3-D MESH GENERATION FOR FINITE-ELEMENT MODELLING OF COMPLEX NATURAL STRUCTURES

Hengjin Liu

Biomedical Engineering Department
McGill University

May 2006

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

Your file Votre référence
ISBN: 978-0-494-28608-1
Our file Notre référence
ISBN: 978-0-494-28608-1

ABSTRACT

The finite-element method is based on the discretization of a distributed system. The system subdivision process is called *mesh generation*. The number, order, geometric characteristics (shape, size and orientation) and physical characteristics (material properties, boundary conditions and load conditions) of elements in the mesh will affect the accuracy of the finite-element solution. The objective of this project is to establish a number of criteria for evaluating 3-D mesh-generation software, and to select the 3-D mesh generator that is most suitable for use in our software pipeline for modelling and simulation of complex natural structures. The evaluation criteria for this project include: ability to preserve the surface mesh during 3-D mesh generation; mesh quality; robustness; time efficiency; and cost. The mesh quality is assessed by visualization methods; histograms of the shape qualities and sizes of elements; solution residuals; condition numbers; and closeness to the exact solution as estimated by a convergence curve. Four unstructured mesh-generation programmes (GiD, Gmsh, GRUMMP and TetGen) have been evaluated. A thin block and three structures (one ligament and two ossicles) of the middle ear were chosen to be the models for the evaluation. A programme was developed to convert the surface definitions describing the models to the native file formats of the mesh-generation programmes, to verify that the surface meshes of these models could be successfully imported into the mesh-generation programmes, and to verify that the resulting volume meshes are topologically correct. The results of the evaluation indicate that the mean value of the shape qualities of elements, the root mean square of the solution residuals, and the closeness to the exact solution are good indicators of the overall mesh quality. The Gmsh programme is finally selected as the best 3-D mesh generator for the purposes of our software pipeline.

RÉSUMÉ

La méthode d'éléments finis est basée sur la discrétisation d'un système distribué. Le processus de subdivision d'un système s'appelle la génération de mailles. Le nombre, l'ordre, les caractéristiques géométriques (forme, taille et orientation) ainsi que les caractéristiques physiques (propriétés matérielles, conditions aux extrémités et conditions de charges) des éléments de la maille affecteront l'exactitude de la solution d'éléments finis. L'objectif de ce projet est d'établir un certain nombre de critères pour évaluer le logiciel de la génération de mailles 3-D, et de choisir le générateur de mailles 3-D le plus approprié à notre chaine de logiciels pour la modélisation et la simulation des structures complexes. Les critères d'évaluation pour ce projet incluent: la capacité de préserver la maille extérieure pendant la génération de la maille 3-D; la qualité de la maille; la robustesse; l'optimisation du temps; et le coût. La qualité de la maille est évaluée par des méthodes de visualisation; des histogrammes des qualités de forme et des tailles des éléments; des résiduels de solution; des nombres de condition; et la proximité à la solution exacte telle qu'estimée par une courbe de convergence. Quatre programmes de génération de mailles non structurés (GiD, Gmsh, GRUMMP et TetGen) ont été évalués. Un bloc mince et trois structures (un ligament et deux osselets) de l'oreille moyenne ont été choisis pour être les modèles d'évaluation. Un programme a été développé pour convertir les définitions des surfaces du modèle en formats de fichier spécifiques aux programmes de génération de mailles, pour vérifier que les mailles extérieures de ces modèles pourraient être importées avec succès dans les programmes de génération de maille, et pour vérifier que les mailles de volume ainsi créées sont topologiquement correctes. Les résultats de l'évaluation indiquent que la valeur moyenne des qualités de la forme des éléments, la moyenne quadratique des résiduels de solution, et la proximité à la solution exacte sont de bons indicateurs de la qualité globale de la maille. Le programme de Gmsh est finalement choisi comme étant le meilleur générateur de mailles 3-D pour notre chaine de logiciels.

ACKNOWLEDGEMENTS

I express my sincere appreciation to my supervisor Professor W. Robert. J. Funnell for his strong support, guidance and insight throughout the project, and for his patience with my English.

I would like to thank M.M. and O.W. Henson, Jr., of the University of North Carolina at Chapel Hill, for the MRM data, and previous members of Auditory Mechanics Laboratory for their contributions to generating the models used in the project.

I express my thanks and appreciation to my family for their understanding, motivation and patience. Lastly, but not least, I am thankful to all my colleagues and friends who made my stay at the university a memorable and valuable experience.

This work was supported by the Canadian Institutes of Health Research (CIHR) and the Natural Science and Engineering Research Council of Canada (NSERC).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

The finite-element method is a general tool for solving various physical problems. The finite-element method is based on the sub-division of a continuum into a finite number of discrete elements. The solution obtained will, in general, be only an approximation to the exact solution, which is rarely available when the continuum is an arbitrarily complex shape. The process of sub-division of the continuum is called *mesh generation*. The number, order, geometric characteristics (shape, size and orientation) and physical characteristics (material properties, boundary conditions and load conditions) of elements in the meshes will affect the computer storage requirements, the computation time and the accuracy of the finite-element analysis (Zienkiewicz and Taylor 2000).

Mesh-generation methods can roughly classified into structured and unstructured. An unstructured mesh is better for fitting a complex boundary than a structured mesh is. Therefore, unstructured mesh-generation methods have gained more attention in biomedical applications where the shapes of structures are arbitrarily complex. Selection of a unstructured mesh generator for these applications involves an assessment of the programme with respect not only to good quality, robustness, time efficiency and cost, but also to specific requirements of the applications.

Researchers have been attempting to obtain a good mesh-quality measure for mesh generation and mesh improvement. Most mesh-quality measures are dependent on geometric characteristics, i.e., the shapes, sizes and orientations of elements. With the availability of the finite-element solution, more accurate finite-element mesh-quality measures have been proposed that are dependent on not only the geometric characteristics but also the physical characteristics of elements. These mesh-quality measures are usually expressed in terms of finite-element solution errors that can be divided into two groups, *a priori* error estimates (solution residuals, condition numbers and closeness to the exact solution as estimated by the convergence curve), and *a posteriori* error estimates. Both

kinds of error estimate can be used to steer an adaptive mesh-refinement process so that a required accuracy of the finite-element solution can be reached.

1.2 Motivation

A number of programmes have been developed by Funnell (2006) in our Auditory Mechanics Laboratory for finite-element modelling and simulation of complex natural structures. These programmes are illustrated in Figure 1.1, where Fie is a programme to segment contours in a stack of images, Tr3 is a programme for triangulating 3-D surface meshes from a series of cross-sectional contours, Tr4 is programme for tetrahedral mesh generation, Fad is a programme for pre-processing finite-element meshes, and Fod is a programme for post-processing meshes. Sap (Bathe *et al.* 1974; Funnell 2006) is a programme for finite-element simulation. It was originally developed at UC Berkeley and has been modified over the years by Dr. WRJ Funnell, including the addition of tetrahedral elements and changes to the handling of shell elements.

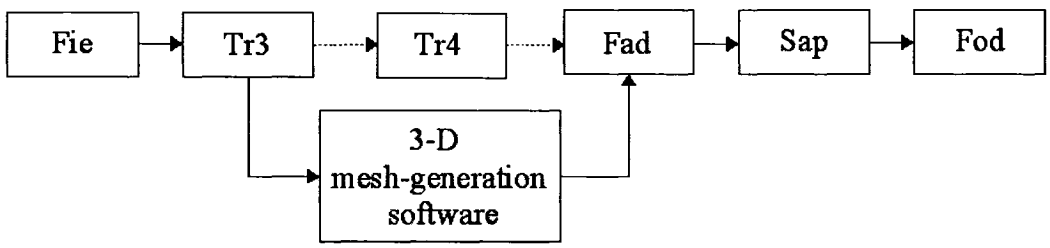


Figure 1.1 Finite-element modeling and simulation pipeline in Auditory Mechanics Laboratory

Tr4 is a bottleneck in this pipeline because it sometimes fails in the mesh-generation process. To address the problem, a former student compared Tr4 with two 3-D mesh-generation programmes, GiD (CIMNE Inc. 2005) and GRUMMP (Ollivier-Gooch 2005). His conclusion was that GiD would be a good choice to replace Tr4 as the 3-D mesh generation software in Figure 1.1 (Siah 2002).

This project continues Siah's investigation of 3-D mesh-generation software by carefully evaluating the finite-element mesh quality with the aim at achieving a reliable and accurate finite-element solution. In addtion to GiD and GRUMMP, Gmsh (Geuzain and Remacle 2005) and TetGen (Hang 2005a) were selected for evaluation. A set of

criteria for assessing mesh quality are proposed for selection of the 3-D mesh generation programme that is most suitable for replacing Tr4 and streamlining the finite-element modelling and simulation pipeline in our lab.

1.3 Thesis outline

Mesh-generation methods and the detailed descriptions of four mesh-generation methods are briefly reviewed in Chapter 2. Mesh-quality measures that are based on the geometric characteristics of elements are discussed in Chapter 3. Mesh-quality measures based on both the geometric characteristics and the physical characteristics of elements are discussed in Chapter 4. The criteria for the evaluation of 3-D mesh-generation software, and the models and methods used for the present evaluation, are described in Chapter 5. The results of the evaluation of the candidate 3-D mesh-generation software are presented and discussed in Chapter 6. Finally, conclusions are summarized and future work is suggested in Chapter 7.

CHAPTER 2

FINITE-ELEMENT MESH GENERATION

2.1 Introduction

A number of comprehensive reviews have been published in the literature about finite-element mesh generation (e.g., Cavendish *et al.* 1985; Boubez *et al.* 1986a,b; Field 1995; Ho-Le 1998; Frey and George 2000; Lo 2002). These reviews covered most mesh-generation methods in two and three dimensions. As the present project focuses on 3-D unstructured mesh-generation methods for complex natural structures, this chapter emphasizes the discussion of four particular mesh-generation methods rather than all of them.

Surface mesh-generation methods are discussed in Section 2.2. Volume mesh-generation methods are discussed in Section 2.3. Conclusions are given in Section 2.4.

2.2 Surface mesh generation

In biomedical fields, methods like computed tomography (CT), magnetic resonance imaging (MRI), ultrasound imaging and histological sections make it possible to obtain planar cross sections of biological objects. The contours (or boundaries) of the structures are created from the planar cross sections by segmentation. Surface meshes are reconstructed from these sets of contours for visualization of objects or for generation of volume meshes for finite-element analysis.

Surface mesh generation is a process that generates boundary triangular meshes for objects of interest by triangulating the contours of adjacent cross sections. The process is composed of generation of the side surface mesh, and triangulation of the top and bottom contours. The most important part of the process is to generate the side surface mesh, i.e., the mesh joining the various cross sections. A number of algorithms for the triangulation of two adjacent sections have been proposed in the literature (e.g., Fuchs *et al.* 1977; Shantz 1981; Ekoule *et al.* 1991; Meyers *et al.* 1992; Chae and Lee 1999).

Overall, surface mesh generation requires a solution to the *correspondence, tiling* and *branching* problems.

The *correspondence* problem involves finding the correct connections between the contours of adjacent sections when there are multiple contours in a section. Soroka (1981) and Meyers *et al.* (1992) approximate the contours by ellipses and then assemble them into cylinders to determine the correspondence, while Wang and Aggarwal (1986) uses the overlapping areas between contours of adjacent sections.

The *tiling* problem involves generating a triangular mesh from the points on contours of adjacent sections. The problem becomes more difficult when two adjacent contours are very different. Optimisation of a metric is a common method to resolve this problem. The metrics proposed include maximum enclosing volume (Keppel 1975), minimum surface area (Fuchs *et al.* 1977; Sloan and Painter 1988), minimum edge-length (Christiansen and Sederberg 1978; Ekoule *et al.* 1991), and minimum triangle narrowness (Funnell 1984).

The *branching* problem arises when an object is represented by a different number of contours in adjacent sections, for example, a blood vessel bifurcating into two branches. Christiansen and Sederberg (1978) and Shantz (1981) proposed to dip down the middle of the bridge to model the saddle point of the branching region. Ekoule *et al.* (1991) proposed to form an intermediate contour between two sections for the case of one-to-many branching. The second method produces less distortion than the first method.

2.3 Volume mesh generation

Most 2-D unstructured mesh-generation methods have successfully been extended to three dimensions with a careful consideration of problems arising in three dimensions. In this section, four 3-D unstructured mesh-generation methods are reviewed. For ease of exposition, examples in two dimensions are given to help understand the similar ideas implemented in three dimensions.

2.3.1 Delaunay-based method

Delaunay (1934) suggested the Delaunay criterion, which simply states that no node is contained within the circumscribed circles of any triangles within the mesh. As a result, it is sometimes called the "empty-circle" criterion in two dimensions. Figure 2.1 illustrates that the vertex P violates the empty-circle criterion because it is located inside the solid-line circle enclosing the triangle ΔABC. The same is true for vertex C because it is located inside the dotted-line circle enclosing the triangle ΔAPB.

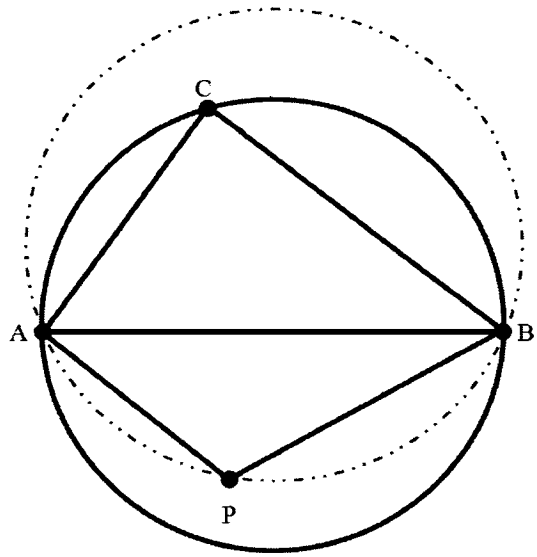


Figure 2.1 Delaunay criterion

It was not until the late 1970's that the Delaunay criterion was utilized to develop algorithms for mesh generation. A number of algorithms have been proposed to implement the Delaunay triangulation, e.g., the flipping algorithm (Lawson 1977), the incremental point-insertion algorithm (Watson 1981; Bowyer 1981), the divide-and-conquer algorithm (Lee and Schachter 1980), and the sweep-line/plane algorithm (Fortune 1987 and O'Rourke 1993). As a typical Delaunay-based method, the incremental point-insertion algorithm is discussed in the next section to show the basic idea of the Delaunay triangulation.

2.3.1.1 The incremental point-insertion algorithm

The general steps of the incremental point-insertion algorithm (Bowyer 1981; Watson 1981) are illustrated in Figure 2.2. The algorithm implemented in three dimensions follows exactly the same steps except that edge, triangle and circumscribed circle are replaced by triangle facet, tetrahedron and circumscribed sphere, respectively.

The steps of this algorithm can be summarized as follows:

- Create a box enclosing the entire domain (Figure 2.2a).
- Insert the boundary nodes to form an initial Delaunay triangulation. Figure 2.2b illustrates the triangulation after inserting one node. Figure 2.2c shows the completed initial triangulation.
- Recover the boundary edges and delete the outside triangles (Figure 2.2d).
- Insert new nodes incrementally inside the initial triangulation (Figure 2.2e). The number of new nodes depends on the desired element size. The Delaunay triangulation process terminates when all new nodes have been inserted.

Three important issues should be considered in the process. They are the Delaunay kernel, the node insertion, and the boundary recovery.

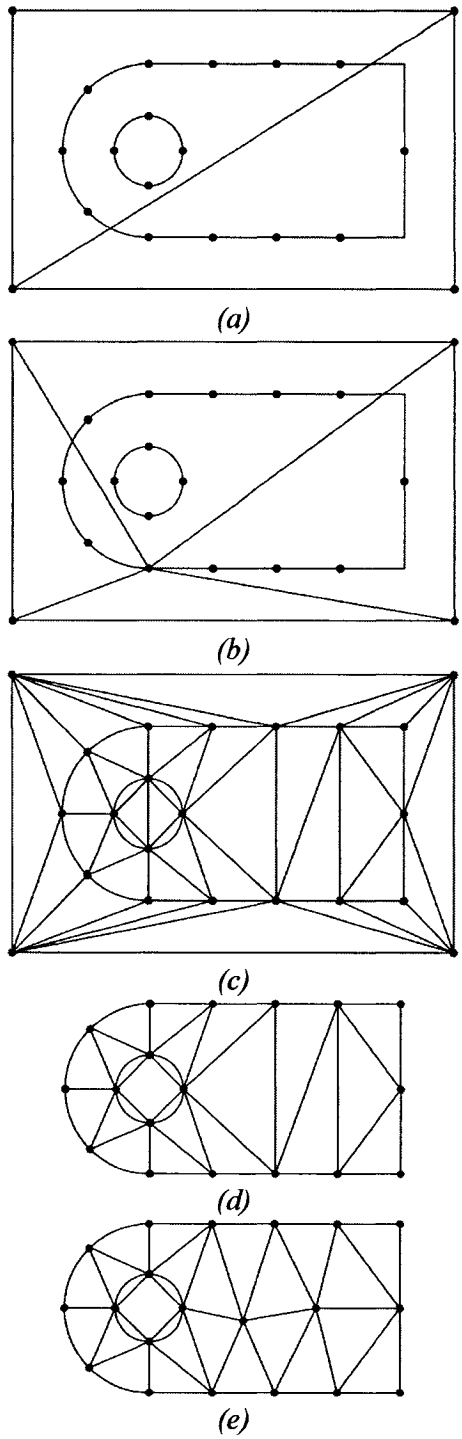


Figure 2.2 The incremental point-insertion algorithm (after Owen 1998)

2.3.1.2 Delaunay kernel

The mesh is re-triangulated locally as each new node is introduced while maintaining the Delaunay criterion. This local triangulation is usually called the Delaunay kernel and it is composed of the following three steps.

- Determine the existing triangles for which the circumscribed circles contain the new node (Figure 2.3a).
- Remove the triangles found in the previous step to form an empty cavity (Figure 2.3b).
- Generate new triangles by linking the new node to vertices of the empty cavity (Figure 2.3c).

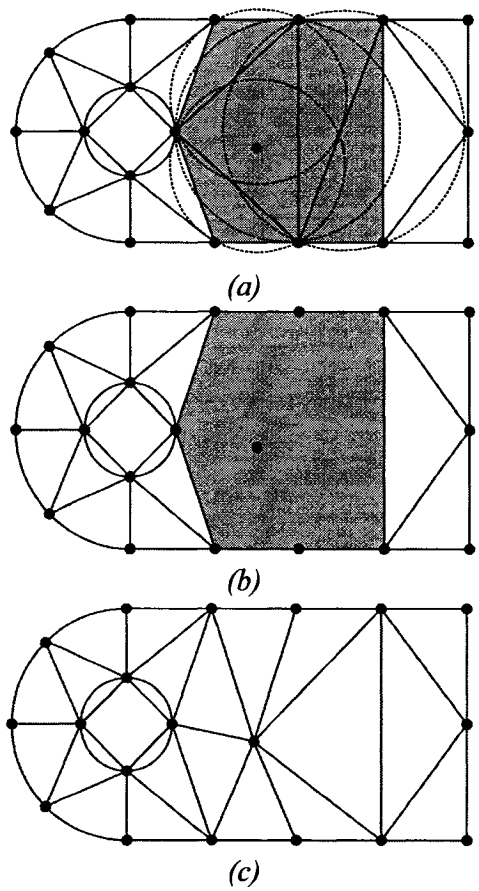


Figure 2.3 Delaunay kernel

2.3.1.3 Node insertion

A number of approaches have been proposed to address where the new nodes are inserted. The first and simplest approach is to define the nodes from a regular grid of nodes covering the domain at a specified nodal density. The second approach is to recursively insert the nodes at the centroids of triangles or tetrahedra provided the underlying sizing function is not violated (Hermeline 1982; Weatherill and Hasson 1994). The third approach is to insert nodes at the centres of the circumscribed circles enclosing the triangle or the circumscribed spheres enclosing the tetrahedron (Homles and Snyder 1988; Chew 1989; Ruppert 1992). The fourth approach is to insert the new nodes along the existing internal edges at a specified spacing (Borouchaki *et al.* 1995; George 1997; Simulog Technologies Inc. 2005). The fifth approach (Marcum and Weatherill 1995) is to determine the positions of the new nodes first using the advancing-front method that is discussed in Section 2.3.2, and then insert the nodes using a Delaunay kernel.

