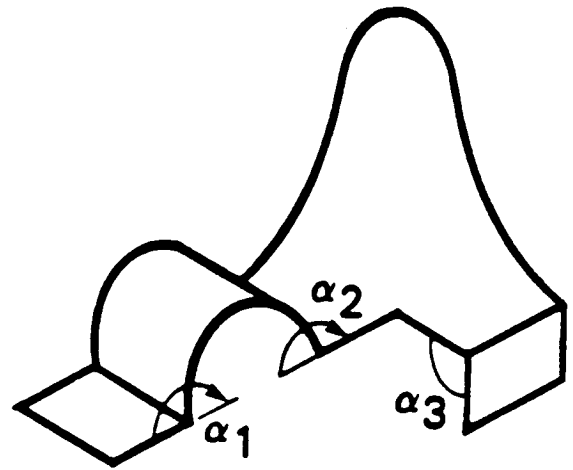
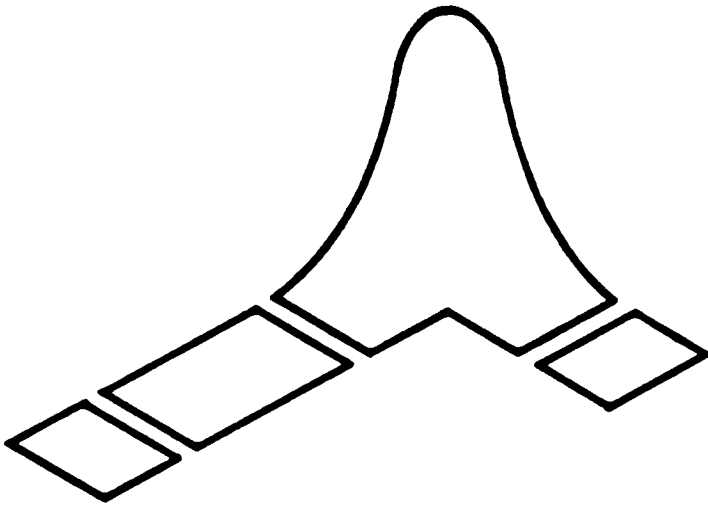
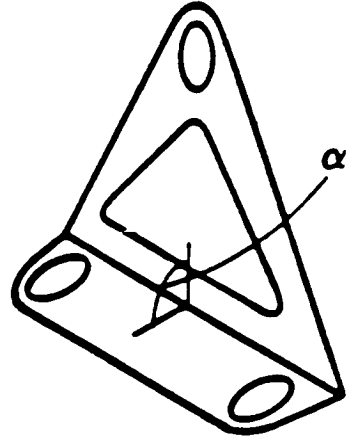
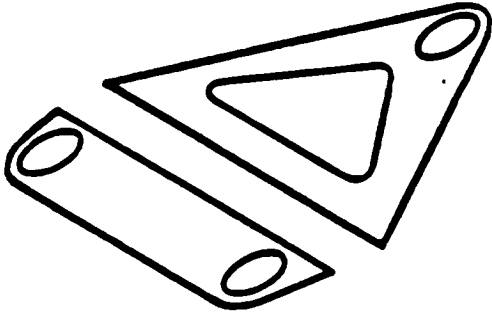
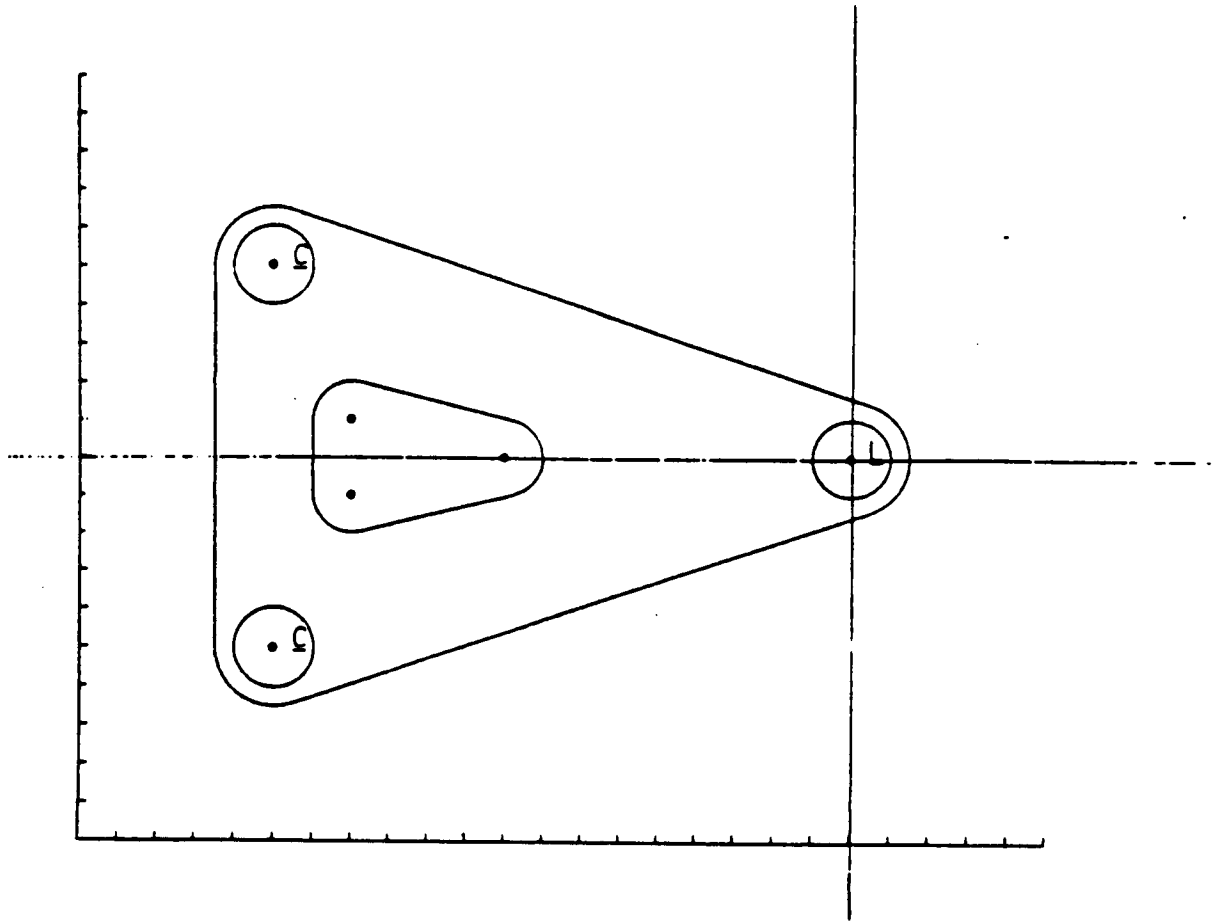


FIG. 7 Assembly and Rotation of Segments



(b) Boundary Design Element Creation

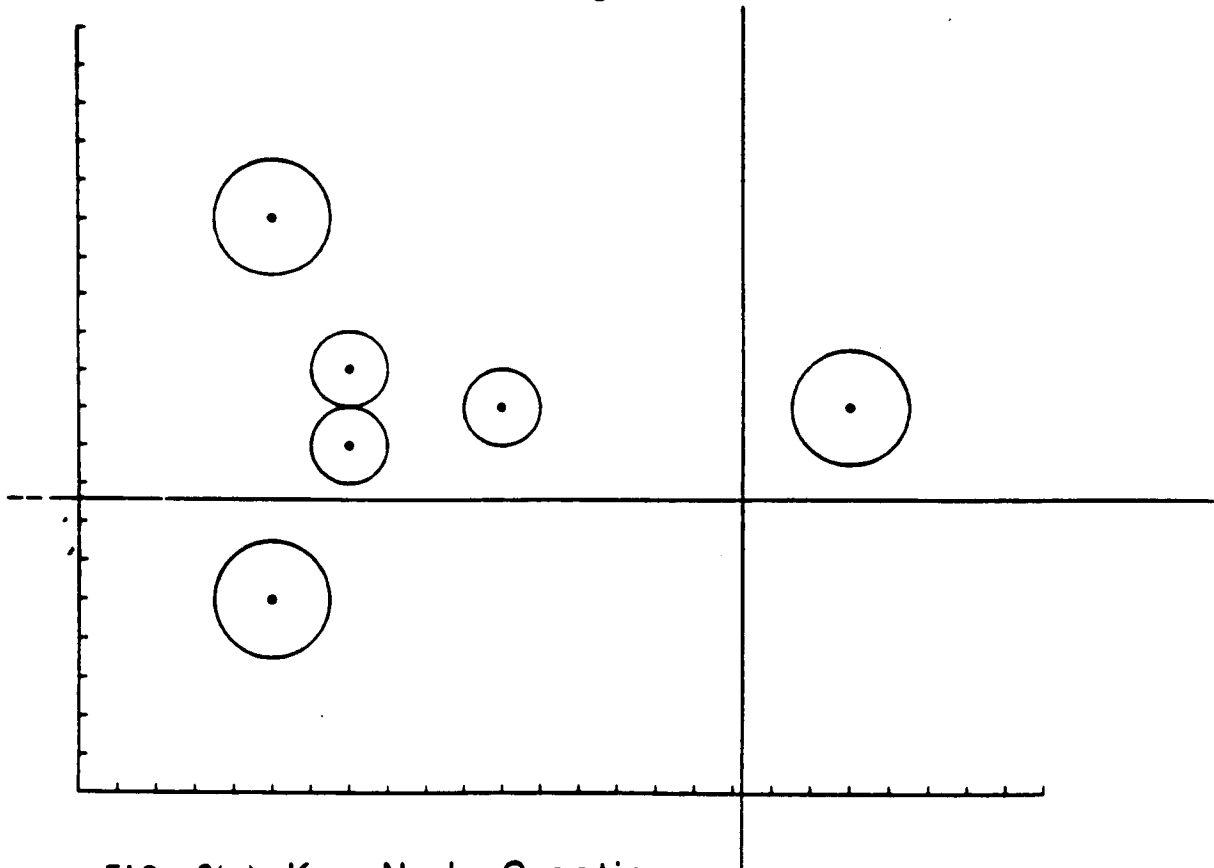
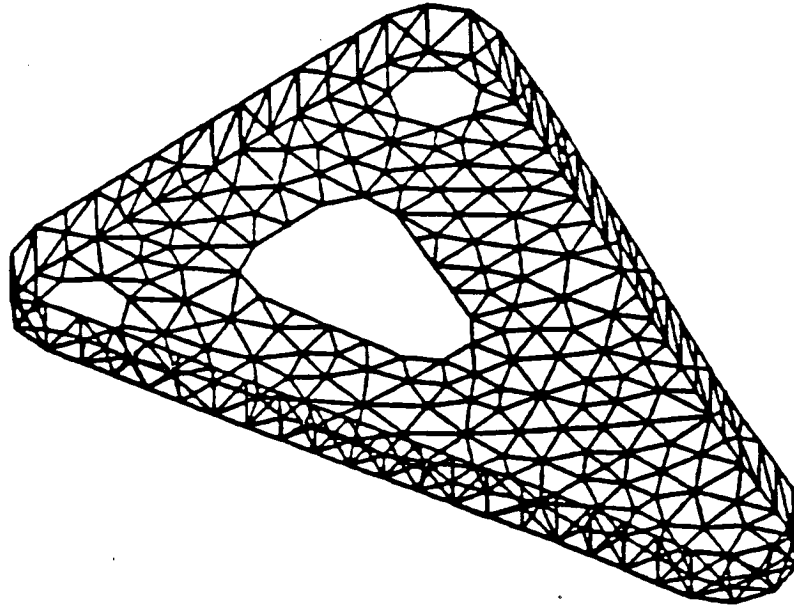


FIG. 8(a) Key Node Creation



(b) Triangular Bracket Finite Element Mesh

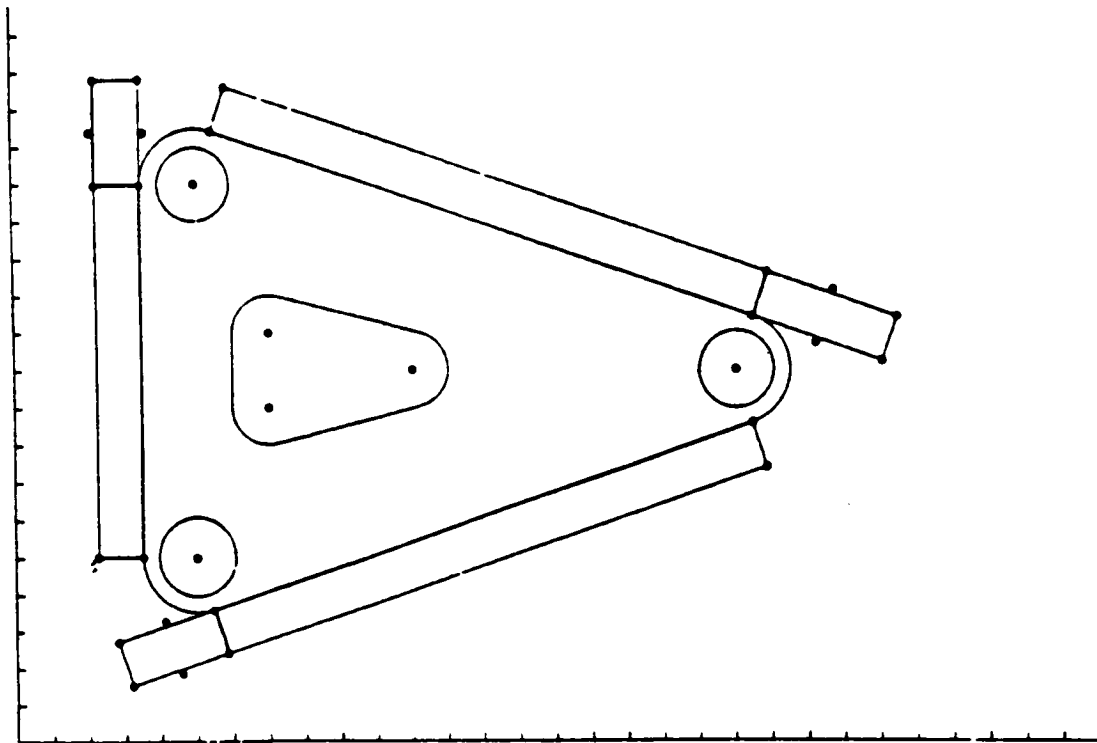
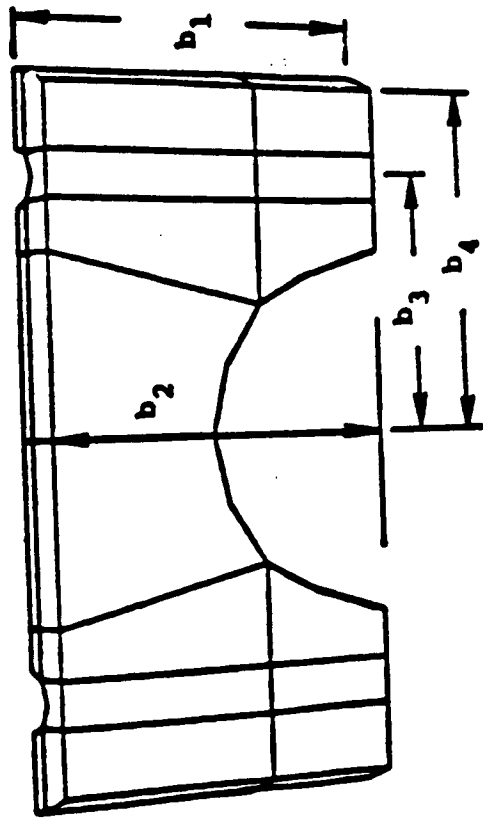


