THREE-DIMENSIONAL FINITE-ELEMENT MESH GENERATION USING SERIAL SECTIONS

by

Toufic I. Boubez, B. Eng.

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering

Department of Electrical Engineering
McGill University
Montréal, Canada
November, 1986

To my parents,

ABSTRACT

This thesis presents a new method for automatic generation of three-dimensional tetrahedral finite-element meshes within objects. This method is especially useful when the object to be modelled has a highly irregular shape. The base geometric model used is a set of serial cross-sections of the object. These sections are first triangulated on a grid of nodes. The object is then divided into slices, each two consecutive cross-sections constituting a slice, and each slice is meshed separately. For each slice, nodes on the top and bottom surface triangulations are joined to form a central mesh of pentahedral prisms. These prisms are in turn shredded into three tetrahedra each to form the core of the tetrahedral mesh. The remaining outer layer of the slice is then assembled into a polyhedral structure and meshed using four operators that work on the topology of the polyhedron by cutting tetrahedra from it. The meshed slices are finally merged to form the global mesh. The shapes of the elements in this mesh are improved by relaxing the internal mesh nodes to an equilibrium position where each node is at the center of gravity of the other nodes attached to it.

SOMMAIRE

Cette thèse présente une méthode nouvelle pour la production automatique de maillages tri-dimensionnels dans un objet, pour la méthode d'éléments finis. Cette méthode est particulièrement utile quand l'objet étudié a une forme très irregulière. Le modèle géométrique de base utilisé est une série de contours de sections de l'objet étudié. Ces sections sont d'abord triangulées sur une grille de points, puis l'objet est divisé en tranches, chaque tranche étant déterminée par deux sections consécutives. Chaque tranche est ensuite maillée individuellement. Pour chaque tranche, des points sur les surfaces supérieures et inférieures sont joints, formant un maillage central de prismes. Ces prismes sont à leur tour divisés en trois tétraèdres chacun, complétant le maillage central. La couche extérieure restante est ensuite assemblée en une structure polyédrale et maillée à l'aide de quatre opérateurs agissant sur la topologie du polyèdre en coupant des tétraèdres. Les tranches maillées sont ensuite assemblées pour former le maillage final. Les formes des éléments dans ce maillage sont améliorées par une méthode de relaxation des points aboutissant à une position d'équilibre où chaque point est situé au centre de gravité de tous les autre points qui lui sont attachés.

ACKNOWLEDGEMENTS

I am very indebted to my two supervisors. Prof. W. Robert J. Funnell who has inspired this work and guided me through it. He has always been extremely supportive and available to discuss anything: cars, life in general and in particular, and even my work (in that order of importance, of course). Prof. Dave A. Lowther for his valuable advice, his support and his patience through all those deadlines that I missed and for being a role model (for the Canadian ski marathon, that is). They both have been extremely supportive through good times and bad times.

Sue Morrison for her inexhaustible support all along and for convincing me that I could actually finish my thesis. I think she was right.

Prof. Godfried Toussaint for teaching me about computational geometry and for his useful comments and help in locating some hard-to-find papers.

The Biomedical Engineering Unit for its great working environment and extremely helpful and competent staff, especially Mr. Carmelo Granja and Mr. Serge Lafontaine.

My friends at the CADLab for providing the entertainment, and especially Andy Froncioni for browbeating me into submitting the thesis, and for rolling up his (Tee-)shirtsleeves at numerous times to help me debug...

Last but definitely not least, to my loving family which has endured everything without complaints and with unwavering faith. They have watched me gradually turn my room into an impenetrable jungle, tolerated my extremely odd behaviour and hours and let me get away with doing less than my share of taking the dog for a walk. There it is, mom, I'm finished...

...for now.

This work was supported by the Medical Research Council of Canada.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 The Finite-Element Method

Solutions to a large class of problems in engineering analysis consist of solving a set of governing differential equations, or minimizing an energy functional over a domain, given some boundary conditions. Analytical methods provide exact solutions to some simple classes of differential equations, but for many interesting two- and three-dimensional problems, exact solutions are impossible to achieve. This can be due to the complexity of the equations in question, the complexity of the domain, or both. The finite-element method has proven to be an excellent computational tool for arriving at numerical solutions to such complex problems [Gallagher 1975, Silvester & Ferrari 1983, Stasa 1985, Strang & Fix 1973, Zienkiewicz 1977].

In the finite-element method, the problem domain is first discretized into a finite number of simpler sub-domains, or elements. A solution is then approximated over each of these elements individually, in terms of the boundary conditions on the element. For each element, the local equations are put in matrix form and used to construct a global system matrix, using the fact that the element boundary conditions are identical at each element-to-element interface. This final matrix defines a set of simultaneous equations that are solved for the global solution.

The method was originally extensively used in two-dimensional problems. However, advances in computer technology in the form of faster machines and better

algorithms are making the solutions of three-dimensional and larger two-dimensional problems possible. These include problems in stress analysis and structural design [Bathe et al. 1973, Zienkiewicz et al. 1970], fluid mechanics and heat transfer [Baker 1983, Chung 1975], electromagnetics [Chari 1980, Silvester & Ferrari 1983, Zienkiewicz 1980], and recently in biomedical engineering [Funnell & Laszlo 1978, Little et al. 1986, Murai & Kagawa 1985, Yamashita & Takahashi 1984].

1.2 The Finite-Element Mesh

The decomposition of an N-dimensional domain into finite elements is called an N-dimensional finite-element mesh, and has to satisfy two main conditions. (1) The union of all the elements in the domain is the domain itself; and (2) The intersection of any two elements is either null or a full i-dimensional face, where $0 \leq i < N$. Thus for a three-dimensional mesh, any intersection is either null, a point, a full edge or a full face. Furthermore, and as a softer requirement, to produce optimal results, the mesh elements must be as 'equilateral' as possible avoiding internal angles that are too large or too small [Babúska & Aziz 1976, Hermeline 1982, Sibson 1978].

Clearly, the size and shape of the elements affects the complexity of the solution and the accuracy of the results and several element shapes have been developed and used for different environments and purposes. Argyris 1965, Argyris 1968, Clough 1969, Hughes & Allik 1969]. However, the most widely used elements have been the triangle and quadrilateral in two-dimensional analysis, and the tetrahedron, wedge (pentahedron) and brick (hexahedron) in three dimensions. In addition, the edges of these elements need not be straight, and curved elements having second order (quadratic) or higher order edges can also be used. Arguments have been proposed as to the superiority of one shape or the other, but usually op-

timality is dictated by several factors, including the desired precision of the results, the size and speed of the computer being used, the geometry of the problem and the cost limit of the analysis.

Tetrahedra will be used in this work for several reasons. Tetrahedral elements are three-dimensional simplices and thus have the smallest number of nodes, faces and edges of all three-dimensional elements. This makes them simpler to manipulate before, during and after the analysis (preprocessing, processing and postprocessing) They are also compatible with most finite-element solver software packages so that the output of a tetrahedral mesh generator can be readily used. In addition, the topological three-dimensional simplex properties of tetrahedra make them ideal for meshing irregularly shaped three-dimensional objects, since any polyhedral object can be described as the union of a set of tetrahedra (Alexandroff 1961).

1.3 Thesis Outline

This thesis proposes an automatic mesh generation method that can be used to mesh arbitrarily shaped objects. This is particularly important in the biomedical field where such shapes abound. The method takes advantage of an easily available data representation by using serial sections of an object as a geometric model.

Chapter 2 provides a survey of the literature on three-dimensional mesh generation techniques with a description of the different methods and comments on the pros and cons of each. It also presents the motivation for this work along with a brief overview of the proposed method.

Chapter 3 introduces the concept of serial sections as a geometric model of the object being analyzed. The data acquisition method is described and the two-dimensional triangular meshing technique that provides a basis for the proposed

three-dimensional method is discussed and its purpose explained.

In Chapter 4 a high-speed method for meshing the core of an object, based on the two-dimensional technique, is presented. This involves creating a regular mesh of triangular prisms and shredding it into tetrahedra. Two theorems on shredding pentahedral prisms are given and proven.

Chapter 5 is concerned with assembling the remaining outer layer of the object and checking its topological consistency. A description of the data structure employed will also be given.

In Chapter 6 Eulerian mesh operators are introduced, and the topological mesh generation technique is described. Previous attempts at providing a complete set of operators will be shown to be lacking and a more complete set will be presented. Three classes of irreducible polyhedra will be introduced and discussed.

A method to measure the quality of the generated mesh and a way to improve the shape of the elements through node relaxation will be proposed in Chapter 7.

The results of the mesh generation and relaxation are presented in Chapter 8.

Finally, Chapter 9 contains an evaluation of the method and comparisons to other methods, along with some further considerations and possible developments.

The reader unfamiliar with topological concepts or the notation used is referred to the Appendix, where a short review of the essential geometric and algebraic topology concepts is presented.

CHAPTER 2

Three-dimensional mesh generation methods

2.1 Introduction

In the finite-element analysis of three-dimensional problems, a major stumbling block has always been that of problem description to the computer. This includes the generation of the three-dimensional finite-element mesh to span the problem domain. Since manual mesh generation is very error prone and time consuming, designers have been seeking to automate the mesh generation procedure as much as possible. Generating two-dimensional meshes automatically is now common practice and done using any one of several methods [Heighway & Biddlecombe 1982, Kamel & Eisenstein 1970, Thacker et al. 1980, Watson & Philip 1984, Zienkiewicz & Phillips 1971]. In three dimensions, on the other hand, automatic, or even interactive mesh generation is not so simple. Due to the importance of the problem in design analysis, quite a few mesh generators exist today, and they can be loosely grouped in four major classes. This chapter will review the different techniques used in three-dimensional mesh generation, and present the motivation for the new method proposed in this thesis, along with a brief description of the method.

2.2 Interactive Graphics

This form of mesh generation [Biffle & Sumlin 1977, Bryant & Freeman

1984, Kamel & Shanta 1974, Kirchhoff 1974, Perucchio et al. 1982, Thornton 1978] is the closest to manual mesh generation in that all the aspects of the procedure are controlled by the user. The main difference lies in avoiding manual entry of nodal coordinates. Instead, the computer is used to visualize the mesh as it is created, and graphic input devices are used to define coordinates. Compared to manual mesh generation methods, this reduces the possibility of error and saves time in generating the mesh. The main problem here lies in the two-dimensional nature of the traditional graphics devices making the proper entry and viewing of three-dimensional data very difficult for anything but very simple shapes. Three-dimensional viewing devices [Fajans 1979, Harris et al. 1986, Jansson et al. 1979, Okoshi 1980, Stover 1981] being still too expensive for widespread use, several methods for improving the two-dimensional display of three-dimensional models have been tried. Some, for example, use projective views of the object being meshed to help the user locate and define nodes in space. Typically, a node is defined by pointing at two different projections in two windows and the three-dimensional coordinates are reconstituted accordingly from the projections. Other methods rely on perspective views, hidden-line removal, real-time rotation of the object or any combination of these to create the same effect. In addition, such utilities as DELETE, MERGE, JOIN, etc. are provided to facilitate certain aspects of the mesh generation, so that complex objects can be divided into simpler modules that are meshed individually and joined later. These generators are usually highly interactive and demand a large amount of work from the user. They are also very impractical for large or complicated objects, and cannot be used in batch mode to save on processing costs. On the other hand, they provide a large amount of control and flexibility in creating the mesh, are easier to implement and require less memory, a fact that can become important where smaller machines are concerned.

2.3 Mapping Techniques

The term mapping is used in this context to include both interpolation mappings [Cook 1974, Haugerud 1978, Imafuku et al. 1980] and deformation mappings [Coulomb et al. 1984, Pissanetzky 1981], methods that have their roots in two-dimensional mesh generation mapping techniques [Kamel & Eisenstein 1970, Zienkiewicz & Phillips 1971], and are by far the most prevalent today. In three-dimensional interpolation methods, the object is defined by a set of cross-sectional curves matching its surface at a number of cutting planes, and by a set of corresponding interpolation functions between the cross-sections. The same number of nodes is generated and joined inside each cross-section, forming a two-dimensional mesh at each level. Corresponding nodes on consecutive sections are then joined by interpolating between them. The resulting three-dimensional elements depend on the nature of the two-dimensional meshing at the cross-section level. Thus brick elements are formed if the cross-sections are divided into quadrilaterals (Figure 2.1.a) and wedge elements are formed where triangular elements are used (Figure 2.1.b).

Deformation methods on the other hand start with an existing topologically correct regular mesh in a parametric space. This parametric mesh is then deformed into the real object space so that the nodes on its surface are relocated to lie on the surface of the object being meshed. This can either be done interactively or by choosing some suitable functions that map the parametric coordinates of the object into Euclidian space (Figure 2.2).

These methods give excellent results for regular shapes possessing some form of cross-sectional symmetry and where the shape of consecutive cross-sections does not change drastically, but present a few problems in other geometries. For example, since the same number of nodes is distributed inside each cross-section, nodes

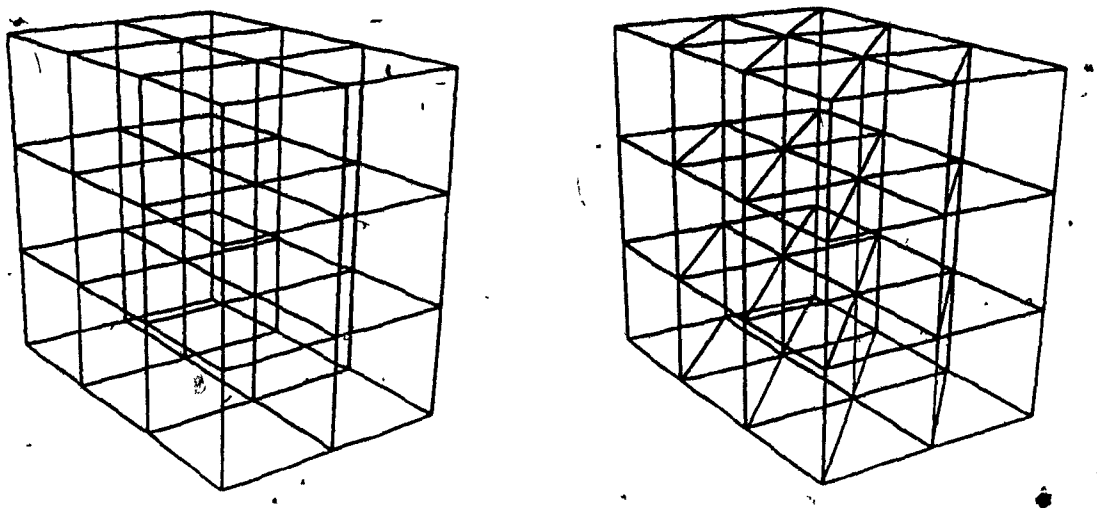

Figure 2.1

Bricks and wedges.
a) Quadrilaterals lead to brick-shaped elements.
b) Triangles lead to wedge-shaped elements.

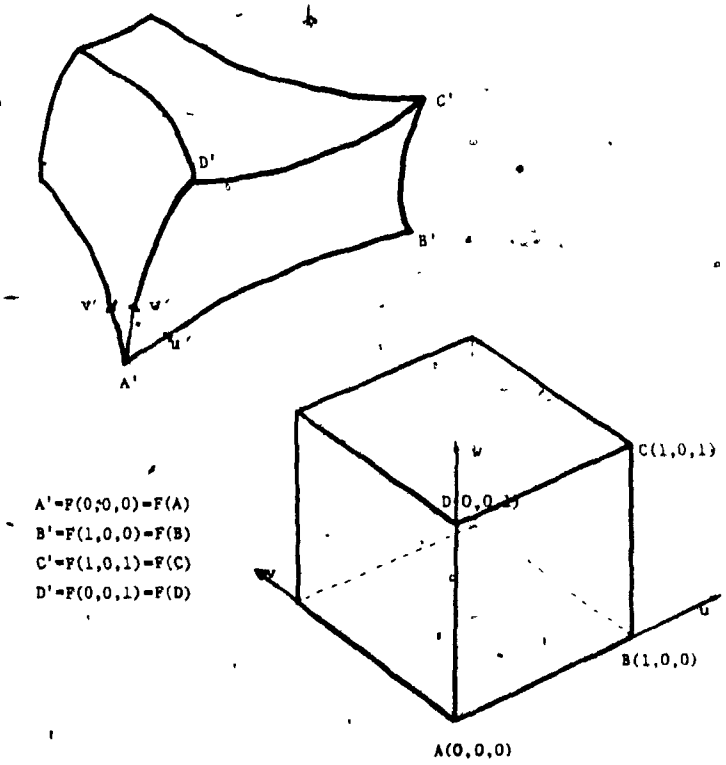

Figure 2.2

Mapping between spaces.
The function F maps the parametric unit brick into Eúclidian space.

in increasingly narrowing cross-sections become uselessly cramped in the smaller cross-section, resulting in an increase in the number of nodes used to solve the problem, and the production of badly shaped, narrow elements. Another problem is encountered when meshing highly irregular shapes. In these instances interpolations from one cross-section to the next often result in bad element shapes in interpolation methods, and a proper mapping function is extremely hard to find for deformation methods. Although requiring some initial directions from the user, mapping methods can be set up for batch processing and can be relatively inexpensive computationally.

2.4 Filling Techniques

These methods are generally used to produce meshes of tetrahedral elements. They consist of generating nodes inside the object to be meshed and filling it with elements by generating tetrahedra to surround the nodes. The most widely used method to achieve this is that of Delaunay triangulations [Brostow et al. 1979, Cavendish et al. 1985, Hermeline 1982, Watson 1981]. For a set of n nodes, a Voronoi polyhedron V_i associated with a node p_i is defined as the set of points that are closer to p_i than any other node:

$$V_i = \{x;\ \forall j \neq i,\ 1 \leq j \leq n,\ d(x,p_i) \leq d(x,p_j)\}$$

This definition implies a partition of the domain into distinct polyhedra. Joining neighbouring nodes (i.e. nodes whose corresponding Voronoi polyhedra are adjacent) produces the dual of this partition, the Delaunay triangulation, which is a valid tetrahedral meshing of the domain, except for some special cases which can be detected and remedied. The two-dimensional case is shown in Figure 2.3.

Other interesting techniques have also been proposed. In [Nguyen-Van-Phai 1982] for example, an edge joining two vertices in the set is selected and surrounded

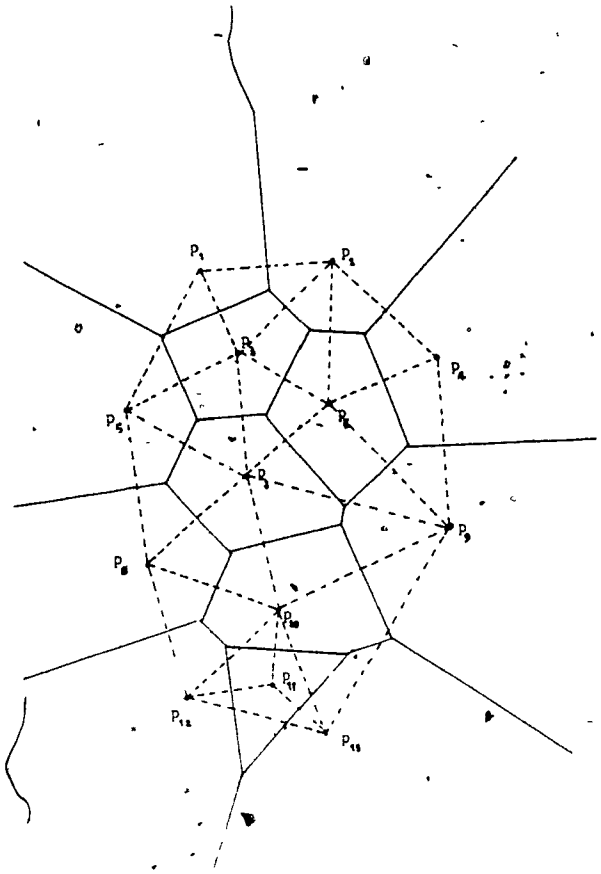

Figure 2.3

2-D Delaunay triangulation
A partition of the plane, by a set of points $p_1, \ldots p_{13}$ into 13 Voronoi polygons, and the corresponding Delaunay triangulation.

with tetrahedra defined by nearby vertices. This procedure is repeated for all the generated edges, making sure that the newly formed tetrahedra do not intersect already existing ones.