2.3.1.4 Boundary recovery

There is no guarantee that the boundary edges are maintained in the initial Delaunay triangulation of the boundary nodes. Figure 2.4 shows an example in which some of the boundary edges, represented by thick dotted line segments, are missing in the initial triangulation. Therefore, an extra step is required to recover the surface triangulation.

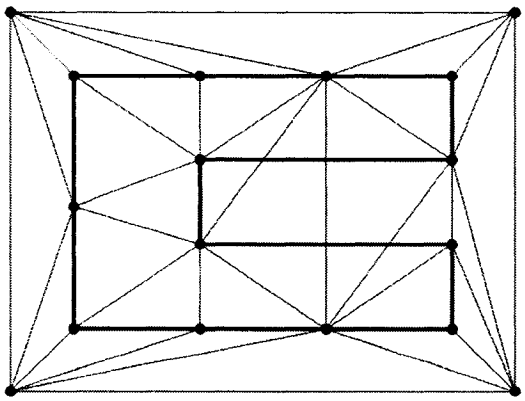


Figure 2.4 Missing boundary edges in the initial triangulation (after Owen 1998)

The missing boundary edges can be recovered by iteratively swapping triangle edges as illustrated in Figure 2.5(a) to (d) where two thick line segments are the missing

edges finally recovered. However, the process becomes more complex in three dimensions since there is no guarantee that the boundary surface mesh can be recovered by just swapping the related edges in the boundary surface mesh. An additional recovery of the surface triangular facets is required through tetrahedral transformations (Weatherill 1996). After recovering the boundary entities, such a mesh is no longer in conformance with the Delaunay criterion.

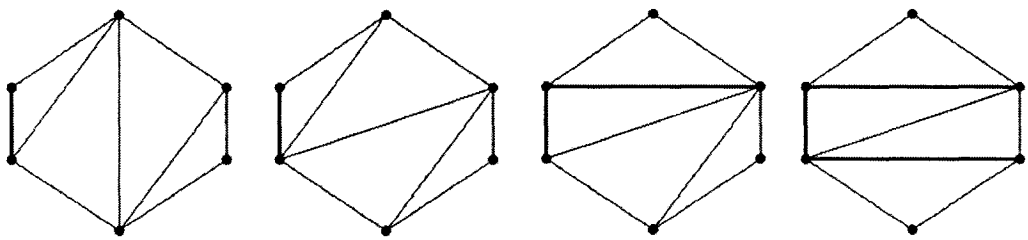


Figure 2.5 Boundary edges (thick line segments) are recovered by diagonal swaps

2.3.1.5 Discussion

The Delaunay-based method is an efficient mesh-generation method because it is possible to construct several elements when each new node is inserted. However, the Delaunay triangulation may give rise to slivers as shown in Figure 2.6 where three vertices of the tetrahedron T_{ABCD} are located on the horizontal circle and the fourth vertex D is located a very small distance Δd above the circle. Having very small volumes, the slivers are not good for finite-element analysis and should be avoided.

As boundary recovery is necessary in the Delaunay-based triangulation, the mesh may not satisfy the Delaunay criterion everywhere.

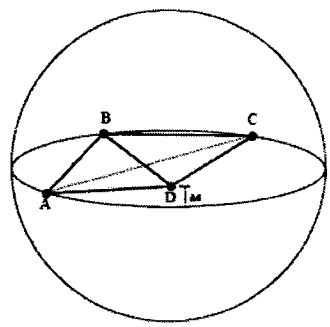


Figure 2.6 Sliver

2.3.2 Advancing-front method

The advancing-front method was first investigated in two dimensions by George (1971), and then extended to three dimensions by Lo (1985). This method is a very powerful unstructured mesh-generation method for triangulating a domain of arbitrary shape, and it is a strong competitor for the Delaunay-based method.

2.3.2.1 Process of advancing-front method

The advancing-front method starts from the boundary entities, i.e., the initial fronts. It advances each time an element is constructed, and updates the fronts and elements continuously throughout the process. Typically, the process is summarized as the following three major steps:

1. Front initialization: The domain is discretized into the boundary edges as shown in Figure 2.7(a). These edges are the initial fronts and are stored in a front list.
2. Element formulation: This step involves the selection of the active front and the selection of the best new point. The discussion here is partial and more details are given in the next sections.
   - A front is selected as the active front to start the triangulation.
   - The method for selection of the best new point for the associated active front is to form a triangle satisfying the desired shape and size criterion, as shown in Figure 2.7(b). The point may be an existing point K in the current mesh if it falls within the circle of which the best point is the centre and the radius satisfies the size criterion, as shown in Figure 2.7(c). The new triangle, as shown in Figure 2.7(d), is formed by the point and the active front. The new triangle is accepted if it does not intersect with any other triangles.
3. Front updating: The new triangle is then added to the element list. The current active front is removed from the front list. The front list is updated on the basis of whether the new triangle introduced new edges.

Steps 2 and 3 are repeated and the whole meshing process terminates when the front list is empty. The resulting mesh is illustrated in Figure 2.7(e).

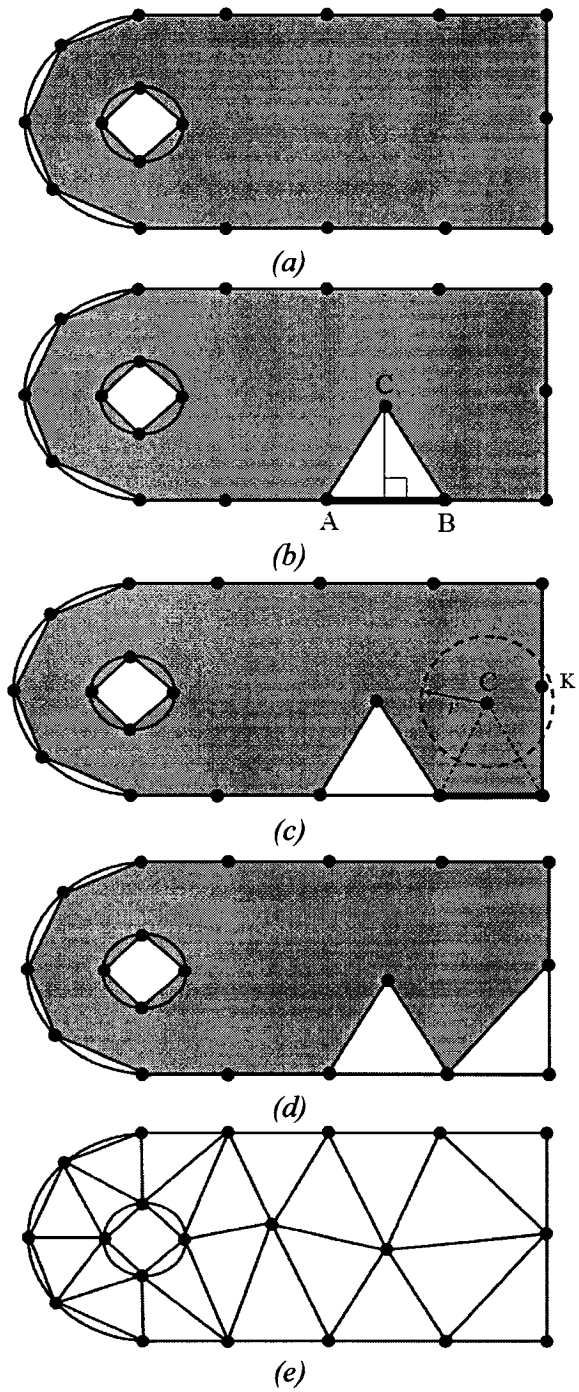


Figure 2.7 The advancing-front method (after Owen 1998)

2.3.2.2 Selection of an active front

The mesh elements are created based on the front entities. The objective of the selection

of an active front is to generate a good mesh without a possible failure in mesh generation. The operation is not a purely local process but involves anticipation of the front evolution. All fronts are taken into consideration and put in order before the selection of the active front. Lohner (1996) suggested that the selection should lead to the smallest new element.

2.3.2.3 Selection of the best point

The selection of the best point aims to form an element with high quality, with the associated active front. The selection should also satisfy the local element size requirement (Peraire *et al.* 1992; Jin and Tanner 1993). In Figure 2.8, the best point P0 is on normal line passing though the centroid of the active front to form an equilateral triangle. An existing point can replace the best point if it is inside the circle with its centre at P0 and with a radius of r. The radius depends on the desired element size. Three points, P1, P2 and P3 shown in Figure 2.8, on the normal line inside the circle are stored in a stack in case no best point can be found later during the mesh-generation process. The programme will revert back to this stage and the next candidate for the best point will be selected to continue the triangulation.

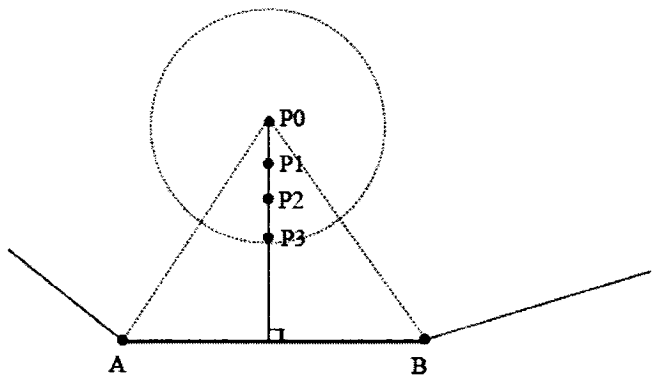


Figure 2.8 The best points associated with a front AB

2.3.2.4 Discussion

The major advantage of the advancing-front method is that the domain boundary always

remains the same throughout the mesh-generation process. However, a successful generation is not guaranteed for any arbitrary domain. A typical case of failure is illustrated in Figure 2.9 in which the large elements cross over the small elements when two very dissimilar fronts are merging together. In this case, the point C is the best point for the small front AB while the existing point G is the best point for the large front ED. Unfortunately, in this example, the new triangles ΔABC and ΔEDG cannot be accepted because they both intersect with current triangles in the mesh.

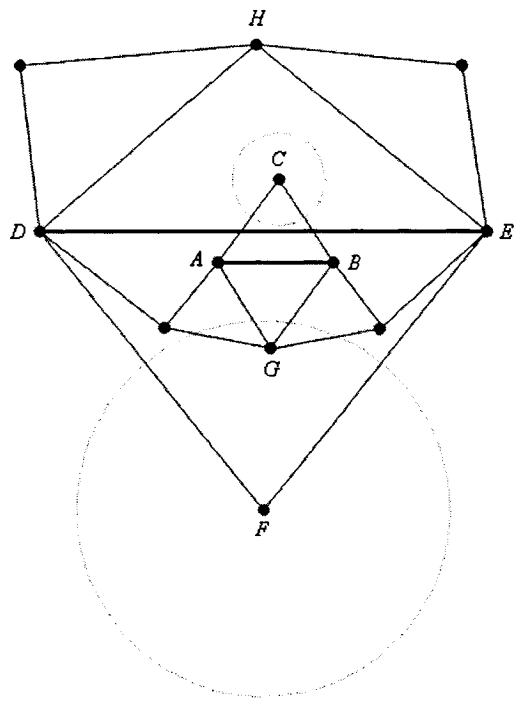


Figure 2.9 Merging two very dissimilar fronts

2.3.3 Quadtree/octree-based method

The quadtree/octree-based mesh-generation method has been a research topic in the past two decades since Yerry and Shephard (1983 & 1984) initiated the application of quadtree/octree encoding to mesh generation. Quadtree encoding refers to tree data structures that identify the nested decompositions of a quadrilateral into four quadrants in two dimensions. A hexahedron is decomposed into eight octants for octree encoding in three dimensions. This method generates quadrilaterals in the interior of the boundary

domain and triangles near to the boundary. Those quadrilaterals could be further broken down into triangles so that all elements in the mesh are triangular elements for finite-element analysis (Shephard and Georges 1991).

2.3.3.1 Process of quadtree-based method

A typical process of the quadtree-based method is composed of the following three steps:

- Create a bounding box that encloses the object domain to be meshed (Figure 2.10a).
- Construct the tree structure by recursively sub-dividing the box into quadrants depending on the intersections between quadrant boundaries and object boundary entities (Figure 2.10b).
  - If the quadrant lies entirely outside the domain, then it is rejected.
  - If the quadrant lies entirely inside the domain, then it requires no further subdivision.
  - If the quadrant lies partially inside and partially outside the domain, then it may or may not require a further subdivision depending on whether the user-defined refinement level has been reached.
- Create the triangular mesh by dividing the quadrants (Figure 2.10c).

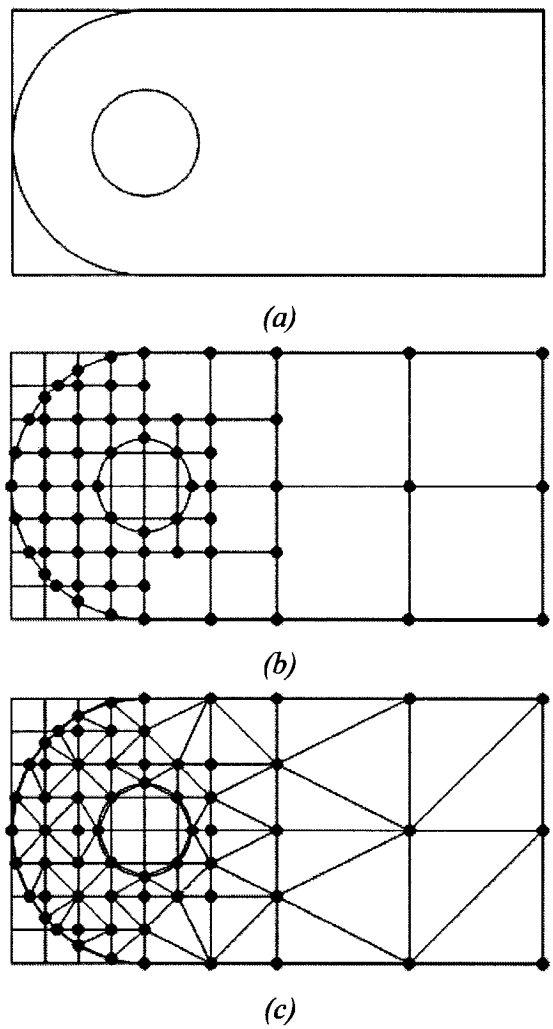


Figure 2.10 The quadtree-based method (after Owen 1998)

2.3.3.2 Discussion

The quadtree/octree-based method is able to guarantee finite-element meshes with well-shaped elements and to generate graded finite-element meshes for objects. However, a small number of very large elements may be needed for the interior while a much larger number of much smaller elements is required for a satisfactory geometric representation of the boundary. As a result, the transition between the neighbouring quadrants may not be smooth because their side lengths are very different, and the resulting mesh is not conforming.

2.3.4 Coring method

Thacker (1980) first proposed this idea for mesh generation in two dimensions. Boubez *et al.* (1986a,b) and Funnell & Funnell (1988) in our lab extended this idea to three dimensions by developing Tr4 (Funnell 2006), a volume mesh generator.

In this method, a grid of regular prisms is constructed in the interior of the domain. Each prism in the core is divided into a number of tetrahedral elements with good quality, leaving a relatively small region around the boundary which is to be discretized into tetrahedral elements. As a result, the quality of the tetrahedral elements near the boundary will be affected by the irregularity of the domain.

This 3-D mesh-generation method is specifically designed to work from sets of boundary contours defined on 2-D cross-sections. The cross-sectional surfaces are each triangulated on a regular grid of nodes, using the same grid for all the sections. Grid points inside each contour are joined to form a set of equilateral triangles. The nodes on the external boundary of this internal mesh are joined to the nodes on the contour boundary to form a triangulation. Adjacent sections contain identical nodes and triangles within their overlapping areas.

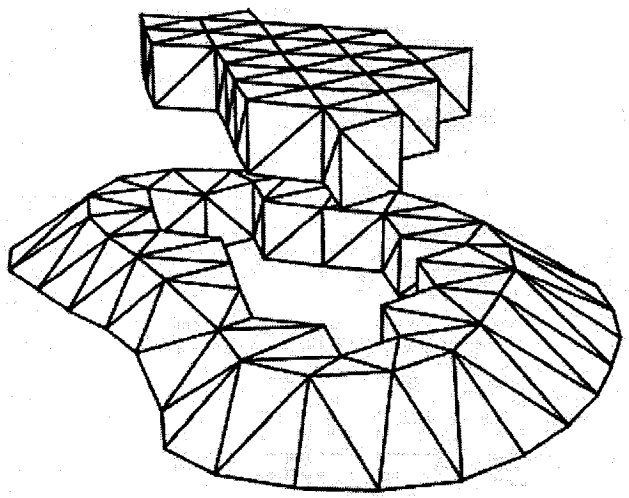


Figure 2.11 The core mesh and the ring mesh of a slice (from Boubez et al. 1986b)

As shown in Figure 2.11, a central core of pentahedral prisms is generated by

pairing up the matching triangles from the upper and lower triangulations on the two adjacent sections. Each of the prisms is then shredded into three tetrahedra. The remaining portion between the two adjacent sections, a torus-like structure surrounding the core, is assembled into a polyhedron represented by lists of related triangular faces, edges and vertices, and then meshed using four topological operators (Boubez *et al.* 1986b). Each pair of adjacent sections is processed in a similar way. All of them are assembled to form the complete three-dimensional mesh. Finally, the internal nodes are iteratively relaxed to improve the quality of the tetrahedral elements near the boundary.

This method generates a core mesh with high quality. Furthermore, the boundary points and their triangulation are preserved, thus retaining the slice structure. However, the mesh quality of the volume mesh near the boundary will be affected by the irregular shape of the domain. Badly shaped triangular elements may often be generated on the side surface mesh when two adjacent contours are very different. Moreover, the meshing of the ring may fail in the special case of a Schonhardt polyhedron, as illustrated in Figure 2.12. There is no way to subdivide this polyhedron into tetrahedra without creating new vertices.