FIG. 9(a) Flanges Added to Triangular Bracket Model



Note:  $b$ 's are design variables

FIG. 10 Generic Model of Engine Bearing Cap

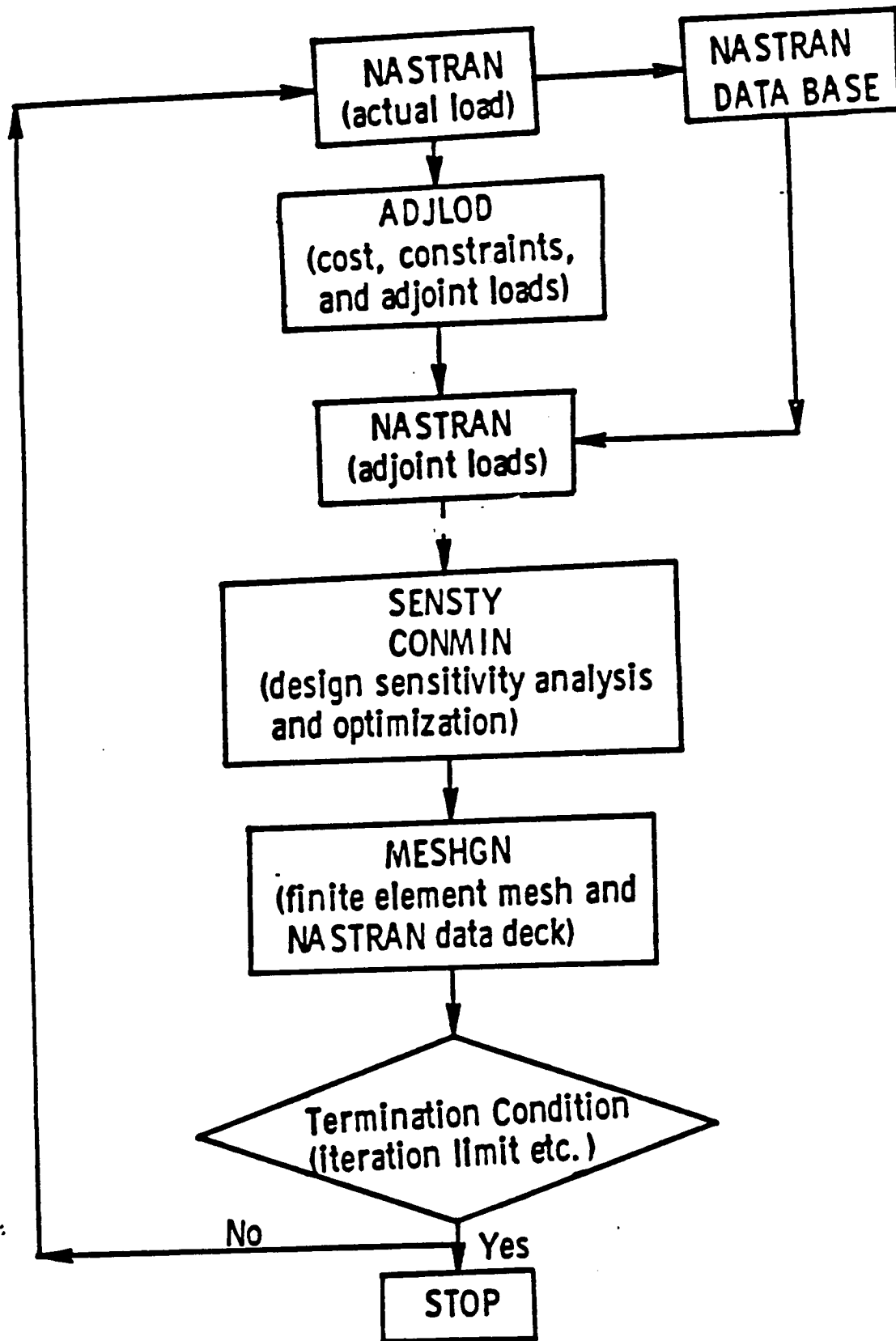


FIG. 11 Flow Chart of Modularized System

All Holes Have Radius of 1 cm

All Dimensions in cm

(i) - Design Element Number

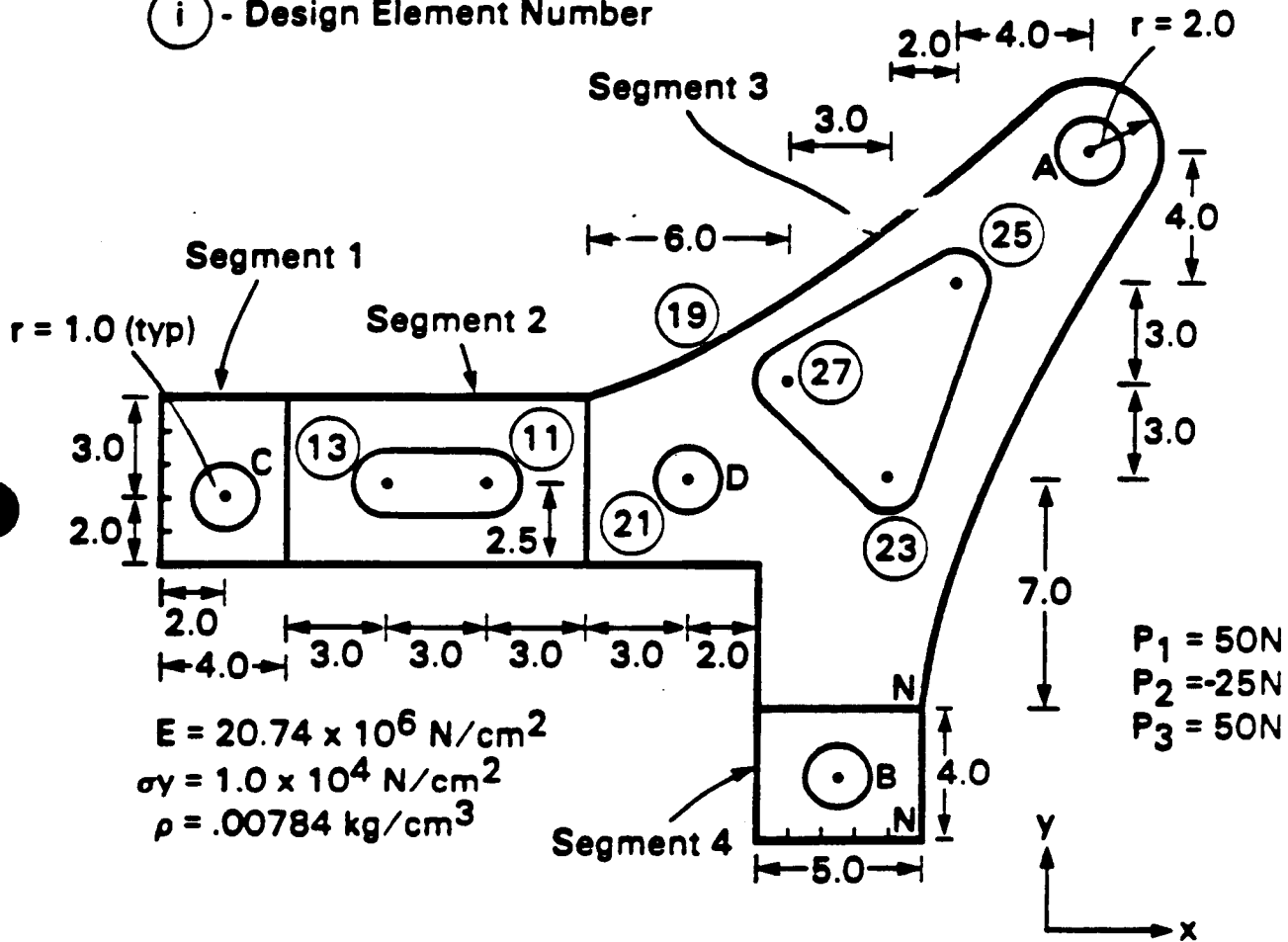


FIG. 12 DIMENSIONS OF EXAMPLE

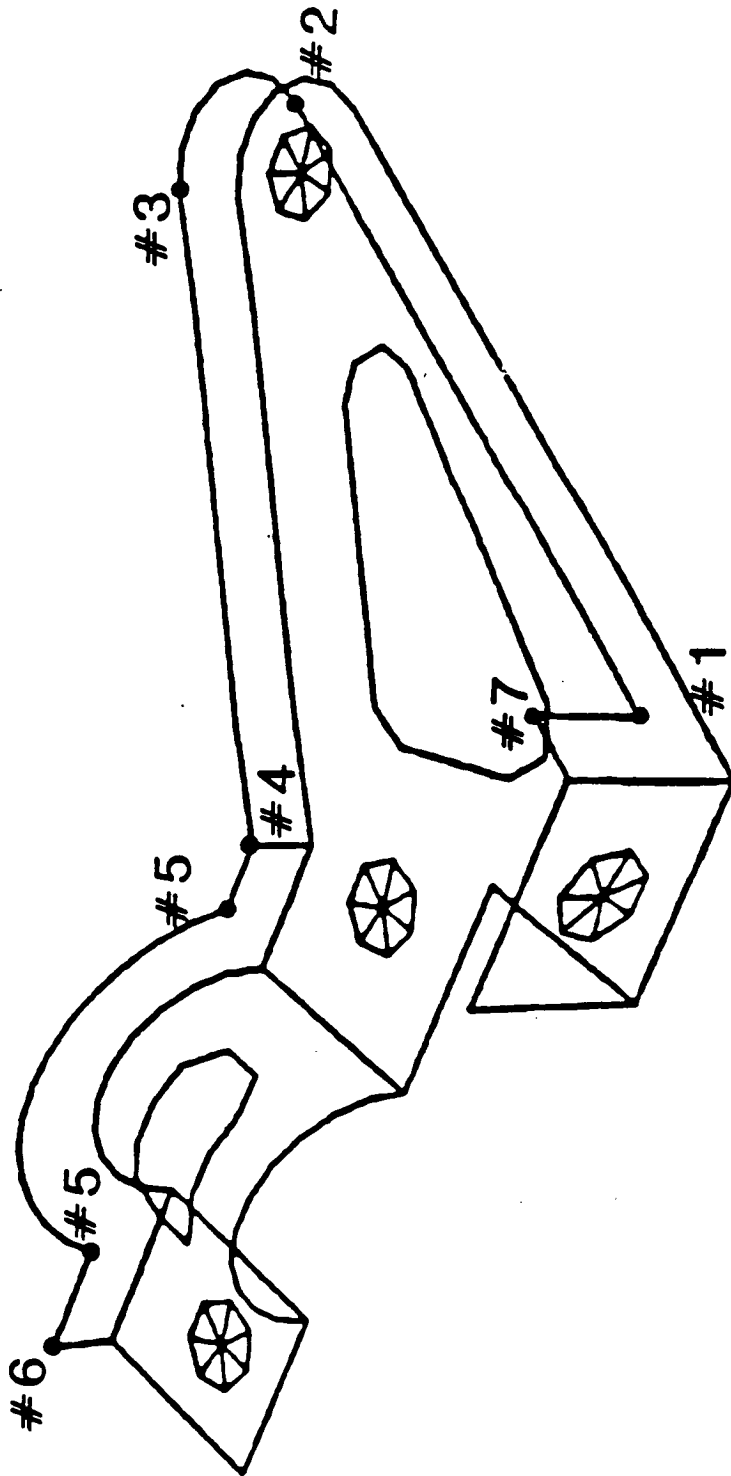


