Automatic Mesh Analysis Technique By Knowledge-Based System

Hyo John Lee[1]

UUCS-89-001

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

January 10, 1989

[1]Thlis work was supported in part by DARPA (DAAK1184K0017 and N00014-88-K-0688). All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

Automatic Mesh Analysis Technique By Knowledge-Based System

Hyo Jong Lee [1]
Computer Science Department
University of Utah
Salt Lake City, UT 84112
Technical Report Number UUCS-89-001

January 10, 1989

[1]* This work was supported in part by DARPA (DAAK1184K0017 and N00014-88-K-0688). All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

Contents

List of Figures

List of Tables

Abstract

The finite element analysis technique has been recognized as a very important tool to solve various engineering problems, such as structural analysis, heat transfer, and fluid dynamics. The key point to the technique is discretization of the domain of interest into many finite elements. A good result is strongly dependent on the number and arrangement of meshes. However, it is very difficult to generate efficient finite element meshes, although there are many finite element analysis techniques available.

The adaptive mesh generation algorithm has been implemented in the expert system in order to save both time and money in the finite element analysis process. It is not required for a user to know detail information about the finite element analysis processes or computer science to test structural analysis. To verify efficiency of *EFEM*, analyses for planar and shell domain models have been performed in two and three dimensions respectively.

1 Introduction

Since Courant (1943) [Cou43] used an assemblage of triangular elements to study torsion problems, the finite element method has been recognized as an important technique to get solutions of difficult engineering problems and developed continuously for structural analysis, heat transfer, weather forecast, fluid flow simulation, mass transport, and electromagnetic potential. Several investigators applied the primitive idea of finite element analysis into building practical fields, such as aircraft and ships [Gal75,Red84]. Since then, many researchers have improved the idea of the finite element method and have presented new algorithms about finite element mesh generation or finite element analysis techniques. The common goals of mesh generation algorithms are to generate optimized meshes and to minimize the difficulty of usage and computer time; meanwhile, the result should be accurate to be accepted.

Although many researchers have developed various finite element mesh generation algorithms, technicians who actually analyze objects find the algorithms difficult to apply. In reality, to analyze an object using the finite element analysis technique requires fluent knowledge of stress analysis and computer applications. Furthermore, it is very difficult to generate optimized nodal points that will guarantee the correct result of the analysis and run efficiently. However, the analysis process is very expensive when hiring knowledgeable professionals to overcome the difficulties. Therefore, required knowledge and sequence of analysis have been translated into a knowledge and rule based system respectively in order to achieve efficient analysis effect with lower cost.

2 Survey to the Algorithms

The finite element analysis technique is a numerical method for solving various problems of engineering and mathematical physics. There has been a lot of research about finite element analysis during the last three decades. The finite element analysis technique has been recognized as a practical method for solving engineering problems with the development of high speed computers.

The method has several advantages in doing structural and non-structural analysis, such as handling irregularly shaped problem domains, handling general load conditions, controlling element size easily depending on needs, and handling many boundary conditions as required.

Analytical solutions are given by a mathematical expression that calculates the values of the problem domain continuously, while numerical solutions are approximate values of the unknowns at discrete points [Log86]. In the numerical method the process of modeling an object by dividing it into an approximate system of smaller unit elements, called finite elements, interconnected at points common to two or more elements is called *discretization*. The common point is called a *nodal point* or a *node*.

Analytical solutions are more accurate than numerical solutions. However, an analytical method requires solving ordinary or partial differential equations, which are not usually obtainable for complex geometry, irregular loadings, and composite material properties. Therefore, a numerical method, such as the finite element analysis, is a very important method to get reasonable solutions. Although its solutions are not exact ones, they are usually acceptable to interpret the problem.

2.1 Existing Algorithms of Mesh Generation

Past research may be subdivided into the categories of general mesh generation algorithms [Byk76,Cav74,FWE70], mesh generation for complex geometric shapes [WT84,JS86], optimized mesh generation [SGA80,MM77,YFRC87a,CH81], and three-dimensional mesh generation [CFF85,Pis81, YS84,YS85]. Researchers have also built up the principles of design for finite element meshes [Zie77, Red84,Kik86], which are very important to the development of finite element analysis techniques. In order to help readers understand the analysis technique, some typical algorithms about finite element analysis are surveyed.

2.1.1 Point Based Mesh Generation TechniquesThis is a common method to generate meshes on the two-dimensional plane for the problems of simple geometry. In this technique, two distinct steps, point generation and mesh construction, will build a mesh in a sequential manner. In terms of the user friendliness of algorithms, the finite element mesh generation methods can be divided into two systems, a fully automatic and a manual system. A fully automatic algorithm requires analysis-dependent input only, such as geometric definition of an object or material properties, and generates meshes of various density determining regions of high and low element densities automatically. A manual algorithm checks for a few conditions only, such as consistency of data points and trivial errors, and a user has to input all data points and all necessary conditions manually. Thus, there is a trade off between the simplicity of system operations and the cost of its implementation.

Considering this, Cavendish [Cav74] tried to develop the semiautomatic triangular mesh generator. In his method, coordinates of boundary nodes, fixed interior nodes, and nodal density information were all the input required. Based on these inputs, algorithms automatically generated additional interior nodal points, which were consistent with the above input data. Triangular elements were generated by interconnecting all nodes such that no elements overlapped and the whole

region was covered with finite elements. Compared to similar methods, such as those of Frederick et al., and Fukuda and Suhara. Cavendish's method had some superior features. For instance, Frederick et al. [FWE70] had plotted all points manually and read them with an electromagnetic graph tracing table. Fukuda and Suhara's [FS72] algorithm did not require the manual plot process; however, the shape of triangles generated was too irregular to get a good analysis result. In Canvendish's method, points were defined by a random number generator to obtain uniformly distributed nodal points statistically. Based on the uniformly distributed points, it was easy to produce better quality triangulations containing elements of more regular shapes compared to others. He also used a smoothing technique to make triangles close to equilateral.

In [LPS84], Lee et al. also used a point-based approach for two-dimensional objects. They first defined uniformly distributed points inside boundaries of the constructive solid geometry (CSG) object. Each primitive shape in a CSG tree has well-distributed points according to a given point density. Then each primitive shape was combined and formed into a larger, single geometric object of well-distributed points. Thus, it was guaranteed that points were always well-distributed. The researchers connected points by trying to make as many quadrilateral elements as possible following the lengthy decision tree to form regular element shapes. They used well-distributed points primarily to connect meshes; however, they had to add or ignore those points when the decision tree could not use existing points or could not find appropriate points. In case they could not build good quadrilateral elements, they built triangular meshes. Generally speaking, their approximation was close to the actual boundary shapes. However, their algorithm could not generate meshes of variable density, which could save analysis cost. Furthermore, the algorithm to form an object of well-distributed points and decision tree for creating an element was too complex and took long computational time.

2.1.2 Mesh Generation Techniques by TreesAnother common approach to generate meshes is to use trees. In [YS83], Yerry and Shephard introduced a quad-tree encoding technique to generate finite element meshes. It was applicable to any kind of two-dimensional domain, and basic mesh information was stored in an integer tree. In this method the object was placed inside a square and then the square was subdivided into four quadrants. Each quadrant was tested to see if it was included inside the object, outside the object, or on the boundary of the object. Based on the position information, every homogeneous quadrant was marked as an integer, which was a two-bit word representing the state of the quadrant. Each subquadrant was tested continuously in the same manner until the object reached the desired level of resolution. However, the quad-tree encoding technique required a much larger number of elements to get satisfactory geometric representation. To shorten the integer tree length, the modified quad-tree technique was introduced. The researchers cut corners off some quadrants, called *cut quads*, which could be fit into small quadrants producing a smooth curve. This method could be easily expanded to accommodate three-dimensional objects. In [YS85], they demonstrated the possible implementation for the three-dimensional objects.

Recently Baehmann et al. [BWS*87] improved the above modified quad-tree technique to ensure the robustness and efficiency of mesh generation. The first deficiency of the original method

was the limited geometric representation of the boundary quadrants. The second was that only one discretized edge segment and one vertex per quadrant were allowed. In order to solve these problems, they changed the whole data structure so that they used explicit storage of vertex data and actual edge intersection. Thus, it would fit the adaptive analysis procedure.

Jackins and Tanimoto [JT80] also used the tree technique for three-dimensional objects. They expanded the quad-tree technique into the oct-tree, which could be used in three-dimensional geometric modeling and space planning. In their method, space array operation was economical in terms of memory space and was performed so that time was increased linearly according to the number of nodes in the oct-trees.

2.1.3 Geometric Modeler-Aided Mesh GenerationThe input phase for a finite element analysis consists of four steps: defining the problem topology, discretizing the problem topology into finite element meshes, specifying the analysis parameter, and reformatting the input data for the selected analysis package. The first step, defining the problem topology, is important because the result of the analysis is dependent on the correctness of the geometric representation. Thus, the complete and unique representation of a domain object by the solid modeler affects the result sequentially in the later steps described above. Recently several investigators have started to build the finite element analysis system onto the interactive geometric modeler [YFRC87b,BWS*87,YS84]. The geometric modeler can help the analyst to define complete geometric representation, which is useful to reduce the complexity of mesh algorithms.

Cavendish et al. [CFF85] also developed an algorithm of discretizing the solid geometry into finite element meshes inside a geometric modeler. With the aid of an interactive solid modeling system, the researchers could get the set of cross-sections of an object. They applied the similar technique that was found in the two-dimensional case [Cav74] in order to insert appropriate nodal points onto each two-dimensional problem domain first. After they defined all nodal points for every cross-section, these points in the two-dimensional cross-sections were interconnected to form tetrahedral elements.

Wu and Abel [WA79] also used an interactive geometric modeler to analyze the objects of a shell structure. In their method they lofted surfaces and found sectional curves by a uniform B-spline. The surface was interpolated between sections by a cardinal spline. The user was allowed to specify a surface-mesh size and this mesh was used for finite element mesh analysis. Throughout their implementation, interactive resources were essential to the system.

2.1.4 Near-Optimized Mesh GenerationIn this method, the measurement of strain energy distribution (SED) is a central concept. The potential energy is calculated by integration of the energy density over the finite element volume. If the degree-of-freedom (DOF) of an object is changed, the value of the difference between the energy integrals for different DOFs provides the measure of sensitivity of the solution to the higher energy participation as the energy varies over the element [MM77]. The near-optimized finite element mesh is generated based on the variation of SED. Recently, self-

adaptive near-optimized mesh generation algorithms have been introduced [SGA80,YFRC87a].

In [SGA80], Shephard et al. tried to get optimal finite element meshes using the variation of SED. Their method started with a coarse initial idealization by an automatic procedure, and a self-adaptive processor iteratively analyzed and improved selected idealization until the discretization errors throughout the domain were acceptably small. It was usually difficult to describe the minimum amount of information required to specify the problem and to optimize the mesh for analysis. Thus, the researchers developed a grid synthesis method that combined the conventional isoparametric approach with the variation of SED.

First of all, initial coarse meshes were generated automatically and were evaluated. In the regular isoparametric method, the key nodal points are those along the boundary whose positions determine the parameters on which isoparametric lines are drawn from this boundary to the opposite side. In addition to those conventional parameters, the SED value was calculated at key nodal points in their method. The result of the initial analysis could be used to generate a near-optimal mesh by rearranging key nodal points along the boundary. Variation of SED was an important criterion to place key nodal points.