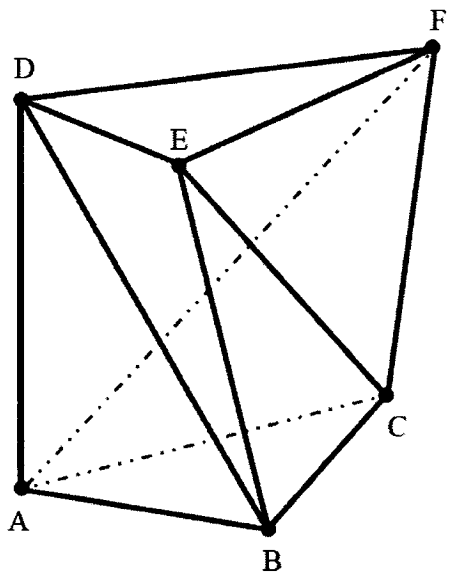


Figure 2.12 Schonhardt polyhedron

2.4 Conclusions

The surface mesh-generation methods used in biomedical applications are described. Three problems (correspondence, tiling and branching) encountered in these methods and their respective remedies are introduced.

Four unstructured mesh-generation methods are discussed. The Delaunay-based method has gained increasing attention because of its supposed robustness and efficiency. Some of the boundary entities may be missing and thus additional procedures are required to recover them. Furthermore, slivers are often generated and they need to be removed by topological modification. The advancing-front method maintains the domain boundary during mesh generation, which is useful for parallel mesh generation, but the mesh generation may fail since the algorithm may not be able to determine an acceptable next point during the process. The quadtree/octree-based method is able to generate good meshes but may introduce a very non-uniform mesh to approximate the object domain when the boundary is an arbitrary shape with high curvature. The coring method generates a core mesh with high quality and retains the slice structure, but may fail in mesh generation of the ring.

Combining two different mesh-generation approaches offers a possible better solution for mesh generation. The combined approach can effectively alleviate or eliminate the drawbacks in each method. For example, in the advancing-front-Delaunay approach (Mavriplis 1992 and Frey *et al.* 1998), the advancing-front method is used to determine the best points with respect to the active front entity while the Delaunay-based method is used to connect vertices. The approach helps not only to reduce the possibility of failure by the advancing-front method, but also to reduce the possibility of generation of slivers by the Delaunay-based method while increasing the efficiency because the Delaunay-based insertion of a single new node results in the creation of several new elements. Similarly, combining the Delaunay-based method and the quadtree/octree-based method or combining the advancing-front method and the quadtree/octree-based method are also possible alternatives to take advantage of the strong points of each method.

CHAPTER 3

GEOMETRIC MESH-QUALITY MEASURES

3.1 Introduction

As the finite-element method receives increasing attention, whether a mesh is good for finite-element analysis is an important topic. One of the main problems in finite-element generation is how to generate well-shaped elements since badly shaped elements often result in poor numerical performance.

Mesh-quality measures are used to evaluate the quality of individual elements in finite-element meshes. These measures can be grouped into two categories depending on how they are derived: geometric mesh-quality measures that are computed purely on the basis of the geometric characteristics of elements, and finite-element mesh-quality measures that are calculated on the basis of information from the finite-element solution.

Geometric mesh-quality measures may characterize the shape and size of an element. These measures consist of two types depending on their derivations. One type, geometrically based, uses direct geometric characteristics of an element. The other type, Jacobian-based, uses the element Jacobian matrices relating the reference, regular and physical spaces. The geometric mesh-quality measures are discussed in this chapter, and the finite-element mesh-quality measures will be discussed in the next chapter.

In this chapter, badly shaped tetrahedral elements are classified in Section 3.2. The geometrically based shape-quality measures are discussed in Section 3.3. The Jacobian-based shape-quality measures are discussed in Section 3.4. The results of previously published comparisons among the shape-quality measures are presented in Section 3.5. Conclusions are given in Section 3.6.

3.2 What are badly shaped tetrahedral elements?

As tetrahedral mesh generators will be investigated in this project, only tetrahedral elements will be discussed in the rest of this chapter. Badly shaped tetrahedral elements can be characterized by the fact that their volumes are nearly zero. These elements may

have an infinite number of shapes, which can be roughly classified into two categories depending on whether the four vertices of the tetrahedron are close to a line, or close to a plane but not to a line (Cheng *et al.* 1999; Freitag and Knupp 2002) .

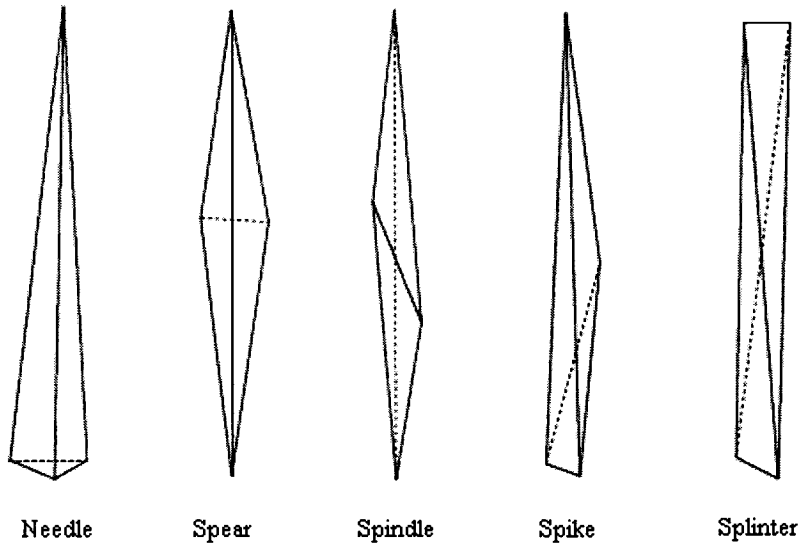


Figure 3.1 Badly shaped narrow tetrahedral elements

Figure 3.1 shows typical badly shaped narrow tetrahedral elements with their four vertices close to a line. This category is further classified by Cheng *et al.* (1998) into five subcategories that are characterized by the relative positions of the four vertices along the line. A *needle* is a tetrahedron with three vertices close to one end of the line and the fourth vertex as the other end of the line. A *spear* is a tetrahedron with two vertices defining the line and the other two vertices near the mid-point of the line. A *spindle* is a tetrahedron with four vertices roughly evenly spaced along the line. A *spike* is a tetrahedron with two vertices close to one end of the line, the third vertex as the other end of the line, and the fourth vertex close to the mid-point of the line. A *splinter* is a tetrahedron with two vertices close to one end of the line and the other two close to the other end of the line.

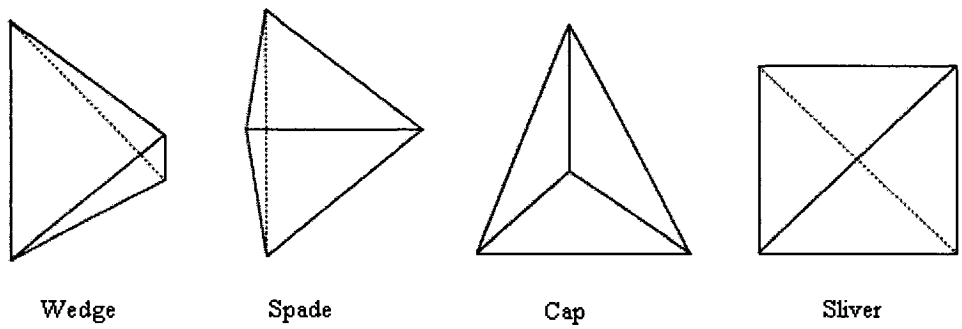


Figure 3.2 Badly shaped flat tetrahedral elements

Figure 3.2 illustrates typical badly shaped flat tetrahedral elements with their four vertices close to a plane but not to a line. Similar to the first category, this category is further classified by Cheng *et al.* (1998) into four subcategories depending on how the four vertices are located with respect to the plane. The further classification is actually characterized by the position of the fourth vertex. If the fourth vertex is close to one of the first three vertices, the tetrahedron is a *wedge*. If the fourth vertex is close to the mid-point of one of the three edges of the triangle, the tetrahedron is a *spade*. If the fourth vertex is close to the geometric centre of the triangle, the tetrahedron is a *cap*. If the fourth vertex and the other three vertices are roughly equally spaced around a circle, but it is raised slightly above the circle, then the tetrahedron is a *sliver*.

3.3 Geometrically based shape quality measures

These mesh-quality measures are directly derived from geometry and they are defined to evaluate the shape of an element, thus they are also called shape-quality measures or simply shape measures.

3.3.1 Attributes for a good shape measure

Field (2000) suggested that a good shape measure for elements should possess the following attributes:

- Ability to detect all possible badly shaped elements
- Dimensionlessness, that is, independence of element size
- Normalization by an optimal value within a range [0,1], where 1 is for an

equilateral tetrahedron and 0 is for a degenerate tetrahedron

- Boundedness, that is, no arbitrarily large value is produced
- Invariance under translation, rotation and uniform scaling

3.3.2 Geometrically based shape measures for tetrahedral elements

A number of shape measures have been proposed to characterize the shape of a tetrahedron. Nguyen (1982) and Van Oosterom and Strackee (1983) proposed the minimum solid angle in a tetrahedron as the shape measure. In Figure 3.3, a solid angle with unit of steradians is represented for the tetrahedron T_{ABCP} by the spherical triangle *abc* of the unit sphere centred at vertex P and bounded by the three triangular faces sharing the vertex P. Cavendish *et al.* (1985) suggested that a tetrahedron can be characterized by the ratio of the circumscribed sphere radius to the inscribed sphere radius. Baker (1989) proposed the combined use of the ratios of the maximum edge length to the inscribed sphere radius, and of the maximum edge length to the minimum edge length. Cougny *et al.* (1990) utilized the tetrahedron volume and the four triangular facet areas to define a dimensionless normalized aspect ratio. Dannelongue and Tanguy (1991) used the ratio of the volume and the average edge length of the six edges of a tetrahedron. They also suggested an alternative by replacing the average edge length with the root mean square of the six edge lengths.

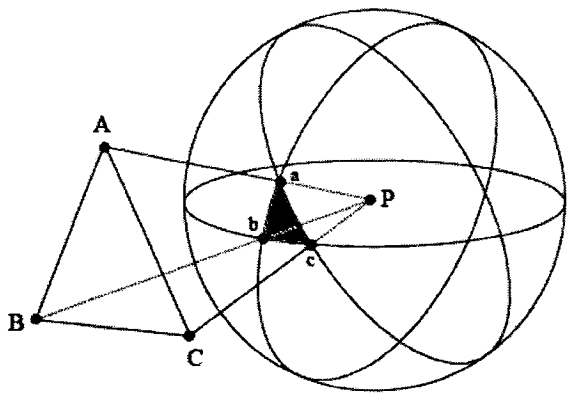


Figure 3.3 Solid angle represented by the shaded area abc in a tetrahedron

Freitag and Ollivier-Gooch (1996) proposed five shape measures based on dihedral angles. A dihedral angle is defined as an angle between two planes, in this case, an angle $\measuredangle EDB$ between the two triangular facets ΔABC and ΔAEC that are illustrated in Figure 3.4. Their shape measures include the maximum dihedral angle, the minimum dihedral angle, the maximum cosine of the dihedral angles, the minimum cosine of the dihedral angles and the minimum sine of the dihedral angles. The last three shape measures are actually trigonometric functions of the first two measures. As a result, only the first two shape measures are listed in Table 3.2.

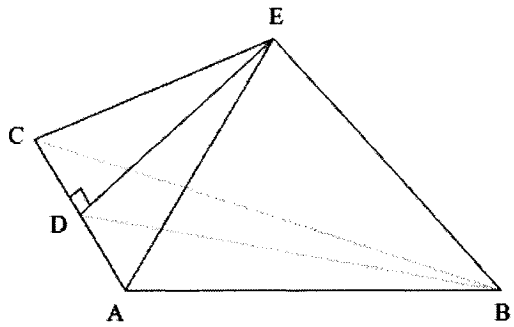


Figure 3.4 Dihedral angle in a tetrahedron

3.4 Jacobian-based shape-quality measures

The Jacobian matrix defined for finite elements can be factored into geometrically meaningful parts including the shape, size and orientation of an element (Knupp 1999 & 2000 & 2001). The measures based on the element Jacobian matrices are able to characterize not only the shape but also the other geometric properties of an element.

In this section, only shape measures are discussed. The element Jacobian matrices are defined in Section 3.4.1. The use of the Jacobian matrices to construct shape measures is discussed in Section 3.4.2.

3.4.1 Element Jacobian matrix

The element matrix is the affine transformation associated with a triangle or a tetrahedron. Taking a triangle, for example, let $t_k \in R^2$, $k=0,1,2$, represent the

coordinates of the three vertices of the triangle in physical space. Let ξ_k represent the coordinates of the four vertices in reference space, where $0 \le \xi_k \le 1$ with $\xi_0+\xi_1+\xi_2=1$. The affine transformation from reference space to physical space is defined by

$$t(\xi_i)=\sum_{k\neq i} \xi_k t_k \tag{3.1}$$

where $k, i=0,1,2$.

By expansion and the fact that $\sum_{k=0}^{2} \xi_k=1$, Equation 3.1 can be explicitly written as

$$t=(1-\xi_1-\xi_2)t_0+\xi_1 t_1+\xi_2 t_2$$

giving

$$t=A_0 u_0+t_0 \tag{3.2}$$

In Equation 3.2, $t=(x, y)^T$ and $u_0=(\xi_1, \xi_2)^T$ represent vertex coordinates, $t_0=(x_0, y_0)$ is a translation vector, and A_0 is a 2-by-2 matrix representing the affine transformation that is referenced to vertex t_0 and is written as

$$A_0=\begin{pmatrix} x_1-x_0 & x_2-x_0 \\ y_1-y_0 & y_2-y_0 \end{pmatrix} \tag{3.3}$$

As a triangle has three vertices, there are three such Jacobian matrices A_k for a triangular element. Similarly, there are four 3-by-3 Jacobian matrices A_k for a tetrahedral element with $k=0,1,2,3$.

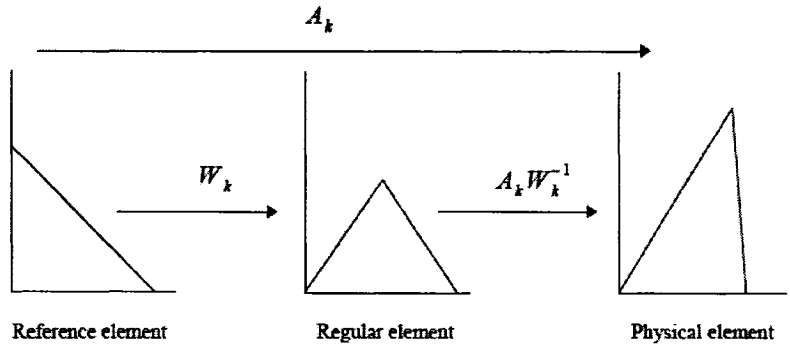


Figure 3.5 Relationships of Jacobian matrices among the reference, regular and physical triangles

Figure 3.5 illustrates the relationships of the element Jacobian matrices among the reference, regular and physical triangles. The three triangles are related via three 2-by-2 matrices W_k, M_k and A_k, where $k=0,1,2$ is the local vertex index in a triangle and the W_k are the 2-by-2 Jacobian matrices of the regular element. The Jacobian matrix A_k is invariant under translation, rotation, scaling, and combinations of scaling and rotation, but it is not invariant to *k*. As a result, the mesh-quality measures based on A_k will vary with *k*. To address the problem, Freitag and Ollivier-Gooch (1996) suggested and proved that the weighted Jacobian matrices $M_k=A_k W_k^{-1}$ are independent of *k*, where the M_k are the linear transformations between the regular element and the physical element. The M_k are dimensionless because W_k and A_k both have units of length. Therefore, the M_k, or simply M, are used to define nodally invariant element-quality measures.

3.4.2 Jacobian-based shape measures for tetrahedral elements

Liu and Joe (1994b) proposed the use of the mean ratio η, which is based on the weighted Jacobian matrix *M*, to characterize the quality of a tetrahedron. The mean ratio η of a tetrahedron in physical space is defined as the ratio of the geometric mean to the algebraic mean of the three eigenvalues $\lambda_1, \lambda_2, \lambda_3$ of the 3-by-3 matrix $M^T M$. As we know, the geometric mean is less than or equal to the algebraic mean for positive numbers. Liu and Joe (1994a) proved that η is a shape measure based on the edge lengths of a tetrahedron:

$$\eta = \frac{3\sqrt[3]{\lambda_1 \lambda_2 \lambda_3}}{(\lambda_1 + \lambda_2 + \lambda_3)}$$
$$= \frac{12\sqrt[3]{9V^2}}{\sum_{0 \le i \le 5} l_i^2} \tag{3.5}$$

Freitag and Knupp (1999) suggested that $c = 3/K(M)$ is capable of characterizing the element quality, where $K(M) = \|M\| * \|M^{-1}\|$ is the condition number of the weighted Jacobian matrix $M = A W^{-1}$. Similar to η, c is a shape measure for a tetrahedral element.

Both η and c possess all the attributes of a good shape measure except that they are not invariant under uniform scaling. As a result, the sizes of elements would be taken into consideration when assessing the shape qualities of elements in the meshes.

3.5 Comparison among shape measures

The shape measures for tetrahedral elements discussed in the previous sections are summarized in Table 3.2. To assist in understanding their mathematical interpretation, the notation is first given in Table 3.1. Most symbols utilized here are the same as those used by Parthasarathy (1993) and Lewis *et al.* (1996).

In Table 3.2, "Optimal value" is the shape measure of an equilateral tetrahedron, and "Bounds" specifies the range normalized by the optimal value.

Table 3.1 Notation used in Table 3.2

CR	Circumscribed sphere radius
IR	Inscribed sphere radius
L	Length of an edge
H	Height with one triangle as the base
S	Surface area of a triangular facet
V	Volume
δ	Dihedral angle
θ	Solid angle

Table 3.2 Shape measures for a tetrahedral element

Expression	Unit	Invariance to uniform scaling	Optimal value	Bounds
$\rho = \frac{IR}{CR}$	No	Yes	$\frac{1}{3.0}$	[0,1]
$\sigma = \frac{IR}{L_{max}}$	No	Yes	0.204124145	[0,1]
$\omega = \frac{L_{max}}{CR}$	No	Yes	1.632993162	[0,1]
$\zeta = \frac{IR}{H_{max}}$	No	Yes	4.0	[0,1]
$\tau = \frac{L_{min}}{L_{max}}$	No	Yes	1.0	[0,1]
$\beta = \frac{V^4}{\left[\sum_{0 \le i \le 3} S_i^2\right]^3}$	No	Yes	$4.572473708 \times 10^{-4}$	[0,1]
$\alpha = \frac{V}{L_{avg}^3}$	No	Yes	0.11785113	[0,1]
$\gamma = \frac{V}{L_{rms}^3}$	No	Yes	0.11785113	[0,1]
δ_{max}	Degree	Yes	70.52877937	[1,2.552149656]
δ_{min}	Degree	Yes	70.52877937	[0,1]
θ_{min}	Steradian	Yes	0.5512856	[0, 1]
$\eta = \frac{12(3V)^{2/3}}{\sum_{0 \le i \le 5} L_i^2}$	No	No	1.0	[0,1]
$c = \frac{3}{K(M)}$	No	No	1.0	[0,1]

Parthasarathy (1993) compared the first seven shape measures listed in Table 3.2 by carrying out four sets of sensitivity tests. The tests were designed to simulate a change from an equilateral tetrahedron to a badly shaped tetrahedron corresponding to four possible badly shaped tetrahedra: needle, wedge, cap and sliver. The α, ρ, σ, γ and β were able to characterize distortions in all four cases. The τ could not detect badly shaped tetrahedra without short edges, such as slivers. The ω was limited to detecting slivers. Finally, Parthasarathy (1993) recommended the use of γ because of its low computational cost.