FIG. 13 Flange Design Variable Locations for Example



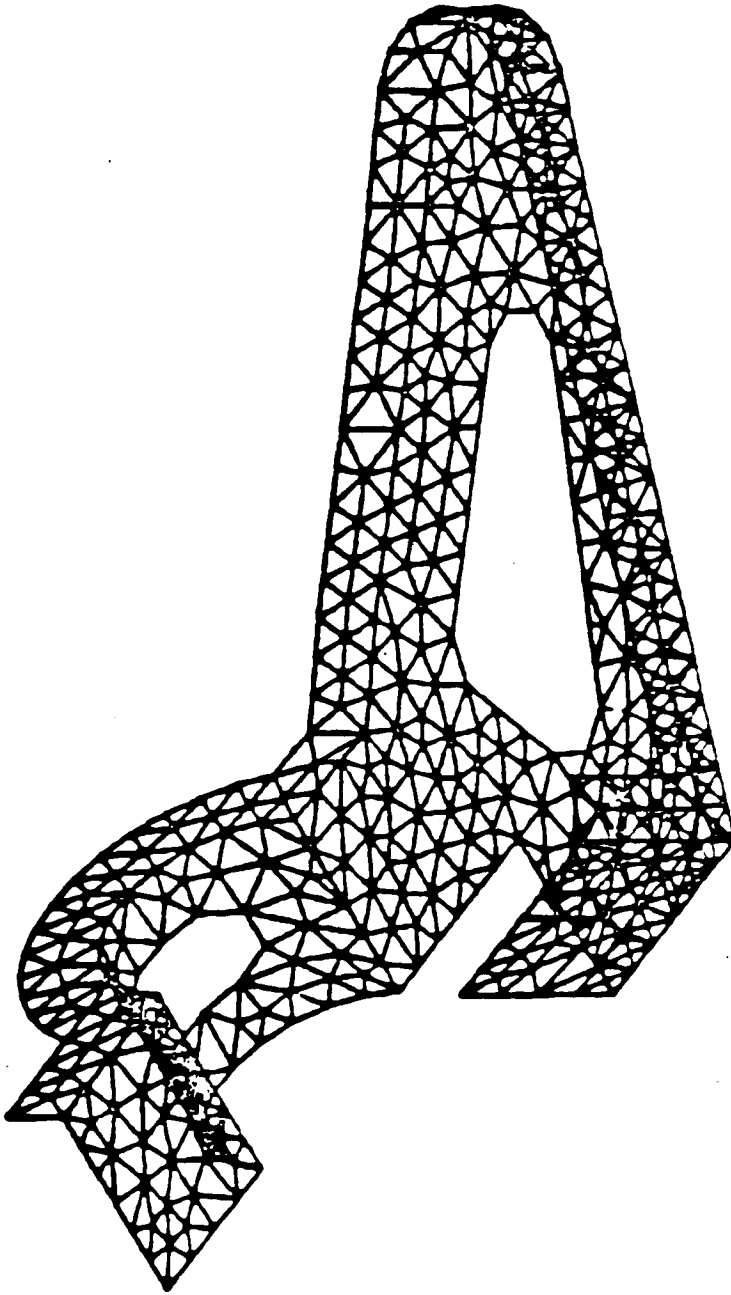
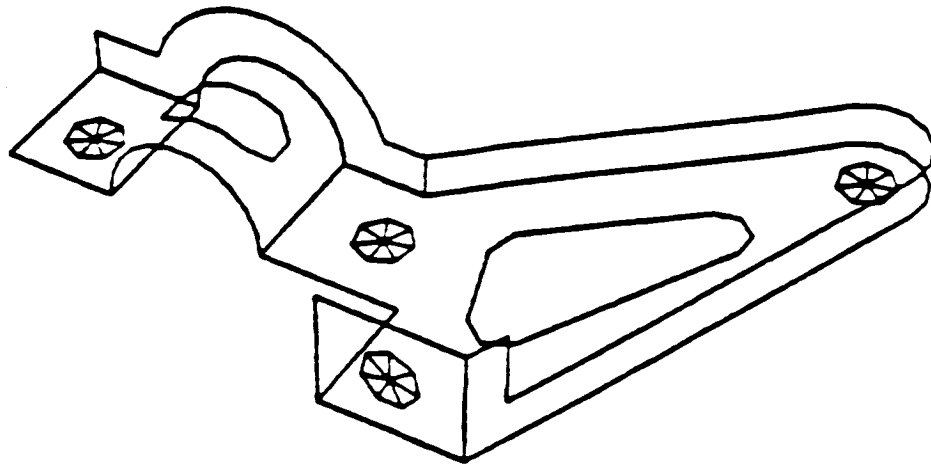
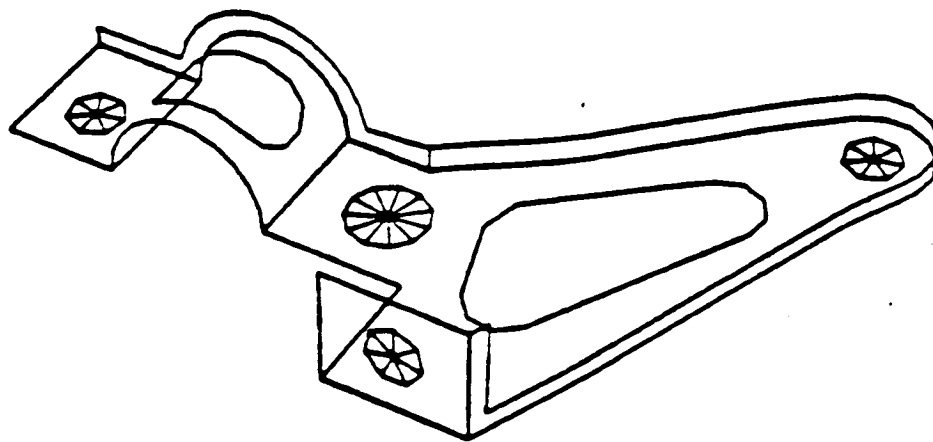


FIG. 14 Transmission Bracket Finite Element Mesh



**INITIAL DESIGN**



**FINAL DESIGN**

**FIG. 15 Initial and Final Designs for Example**

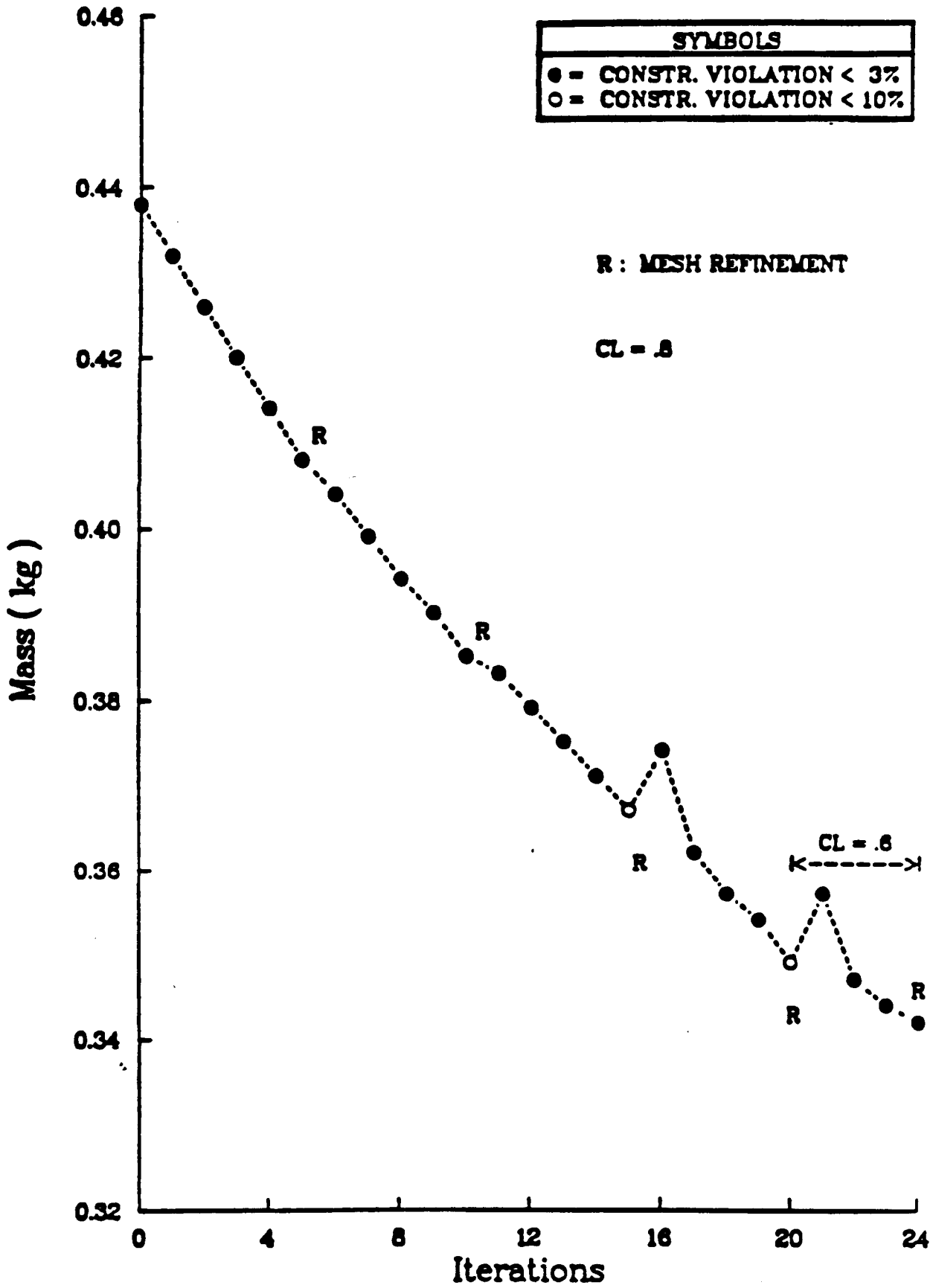
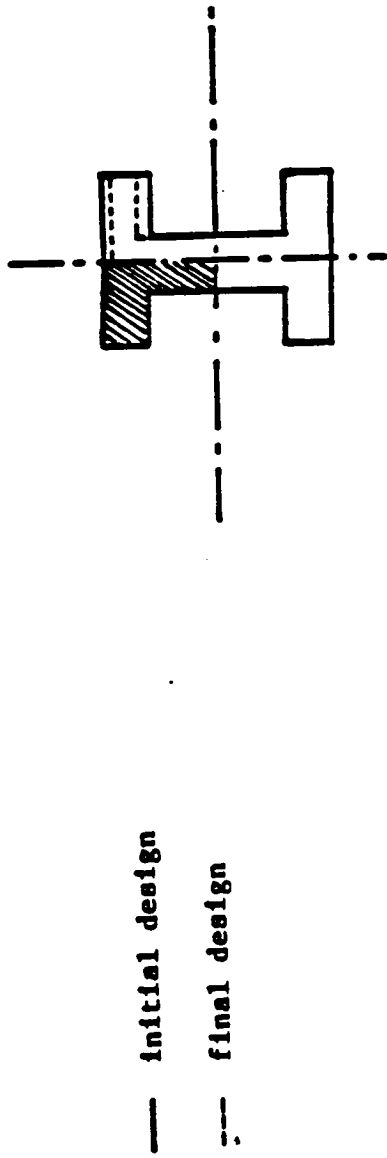
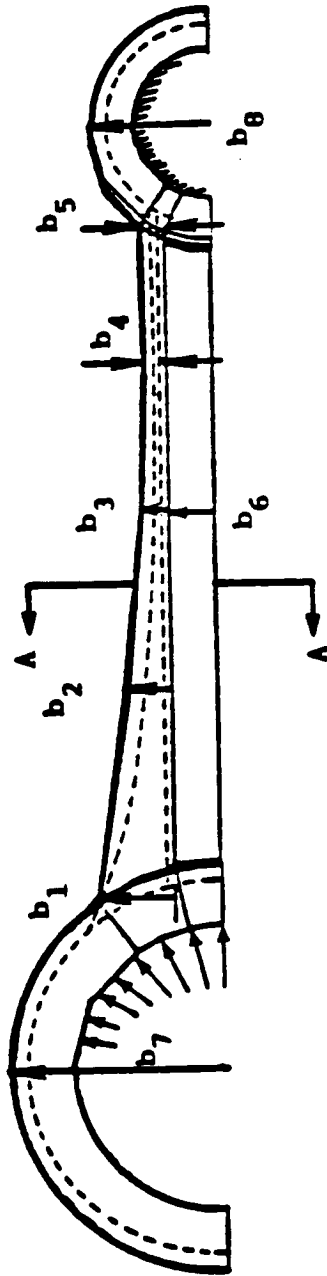