In order to find the proper position of a key nodal point, they defined the total absolute energy difference, ϵ, which was evaluated by difference of SEDs at adjacent nodes as

$$\epsilon = \sum_{i=1}^{P} ABS(SED_i - SED_{i-1})$$

in which P was the number of data points along a curve where SED had been calculated. Strain energy difference between adjacent key nodal points, δ, can be defined as

$$\delta = \frac{\epsilon}{(n-1)}$$

where n is the number of key nodal points to be placed on the boundary. The first key nodal point was placed at the start of the curve. Then the algorithm started to calculate ϵ again until the value of ϵ was greater than or equal to δ. The next key nodal point was placed there and δ was subtracted from the current summation of ϵ. This process continued until the penultimate key nodal point had been placed. The last one was located at the end of curve. After every key nodal point was rearranged in boundary space, a grid could be synthesized automatically or interactively by users. Since the main idea to generate the near-optimal mesh was to rearrange points such that the SED variation between two nodes was equal as much as possible, the approach optimized the mesh density. However, key nodal points were allowed to be placed only along the boundary curves. Consequently, the method could not handle the case when the peak of SED occurred in the interior of the object.

Yen et al. [YFRC87b,Yen85] also developed the near-optimal mesh generation algorithm that was based on the SED, with the aid of the geometric modeler. The data structure used in the model representation was hierarchical and extensible. All of the problem parameters were associated

with the geometry and nongeometric attributes such as material property or load information. In their modeling technique, memory space for the attribute was not allocated unless or until it was needed. The user designed the object with the set of subregions and specified the number of coarse elements for each region. In the first analysis of coarse mesh, the user was able to make sure that all parameters were set correctly. After the coarse meshes were generated successfully, they subdivided the geometric model based on the variation of SED that was applied in Shephard et al. [SGA80]. B-spline manipulation allowed for the approximation and refinement of surfaces easily [CLR80]. Because they used a sophisticated B-spline based geometric modeler, subdivision of the surface by the variation of SED could be performed well. The researchers refined meshes in the region where SED changed dramatically and they combined the too-fine meshes where the mesh size was too fine to begin with. Thus, near-optimal meshes were generated after the second cycle of their analysis.

3 Drawbacks of Existing Algorithms

Thanks to active research, the cost of finite element analysis has been remarkably reduced and the application area becomes wider. However, most existing algorithms had difficulty such that they required the highly-trained technicians to analyze the object properly. Some drawbacks can be pointed out as follows:

1. **Most algorithms were just preprocessors:** The finite element analysis technique is too complex to be applied by a person who has not been trained. If the algorithms just generate nodal points only to form meshes as a preprocessor, it is still difficult to get a good analysis result unless actual analysis process and postprocess are performed appropriately. Furthermore, it is very difficult to generate the optimal meshes in this approach because the mesh generation algorithm does not consider the computational errors occur.

2. **Most algorithms generated triangular elements:** Choice of an appropriate element type is dependent on several factors, such as problem domain and analysis environment. Common basic element types for two-dimensional objects are triangles, rectangles, and quadrilaterals. Assuming each element has nodal points only on its vertices (no extra DOF), it is known that the rectangular or the quadrilateral element is superior to the triangular element type. Since the stress is linearly proportional to the energy for the most of materials, a constant-strain triangular element is not appropriate to approximate the linear increase of the energy with discrete constant values.

3. **Generated meshes were not optimized:** The construction of optimized mesh depends on two factors, the number of finite elements and the labeling sequence of nodal points. Generally speaking, more elements guarantee better accuracy of the analysis result; however, having elements which are too dense is expensive. Since there is the trade-off between accuracy of result and analysis cost, it is required to set an appropriate number of elements. In order to control the number of elements, meshes of different density should be generated depending on necessity.

The bandwidth of the stiffness matrix, which also affects the analysis cost severely, depends on the the order of placed nodal points. Therefore, the sequence of labeling nodal points is also important to generate efficient meshes. Most methods presented could not generate efficient labeling schemes.

Several systems developed recently tried to overcome these deficiencies. However, they still have difficulties that will be described in Chapter 4. In the design of *EFEM*, one of the primary goals is to delete all exposed problems. However, self-adaptive near-optimal mesh generation mechanisms found in [YFRC87a,SGA80] are applied primarily in the *EFEM* because the technique could be translated into rules easily and generated reasonably optimal meshes.

4 Analysis Using Conventional System

As mentioned earlier there are many analysis packages developed for the finite element methods because the importance of the technique has been recognized widely. They can be grouped as two groups based on their purposes. One is called *general-purpose* packages, which includes many types of problems, such as two-dimensional, three-dimensional, truss, beam, shell, and fluid. The other is called *special-purpose* packages, which is developed for small but specific problems only. In this section the usage of other systems, such as *ADINA* [ADI81] and *I-DEAS* [WPWW87,Rud88] are described to understand the characteristics of different interfaces. *ADINA* package was developed early 1980 and *I-DEAS* is relatively new package.

4.1 Object Modeling

Each package has own modeling tool to design the test objects, such as *ADINAIN* [ADI83] for *ADINA*, and *Geomod* [SDR88] for *I-DEAS*. In order to visualize this discussion, an I-shaped thin aluminum plate is selected as the test model. Its specifications are: width 5, length 2, thickness 0.1, and it has an inverted elliptical cut of 0.2 at both sides in the middle. The problem specification of this example is shown in Figure 1.

The bottom part is held firmly so that there is no X and Y direction movement, while the plate is subject to opposing tensile force of 100 boundary load at the top. Assuming the plate is made of aluminum, its elastic modulus and Poisson's ratio are 1.0e+07 and 0.33 set, respectively. Since the geometry of the plate is elongated vertically, it is better for a user to design the whole model into two regions because the analysis result would be more accurate in that way.

Since the *ADINA* does not provide a geometric modeler besides *ADINAIN*, a user has to specify the geometric information of the test object using *ADINAIN*. However, the process is very tedious and time consuming. The portion of an input file for the I-shaped sample plate described in the above looks like:

```
DATABASE CREATE
```

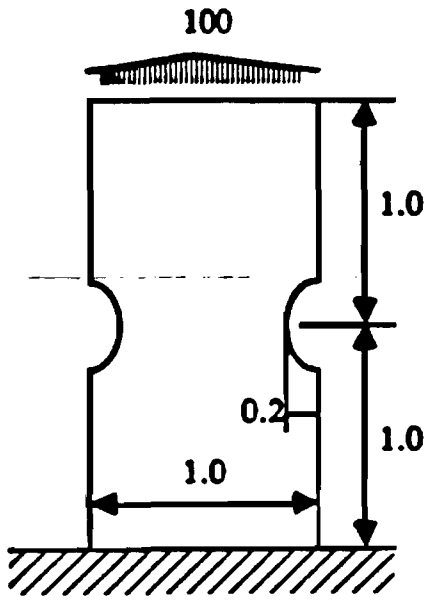

Figure 1: The Problem Specification of the I-shaped Plate

```
HEADING ' I-SHAPED THIN PLATE'
MASTER IDOF=100111
COORDINATES / ENTRIES NODE YZ
1 0.0 2.0 / 2 1.0 2.0
3 0.0 1.8 / 4 1.0 1.8
5 0.0 1.0 / 6 0.2 1.0 / 7 0.8 1.0 / 8 1.0 1.0
9 0.0 0.8 / 10 1.0 0.8
LINE STRAIGHT N1=1 N2=2 EL=4 NFIRST=13 RATIO=1
LINE STRAIGHT N1=6 N2=4 EL=4 NFIRST=16 RATIO=1
LINE STRAIGHT N1=1 N2=3 EL=2
LINE ARC 3 6 5 EL=2
LINE COMBINED 1 6 3
LINE STRAIGHT N1=2 N2=4 EL=2
LINE ARC 4 7 8 EL=2
LINE COMBINED 2 7 4
```

Since a user designs the test model in his/her mind without visual aids, the process is very tedious and easy to make errors. Furthermore, a user has to find the appropriate nodal and element sequence during the designing process. Later packages tried to reduce the tedious steps during the preprocess of the finite element analysis occurred in the above case. They are very easy to use and

powerful in their functions, although they have some difficulties mentioned before, such as nested menu driven interaction and node-orientation node generation. *SUPERTAB* is one of those recently developed packages. It is designed to interact with a geometric modeler, a drafting package, and a mechanical data analysis package.

The geometry information can be specified in various way using *I-DEAS* package. For an example, the test object may be specified by defining the nodal points inside *SUPERTAB*. In *SUPERTAB* every node has five attributes of a definition coordinate system, a displacement coordinate system, local coordinates, global coordinates, and a nodal color. To create a new node means to set all the attributes, although the important attribute is only the coordinates of the point. The system provides a default menu that sets up these attributes for every nodes subsequently created. A new nodal point can be created by various menu options, such as keying in coordinates, digitizing locations, interpolating two nodes, copying existing nodes to new locations, picking positions, and reflecting existing nodes. In the case of the I-shaped plate, the positions may be keyed in or picked by a mouse. After creating all nodes, the orientation of nodes can be changed, if necessary. Nodal bandwidth is also possibly reduced by selecting bandwidth management menu here.

4.2 Analysis Process

As it was shown in the above, after a user of *ADINAIN* specified the correct geometry information, boundary constraints, loading, and surface group information should be also keyed in correctly based on the sketch. The example codes looks like:

```
BOUNDARY 11111 / 5 8
MATERIAL 1 ELASTIC E=10.E6 NU=.33
EGROUP 1 PLANE STRESS2 MATERIAL=1
GSURFACE 1 2 6 7 EL1=4 EL2=4
.......
LOADS ELEMENT
1 0.0 0.0 100.0
2 0.0 0.0 100.0
.......
```

The bandwidth of stiffness matrix is completely dependent on a user's nodal labeling sequences. Another problem is that the user's task becomes more tedious and it is possible to make more errors, if the meshes get bigger due the increased complexity of the test model. Furthermore, since the analysis is performed in batch mode, the trial and error method is only a way to fix the errors.

The analysis process of *I-DEAS* is much easier and powerful compared to the one of *ADINA*. After satisfactory nodes are generated, elements should be created filling elemental attributes of a family name, an order, a topology, physical properties, material properties, and a color for every element. The defaults menu of element creation also sets up these attributes for all elements subsequently created. A new element can be created by the several menu options, such as picking nodes,

copying elements, reflecting elements about a plane, sweeping elements along a vector, revolving elements about a vector, extruding elements, and extracting elements. Assuming that the new elements are created by picking nodes, the following sequences should be performed correctly:

1. Enter the label of the first element to be created and increment used to calculate the label for each subsequent element
2. Set the type of elements
3. Specify material and physical properties (can be skipped by setting defaults)
4. Select the nodes for each element.

Assuming that 4×4 discretization for each region is reasonable, 16 elements should be specified in the above sequences. The elements for the bottom region can be reflected using symmetric property, such as:

1. Pick the 16 elements created in the above
2. Define the plane including the base line of the top region
3. Enter a node reflection tolerance as 0.1 to avoid copying the base elements
4. Enter the label of the first element reflected and an element label increment, used to compute the label of each subsequent element reflected
5. Enter the label of the first node reflected and a node label increment, used to compute the label of each subsequent node reflected
6. Enter a label increment for material property tables
7. Enter a label increment for physical property tables
8. Specify symmetric or nonsymmetric warping
9. Examine the new elements
10. Accept or reject the new elements.

One of disadvantages in this kind of system is that most menus require lengthy sequences, as shown in the process of reflecting elements. Loading vectors can be specified by picking menus of nodes, edges, surfaces, and a model. The value of vector is specified by selecting the submenu of constant or data points. The constant menu sets the same value for each node, edge, or surface. The data menu sets the value which is the function of selected data points. Boundary constraints also can be set in similar ways by selecting menus and keying data.

After the analysis is successful with the initial settings, the adaptive meshes can be generated based on the elemental distortion or the strain energy. For example, the adaptive meshes can be created by selecting the strain energy method in the following sequences:

1. Select the analysis data set that contains the element strain energy
2. Select the method used to improve results (i.e., move nodes), if necessary
3. Select element splitting method
4. Enter the option for controlling the smoothing process
5. Enter the percentage of element that can be split to reduce distortion
6. Execute mesh smoothing process
7. Display the new meshes
8. Accept or reject the new meshes.