All these methods have similar problems. The first one is that for any non-simple shape, generating the nodes inside the region is not a trivial operation. Second, the resulting mesh is a three-dimensional triangulation of the *convex hull* of the generated nodes, irrespective of the actual shape of the object. A substantial amount of additional work is then required to take care of holes and concavities. Third, these methods can be computationally expensive for large systems of nodes. The main advantage of node-insertion methods like the Delaunay triangulation is that they are naturally suited to adaptive mesh refinement techniques, where a coarse mesh is first constructed and subsequently refined by inserting nodes in the locations of greatest error [Cendes & Shenton 1985].

2.5 Cutting Techniques

More recently, topology and topological operations have taken a more prominent role in three-dimensional mesh generation [Ewing et al. 1970, Mäntylä 1983, Woo & Thomasma 1984, Wördenweber 1984], due to some perceived deficiencies in the traditional approaches. Topological methods involve cutting off tetrahedra from an object, as opposed to filling its volume with tetrahedra as in the previous method. The object is usually defined as a polyhedron with lists of faces, edges and vertices, along with pointers defining their topological relationships. The cuts are done according to rules ensuring the topological consistency of the resulting mesh, without generating any new nodes. The two main operators are common to all the methods, however, and are the actual cutting operators. They act on a structure by removing a tetrahedron identified by a vertex or an edge, respectively. Again,

these techniques are used to produce tetrahedral element meshes, and at least two methods have already been proposed using very similar operators. It will be shown in this thesis, however, that these operators are incomplete, failing on a large class of polyhedra, and a more complete set of operators will be introduced. Generally these methods lend themselves well to automatic mesh generation and produce consistent meshes, even for arbitrary shapes. They are, however, relatively slow and require a polygonal surface description of the object, which can be hard to produce for non-simple shapes.

2.6 Still Another Method?

It must be clear by now that there is a wide variety of three-dimensional mesh generators that are capable of meshing well behaved shapes. On the other hand, these mesh generators are not well suited for meshing irregularly shaped objects. Recently, however, scientists have been interested in three-dimensional finite-element analysis in fields where the objects under study can take on very irregular shapes and are very difficult to mesh properly. The lack of appropriate mesh generators in these cases is apparent. The main objective of this work was thus to develop a fully automatic mesh generator to be used for such shapes.

As for any mesh generation method, this method starts with a geometric model of the object under study. The geometric model in this case consists of a set of serial sections, as described above for the interpolation techniques. Serial sections have been extensively used in several areas like anatomy and geology to describe geometrical properties of objects and shapes, mostly for surface and volume reconstruction [Fuchs et al. 1977, Gaunt 1971, Keppel 1975, Tipper 1976, Veen & Peachey 1977]. In some instances, serial sections are the only available representation of the object under study. Since serial sections are often already available for

objects in the life sciences and can be easily produced in other engineering applications, a mesh generator that takes advantage of this fact would greatly reduce the amount of time required for modelling the object prior to meshing it, while at the same time simplifying a large portion of the meshing procedure.

2.7 The Proposed Method

Serial cross-sections through an object logically divide it into slices of finite thickness, with each two consecutive sections taken to define the top and bottom surfaces of a slice. Although not a requirement, we are assuming here that the cutting planes are parallel. The cross-sectional surfaces are each triangulated on a grid plane, using the same grid for all the sections. This will cause each two consecutive sections, whose projections on the grid plane will usually overlap, to contain identical nodes and triangles within the interlap area. At this point each slice is processed individually.

First a central core of pentahedral prisms is generated by pairing up matching triangles from the upper and lower triangulations, and the prisms are shredded into tetrahedra. The remaining portion of the slice, usually a torus-like structure surrounding the core, is assembled into lists of related triangular faces, edges and vertices, then meshed using a set of four topological operators. When all the slices have been processed, they are put back together to form the whole three-dimensional mesh. This can be easily done because the slices interface exactly at nodes, edges and faces, since, for any two consecutive slices, the top section of the lower slice and the bottom section of the upper one are identical. Finally, the internal nodes are relaxed to improve the shape of the mesh elements.

CHAPTER 3

Setting up the cross-sections

3.1 Introduction

The need to describe the problem geometry to the computer is a problem in itself, since mesh generators usually require a geometric model of the object to be shredded. This problem is usually solved through the use of a geometric modeller which, from some user-provided input data, produces a geometrical description of the object, compatible with the analysis program that will be making use of it. A standard specification for geometric models has been developed [Smith et al. 1983], and several geometric modellers are in use today [Baeret al. 1979], the main requirement being that the geometric properties of the modelled object be described to a certain degree of accuracy. Serial sections of an object certainly define the outside shape of the object with an accuracy proportional to the slicing resolution (the distance between two consecutive slicing planes) and can thus be used, in digitized form compatible with processing on a computer, as a geometric model. In this chapter, the acquisition and digitization of the serial sections will be covered, and the two-dimensional triangular mesh generation method, used to generate the two-dimensional contour meshes that are the base of the three-dimensional mesh, will be described.

3.2 The Serial Sections

Serial sections consist of a set of two-dimensional cross-sectional contours of the object, usually parallel and taken at regular depth intervals. They can be obtained by one of several methods, depending on the nature of the object. Biological sections are, for example, taken either by actually slicing the object and preserving the slice, by computed tomography (CT), or by using a scanning microscope, adjusting the focus at different depth levels and photographing the focus plane. The cross-sections can be taken in any direction, but the node coordinates are adjusted so that the X-Y plane is parallel to the slicing planes and the Z axis parallel to the slicing direction (normal to the slicing planes). This convention is easy to implement and greatly facilitates the modelling.

3.3 The Serial Contours

The geometric data file containing the geometric model of an object is next generated from these cross-sections by storing the X-Y coordinates of points on each of the contours, along with a distinct identifier corresponding to the Z coordinate of that contour. To do that, the origin and X-Y axes have to be identified on each slice, and defined on the digitizing plate. This insures that all contours are digitized on the same frame of reference, therefore correctly describing the object. The axes on the digitizer and on each contour are then superimposed and the contour followed on the plate with a graphic input device such as a stylus. Points on the contour are selected, digitized and sent to the file. Due to the amount of data that can be generated this way, a data organization scheme is usually necessary at this point. The recorded points need not be equidistant on the section contour, nor constrained in any other way, except for being consecutive in one direction on the oriented contour. Since each slice lies at a different depth, the depth value (the Z

coordinate of the slicing plane) has to be input separately along with each slice. The distance between successive Z coordinate levels determines the standard measure, that will be used as a unit step throughout the remaining procedures

3.4 The Uniform Grid Method

Once the contours are obtained, the first step in the mesh generation algorithm is started Its purpose is to generate loosely matching triangular meshes on consecutive sections in such a way that as many of their nodes as possible can be joined by vertical edges to form pentahedral prisms from pairs of triangles (Figure 3.1). This is accomplished by constraining all the triangulations to be performed on the same grid of nodes, following the procedure outlined below Fuhnell 1983

1 Define the grid plane and generate the grid nodes The grid plane is spanned by two oblique integer coordinate axes, the I and J axes, lying at 60 with respect to each other This angle has been chosen in order to obtain a mesh of equilateral triangles when the grid points are joined together (Figure 3.2) The grid lattice is bounded by the parallelogram enclosing all of the contours, and whose lower and left sides are the I and J axes, respectively The I axis is determined by the global minimum Y coordinate in the node coordinates file The J axis is determined by the intersection point of a sliding line, forming a 60° angle with the I axis, and the union of the contours The origin is obviously at the intersection of the two axes The unit distance on the axes is taken to be the slicing resolution measure as defined in section 3.1 Nodes on the plane are identified by their I-J coordinates on the two-dimensional lattice

2 Project the contour to be triangulated onto the grid plane generated in step 1

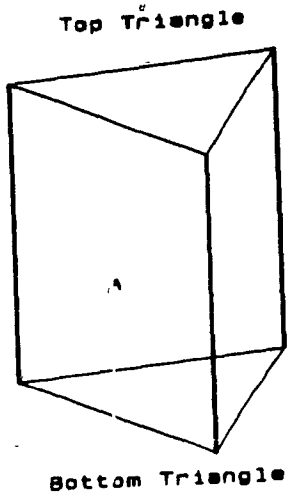

Figure 3.1

Pentahedral prism.
Construction of a pentahedral prism by joining two triangles.

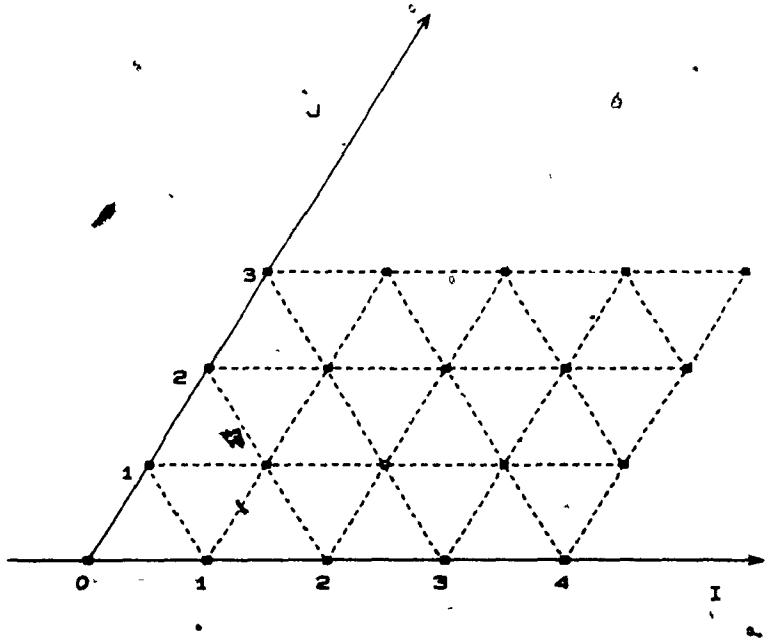

Figure 3.2

The grid of nodes.
Joining the grid points gives a mesh of equilateral triangles.

Since the object has already been aligned with the Z axis, and the grid plane is assumed to be the X-Y plane, the projection can simply be done by setting the Z coordinates of the contour nodes to zero.

3. The contour is redivided into new equal segments as close to unit step length as possible, generating a new set of equidistant boundary nodes and edges of equal length. This is done by calculating the perimeter of the contour, dividing this number by the unit step length and truncating the result to determine the number of nodes that should be fitted on the contour. These nodes are then uniformly distributed on the perimeter of the contour, generating the new segments along the way. The size of these segments will generally be different from the unit step, unless the perimeter happens to be an integer number of steps. This step can be explicitly overridden by the user when the specified contour nodes should not be moved, as in the case where they define a naturally occurring boundary on the object surface, for example.

4. Mark the nodes that lie inside the contour, to within a given tolerance away from the boundary (Figure 3.3). The tolerance is used to eliminate nodes that are too close to the boundary and that could cause narrow triangles to be generated. According to the Jordan curve theorem (Courant & Robbins 1941), a point lies inside a closed curve if any path emanating from that point intersects the curve in an odd number of places. This theorem is used here by checking, for each node, the number of contour edges that intersect a horizontal half-line originating at that node. The node is inside the contour if the half-line intersects an odd number of contour edges.

5. Join marked neighbouring nodes to form a triangular mesh of equilateral

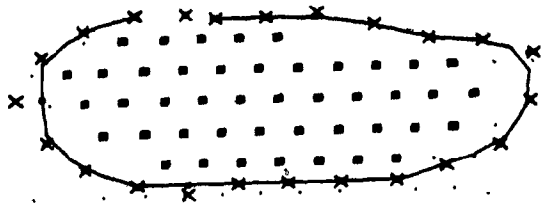

Figure 3.3

Flagging internal nodes.
Nodes that are inside the contour, but not too close to the boundary, are flagged.

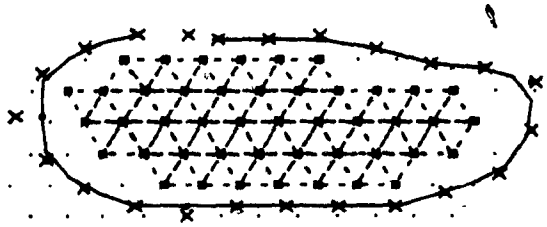

Figure 3.4

Making the equilateral mesh.
The internal nodes are joined to make an equilateral mesh inside the contour.

triangles inside the contour (Figure 3.4).

6. Determine the boundary nodes and edges of this mesh.

7. Join the boundary nodes of the internal mesh to the external contour boundary nodes, surrounding the internal mesh with a boundary layer of triangles and completing the mesh (Figure 3.5). The method used [Fuchs et al. 1977, Funnell 1984] relies on joining neighbouring nodes on the two boundaries to form triangles by optimizing a cost function. Any one of several cost functions can be used, including the area and the narrowness of the triangles.

Repeat steps 2–7 for all contours.

3.5 Conclusion

Although some other methods, not necessarily generating equilateral triangles, could have been used to the same effect, the most important result of the triangulation described in this chapter is the production of a large set of identical triangles inside the two-dimensional meshes of consecutive sections. This result will be used in the next step of the algorithm to generate the pentahedral prisms forming the core of the three-dimensional mesh. It should be noted here that this particular method of triangulating the cross-sections results in a different number of nodes being generated at each level, according to the size of the cross-section, eliminating the node cramping problem that could be encountered otherwise. Now that the triangular meshes inside the contours have been generated, they will be used to form tetrahedral elements in two consecutive steps. First, corresponding nodes on each layer will be joined to form a mesh of prisms and the prisms shredded into tetrahedra. The core that forms a large part of the bulk of the object is thus cheaply meshed, and a more expensive topological method will then be used on the

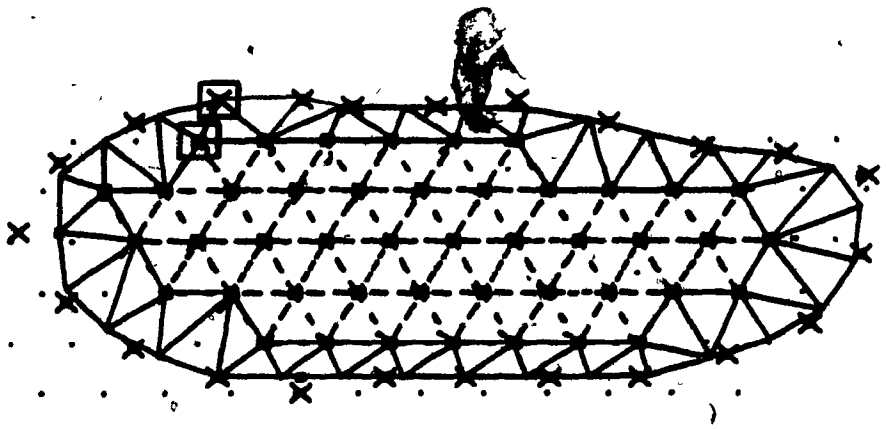

Figure 3.5

Completing the mesh.
The boundary of the internal mesh is joined to the boundary of the contour.

remaining portions.

CHAPTER 4

The 3-D mesh core

4.1 Introduction

Pure topological shredding techniques are very slow, and not worth using on a whole slice of the object. Instead, shredding by interpolation or using a look-up table will be used as much as possible Corresponding nodes on different levels will be joined in this step to form prism elements. Due to the use of the grid in the triangulation, these nodes can be joined directly by straight line segments without the need for interpolation functions. The resulting pentahedral mesh will be refined into a tetrahedral one by using a look-up table to divide each prism into three tetrahedra.

4.2 Building Prisms

Each grid node in the grid plane has been permanently assigned a pair of integer grid coordinates (I-J coordinates). These coordinates remain with the node and are stored in the node array of the two-dimensional triangulation of that particular contour. Nodes on two consecutive sections having the same grid coordinates correspond to the same position on the grid plane. They consequently lie exactly 'above' each other in the slice and can be joined by a vertical line segment. If three such pairs of corresponding nodes, belonging to a pair of corresponding triangles, are joined, they form a pentahedral prism. If all such corresponding triangles are

joined, a mesh of pentahedral elements can be constructed. Going through the list of triangular elements for the top and bottom two-dimensional contour triangulations, points forming pairs of corresponding nodes are flagged. Triangles whose three nodes are each part of a corresponding pair are then joined to form elements in the core mesh of pentahedral prisms.

4.3 Shredding a Prism

The set of prisms obtained constitutes the core of the final three-dimensional finite-element mesh. According to mesh rules, it is a topologically correct mesh and could be used in finite-element computations if prism elements are to be used. This not being the case however, it is clear then that this mesh would have to be refined to a tetrahedral one by shredding the prisms into tetrahedra.

This shredding has to be done while conserving the integrity of the mesh. It can easily be shown that, when no extra nodes are to be generated, a prism can be divided into three tetrahedra according to two different configurations. To show this, we first need to define the planar graph equivalent of a polyhedron. A planar graph is a set of co-planar nodes connected by a set of arcs in the plane, such that no two arcs intersect. Any three-dimensional polyhedron can be represented by a planar graph [Preiss 1981]. Planar graphs representing a prism and a tetrahedron are shown in Figure 4.1. Shredding a prism into tetrahedra can be reduced to finding an appropriate triangulation of its three quadrilateral sides (Figure 4.2). Three distinct triangulations T_1, T_2 and T_3 are possible, but it is easily seen that shreddings are possible only along T_1 and T_2. We shall denote the triangulated sides on T_1 and T_2 by

$$T_1 = (A\ B\ C)$$

$$T_2 = (D\ E\ F)$$

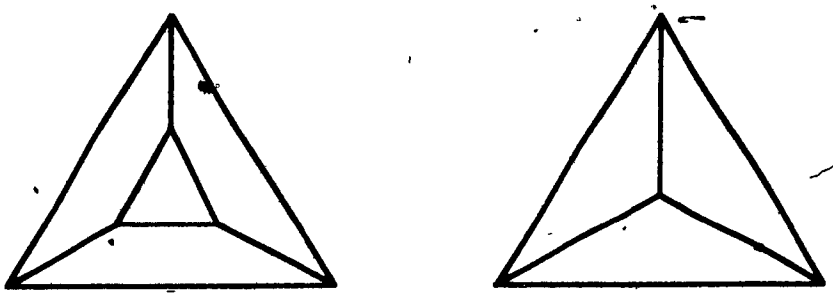

Figure 4.1

Planar graphs.
Planar graphs of a prism and a tetrahedron. The planar graphs are topologically equivalent to their corresponding polyhedra.

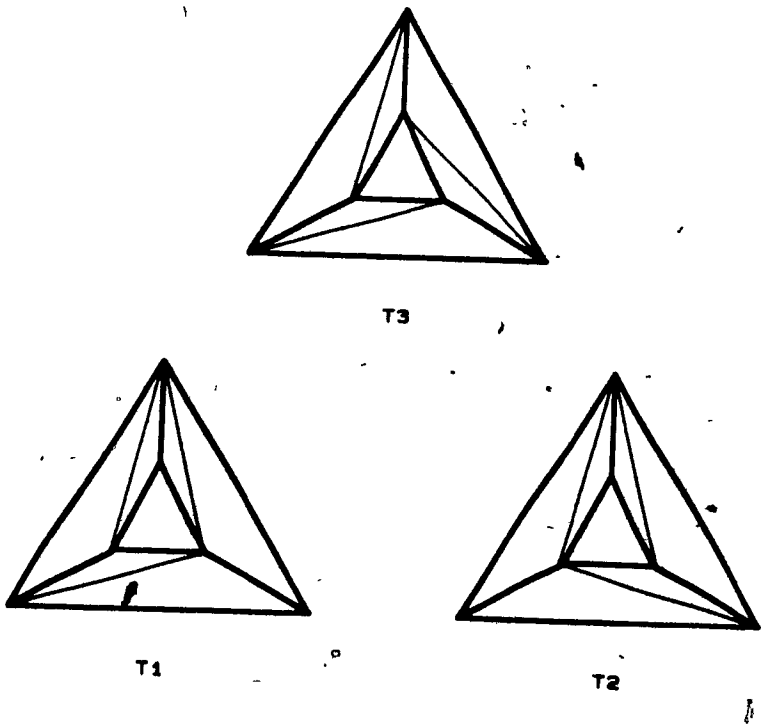

Figure 4.2

Possible triangulations of a prism.
The three possible triangulation configurations for a planar graph of a pentahedral prism.

as shown in Figure 4.2. The corresponding shreddings are shown in Figure 4.3.