FIG. 16 Design History for Example

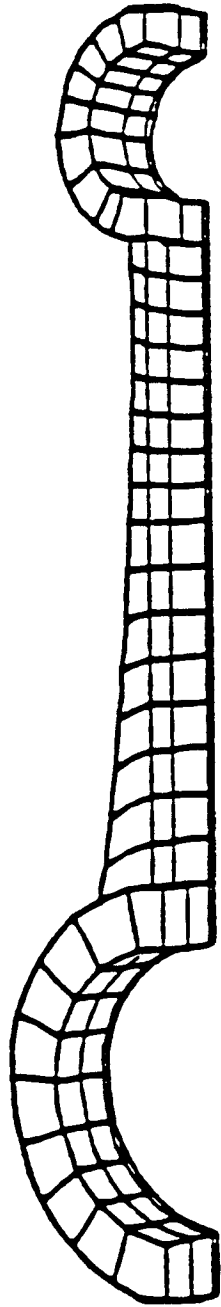


section A-A



Note: b's are design variables

FIG. 17 Generic Model of Engine Connecting Rod



**FIG. 18 Finite Element Mesh of Connecting Rod**

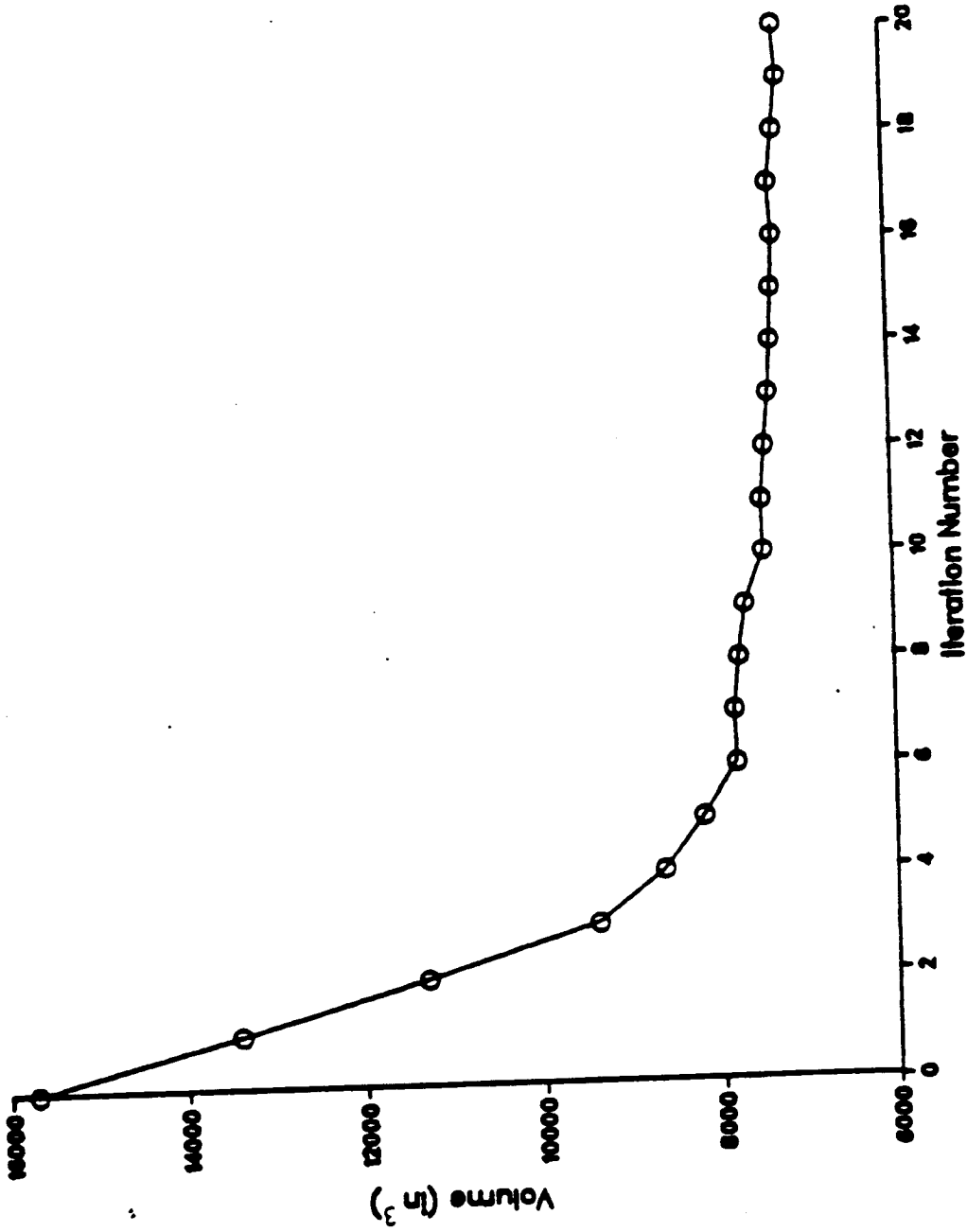


FIG. 19 Design History of Engine Connecting Rod

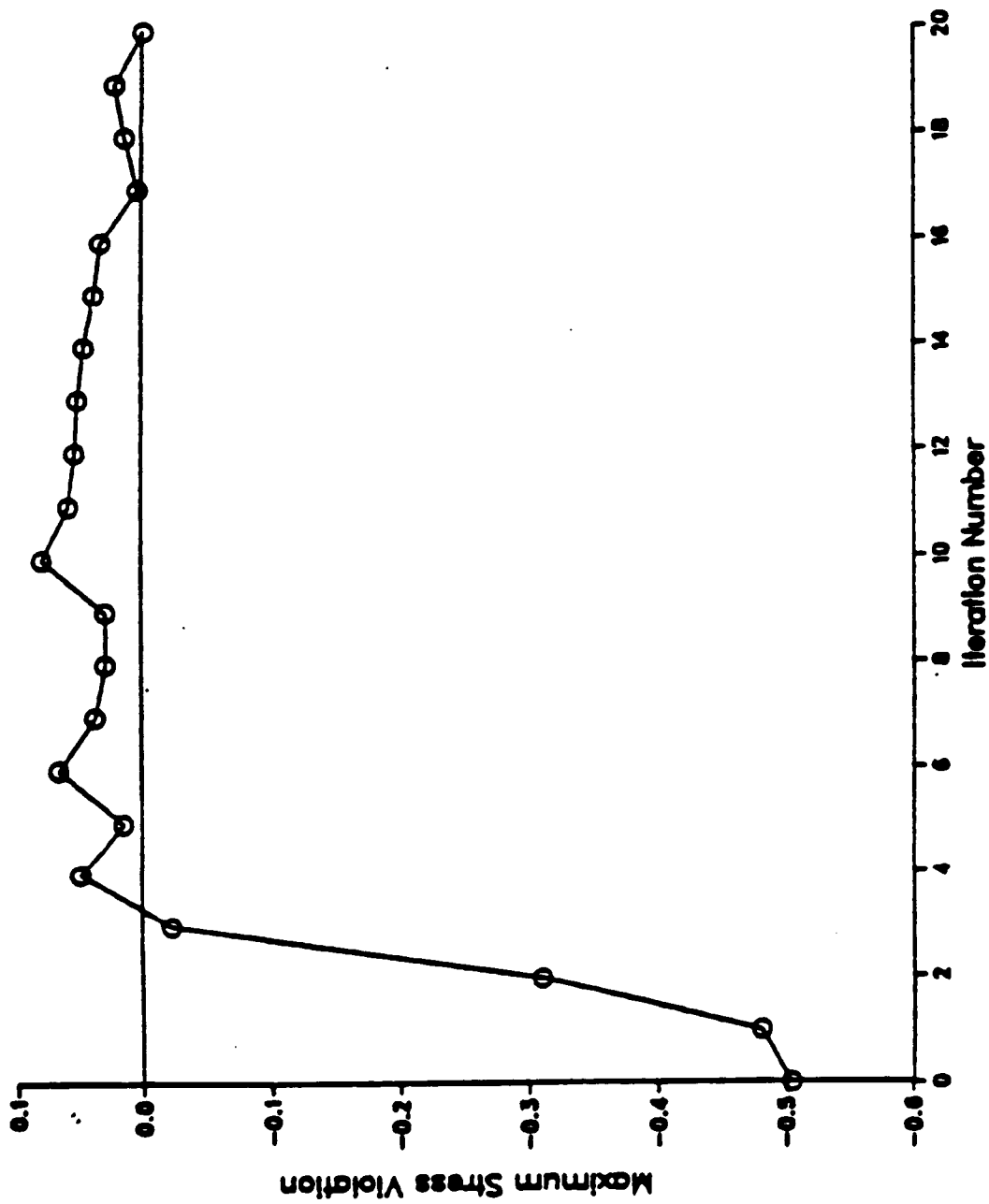


FIG. 20 Design History of Engine Connecting Rod

POSTPROCESSING TECHNIQUES FOR 3D NON-LINEAR STRUCTURES  
Richard S. Gallagher  
Hibbitt, Karlsson & Sorensen, Inc., Providence RI

#### ABSTRACT

This paper reviews how graphics postprocessing techniques are currently used to examine the results of 3D non-linear analyses, some new techniques which take advantage of recent technology, and how these results relate to both the finite element model and its geometric parent.

#### INTRODUCTION

The end result of most finite element postprocessing remains the interpretation of a single result quantity in the form of a single, static picture. Because there is a natural mapping between such plots and the increment-by-increment output data files produced in non-linear analysis, most such analyses today essentially treat individual steps and increments of non-linear analyses as degenerate cases for these linear techniques.

Current methods to view structural analysis results have their origins in display of univariate data for linear analysis. Indeed, the majority of techniques used in today's result displays came into use in the late 1970's and early 1980's, with incremental enhancements to take advantage of improving graphics display technology.

From a human perspective, areas for improvement in evaluating 3D non-linear results include improving one's insight into time-dependent behavior, rapidly finding critical behavior in complex 3D structures, and putting more result information into a given picture.

#### CURRENT POSTPROCESSING TECHNIQUES

Techniques used today in generating analysis result plots include the following:

##### 1. Deformed shape plotting

These plots showing the deformation of a structure under load generally overlay deformed and undeformed plots, with a magnification factor applied for small displacements. Hidden line removal or boundary plotting is commonly employed to reduce the visual complexity of these plots.

##### 2. Vector result plotting

Quantities which vary at points across the structure are displayed as arrow or line vectors. This technique is one of few which display the directionality as well as the magnitude of a quantity. On the other hand, such plots easily become "busy" and difficult to interpret unless applied to limited plot areas.



### 3. Contour line plotting

Like a topographic chart, lines are constructed on the surface of a structure to outline transitions between result levels. Either coded colors or alphanumeric labels are used to differentiate levels.

### 4. Shaded result plotting

In the late 1970's, when few of the color graphics devices in use could display more than 8 or 16 simultaneous colors, shaded result plotting essentially "filled in" the areas spanning contour levels with discrete colors.

Today, increased levels of color and firmware shading capabilities have made it easier to produce fully shaded plots showing the variation of a quantity in a smooth, continuous manner. Even as such capabilities become more of standard among display devices, plots using limited numbers of discrete colors remain popular - often, a discrete plot gives a more rapid overview of where critical behavior exists.

### PROBLEMS IN 3D NON-LINEAR POSTPROCESSING

While techniques such as these are commonly used to look at single-frame results in 3D non-linear results, they carry with them a number of drawbacks as currently used. Some of these include:

-Range limits within a frame. Non-linear results can vary across a large range across time steps, yet each individual step may encompass only a small part of this range.

For example, if 16 colors are used to represent the full global range of behavior in a non-linear analysis, a given step might only use one or two of these colors. But more common is the opposite problem: each frame in an analysis maps its full range of colors to the LOCAL step data, making it difficult to correlate frame-to-frame behavior after the fact.

-Loss of time perspective. Evaluating non-linear behavior by review of individual frames carries with it the same loss of insight that differentiates nodal result printouts from graphic plots.

-In 3D structures, critical results may be interior or rearward-facing relative to a 2D plot image of its results. While it is often true that critical results occur on exterior surfaces, this is not always the case - and moreover, the ability to quickly evaluate interior results increases the design complexity which can be analyzed and interpreted within a given time frame.

Limitations such as these continue to exist in the non-linear area due to a number of factors. First, it has only been in the past two to three years that graphics display devices with extended color and three dimensional display capabilities have become common. Here, the key word is COMMON - technology for

shaded and 3D display have existed since the early days of the computer graphics field, but only recently have they been available from major suppliers with the kind of price/performance relationship that would encourage common use among engineers.

Further hardware enhancements that affect this area, such as real-time 3D display of substantial models, and computer-driven animation hardware, still generally remain at the point where they are the domain of the well-funded and technologically courageous.

Second, as time progresses, we are seeing more of a "critical mass" of users in this area to influence CAE techniques.

As CAE has increased analysis productivity in general, there has been a trend towards increasing complexity in analysis. This natural progression has led to a wider interest in non-linear analysis - and often, more from design groups applying CAE for the first time to their traditional non-linear problems as well as existing CAE users expanding the scope of their activities.

Finally, CAE tools add changes to design procedure as well as increased productivity, and non-linear users have had to absorb the same existing tools as other analysis users over the past decade.

This latter point bears some explaining. While technology itself can certainly proceed in parallel for different applications areas, current acceptance of CAE makes it more possible to implement new techniques to assist result display. From the vantage point of a commercial software developer, the penetration of state-of-the-art display hardware and tools among non-linear users would not have justified advanced graphics development in the early 1980's. Today, acceptance of current CAE tools and equipment makes it economically feasible to develop more advanced tools.

#### ENHANCEMENTS TO ADDRESS 3D NON-LINEAR PROBLEMS

There are a number of areas which can be pursued to address improved productivity for non-linear analysis work. Some techniques that look attractive because they can provide more informative displays to cope with the larger data output of a non-linear analysis are as follows:

##### 1. Translucency

Techniques to display surfaces which can be seen through have existed since early work by people such as Atherton(1) in 1981. Early scan-line based techniques in this area would sort surfaces into their requisite display pixel locations, applying a tint function to surfaces "behind" the translucent surface at a given pixel location. It was clearly limited to devices with a large number of simultaneously displayable colors.

Now that local rendering of polygons have become a common feature of graphics display devices, many devices and/or software now generate "translucent" polygons by displaying some, but not all, of a polygon's pixels, in a regular pattern.

Either approach makes it easier to make OPACITY an attribute of a result's color in a shaded result display. In this manner, interior critical results become more visible, as shown in the slide figure.

The technique has two apparent drawbacks: multiple layers of translucent surfaces may still obliterate the view of opaque results unless a very fine pattern of translucency is used, and such a technique requires display processing of interior surfaces which would normally be discarded in opaque processing.

## 2. Auto-clipping

This technique is also useful in looking at interior results. Here, hardware or software Z-clipping is used to remove surfaces which obscure the view of critical results. This is done by positioning the front and/or back clip plane at the first Z location where a critical result value is detected.

The slide figure example shown was performed using hardware Z-clipping capabilities in a Tektronix 4129 display system.

## 3. Animation

Two kinds of animation are clearly of interest in non-linear analysis: progressive display of incremental results data, and animation of a final state of behavior from rest. Slide figures show examples of animation frames for two engineering models.

Currently, many display devices allow what could be called "segment animation", where separate frames of animation are loaded into separate "segments" of display memory and then cycled through in sequence.

This technique is particularly effective on real-time 3D display devices, as it allows the user to dynamically adjust the view of a deforming or changing model. Unfortunately, such techniques have crude refresh rates in many cases, and severely limited display capacity at present in all cases.

More promising in the longer term is frame-by-frame animation, where individual frames of animation are computed, displayed, and then captured under software control on a medium such as film or videotape. Frame-by-frame capture hardware does exist today with media such as videotape and interactive read/write videodisk, but is very expensive, disjoint and lacks any unified vendors aimed at the engineering market.

Of further interest downstream is path of motion control for structural models as rigid bodies, for better visualization. Techniques exist at a practical level today, with primary issues being acceleration, deceleration and continuity of motion across

changes in path. Reference (2) is one of a number of examples to further codify these kinds of motion.

The technology behind engineering animation is well in hand, and more limited by commercial hardware availability than anything else at this point. In the author's opinion, animation will become a major factor in CAE once low-cost frame-by-frame capture and display equipment exists which is supported by major CAE hardware and software.

#### 4. Correlation of result plots with history data

Most discussion to this point has centered on model-based result plotting. Equally important in this application area is history data - plotted or printed output of result variables versus time or each other.

Graphics alone do not suffice in the engineer's determination of structural behavior. As stated in a recent issue of the Engineers' Digest in the UK (3), "While graphics have resulted in increased acceptance of the (FEA) technique, it is the print out that provides the proof to the purchaser."

In designing an interactive display package, an emphasis must be placed on managing the duality between model and history data - particularly in making it easy to select history data based upon what is noticed and selected from model result plots.

#### 5. Management of display data across steps

As mentioned earlier, the potential differences between local and global result ranges in a non-linear analysis require an intelligent approach to the use of color. Techniques under study include specification of macro versus micro color levels, as well as taking advantage of displays with larger numbers of simultaneous colors to modify display ranges locally via the color table.

Many of these techniques are still being evaluated at an experimental stage at present. A key component of the above efforts is the ability to tie directly into the database of an existing non-linear analysis package to manipulate the large amounts of data involved in input to these and other display functions.

### THE RELATIONSHIP OF POSTPROCESSING DATA TO GEOMETRY AND FINITE ELEMENTS

To this point, we have primarily discussed postprocessing data as it relates to finite element level displays. For a large percentage of current ABAQUS users, this finite element model is created at least in part due to operations on a geometric model. Commonly, a solid modeling or other CAD system is integrated with a finite element modeler for creation of the analysis data.

Upon completion of the analysis, the question remains of how -

or, in fact, whether - to relate this information to this geometric database. Often, the geometric data is available across design disciplines, while its finite element model is specific to the individual analysis group.

In this era of automated adaptive analysis, meshes change - while geometry, generally, does not. The end result of an analysis is a state vector expressed at points which correspond to parametric or spatial locations in this geometry. In theory, the finite element mesh itself need not even remain as permanent data.

Some practical considerations interfere with this concept, however. The purpose of saving analysis data is to display or interrogate it later. Given the largely polygon-based methods of model result display, a polygonalization of some form will generally be required for graphics display - with preservation of this mesh data being an ideal polygon representation in most cases.

Furthermore, direct association of results with geometry removes a link to re-starting or replicating the analysis data from its final state - although, arguably, initial conditions alone combined with the same adaptive meshing approach would allow a reproduction to this point in theory.

Currently, most solid modeling systems which integrate FEM capabilities treat analysis results as purely an attribute of the mesh. While it of course relates ultimately to the geometry itself, both display and analysis techniques in use today clearly point to a representation where the geometry is the parent of the mesh, and the mesh is the parent of the results, with both parts of the linkage remaining intact.