Liu and Joe (1994b) established a relationship among θ_{min}, ρ and η. In short, for any two different measures μ and ν ($0 \le \mu, \nu \le 1$) among the three measures, the relationship between μ and ν can be represented by a form $c_0 \mu^{e_0} \le \nu \le c_1 \mu^{e_1}$ where c_0, c_1, e_0 and e_1 are positive constants. The relationship implies that if one measure approaches zero for a badly shaped tetrahedron, so do the others. They concluded that all three measures are equivalent in this sense but that they do not approach 0 or 1 at the same rates for tetrahedra with different shapes. Any one of the three measures can approach 0 faster than the others for different badly shaped tetrahedra. They are, however, in a fixed order that is $\theta_{min} < \rho < \eta$ when they are close to 1 for a regular tetrahedron. ρ is more uniformly distributed in the interval of $[0,1]$ than the others.

Dompierre *et al.* (1998) compared θ_{min}, δ_{min}, ρ, η, τ and σ using unit tetrahedron, cube and sphere. They obtained the order $\theta_{min} < \tau < \delta_{min} < \rho < \eta$ for the average of the above shape measures in a mesh. This order is consistent with the order obtained by Liu and Joe (1994b).

Freitag and Knupp (2002) compared c with the two shape measures δ_{min} and γ by doing experiments on a series of badly shaped tetrahedra. They concluded that both γ and c are able to efficiently detect all nine types of badly shaped tetrahedra. They found, however, that δ_{min} is unable to detect needles, spears or spindles, and that it is also computationally expensive.

3.6 Conclusions

A mesh-quality measure is used for evaluating geometric properties, in most cases the shape of an element. It may be derived from the geometry of the element, or from the element Jacobian matrices.

The mesh-quality measures discussed in this chapter are used for tetrahedral elements. Many geometrically based shape measures are available to characterize the shape qualities of tetrahedral elements. The measures α, ρ, σ, γ, β, θ_{min}, η and c are able to detect some of or all of the nine types of badly shaped tetrahedral elements, but they do not approach 0 or 1 at the same rates for the different types.

Jacobian-based mesh-quality measures are based on element Jacobian matrices, W, M and A in the reference, regular and physical spaces, respectively. The weighted Jacobian matrix $M=AW^{-1}$ is used to construct the shape measures η and c. They satisfy all the requirements of a good mesh-quality measure except that they are not invariant under uniform scaling.

Apart from the shape of an element, the element Jacobian matrices also contain the other geometrical information, i.e., the size and orientation of the element. Therefore, the Jacobian-based mesh-quality measures are able to quantify these geometrical characteristics, which could provide comprehensive geometrical information about elements for the purpose of finite-element analysis.

The shape measures usually do a good job in identifying badly shaped tetrahedra in meshes. However, the appropriateness of the meshes for finite-element analysis may not be decided by the shape measures alone. Since the sizes of elements play an important role in characterizing finite-element meshes (Shewchuk 2002), a better geometric mesh-quality measure should be able to assess both the shape qualities and the sizes of elements.

CHAPTER 4

FINITE-ELEMENT MESH-QUALITY MEASURES

4.1 Introduction

The objective of the finite-element method is to seek a reliable approximate solution. The reliability of the approximate solution has to be judged by the difference between the exact solution and the approximate solution. As the exact solution is rarely known in practical problems, the assessment of the reliability of the approximate solution becomes one of the most difficult aspects of finite-element analysis (Babuška and Strouboulis 2001). However, it is possible to construct quantitative estimates for the solution error and to determine the rate of change of the error as the number of degrees of freedom in the finite-element model increases. A good mesh usually results in a small solution error, for this reason, error estimates are regarded as finite-element mesh-quality measures in this chapter.

A considerable amount of research has been devoted in the past two decades to the development of reliable error estimates and feedback procedures by which the required accuracy of a finite-element solution can be reached at a reasonable computational cost (e.g., Babuška and Rheinboldt 1978a,b; Kelly *et al.* 1983; Oden *et al.* 1986; Zienkiewicz and Zhu 1987; Ewing 1990; Verfurth 1994; Ainsworth and Oden 2000; Gratsch and Bathe 2005). Researchers often refer to two types of error estimates, *a priori* error estimates and *a posteriori* error estimates. It is unclear what is used to explicitly differentiate between *a priori* and *a posteriori*. In this project, *a priori* error estimates are computed on the basis of the system stiffness matrices or the global solution, and they include solution residuals, condition numbers and convergence rate estimates. In contrast, *a posteriori* error estimates are calculated on the basis of solutions for individual elements. Both types of error estimate are able to provide more accurate estimates of mesh quality than geometric mesh-quality measures do, and to steer mesh refinement.

In this chapter, sources of error in finite-element solutions are summarized in

Section 4.2. Basic requirements for error estimators are described in Section 4.3. Section 4.4 introduces error norms, a model problem as a basis for explanations in later discussions, and error measures. *A priori* error estimators and *a posteriori* error estimators are discussed in Sections 4.5 and 4.6, respectively. Mesh refinement strategies based on error estimates are introduced in Section 4.7. Finally, conclusions are given in Section 4.8.

4.2 Sources of error

There are a number of sources of error in finite-element solutions. They are generally classified into five groups (Noor and Babuška 1987; Tarnhuvud *et al.* 1990):

- Reliability of the mathematical model that is used for describing a physical model being analysed
- Uncertainties in the parameters (geometry, material properties, boundary conditions and load conditions) describing the mathematical model
- Discretization errors caused by the numerical discretization of the continuous mathematical model, which are greatly influenced by the types, shapes and sizes of elements in the mesh
- Interpolation errors that depend on the polynomial order of the shape function, that is, the orders of elements in the mesh
- Round-off errors and truncation errors that occur in numerical computations on computers with finite precision

In fact, the discretization errors and the interpolation errors are both directly influenced by the element characteristics in the mesh. Thus, they are closely interrelated in the finite-element method. In most cases, the term 'discretization error' is used to refer to the difference between the exact solution and the approximate solution. The interpolation error is then implicitly assumed to be one source of the discretization errors.

4.3 Basic requirements for an error estimator

The purpose of a finite-element error estimator is to provide an estimate for the solution error. Such an error estimator should possess the following characteristics (Gratsch and

Bathe 2005):

- be accurate, that is, the predicted error should be close to the exact error, which is however generally unknown
- ensure that the error estimate asymptotically approaches zero at the same rate as the exact error does when the number of degrees of freedom increases
- be able to yield guaranteed and sharp lower and upper bounds for the exact error
- be computationally inexpensive
- be robust, that is, it can be applied in a wide range of applications
- be able to steer mesh refinement so as to optimize the mesh with respect to the required accuracy

It is difficult to find an error estimator that meets all these requirements. This is mainly caused by either a very high computational cost or a lack of guaranteed bounds for the errors in practical problems.

4.4 Error norms and error measures

In this section, a model problem and its application in solid mechanics are described in Section 4.4.1. The concepts of error norms are introduced in Section 4.4.2. Error measures are presented in Section 4.4.3.

4.4.1 Model problem

To illustrate the error estimators, let us consider an elliptic problem in a domain Ω with a boundary condition Γ, which is partly Dirichlet and partly Neumann (Zienkiewicz and Taylor 2000). The problem is given by

$$\boldsymbol{L}u = f \tag{4.1}$$

where $\boldsymbol{L}$ is a linear differential operator, u is the exact solution and f is the source function.

Let $\hat{u}$ be the approximate solution. The finite-element solution error can then be

written as

$$e = u - \hat{u} \tag{4.2}$$

In solid mechanics, e, u and $\hat{u}$ may refer to displacement, stress or strain. To derive the operator $\boldsymbol{L}$, we need to use the relation between stress and strain as given by

$$\varepsilon = S u \tag{4.3}$$

and Hooke's law as given by

$$\sigma = D \varepsilon \tag{4.4}$$

where ε and σ are strain and stress, respectively.

The operator $\boldsymbol{L}$, derived by Zienkiewicz and Zhu (1987), takes the form

$$\boldsymbol{L} = S^T D S \tag{4.5}$$

4.4.2 Error norms

Error norms are introduced to quantitatively measure the overall magnitude of errors in finite-element solutions. There are three measures commonly used in the finite-element method: the energy norm $\|e\|_E$ or L_1 norm, the mean-square norm $\|e\|_0$ or L_2 norm, and the maximum norm $\|e\|_\infty$ or L_∞ norm.

The energy norm or L_1 norm of an error is the square root of the energy of the error over the entire domain:

$$\|e\|_E = \left(\int_\Omega e^T \boldsymbol{L} e \, d\Omega \right)^{1/2} \tag{4.6}$$

The mean-square norm or L_2 norm of an error measures the root-mean square of the error over the entire domain:

$$\|e\|_0=\left(\int_\Omega e^T e\, d\Omega\right)^{1/2} \tag{4.7}$$

The maximum norm or L_∞ norm of an error measures the maximum absolute value of the error over the entire domain:

$$\|e\|_\infty=\max_{x\in\Omega}|e(x)| \tag{4.8}$$

The energy norm $\|e\|_E$ and the mean-square norm $\|e\|_0$ are more often used in finite-element analysis than the maximum norm $\|e\|_\infty$. $\|e\|_E$ involves the derivatives introduced by the linear differential operator $\boldsymbol{L}$, and thus contains the physical meaning of the model. $\|e\|_0$ is a straightforward and easily computed norm without an involvement of derivatives, but it has little physical meaning.

4.4.3 Error measures

A finite-element solution is more accurate when the error norm is smaller. Hence, the error norm is a natural and exact error measure in the finite-element method (Chellamuthu and Ida 1994).

To derive an error in terms of the energy norm for the model problem in Equation 4.1, we substitute Equation 4.5 into Equation 4.6 and obtain

$$\begin{aligned}\|e\|_E&=\left[\int_\Omega (Se)^T D(Se)\, d\Omega\right]^{1/2}\\&=\left[\int_\Omega (S(u-\hat{u}))^T D(S(u-\hat{u}))\, d\Omega\right]^{1/2}\\&=\left[\int_\Omega e_\varepsilon^T D e_\varepsilon\, d\Omega\right]^{1/2}\\&=\left[\int_\Omega e_\varepsilon^T e_\sigma\, d\Omega\right]^{1/2}\end{aligned} \tag{4.9}$$

where $e_\varepsilon=\varepsilon-\hat{\varepsilon}$ is the error of strain and $e_\sigma=\sigma-\hat{\sigma}$ is the error of stress.

Equation 4.9 shows that the energy norm of an error corresponds to the square

root of the strain energy of the error of the finite-element solution.

The three error norms (L_1, L_2 and L_∞) are defined over the entire domain, and the square of each of them can be obtained by summation of element contributions, that is

$$\|e\|^2 = \sum_{k=1}^{m} \|e_k\|^2 \tag{4.10}$$

where m represents the total number of elements.

The global relative error norm is defined by

$$\eta = \frac{\|e\|}{\|u\|} \tag{4.11}$$

The exact solution is approximated by summation of the approximate solution and the estimated error in order to compute the global relative error norm as

$$\eta \approx \frac{\|e\|}{(\|\hat{u}\|^2 + \|e\|^2)^{1/2}} \tag{4.12}$$

Assuming that the energy of the global solution error is equally distributed among elements, the admissible local element-error norm can be derived from the global relative error norm and the total number of elements:

$$\|e\|_a = \eta \left[\frac{\|\hat{u}\|^2 + \|e\|^2}{m} \right]^{1/2} \tag{4.13}$$

The admissible local element-error norm is used for adaptive mesh refinement as will be discussed in Section 4.7.

4.5 *A priori* error estimators

A priori error estimators provide information about the errors of the global solution rather

than of the individual solutions for each element. They include the use of solution residuals, condition number, and convergence rate estimate.

4.5.1 Solution residuals

Solution residuals are a way to assess the finite-element solution error, which in turn leads to the assessment of the overall mesh quality. Consider the model problem as a linear static finite-element analysis, where the linear differential operator is the system stiffness matrix. The problem is then expressed by

$$Ku = f \tag{4.14}$$

where K is the system stiffness matrix, u is the displacement vector and f is the load vector. The global solution residual is defined by

$$R = f - K\hat{u} \tag{4.15}$$

where $\hat{u}$ is the approximate solution obtained by the finite-element method.

By substituting Ku for f into Equation 4.15, the residual becomes

$$R = K(u - \hat{u}) = Ke \tag{4.16}$$

and the solution error e is then expressed by

$$e = K^{-1}R \tag{4.17}$$

In practice, the purpose of calculating solution residuals is to see how inaccurate the matrix inversion might have been. An approximate solution that is more accurate must have a small R, which results in a small solution error as long as K is well conditioned (Bathe 1996).

4.5.2 Condition number

The condition number of the system stiffness matrix is another possible way to assess the solution error (Bathe 1996). If K and u in Equation 4.14 are modified by small amounts, then the problem in Equation 4.1 is expressed by

$$(K+\delta K)(u+\delta u)=f \tag{4.18}$$

By substituting Equation 4.14 into Equation 4.18 and eliminating the small terms $\delta K \delta u$, the small displacement change δu is given by

$$\delta u=-K^{-1}\delta K u \tag{4.19}$$

Taking the energy norm on both sides of Equation 4.19 gives

$$\frac{\|\delta u\|}{\|u\|}\leqslant\frac{\lambda_n}{\lambda_1}\frac{\|\delta K\|}{\|K\|} \tag{4.20}$$

where λ_n is the largest eigenvalue of the system stiffness matrix K, λ_1 is the smallest eigenvalue, and λ_n/λ_1 is defined as the condition number (Bathe 1996).

A large condition number indicates that large solution errors are more likely to appear. The formulation of the system stiffness matrix K requires the availability of the boundary conditions and material properties.

The condition number could be used to assess mesh quality and mesh uniformity. If the mesh has no badly shaped tetrahedra, the largest eigenvalue, λ_n, is related to the longest length in the entire mesh. The smallest eigenvalue, λ_1, is related to the smallest element volume. Non-uniform meshes and meshes with badly shaped tetrahedra will have larger condition numbers (Shewchuk 2002). Condition numbers are also affected by boundary conditions and material properties.

4.5.3 Convergence error estimate

A finite-element approximation is known to converge in terms of the energy norm (Bathe 1996). Equation 4.21 expresses the nature of the convergence of the error norm:

$$\|e\|_E \leq ch^p \tag{4.21}$$

where $\|e\|_E$ represents the energy norm of the solution error, p is a positive integer representing the polynomial order of the shape function, h denotes the maximum normalized element size ($h < 1$), and c is a constant independent of h but dependent on material properties.

As the element size h tends to zero or the polynomial order p of the shape function goes to infinity, the approximate solution will asymptotically approach the exact solution as long as no singularities exist (Noor and Babuška 1987).

The convergence curve can be drawn on the basis of the approximate solutions and the corresponding total numbers of nodes, or elements, or degrees of freedom. With the curve, it is possible to estimate the qualities of the meshes generated by 3-D mesh generators for the same problem by looking at how close the solution based on a mesh is to the exact solution as estimated by the convergence curve.

4.6 *A posteriori* error estimators

This section is mainly based on Zienkiewicz and Taylor (2000) and Gratsch and Bathe (2005).

A posteriori error estimators use information obtained in the solution process to compute individual element-error estimates for the solution. These error estimates accomplish two goals. Firstly, they provide a quantitative idea of the exact error. Secondly, they are often used to steer adaptive mesh refinement. In this section, three *a posteriori* error estimators will be briefly discussed: explicit, implicit and Z-Z.

4.6.1 Explicit error estimators

Explicit error estimators are directly based on the approximate finite-element solution.

Babuška and Rheinboldt (1978a,b) established the fundamental of the explicit error estimators, and derived, for the model problem described in Equation 4.1, a bound for the explicit error estimates as

$$\|e\|^2 \leq \sum_{k=1}^{m} \left[c_1 h_k^2 \|\Delta R_k\|^2 + c_2 h_k \|J_{\delta_k}\|^2 \right] \tag{4.22}$$

where m is the total number of elements, k is the global element index, c_1 and c_2 are constants, h_k is the size of the element k, ΔR_k is the interior element residual of the element k and J_{δ_k} is the jump of the gradient across the boundary δ_k of the element k. The residual ΔR_k of element k is given by

$$\Delta R_k = f_k - L\hat{u}_k \tag{4.23}$$

where f_k is the load vector for the element k, and $\hat{u}_k$ is the approximate solution of element k. The jump J_{δ_k} is given by

$$J_{\delta_k} = \begin{cases} n \cdot \nabla u_k + n' \cdot \nabla u_{k'} & \text{if } \gamma \notin \Gamma \\ g_k - n \cdot \nabla u_k & \text{if } \gamma \in \Gamma_N \\ 0 & \text{if } \gamma \notin \Gamma_D \end{cases} \tag{4.24}$$

where n is the outward unit vector normal to Γ, g_k is the Neumann boundary value, ∇ is the gradient operator, γ are the element edges that separate the element k and its neighbouring element k' when they are inter-element edges.

The expression in Equation 4.22 directly leads to the local element-error norm given by

$$\|e_k\|^2 = c_1 h_k^2 \|\Delta R_k\|^2 + c_2 h_k \|J_{\delta_k}\|^2 \tag{4.25}$$

The constants c_1 and c_2 are in general unknown and depend on the specific

problem. However, the element-error norm defined in Equation 4.25 is often used, with approximate constants, to steer adaptive mesh refinement.

4.6.2 Implicit error estimators

Implicit error estimators require the solution of auxiliary local boundary-value problems whose solutions may yield accurate approximations to the solution error. The boundary value problems are to be locally solved on individual elements or sub-domains. Hence there are two methods for implicit error estimators, the element residual method and the sub-domain residual method.

The element residual method requires the approximation of the prescribed Neumann boundary conditions on each individual element. The upper error bound derived from this method is guaranteed only when the local problems are computed exactly. On the other hand, the sub-domain residual method implicitly takes the inter-element jumps of stress into consideration because the local problems on each sub-domain have already considered the inter-element edges. However, this method is computationally very expensive since each element is considered several times (Gratsch and Bathe 2005).

4.6.3 Z-Z error estimators

Z-Z error estimators were named after Zienkiewicz and Zhu (1987), who suggested post-processing the discontinuous gradient to obtain a more accurate strain σ^* that is interpolated using the same shape function as for the displacement $\hat{u}$.

In a finite-element formulation, the displacement is approximated by

$$u \approx \hat{u} = N\bar{u} \tag{4.30}$$

where N is the shape function, u is the exact displacement, $\hat{u}$ is the approximate displacement, and $\bar{u}$ is the vector of unknown coefficients that need to be determined.

For the Z-Z method, the stress is approximated by

$$\sigma^* = N\bar{\sigma}^* \tag{4.31}$$

where N is the shape function which is the same as for the displacement, σ^* is the approximate displacement, and $\bar{\sigma}^*$ is the vector of unknown coefficients that need to be determined.

The approximate stress solution $\hat{\sigma}$ ensures (Zienkiewicz and Zhu 1987) that

$$\int_\Omega N^T(\sigma^* - \hat{\sigma})d\Omega = 0 \tag{4.32}$$

where $$\hat{\sigma} = DS\hat{u} = (DSN)\bar{u}$$

Substitution of Equation 4.31 into Equation 4.32 yields an approximate $\bar{\sigma}^*$ given by

$$\bar{\sigma}^* = A^{-1}\bar{u}\int_\Omega N^T DSN d\Omega \tag{4.33}$$

where $$A = \int_\Omega N^T N\, d\Omega$$

The difference between the exact stress and the approximate stress is the error for the stress, that is written as

$$e_\sigma \approx \sigma^* - \hat{\sigma} \tag{4.34}$$

The approximate stress σ^* is then obtained by using Equation 4.31. Zienkiewicz and Zhu (1987) proved that the σ^* in Equation 4.34 is a better approximation to the exact stress than $\hat{\sigma}$ is. Therefore, it is convenient to use e_σ to evaluate various error norms.