4.4 Shredding a Mesh of Prisms

To have a sound mesh shredding, the tetrahedra created by shredding the prisms must meet the two mesh requirements specified in 1.2. Most importantly here, they can only intersect along a full face, a full edge or a vertex. To achieve this, each side of a triangulated prism has to be matched to the adjoining side on a neighbouring prism. Ideally, one would like to choose one shredding configuration, T_1 say, and convert all the prisms in the mesh accordingly. This, however, is not possible, as will be shown below.

Theorem 4.1 A mesh of pentahedral elements cannot be shredded into a mesh of tetrahedral elements using a single shredding configuration.

Proof To prove this theorem, we shall make use of a planar graph representation of four prisms in the mesh. Figure 4.4 shows a central prism P_0 surrounded by three others (P_1, P_2, and P_3), one at each of its three sides.

Assuming that P_0 is shredded according to T_1 (Figure 4.5.a), the adjoining sides on the three other prisms have to be matched symmetrically (Figure 4.5.b). By identifying each of these prism shreddings with T_1 (Figure 4.5.c), we can start assembling a rule:

Face A on P_0 interfaces with face C on P_3, leading to

1. Every face A matches a face C.

Face C on P_0 interfaces with Face A on P_2, leading to

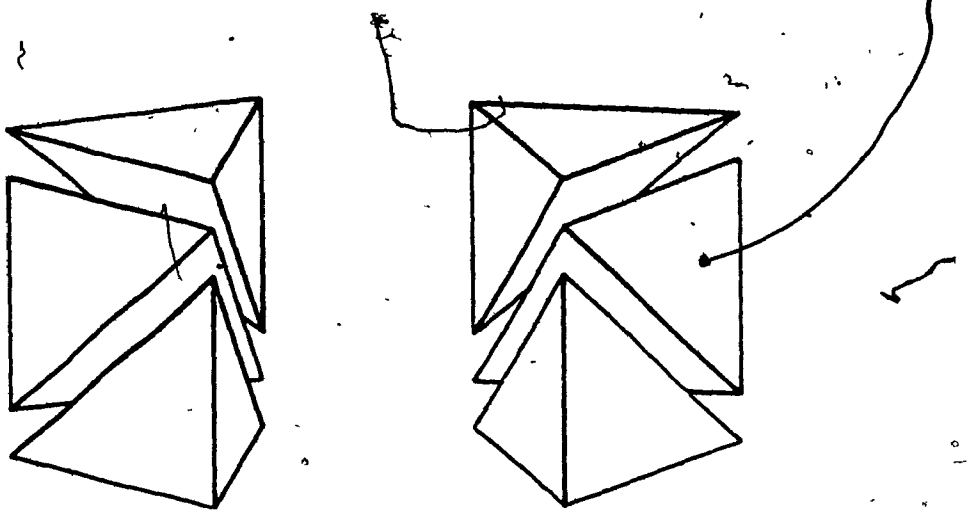

Figure 4.3

Possible shreddings of a prism.
The only two physically possible shreddings of a prism. All other shreddings are topologically equivalent.

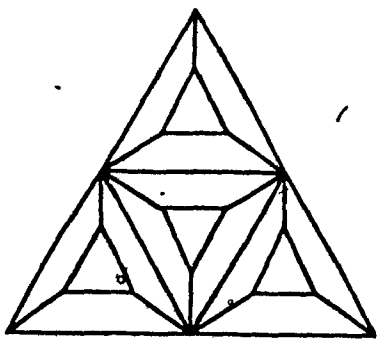

Figure 4.4

The mesh of four prisms.
There are four pentahedral prisms in this mesh, represented by their planar graphs.

2. Every face C matches a face A.

A consistent shredding would then require every face B to interface with a similar face B on another prism. This is however impossible here, since Face B on P_0 interfaces with Face A on P_1, violating the previous statements. Any other starting combination and matching combination can easily be shown to lead to a similar conflict, proving that no single shredding configuration is sufficient to shred the whole mesh.

Theorem 4.2 Two shredding configurations are necessary and sufficient to shred a regular pentahedral mesh into a tetrahedral one.

By regular pentahedral mesh it is meant a mesh whose triangular faces are equilateral and thus form a regular lattice of triangles at each cross-section level. By extension, this definition also applies to deformed regular meshes, thus including meshes where nodes that are not on any boundary curves have a connectivity of six on their cross-section level, i.e. each such node is connected to exactly six others on its level (Figure 4.6).

Proof This theorem will be proven in two parts. Again, the planar graph representation of four prisms in a mesh will be used.

First, assume that the central prism P_0 is shredded according to T_1. Similarly to the previous proof, each adjoining side is matched symmetrically. This time, by matching each prism shredding with T_2, a rule can be induced (Figure 4.7)

1. Every face A matches a face E.
2. Every face B matches a face F.

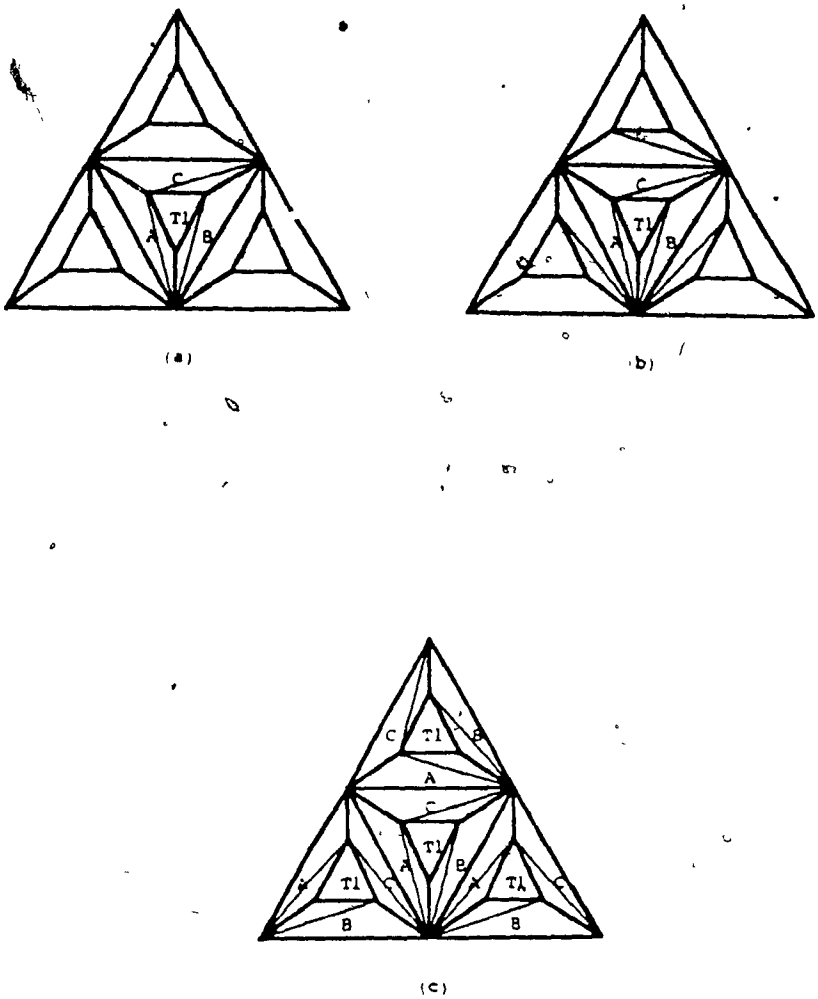

Figure 4.5

Possible shreddings of the mesh.
a) Shredding the central prism.
b) Matching the neighbouring sides
c) Completing the mesh

3. Every face C matches a face D

This shows that any T_1 configuration can also be surrounded by T_2 configurations, and, most importantly, all the T_2 configurations have the same orientation. Second, assuming that P_1 is shredded using T_2 and applying the same procedure, another rule can be induced (Figure 4.8)

1. Every face D matches a face C

2 Every face E matches a face A

3 Every face F matches a face B

showing that a T_2 configuration can also be surrounded by T_1 configurations having the same orientation Since the two rules are consistent with each other, it can be concluded that the whole mesh can be shredded by surrounding each T_1 configuration with three T_2 configurations, and vice-versa, in the checkered pattern shown in Figure 4 9 This proves the sufficient part of the theorem, and the necessary part follows from the previous theorem

4 5 The Look-up Table

Having determined that two configurations, used in a checkered pattern, will shred a regular mesh of prisms, the mesh can now be shredded accordingly The two configurations are coded into a look-up table which, for each one, outputs a total of 12 triangles, four for each of the three tetrahedra in the shredding. Given a prism, defined by its 6 nodes as shown in Figure 4 10, two different sets of tetrahedra can be produced:

$$\{(1\ 2\ 3\ 5),\ (1\ 3\ 4\ 5),\ (3\ 4\ 5\ 6)\}$$

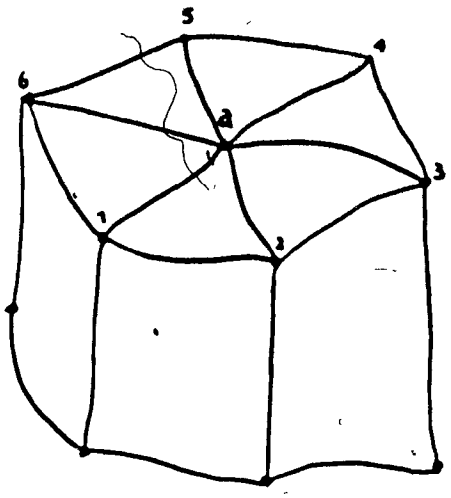

Figure 4.6

A regular pentahedral mesh.

An example of a regular pentahedral mesh All the element in the mesh are pentahedra, and each internal node is connected to six other nodes on the same plane

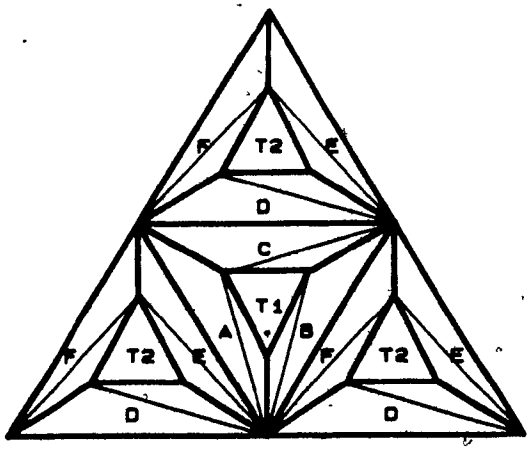

Figure 4.7

Possible shredding using two configurations.

This is one of the possible shreddings of the example mesh of four prisms, using the two different configurations.

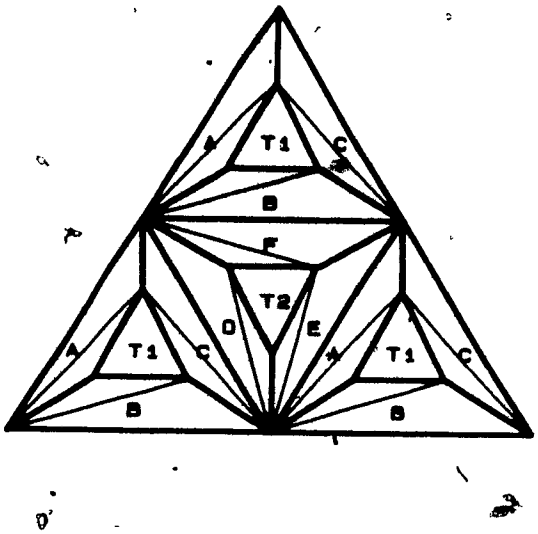

Figure 4.8

Reversing the configurations
Shredding of the same mesh by reversing the two shredding configurations

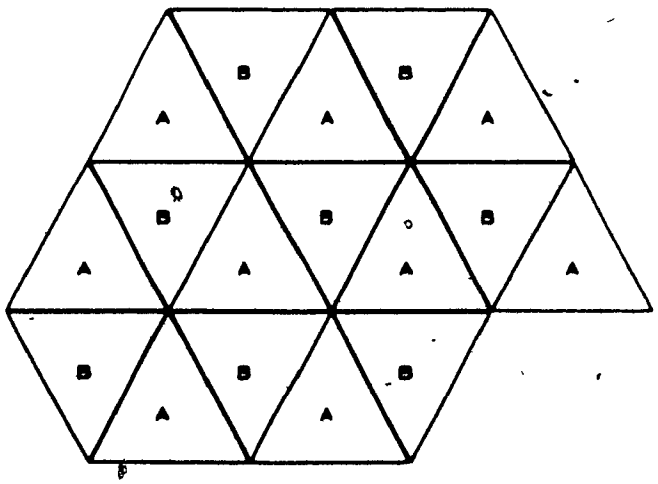

Figure 4.9

The checkered pattern.
The checkered pattern used to mesh a regular pentahedral mesh. The two shredding configurations are noted by A and B, respectively.

and

$$\{(1\ 2\ 3\ 6),\ (1\ 2\ 5\ 6),\ (1\ 4\ 5\ 6)\}$$

The appropriate set to use for each prism is determined from its position in the checkered pattern of the pentahedral mesh.

4.5 Conclusion

It has been shown that a regular pentahedral mesh can be shredded into a sound tetrahedral mesh. This can actually be done by assembling a look-up table of the two different configurations, and automatically generating the appropriate shredding for each prism depending on its position.

It is important to note here that only a regular pentahedral mesh can be shredded using two configurations. Other kinds of pentahedral meshes require more than two simultaneous configurations, and certainly more than the one erroneously assumed by Yamashita & Takahashi 1984], whose method produces inconsistent meshes.

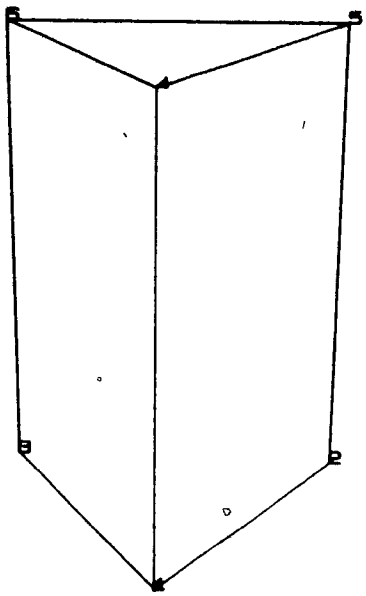

Figure 4.10

Prism node numbering.

The prism node numbering employed in the look-up table.

CHAPTER 5

Assembling The Ring

5.1 Introduction

By removing the core mesh from the slice, a hollow, torus-like structure is usually left (Figure 5.1). By convention, we shall refer to this structure as the 'ring', although other shapes might also occur. For example, if no triangles have been matched on the two surfaces defining a slice, no prisms will be shredded and the 'ring' will consist of the whole slice (Figure 5.2).

For the finite-element mesh to be complete, this structure has to be meshed too. Furthermore, at its inner side, where it is connected to the core mesh, both meshes have to interface exactly at every node, edge and face. Before any further work can be done however, a topological and geometrical characterization of the structure is needed. The information available has to be preprocessed and entered into a coherent data structure that will be acted on by the shredder.

5.2 The Data Structure

The data structure used here defines three data entities in the form of three lists.

1. The VERTEX list contains the X, Y, and Z coordinates of each vertex, along with its connectivity (number of edges meeting at a vertex) and pointers into

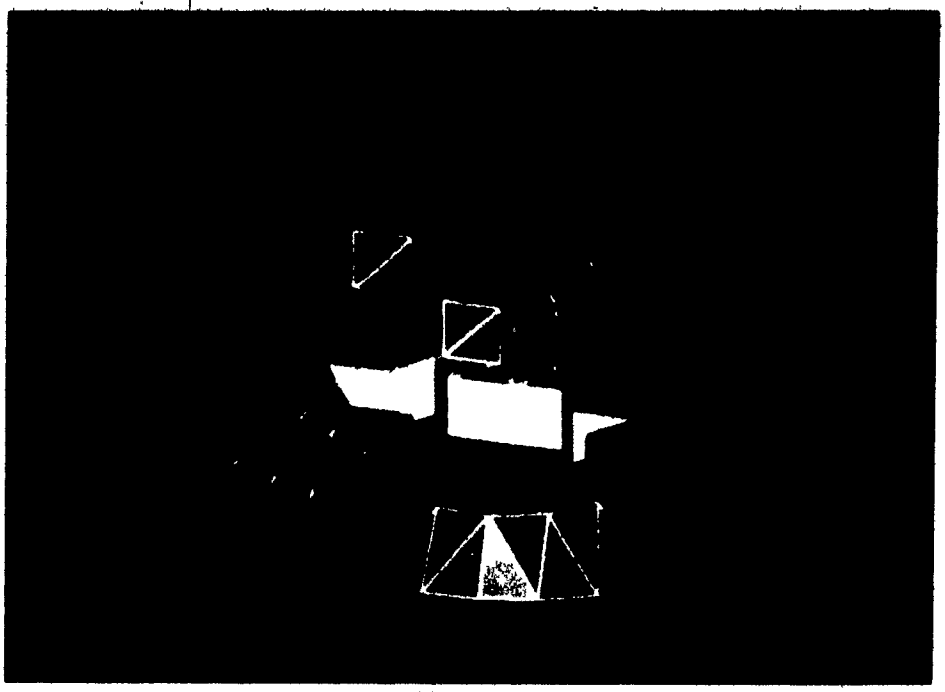

Figure 5.1
The remaining torus, after the core mesh has been removed

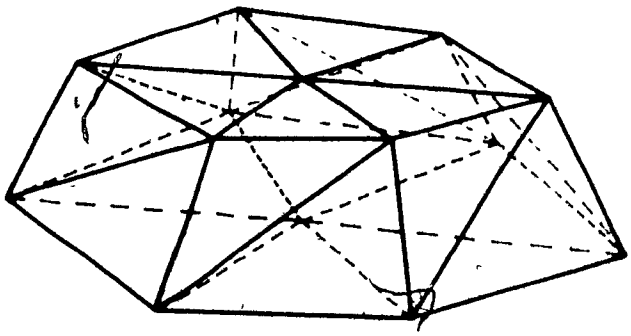

Figure 5.2
When no core mesh can be removed, the 'ring' consists of the whole slice

the EDGE list to those edges meeting at the vertex.

2. Each entry in the EDGE list consists of a pair of pointers pointing into the VERTEX list to the two vertices that bound that edge.

3. The FACE list stores faces as triplets pointing into the VERTEX list to the three nodes of each face.

In addition, a few functions are used by the program to perform data retrieval operations on the data structure. The function *E_FACES* returns two pointers to the two faces in the FACE list that have the given edge in common. The function *V_EDGE* finds an edge in the EDGE list, given its two endpoints. The function *V_FACE* finds a face in the FACE list, given its three vertices. Other functions, *V_DEL*, *E_ADD*, *E_DEL*, *F_ADD*, and *F_DEL* perform insertions/deletions of vertices, edges and faces in the structure, while maintaining the overall consistency and updating all the appropriate parameters.

5.3 Creating the Ring

Having defined the data structure, the four parts of the ring, namely the top, bottom, internal and external surfaces, are individually generated and merged into the structure. First the inner side, which is the interface of the ring with the core mesh (Figure 5.3), is generated in the following manner.