However, at a database and user interface level, more work clearly needs to be done to make this linkage transparent to the user. Ideally, the user would rather not create or care about the finite element mesh en route to the overall goal of evaluating geometric behavior. While numerous obstacles remain on the way to this goal, the longer term goal is to eventually make this the level at which the user operates in postprocessing.

#### CONCLUSION

While graphics display techniques have done much to increase insight into non-linear 3D structural problems, these problems contain unique display issues which are not completely addressed by current techniques. Approaches such as translucency, clipping, animation and management of color have potential to increase understanding of these phenomena further. Moreover, in time these results must be treated to the user's view as an attribute of the user's primary medium of exchange, the geometric model itself.

## REFERENCES

1. Atherton, P., "A Method of Interactive Visualization of CAD Surface Models on a Color Video Display", ACM SIGGRAPH '81 Proceedings, August 1981, pp. 279-287.
2. Wilhelms, J., "Towards Automatic Motion Control", IEEE Computer Graphics and Applications, v.7 n.4, April 1987, pp. 11-22.
3. "Stress Analysis - Potential and Problems", Engineers' Digest UK, Dec/Jan 1987, p. 37.
4. Gallagher, R.S., "The Computational Laboratory Concept", SAE Technical Paper no. 850786, April 1985.

SLIDES:

- Review of current postprocessing techniques
  - Geometry-based
    - Deformed shape plots
    - Vector plots
    - Contour line plots
    - Discrete fringe contour plots
    - Continuous tonal plots
  - Result-based
    - XY plots
    - 3D data surfaces
  
- Some newer techniques
  - Translucency
  - Auto-clipping
  - Animation
  
- Results as a sub-level to the mesh, as a sub-level to the geometry

N88-19125

514-61  
125800  
318

GEOMETRIC VERSUS FINITE ELEMENT MODELING  
CURRENT AND FUTURE TRENDS AT NORTHROP

Shiv K. Bajaj  
Systems Technical Specialist  
NCASA Development  
Northrop Aircraft Division  
Hawthorne, CA 90250

ABSTRACT

Engineering Automation at Northrop encompasses the various design and analytical phases of air vehicle development. Design systems addresses automation of engineering/tooling design and computer-aided manufacturing processes. The analysis systems automate aeroelastic modeling and postprocessing analysis results. These systems interface with aircraft loft and geometric entities thru localized transfer techniques. However, total integration effort based on a geometric database nucleus with peripheral design, analytical and manufacturing systems is well underway. An outline of the present and future trends is presented to help channel the RPI effort in this direction.



# Integrated CAD/CAM/CAE

---

Topic

Geometric vs Finite Element Modeling  
Current and Future Trends at Northrop

Presentation to RPI on May 12, 1987 by :

Shiv K. Bajaj

Systems Technical Specialist, NCASA Development

# Integrated CAD/CAM/CAE

---

## RPI/CAM-I Objective

To develop Functional Requirements for an Integrated  
Geometric Modeling and Engineering Analysis Subsystem  
as Part of an Overall PRODUCT MODELING SYSTEM

## Presentation Outline

- Brief Overview of Current Capabilities at Northrop
- Recommendations for an INTEGRATED System
- Suggested Phased Implementation Plan

# Integrated CAD/CAM/CAE

---

## Abstract

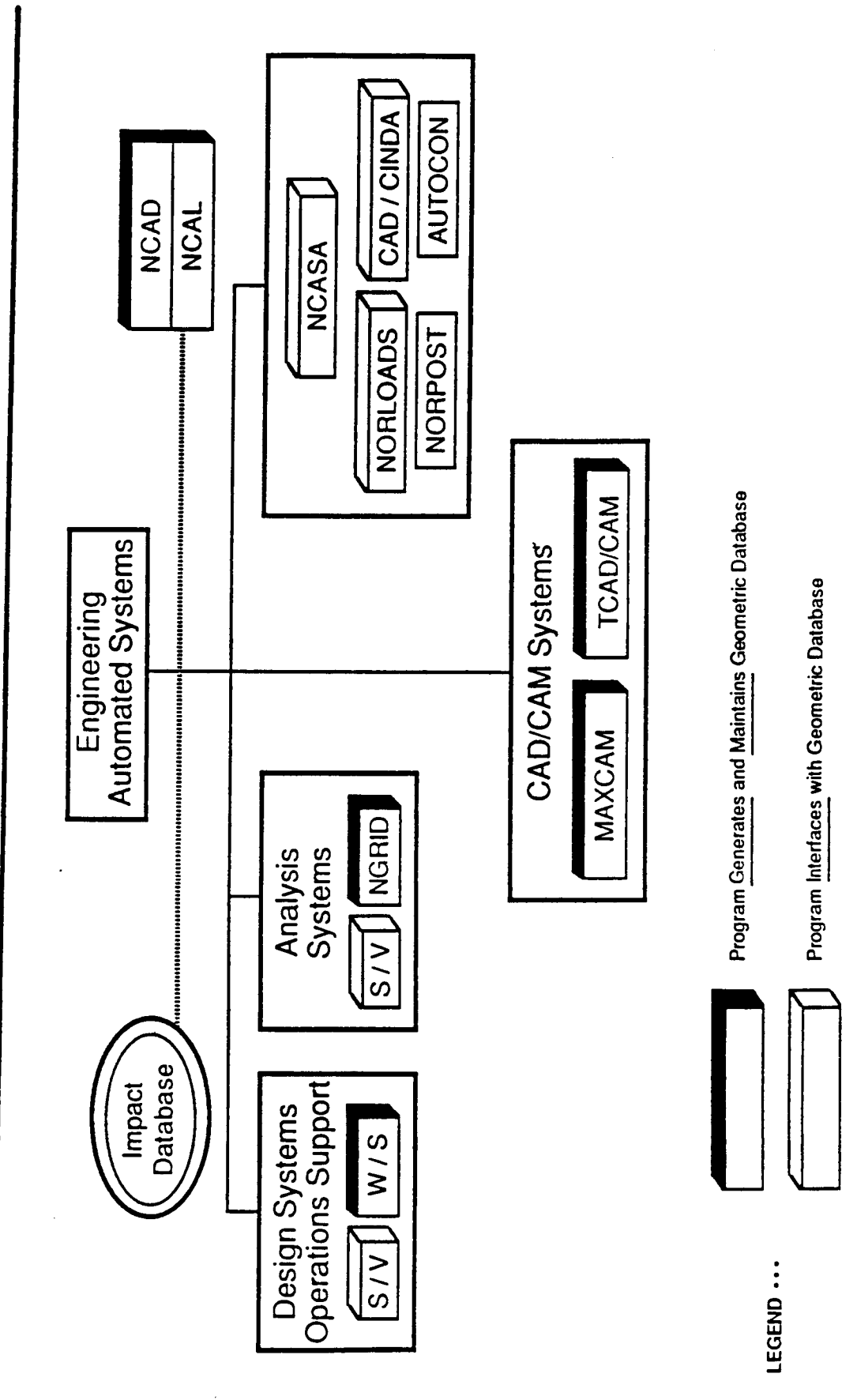
**Engineering Automation** at Northrop encompasses the various design and analytical phases of air vehicle development.

**Design systems** address automation of engineering/tooling design and computer-aided manufacturing processes.

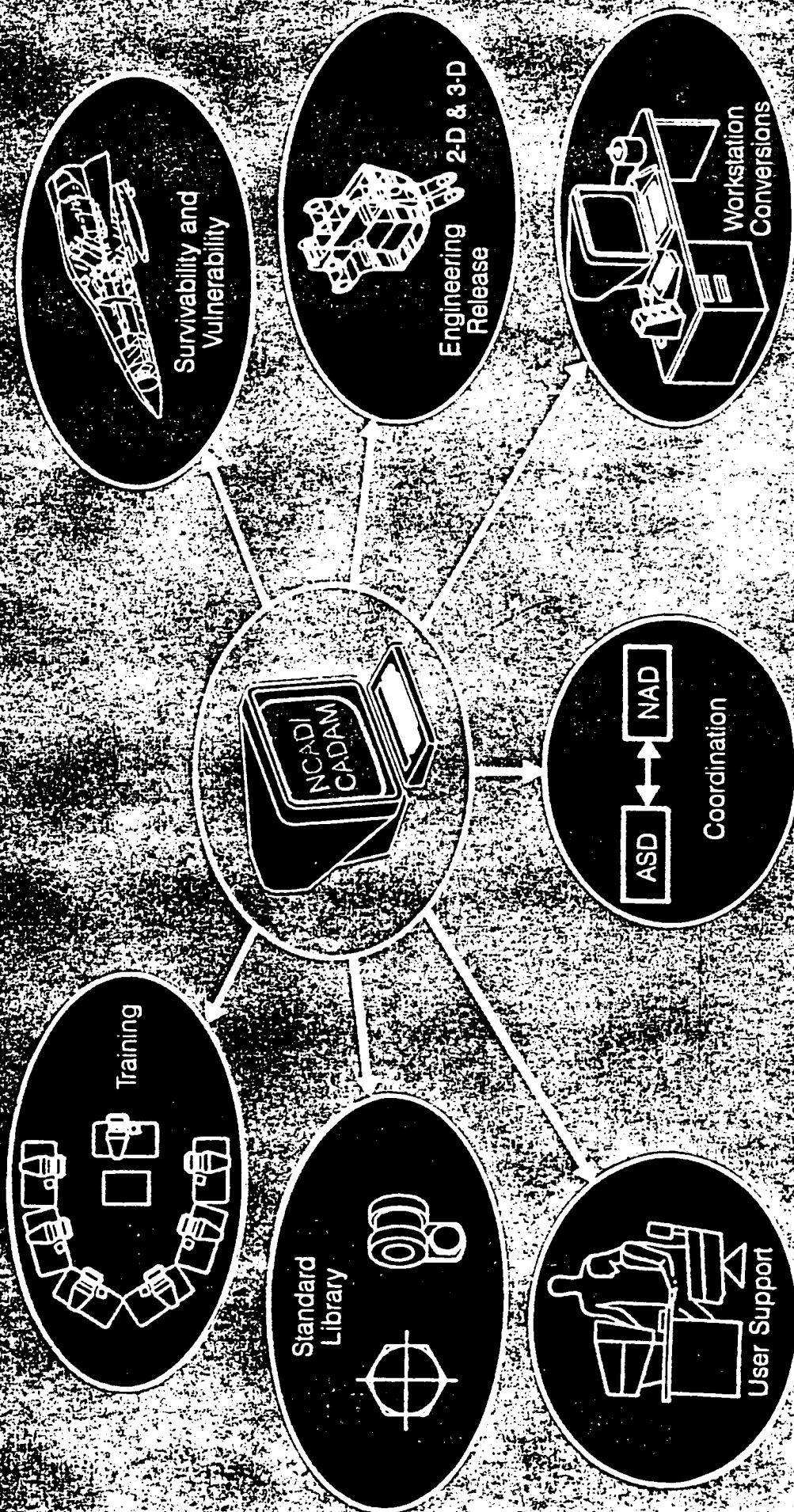
**Analysis systems** automate aerolastic modeling, analysis and postprocessing.

**These systems Interface with Aircraft Loft and Geometric Entities** thru localized transfer techniques. However **Total Integration** effort based on a **Geometric Database Nucleus with Design, Analytical and Manufacturing Systems Peripherals** is well underway. An outline of the present and future trends is presented to help channel the RPI/CAM-I effort in this direction.

# Engineering Automation Projects Requiring Geometric Interface

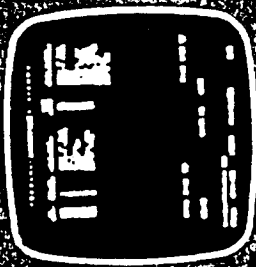


# Design Systems

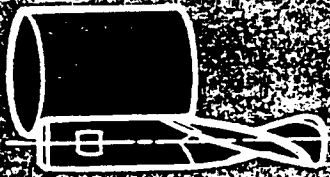


# CAD/CAM Systems

Expanded Machining Strategy



Automated Cutting Tool Selection/Definition From Pre-Released Standard Tool Library



Automated Feed and Speed Generation



Machining Tool Simulation



Master Tool Design and Fabrication



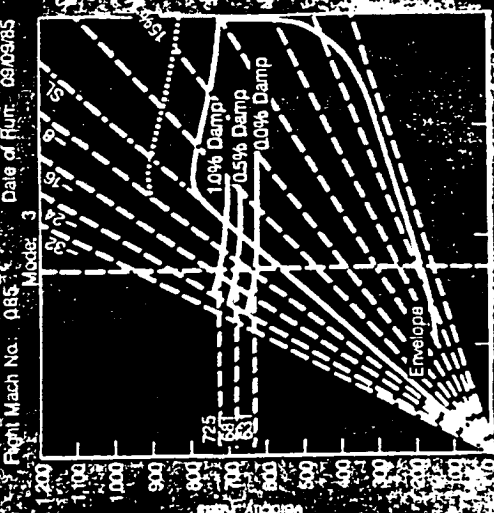
Engineering Assembly Fixtures



3D Engineering Model

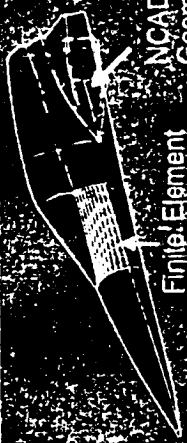
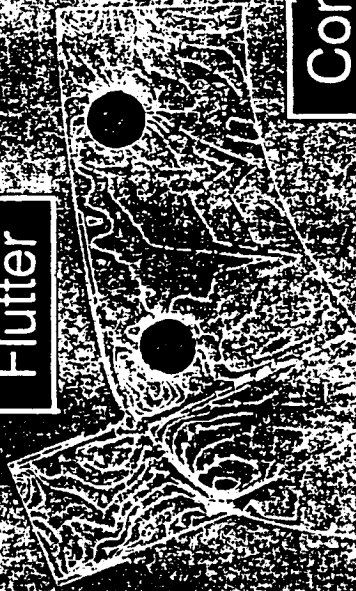