At this point a user of the *SUPERTAB* may generate other optimal meshes repeatedly by selecting the menu of adaptive meshes.

5 Analysis Using *EFEM*

5.1 Designing an Object Model

A user should specify an object using *Shape Editor* operators in order to analyze the object model. All basic geometric entities provided in *Shape Editor* can be used to represent two-dimensional or shell structure models. *EFEM* reads the object model in three different methods using predefined functions of *Shape Editor* for overcoming the difficulty of the sophisticated geometric modeler. The possible design methods are by points, by curves, and by surfaces with the *Alpha_1* modeler. A user can design a model of problem domain with one of these methods.

Design by points . This is the simplest method of representing object models. A designer simply specifies every vertex point of an object in sequence. A system will build edges connecting the sequential vertices. The minimum number of vertices to form a plane object is three. Insufficient vertices would be detected by an error handler along with other possible errors of multiply defined points and intersecting edges.

Design by curves . Some models may have arcs or irregular curves. In that case, defining points following the irregular curves will be very tedious. A designer will use the geometric entities such as a line, an arc, a curve, or a profile to represent a model. A user may use points to

define line type entities mentioned above. However, points cannot represent the object model directly. An error handler will also check if the object is a closed model in terms of boundary. In order to detect correct intersections, every constructor for each entity such as points in the line operation is required to be input.

Design by surfaces . In this method, a user specifies one or more surfaces to represent an object. A desired surface may be defined by low-level or high-level geometry constructors. Most shell domain objects can be designed by surfaces, while the two-dimensional planar domain can be designed by points or curves. This is the most elegant but difficult method to specify an object.

If an object model is designed correctly, all geometry information is specified. In the case of most preprocessors, a user must specify those nodal points. However, *EFEM* will find appropriate nodal points for the mesh and need only analysis-dependent information such as a Young's modulus, Poisson's ratio, loading vectors, and displacement constraints.

The same I-shaped thin aluminum plate can be defined with *Shape_edit* easily. Because the two-dimensional plate is elongated vertically, a user may decide to make two region over the plate. After vertex points were defined, four boundaries, L1, L2, L3, and L4 specified the top portion of the plate and other three boundaries, L5, L6, and L7 would specify the bottom portion with L2.

5.2 Analysis Process

The designed model will be loaded with the defined geometry entities. The script file of the execution of this analysis is shown in Appendix A. The plate was analyzed under verbose mode so that every step taken displayed. After *EFEM* examined the domain geometry, it drew each boundary curve with its associated boundary type. After *EFEM* asks for the mesh density, the material name, and the thickness of the object, then the loading vector will be specified as the list of a region number, X direction component, Y direction component, and Z direction component. In this example, it is specified as (1 0 100) because the tensile force was 100 upward (Y direction) without any X or Z component. At this stage, *EFEM* also asks boundary constraints as many as the number of regions. Since the boundary configuration displays two regions marked clearly with their associated types, a user should be able to set boundary related information such as boundary constraints and loading vectors easily.

Uniform meshes (Figure 2) are automatically generated and the *ADINA* input data file is created at this level. As rules are triggered continuously, *EFEM* invokes *ADINA* to analyze the uniform meshes and evaluate the meshes. The output data file of *ADINA* is parsed and the deformation and the strain energy information is attached to each regional geometry respectively. Deformed meshes are obtained from the uniform finite element mesh offset by the corresponding nodal point displacement exaggerated by a magnifying factor. The deformed meshes are shown in Figure 2. With the graphical tools provided in *Alpha_1*, the original plate and the deformed plate can be also rendered in a transparent color to compare the displacement.

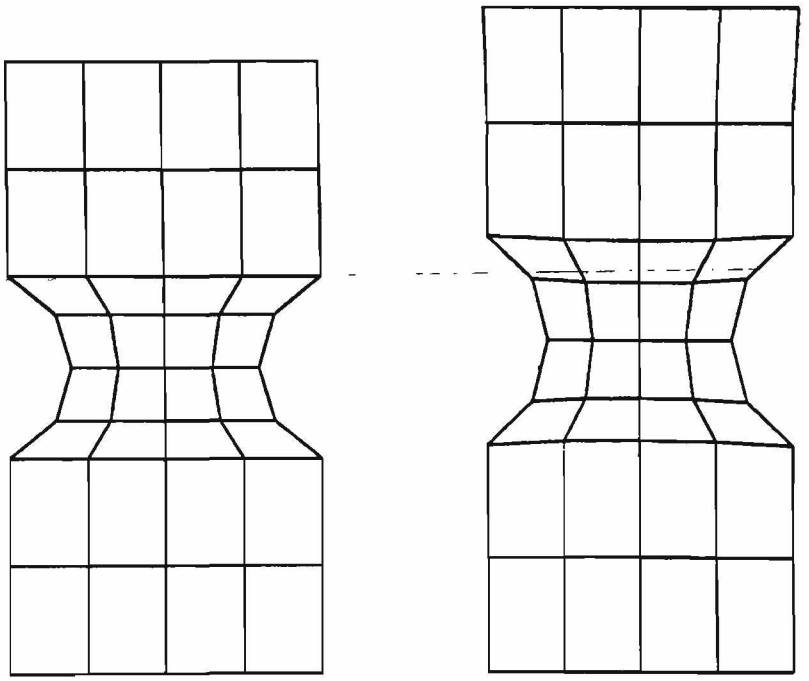

Figure 2: Uniform Meshes and Deformed Meshes of the I-shaped Plate

The SED is also visualized by extracting the visible boundary surface with strain energy values. In order to see the contour line of strain energy, the strain energy difference has been interpolated into different color variations linearly. The rendered picture is shown in Figure 3.

A criterion surface, four-dimensional surface including a strain energy field is constructed at this state automatically. The critical value of subdivision is set based on the option of mesh density such that 50% is for coarse mesh, 33.3% is for normal mesh, and 25% is for dense mesh. The denser the mesh density selected, the lower the critical value assigned so that the level of subdivision is getting deeper. In this analysis the critical value of subdivision of criterion surface is selected as 50% of maximum strain energy, which is 2.185×10^{-3} in this analysis. The criterion surface is subdivided recursively into four subsurfaces until the SED over the subsurfaces is less than the given critical value. An optimal mesh (Figure 4) is generated from the final state of a subdivided criterion surface. The optimal meshes and uniform meshes have an equal number of elements, but the strain energy of the optimal mesh is increased by 5.1% and the maximum YY stress is increased by 15.6%. After the first analysis has been performed with generation of the optimal mesh, *EFEM* asks the user the selection of the next job from three choices: `Quit`, `Critical value change`, and `Loading vector change`. The script file in Appendix A shows that the selection was made as `Critical`

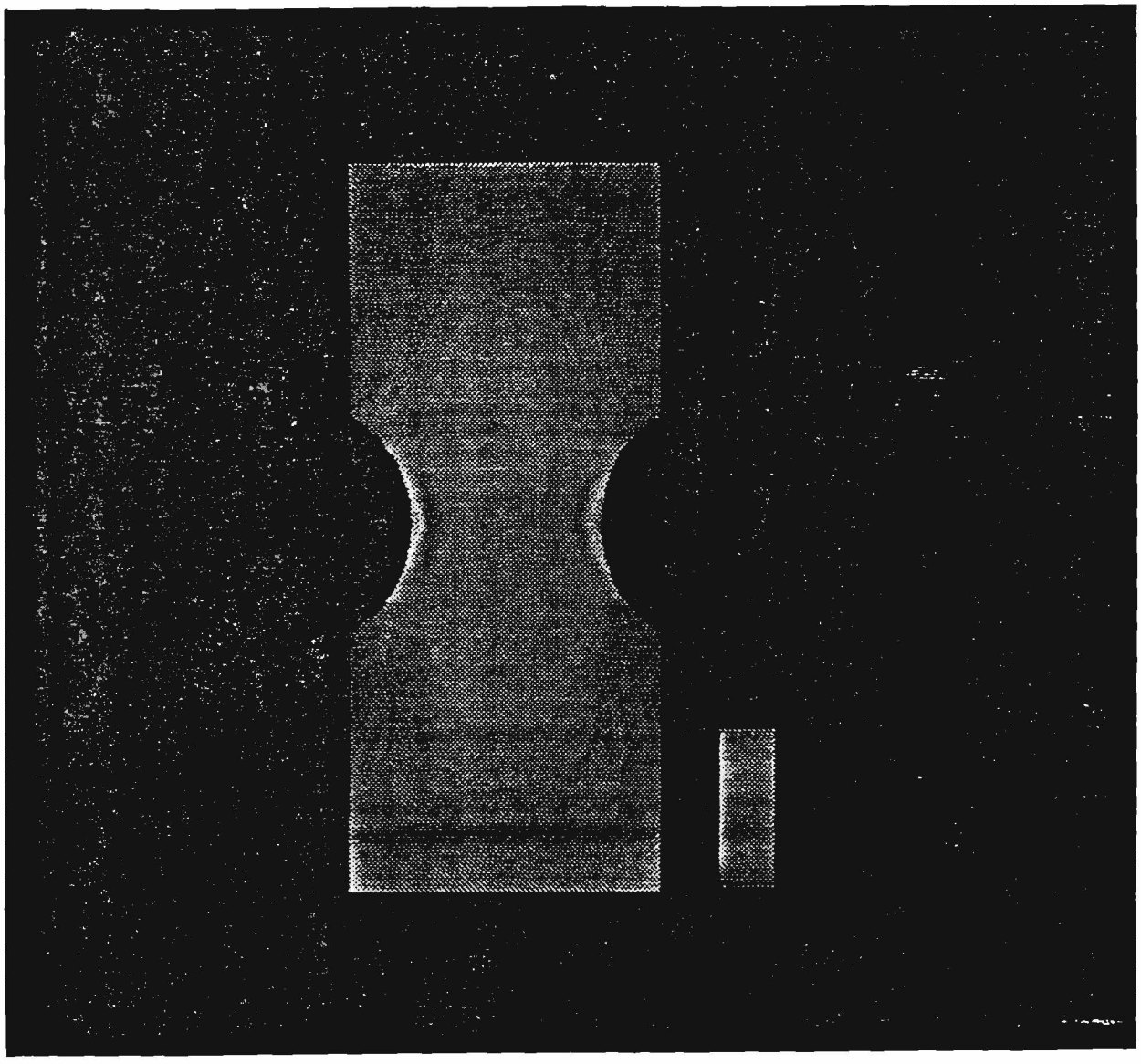

Figure 3: The Strain Energy Distribution (SED) of an I-shaped Plate

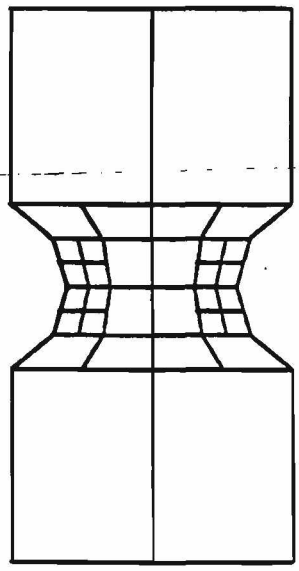

Figure 4: The Optimal Mesh of the I-shape Plate

value change from 50% to 25% and another analysis for the second optimal meshes has been run. The optimal meshes generated with the critical value of 25% of maximum strain energy show the best result in terms of total strain energy increased by 15.0% and maximum YY stress value increased 63.3%, although mesh size is increased by 262.5%. Table 1 summarizes three different analyses,

the uniform mesh, the optimal mesh with the critical value of 50% MSE, and the optimal mesh with the critical value of 25% MSE, in terms of the number of elements, the number of nodes, the degree of freedom, and strain energy values.