The surface of the whole slice consists of the top, bottom and external surfaces, and is denoted by

$$S^2 = T_3^2 - B_3^2 + E_3^2$$

(Refer to the Appendix for the notation used). Similarly, the core mesh surface

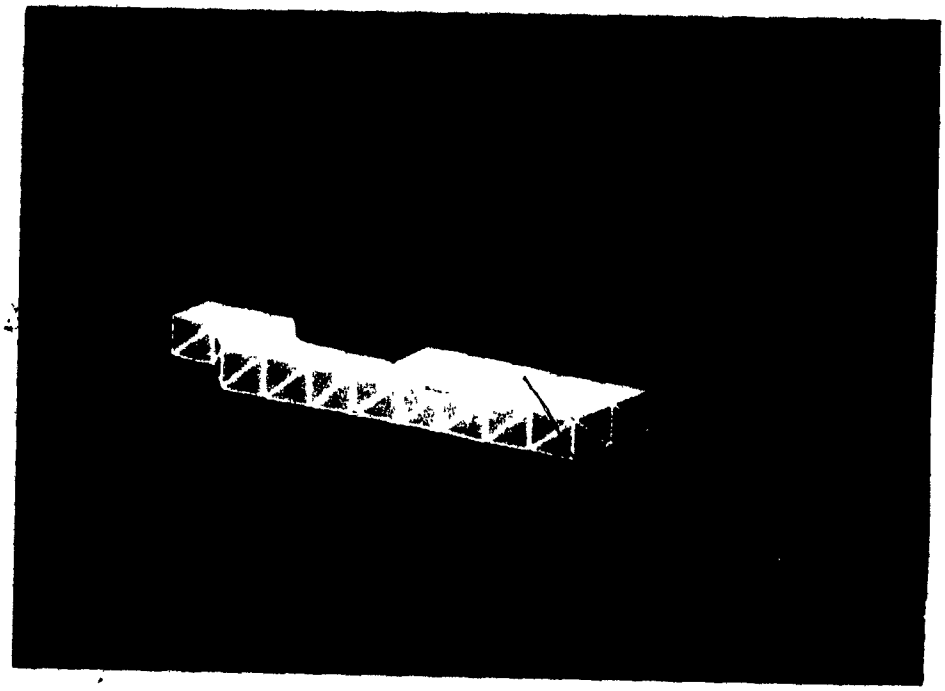

Figure 5.3

The inner surface of the ring.

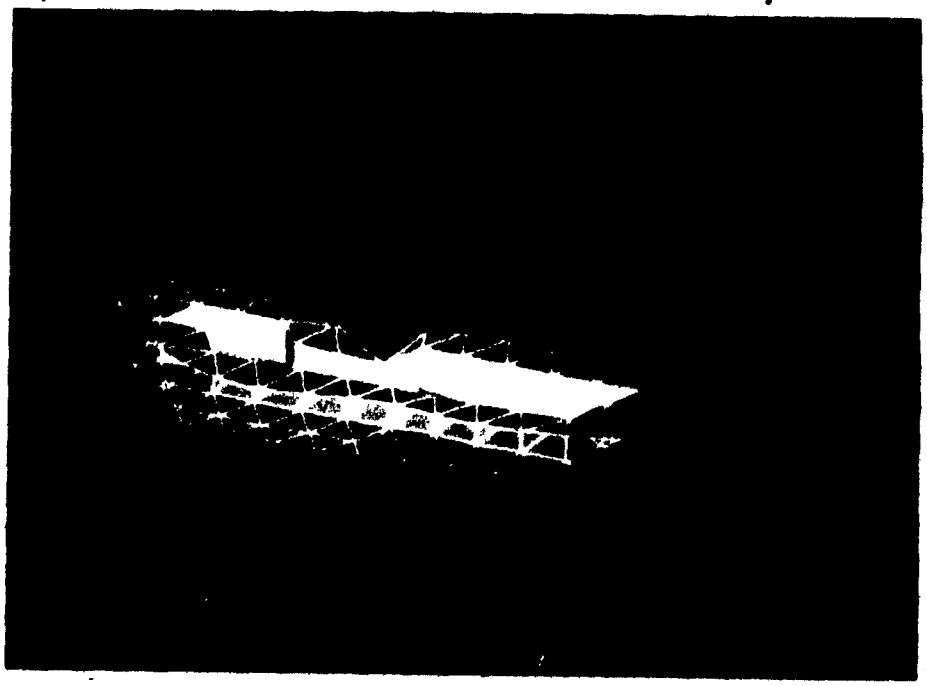

Figure 5.4

The top and bottom surfaces of the ring.

consists of

$$C_c^2 = T_c^2 + B_c^2 + E_c^2$$

The ring, on the other hand, includes an extra surface portion at its interface with the core, namely the internal surface, so that

$$R^2 = T_r^2 + B_r^2 + E_r^2 + I_r^2$$

Since the slice is composed of the union of the ring and the core, the equations

$$\begin{aligned} S^2 &= C^2 - R^2 \\ &= T_c^2 - B_c^2 - E_c^2 + T_r^2 + B_r^2 + E_r^2 + I_r^2 \\ &= T_s^2 - B_s^2 - E_s^2 p \end{aligned}$$

must hold. We know, from the way those surfaces were generated, that the union of the core top and the ring top make the slice top surface, the union of the core bottom and the ring bottom make the slice bottom surface and the external ring side surface is the same as the slice external side. This is translated into

$$T_s^2 = T_c^2 - T_r^2$$

$$B_s^2 = B_c^2 + B_r^2$$

$$E_s^2 = E_r^2$$

resulting in

$$E_c^2 + I_r^2 = 0$$

$$I_r^2 = -E_c^2$$

meaning that I_r^2 has the same faces as E_c^2, but wiith the opposite orientation. To compute this, the triangular faces of the tetrahedra forming the core mesh are sorted and all triangles occurring twice are removed, leaving the surface triangles. Of these, the external side triangles of the core are the ones that do not lie in the horizontal top or bottom planes. Reversing the node numbering of these triangles will yield the triangles on the internal side of the ring.

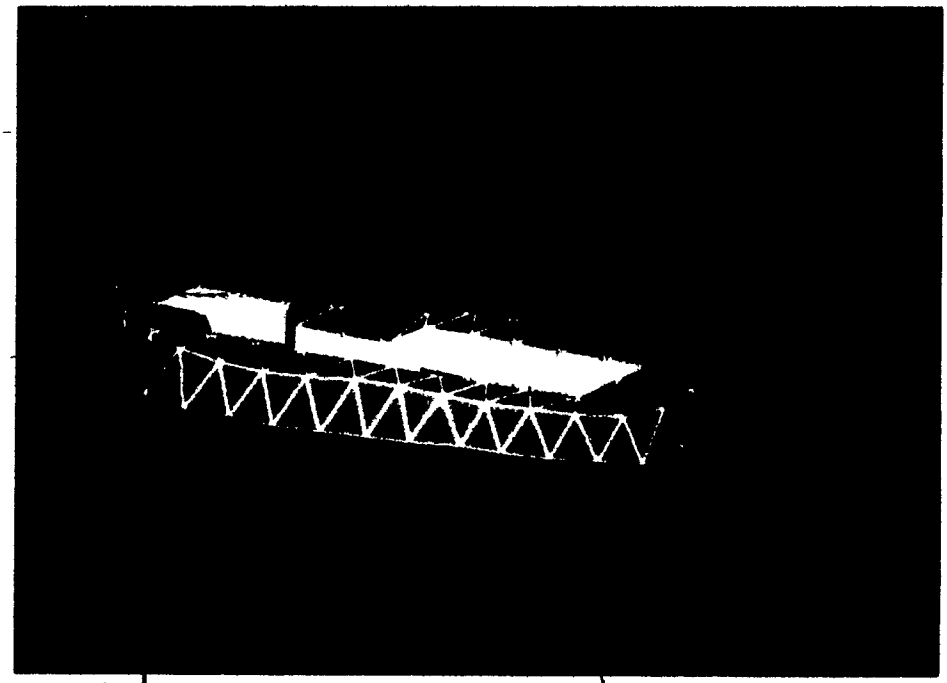

Figure 5.5
The completed surface of the ring.

The top surface is then generated from the two-dimensional triangulation of the top contour by keeping the triangles that were not used in the prism assembly of the core mesh. This involves scanning through the triangle list of the two-dimensional top contour triangulation and copying the unflagged triangles, i.e. triangles that were not part of a pair of corresponding triangles (section 4.2). The counterclockwise numbering of the triangle nodes is kept, so that the normal vectors point upwards, out of the ring.

The bottom surface is similarly generated from the bottom contour two-dimensional triangulation, except that the node numbering of the triangles is flipped to make the normals point down and outwards, keeping with the consistent orientation of the structure (Figure 5.4).

Finally, the external side surface of the ring (Figure 5.5) is generated by triangulating between boundary nodes on the top and bottom contour nodes. The method is the same one used for joining the inner and outer boundaries in the two-dimensional mesh (section 3.4) and consists of joining neighbouring nodes to optimize an area or a volume cost function [Fuchs et al. 1977, Funnell 1984].

5.4 Topological Checks

After the ring has been generated, a topological consistency check is performed on its structure. Since the ring is a three-dimensional manifold, its surface should have no boundary curves, i.e. $\dot{R}^2 = 0$. This is easily checked by counting the number of times each edge occurs in the list of triangles, and its orientation each time. For a consistent topology, each edge should only occur twice (once in two different triangles) in the whole structure, and in opposite directions, reducing the sum $\dot{R}^2 = \sum \dot{x}_i^2$ to zero. As shown in section A.5, this check also makes sure

that the orientation of all the faces is the same. This orientation is the correct one if the vectors normal to the faces point outwards. It can be verified, for example, if the normal vector to one of the faces on the bottom side points downwards in the negative Z direction. The Euler number of the structure is then calculated, and used to keep track of the structure as it is being meshed.

5.5 Conclusion

So far, the center of the slice has been processed to produce a core mesh of tetrahedra and the remaining portion assembled into a consistent data structure using results from previous steps. This 'ring' was formed in four parts. The top and bottom parts were obtained from the two-dimensional triangulation of the contours, the internal side computed from the tetrahedral core mesh and the external side by triangulating between the nodes on the the two consecutive contours defining the slice. The data structure, consisting of a Vertex list, an Edge list, a Face list and a vertex connectivity array is now ready to be operated on by the topological shredder using the Euler operators to be introduced in the next chapter.

CHAPTER 6

Meshing The Ring

6.1 Introduction

In this chapter the basic shredding operations will be developed. They consist of a set of operators that are applied to a polyhedron in a certain sequence, reducing it to a final tetrahedron and compiling a tetrahedral mesh along the way. These operators make use of the Euler characteristic equation for polyhedra and its differential form, and are therefore called Eulerian operators. It will be shown that the previously published methods are inadequate for a large set of polyhedra and additional operators will be presented. Examples of completely irreducible polyhedra will also be given, and the constraints they impose will be discussed.

6.2 The Eulerian Operators

Meshing a polyhedron P can be thought of as a sequence of operations to remove tetrahedra from the initial structure in order to reduce it to an empty set. Denoting the set of operations by

$$O = \{O_i,\ i = 1, 2, 3....N\}$$

where O_i is the ith operator and N the total number of operations, then the operation is

$$O(P) = T$$

where

$$T = \{T_i;\ i = 1,2,3...N\}$$

is the set of generated tetrahedra. The side effect of this reduction scheme is the compilation of the set of tetrahedra T_i that have been removed from the structure. The union of these tetrahedra constitutes a tetrahedral mesh spanning all of P. The progress of the meshing procedure is shown below, where P is the polyhedron representation of the ring, S_1 is the initial data structure and S_i the data structure after $(i-1)$ operations

$$S_1 \leftarrow P$$

$$O_1(S_1) \rightarrow T_1 + S_2$$

$$O_i(S_i) \rightarrow T_i + S_{i+1}$$

$$O_{N-1}(S_{N-1}) \rightarrow T_{N-1} + S_N$$

$$S_N \rightarrow T_N + \emptyset$$

The operations are carried out by a set of operators that work on the topology of the structure. By cutting out a tetrahedron, operators might change the number of vertices, edges or faces of the structure. Since the initial polyhedron satisfies Euler's equation

$$V - E + F = 2 - 2G$$

any changes brought on by the operators have to satisfy the differential form of that

formula, namely

$$\Delta V - \Delta E + \Delta F = -2\Delta G$$

or

$$\Delta G = -\tfrac{1}{2}(\Delta V - \Delta E + \Delta F)$$

In the next sections a thorough description of these Eulerian operators will be given.

6.3 The Vertex Operator (*V_OP*)

An accepted method for the triangulation of two-dimensional polygons is the corner-cutting technique. It consists of locating a corner at a convex vertex of the polygon, and cutting off the triangle formed there (Figure 6.1). By analogy, the Vertex operator Woo & Thomasma 1984, Wördenweber 1984 acts on a trivalent (belonging to only three edges), convex (the summit of a convex three-dimensional corner) vertex v_i. The tetrahedron having for vertices v_i and the three other attached vertices is removed from the structure in one cut (Figure 6.2). In the process, one vertex, three edges and three faces are deleted, and one face added. From the equation below we can see that the genus of the object is left unchanged.

$$\Delta G = -\tfrac{1}{2}\big(-1 - (-3) + (1 - 3)\big) = 0$$

This cut cannot be performed without some preliminary intersection checks to ensure the geometric consistency of the cutting procedure.

1. The four vertices cannot lie in the same plane. Violations of this rule produce degenerate, or flat, tetrahedral element that are useless in finite-elements computations. In addition, producing flat tetrahedra with this operator can place the algorithm in an infinite loop, by piling flat tetrahedra onto each other, without changing the volume of the object.

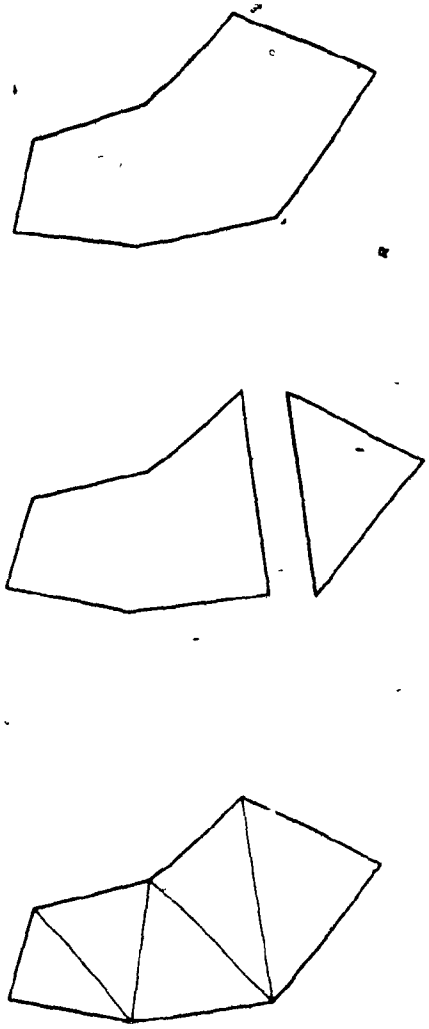

Figure 6.1

2-D corner-cutting

a) The polygon to be triangulated.
b) Locating a corner and cutting it off
c) The completed mesh

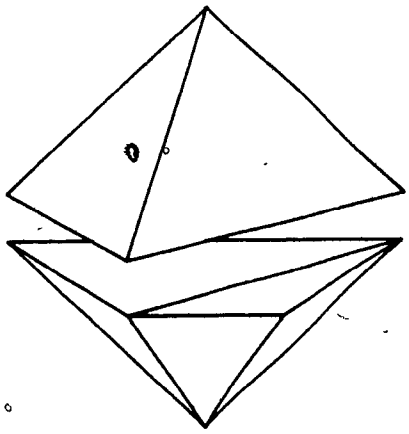

Figure 6.2

The vertex operator.
The vertex operator locates a trivalent convex vertex and removes the tetrahedron attached to it in one cut

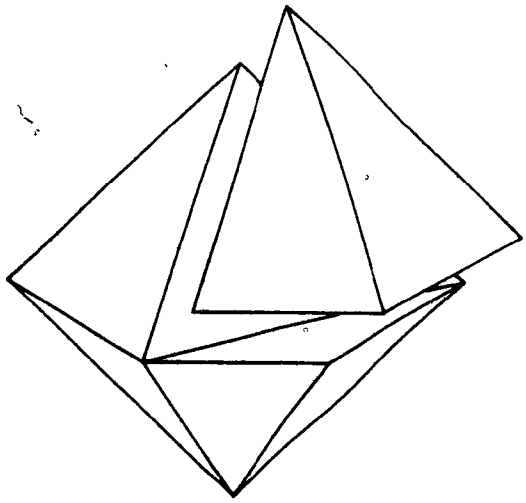

Figure 6.3

The edge operator.
The edge operator locates a convex edge and removes the tetrahedron attached to it in two cuts.

2. The tetrahedron to be removed has to be empty, i.e. no vertices or edges belonging to the remaining structure can fall inside its volume. This rule ensures that the remaining structure is not self-intersecting.

3. The newly created face cannot intersect any edge or face of the remaining structure.

6.4 The Edge Operator (*E_OP*)

The corner-cutting technique alone, implemented by the vertex operator, does not always succeed in completely meshing a three-dimensional polyhedron. An alternate operator is needed when it fails to find an appropriate vertex. The Edge operator (Woo & Thomasma 1984, Wördenweber 1984) acts on a convex edge. The two faces with the common edge have four vertices altogether. In two cuts, the operator cuts off from the structure the tetrahedron formed by the four vertices (Figure 6.3). In the process, one edge and two faces are deleted, and one edge and two faces added. Again, we see that the genus of the object is unchanged

$$\Delta G = -\tfrac{1}{2}(0 - (1 - 1) + (2 - 2)) = 0$$

Here too, some preliminary intersection checks have to be carried out before the operation is allowed.

1. Conditions 1, 2 and 3 from the previous section (non-planar vertices, empty tetrahedron and face intersections) still apply.

2. In addition, the newly created edge cannot intersect any existing face.

6.5 The Topological Cut Operator (*C_OP*)

The two previous operators do not change the genus of an object. This

should not pose any problems if the object being shredded is simply connected ($G = 0$), because successive applications of the operators could, without violating the differential form of Euler's equation, resolve it down to the final tetrahedron T_N. However, for multiply connected domains, like a torus for example, this differential equation implies that a structure with a positive genus cannot be reduced to a tetrahedron using the Vertex and Edge operators. A new operator has to be used to perform the topological cuts needed to transform multiply connected domains into simply connected ones by reducing their genus. This operator is used whenever *V_OP* and *E_OP* are unable to cut a tetrahedron off the structure. When this point is reached, at least one triangular cross-section through the domain exists [Woo & Thomasma 1984] The Cut operator locates such a cross section and makes a cut by creating two faces at the cross-section site. In the process, three vertices, which are duplicates of the three vertices at the cross-section, three edges, duplicates of the three edges at the cross-section, and two faces are added to the data structure (Figure 6.4) From the equation

$$\Delta G = -\tfrac{1}{2}(3 - 3 + 2) = -1$$

it is seen that the genus is effectively decreased by one. *V_OP* and *E_OP* can then resume the shredding.

6.6 Irreducible and Seemingly Irreducible Polyhedra

A set of Eulerian operators similar to the one presented above was claimed to be complete by [Woo & Thomasma 1984] and a proof of the correctness of the topological shredding of polyhedra using these operators was given. This proof is however flawed in that it assumes that any polyhedron can be constructed from *solid* tetrahedra, which is not the case. The most simple counterexample to this assumption is the triangulated prism shown in Figure 6.5. This polyhedron cannot

be shredded into solid tetrahedra, and consequently cannot be constructed from solid tetrahedra. We shall come back to this example later. In a more general way, there are three classes of polyhedra that block the progress of *V_OP* and *E_OP*.

6.6.1 Polyhedra of the First Class

Polyhedra of the first class, such as the one shown in Figure 6.6.a, are not a major stumbling block. They can usually be processed by dividing them into two separate polyhedra using *C_OP* (Figure 6.6.b), reducing the connectivity of three of the vertices. Since a cut through a simply connected domain divides it into two separate sub-domains, it is important to keep track of the number of separate polyhedra being generated by *C_OP*, as this will affect the Euler number of the global structure. To see this, assume N separate polyhedra have already been generated. The Euler formula for one of them is

$$V_i - E_i + F_i = 2$$

and the total number is

$$\begin{aligned} V - E + F &= \sum V_i - \sum E_i + \sum F_i \\ &= \sum (V_i - E_i + F_i) = 2N \end{aligned}$$

As a solution, a pointer to a vertex in each separate polyhedron is kept and updated to another vertex if that vertex is deleted, until the whole polyhedron has been shredded. This procedure can be repeated until one of three conditions becomes true: 1) *V_OP* or *E_OP* can be applied, 2) all the polyhedra have been reduced to separate tetrahedra or 3) all the polyhedra have been reduced to polyhedra of the second class.