# Northrop Computer-Aided Structural Analysis (NCASA) System

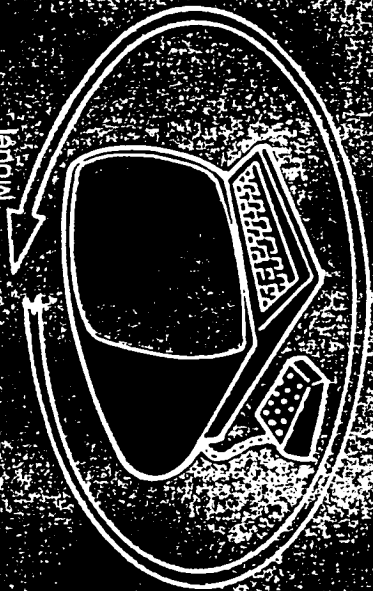


Flutter

Contour Plots



Design Interface



Analysis Programs Interaction

- NASTRAN
- Maneuver Loads
- Thermal Analysis
- Flutter Analysis



Structural Modeling

Maneuver Loads

87-20086  
10A NCAS

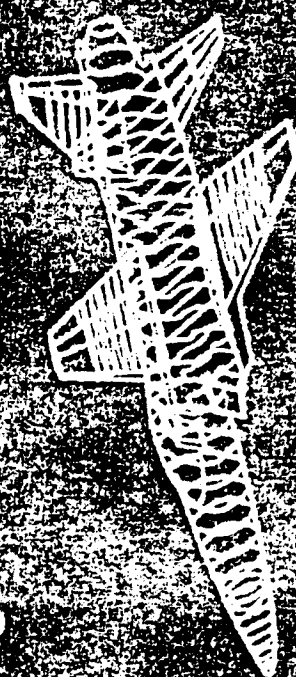


# NCASA Overview

## Design Interface

- Data Transfer From
  - NCAD/NCAL-Loft
  - NCAD 3-D Layouts
  - CADAM-Engineering Files
- Interactive Grid Generation

## NCASA Modeling

- Generate Airframe Model
  - Specify Materials and Properties
  - Identify Loads and Constraints
  - Integrate External Loads
- 
- Create Freebody Models and Fine Mesh for Local Effects

## Batch Program



# NCASA Overview (Cont)

## Batch Programs

- NASTRAN: Structural Analyzer
- MLOADS: Maneuver Loads Analyzer
- CINDA: Thermal Analyzer
- F037: In-House Flutter Analyzer

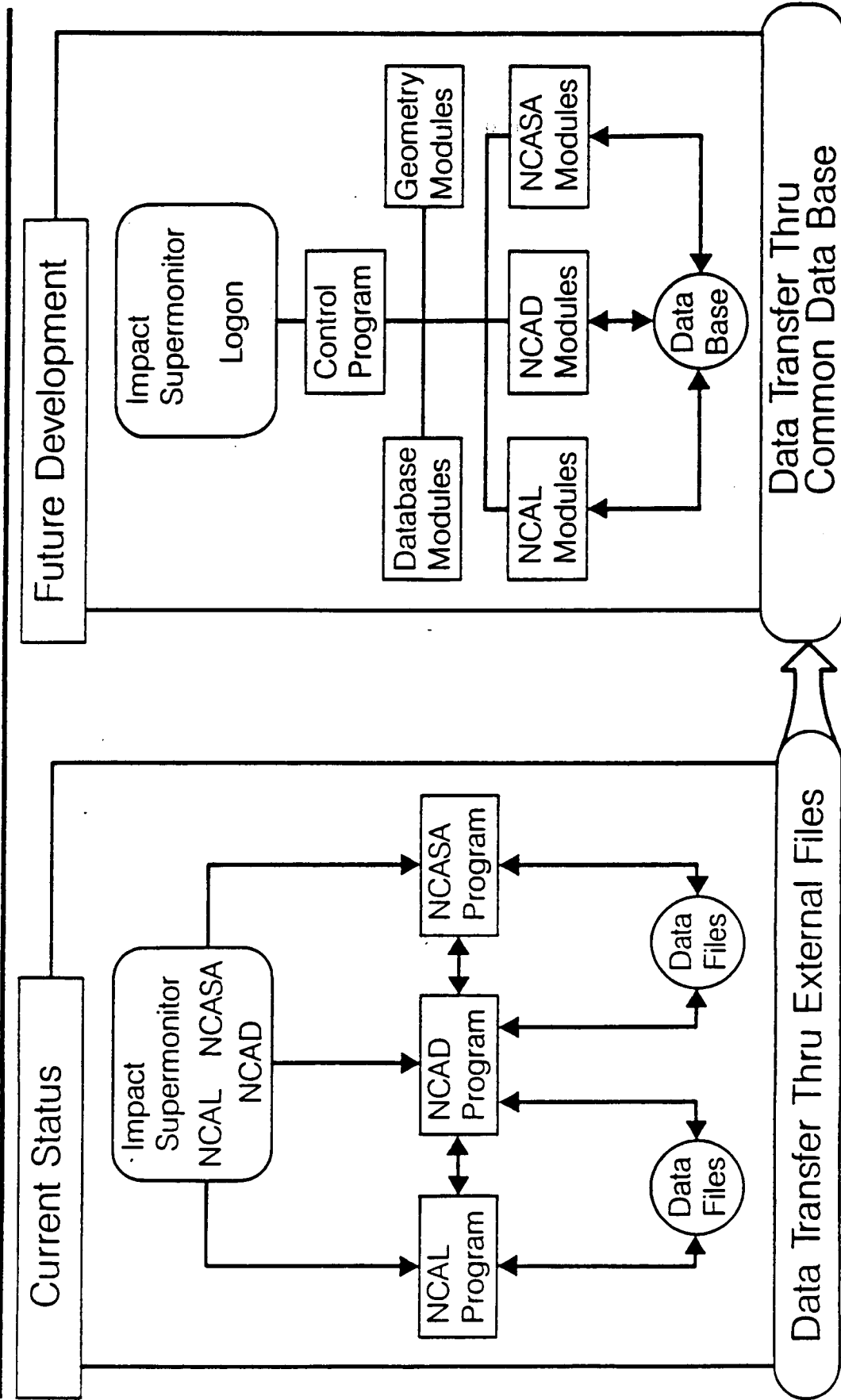
Data Base

Analysis Results

## NCASA Postprocessor

- Statics Analysis
- Dynamics Analysis
- Thermal Analysis
- NORLOADS
- Deformation, Loads and Reactions
- Element Stresses and Strains
- Vibration Modes and Frequencies
- Gust Response and Flutter Summary
- Temperatures and Heat Flux
- Aerodynamic Coefficients
- Panel Loads and Time History

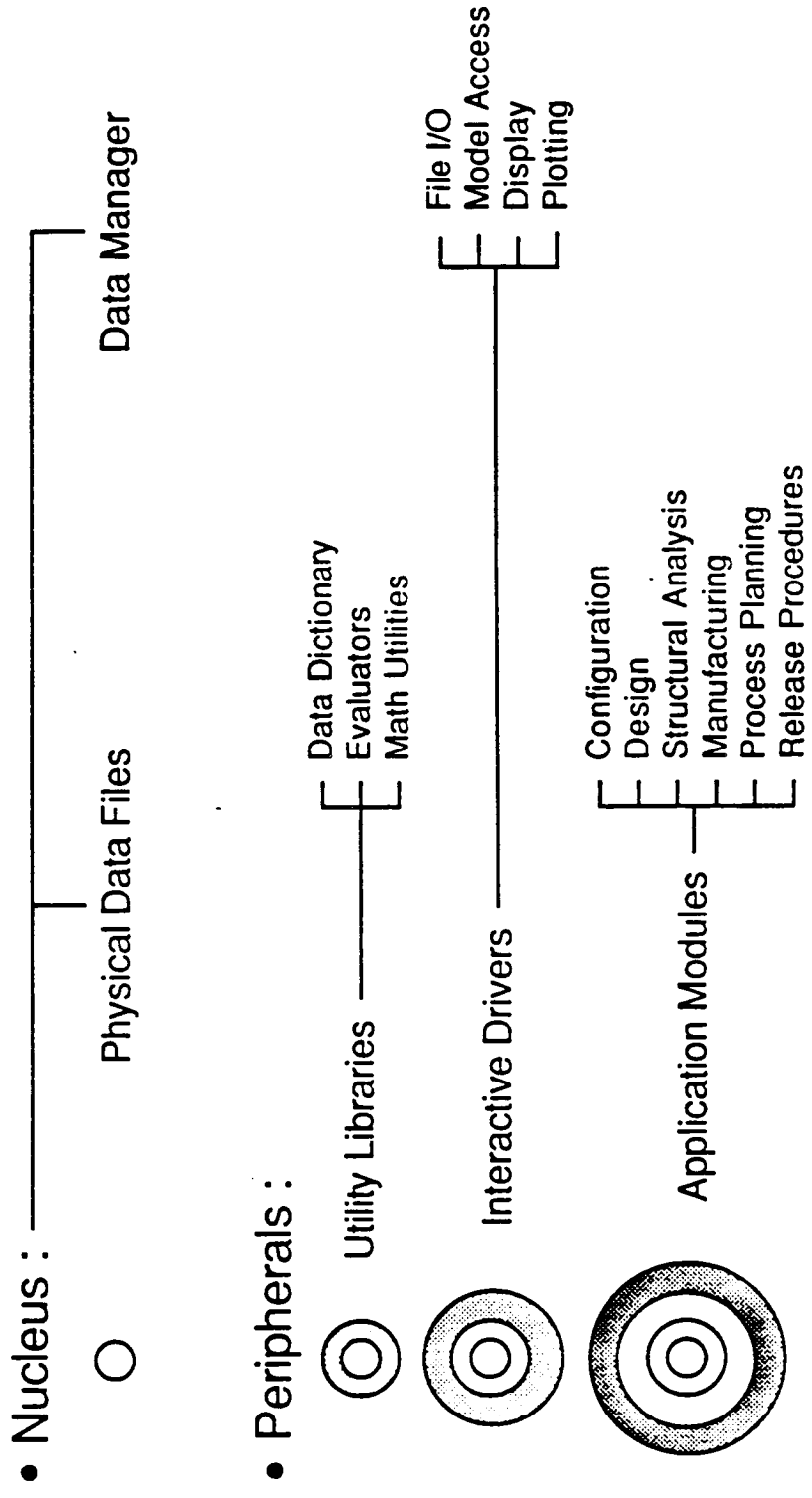
# The IMPACT System Current & Future Trends



CL-1067-86

# Integrated CAD/CAM/CAE

## Essentials



# Model Data Manager

---

## Design Criteria

- Unlimited Model Size :  
By Use of Paging and Interactive Swapping between In-Core Memory and Disk Storage
- Data Structure :  
Primitives / Non-Primitives / Attributes  
Three Level Element Data Hierarchy
- Data Access  
Through Data Servers
- Fast Search Data Structure using Correlation Value
- Self Packing Data Structure Scheme
- Dynamic Memory Management
- Ease of Incorporation of New Data Structure
- Ease of Extending Existing Data Structure Characteristics
- Compatible with Evolving Industry Standards, viz., PHIGS, PDES & IGES
- Traceability of Changes
- Portability :  
to IBM Mainframes and Workstations, Vector and Raster Graphics Tubes

# Integrated CAD/CAM/CAE

---

## Phased Implementation Guidelines

- Allow Orderly Transition from Present Geometry Systems
- Allow Upward Compatibility of Existing Models
- Implement Evaluator Driven Geometry Utilities
- Specify Milestones with Incremental Benefits

# Integrated CAD/CAM/CAE

---

## *Conclusions*

Design of an INTEGRATED CAD/CAM System Should Not be Limited to  
Geometry and Finite Element Modeling Only

*But*

It Should Also Address Various Engineering Analysis,  
Postprocessing and Manufacturing Applications

## **IDEALIZED FINITE ELEMENT MODELS**

**Mark S. Shephard  
Rensselaer Polytechnic Institute**

**Concerned with the evolution from the Augmented Model, to the Idealized Model, to the Finite Element Model.**

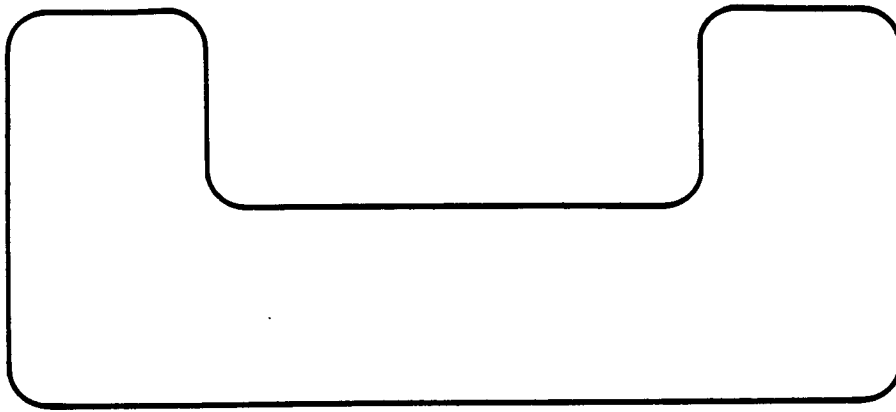
**Augmented Model - Original geometric model plus analysis attributes.**

**Idealized Model - The geometric representation plus analysis attributes that is discretized into the finite element model.**

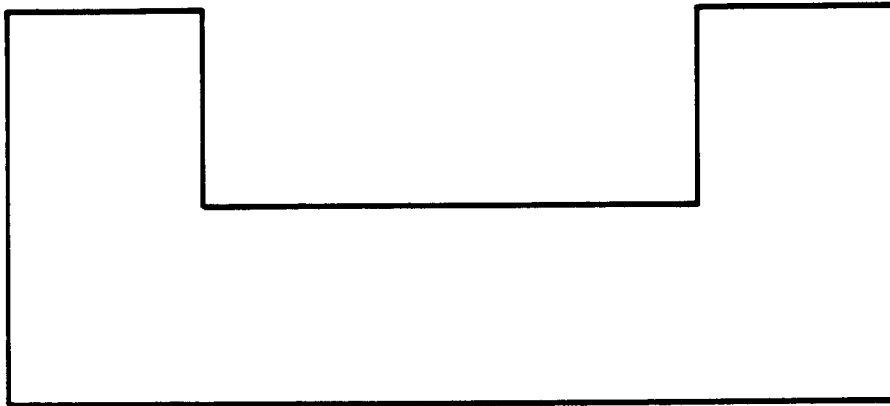
**Finite Element Model - The discrete model sent to the finite element analysis program.**

**Differences Between Augmented Model and Idealized Model**

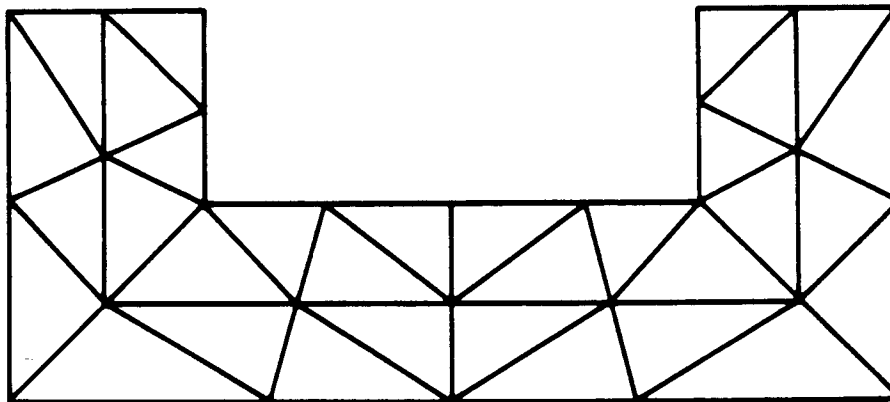
- 1. Geometric simplification - ignoring specific geometric features such as small holes and fillets.**
- 2. Geometric Enrichment - including geometry in the numerical analysis model not originally represented in the augmented model (air around a model and zero thickness interfaces, etc).**
- 3. Geometric Dimension Reduction - Replacing portions of a model with reduced dimension entities with the eliminated dimensions represented by section properties tied to the reduced dimension elements.**