6 More Examples

6.1 A Square Plate with a Central Hole

The problem considered in this analysis is that of tensile loading of a square plate with a small central hole as depicted in Figure 5. The width, height, thickness of the suare plate, and the radius of a central hole are 10, 10, 0.1, and 2 respectively. Because tensile forces at the left and right side boundaries are pulled out horizontally assuming plane stress, the analysis can be done by $\frac{1}{4}$ of the

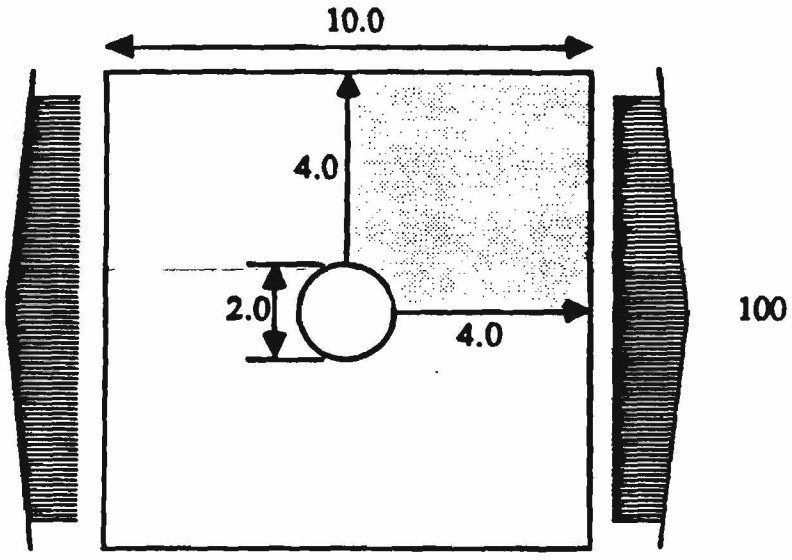

Figure 5: The Problem Specification for the Square Plate

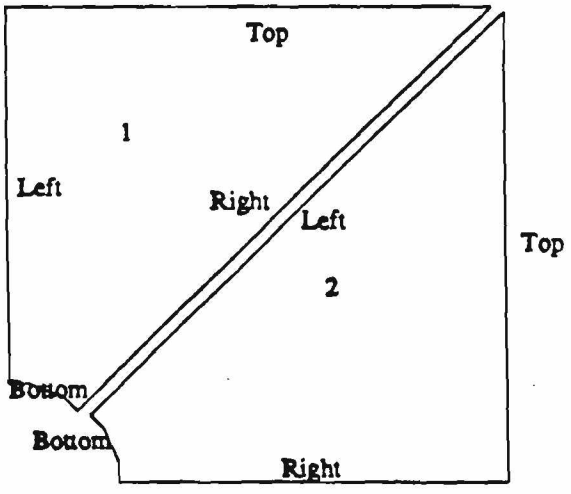

Figure 6: The Boundary Configuration of the Square Plate

whole area using symmetric property. In the real world, many objects have symmetric properties and actual analysis can save computer time by the analysis of minimum area. The result will be appended into the whole field with graphical tools later. In this case, a user selected only the top-left quadrant of the whole plate, which is represented by shading area in Figure 5. Another restriction required as background in this example is to know how to segment the selected problem domain appropriately. Since the problem domain is not quadrilateral, it is necessary to convert the object into the combination of two quadrilateral elements as show in Figure 6.

The analysis sequence is very similar to that for the I-shaped plate. As was discussed earlier, *EFEM* will generate quadrilateral elements only in order to feed the data into *ADINA*. The analysis object should be defined in tensor product format. It is assumed that the analysis domain has been selected as a quadrant of a whole problem domain to save analysis costs by the user's expertise in this example. The model has also been designed such that the analysis domain consists of two quadrilateral elements. Based on the geometry information, *EFEM* decides each boundary type so that the orientation of each boundary is consistent and *EFEM* generates the boundary configuration for reference shown in Figure 6. The boundary type of Region 1 has been decided first and *EFEM* tried to find the boundary type of Region 2. The top of Region 2 is set to the right boundary in order to be consistent in orientation at the right boundary of Region 1. The boundary configuration helps a user to specify the appropriate boundary type when it is necessary to input the data associated with the boundary type such as a loading vector and displacement constraints.

Because vertical edges of the original plate were pulled out horizontally by an 100 tensile force along the X-axis, the loading vector would be given onto the top boundary of Region 2 as a distributed loading. Thus, the loading vector was specified as (2 100 0). To decide the displacement constraints properly will require complicated thinking in this analysis. Considering the target area is only a quadrant of a whole plate, it is obvious there will be no X directional translation following the left boundary of Region 1 because it is on a symmetric axis. Other boundaries of Region 1, top, bottom, and left will have full degree of freedom. In Region 2 the same symmetric property is held but in the X direction. Because it is the top half only, there will be no Y directional translation

Table 1: Mesh Analysis for an I-shaped Plate

Items	Uniform Mesh	Optimal Mesh (MSE x 0.5)	Optimal Mesh (MSE x 0.25)
Number of elements	32	32	116
Number of nodes	45	49	157
Maximum strain energy	2.185e-03	3.004e-03	5.968e-03
Total strain energy	1.034e-03	1.086e-03	1.189e-03
Total degrees of freedom	80	92	304
Maximal YY stress values	90.985	105.05	148.6

Table 2: Mesh Analysis for a Square Plate

Items	Uniform Mesh	Optimal Mesh
Number of elements	72	38
Number of nodes	91	58
Maximum strain energy	4.156e-03	3.834e-03
Total strain energy	12.398e-03	12.608e-03
Total degrees of freedom	168	107
Maximal XX stress values	99.28	68.945

along the right boundary of Region 2. Thus, displacement constraints, `(Left Xt)` and `(Right Yt)`, were given for Region 1 and Region 2, respectively.

In this analysis mesh density was selected as a `normal` option and a global size of the option is set to 6. Thus each region is divided into 6 × 6 elements. After generating the uniform mesh (Figure 7), the deformation mesh is drawn as described in the previous example and shown in Figure 8. Although actual analysis is done over a quadrant, mesh drawings are appended to complete the whole plate using the symmetric property. The SED (Figure 9) is rendered by interpolating energy from zero to maximum strain energy (4.156×10^{-3}). Figure 9 shows that maximum strain energy occurs around the top and bottom of the central hole.

The critical value of subdivided criterion surface is selected as 33.3% of maximum strain energy in this example since the mesh density was selected as a `normal` option. After subdividing the criterion surface until the SED is less than the given critical value over all subsurfaces, the optimal mesh has been constructed from the final criterion surface (Figure 11). The comparison of uniform mesh (6×6) and optimal mesh is shown in Table 2. The number of elements in the optimal meshes is only about 52% of one of the uniform meshes; however, total strain energy is increased by 1.7%.

At this point a cycle of whole analysis is finished and *EFEM* inquires for the selection of the next job. In this example, Loading vector change is selected to demonstrate the capability that lets a user simulate several analyses in one session. If a user requests to start another analysis cycle with a new loading vector, a rule calls a function that triggers the setting of a loading vector. If the selected next job is to rerun the system with a new loading vector, then *EFEM* interacts with a user to ask a new loading vector. The interaction will trigger the next rule and *EFEM* will repeat the whole process from reading the new loading vector.

In the second analysis, a compressional force of 100 is assumed on the center of the top and bottom, respectively, as shown in Figure 11. The type of loading is concentrated loading rather than distributed loading on the boundary. After *EFEM* interacts with a user to get the required information, a new deformation mesh (Figure 12) is drawn. A new optimal mesh (Figure 13) is also generated by the new strain energy distribution (Figure 14) according to the concentrated loads.