6.6.2 Polyhedra of the Second Class

This second class of polyhedra, a simple example of which is the prism men-

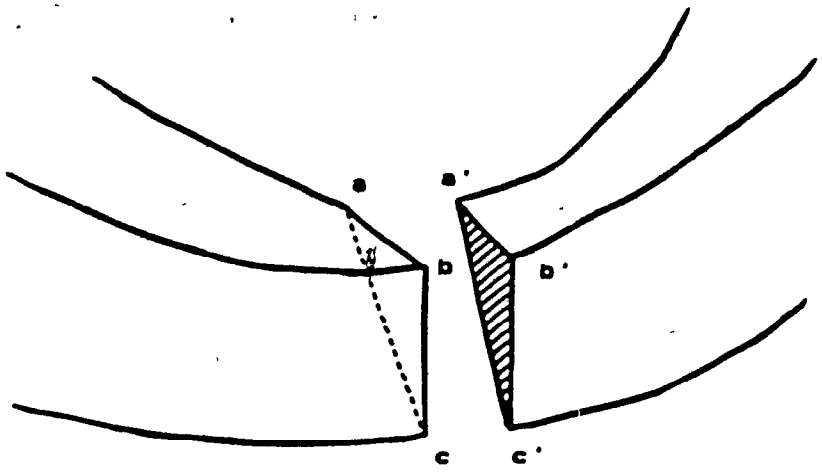

Figure 6.4

The Topological Cut

The topological cut operator performs a cut through a triangular cross-section, reducing the genus of the object

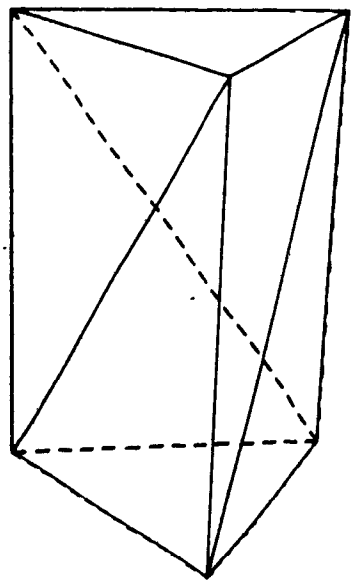

Figure 6.5

An irreducible triangulated prism.

This is one of the possible triangulations of the faces of a prism. The resulting polyhedron cannot be divided into solid tetrahedra

tioned previously (Figure 6.5), is more of a problem. Polyhedra in this class can usually be transformed by flipping one of their diagonals. A diagonal is defined as an edge common to two triangles, forming the diagonal of a planar quadrilateral. Two approaches are used to do this, depending on the position of the diagonal. During the sequence of operations O_i, such a polyhedron can either occur at the edge of the ring or completely inside. Consider the first case where the polyhedron is at the edge of the slice. Flipping a diagonal effectively alters the external surface triangulation of the slice. This, however, is acceptable, since the triangulation, which was optimized with respect to a cost function, is thus transformed into a sub-optimal, but still valid, triangulation.

If, on the other hand, the polyhedron lies completely inside the slice, another approach is taken. The fact that the polyhedron is inside the slice means that it interfaces with other tetrahedra of the mesh at all its vertices, edges and faces. Flipping a diagonal would violate the mesh rules by creating unwanted intersections between neighbouring elements at the site of the flipped diagonal. This is remedied by creating an Interface Tetrahedron as an interface between the polyhedron in question and the rest of the meshed structure (Figure 6.7) The fact that this interface tetrahedron is flat, however, will be taken care of in the next section by relaxing the nodes to improve the quality of the mesh, 'unflattening' flat tetrahedra along the way. The operator that generates the interface tetrahedron is very similar to *E_OP* in its actions. It acts on an edge whose two connected faces are coplanar by forming a flat tetrahedron out of the four connected vertices, but is used only when all other operators have failed, indicating the presence of one or more Class 2 polyhedra. The same alterations are produced, namely one edge and two faces are deleted, and one edge and two faces are added. None of the intersection checks are performed, being irrelevant in this case. The effect of this operator is in fact to flip

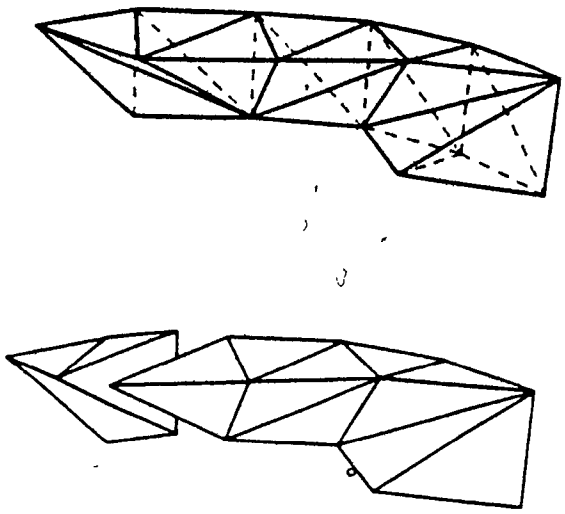

Figure 6.6

a) A polyhedron of the first class

b) A topological cut is performed on it dividing it into two separate polyhedra

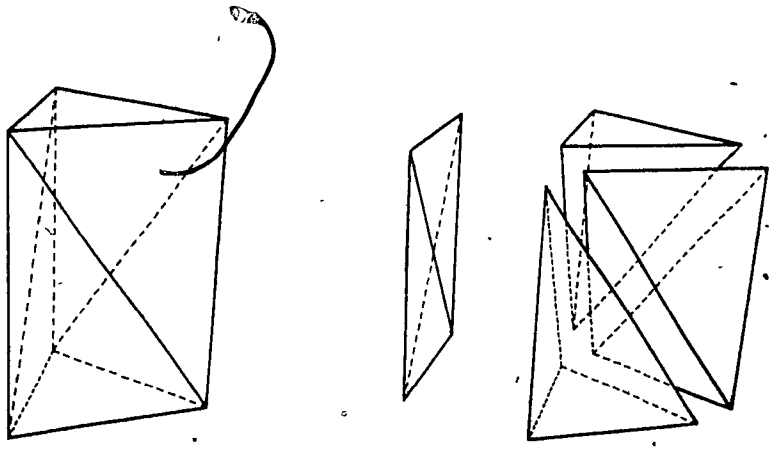

Figure 6.7

The Interface tetrahedron.

By using the flat Interface Tetrahedron, the irreducible polyhedron on the left is converted to a reducible one, then shredded.

a diagonal on S_t, making it more susceptible to shredding, while keeping the mesh consistency intact.

6.6.3 Polyhedra of the Third Class (Monsters)

Finally, the third class represents truly irreducible polyhedra. These polyhedra cannot at all be divided into tetrahedra without creating additional nodes. The existence of irreducible polyhedra was first shown by [Lennes 1911] and [Schönhardt 1928] proved that all polyhedra having five or less vertices are decomposable, i.e. that an irreducible polyhedron must have at least six vertices. The polyhedron shown in Figure 6.8 is due to Bagemihl 1948], and the one in Figure 6.9 was found during the course of this research.

If, while meshing a slice, such a polyhedron is encountered, the mesh generation procedure fails. Nonetheless, this conclusion is not as final as it sounds. It has been found that changing the starting point of the meshing will often result in a successful meshing. This has been implemented by storing the Vertex, Edge and Face lists in circular forward- and backward-chaining rings. If a Class 3 shape is encountered during the meshing, the procedure is aborted, the original slice data structure restored and the pointer to the beginning of the Edge list is advanced by one position. The procedure is then restarted on the new data structure.

6.7 Conclusion

In this chapter, the topological shredding procedure was characterized as a set of operations on a data structure, and the three main operators, the vertex, edge and cut operators, were presented, the first two of which are modifications of previously published ones. It was then shown that these operators do not form a complete set, in the sense that they are not sufficient to reduce any polyhedron to a

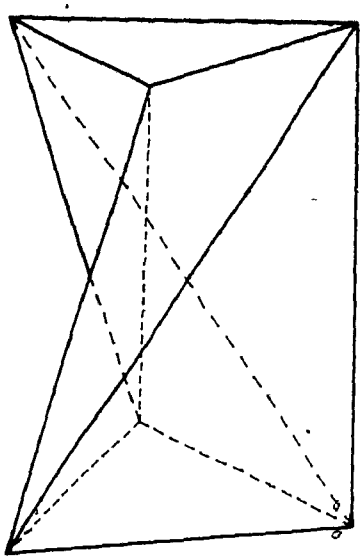

Figure 6.8

An irreducible polyhedron.
This polyhedron, due to Schönhardt, cannot be divided into tetrahedra.

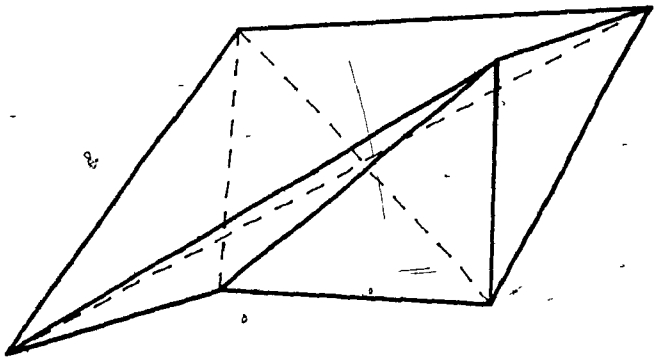

Figure 6.9

Another irreducible polyhedron.

set of tetrahedra. Three classes of seemingly irreducible polyhedra were introduced. A way to convert two of these classes into reducible polyhedra was shown. The case of the third and completely irreducible class was solved by restoring the whole data structure for the slice and restarting the mesh generation procedure at a different point in the structure.

CHAPTER 7

The Quality of the Mesh

7.1 Introduction

Tetrahedra produced by the mesh generation can have a variety of shapes, including long and narrow tetrahedra, and the flat interface tetrahedra. Narrow tetrahedra can be acceptable, but tend to be avoided in the finite-element method because they can reduce the accuracy of the solutions [Barnhill & Whiteman 1973, Ciarlet 1973, Hermeline 1982, Strang & Fix 1973]. Flat tetrahedra, on the other hand, are absolutely unacceptable. It is important then to be able to detect badly shaped tetrahedra and correct them to improve the quality of the finite-element mesh. In this chapter, a way to characterize the overall quality of a mesh, and a method used to improve the shapes of its elements, will be described.

7.2 The Quality of an Element

A mesh generation scheme can produce a mesh that is topologically correct, but of little value for finite-element computations because some of its elements are badly shaped. In an ideal mesh, on the other hand, all the elements would be equilateral, since the best computation results are obtained with equilateral elements [Hermeline 1982]. It is therefore preferable that the tetrahedra in a mesh be as equilateral as possible. This state is usually very hard to achieve in real situations, unless the object itself is highly regular and lends itself to such a mesh.

In general, the usefulness of a mesh is a function of the quality of its individual elements, and particularly that of its worst elements. For this reason, the quality of a mesh has to be assessed, through a computed criterion, before the mesh can be used.

The quality of an individual element can be measured by computing what is called the aspect ratio of the element. The aspect ratio, a number defined to be 0.0 for a degenerate element (e.g. a flat tetrahedron) and 1.0 for an equilateral element, can be computed in a number of different ways. In two-dimensional triangular meshes, a widely used measure of the quality of a triangular element is the value of its smallest angle, since the bound on the error is inversely proportional to the sine of the smallest angle in the mesh [Strang & Fix 1973, Bramble & Zlámal 1971, Zlámal 1973]. To obtain a measure of an element's aspect ratio, the value of the smallest angle in that triangle is normalized by dividing it by 60°, since 60° is the value of an angle in an equilateral triangle. In three dimensions, a similar measure, proportional to the narrowest solid angle in a tetrahedron, is used here [Nguyen-Van-Phai 1982, Van Oosterom & Strackee 1983].

Consider the unit sphere centered at one of the nodes of a tetrahedron. Lines along the three edges emanating from that node intersect the sphere in three points. The three great circles passing through those three points define a spherical triangle on the surface of the sphere, with internal angles A, B, and C. In spherical geometry [Greenberg 1974], it is known that

$$A + B + C \geq \pi$$

and the difference

$$\delta = A + B + C - \pi$$

is called the spherical excess of the triangle. The excess increases with the

curvature of the triangle, so that a wide solid angle at the center of the sphere defines a triangle with a large spherical excess. Vice-versa, a narrow angle defines a flatter triangle with a smaller spherical excess. Computing the excess for the triangles defined by all four nodes of a tetrahedron and taking the smallest value gives a good measure of the quality of the tetrahedron, since it measures the narrowest solid angle in the tetrahedron. Normalizing this value to the ideal spherical excess, defined by a node of an equilateral tetrahedron ($\delta_e = 0.55128rad$), will yield a measure of the aspect ratio of the tetrahedron in question, giving a number between zero and one, with an equilateral tetrahedron having an aspect ratio of one and a flat tetrahedron an aspect ratio of zero. As an example, Figure 7.1 shows a plot of the aspect ratio variation as a function of the height of a node in a tetrahedron In this example the node is started at the center of gravity of the base equilateral triangle, making a flat tetrahedron, and raised vertically by small increments. As seen on the plot, the aspect ratio function starts at zero for the flat tetrahedron, attains its maximum of one at the point where the tetrahedron is equilateral and decays back to zero as the tetrahedron gets narrower Figure 7.2 shows different tetrahedra and their corresponding aspect ratios

7.3 The Quality of the Mesh

When taken one step further, the method just described provides a way to look at the quality of the whole tetrahedral mesh. Once the individual meshes for all the slices are available, they are combined to form the global mesh. The data structure is then 'cleaned' by removing duplicate node entries and rearranging the pointer arrays correspondingly. The resulting data structure corresponds to the desired mesh that models the object, and consists of a list of tetrahedra pointing to a list of triangular faces that in turn point to a list of vertices.

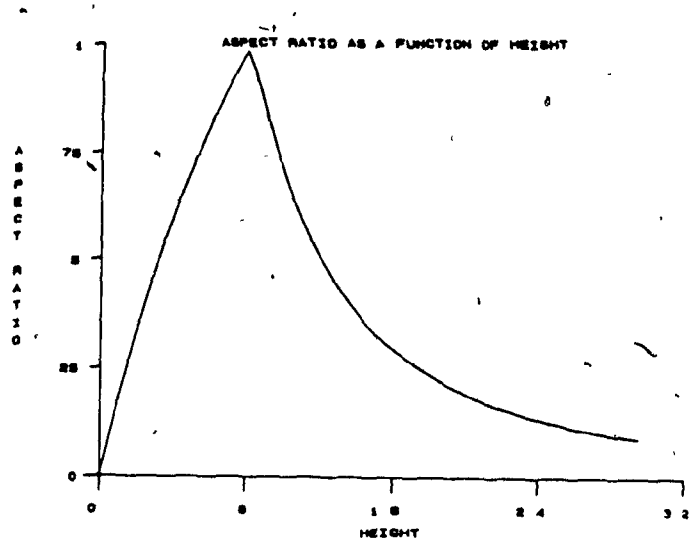

Figure 7.1

The aspect ratio variation
The aspect ratio variation as a function of the height of a tetrahedron node from the base equilateral triangle

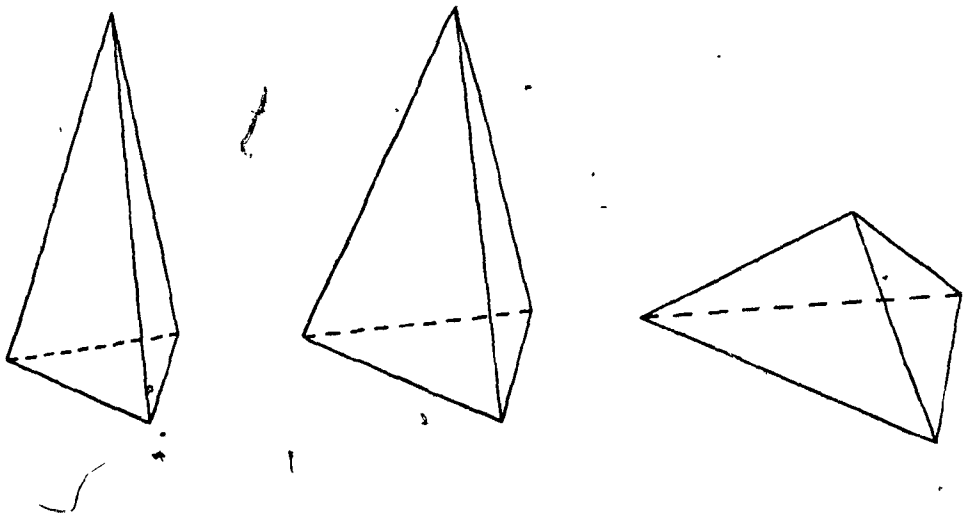

Figure 7.2

Some different aspect ratios.
Tetrahedra having aspect ratios of 0.2, 0.4 and 0.8, respectively.

An indication of the quality of this mesh can then be implemented by computing the aspect ratio for each tetrahedron in the mesh and constructing a histogram of the distribution of aspect ratios in the whole mesh. Such a histogram is shown in Figure 7.3. As an additional indication of mesh quality, the aspect ratio of the worst element in the mesh is determined and the sum of the aspect ratios of all the elements in the mesh is computed.

Although zero-volume elements are definitely unacceptable, there is no well-defined cutoff threshold for an element to be unacceptable, since the accuracy of the solution decreases gradually with the aspect ratios of the mesh elements. It is clear however that a mesh like the one whose aspect ratio histogram is shown in Figure 7.3 needs improving, since it contains four zero-volume elements and several low-aspect-ratio elements In two-dimensional meshes, improving a mesh can be achieved by relaxing its nodes to an equilibrium position where each node is positioned at the geometric center of all the nodes connected to it [Thacker et al. 1980] The same relaxation method is followed here to improve the overall shape of the mesh elements and achieve a better distribution.

7.4 The Relaxation

Every internal node in the global mesh is assumed to be acted upon by forces along the edges connecting it to other nodes around it, in analogy to a set of springs of similar stiffness and negligible initial length, now under tension. These forces tend to displace the node towards an equilibrium position at the center of the other nodes connected to it. This is equivalent to solving a set of simultaneous

equations [Thacker et al. 1980] for the new internal node positions

$$x_k = \frac{1}{N_k} \sum_{i=1}^{N_k} x_{n_i(k)}$$

$$y_k = \frac{1}{N_k} \sum_{i=1}^{N_k} y_{n_i(k)}$$

$$z_k = \frac{1}{N_k} \sum_{i=1}^{N_k} z_{n_i(k)}$$

where $n_i(k)$ are the indices of the N_k points connected to the point k being displaced. The system is solved iteratively using the Gauss-Seidel method by displacing each of the nodes until the displacements converge and all the internal nodes are more or less at the center of their attached nodes. It is important to note here that, unlike [Thacker et al. 1980], the external nodes, i.e. the nodes lying on the surface of the object, are fixed. In this way, they provide a rigid frame, holding the shape of the object and preventing its collapse under the spring forces. This might cause a few problems, as will be seen later.

An example of two-dimensional relaxation of a triangular mesh is shown in Figure 7.4. As seen on the figure, some elements have lost their equilateral property but the overall quality of the mesh has improved.