**A) original geometry**



**B) simplified geometry**



**C) finite element model**

**FIGURE 3. GEOMETRIC SIMPLIFICATION**



# CH-47D PRIMARY FUSELAGE STRUCTURE

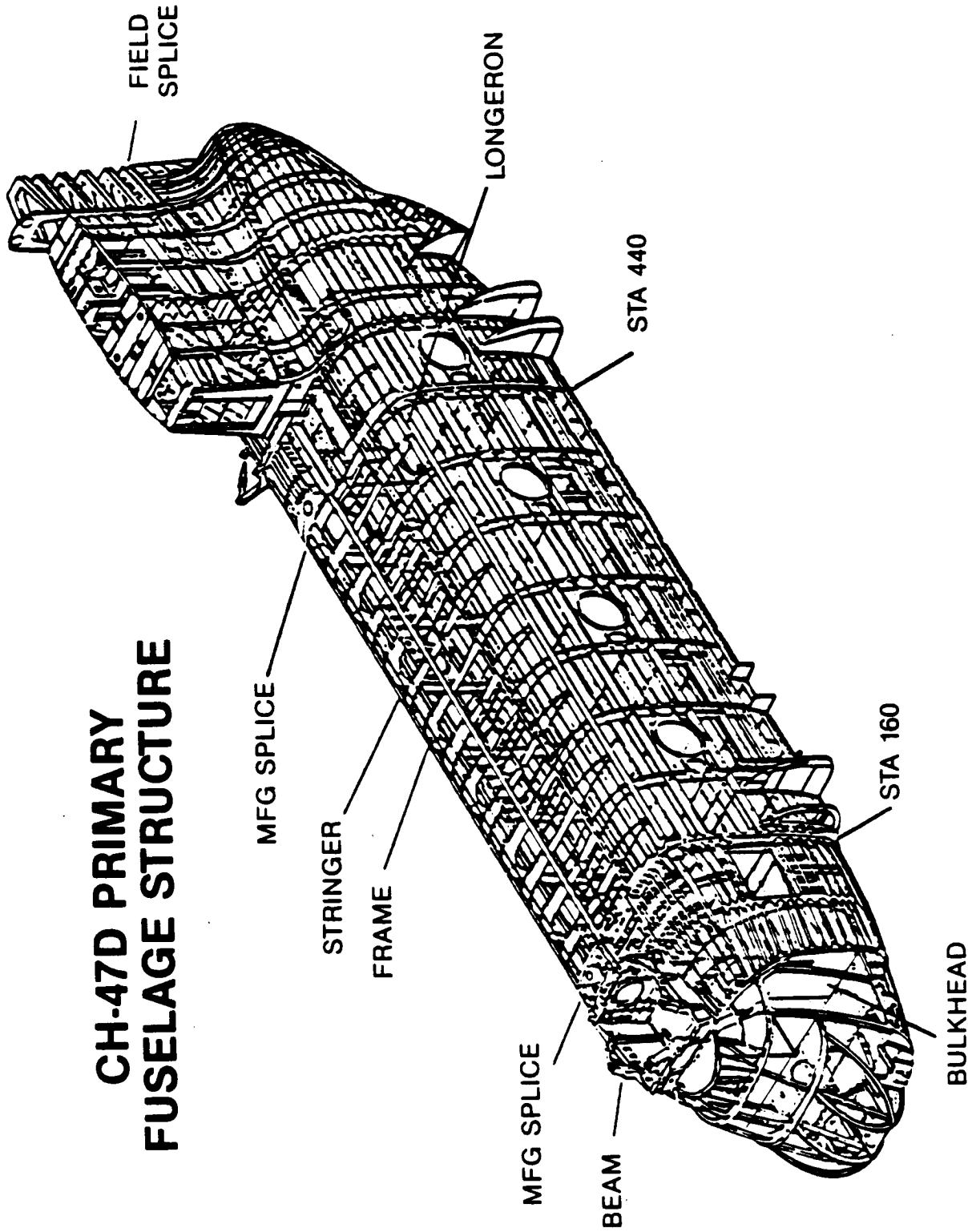


FIGURE 4. GEOMETRIC REPRESENTATION OF  
HELICOPTER AIRFRAME STRUCTURE

# STATIC MODELING

## CH-47D NASTRAN STRUCTURAL MODEL

### NASTRAN MODEL

1,883 STRUCTURAL NODES  
5,758 STRUCTURAL ELEMENTS

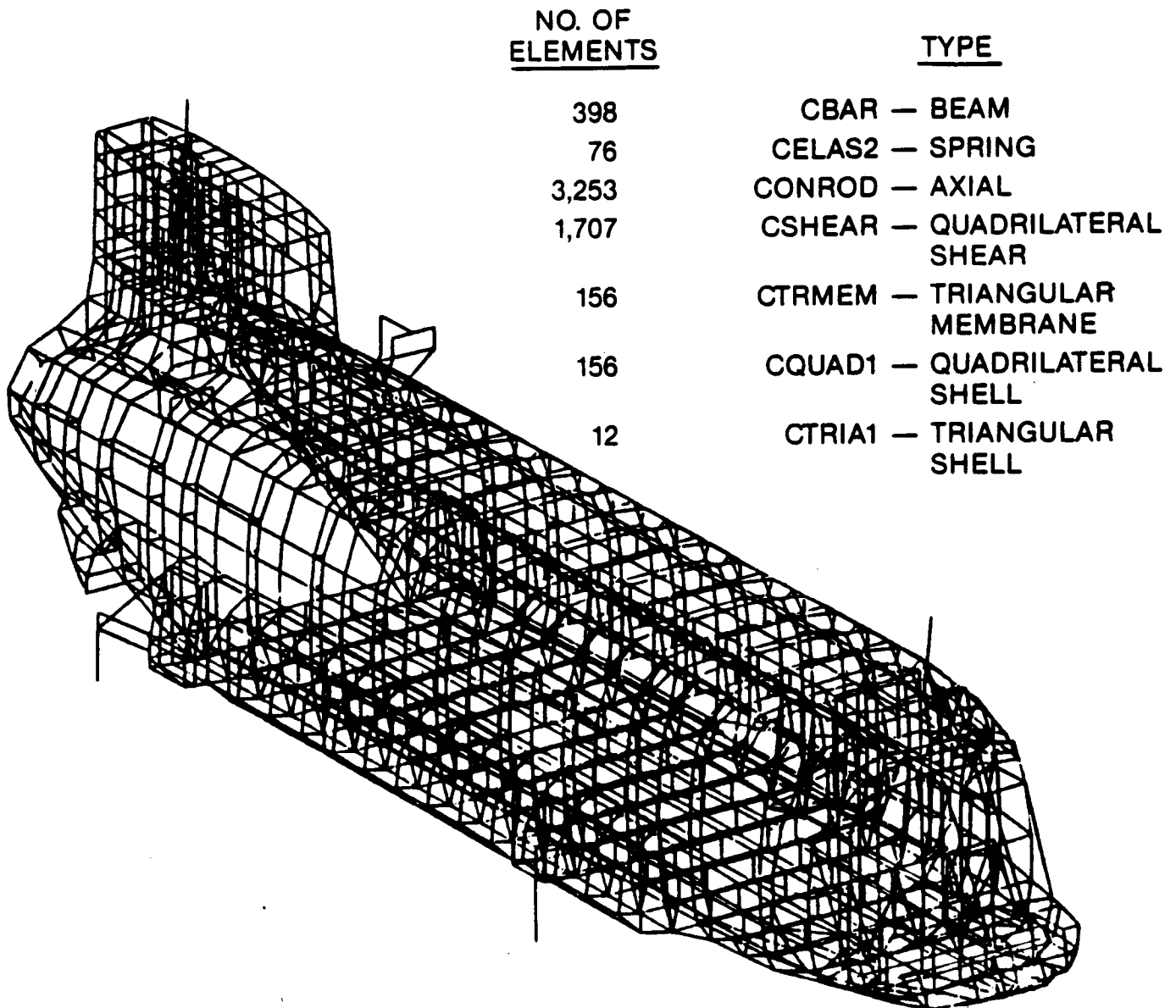
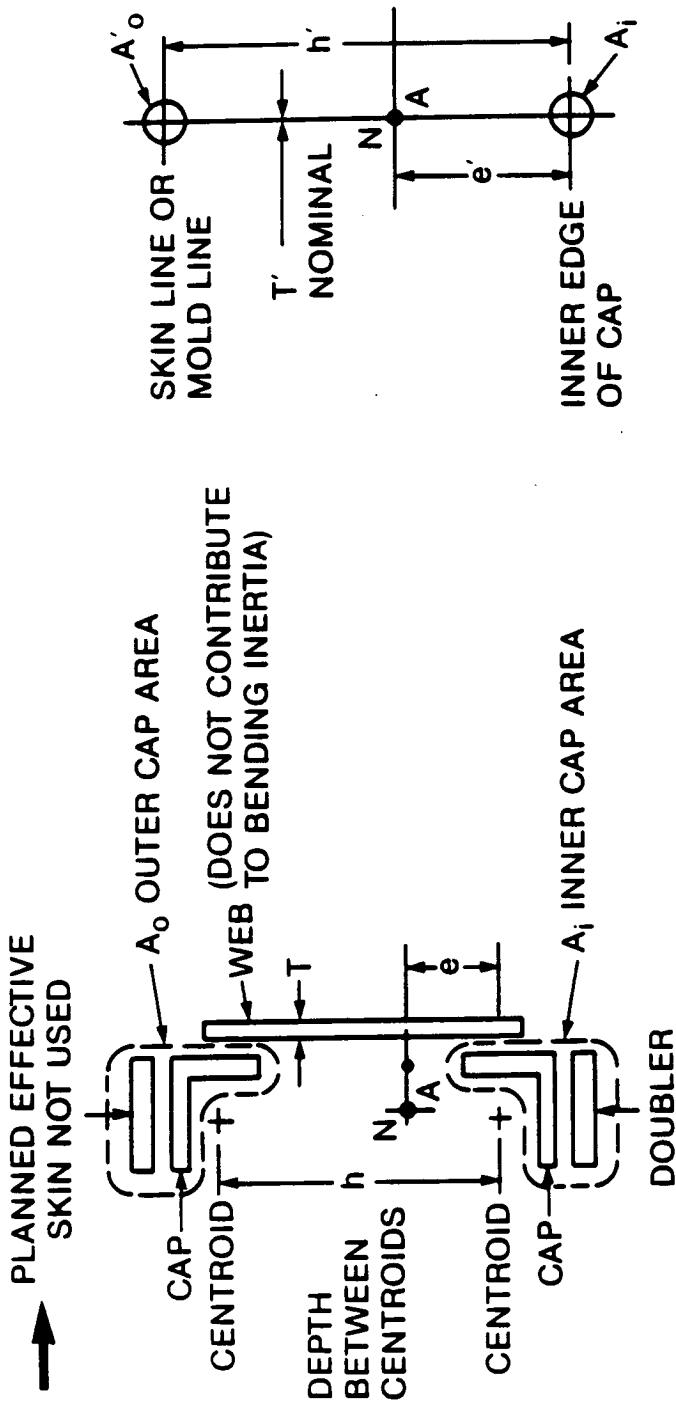


FIGURE 5. FINITE ELEMENT MODEL OF AIRFRAME STRUCTURE

# STATIC MODELING GUIDES — FRAMES DETERMINING EQUIVALENT BENDING INERTIA

REAL FRAME CROSS SECTION      NASTRAN FRAME CROSS SECTION MODEL



$$e = \frac{A_0 h}{A_0 + A_1}, \quad \boxed{I_{N/A} = A_0 (h-e)^2 + A_1 e^2}$$

$$e' = \frac{A'_0 h'}{A'_0 + A'_1}, \quad I_{N/A} = A'_0 (h'-e')^2 + A'_1 e'^2$$

LET,  $A'_0 = \left(\frac{h'}{h}\right)^2 A_0$  AND  $A'_1 = \left(\frac{h'}{h}\right)^2 A_1$

$$e' = \frac{A_0 h}{A_0 + A_1} = \frac{h'}{h} e$$

SAME AS REAL FRAME

$$I_{N/A} = \left(\frac{h'}{h}\right)^2 A_0 \left(h' - \frac{h'e}{h}\right)^2 + \left(\frac{h'}{h}\right)^2 A_1 \left(\frac{h'}{h}\right)^2 e^2 = \boxed{A_0 (h-e)^2 + A_1 e^2}$$

FIGURE 6. PROPERTY CALCULATION FOR AN INDIRECT ELEMENT TYPE

## **COMMON APPROACHES TO DEVELOPING IDEALIZED MODELS**

### **DIRECTLY DEFINE IDEALIZED MODEL**

The majority of geometric representations used in finite element modeling are defined solely for that purpose. That is the augmented model and idealized model are the same. This is an inefficient approach and does not make the best use of available technology.

### **MODIFY AUGMENTED MODEL TO BECOME IDEALIZED MODEL**

Carry out modeling operations to alter the augmented model evolving it into the idealized model.

### **TREAT IDEALIZATION INFORMATION AS NUMERICAL MODELING ATTRIBUTES TIED TO THE AUGMENTED MODEL**

Indicate what entities are to be altered and have the appropriate information automatically tied to entities in the augmented model as attribute information. The discretization procedures would then be responsible for insuring that the finite element model reflects the idealizations.

## **MODIFY AUGMENTED MODEL TO BECOME IDEALIZED MODEL**

### **Advantages -**

It is reasonably straight forward to see how this approach would operate. The user would have a first hand understanding of the modifications.

### **Disadvantages -**

The user is required to perform geometric modeling modifications manually. Could not support use of adaptive idealization procedures.

### **Technical Issues -**

**Data Structures -** should there be two identical structures for the augmented and idealized model?

**Recovery -** how does one recover portion of a model if the idealization process is changed?

**TREAT IDEALIZATION INFORMATION AS  
NUMERICAL MODELING ATTRIBUTES  
TIED TO THE AUGMENTED MODEL**

**Advantages -**

Would support the evolution to automated, adaptive techniques for developing idealized models thus potentially being more efficient and robust. Would reduce total amount of storage needed making it easy to track the modeling assumptions used.