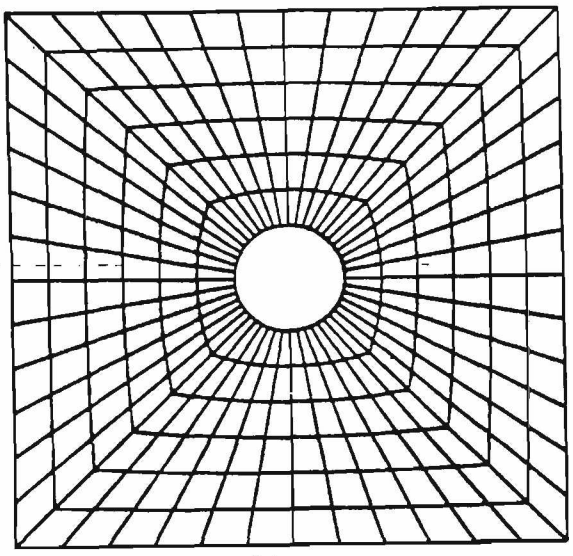

Figure 7: Uniform Meshes of a Square Plate by Distributed Loads

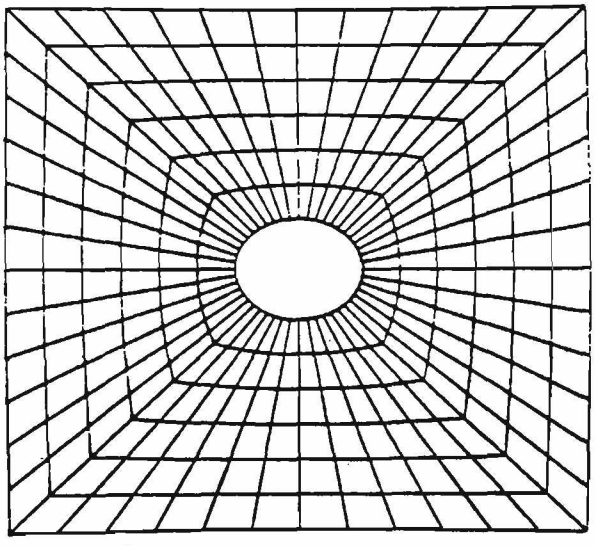

Figure 8: Deformed Meshes of a Square Plate by Distributed Loads

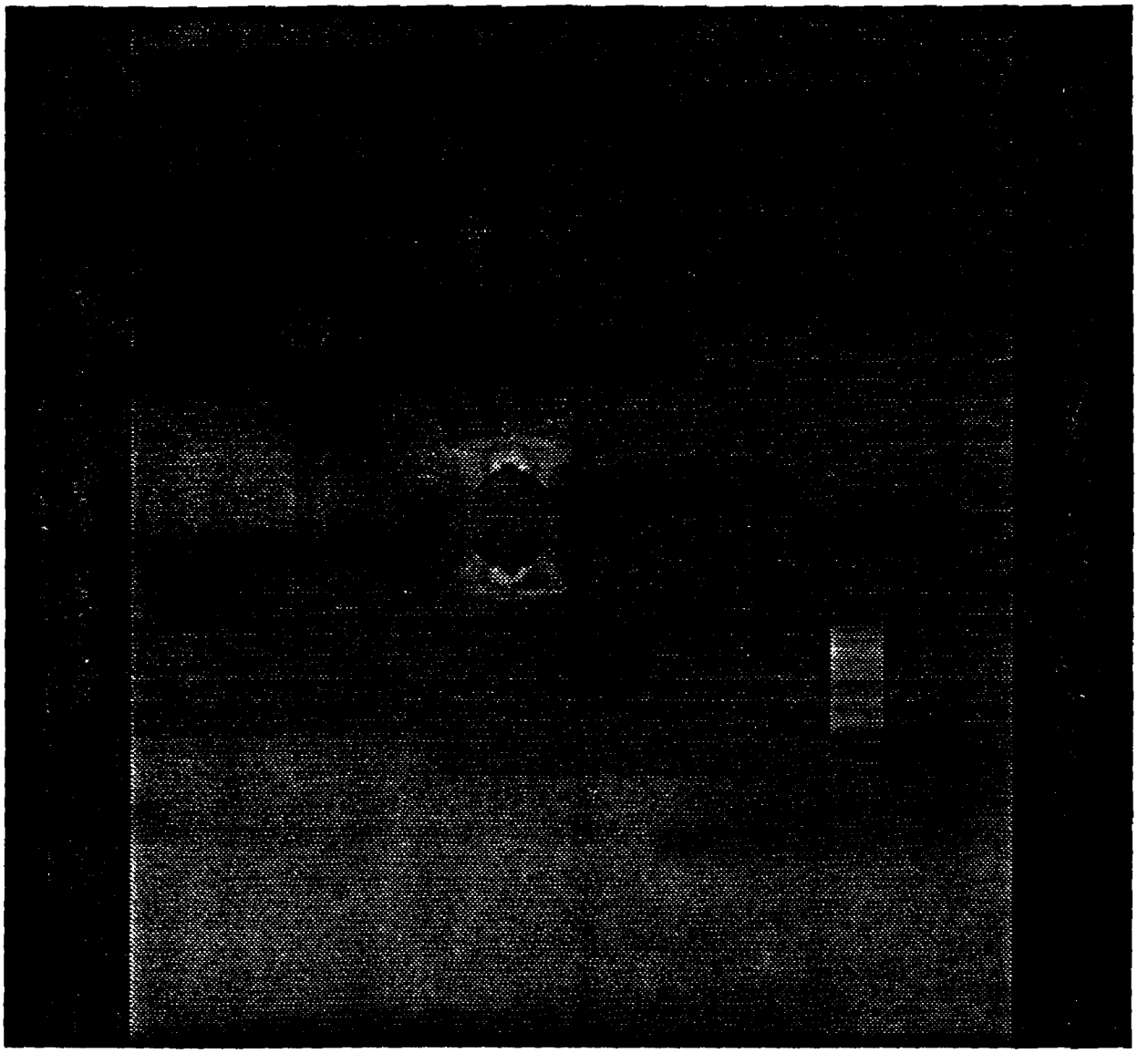

Figure 9: The SED of the Square Plate by Distributed Loads

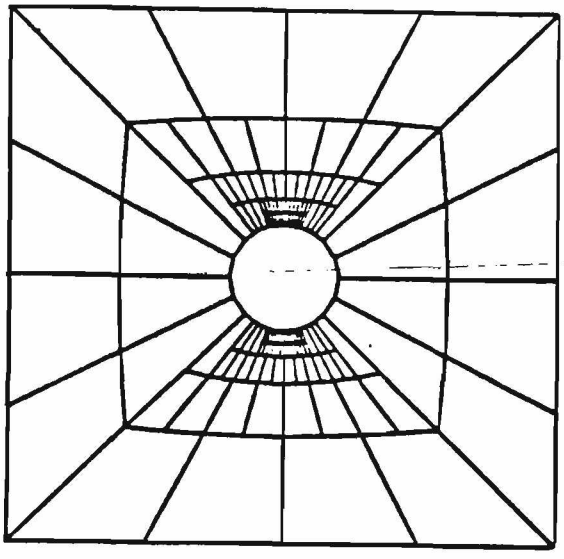

Figure 10: The Optimal Meshes of the Square Plate

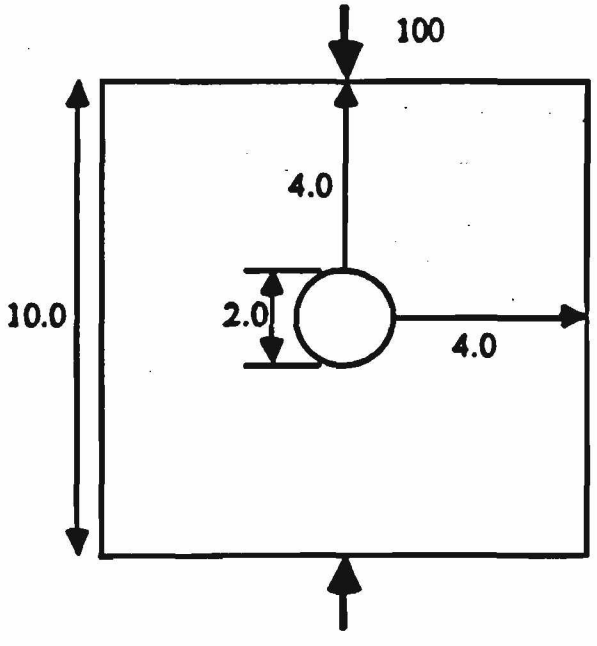

Figure 11: New Concentrated Loads on the Center of Top and Bottom

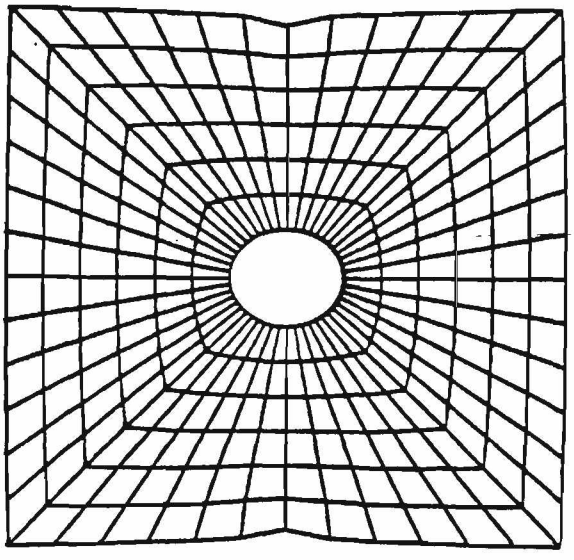

Figure 12: Deformed Meshes of the Square Plate by Concentrated Loads

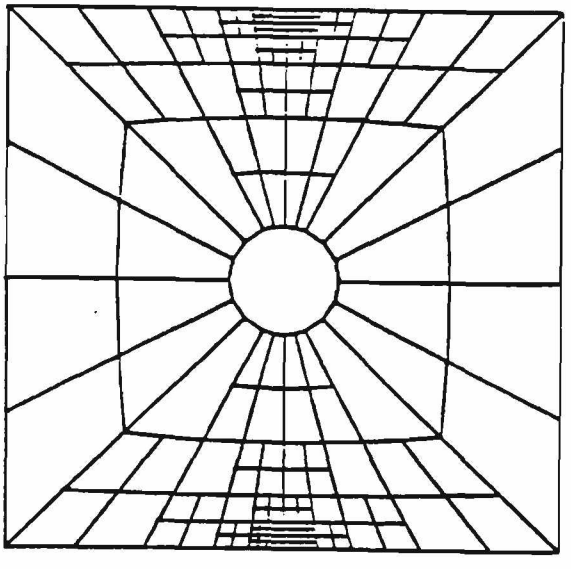

Figure 13: Optimal Meshes of the Square Plate by Concentrated Loads

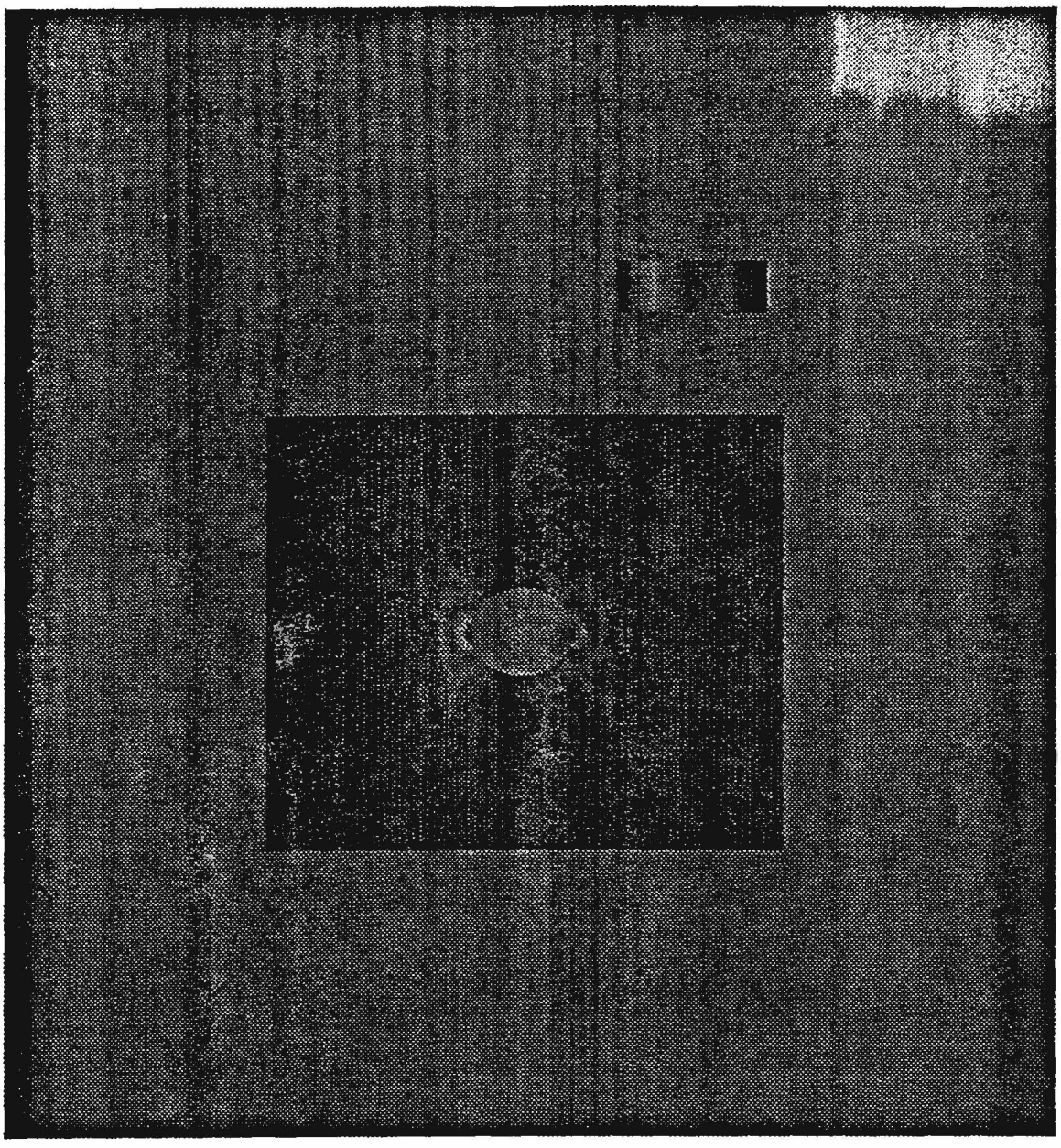

Figure 14: The SED of the Square Plate by Concentrated Loads

A user can continue other analyses with new conditions simultaneously if necessary. This feature of *EFEM* allows analysts to generate optimal meshes easily and finish their tasks in an efficient way.

6.2 Automobile Bumper

Shell structure represents objects that are existing in a three-dimensional space in *EFEM*, although shell analysis can be done in either two-dimensional or three-dimensional space. The thickness of shell structure should be less than 10% of the longest boundary length in order to get a satisfactory analysis result. Since there are many shell structured objects in the real world, shell domain analyses are very useful. Automobile bumper was selected as an example to demonstrate the functionality of the shell domain analyses.