The relaxation procedure may sound straightforward, but has two inherent pitfalls. First, every internal node has, associated with it, a polyhedron which we will call its enclosing polyhedron. This polyhedron is formed by the set union of all the tetrahedra attached to the node. The boundary of the enclosing polyhedron for a given internal node is therefore delimited by the faces formed by all the nodes attached to the node in question. A two-dimensional analogy, the enclosing polygon of a node, is shown in Figure 7.5. A polyhedron is said to be star-shaped if there exists a point z such that for all points p in the polyhedron, the line segment zp

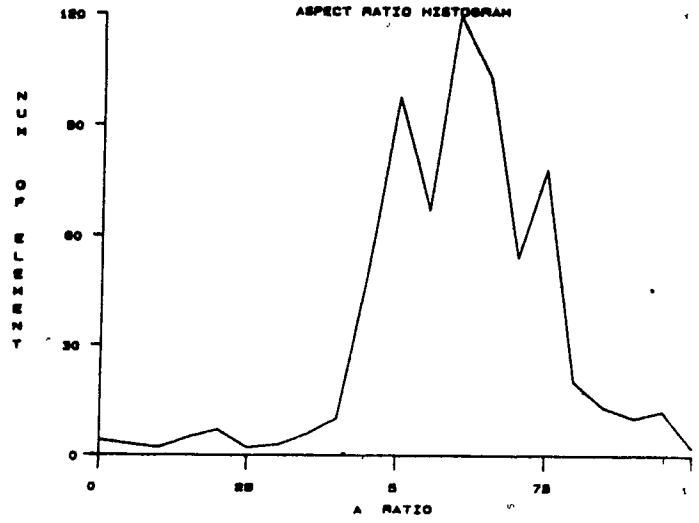

Figure 7.3

An aspect ratio histogram.
The aspect ratio of each element in the mesh is computed and the histogram of the aspect ratio distribution is constructed

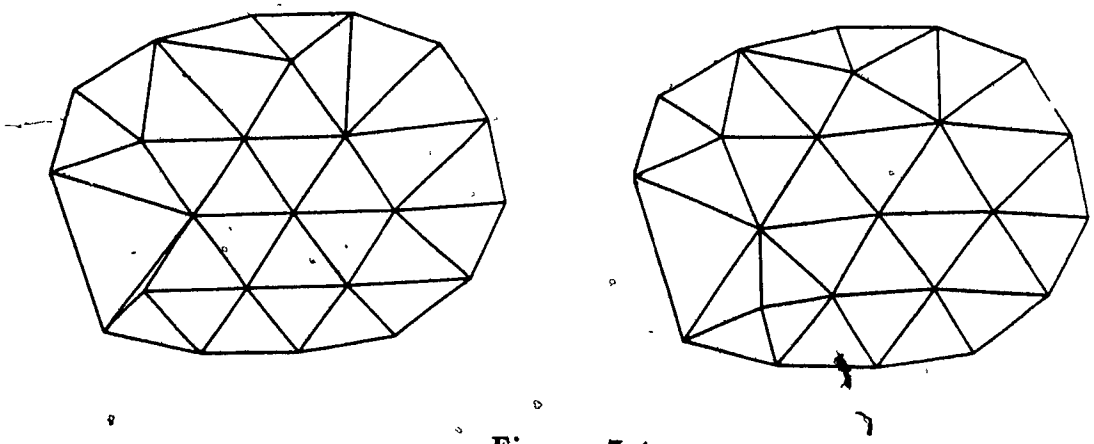

Figure 7.4

The 2-D node relaxation.
a) The mesh before the relaxation. Notice the bad shape of some of the triangles.
b) The mesh after the relaxation. Some triangles have lost their equilateral property, but the overall mesh has improved.

lies inside the polyhedron [Shamos 1978]. The kernel of a star-shaped polyhedron is the set of all the points z. An enclosing polyhedron therefore must be star-shaped, with the node it encloses lying in the kernel. In certain cases where there is a large concavity in the enclosing polyhedron, the node being relaxed may be displaced to a point outside the kernel, creating ill-formed tetrahedra and resulting in an inconsistent set of equations for the finite element solution. Furthermore, if such a concavity occurs at the external surface of the object being modelled, the relaxed node might end up outside the problem domain. In relaxing a node, therefore, care is taken to displace it only up to the boundary of the kernel of the enclosing polyhedron. This can be illustrated in a two-dimensional example. In Figure 7.6, moving the node to the center of its neighbouring nodes , while keeping it inside its enclosing polygon, puts it outside the kernel and produces degenerate elements. In Figure 7.7, on the other hand, displacing the node puts it completely outside the polygon.

Second, since external nodes are fixed, only internal nodes are displaced in the relaxation. Consequently, tetrahedra that have four nodes lying on the external shell stay fixed. If a badly shaped tetrahedron happens to have four fixed nodes, its shape cannot be improved by the relaxation. A solution to this problem might lie in screening the tetrahedra produced during the meshing by modifying the Vertex and Edge operators to reject any bad tetrahedra with four fixed vertices. Another solution might be to allow the relaxation of surface nodes, while constraining them to lie on a splined contour curve that passes through all the boundary nodes at that particular cross-section. A similar technique has successfully been used in two-dimensional node relaxation [Thacker et al. 1980].

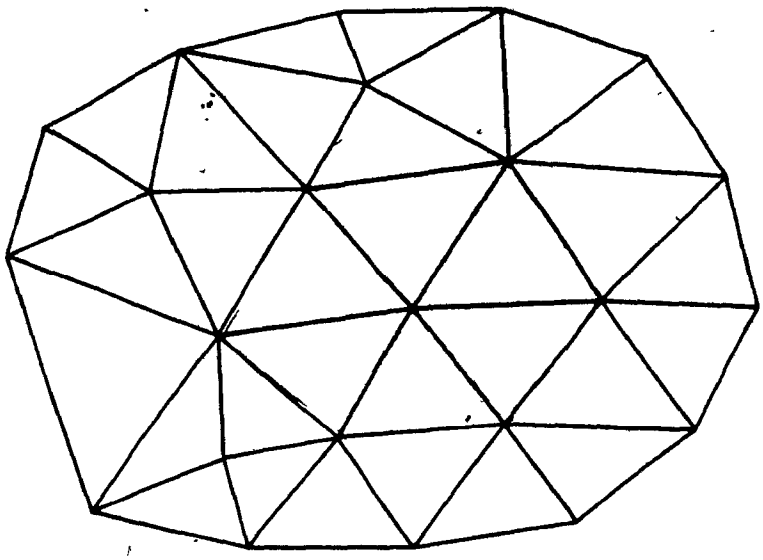

Figure 7.5

The enclosing polygon of a node.
The shaded polygon represents the enclosing polygon of the mesh node at its center. It is made of all the elements that the node belongs to.

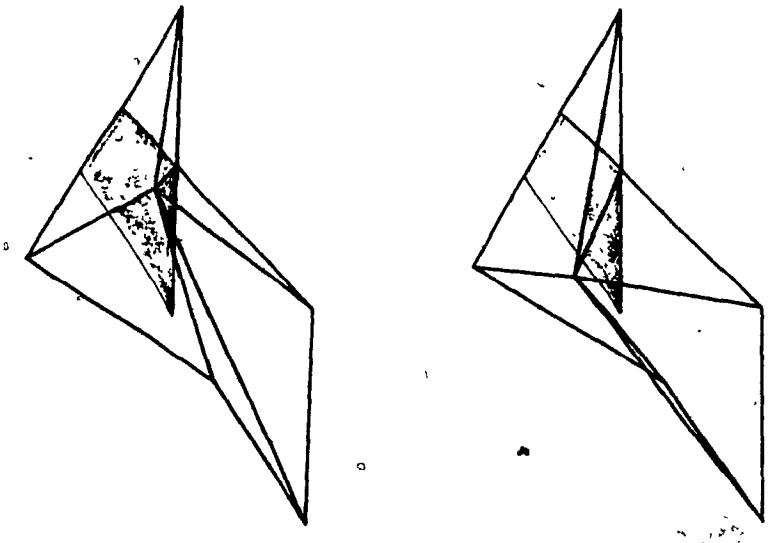

Figure 7.6

Relaxing a node outside the kernel.
Moving a node to the center of the nodes connected to it can put it outside the polygon kernel (shaded in the figure), producing degenerate elements.

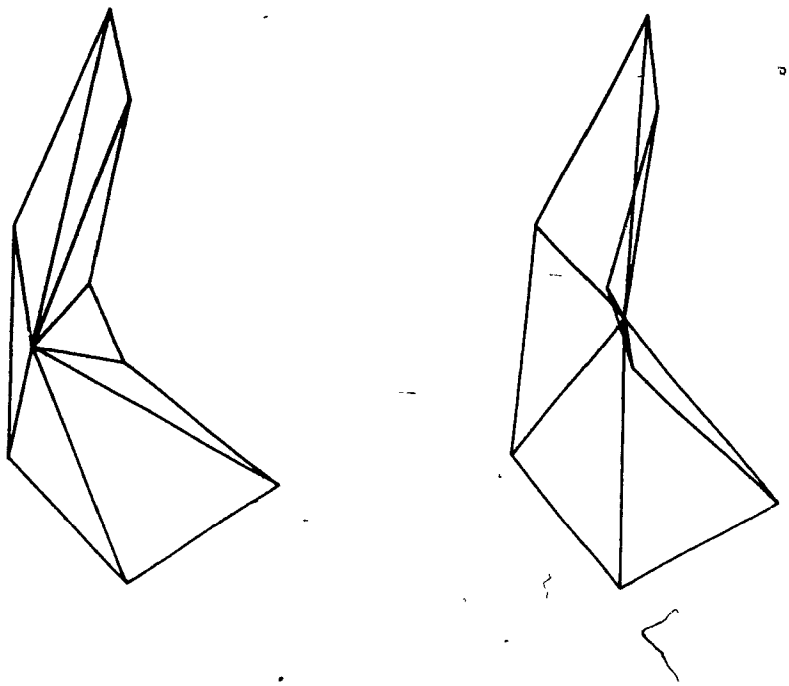

Figure 7.7

Relaxing a node outside the polygon
Moving a node can leave it completely outside the polygon, also producing degenerate elements.

7.5 Conclusion

Once all the slices have been processed, the individual meshes are merged to form the global mesh. The need for improving the global quality of the mesh has led to the use of relaxation techniques. The nodes inside the mesh are relaxed to a position at the geometric center of the nodes attached to them. The aspect ratio distribution for the mesh, reflecting the overall quality of the mesh, is then computed. Two major problems that can occur during the relaxation phase were described, and possible solutions to them were presented.

CHAPTER 8

Results

8.1 Introduction

Once an algorithm has been developed, it is important to analyze its usefulness in terms of speed and quality of results. For this purpose, the mesh generation method described in this thesis was tested on several problems of varying complexity. The number of nodes was used to characterize the size of each of the test problems used in measuring the performance of the method. The testing procedure will be described in this chapter and the results of the mesh generation and the relaxation will be presented.

8.2 The Testing Procedure

The method was implemented in 4500 lines of FORTRAN code on a Digital Equipment Corporation VAX 11/750 and MicroVAX II. The testing was done by generating several objects of varying complexities, using the program to mesh them and measuring and analyzing the corresponding run times. The complexity of a problem can usually be measured either by the number of nodes (V) or by the number of elements (T) in the mesh needed to model it. These two measures are closely related, and the bounds on the number of tetrahedra that an object can be discretized into [Woo & Thomasma 1984] is given by the relation

$$(V - 3) \leq T \leq \frac{(V - 3)(V - 2)}{2}$$

Given the same problem with the same number of nodes, several meshes, having different numbers of tetrahedra, can be produced to mesh it. Being the only controllable input parameter, the number of nodes was used in this instance to measure the complexity of a test problem.

To generate the test problems, a few contours were digitized and the two-dimensional triangulation step was performed, using a set of different resolutions for each of them. These triangulated contours were used to define several slices, having different numbers of nodes and constituting a test problem each. The meshing program was then run several times on each of the slices, using a different starting edge at each time. Data from each run was used to produce a plot of the meshing times for each of the slices, from the different starting edges. Although not complete problems in themselves, the produced slices are portions of complete three-dimensional problems.

8.3 Running Time and Complexity

Figure 8.1 shows the running time plots for problems of 26 and 80 nodes. The times shown were measured in CPU seconds on the MicroVAX II. As can be seen, there are some large fluctuations in the time it takes to mesh a slice, depending on the starting point for the meshing. Furthermore, these times fall into two or three distinct classes, depending on their value. This can be due to the fact that, depending on the starting point, the meshing rapidly converges to one of a few possible final mesh states, resulting in similar running times for starting points converging to the same state. Finding the optimal starting point, however, is still an open problem.

Problems of 22, 26, 36, 46, 58, 68, 74, 80 and 94 nodes were used in trial runs

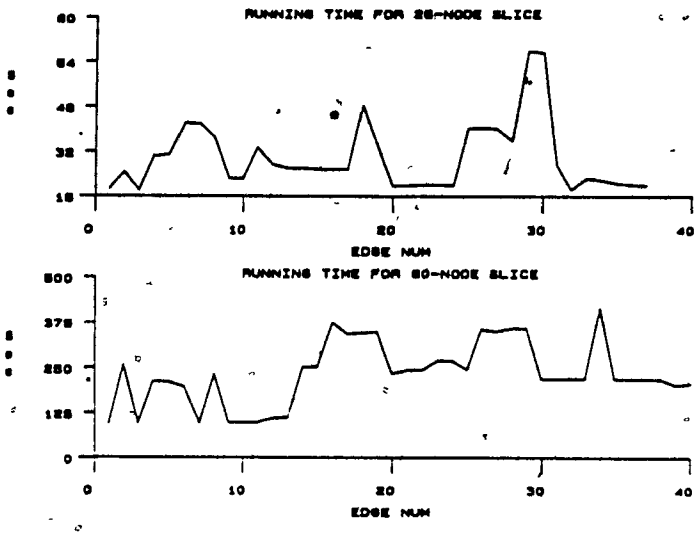

Figure 8.1

Running time plots.
The two plots shown are for slices of 26 and 80 nodes, respectively. The abcissa corresponds to the number of the starting point in the shredding and the ordinate to the time, in seconds, taken to shred the slice from a particular starting point.

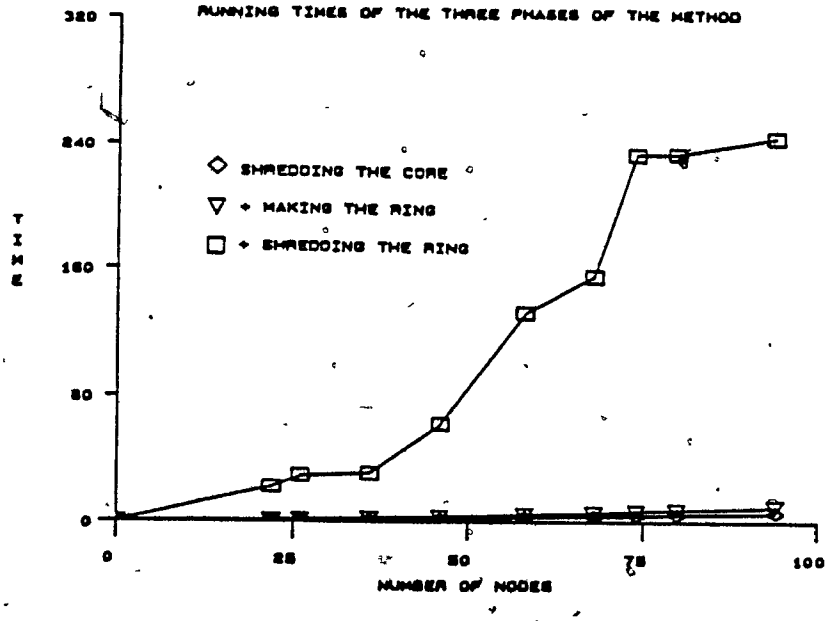

Figure 8.2

Time spent in the different phases.
The time taken to shred the core and assemble the ring is negligible when compared to the time taken to shred the ring.

to estimate the time complexity of the program. Figure 8.2 shows the average time taken in each of the three main stages of the method: meshing the core, assembling the ring and meshing the ring, the relaxation stage being negligible It is clear that most of the time is spent in the topological shredding phase. An estimate of the order of complexity of the method can readily be obtained from this measured data by fitting the mean running time for each problem against the problem size. Using a least squares fit, it is found that the polynomial

$$T = 0.079N^{1.8} - 0.043N$$

where N is the number of nodes in the final mesh, accounts for 94.0 % of the variance and provides a good model of the time complexity as a function of problem size (Figure 8.3). This suggests a time complexity of $O(N^2)$ for the method to process a slice of N nodes. A generalization of this result to an object consisting of several slices follows.

Consider an object composed of s slices and having a total of N nodes For any slice, the number of nodes n_i is given by

$$n_i = t_i - b_i$$

where t_i is the number of nodes in the top surface triangulation of the slice and b_i that in the bottom surface. Assuming that slice index i increases from top to bottom, the total number of nodes in the object is given by

$$N = t_1 + t_2 + \ldots + t_s + b_s$$

An upper bound on N is easily obtained:

$$\sum_{i=1}^{s} n_i = t_1 - b_1 + t_2 + b_2 + \ldots + t_s + b_s$$
$$= t_1 + 2(t_2 + t_3 + \ldots + t_s) + b_s$$

since

$$b_i = t_{i+1} \quad 1 \leq i < s$$

Subtracting the value of N yields

$$\sum_{i=1}^{s} n_i - N = t_2 + t_3 - \ldots + t_s \quad \Rightarrow \quad N < \sum_{i=1}^{s} n_i$$

A lower bound on N can also be obtained:

$$\frac{1}{2}\sum_{i=1}^{s} n_i = \frac{1}{2}t_1 + (t_2 + t_3 + \ldots + t_s) + \frac{1}{2}b_s$$

adding $\frac{1}{2}(t_1 + b_s)$ to both sides gives

$$\frac{1}{2}\sum_{i=1}^{s} n_i + \frac{1}{2}(t_1 + b_s) = t_1 + t_2 + t_3 + \ldots + t_s + b_s = N$$

which results in

$$\frac{1}{2}\sum_{i=1}^{s} n_i < N < \sum_{i=1}^{s} n_i$$

or, as an approximation, taking $n_i = n$ for all the slices,

$$\frac{1}{2}sn < N < sn$$

Assuming $O(n^2)$ for each slice using the method, the time complexity is $O(sn^2)$. For $s > 4$, this is better than $O(N^2)$

8.4 The Relaxation

When slices forming a single object are merged and the aspect ratio histogram of the complete mesh is computed, a few badly shaped elements will sometimes appear, as shown in Figure 8.4. The mesh represented in this example has an aspect ratio sum of 141.8 and contains four zero-volume elements that were created by the interface tetrahedron operator. As seen in Figure 8.5, the node relaxation

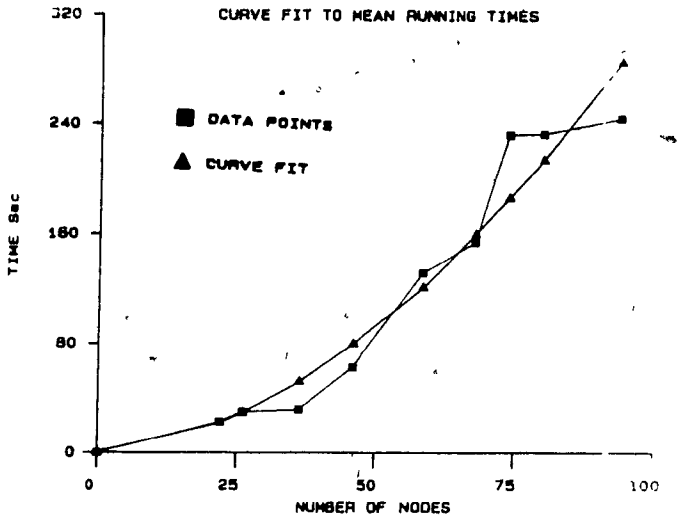

Figure 8.3

Curve fit to running time data.
The average time taken to shred each slice is plotted against the number of nodes in the slice, and the resulting data points fitted to a curve

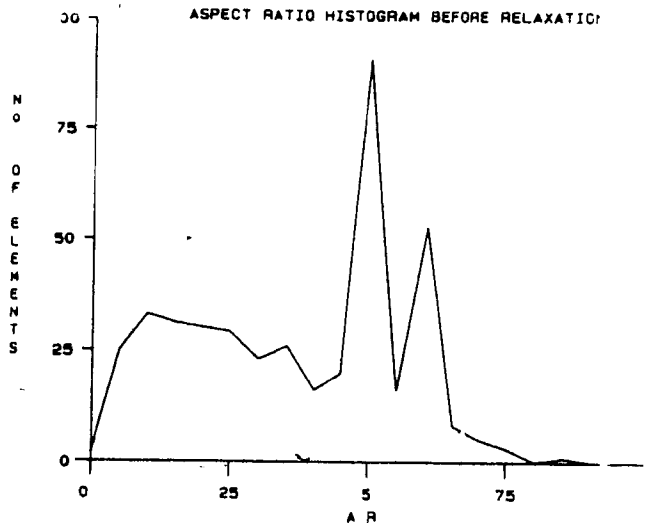

Figure 8.4

Aspect ratio histogram of a mesh before the relaxation.
Notice that there are some zero-volume elements in the mesh, making it useless for computations. The two peaks are due to the two types of tetrahedra generated by shredding the core, which makes the bulk of the mesh.

improves the mesh by shifting its histogram to the right and eliminating all the zero-volume elements. In this particular example the aspect ratio of the worst element in the mesh is increased from 0.0 (degenerate) to 0.1 and the aspect ratio sum is increased to 196.8, reflecting the overall improvement of the elements in the mesh.