**Disadvantages -**

Do not know how to handle such an approach fully enough at this time.

**Technical Issues -**

Idealization procedures - do not know all the idealization procedures desired well enough to try to define geometric operators to support them.

Data structures - do not fully know how to house all the possible idealization attributes in the augmented model.

Discretization - the discretization process would become more than just mesh generation in this case, must have procedures to account for model differences automatically.

**TECHNICAL AREAS IMPORTANT TO  
THE AUTOMATION OF  
IDEALIZED MODEL GENERATION**

**Attribute Data Structure of Augmented Model**

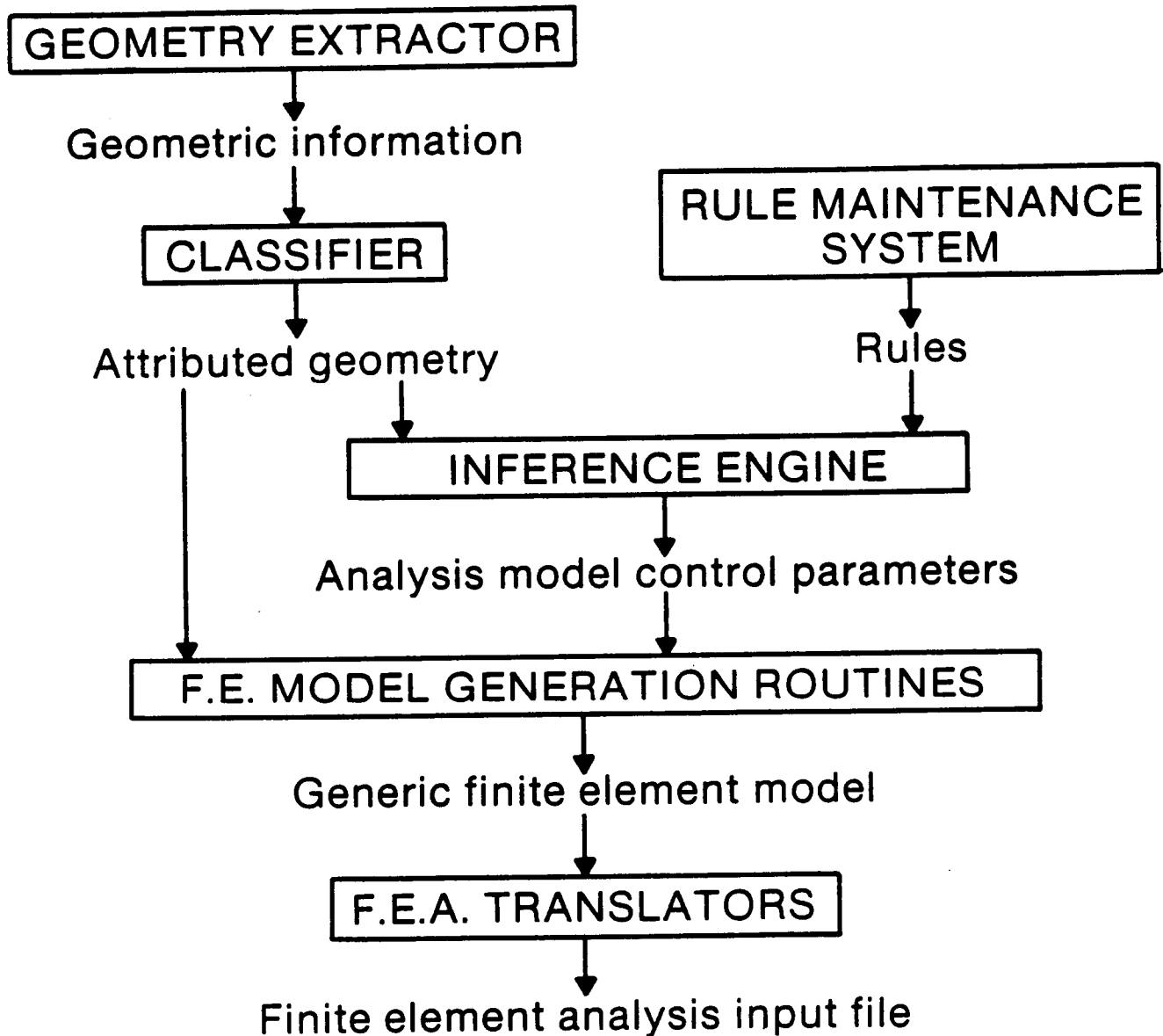
**Geometric Operators to Support the Generation of the  
Idealized Model from the Augmented Model**

**Feature Recognition Techniques**

**Knowledge-Based Modeling Procedures**

**Adaptive Analysis Techniques for Determining Idealizations**

# A KNOWLEDGE-BASED APPROACH FOR DEVELOPING IDEALIZED MODELS



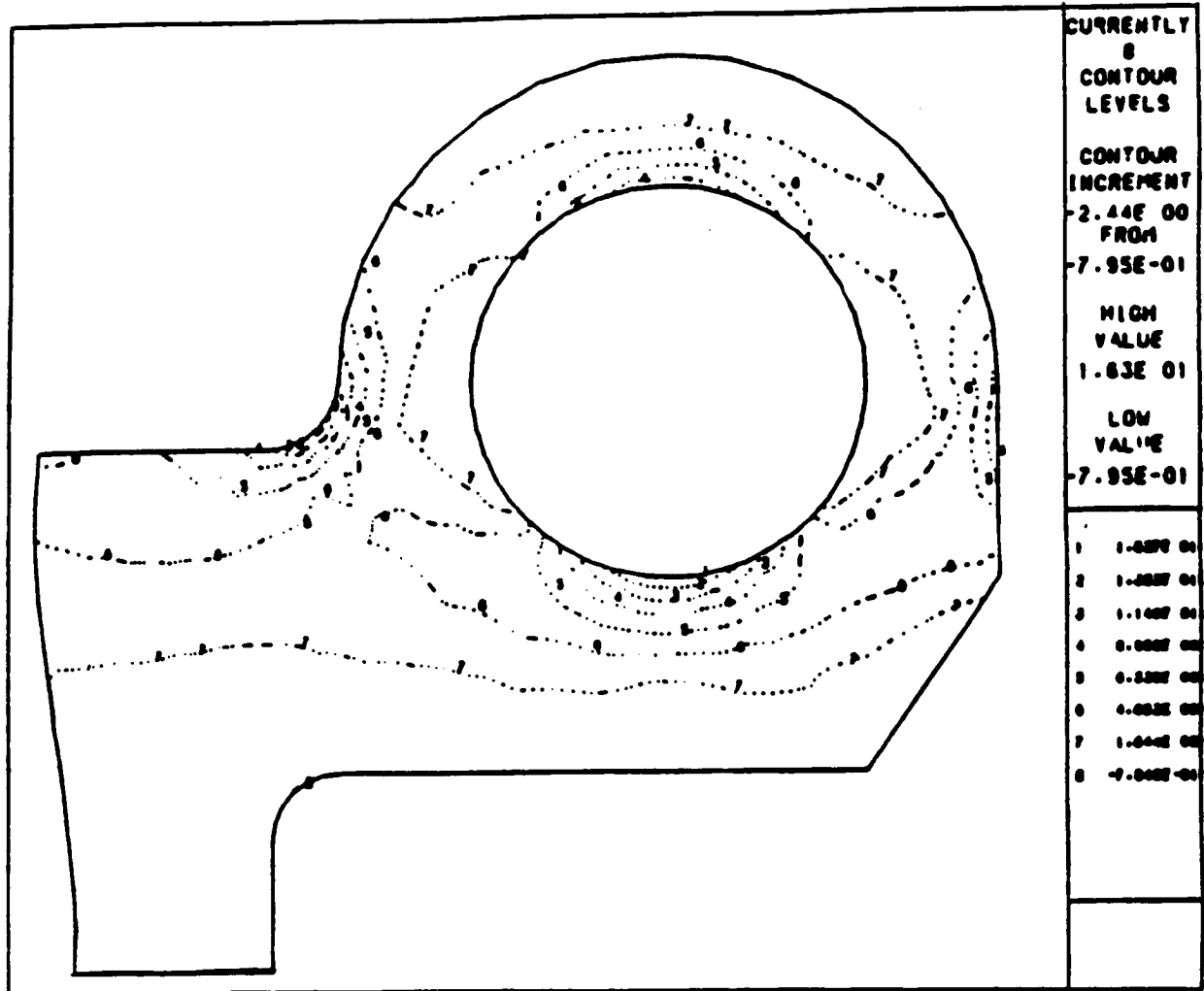


**A COMBINED KNOWLEDGE-BASED AND  
ADAPTIVE TECHNIQUE FOR  
ONE FORM OF GEOMETRIC SIMPLIFICATION:  
IGNORING CIRCULAR HOLES  
IN 2-D STRESS ANALYSIS**

**Approach -**

1. Determine candidate holes - those that are less than some percent of the net section through object at that location, and not too close to an edge.
2. Analyze object ignoring all candidate holes. This gives basic flow of loads to supports.
3. Apply correction factors to the stress at the locations of the ignored holes based on 'standard analytic' formulae.
4. Include only those holes with estimated values higher than some fraction of the limiting stress.





ORIGINAL PAGE IS  
OF POOR QUALITY

Figure 11. Stress contours with holes ignored.

ORIGINAL PAGE IS  
OF POOR QUALITY

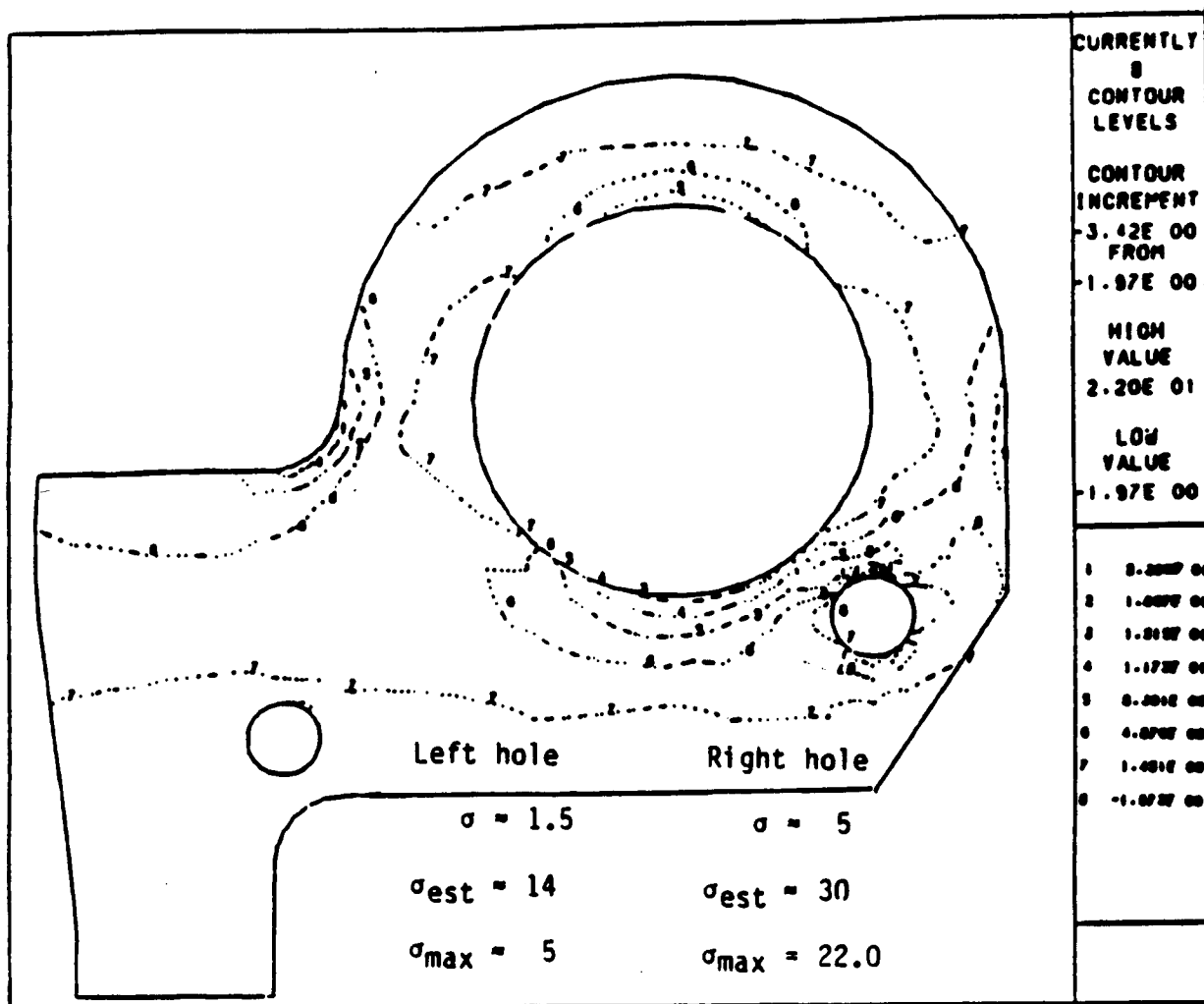


Figure 12. Stress contours with holes included.

## **BUILDING FINITE ELEMENT APPLICATIONS USING NON-MANIFOLD BOUNDARY OPERATORS**

An Approach to a dynamic interface that is a level above those discussed above. Application programs would employ both the modeling functionalities and data structures of the geometric modeling system without knowing the details of either.

This is consistent with object-based procedures that are becoming popular.

A start to such a capability employing the Radial-Edge non-manifold data structure is proposed by Kevin J. Weiler in his Ph.D. thesis for the process of defining geometric models.

A complete set of Non-Manifold Boundary Operators needed to support this approach.

## **BUILDING FINITE ELEMENT APPLICATIONS USING NON-MANIFOLD BOUNDARY OPERATORS**

### **Classes of Operators Needed -**

**Obtaining Objects Based on Type -** ability to find objects of given types.

**Determining Object Adjacencies -** find how an object is related to others of a given type.

**Geometric Interrogations -** determine a geometric property of an object.

**Attribute Interrogations -** determine the attributes of an object.

**Attribute Assignment -** tie attribute to objects.

**Geometric Modification -** carry out a geometric modeling operation based on a given set of objects.

# **BUILDING FINITE ELEMENT APPLICATIONS USING NON-MANIFOLD BOUNDARY OPERATORS**

Typical Objects -

Topological entities

Geometric entities

Attributes

The topological entities represent the 'glue' needed to hold such a system together, however this can be transparent to the applications built on it.

The approach is in a very early phase of investigation. It is not clear if it will work.