The Z-Z error estimators are effective in problems where the order of the shape functions is one. However, they still have two drawbacks. One drawback is that they cannot handle the case when material discontinuities exist. The other drawback is the implicit assumption that smooth stresses mean accurate stresses, an assumption that may

not be true in practice (Gratsch and Bathe 2005).

4.7 Mesh refinement

A mesh can be refined (or coarsened) to achieve an desired accuracy of the finite-element solution. Error estimates are essential to steer either global mesh refinement or local adaptive mesh refinement.

A priori error estimates are usually used to drive global mesh refinement in which the element size h is reduced or the polynomial order p of the shape function is increased over the entire domain so as to reach a desired accuracy of the solution (Janicke and Kost 1996) .

A posteriori error estimates are often intended to steer local adaptive mesh refinement. By the end of the refinement, the contribution of each element to the total error should be about the same. This can be translated into checking whether the element-error norm is less than or equal to the admissible element-error norm. An effectivity index is introduced to determine where refinement is necessary:

$$\xi = \frac{\|e_k\|}{\|e\|_a} \tag{4.35}$$

where $\|e_k\|$ is element-error norm of the k^{th} element and $\|e\|_a$ is the admissible element-error norm.

Local adaptive mesh refinement is an iterative process, which can generally be summarized in the following steps (Verfurth 1994):

1. Construct a coarse mesh approximating the geometric domain of a finite-element model.
2. Obtain the finite-element solution on the initial coarse mesh.
3. Compute *a posteriori* error estimates and then the effectivity index of each element in the mesh.
4. Check the effectivity index on each element to decide whether or not the element has to be refined or coarsened. The ideal effectivity index for each

element should be one.

$\xi_k > 1+c$ Refine mesh

$\xi_k < 1-c$ Coarsen mesh

$1-c < \xi_k < 1+c$ No change

where c is a small positive number. Noor and Babuška (1987) suggested $c=0.2$.

5. Replace the previous mesh by the new refined mesh and repeat steps 2 to 4 until each element's effectivity index is close to one.

There are four versions of local adaptive mesh refinement (Ewing 1990).

- *h* version

The local elements are subdivided into elements with smaller sizes or combined into elements with larger sizes. The type of the new elements is the same as that originally used and the same polynomial order p (typically $p = 1$ or 2) of the shape function is maintained.

- *p* version

The local elements are not subdivided, but the polynomial order p of the shape function is allowed to increase to a higher value. As a result, new nodes are introduced on the edges of and/or inside existing elements.

- *hp* version

A mixed version that combines the *h* version and the *p* version.

- *r* version

A version that is based on node relocation in a mesh by moving nodes to regions where large errors are identified.

The *h* version requires more computer storage than the other versions do. The *p* version results in more dense matrices than the *h* version and is more convenient to implement (Noor and Babuška 1987). Both the *h* version and the *p* version have a polynomial rate of convergence in terms of the number of degrees of freedom (Verfurth 1994). The *hp* version is based on a sequence of refinement steps. Only the *h* version or

the *p* version is allowed in each step, with the typical refinement sequence being first the *h* version and then the *p* version. Unlike the *h* version and the *p* version, the *hp* version achieves an exponential rate of convergence with respect to the number of degrees of freedom.

The *h* version and the *p* version are more commonly used in practical problems than the *hp* version and the *r* version are. This is mainly due to the fact that the mathematical foundations for the *hp* version and the *r* version are less developed than for the other versions (Noor and Babuška 1987).

4.8 Conclusions

In the finite-element method, any error estimators involve questions of reliability, accuracy and computational cost.

As *a priori* error estimates, solution residuals, condition numbers and convergence rate estimates may be used for evaluating the overall mesh quality and to steer global mesh refinement so that the approximate solution asymptotically approaches the exact solution. However, the drawback of global mesh refinement is that it also subdivides regions of low error, thus causing unnecessary high computational cost.

A posteriori error estimates obtained during the finite-element solution process not only provide quantitative estimates of the element errors but also can be used to steer adaptive mesh refinement in which only the particular elements introducing large errors are refined. Among *a posteriori* error estimators, the Z-Z error estimators have gained more attention than others because of their high efficiency and reasonable accuracy. It is in general difficult for error estimators to provide guaranteed error bounds because error bounds for complex problems are either guaranteed but hardly computable or computable but not guaranteed.

Compared with the geometric mesh-quality measures discussed in Chapter 3, *a priori* error estimates and *a posteriori* error estimates are able to provide more accurate information about the solution error. The finite-element mesh-quality measures are able to assess not only the geometric characteristics (shape, size and orientation) but also the physical characteristics (material properties, boundary conditions and load conditions) of elements in a mesh. Therefore, it is reasonable to use the finite-element mesh-quality

measures to compare mesh qualities when the same finite-element definitions are maintained for a model with its volume meshes generated by different 3-D mesh-generation programmes.

CHAPTER 5

EVALUATION OF

3-D MESH-GENERATION SOFTWARE

5.1 Introduction

In this chapter, Section 5.2 outlines our guidelines for the selection of candidate 3-D mesh-generation software and describes the basic features of the candidate software selected. The evaluation criteria for our project are presented in Section 5.3. The models used for the evaluation are introduced in Section 5.4. The mesh processing is discussed in Section 5.5. The methods used for the evaluation of mesh quality are discussed in Section 5.6. The hardware and software environment for the evaluation is described in Section 5.7. Finally, conclusions are given in Section 5.8.

5.2 Selection of candidate software

Our guidelines for the selection of candidate software for further evaluation can be briefly summarized as follows:

- Candidate permits a surface mesh as its input for 3-D mesh generation
- Candidate generates unstructured volume meshes that include tetrahedral meshes
- Candidate is available for Linux and/or Windows operating systems

The 3-D mesh-generation software was chosen from two Web sites, *Mesh Generation & Grid Generation on the Web* (Schneiders 2005), and *Mesh Research Corner* (Owen 2005). The first Web site lists 158 software products, among which 64 products generate tetrahedral elements. The second Web site lists 94 software products, among which 42 products generate tetrahedral elements. There is some overlap in the lists on the two Web sites, which list most of the 3-D mesh-generation software currently available, both free and commercial.

Table 5.1 lists seven free and four commercial 3-D unstructured mesh-generation programmes selected based on the guidelines given above. Of the free programmes, COG no longer provides any support; SolidMesh has no version available for either Windows or Linux; DistMesh requires that MATLAB be installed, which introduces an extra cost for the MATLAB licence; and the mesh-generation code of NETGEN is included in Gmsh. Therefore, COG, SolidMesh, DistMesh and NETGEN are excluded from evaluation. Among the commercial programmes, the one-year licenses for ADINA, ANSYS and HyperMesh cost a few thousand dollars, which is very expensive. GiD is much less expensive, and we already have a permanent licence for version 6.1.2 that was initially used by Siah (2002) for his thesis project. In this project, version 7.2 of GiD is evaluated (as a trial version).

Table 5.1 Unstructured mesh-generation software

Free software				
Software	**Developers**	**Platform**	**Algorithm**	**Optimization**
COG	Schmelzer I	Linux	Delaunay	Yes
DistMesh	Perrson P-O	Linux & Windows using MATLAB	Delaunay	No
Gmsh	Geuzaine C	Linux, Mac & Windows	Delaunay	Yes
GRUMMP	Ollivier-Gooch C	Linux	Delaunay	Yes
NETGEN	Schöberl J	Linux, Unix & Windows	Delaunay	Yes
TetGen	Hang S	Linux & Windows	Delaunay	Yes
SolidMesh	MSU-ERC	Unix	advancing-front	N/A
Commercial software				
Software	**Developers**	**Platform**	**Algorithm**	**Optimization**
ADINA	ADINA Inc.	Linux, Mac & Windows	Delaunay or advancing-front	Yes
ANSYS	ANSYS Inc.	Linux, Mac & Windows	Delaunay or advancing-front	Yes
HyperMesh	Altair Inc.	Linux, Mac & Windows	advancing-front	Yes
GiD	CIMNE	Linux & Windows	advancing-front	Yes

Finally, GiD, Gmsh, GRUMMP and TetGen were chosen for the evaluation. GiD utilizes the advancing-front method and the remaining three programmes utilize the Delaunay-based method. Their important features are summarized in Table 5.2, which lists the element types that the candidate programmes can generate; the file formats, including the formats peculiar to themselves and the formats used by other software; and the mesh-density and mesh-gradation control mechanisms used. Every programme but GRUMMP has a GUI.

Table 5.2 Major features of candidate software

Features		**GiD**	**Gmsh**	**GRUMMP**	**TetGen**
Element types		quadrilateral, triangle, hexahedron and tetrahedron	triangle and tetrahedron	triangle and tetrahedron	triangle and tetrahedron
File formats handled	native	.msh and .gid	.geo and .msh	.bdry, .smesh and .vmesh	.node, .ele, .face, .smesh and .poly
	other	NASTRAN (.nas), AutoCAD (.dxf), Stereolithography (.stl) and IGES (.iges or .igs)	Stereolithography (.stl)	No	Medit (.mesh), Geoviw (.off) and Stereolithography (.stl)
Mesh-density control		element size	characteristic length	length scale (resolution) or coarsen routine	Maximum-tetrahedron-volume constraint
Mesh-gradation control		size-transition coefficient	characteristic length	length scale (grading)	No
GUI		Yes	Yes	No	Yes with TetView (Hang 2005b)

5.3 Evaluation criteria

The goal of our evaluation is to pick the programme that is most suitable for our overall purposes, which leads to specific criteria for the present evaluation. There are five criteria that are discussed in the following sub-sections.

5.3.1 Preservation of boundary-surface mesh

As discussed in Chapter 1, the goal of our project is to streamline our software pipeline so as to generate finite-element volume meshes for complex structures with multiple parts. Each individual part with its surface mesh definition is imported for 3-D mesh generation separately, then the resulting volume meshes are joined together to form the complex structure. The join operation requires that those parts of the surfaces touching each other should be identical or almost identical. Moreover, it is necessary that the boundary conditions and the load conditions defined in the Fie programme for the surface mesh can be assigned to the same nodes on the resulting volume mesh. It is therefore necessary that the initial triangulated boundary surface mesh be preserved unchanged by the volume-mesh generation process.

5.3.2 Mesh quality

The candidate programme should be able to generate meshes with high quality as assessed by a number of methods, i.e., visual inspection, histograms of the shape quality and of the size of elements, solution residuals and condition numbers, and closeness to the exact solution approximated by the convergence curve.

5.3.3 Robustness

The candidate programme should be robust in that it can always succeed in generating volume meshes for all models used for the evaluation.

5.3.4 Time efficiency

Based on the current hardware and software environment, the time required for 3-D mesh generation by the candidate programme should be reasonably short.

5.3.5 Cost of programme

The candidate programme should be low-cost or free of charge.

5.4 Models

To evaluate the 3-D mesh-generation software, four models with different complexities of shape were selected.

One model is a simple thin block created in GiD. The model has a length of 10, a width of 6 and a height of 1. An element size of 0.5 was selected to generate the surface triangular mesh.

The remaining three models consist of one ligament that is the lateral bundle of the posterior incudal ligament, referred to simply as pillat, and two ossicles, the incus and malleus. These models were based on high-resolution magnetic-resonance images of the human middle ear, provided by Henson and Henson (2005). The segmentation was done by various members of our lab and the surface triangular meshes were generated using our Tr3 programme.

Information about the surface meshes for all four models is listed in Table 5.3.

Table 5.3 Information about surface meshes of models

Model	No. of nodes	No. of elements (triangles)
thin block	362	720
pillat	241	478
incus	1870	3736
malleus	2118	4232

The thin block is shown in Figure 5.1 and the remaining three models are illustrated in Figure 5.2. All of them are rendered with flat lighting in GiD.

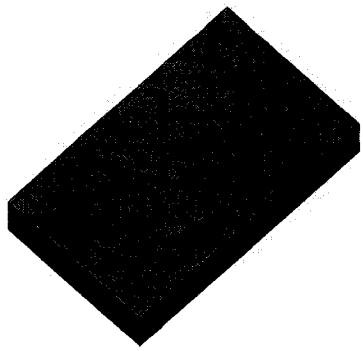


Figure 5.1 Thin-block model

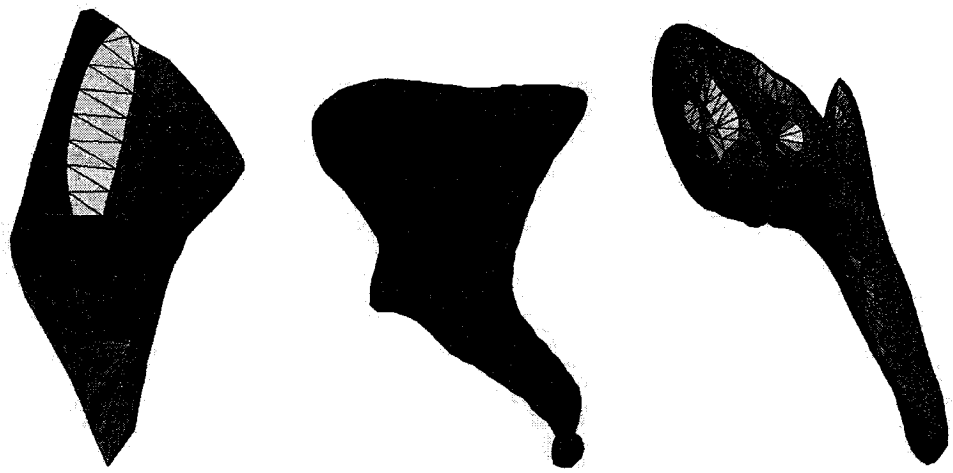


Figure 5.2 Three structures of middle ear

5.5 Mesh processing

This section discusses the processing of mesh files and meshes. Section 5.5.1 discusses the conversion from a surface mesh definition generated by the Tr3 programme to the native files of the candidate programmes for 3-D mesh generation. Section 5.5.2 discusses how the mechanical parameters are added to the volume mesh for the purpose of finite-element analysis. The pre-processing of the surface mesh and the post-processing of the volume mesh are discussed in Sections 5.5.3 and 5.5.4, respectively. A programme called Fcf was developed to realize all of these functions.

5.5.1 Mesh file format conversion

The Tr3 programme can produce three different file formats: the native file format for GRUMMP, the file format with an extension of .wrl for visualization in Virtual Reality Modelling Language (VRML) viewers, and the file format with an extension of .sap for finite-element analysis using the Sap programme. These formats cannot be directly imported into GiD, Gmsh or TetGen. The Fcf programme was used here to convert the surface mesh file with an extension of either .wrl or .sap to the native file formats for GiD, Gmsh and TetGen.

5.5.2 Assignment of the mechanical parameters to the volume mesh

For the purpose of finite-element simulation with the Sap programme, the Fcf programme reads the boundary conditions and the load conditions defined at nodes in the surface mesh file with an extension of .sap that is created by our Tr3 programme, and assigns these conditions to the corresponding nodes on the surface of the resulting volume mesh. In addition, the material properties of the shell elements of the surface mesh are assigned to the tetrahedral elements of the volume mesh. These properties include the Young's modulus and the Poisson's ratio.

5.5.3 Pre-processing the surface mesh

A surface mesh generated by the Tr3 programme needs to be verified to ensure that it has consistent triangle orientations and that it is simple and closed. Once the surface mesh passes the pre-processing test, it can proceed to 3-D mesh generation.

5.5.3.1 Surface closure check

The surface mesh represents a simple closed surface if every one of its edges is shared by exactly two neighbouring triangles, neither more nor fewer. The Fcf programme reads the surface mesh produced by the Tr3 programme, with its definition containing all vertex coordinates followed by lists of vertices defining elements. The Fcf programme then checks for closure of the surface mesh. The programme starts from an initially selected triangle and scans all other triangles to check whether the three edges are shared by

neighbouring triangles. For each edge, a flag is initially set to 1 before checks start and the flag is incremented by 1 only if the edge is shared by a neighbouring triangle. The process is repeated until all triangles on the surface mesh have been checked. Upon completion of the process, if all edges on the surface mesh have flag values of 2, the surface mesh is confirmed to be closed. In contrast, those edges with flag values of 1 are boundary edges, and those edges with flag values larger than 2 correspond to locations where two or more surfaces touch each other. In any case, the global vertex indices and coordinates of these edges are exported to a text file. Based on this file, a corrective modification can be performed manually in the Fie programme so that a simple closed surface mesh can be generated.

5.5.3.2 Surface triangle orientation detection and correction

The surface triangle orientations are consistent if the vertices of any triangle on the surface mesh are numbered counter-clockwise (CCW) when viewed from outside the boundary surface. Incorrect orientations would cause new nodes to be inserted outside rather than inside the boundary-surface mesh during the generation of the volume mesh. For example, the advancing-front method inserts points with respect to the orientations of the active fronts to make sure that the points are inserted inside rather than outside the model domain. The same is true for the Delaunay-based method. Inconsistent orientations may be caused by software error when the surface mesh is created or by a user's mistake while doing segmentation manually. In addition to flagging those triangles that have incorrect orientations, the Fcf programme can also automatically correct them. The steps for detecting and correcting the incorrect orientations are summarized in the following:

1. Finding a starting triangle
    - Select a point outside the model domain: determine a box enclosing the model, and then select a point along the x axis at a distance from the geometrical centre of the box that is equal to the x length of the box
    - Define a ray: the ray has its origin at the point selected in step 1 and its direction pointing to the geometrical centre of the box.
    - Find the intersection points where the ray passes through the surface mesh of the model: the number of intersection points should be even if the

surface mesh is closed (O'Rourke 1998). It is possible that there are no intersection points. In such a case, the ray can be slightly rotated around an axis that passes through the geometrical centre of the box and is parallel to the x, y or z axis until an intersection occurs.

- Find the starting triangle: the intersection point nearest to the origin of the ray is determined by comparing the distances between the intersection points and the origin of the ray. The triangle on which the nearest intersection point is located is defined as the starting triangle for the purpose of mesh-orientation detection and correction.

2. Detecting and correcting the orientation of the starting triangle
   - Whether or not the triangle is correctly oriented is determined by the sign of the volume of the tetrahedron with its four vertices (x_i, y_i, z_i) (i=0,1,2,3), of which three vertices (i=0,1,2) correspond to the starting triangle and the fourth vertex (i=3) is the origin of the ray. The volume is given by:

$$V=\frac{1}{6}\begin{vmatrix} x_0 & y_0 & z_0 & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} \tag{5.1}$$

   If the sign of the volume is positive, then the orientation of the starting triangle is correct; otherwise, the orientation incorrect.
   - The correction is performed on the basis of the sign of the tetrahedron volume. If the sign of the volume is negative, swapping any two vertices in the starting triangle results in the correct orientation.
3. Detecting and correcting orientations of the remaining triangles
   - An edge shared by two neighbouring triangles should have opposite orientations in the two triangles if they are numbered consistently. The orientations of neighbouring triangles are detected on the basis of the correct orientation of the current triangle. A neighbouring triangle which

has an incorrect orientation is corrected by swapping any two of its vertices.