8.5 Conclusion

The method was successfully tested by using it to mesh several test problems of varying sizes. The mean running times for the problems were then fitted to the problem sizes using a least-squares approximation method, obtaining a complexity of $O(n^{1.8})$ in the range being fitted. This suggests a time complexity function of order $O(n^2)$ for a slice. The iterative node relaxation method used was found to improve the overall shape of the produced meshes and eliminate degenerate tetrahedra in most of the cases. Variations of the method or manual node repositioning could be used to completely eliminate degenerate elements when necessary

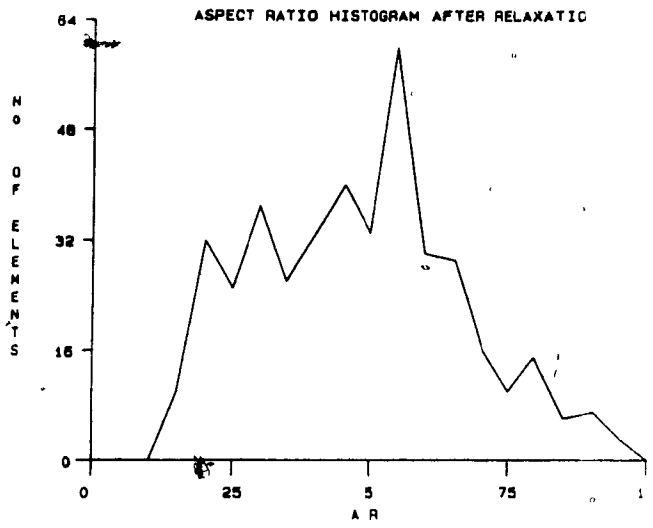

Figure 8.5

Aspect ratio histogram after the relaxation
The zero-volume elements have been eliminated and the overall quality of the mesh improved, as seen by the shifting of the histogram to the right

CHAPTER 9

Discussion

9.1 Summary

Although three-dimensional mesh generation has already been treated by several authors, the problem of modelling highly irregular structures had not been properly addressed yet. This thesis has presented a new method for automatic generation of three-dimensional finite element meshes that is especially useful in meshing irregular domains.

In this method, the geometry of the object is described by a set of cross-sectional contours, generally referred to as serial sections. The contours are first triangulated on a grid of nodes, then paired up to form slices. Each slice is meshed separately and later merged with all the others to form the global mesh.

The center of each slice is easily shredded by joining nodes from the triangulated top and bottom surfaces of the slice to form pentahedral prisms that are subsequently divided into three tetrahedra each. Once this core has been removed, a vertex-edge-face polyhedral representation of the remaining portion of the slice is assembled and passed on to a topological shredder.

The topological shredding is done through four operators that act on the topology of the object by modifying the number of vertices, edges and faces. Two cutting operators remove tetrahedra from the object, according to certain rules, and

another operator is used to reduce the genus of a multiply-connected object by performing a topological cut. Still, not all polyhedra are guaranteed to be shredded this way, by using only those three operators, and three classes of irreducible polyhedra have been encountered. A fourth operator can further reduce two of these classes by flipping external diagonals, or by introducing a flat interface tetrahedron. If a polyhedron of the third class is encountered, however, the mesh generation must be halted, but it has been found that the slice can usually still be shredded by restoring the slice structure and restarting the shredding at a different place.

Once all the slices have been completed, they are merged and the data structure rearranged by replacing the local node numbering for each slice by a global one for the mesh, and deleting all duplicate entries. The internal nodes are then relaxed to improve the shape and quality of the elements. The method was successfully tested on a number of problems and its time complexity was found to be of $O(sn^2)$ in the number of nodes.

9.2 Modifications and Further Developments

9.2.1 Branching Objects

As it stands now, the program cannot handle branching objects, although the same concepts and operators apply. The main stumbling block lies in the fact that a branching object will have different numbers of contour curves at cross-sections before and after the branch (Figure 9.1), so that cross-sections of the branches have to be matched to their generating cross-section. Several methods have been suggested [Sabin & Funnell 1984, Zsuppán & Réthelyi 1985] and can be incorporated in the method.

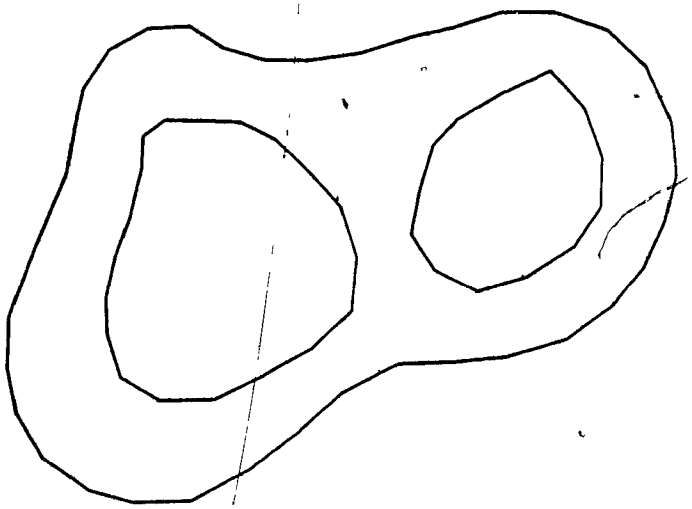

Figure 9.1 .

Sections of a branching object.
Cross-sections of a branching object The large contour is the cross-section before the branch. The two smaller ones are the cross-sections after the branch.

9.2.2 Optimal Starting Point

Another interesting development consists of finding an optimal starting point to mesh a given slice. Presently, a starting point is chosen at random. It is rejected if the meshing fails, and kept if it succeeds, even if the resulting mesh is far from optimal. Locally and globally optimal two-dimensional triangulations can be produced [Fuchs et al. 1977] without having to try all the possible triangulations of a domain but no such solution for this problem has been found yet in three-dimensional meshing

9.3 Conclusion

The method proposed in this thesis is a mixed method combining the speed and simplicity of look-up table shredding when possible (in the core) and the thoroughness of topological shredding when necessary (in the remaining layer). This gives it several advantages over existing methods. Mainly, it makes possible the handling of arbitrarily-shaped, highly irregular objects at relatively low cost while producing meshes of good quality. The only other methods to allow handling of arbitrary shapes with good results are the filling methods discussed in section 2.2.3. However, filling techniques are computationally more expensive, since a theoretical lower bound of $O(N^2) = O(s^2n^2)$ has been shown for the running time of three-dimensional Delaunay triangulations [Shamos 1978], and they present other problems, related to non-convex shapes and objects with holes, that have already been discussed. Since the other topological methods have been shown to be incomplete, the method compares well with other existing automatic methods, being computationally cheaper than all but some of the simple mapping techniques, and producing meshes of better quality for irregular shapes.

APPENDIX

A Brief Review of Topology

A.1 Topology and Topological Properties

It is not the intention of this appendix to give a course in topology. Indeed, several excellent books, ranging from the introductory to the highly abstract, cover the subject very thoroughly [Alexandroff 1961, Armstrong 1983, Courant 1941, Griffiths 1981, Hilbert & Cohn-Vossen 1952, Lakatos 1983]. However, a brief introduction to topological concepts is in order to provide a mathematical background to the discussions found in some of the chapters. Let X and Y be two spaces with a distance function d defined on them, having the following properties: For $x, y, z \in X, Y$

I. $d(x,y) \geq 0$ and $d(x,y) = 0 \iff x = y$.

II. $d(x,y) + d(y,z) \geq d(x,z)$ (The Triangle law).

III. $d(x,y) = d(y,x)$ (The Symmetry law).

Let the function f be a transformation from X to Y. If, for $x_1, x_2 \in X$ and $y_1, y_2 \in Y$, we have

$$(x_1 \neq x_2) \Rightarrow f(x_1) \neq f(x_2)$$

or equivalently,

$$\exists f^{-1}; f(x_1) = y_1 \Rightarrow f^{-1}(y_1) = x_1$$

then we say that f is one-to-one, denoted as 1-1.

The transformation f is said to be continuous at the point x_1 if, given $\sigma > 0$, $\exists \rho > 0$ such that

$$\text{if } y_2 = f(x_2) \text{ then } d(y_1, y_2) < \sigma \iff d(x_1, x_2) < \rho.$$

In other words, to have y_2 near y_1 it is sufficient to take x_2 near x_1. If f is continuous at all points, it is said to be a continuous transformation or a mapping. A very simple example of a mapping is the mapping of the unit line segment $0, 1)$ onto the unit circle by the function

$$f : 0, 1) \rightarrow C$$

$$f(t) = (cos\ 2\pi t, sin\ 2\pi t), \quad 0 \leq t < 1$$

Topology is concerned with those properties of point sets which are invariant under continuous deformations. These properties are called topological properties. For example, one property of a closed curve drawn on a rubber sheet is that it divides the sheet into two regions, one inside the curve and the other outside, even if the rubber sheet is deformed, without being torn. The deformations in question are called topological mappings or homeomorphisms, and they have two basic properties.

1. They are one-to-one.

2. They are bicontinuous, meaning that both f and f^{-1} are continuous at all points.

Sets of points that can be mapped into each other by topological transformations are said to be homeomorphic or topologically equivalent. As an example

consider the mapping of the circle

$$C_1 : x^2 + y^2 = 1$$

to the circle

$$C_2 : x^2 + y^2 = 4$$

by the function

$$f \cdot C_1 \rightarrow C_2,\ f(x,y) = (2x, 2y)$$

The two circles are clearly homeomorphic. Note that, unlike this mapping, the mapping of the unit segment onto the circle is not a homeomorphism, since f^{-1} fails to be continuous at the point $(1,0)$. This makes sense, as it is clear to the eye that two circles are topologically equivalent, whereas a line segment and a circle are not. Properties that are not changed under topological mappings, for example the number of cuts (one) it takes to turn a circle into a segment, are called topological invariants.

A.2 Euler's Formula for Simple Polyhedra

One of the earliest (and most important for our purposes) topological invariants to be identified was the so-called Euler (or Euler-Poincaré) formula for simple polyhedra. For our present purposes, a polyhedron is defined as a solid whose surface consists of polygonal faces, and a simple polyhedron as a polyhedron with no 'holes' in it so that its surface can be deformed continuously into the surface of a sphere [Courant 1941]. The formula was first used by Descartes and rediscovered by Euler. It states that, for any simple polyhedron,

$$V - E + F = 2$$

where V is the number of vertices in the polyhedron, E the number of edges and F the number of faces (Figure A.1.a). Proofs can be found in several texts [Courant

1941, Lakatos 1983]. Since it is a topological invariant, continuously deforming the polyhedron does not affect the formula, and it stands for curved polyhedra, i.e. polyhedra with curved surfaces. This formula is not correct however in characterizing objects with holes (Figure A.1.b) since simple polyhedra cannot be continuously deformed into polyhedra with holes, and vice-versa. A modified formula taking into account the number of holes will be introduced later.

A.3 Connectivity and Genus

Two surfaces are shown in Figure A.2; they are obviously not topologically equivalent. The question is whether there is any way to classify them as different. Now consider this; any closed curve in Figure A.2.a can be shrunk down to a point inside the surface without crossing the boundary. This type of surface is known as a simply connected domain. This is not the case in Figure A.2.b, as shown by the closed curve drawn inside the surface. This surface represents a multiply (doubly in this case) connected domain. Transforming this domain into a simply connected one will alter its topological characteristics. The operation is called a topological cut and involves joining the two boundary curves by a double arc (Figure A.3). The number of cuts needed to transform a multiply connected domain into a simply connected one is referred to as the genus of the domain. After performing the cut, it is clear that the new surface is simply connected.

A.4 Orientation

We will start this section by defining the concept of a simplex. Generally, an N-dimensional simplex is the smallest convex set containing $(N+1)$ vertices, such that the $(N+1)$ vertices are not contained in an $(N-1)$-dimensional hyperplane. Thus a one-dimensional simplex is a straight line segment; a two-dimensional sim-

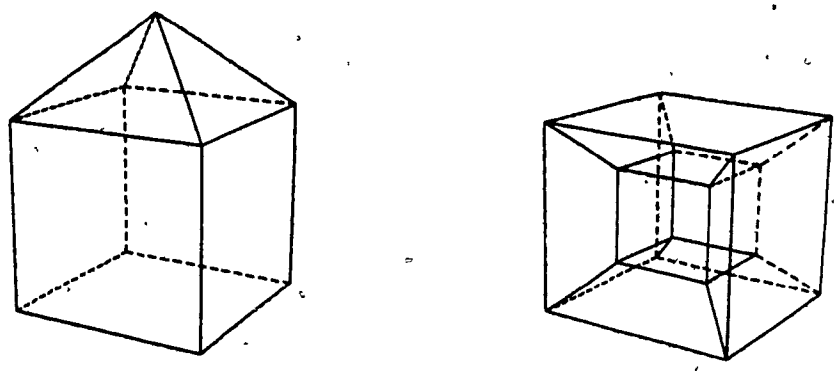

Figure A.1

Euler's equation for simple polyhedra.
a) Simple polyhedron $V - E + F = 2$
a) Polyhedron with a hole: $V - E + F \neq 2$

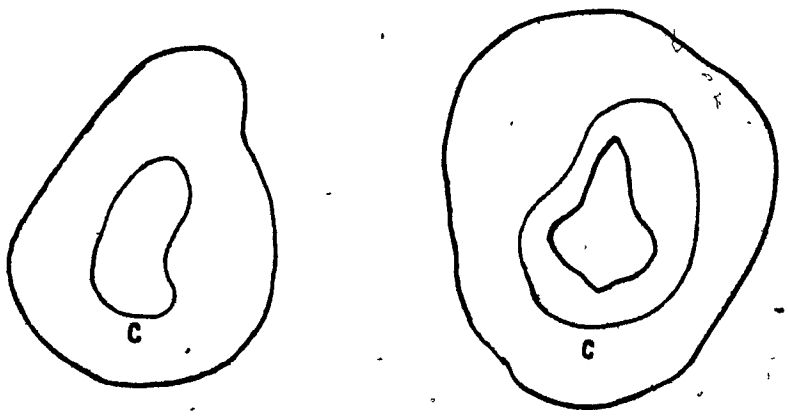

Figure A.2

(a) Simply and (b) doubly connected domains

plex is a planar triangular surface; a three-dimensional simplex is a tetrahedron, and so on. A geometrical complex is a set of simplices such that any two simplices either have no points in common, or have a face (of any dimension) in common. A finite-element mesh is an example of a geometric complex.

Another important concept is that of orientation. An oriented one-dimensional simplex is a directed line segment $(a_0 a_1)$, i.e. a segment traversed from a_0 to a_1. If we denote the oriented simplex by

$$x^1 = (a_0 a_1)$$

then

$$x^1 = (a_1 a_0)$$

and

$$x^1 = a_0 a_1 = -a_1 a_0$$

Similarly, an oriented two-dimensional simplex is a triangle with a particular sense of rotation or vertices ordering, so that

$$x^2 = (a_0 a_1 a_2) = (a_1 a_2 a_0) = (a_2 a_0 a_1)$$

$$-x^2 = (a_0 a_2 a_1) = (a_2 a_1 a_0) = (a_1 a_0 a_2)$$

Note that to invert the orientation of a triangle it suffices to flip the ordering of two of its nodes, i.e.

$$(a_0 a\ a_2) = -(a_0 a_2 a_1)$$

Denoting the boundary of x^2 by $\dot{x}^2$, we have

$$\dot{x}^2 = (a_0 a_1) + (a_1 a_2) + (a_2 a_0)$$

$$\dot{x}^2 = (a_0 a_1) + (a_1 a_2) + (a_0 a_2)$$

An oriented geometric complex is a geometric complex whose simplices are oriented. Now consider the two-dimensional oriented complex C^2 shown in Figure A.4.

$$C^2 = \sum x_i^2 = x_1^2 + x_2^2$$

and taking

$$C^2 - \sum x_i^2 = x_1^2 + x_2^2$$

we have that

$$x_1^2 = (a_0a_1) - (a_1a_3) + (a_3a_0)$$

$$x_2^2 = (a_1a_2) - (a_2a_3) - (a_3a_1)$$

$$= (a_1a_2) + (a_2a_3) - (a_1a_3)$$

this gives

$$C^2 = (a_0a_1) + (a_1a_2) + (a_2a_3) + (a_3a_0)$$

which is the oriented boundary curve of the complex. These concepts are obviously extensible to N-dimensional point sets, but most important to this thesis are the three-dimensional topology considerations.