Suppose a thin automobile rear bumper, with length 40, width 6.75, and thickness 0.2, is supported by rigid holders around both ends. Assuming that the bumper is subject to opposed concentrated loading of 100 at the middle, caused by rapid impact with the third object, the load will be given at the intersection of the middle lines, such that the applied compressional force should be modeled by a concentrated load at the point. Its problem specification is shown in Figure 15. Since symmetric property is held along the Y direction, only the right half of Figure 15 will be modeled in this example. In a shell domain analysis, the input object should be the list of one or more surfaces. For instance, after defining all points required, two surfaces are modeled for the top and the bottom part of a bumper by a user in *Alpha_1* as follows:

```
TopCrv := curve( parmInfo( QUARTIC, EC_OPEN, KV_UNIFORM ),
              list( PtA, PtB, PtC, PtD, PtE,  PtF, PtG ) )$

BottomCrv := curve( parmInfo( QUARTIC, EC_OPEN, KV_UNIFORM ),
              list( PtG, PtH, PtI, PtJ, PtK, PtL, PtM ) )$

TopBumper :=  sweepConstantWidth( CenterLn, TopCrv, 1.0, NIL )$
BottomBumper :=
           sweepConstantWidth( CenterLn, BottomCrv, 1.0, NIL )$
```

Two surfaces can be fed into *EFEM* through a main function such as

```
(efem 'Shell (list psl:TopBumper psl:BottomBumper)).
```

Primarily, *EFEM* draws each boundary (Figure 16) with its associated boundary type. From the same figure, it is obvious that the loading vector should be set to a negative Y direction because the bumper was modeled such that it was placed in the X-Z plane. The loading vector is attached to the bottom left corner of Region 1. There is no displacement constraint in the middle of the bumper, but both ends have displacement constraints of all six degrees of freedom.

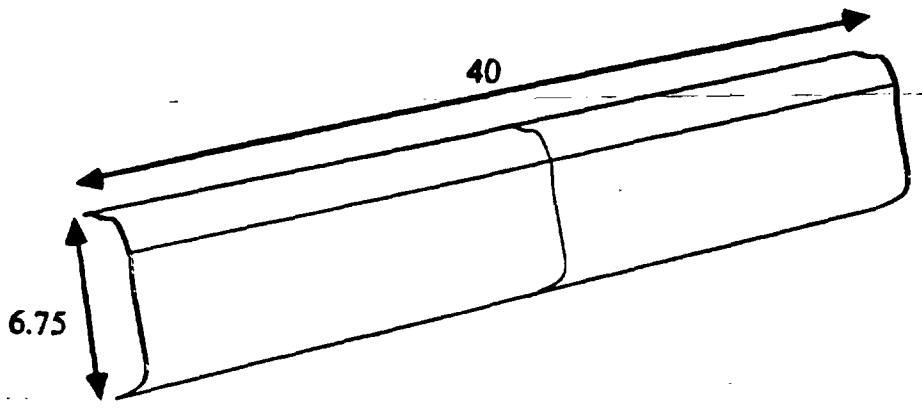

Figure 15: The Problem Specification of the Bumper

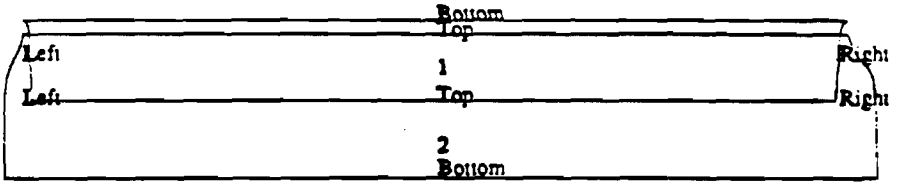

Figure 16: The Boundary Configuration of the Bumper

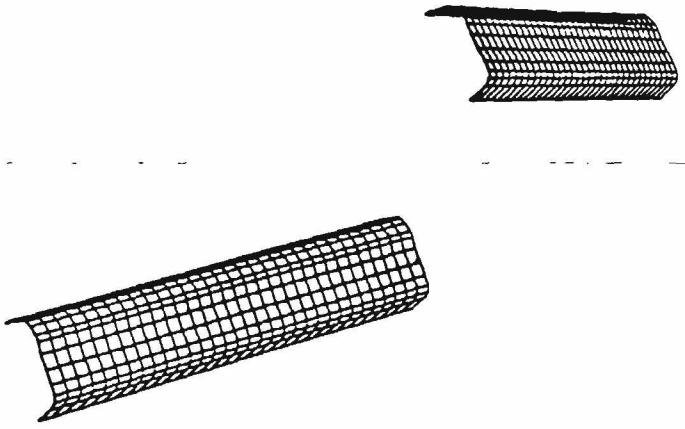

Figure 17: Uniform Meshes of the Bumper

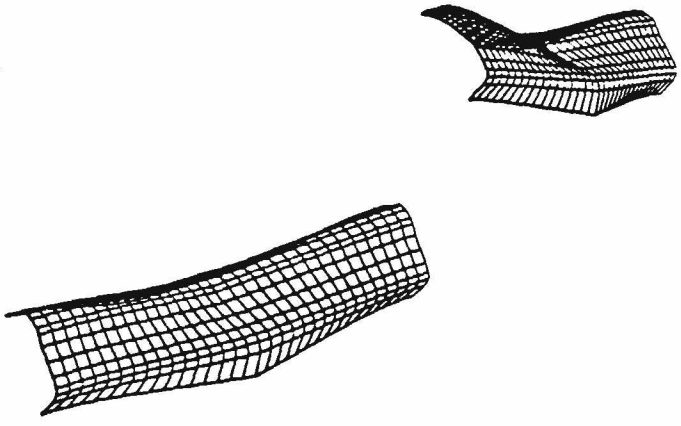

Figure 18: Deformed Meshes of the Bumper

Table 3: Mesh Analysis for an Automobile Bumper

Items	Uniform Mesh	Optimal Mesh
Number of elements	256	68
Number of nodes	289	102
Maximum strain energy	0.14687	2.726
Total strain energy	2.922	17.280
Total degrees of freedom	527	166
Maximal XX stress values	2281	384.5
Maximal YY stress values	663.4	366.65
Maximal XX stress values	391.7	902.35

In this example, mesh density was set as an **arbitrary** option and the number of element in a row and column was specified as 8×16. A uniform mesh (Figure 17) is automatically generated by subdividing each region uniformly into smaller shell elements. The deformed mesh after the concentrated loading is applied is shown in Figure 18.

After *EFEM* parsed the *ADINA* output file, the calculated strain energy was attached into the surface and the criterion surface is constructed such that the fourth coordinate of the surface function interpolates the nodal strain energy. The strain energy distribution was visualized by interpolating SED linearly into color variations in Figure 19.

Because mesh density was selected as an **arbitrary** option, *EFEM* inquires how much critical value would be used for subdividing the original surface. In this analysis 50% of the maximum strain energy (1.469×10^{-1}) was set as a critical value. The criterion surface is subdivided recursively into four subsurfaces until the *SED* of every subsurface is less than the given critical value set. The optimal mesh (Figure 20) is generated from the final state of the criterion surface. Table 3 compares two results of the uniform meshes and optimal meshes. The number of optimal meshes is less than 27% of one of uniform meshes; however, the total strain energy is increased by over 170%.

6.3 A Spoon

In this example, a structural analysis is performed on a spoon that is topologically more complex than previous examples to demonstrate that *EFEM* can be applied to many real objects. All previous examples analyzed were designed by the B-splines representation technique. Furthermore, the tensor product B-splines surfaces were considered in order to be compatible with the *ADINA* package. Although the appearance of a spoon is simple, it would not be easy to model parts accurately such as the warped bowl part, the smooth curved handle, and the connection between the two parts. Cobb [Cob84] modeled a spoon when she developed a sculptured surface designing technique with a B-spline representation. In order to start the analysis of the spoon, a simple

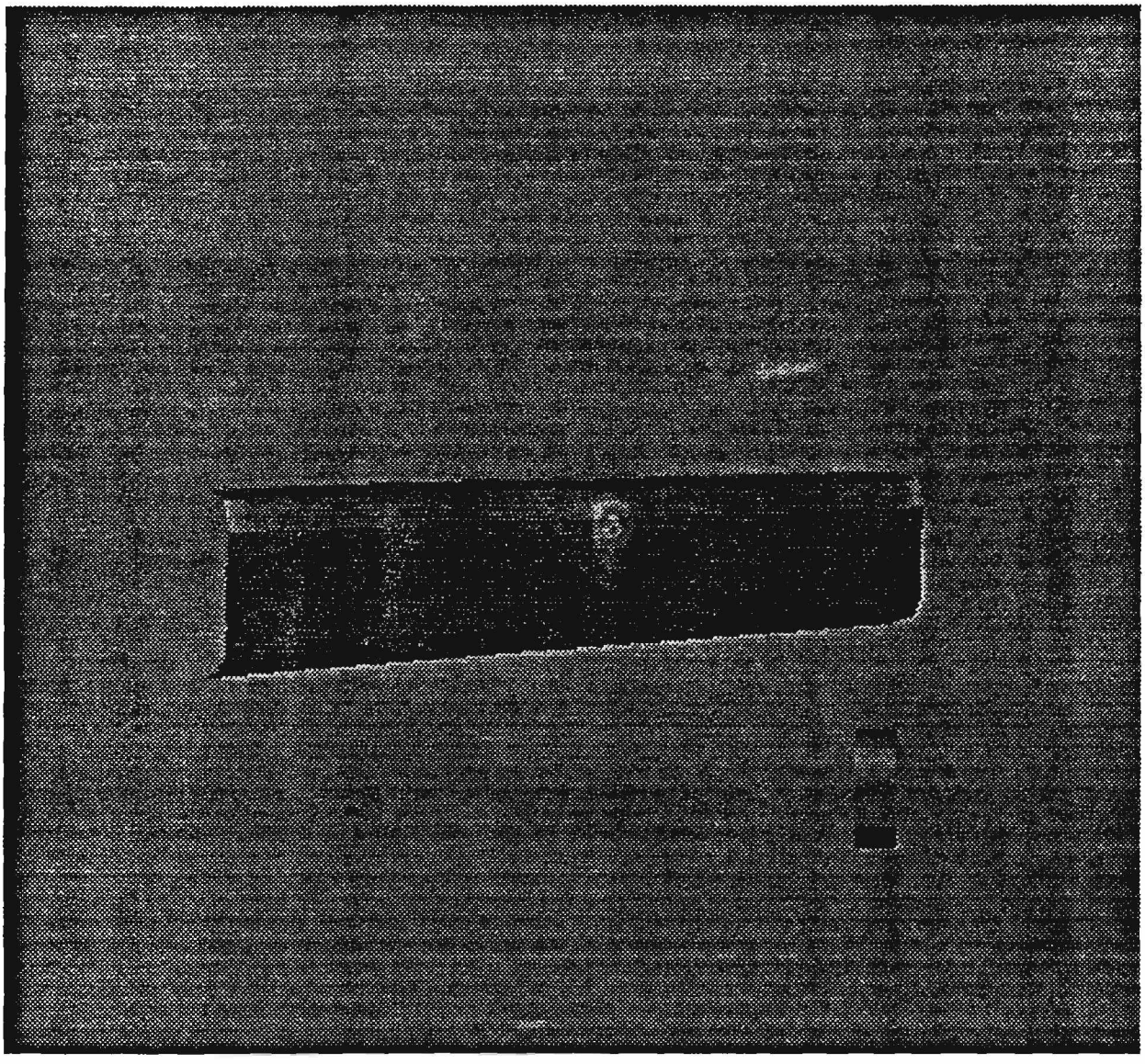

Figure 19: The SED of the Bumper

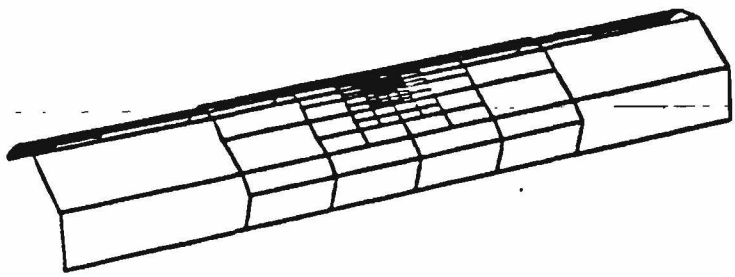

Figure 20: The Optimal Mesh of the Bumper

function call such as

```
(efem 'Shell (list psl:FinalSpoon))
```

will trigger firing rules and start to update the knowledge base.