- The process of detection and correction is iterated. The overall process terminates when all triangles in the mesh have been processed.

5.5.4 Post-processing the volume mesh

A volume mesh may be topologically incorrect because of failure of the mesh-generation algorithms. For example, some nodes may be inserted outside the model domain during 3-D mesh generation, or some elements may have negative volumes, or there may be overlapping elements, or gaps between elements.

To detect such problems, the Fcf programme was used to confirm the correctness of a volume mesh by checking whether each element in the mesh has a positive volume, and whether the volume enclosed by the surface mesh equals the volume obtained by summing together the volumes of all tetrahedral elements. The tolerance for checking for volume equality of the two floating-point numbers is set to 10^{-25}.

5.6 Mesh evaluation

The overall mesh quality of a volume mesh is assessed by a number of methods as described in the following subsections. Visual inspection is discussed in Section 5.6.1. Histograms of the shapes and sizes of elements are discussed in Section 5.6.2. Finite-element solution residuals and condition numbers are discussed in Section 5.6.3. Closeness of the finite-element solution to the exact solution approximated by a convergence curve is discussed in Section 5.6.4.

5.6.1 Visualization of the volume mesh

Few methods are available to effectively visualize the interior elements of a volume mesh. The mesh quality of an element can be roughly evaluated by visual inspection. However, the visual inspection becomes more difficult for a volume mesh with a lot of elements. Nevertheless, several methods have been attempted for this purpose.

Figure 5.3(a) shows an example of the wire-frame viewing method, which

provides an overview of mesh density at different locations but does not give a clear impression of the shapes of the interior elements.

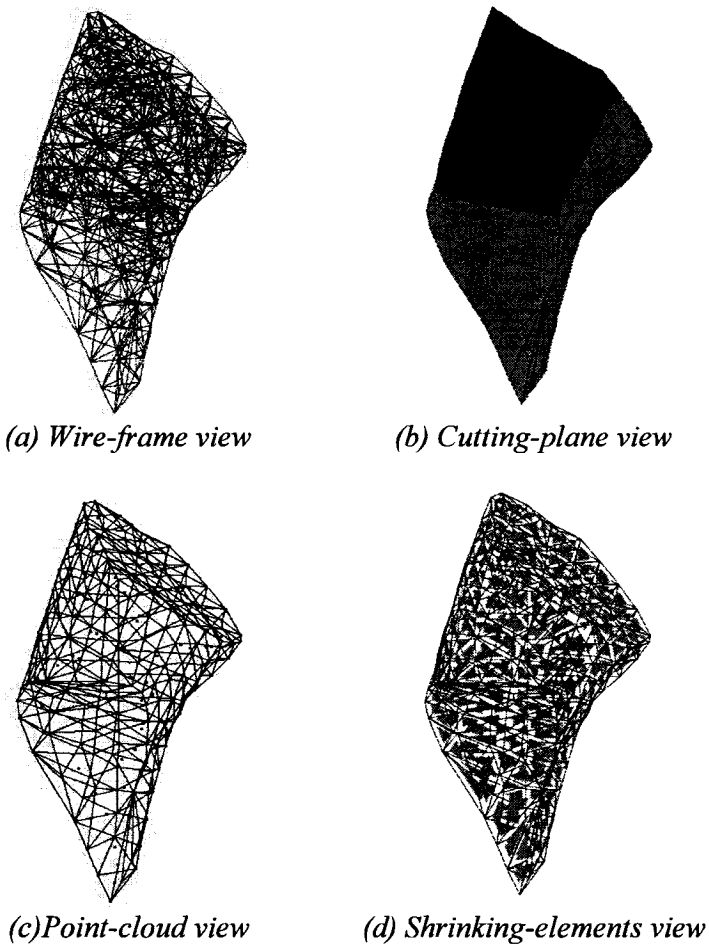
Figure 5.3(b) gives an example of the cutting-plane viewing method, which provides the shapes of elements intersecting with a user-specified plane. The plane is selected to be parallel to the *x-y* plane in this example. By moving the plane along the *z* axis, the user is able to inspect the entire mesh. In the figure, aqua is used for the elements intersecting the user-specified plane and fuchsia indicates the remaining elements beneath the cutting plane.

Figure 5.3(c) provides an illustration of the point-cloud viewing method that display the boundary-surface mesh and the nodes inside the boundary surface. This method is better for visualizing point spacing for 2-D meshes than for 3-D meshes (Remotigue *et al.* 1994).

Figure 5.3(d) shows a representation of the shrinking-element viewing method. It allows the user to inspect the shapes of the interior elements, but it gives only a limited idea of mesh quality through gaps between the shrunk elements.

In addition, Haimes *et al.* (1993) proposed a visualization method in which elements can be opaque or transparent depending on their shape qualities. The implementation of this method also appears in a commercial software package called PASTEK (NECS Inc. 2005). This method is good for determining the locations of badly shaped or well-shaped elements by modifying the threshold value. It is difficult, however, to determine the number of opaque elements, and also difficult to determine the overall mesh quality.

Interactivity is generally a great help in visualization via rotation, translation and zooming. It is commonly used in combination with the above methods in practical applications.



(a) Wire-frame view *(b) Cutting-plane view*

(c)Point-cloud view *(d) Shrinking-elements view*

Figure 5.3 Four visualization methods

5.6.2 Histograms of the shape qualities or sizes of elements

As discussed in Chapter 3, most mesh-quality measures are dependent on the shapes of elements. Three tetrahedral shape measures (θ_{min}, ρ and η) are chosen here for the evaluation. The first measure, θ_{min}, is the minimum solid angle of a tetrahedron. This measure is able to detect most types of badly shaped tetrahedra. The second measure, ρ, is the ratio of the radii of the circumscribed sphere to the inscribed sphere of a tetrahedron. This ratio is used in the Delaunay-based mesh-generation method. The third measure, η, is the ratio between the volume and the sum of the squares of the edge lengths of a tetrahedron, which is used in the advancing-front method.

Apart from the shape qualities of elements, the sizes of elements in a mesh also influence the finite-element solution accuracy. As a result, the sizes of elements are also taken into consideration in the evaluation. The size of an element is characterized by the volume v or by the average of the edge lengths of the element, l_{avg}.

The histogram is used here for addressing overall mesh-quality assessment. It gives the user a sense of mesh quality throughout the entire domain. In a shape-quality histogram, the horizontal axis represents the shape quality, that is normalized within a range between 0 and 1, where 0 represents a flat tetrahedron and 1 represents an equilateral tetrahedron; the vertical axis represents the percentage of elements for each bin. The number of bins is chosen here to be 50. Using shape-quality histograms, it is easy to make comparisons among the four 3-D mesh-generation programmes by looking at the distributions of badly shaped tetrahedra and their normalized shape qualities. Similarly, in the size histograms, the horizontal axis represents the volumes or the averages of the edge lengths of the elements, and the vertical axis represents the percentage of elements for each bin. The number of bins is again chosen to be 50. In both the shape-quality and size histograms, the linear scale will be replaced by a logarithmic scale when the region of interest in the histograms is concentrated at very small values.

To quantitatively describe the histograms for models, statistical information about the three shape measures and the two size measures are computed. The information includes:

- Maximum, minimum, mean, standard deviation and skewness of normalized values of the three shape measures.
- The number and the percentage of tetrahedra having normalized values less than 0.1. Such tetrahedra are taken to be badly shaped.
- The number and the percentage of tetrahedra having normalized values larger than 0.9. Such tetrahedra are taken to be well shaped.
- Maximum, minimum, mean, standard deviation and skewness of the two size measures.

The standard deviation and skewness are two measures for assessing the variability of a data set. The standard deviation is a measure for characterizing how the

data values spread out in the data set, and it is defined by

$$std=\sqrt{\sum_{i=1}^{N}(x_i-\bar{x})^2/(N-1)} \tag{5.2}$$

where N is the number of samples in the data set and $\bar{x}$ is the mean value. The mean value and standard deviation become less meaningful when the histogram is asymmetrical. The skewness is a measure of the degree of asymmetry of a distribution, and it is defined by

$$skewness=\frac{\sum_{i=1}^{N}(x_i-\bar{x})^3}{(N-1)std^3} \tag{5.3}$$

Any symmetric distribution will have a skewness of zero. Negative skewness values indicate data that are skewed to the left and positive skewness values indicate data that are skewed to the right. 'Skewed to the left' means that the left tail is long relative to the right tail. Similarly, 'skewed to the right' means that the right tail is long relative to the left tail.

It is better to use the percentage of badly shaped or well-shaped tetrahedra, rather than the number, when comparing volume meshes. A higher percentage of tetrahedra with normalized values less than 0.1 indicates that the overall mesh quality is likely to be worse. In contrast, a higher percentage of tetrahedra with normalized values larger than 0.9 means that the overall mesh quality is probably better.

The Fcf programme is used to compute the three shape measures and the two size measures. MATLAB (The MathWorks, 2006) is used to compute the maximum, minimum, standard deviation and skewness of the above measures, and to draw the histogram figures.

5.6.3 Solution residuals and condition number

As discussed in Sections 4.6.1 and 4.6.2, solution residuals and the condition number of

the system stiffness matrix may be used to evaluate the overall mesh quality. The same mechanical parameters (boundary conditions, load conditions and material properties) were defined for the volume meshes produced by all of the programmes, and then simulations were run using the Sap programme. The root mean squares of residuals were computed and compared. The condition number was calculated using Luk, a programme developed by Funnell (2006). The Luk programme reads the system stiffness matrix produced by the Sap programme, uses Eispack (NetLib 2005) routines to compute eigenvalues, and computes the condition number.

5.6.4 Closeness to the exact solution

The convergence curve discussed in Section 4.5.3 shows that the approximate solution approaches the exact solution as the mesh density increases. A convergence curve for the thin-block model was obtained by running COMSOL (COMSOL Inc. 2006), a commercial finite-element package for which we already have a licence. Using COMSOL, it is easy to generate the regular geometric shape of the thin-block model, to assign mechanical parameters, and to do the finite-element simulations. Moreover, the user does not have to use the Fcf programme twice, once for the file format conversion of the surface mesh and once for the assignment of mechanical parameters to the resulting volume, for each different mesh density.

The quality of the volume meshes can be evaluated by looking at how close the solutions based on those volume meshes are to the exact solution that is estimated by the convergence curve.

5.7 Evaluation environment

The same computing environment was maintained throughout the evaluation. GiD, Gmsh and TetGen provide both a Windows version and a Linux version, while GRUMMP provides a Linux version only. As a result, Linux was chosen as the operating system. Moreover, to accurately calculate the time required for the 3-D mesh-generation process, no other processes were permitted to run during 3-D mesh generation.

The operating system used was Debian GNU/Linux version 3.1 ('Sarge') with

kernel version 2.6.8. The computer hardware specifications are listed below:

- CPU: Intel Pentium 3.0 GHz
- Hard drive: WDC 80G 7200 RPM
- Memory: 1 Gbyte
- Video card: Nvidia 5200FX

5.8 Conclusions

Four models (a thin block, and one ligament and two ossicles of the human middle ear) with different complexities of shape were chosen as the input surface triangular meshes for the evaluation.

A programme called Fcf was developed to verify the closure and consistency of the input boundary-surface meshes before 3-D mesh generation, and to verify the topological correctness of the resulting volume meshes. The Fcf programme was also used for file format conversion, and for retrieving the mechanical parameters from the input boundary-surface mesh and then assigning them to the resulting volume mesh.

To evaluate the candidate software, five criteria have been proposed. The first criterion is that the candidate software is able to preserve the boundary-surface mesh. With the boundary-surface mesh preserved, the mechanical parameters for the input surface mesh can be accurately assigned to the resulting volume mesh, and furthermore two or more models can be accurately joined at their interfaces. The second criterion is that the volume mesh generated by the candidate software should possess a high quality. The mesh quality was evaluated in four ways. Firstly, visual inspection using the wire-frame viewing method and the cutting-plane method provided the first impressions. Secondly, the overall mesh quality was evaluated based on shape-quality histograms and size histograms. Three shape measures and two size measures were selected for evaluating mesh quality. Thirdly, solution residuals and condition numbers were used to evaluate mesh quality in terms of finite-element solution accuracy. Lastly, closeness to the exact solution estimated by the convergence curve was also used to evaluate mesh quality.

According to the remaining criteria, the candidate software should be robust,

should be low cost or free of charge, and should require a short time for 3-D mesh generation.

During the evaluation, the two methods for visual inspection will be compared, and the same is true for the three shape measures as well as the two size measures. The methods that are able to effectively evaluate mesh quality will be summarized.

CHAPTER 6

RESULTS

6.1 Introduction

In this chapter, the results of the evaluation of four 3-D mesh-generation programmes (GiD-7.2, Gmsh-1.60, GRUMMP-0.3.0 and TetGen-1.3.4) are presented and discussed. Section 6.2 presents an initial evaluation using a thin-block model to determine how the candidate software controls mesh density. Section 6.3 presents the results of evaluating mesh quality based on visualization methods and on histograms of the shape qualities and sizes of elements in the meshes of three structures of the human middle ear. Sections 6.4 to 6.6 presents the results of the evaluation of the residuals of the finite-element solution, the condition numbers of the system stiffness matrix, and the closeness to the exact solution as estimated by a convergence curve, respectively. Finally, conclusions are presented in Section 6.7.

For ease of explanation in what follows, 'surface mesh' is used to refer to triangular meshes while 'volume mesh' is used to refer to tetrahedral meshes.

6.2 Initial evaluation

As discussed in Chapter 5, the first requirement of the evaluation is the ability to preserve the boundary-surface mesh unchanged during 3-D mesh generation. An initial evaluation is done by using a simple thin-block model to determine what mesh-density parameters are appropriate for preserving the boundary-surface mesh. The results of the initial evaluation will lead to choosing the proper mesh-density parameters for the remaining models. The boundary-surface mesh is preserved when the candidate programmes do not produce extra nodes on the boundary-surface mesh in the resulting volume mesh.

GiD (CIMNE Inc. 2005) utilizes a parameter called *element size* to control mesh density. The element size is set either automatically, by averaging edge lengths of the active-front triangles, or manually by the user. The relationship between the number of extra surface nodes and the element size is illustrated in Figure 6.1, in which the number

of extra surface nodes decreases as the element size increases to 0.86, which is the average edge length within the mesh. There are no extra surface nodes when the element size is larger than 0.86.

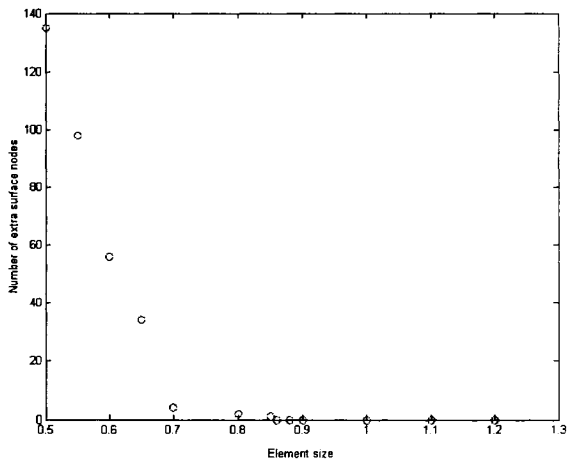


Figure 6.1 Number of extra surface nodes vs. element size in GiD

Similar to GiD, Gmsh (Geuzain and Remacle 2005) uses a parameter called *characteristic length* to modify mesh density. The relationship between the number of extra surface nodes and the characteristic length is illustrated in Figure 6.2, in which the number of extra surface nodes decreases as the characteristic length increases to 0.5 and remains zero when the characteristic length is larger than 0.5.

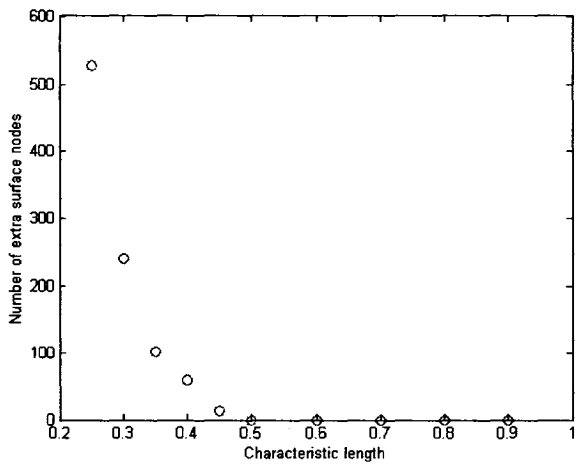


Figure 6.2 Number of extra surface nodes vs. characteristic length in Gmsh

GRUMMP (Ollivier-Gooch 2005) utilizes a parameter called *length scale* to modify mesh density. The relationship between the number of extra surface nodes and the length scale is illustrated in Figure 6.3, in which the number of extra surface nodes decreases as the length scale decreases and remains at 47 when the characteristic length is less than 0.1. As a result, the boundary-surface mesh is not preserved. GRUMMP also offers a coarsen routine which does help to remove some of the extra surface nodes, but the coarsen routine greatly modifies the topological relations among nodes of the boundary-surface mesh of the resulting volume mesh, as shown in Figure 6.4, which illustrates the same region of the resulting volume mesh before and after use of the coarsen routine.

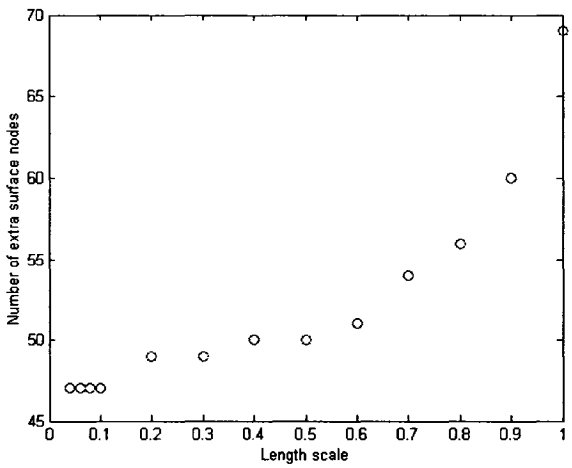


Figure 6.3 Number of extra surface nodes vs. length scale in GRUMMP

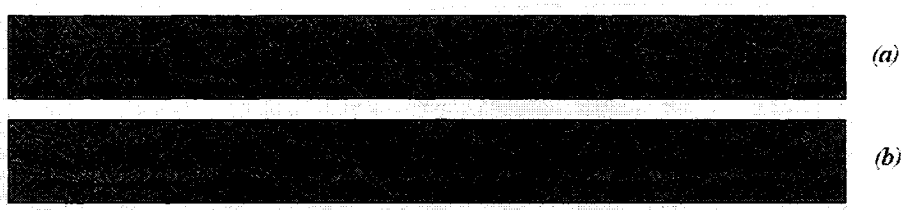


Figure 6.4 Calling coarsen routine in GRUMMP
(a) Initial mesh (b) Mesh after calling coarsen routine

To preserve the boundary-surface mesh, the original version of GRUMMP was modified (in the 'tetra' function of its source code) so as to generate an initial tetrahedralization only, without the further mesh improvement that is normally done by

default. The initial tetrahedralization does preserve the boundary-surface mesh for the thin-block model. It should be mentioned that this modification violates the design philosophy of GRUMMP. The modified version of GRUMMP is referred to below as GRUMMP-m.