A.5 Three-dimensional Topology

In three dimensions, a simply connected polyhedron is homeomorphic to a sphere, and has a genus of 0. A polyhedron with one hole is homeomorphic to a torus. A torus is not simply connected since one cut is not always sufficient to divide it in two pieces. Transforming a torus into a simply connected domain involves one cut (Figure A.5), thus a torus and all its homeomorphs have a genus of 1. Incorporating the genus concept into the Euler formula (A.1), the modified Euler formula states that, for any polyhedron,

$$V - E + F = 2 - 2G.$$

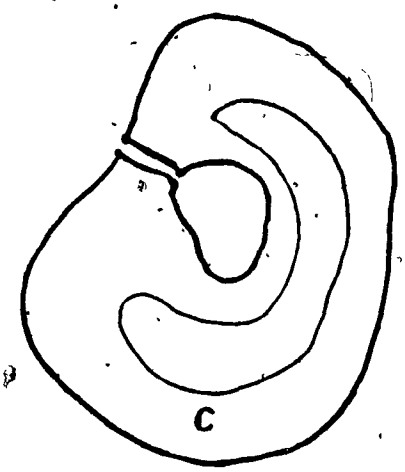

Figure A.3

Reducing the genus of a surface
Performing a topological cut on a surface by joining the two boundary curves

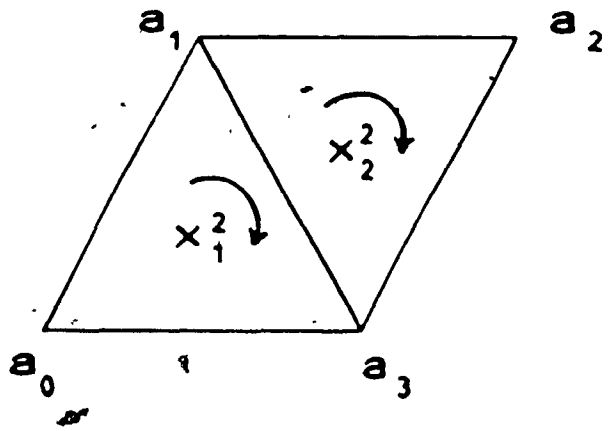

Figure A.4

A 2-D oriented complex

where G is the genus of the polyhedron. Again in three dimensions, considering as a two-dimensional complex the surface of a three-dimensional polyhedron will yield another interesting result. Denoting the complex shown in Figure A.6 by

$$C^2 = \sum x_i^2 = x_1^2 - x_2^2 + x_3^2 + x_4^2$$

and its boundary curve by

$$C^2 = \sum x_i^2 = x_1^2 - x_2^2 + x_3^2 + x_4^2$$

and, without loss of generality, if we take

$$x_1^1 = (a_0 a_1)$$

$$x_2^1 = (a_1 a_2)$$

$$x_3^1 = (a_2 a_0)$$

$$x_4^1 = (a_0 a_3)$$

$$x_5^1 = (a_1 a_3)$$

$$x_6^1 = (a_2 a_3)$$

then substituting for

$$x_1^2 = x_1^1 - x_4^1 - x_5^1$$

$$x_2^2 = x_2^1 - x_5^1 + x_6^1$$

$$x_3^2 = x_3^1 - x_4^1 - x_6^1$$

$$x_4^2 = -x_1^1 - x_2^1 - x_3^1$$

results in

$$\dot{C}^2 = \sum \dot{x}_i^2 = 0$$

which makes intuitive sense, since a three-dimensional solid cannot have a boundary *curve*. This will provide a way to check the consistency of generated structures and meshes. In this example, we have assumed that all simplices are identically

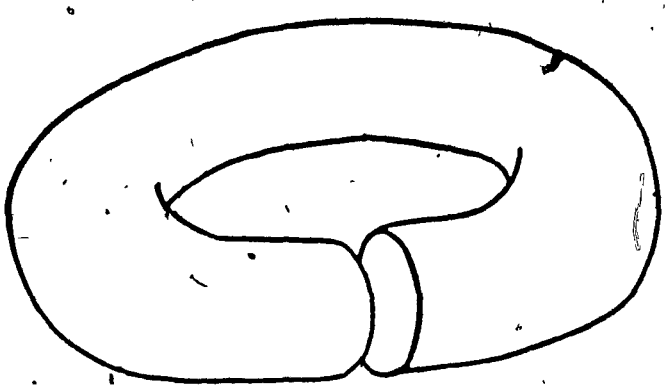

Figure A.5

Reducing the genus of a torus
Performing a topological cut on a torus to reduce its genus

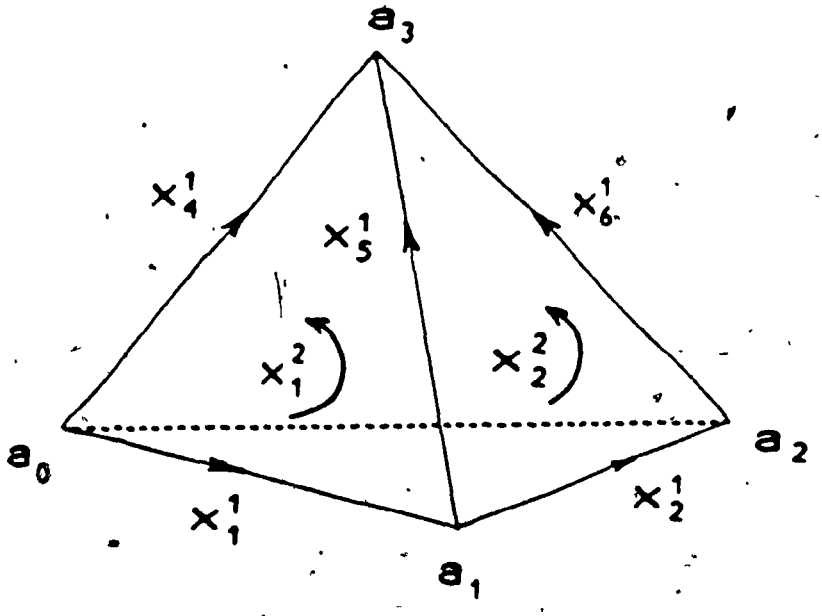

Figure A.6

A 3-D oriented complex

oriented, resulting in a null boundary curve for the complex. If, on the other hand, we consider a complex with differently oriented simplices as shown in Figure A.7, the sum

$$\dot{C}^2 = \sum \dot{x}_i^2$$

will yield a non-zero result

$$\dot{C}^2 = 2\big((a_0a_1) + (a_1a_2) + (a_2a_3) + (a_3a_4) + (a_4a_0)\big) = 2B^1$$

where B^1 is the boundary curve separating the two differently oriented regions. In the same spirit, the orientation concept is useful while generating the mesh itself. All tetrahedra in the mesh should have the same orientation. This makes it easier, for example, to compute the boundary surface of the domain for boundary conditions considerations. once the mesh is complete, by performing the sum

$$Surface = \sum x_i^3$$

where $\dot{x}_i^3$ are the three-dimensional simplices, i.e. the tetrahedra. This property is also later used in determining the convexity of a vertex or an edge. In addition, all triangles on the surface of the domain should be similarly oriented. The orientation adopted in this work is by listing the nodes counter-clockwise as viewed from the outside, making the vectors normal to the faces point outwards and away from the polyhedron. Any number of triangles that are not consistently oriented will cause the sum $(\sum \dot{x}_i^2)$ to be different from zero, and equal to twice the separating boundary.

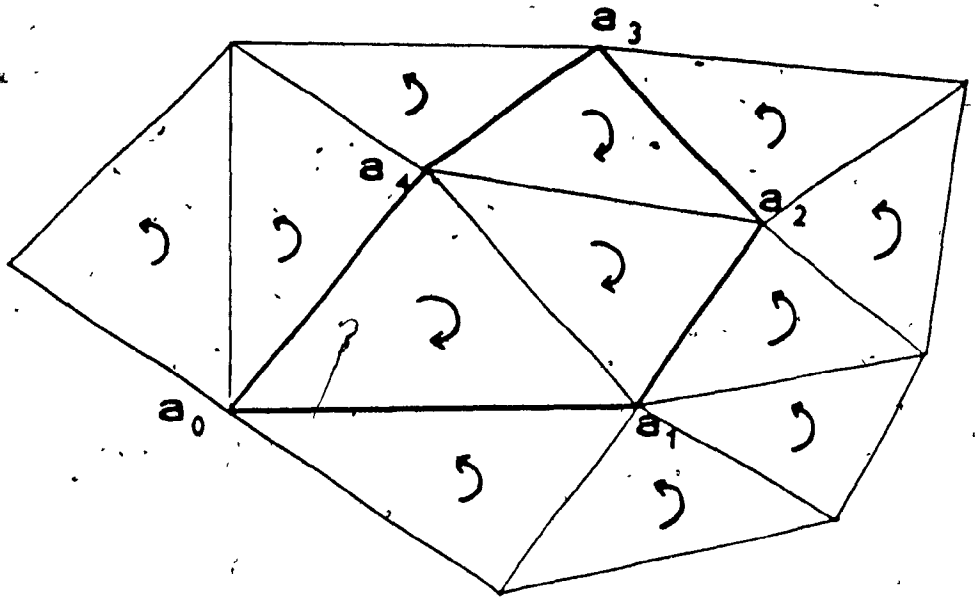

Figure A.7

A non-oriented 2-D complex.

A geometric complex with differently oriented simplices The sum of all the simplices will give the external boudary of the complex and twice the internal boundary of the area of differently oriented simplices.

References

Alexandroff, P., *Elementary Concepts of Topology*, Dover Publications, New York, 1961.

Argyris, J.H., " Tetrahedron elements with linearly varying strain for the matrix displacement method.," *J. R. Aero. Soc.*, vol. 69, 1965, pp 877–880.

Argyris, J.H., " The TET 20 and the TEA 8 elements for the matrix displacement method.," *J. R. Aero Soc.*, vol. 72, 1968, pp. 618–623.

Armstrong, M A., *Basic Topology*, Springer-Verlag, New York, 1983.

Babŭska, I., and Aziz, A.K., " On the angle condition in the finite element method.," *SIAM J. Numer. Anal.*, vol. 13, no. 2, 1976, pp. 214–226.

Baer, A., Eastman, C., and Henrion, M., " Geometric modelling: a survey.," *Comp. Aided Design*, vol. 11, no. 5, 1979, pp. 253–271.

Bagemihl, F., " On indecomposable polyhedra.," *Amer. Math. Monthly*, vol. 55, 1948, pp. 411–413.

Baker, A.J., *Finite Element Computational Fluid Mechanics*, McGraw-Hill. New York, 1983.

Barnhill, R.E., and Whiteman, J.R., " The Mathematics of Finite Elements and Applications.", edited by J.R. Whiteman, Academic Press, New York, 1973, pp. 83–112.

Bathe, K.J., Wilson, E.L., and Peterson, F.E., " SAP IV: A structural analysis program for static and dynamic response of linear systems," *Report EERC 73-11*, 1973.

Biffle, J.H., and Sumlin, H.A., " Three-dimensional structural analysis using interactive graphics.," *Comput. & Structures*, vol. 2, 1977, pp. 67–73.

Bramble, J.H., and Zlámal, M., " Triangular elements in the finite element method.," *Math. Comput.*, vol. 24, no. 112, 1971, pp. 809–820.

Brostow, W., Dussault, J.P., and Fox, B.L, " Construction of Voronoi polyhedra," *J. Comp. Phys.*, vol. 29, 1979, pp. 81–92.

Bryant, C.F., and Freeman, E.M., " Three dimensional finite element preprocessing of magnetic field problem.," *IEEE Trans. Magnetics*, vol. 20, no. 5, 1984, pp. 1897–1899.

Cavendish, J.C., Field, D.A., and Frey, W.H., " An approach to automatic three-dimensional finite element mesh generation.," *Int. J. Numer. Meth. Eng.*, vol. 21, 1985, pp. 329–347.

Cendes, Z.J , and Shenton, D.N., " Adaptive mesh refinement in the finite element computation of magnetic fields.," *IEEE Trans. Magnetics*, vol. 21, no 5, 1985, pp. 1811–1816

Chari, M.V.K., " Finite Elements in Electrical and Magnetic Field Problems.", edited by M.V.K. Chari & P.P. Silvester, John Wiley & Sons. 1980, pp 87–107

Chung, T.J., *Finite Element Analysis in Fluid Dynamics*, McGraw-Hill, New York, 1975.

Ciarlet, P.G., " Orders of convergence in finite element methods.," *The Mathematics of Finite Elements and Applications.*, edited by J.R. Whiteman, Academic Press, New York, 1973, pp. 113–129.

Clough, R.W., " Comparison of three dimensional finite elements." *Proc. Symp. Application of Finite Element Methods in Civil Eng.*, 1969, pp. 1–26.

Cook, W.A., " Body oriented (natural) co-ordinates for generating three-dimensional meshes.," *Int. J. Numer. Meth. Eng.*, vol. 8, 1974, pp. 27–43.

Coulomb, J.L., Terrail, Y. Du, and Meunier, G., " Two 3D parametered mesh generators for the magnetic field computation.," *IEEE Trans. Magnetics*, vol. 20, no. 5, 1984, pp. 1900–1902.

Courant, R., and Robbins, H., *What is Mathematics?*, Oxford University Press, New York, 1941.

Ewing, D.J.F., Hawkes, A.J., and Griffiths, J.R., " Rules governing the number of nodes and elements in a finite element mesh.," *Int. J. Numer. Meth. Eng.*, vol. 2, 1970, pp. 597–601.

Fajans,, " 3-dimensional display," *Adv. Disp. Tech., Proc. SPIE*, vol. 199, 1979.

Fuchs, H., Kedem, Z.M., and Uselton, S.P., " Optimal surface reconstruction from planar contours.," *Comm. ACM*, vol. 20, no. 10, 1977, pp. 693–702.

Funnell, W.R.J., and Laszlo, C.A., " Modeling of the cat eardrum as a thin shell using the finite-element method.," *J. Acoust Soc. Am.*, vol. 63, 1978, pp. 1461–1467.

Funnell, W.R.J., " On the undamped natural frequencies and mode shapes of a finite-element model of the cat eardrum," *J. Acoust. Soc. Am.*, vol. 73, 1983, pp. 1657–1661.

Funnell, W.R.J., " On the choice of a cost function for the reconstruction of surfaces by triangulation between contours.," *Comput. & Structures*, vol. 18, 1984, pp. 23–26.

Gallagher, R.H., *Finite Element Analysis Fundamentals*, Prentice Hall, Englewood Cliffs, N.J., 1975

Gaunt, W.A., *Microreconstruction*, Pitman Medical, London, 1971.

Greenberg, M.J., *Euclidian and Non-Euclidian Geometries*, W.H. Freeman & Co., San Francisco, 1974.

Griffiths, H.B., *Surfaces*, Cambridge University Press, Cambridge, 1981.

Harris, L.D., Camp, J.J., Ritman, E.L., and Robb, R.A., " Three-dimensional display and analysis of tomographic volume images utilizing a varifocal mirror.," *IEEE Trans. Med. Imaging*, vol. 5, no. 2, 1986, pp. 67–72.

Haugerud, M.H., " Interactive 3D mesh generation by an idealization and mapping technique.," *Third International Conf. and Exhibition on Computers in Engineering and Building Design*, 1978.

Heighway, E.A., and Biddlecombe, C.S., " Two dimensional automatic triangular mesh generation for the finite element electromagnetics package PE2D.," *IEEE Trans. Magnetics*, vol. 18, no. 2, 1982, pp. 594–598.

Hermeline, F., " Triangulation automatique d'un polyèdre en dimension N.," *R.A.I.R.O. Analyse Numérique*, vol. 16, no. 3, 1982, pp. 211–242.

Hilbert, D., and Cohn-Vossen, S., *Geometry and the Imagination*, Chelsea Publishing Company, New York, 1952.

Hughes, J.R., and Allik, H., " Finite elements for compressible and incompressible continua," *Proc. Symp. Application of Finite Element Methods in Civil Eng.*, 1969, pp. 27–62.

Imafuku, I., Kodera, Y., and Sayawaki, M., " A generalized automatic mesh generation scheme for finite element method.," *Int. J. Numer. Meth. Eng.*, vol. 15, 1980, pp. 713–731.

Jansson, D.G., Berlin, E.P., Straus, I., and Goodhue, J.T., " 3-dimensional computer display.," *Proc. 1st Annual Conf. Comp. Graph. in CAD/CAM Systems*, 1979, pp. 282–292.

Kamel, H.A., and Eisenstein, H.K., " Automatic mesh generation in two and three dimensional inter-connected domains.," *Symp. High Speed Computing of Elastic Structures.*, 1970.

Kamel, H.A., and Shanta, P.J., " A solids mesh generator and display package.," *ASME Papers*, no. 74-PVP-34, 1974.

Keppel, E., " Approximating complex surfaces by triangulation of contour lines.," *IBM J. Res. & Dev.*, vol. 19, no. 1, 1975, pp. 1–11.

Kirchhoff, L.W., " Computer Graphics for 3-D finite element models.," *ASME Papers*, no. 74-PVP-31, 1974.

Lakatos, I., *Proofs and Refutations*, Cambridge University Press, Cambridge, 1983.

Lennes, N.J., " Theorems on the simple finite polygon and polyhedron.," *Amer. J. Mathematics*, vol. 33, 1911, pp. 37–62.

Little, R.B., Wevers, H.W., Siu, D., and Cooke, T.D.V., " A three-dimensional finite element analysis of the upper tibia.," *J. Biomech. Eng.*, vol. 108, 1986, pp. 111–119.

Mäntylä, M., " Topological analysis of polygon meshes.," *Comp. Aided Design*, vol. 15, no. 4, 1983, pp. 228–234.

Murai, MT., and Kagawa, Y., " Electrical impedance computed tomography based on a finite element model.," *IEEE Trans. Biomed. Eng.*, vol. 32, no. 3, 1985, pp. 177–184.

Nguyen-Van-Phai,, " Automatic mesh generation with tetrahedron elements.," *Int. J. Numer. Meth. Eng.*, vol. 18, 1982, pp. 273–289.

Okoshi, T., " Three-dimensional displays.," *Proc. IEEE*, vol. 68, no. 5, 1980, pp. 548–564.

Perucchio, R., Ingraffa, A.R., and Abels, J.F., " Interactive computer graphic preprocessing for three-dimensional finite element analysis.," *Int. J. Numer. Meth. Eng.*, vol. 18, 1982, pp. 909–926.

Pissanetzky, S., " KUBIK: An automatic three-dimensional finite element mesh generator.," *Int. J. Numer. Meth. Eng.*, vol. 17, 1981, pp. 255–269.

Preiss, K., " An algorithm to check the topological consistency of a general 3-D finite element mesh.," *Adv. Eng. Software*, vol. 3, no. 3, 1981, pp. 98–100.

Sabin, N., and Funnell, W.R.J., *A system for reconstruction of three-dimensional branching objects from serial sections.*, Unpublished project report, Biomedical Engineering Unit, McGill University, Montréal, Québec., 1984.

Schönhardt, E., " Über die Zerlegung von Dreieckspolyedren in Tetraeder.," *Mathematische Annalen*, vol. 89, 1928, pp. 309–312.

Shamos, M.I., *Computational geometry*, Yale University, Ph.D., 1978.

Sibson, R., " Locally equiangular triangulations." *Comp. J.*, vol. 21, no. 3, 1978, pp. 243–245.

Silvester, P.P., and Ferrari, R.L., *Finite Elements for Electrical Engineers.*, Cambridge University Press., New York, N.Y., 1983.

Smith, B., Brauner, K.M., Kennicot, P.R., Liewald, M., and Wellington, J., *Initial graphics exchange specification (IGES), Version 2.0.*, U.S. Department of Commerce, 1983.

Stasa, F.L., *Applied Finite Element Analysis for Engineers*, Holt, Rinehart & Winston, New York, 1985.

Stover, H.S., " Terminal puts three-dimensional graphics on solid ground.," *Electronics*, vol. , 1981, pp. 150–155.

Strang, G., and Fix, G.J., *An Analysis of the Finite Element Method.*, Prentice-Hall, Englewood Cliffs, N.J., 1973.

Thacker, W.C., Gonzalez, A., and Putland, G.E., " A method for automating the construction of irregular computational grids for storm surge forecast models.," *J. Comp. Physics*, vol. 37, no. 3, 1980, pp. 371–387.

Thornton, R.W., " Interactive modeling in three dimensions through two-dimensional windows.," *Proc. CAD78*, 1978.

Tipper, J.C., " The study of geological objects in three dimensions by the computerized reconstruction of serial sections.," *J. Geology*, vol. 84, 1976, pp. 476–484.

Van Oosterom, A., and Strackee, J., " The solid angle of a plane triangle.," *IEEE Trans. Biomed. Eng.*, vol. 30, no. 2, 1983, pp. 125–126.

Veen, A., and Peachey, L.D., " TROTS: A computer graphics system for three-dimensional reconstruction from serial sections.," *Comput. & Graphics*, vol. 2, 1977, pp. 135–150.

Watson, D.F., " Computing the n-dimensional Delauney tessellation with applications to Voronoi polytopes.," *Computer J.*, vol. 24, no. 2, 1981, pp. 167–172.

Watson, D.F., and Philip, G.M., " Systematic triangulations.," *Comp. V. Graph. & Image Proc.*, vol. 26, 1984, pp. 217–223.

Wördenweber, B., " Finite element mesh generation.," *Comp. Aided Design*, vol. 16, no. 5, 1984, pp. 285–291.

Woo, T.C., and Thomasma, T., " An algorithm for generating solid elements in objects with holes.," *Comput. & Structures*, vol. 18, 1984, pp. 333–342.

Yamashita, Y., and Takahashi, T., " Use of the finite element method to determine epicardial from body surface potentials under a realistic torso model.," *IEEE Trans. Biomed. Eng.*, vol. 31, no. 9, 1984, pp. 611–621.

Zienkiewicz, O.C., Irons, B.M., Scott, F.C., and Campbell, J.S., " Three dimensional stress analysis.," *Symp. High Speed Computing of Elastic Structures.*, 1970.

Zienkiewicz, O.C., and Phillips, D.V., " An automatic mesh generation scheme for plane and curved surfaces by 'isoparametric' co-ordinates.," *Int. J. Numer. Meth. Eng.*, vol. 3, 1971, pp. 519–528.

Zienkiewicz, O.C., *The Finite Element Method*, McGraw-Hill, 1977.

Zienkiewicz, O.C., " Finite elements—the basic concepts and an application to 3-D magnetostatic problems," *Finite Elements in Electrical and Magnetic Field Problems*, edited by M.V.K. Chari & P.P. Silvester, John Wiley & Sons, 1980, pp. 11–31.

Zlámal, M., " Some recent advances in the mathematics of finite elements.," *The Mathematics of Finite Elements and Applications.*, edited by J.R. Whiteman, Academic Press, New York, 1973, pp. 59–81.

Zsuppán, F., and Réthelyi, M., "Approximation of missing sections in computer reconstruction of serial EM pictures.," *J Neurosci. Meth.*, vol. 15, 1985, pp. 203–212.