The boundary configuration (Figure 21) consists of a region that is obviously named. In this analysis, the end of the bowl part (right boundary) will be held firmly so that there will be no degree of freedom here. A displacement constraint, (Right Xt Yt Zt Xr Yr Zr), is given to satisfy the right boundary constraint. A boundary loading will be applied to the end of the handle part (left boundary) in negative Z direction. Thus, the loading vector was specified as (1 0 0 -100) because the spoon was modeled in the X-Y plane. In this example, *EFEM* also set the material properties by allowing the user to select chromium as a material name. The mesh density was selected as an arbitrary option to avoid too many meshes. The uniform mesh (Figure 22) was generated by the subdivision of a spoon surface with the size of 6 × 30. The deformation mesh is also constructed as in previous examples and is shown in Figure 23 after applying the boundary loading vector onto the handle part.

The strain energy field is attached into the original surface forming a criterion surface. The color interpolation of the strain energy field is shown in Figure 24. The highest strain energy (5.433 × 10^7) occurs around the joint area of the bowl and the handle part. The tip of bowl, which is constrained not to move in X, Y, and Z direction, also shows the high SED due to the restriction of movement.

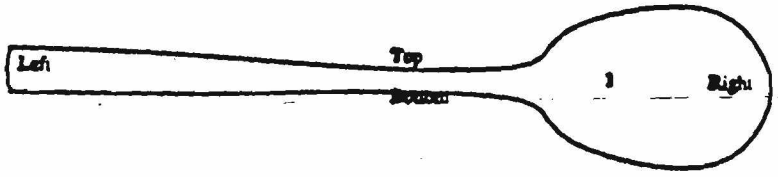

Figure 21: The Bounary Configuration of the Spoon

Figure 22: The Uniform Mesh of the Spoon

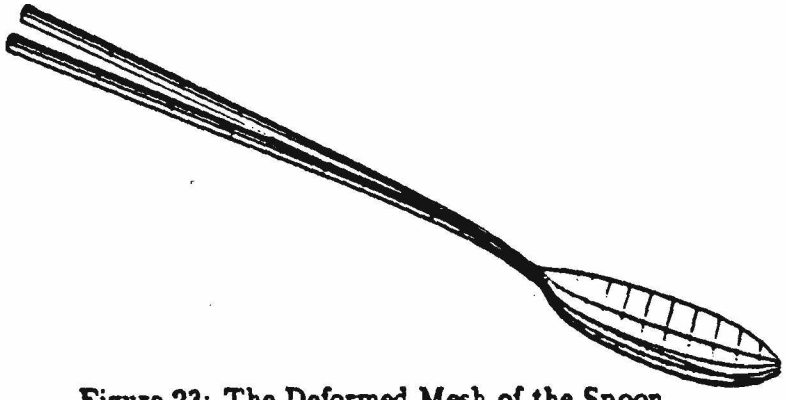

Figure 23: The Deformed Mesh of the Spoon

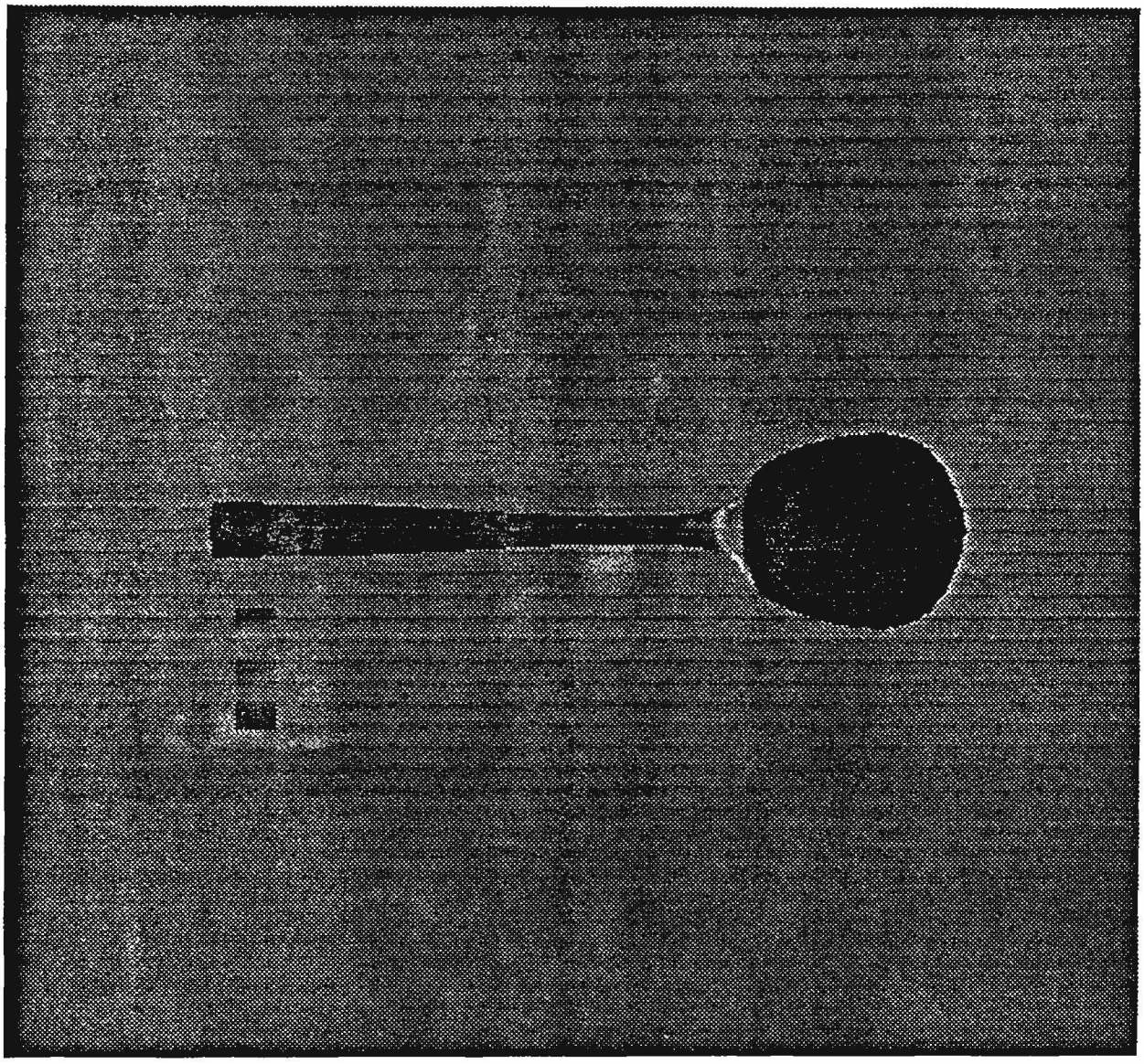

Figure 24: The SED of the Spoon

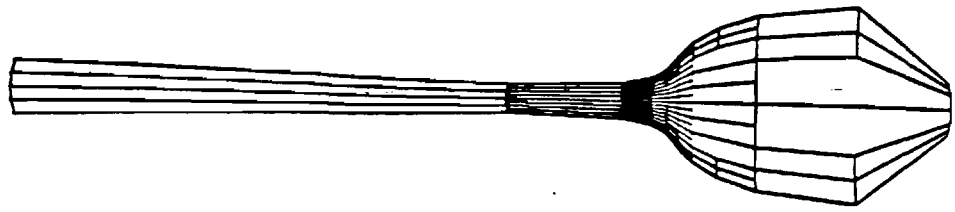

Figure 25: The Optimal Mesh of the Spoon

Table 4: Mesh Analysis for a Spoon

Items	Uniform Mesh	Optimal Mesh
Number of elements	180	100
Number of nodes	217	137
Maximum strain energy	54.320e+06	0.794e+06
Total strain energy	1437.0e+06	15.91e+06
Total degrees of freedom	420	260
Maximal XX stress values	17.17e+06	1.395e+06
Maximal YY stress values	7.807e+06	2.174e+06
Maximal ZZ stress values	8.591e+06	0.790e+06

The optimal mesh (Figure 25) is also generated by the subdivision technique of the B-splines surface as explained in previous examples. The given critical value was 50% of the maximum strain energy in this case. Table 4 summarizes the analysis result of uniform meshes and optimal meshes. The number of elements in optimal meshes has been reduced by 55.5% of one of uniform meshes; however, the total strain energy of the optimal meshes is only 1.1% of one of uniform meshes. As depicted in Figure 25, the boundary curves of the optimal mesh were not good in boundary approximation because the strain energy was relatively low around the bowl part compared to the connection part so that the subdivision of surfaces could not occur there.

7 Conclusion

As science and technology develops, it is becoming more and more necessary to use the finite element analysis technique in various areas. A user should design an object model precisely and analyze it correctly. Recently many researchers have tried to develop algorithms that reduce the effort in the input preparation and interpret the output of finite element analysis easily. However, that kind of task is still a time-consuming and error-prone step. Furthermore, it requires that a user have background knowledge of both computer-aided geometry and finite element analysis.

In this research an expert system for finite element analysis (*EFEM*) has been developed to overcome the difficulties of structural analysis processes and to spread the finite element analysis expertise.

The first analysis with uniform mesh calculates deformation and generates a criterion surface that is the result of synthesizing the strain energy distribution (SED) with the problem in domain geometry, such that the strain energy value takes place in the fourth component of the geometry field. This criterion surface is recursively subdivided into four subsurfaces until the SED over subsurfaces is less than the specified critical value. The optimal mesh was constructed for the final state of subdivided criterion surfaces.

7.1 Performance

The implemented system was verified with planar domain and shell domain data. It demonstrates that the expert system can be a useful tool that helps many industrial areas. The computer time required for an analysis is closely related to the mesh size and it takes a noticeably longer processing time, if the number of meshes is over 200. The computer time, measured in CPU seconds, for the generation and the analysis of uniform meshes with their mesh sizes is summarized for four objects in Table 5. Time for *ADINA* is only for generation of uniform meshes assuming an input data file has been correctly created, while the time for *EFEM* includes all process time such as setting attributes and writing the input file. The column of 'complete analysis' represents the total duration of a complete analysis by *EFEM*. The ratio of total analysis duration by *EFEM* and *ADINA* is calculated in the last column. It is assumed that *ADINA* took the same time as analysis for preparing an input data file for the uniform meshes. The table shows that *EFEM* is robust for larger meshes, although these larger meshes do take longer due to the length of the *ADINA* processing, with *EFEM* taking almost constant time for any number of elements.

7.2 Comparison to Existing Systems

It was tedious and error prone to use early finite element analysis packages because of their poor interfaces. Recently some packages such as *NAVGRAPH* [Ben88] and *SUPERTAB* [SDR88] tried to solve the difficulties of early systems. However, their interfaces have disadvantages such as deeply nested menu items or a node-oriented mesh generation approach rather than an object-oriented one.

While other systems require extensive knowledge to generate optimal meshes, *EFEM* can handle many tedious steps easily. However, most general purpose analysis packages have a wider selection of problem domains such as beam, truss, thick shell and fluid flow besides the two-dimensional plate and shell, while *EFEM* supports the latter domains only.

7.3 Summary

This thesis work demonstrates that an expert system can be applied to generate optimal mesh in an efficient and intelligent way. Since analysis through an expert system does not require technical knowledge about the finite element analysis process, the implemented rule base generates optimal mesh with few interactions. After the first optimal mesh has been generated, other optimal mesh is easily generated by changing the critical value of the subdivision. Furthermore, the rule system demonstrated that the different analyses on the same object can be done easily by controlling rules. Considering the difficulties of existing systems, it is expected that the knowledge based system would contribute substantial benefits to industrial areas.