TetGen (Hang 2005a) uses a parameter called *maximum-tetrahedron-volume constraint* to modify mesh density. The actual initial maximum tetrahedron volume is available once the initial mesh, a mesh that has not been optimized yet by calling optimization routine, has been generated. The relationship between the number of extra surface nodes and the maximum-tetrahedron-volume constraint is illustrated in Figure 6.5, in which the number of extra surface nodes increases as the maximum-tetrahedron-volume constraint decreases. The number of extra surface nodes remains at 4 when the constraint is larger than 0.13, which is the actual initial maximum tetrahedron volume for the thin-block model. Thus, the boundary-surface mesh is not preserved for any value of the constraint.

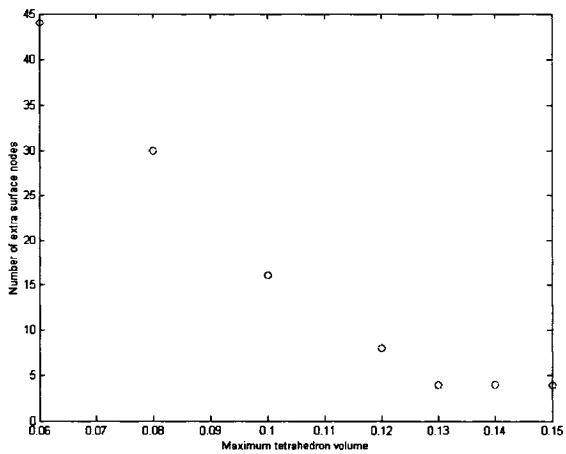


Figure 6.5 Number of extra surface nodes vs. maximum-tetrahedron-volume constraint in TetGen

In summary, GiD uses the average edge length, Gmsh uses the characteristic length that is equal to the average edge length, and GRUMMP-m uses the default length scale, to preserve the boundary-surface mesh. GRUMMP and TetGen are unable to preserve the boundary-surface mesh for the thin-block model, which may not be true for the remaining models. As the evaluation focuses on the assessment of mesh quality,

GRUMMP and TetGen are evaluated along with GiD, Gmsh, GRUMMP-m.

6.3 Evaluation of mesh quality

The remaining three models described in Section 5.4 are evaluated. The mesh-density parameters used here for all three models are based on the results of the evaluation in the previous section. The element-size parameter for GiD was set to 0.8, and the characteristic-length parameter for Gmsh was set to 0.01. TetGen, GRUMMP and GRUMMP-m utilize the default parameters for their initial meshes.

Information about the volume meshes generated for the models are listed in Tables 6.1 to 6.3. The information includes the numbers of nodes and elements; the numbers of nodes inserted on and inside the boundary-surface mesh; and the times spent for 3-D mesh generation.

Table 6.1 Information about volume mesh for the pillat model

Software	No. of nodes	No. of elements (tetrahedra)	Inserted nodes		Time (Seconds)
			On surface	Inside domain	
GiD	257	694	0	16	2.0
Gmsh	298	909	0	57	1.8
GRUMMP	1400	4717	838	321	2.2
GRUMMP-m	241	654	0	0	0.6
TetGen	1348	4607	863	244	0.4

Table 6.2 Information about volume mesh for the incus model

Software	No. of nodes	No. of elements (tetrahedra)	Inserted nodes		Time (Seconds)
			On surface	Inside domain	
GiD	2424	8308	0	554	61.8
Gmsh	2348	7630	0	478	121.2
GRUMMP	26227	109557	12927	11430	18.6
GRUMMP-m	1872	6060	2	0	3.8
TetGen	12808	47983	7698	5110	4.9

Table 6.3 Information about volume mesh for the malleus model

Software	No. of nodes	No. of elements (tetrahedra)	Inserted nodes		Time (Seconds)
			On surface	Inside domain	
GiD	fail				
Gmsh	2635	8559	0	526	154.5
GRUMMP	13535	50662	7154	6381	24.8
GRUMMP-m	2122	6812	4	0	4.0
TetGen	15735	59229	9554	6181	6.4

The time spent on 3-D mesh generation by either Gmsh or GRUMMP is the total time including the initial mesh generation and the use of an optimisation routine a number of times. The number of uses of the optimisation routine was determined by calling it repeatedly until there was no noticeable further change in the resulting mesh. The number of optimisations for Gmsh is three, and for GRUMMP it is one. In contrast, the times for both GiD and TetGen include only the time spent on the initial mesh generation since there is no optimisation routine available for them.

6.3.1 Visualizations

The wire-frame viewing method and the cutting-plane viewing method were used to visualize the volume meshes. The wire-frame views of models are generated in GiD. The cutting-plane views of models are generated using TetView (Hang 2005b), a visualization programme for TetGen.

6.3.1.1 Wire-frame views

The wire-frame viewing method provides a better overview of mesh density than of mesh quality. Figures 6.6 to 6.8 illustrate that GiD, Gmsh and GRUMMP-m generate similar mesh densities, and that both GRUMMP and TetGen generate higher mesh densities than the others do. It can be seen in these figures that GRUMMP-m generates many badly shaped tetrahedra in the volume meshes. Most badly shaped tetrahedra are long and thin.

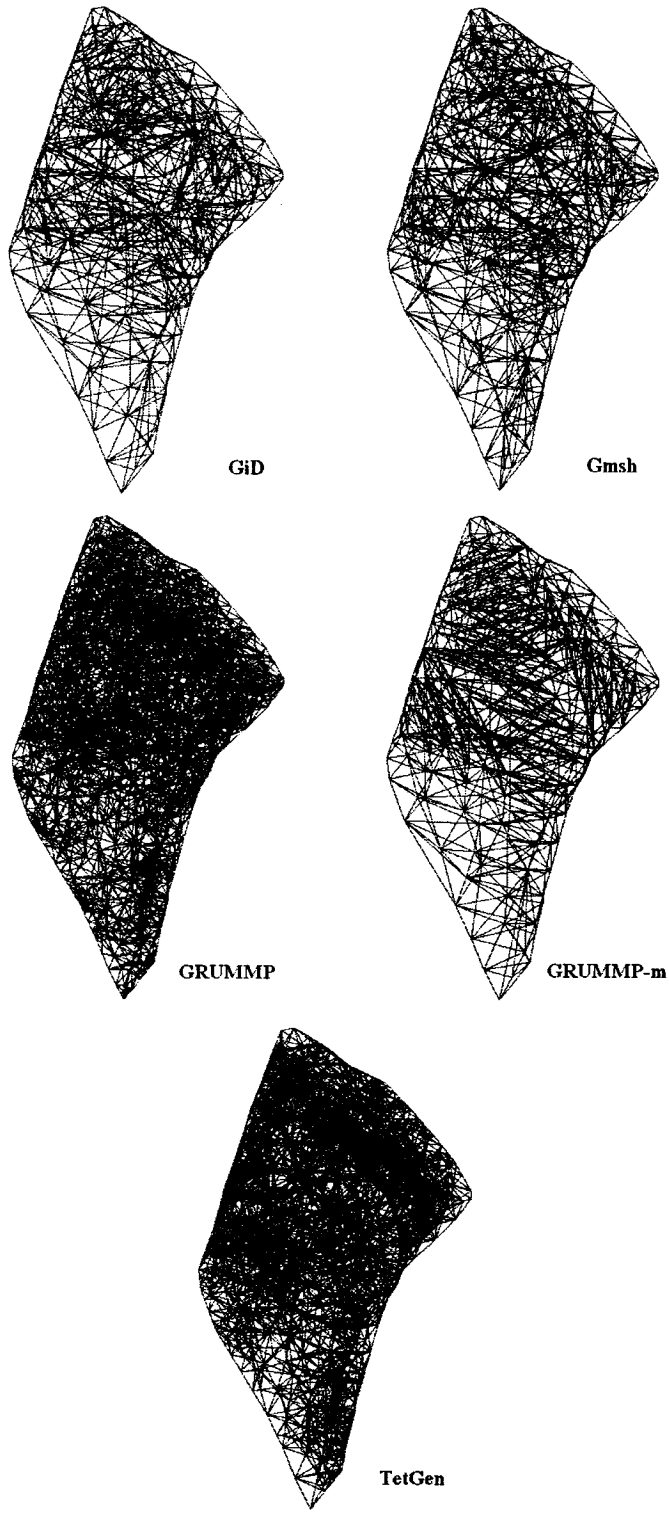


Figure 6.6 Wire-frame views of the pillat model

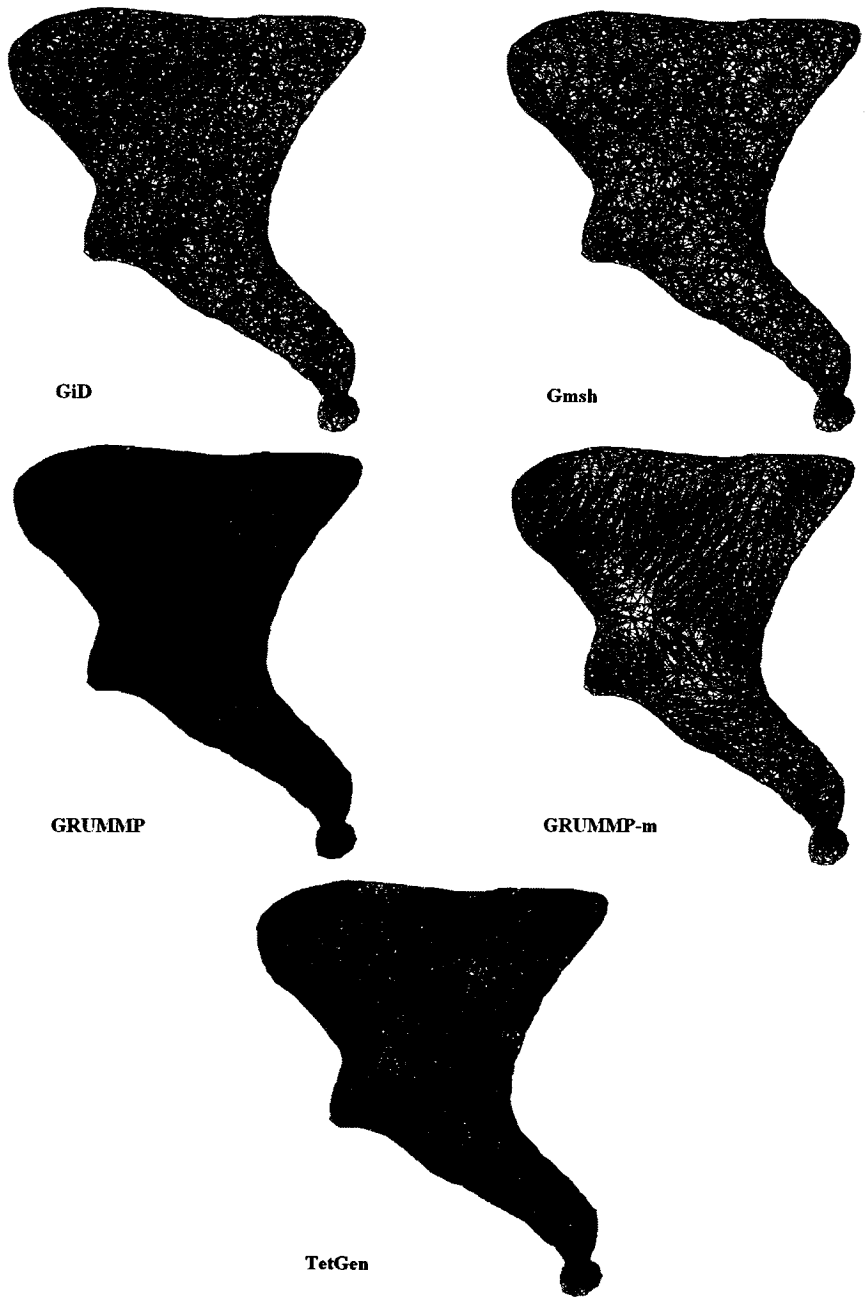


Figure 6.7 Wire-frame views of the incus model

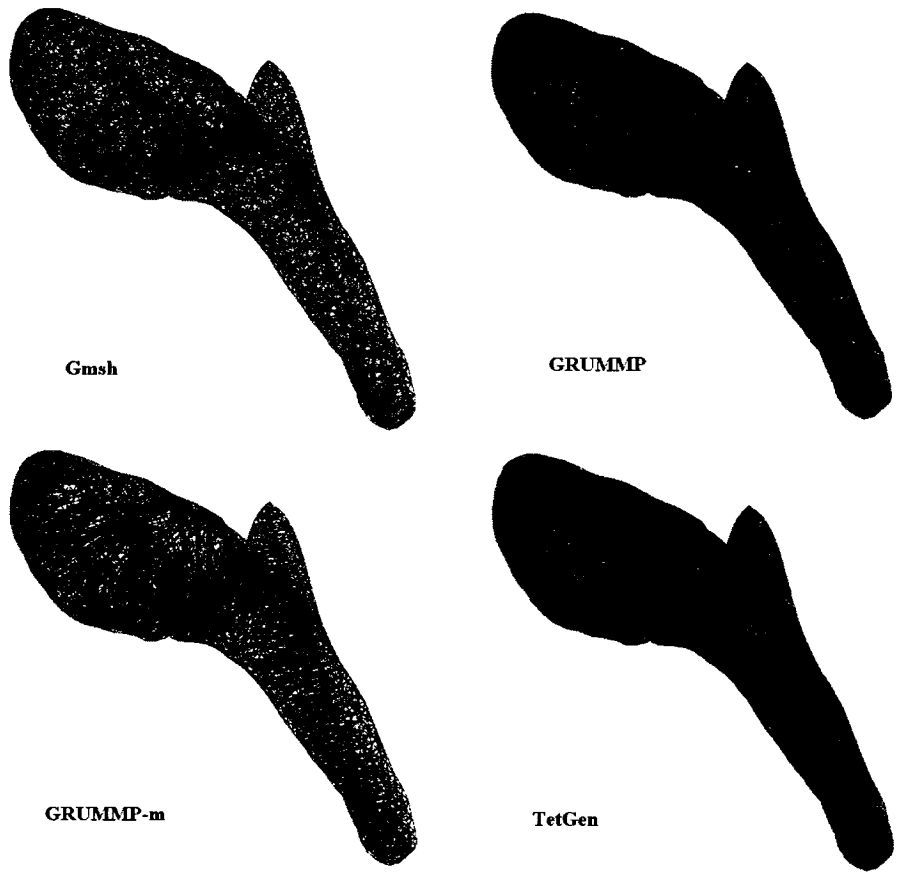


Figure 6.8 Wire-frame views of the malleus model (GiD fails in this case)

6.3.1.2 Cutting-plane views

The cutting-plane viewing method is able to provide a local visual assessment of both mesh density and mesh quality in a volume mesh. The overall mesh was visualized by moving the cutting-plane along the x, y and z axes. In Figures 6.9 to 6.11, the cutting-plane is selected at a plane that is parallel to the x-y plane and passes through the centre of the z length of the box enclosing the model. Aqua is used for the elements intersecting the user-specified plane, and fuchsia is used for the remaining elements beneath the cutting plane. Similar to the results of the wire-frame viewing method, these figures illustrate that GiD, Gmsh and GRUMMP-m produce lower mesh densities than GRUMMP and TetGen do. Furthermore, it is easy to identify that GRUMMP-m generates a number of badly shaped tetrahedra that are long and thin.