8 Appendix A: A Sample Session Of *EFEM*

Following is the script file of *EFEM* interactive session for the analysis of I-shaped aluminum sample plate with rule verbose option on. In this example the initial uniform mesh was selected as Coarse Mesh option and the optimal mesh was generated by the critical value of half of highest strain energy. After the first cycle is finished, the critical value was reduced as one fourth of the highest strain energy again.

```
shape\_edit -  Alpha_1 PSL 3.4 Shape Editor with PCLS, 20-May-88
8 lisp (1) [USER:] > !>(efem '2d (list
8 lisp (1) [USER:] > !>      (list psl:L1 psl:L2 psl:L3 psl:L4)
8 lisp (1) [USER:] > !>      (list psl:L2 psl:L7 psl:L5 psl:L6)))
Type Analysis Heading -> !>"I-shaped aluminum sample piece"
```

Table 5: Performance of *EFEM*

Analysis Objects	Mesh Size	Uniform Mesh Generation (sec)	Complete Analysis (sec)	ADINA (sec) (Uniform)	EFEM/ADINA
I-shaped	72	96.4	305.5	22.0	6.9
Square	72	134.1	327.7	24.3	6.7
Bumper	256	556.3	1337.0	462.0	1.4
Spoon	180	411.0	1696.4	701.2	1.2

```
Type Data Group Name -> !>"I"
Reading geometry from points or curve...
Calculating Bounding box for 2D...
Calculating centroid point...
Looking around points to find top and bottom...
Looking around points to find top and bottom...
Looking around points to find top and bottom...
Looking around points to find top and bottom...
Assigning a point for final sequence...
Assigning a point for final sequence...
Assigning a point for final sequence...
Assigning a point for final sequence...
Moving to next region...
Finding a region which has a shared edge...
Finding shared points...
Being successful to find shared edge...
Calculating bounding box for other regions...
Finding the opposite edge...
Keeping the final points sequence temporarily...
Checking the consistency of final sequence...
Setting the final points sequence permanently...
Building boundaries from curves...
Displaying boundaries...

Boundary list is stored into Boundi.ai
Done with geometry class...
Building all regions...
Select mesh type from Coarse, Normal, Dense, or Arbitrary -> !>C
Filling dimensions...
Filling next dimension...
Finding neighbors, Top...Bottom...Left...Right...
Building PSL regions...
```

Type	Name	Type	Name	Type	Name
1	Aluminum	2	Acryl	3	Chromium
4	Copper	5	Glass	6	Gold
7	Iron	8	Nickel	9	Platinum
10	Silver	11	Tin	12	Tungsten
13	Wood(oak)	14	Yellow Brass		

```
Select your material type number -> !>1
Propagating material id to other regions...
Attaching Poisson's ratio to regions...
Attach Young's module into regions...
Enter the thickness -> !>0.1
Propagating thickness to other regions...

Enter load vector in a list form of (RegionId X Y Z) where
X, Y, and Z are loading vectors such as (1 1 0) or (2 0 -1 0) ->
!>(1 0 100)
Attaching thickness into other regions...
Is this Corner or Boundary load? -> !>B
Which Side: 1)Top 2)Bottom 3)Left or 4)Right -> !>1

Displacement Constraint can be specified the following form;
((Side1 Constraint1a Constraint1b...)
 (Side2 Constraint2a Constraint2b...)...)
where constraints are Xt Yt Zt Xr Yr Zr.
For example, ((left Xt)(bottom Yt Yr))
Enter your Displacement constraint for Region 1
> !>()
Enter your Displacement constraint for Region 2
> !>((bottom Xt Yt))
Filling boundary constraint...
Attaching all information...
Generating uniform mesh for 2D...
Enter symmetric information, 0:None 1:X 2:Y 3:Z 4:XY 5:XYZ -> !>0
Displaying uniform mesh...

Uniform mesh is stored into Uniform2.a1
Writing FEM input file...
Running Adina for Uniform mesh...
Evaluating mesh...
Finding reasonable display scale...
Drawing deformation picture...
Deformation file is stored into I-deform.a1
Building criterion surface...
Drawing criterion surface(2D)...
Criterion surface file is stored into I-crit.a1
Calculating(Asking) subdivision critical value...
Generating domain for near-optimal mesh...
Writing FEM input file for near-optimal mesh...
```

```
Running Adina for near-optimal mesh...
Evaluating near-optimal mesh...
Drawing near-optimal mesh...
Near-optimal mesh is stored into I-N0.a1
Asking next job...

Requested analysis is finished. Select next choice.
 <Q>uit, <C>ritical value change, or <L>oading vector change -> !>C
Changing subdivision critical value...
The current subdivision critical value is 0.00109262.
Type a new subdivision critical value -> !>0.00054631
Generating domain for near-optimal mesh...
Writing FEM input file for near-optimal mesh...
Running Adina for near-optimal mesh...
Evaluating near-optimal mesh...
Drawing near-optimal mesh...

Near-optimal mesh is stored into I-N0.a1
Asking next job...

Requested analysis is finished. Select next choice.
 <Q>uit, <C>ritical value change, or <L>oading vector change -> !>Q
Process EFEM is finishing...
Setting symmetric information...
Propagate displacement constraints...
Getting displacement constraint...
Getting loading information...
Filling thickness...
Defining material...
Calculating(Asking) mesh density...
NIL
NIL
9 lisp (1) [USER:] >
```

References

[ADI81] ADINA Engineering. *ADINA User's Manual.* ADINA Engineering, Inc., Sept. 1981.

[ADI83] ADINA Engineering. *ADINA-IN User's Manual.* ADINA Engineering, Inc., December 1983.

[Ben88] Benzley. *NAVGRAPH User's Manual.* Brigham Young University, Provo, August 1988.

[BWS*87] Peggy L. Baehmann, Scott L. Wittchen, Mark S. Shephard, Kurt R. Grice, and Mark A. Yerry. Robust, Geometrically Based, Automatic Two-Dimensional Mesh Generation. *International Journal for Numerical Methods in Engineering*, 24:1043–1078, 1987.

[Byk76] A. Bykat. Automatic Generation of Triangular Grid: I-Subdivision of a General Polygon into Convex Subregions. *International Journal for Numerical Methods in Engineering*, 10:1329–1342, 1976.

[Cav74] James C. Cavendish. Automatic Triangulation of Arbitrary Planar Domains for the Finite Element Method. *International Journal for Numerical Methods in Engineering*, 8:679–696, 1974.

[CFF85] James C. Cavendish, David A. Field, and William H. Frey. An Approach to Automatic Three-Dimensional Finite Element Mesh Generation. *International Journal for Numerical Methods in Engineering*, 21:329–347, 1985.

[CH81] G. F. Carey and D. L. Humphrey. Mesh Refinement and Iterative Solution Methods for Finite Element Computations. *International Journal for Numerical Methods in Engineering*, 18:1717–1734, 1981.

[CLR80] Elaine Cohen, Tom Lyche, and Richard Riesenfeld. Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics. *Computer Graphics and Image Processing*, 14:87–111, 1980.

[Cob84] Elizabeth Cobb. *Design of Sculptured Surfaces Using the B-spline Representation.* PhD thesis, University of Utah, June 1984.

[Cou43] R. Courant. Variational Methods for the Solution of Equilibrium and Variations. *Bulletin of the American Mathematical Society*, 49:1–23, 1943.

[FS72] J. Fukuda and J. Suhara. Automatic Mesh Generation for Finite Element Analysis. *Advances in Computational Methods in Structural Mechanics and Design*, 607–624, 1972.

[FWE70] C. O. Frederick, Y. C. Wong, and F. W. Edge. Two-Dimensional Automatic Mesh Generation for Structural Analysis. *International Journal for Numerical Methods in Engineering*, 2:133–144, 1970.

[Gal75] Richard H. Gallagher. *Finite Element Analysis Fundamentals.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[JS86] B. Joe and R. B. Simpson. Triangular Meshes for Regions of Complicated Shape. *International Journal for Numerical Methods in Engineering*, 23:751–778, 1986.

[JT80] Chris L. Jackins and Steven L. Tanimoto. Oct-Trees and Their Use in Representing Three-Dimensional Objects. *Computer Graphics and Image Processing*, 14:249–270, 1980.

[Kik86] N. Kikuchi. *Finite Element Methods in Mechanics.* Cambridge University Press, Cambridge, 1986.

[Log86] Daryl L. Logan. *A First Course in the Finite Element Method.* PWS Engineering, Boston, 1986.

[LPS84] 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*, 3(4):287–311, October 1984.

[MM77] R. J. Melosh and P. V. Marcal. An Energy Basis for Mesh Refinement of Structural Continua. *International Journal for Numerical Methods in Engineering*, 11:1083–1092, 1977.

[Pis81] S. Pissanetzky. KUBIC: An Automatic Three-Dimensional Finite Element Mesh Generator. *International Journal for Numerical Methods in Engineering*, 17:255–269, 1981.

[Red84] J. N. Reddy. *An Introduction to the Finite Element Method.* McGraw-Hill Book Company, New York New York, 1984.

[Rud88] B W Rudd. Impacting the Design Process Using Solid Modelling and Automated Finite Element Mesh Generation. *Computer-Aided Design*, 20(4):212–216, May 1988.

[SDR88] SDRC. *I-DEAS Supertab, Engineering Analysis User's Manual.* Structural Dynamics Research Corporation, 1988.

[SGA80] Mark S. Shephard, Richard H. Gallagher, and John F. Abel. The Synthesis of Near-Optimum Finite Element Meshes with Interactive Computer Graphics. *International Journal for Numerical Methods in Engineering*, 15:1021–1039, 1980.

[WA79] Sheng-Chuan Wu and John F. Abel. Representation and Discretization of Arbitrary Surfaces for Finite Element Shell Analysis. *International Journal for Numerical Methods in Engineering*, 14:813–836, 1979.

[WPWW87] P. Ward, D. Patel, A. Wakeling, and R. Weeks. Application of Structural Optimization Using Finite Elements. *Computer-Aided Design*, 19(3):148–156, April 1987.

[WT84] Tony C. Woo and Timothy Thomasma. An Algorithm for Generating Solid Elements in Object with Holes. *Computers & Structures*, 18(2):333–342, 1984.

[Yen85] Wu-chien J. Yen. *On Representation and Discretization of Finite Element Analyses.* PhD thesis, University of Utah, December 1985.

[YFRC87a] Wu-chien J. Yen, Russell D. Fish, Richard F. Riesenfeld, and Elaine Cohen. *An Algorithmic Approach Toward Near-Optimum Finite Element Mesh Generation.* Technical Report, Dept. of Computer Science, Univ. of Utah, June 1987. (27 Pages).

[YFRC87b] Wu-chien J. Yen, Russell D. Fish, Richard F. Riesenfeld, and Elaine Cohen. *An Attribute Modelling Technique for Problem Specification and Result Interpretation in Finite Element Analyses.* Technical Report, Dept. of Computer Science, University of Utah, June 1987. (38 Pages).

[YS83] Mark A. Yerry and Mark S. Shephard. A Modified Quadtree Approach to Finite Element Mesh Generation. *IEEE Computer Graphics and Application*, 3:39–46, January 1983.

[YS84] Mark A. Yerry and Mark S. Shephard. Automatic Three-Dimensional Mesh Generation by the Modified-Octree Technique. *International Journal for Numerical Methods in Engineering*, 20:1965–1990, 1984.

[YS85] Mark A. Yerry and Mark S. Shephard. Trends in Engineering Software and Hardware - Automatic Mesh Generation for Three-Dimensional Solids. *Computers & Structures*, 20(1):31–39, 1985.

[Zie77] O. C. Zienkiewicz. *The Finite Element Method.* McGraw-Hill, Inc., New York, 1977.