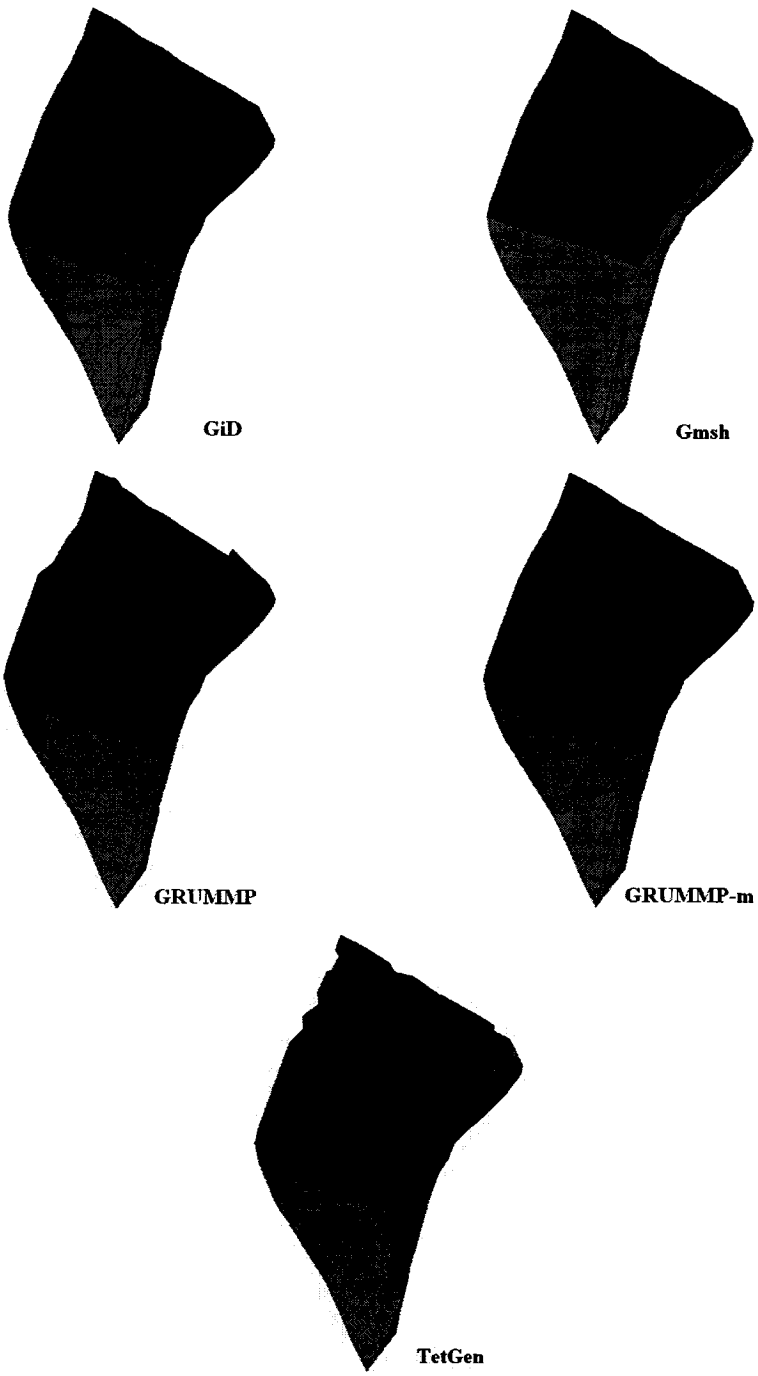


Figure 6.9 Cutting-plane views of the pillat model

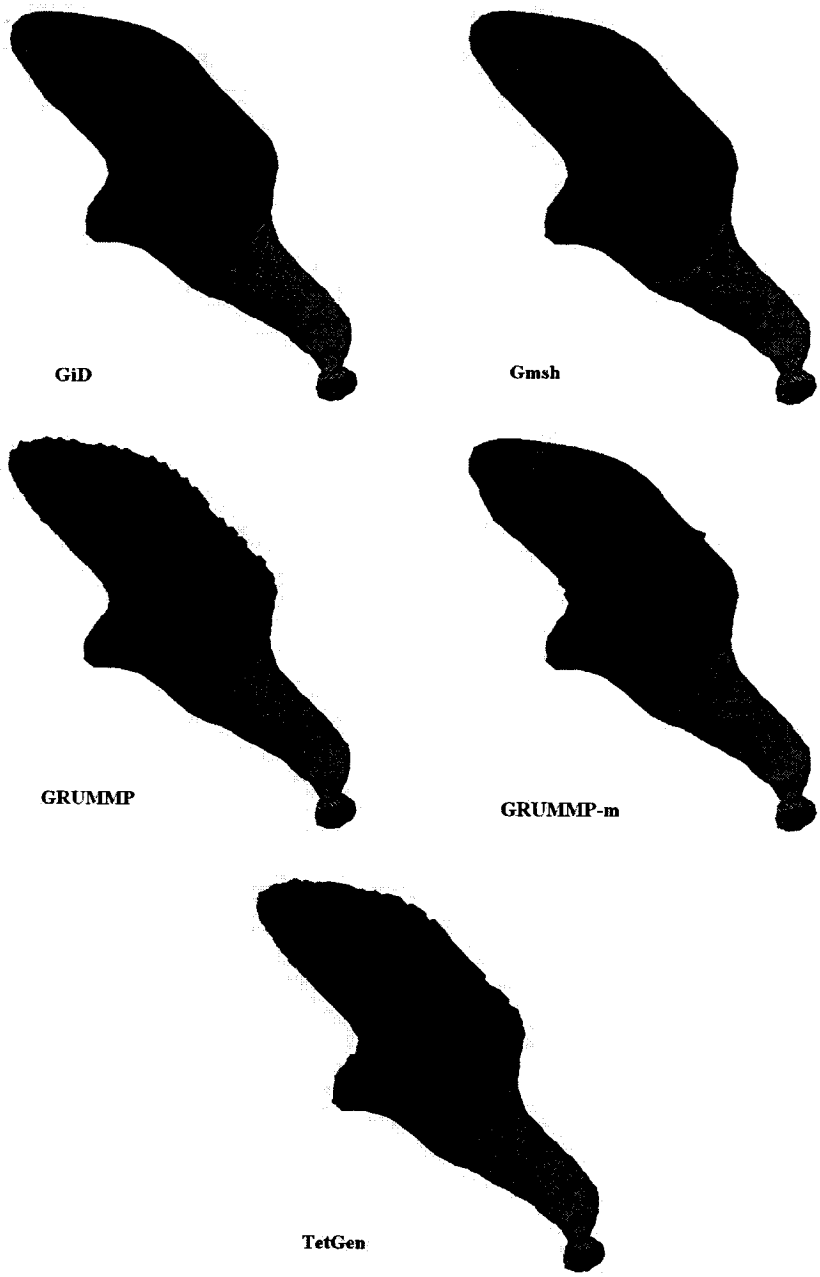


Figure 6.10 Cutting-plane views of the incus model

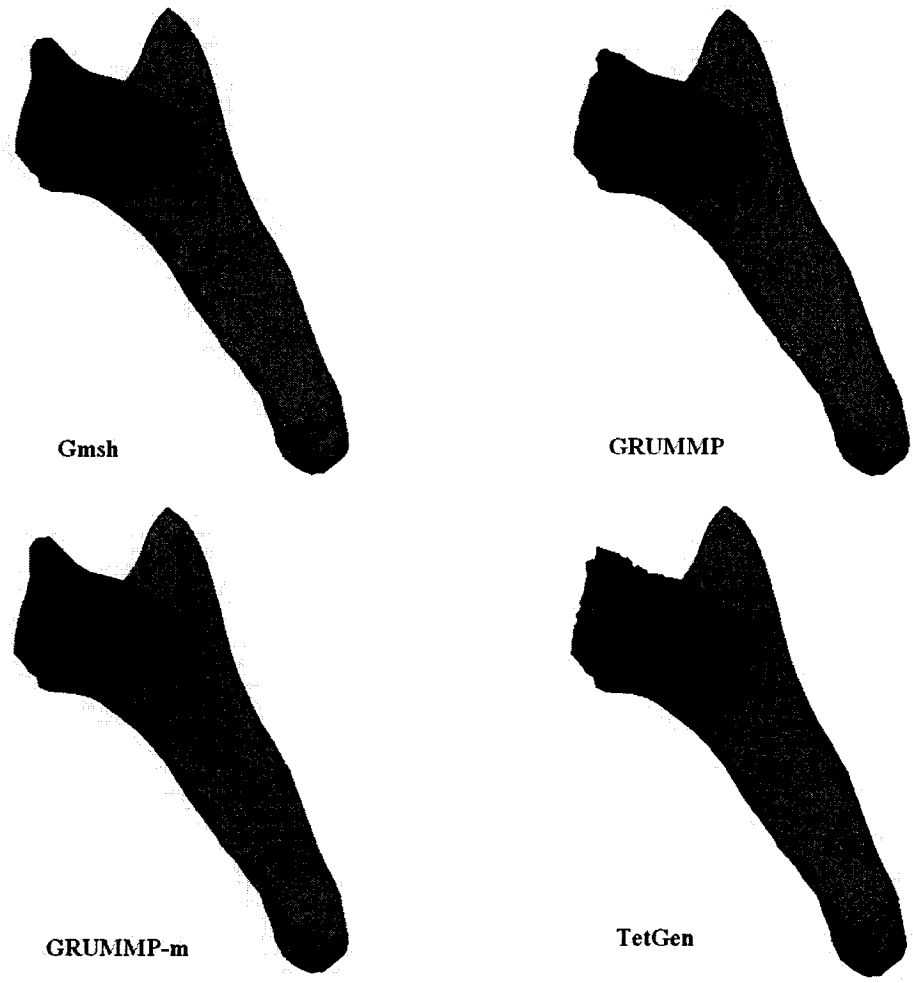


Figure 6.11 Cutting-plane views of the malleus model (GiD fails in this case)

6.3.1.3 Discussion

The two visualization methods are able to provide limited visual assessments of mesh density. They may be used to visually evaluate mesh quality by identifying the presence of badly shaped elements in a volume mesh.

6.3.2 Histograms of shape and size measures

Histograms of the three shape measures and two size measures and the statistical information about these measures, discussed in Section 5.6.2, are presented in this

section. The three shape measures in the evaluation include the minimum solid angle, θ_{min}; the ratio of radii between circumscribed sphere and inscribed sphere, ρ; and the ratio between volume and sum of squares of edge lengths, η. The two size measures include the volume v and the average of edge lengths, l_{avg}. The statistical information about these measures includes the maximum, minimum, mean, standard deviation and skewness.

6.3.2.1 Pillat model

The histograms of the three shape measures are skewed distributions, as shown in Figures 6.12 to 6.14. The histograms of θ_{min} are all skewed to the right. The histograms of ρ and η for all programmes except GRUMMP-m are skewed to the left, and they are similar to each other for each programme.

The order GRUMMP>Gmsh>TetGen>GiD>GRUMMP-m is obtained from Tables 6.4 to 6.6 for the mean values of all three shape measures. TetGen is the programme that generates both the best and worst values of all three measures, which may explain why it is in the third position in the above order. Different orders are obtained when considering the standard deviations across the three measures. Different orders are also obtained when comparing the skewness values across the three measures. GRUMMP-m is an exception because it generates positive skewness values for all three measures. Considering the percentage of tetrahedra with values less than 0.1 for the three measures, one obtains the result that GRUMMP generates the smallest percentages and GRUMMP-m generates the largest percentages, while GiD, Gmsh and TetGen present themselves in different orders. Considering the percentage of tetrahedra with values larger than 0.9 for the three measures, one obtains the result that GRUMMP generates the largest percentages and GRUMMP-m generates the smallest percentages, while GiD, Gmsh and TetGen present themselves in different orders.

The mean values of the two size measures decrease as the mesh density increases. For the volume histograms in this section and the next two sections, the horizontal axis uses a logarithmic scale, and v is scaled by 10^{12} and l_{avg} is scaled by 10^4. One obtains the result for both size measures that GRUMMP and TetGen generate smaller mean

values than the others because of their high mesh densities.

It can be seen in Tables 6.7 and 6.8 that TetGen generates the smallest minimum sizes and GRUMMP-m generates the largest maximum sizes for the two size measures. The order GRUMMP<TetGen<Gmsh<GiD<GRUMMP-m is obtained for the standard deviations of v with Gmsh and GiD having very similar values. A different order, GRUMMP<TetGen<GiD<Gmsh<GRUMMP-m, which is different from the above order only in swapping Gmsh and GiD, is obtained for the standard deviations of l_{avg}. The results based on the standard deviation for both size measures imply that GRUMMP generates the most uniform meshes and GRUMMP-m generates the least uniform meshes. Considering the skewness values, Gmsh generates the largest positive values for the two size measures, which implies that its histograms have the longest right tails.

Overall, GRUMMP is the best and GRUMMP-m is the worst programme in terms of the mean values for the three shape measures and the standard deviations of the two size measures. TetGen is the third best programme in terms of the three shape measures and the second best in terms of the two size measures. Since neither GRUMMP nor TetGen can preserve the boundary-surface mesh, Gmsh becomes the best programme.

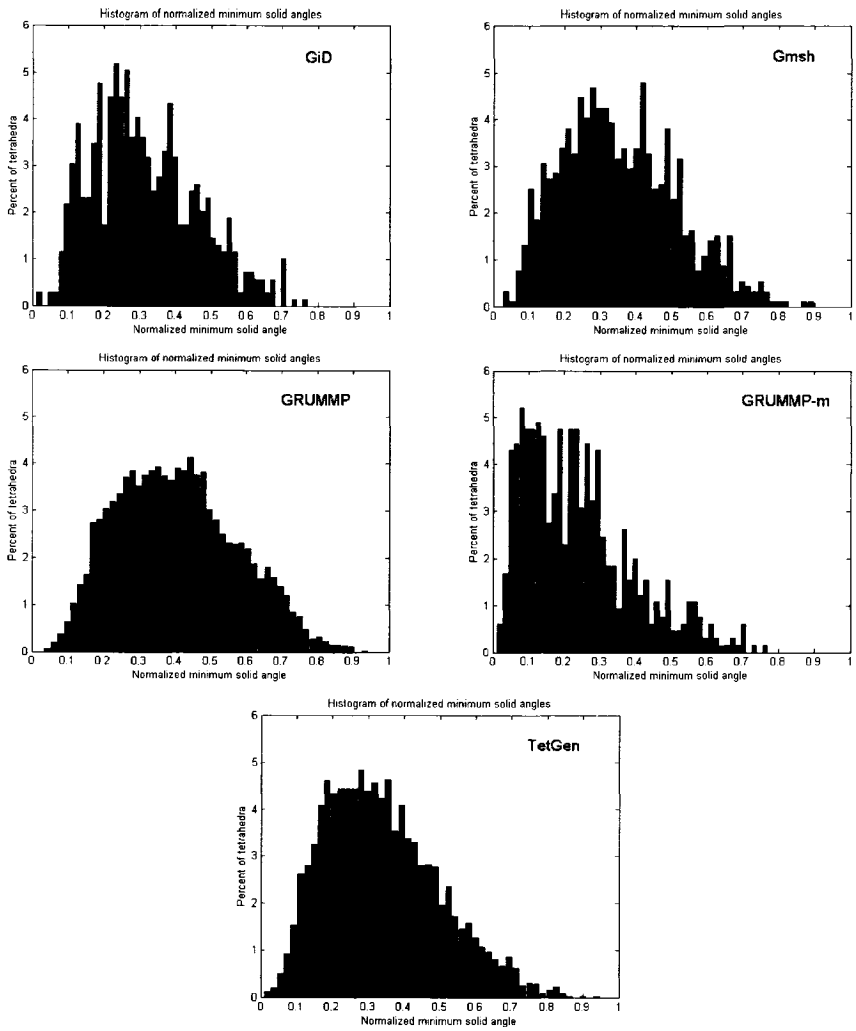


Figure 6.12 Histograms of θ_{min} for the pillat model

Table 6.4 Statistical information about θ_{min} for the pillat model

θ_{min}	GiD	Gmsh	GRUMMP	GRUMMP-m	TetGen
max	0.771	0.897	0.940	0.771	0.947
min	0.121	0.026	0.035	0.121	0.008
mean	0.313	0.352	0.406	0.238	0.338
std	0.147	0.161	0.168	0.152	0.159
skewness	0.536	0.437	0.323	0.932	0.562
< 0.1	24 (3.46%)	30 (3.26%)	47 (0.10%)	129 (19.72%)	139 (3.02%)
> 0.9	0 (0%)	0 (0%)	1 (0.02%)	0 (0%)	1 (0.02%)

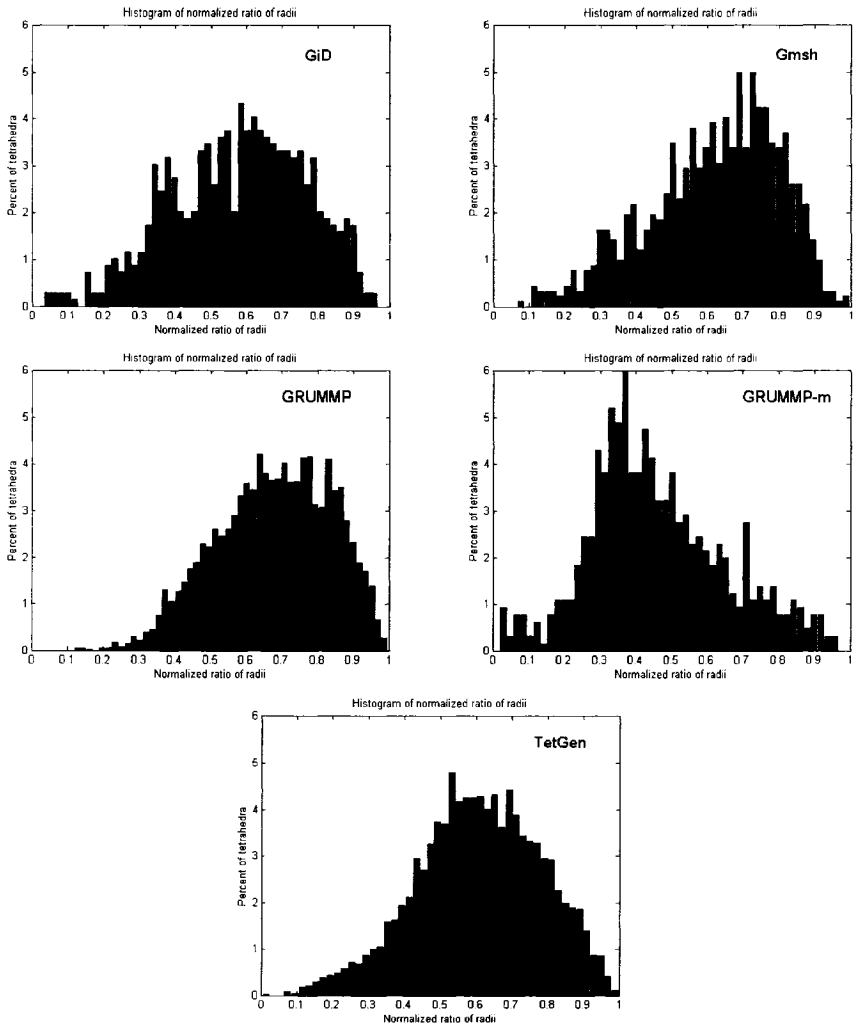


Figure 6.13 Histograms of ρ for the pillat model

Table 6.5 Statistical information about ρ for the pillat model

ρ	GiD	Gmsh	GRUMMP	GRUMMP-m	TetGen
max	0.965	0.9958	0.996	0.965	0.998
min	0.035	0.068	0.119	0.020	0.005
mean	0.576	0.618	0.677	0.460	0.607
std	0.188	0.181	0.160	0.190	0.174
skewness	-0.306	-0.540	-0.328	0.371	-0.252
< 0.1	8 (1.16%)	1 (0.11%)	0 (0%)	18 (2.75%)	6 (0.13%)
> 0.9	14 (2.02%)	21 (2.28%)	325 (6.90%)	10 (1.53%)	164 (3.56%)

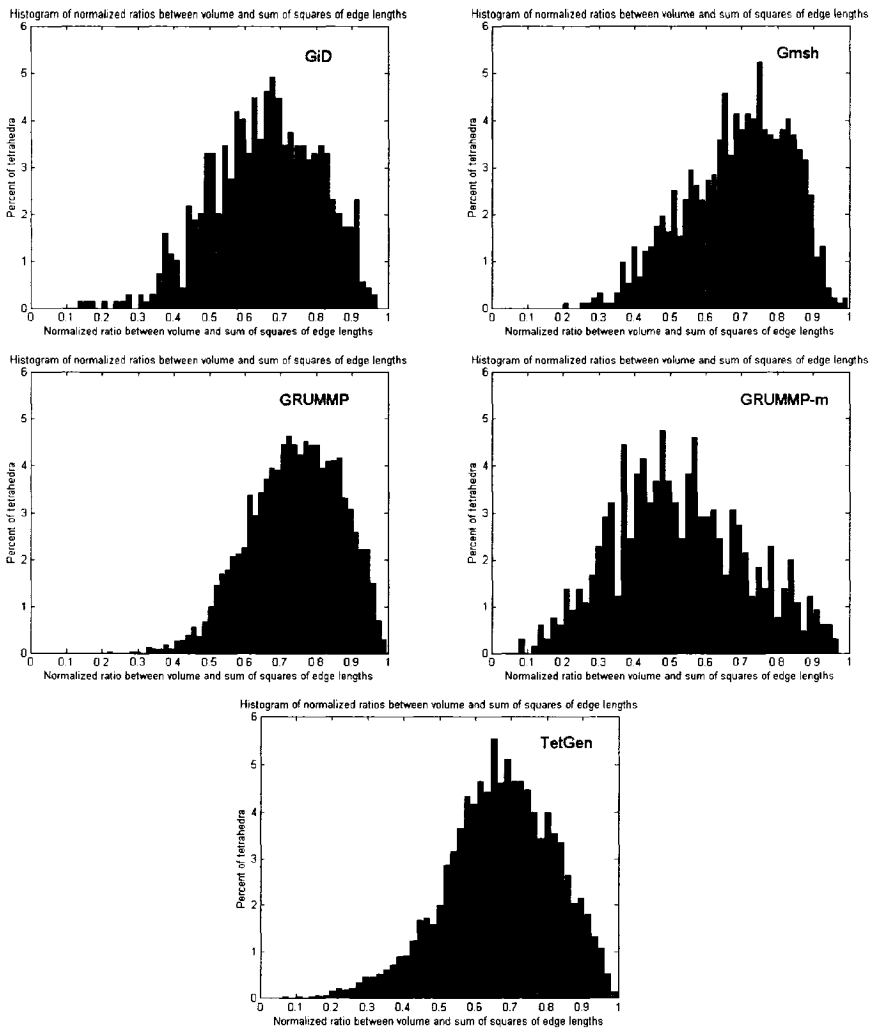


Figure 6.14 Histograms of η for the pillat model

Table 6.6 Statistical information about η for the pillat model

η	GiD	Gmsh	GRUMMP	GRUMMP-m	TetGen
max	0.972	0.997	0.997	0.972	0.999
min	0.130	0.199	0.213	0.074	0.061
mean	0.654	0.686	0.738	0.526	0.669
std	0.150	0.145	0.185	0.127	0.150
skewness	-0.313	-0.456	-0.350	0.184	-0.393
< 0.1	0 (0%)	0 (0%)	0 (0%)	2 (0.31%)	1 (0.02%)
> 0.9	26 (3.75%)	32 (3.48%)	470 (9.96%)	16 (2.45%)	251 (5.45%)

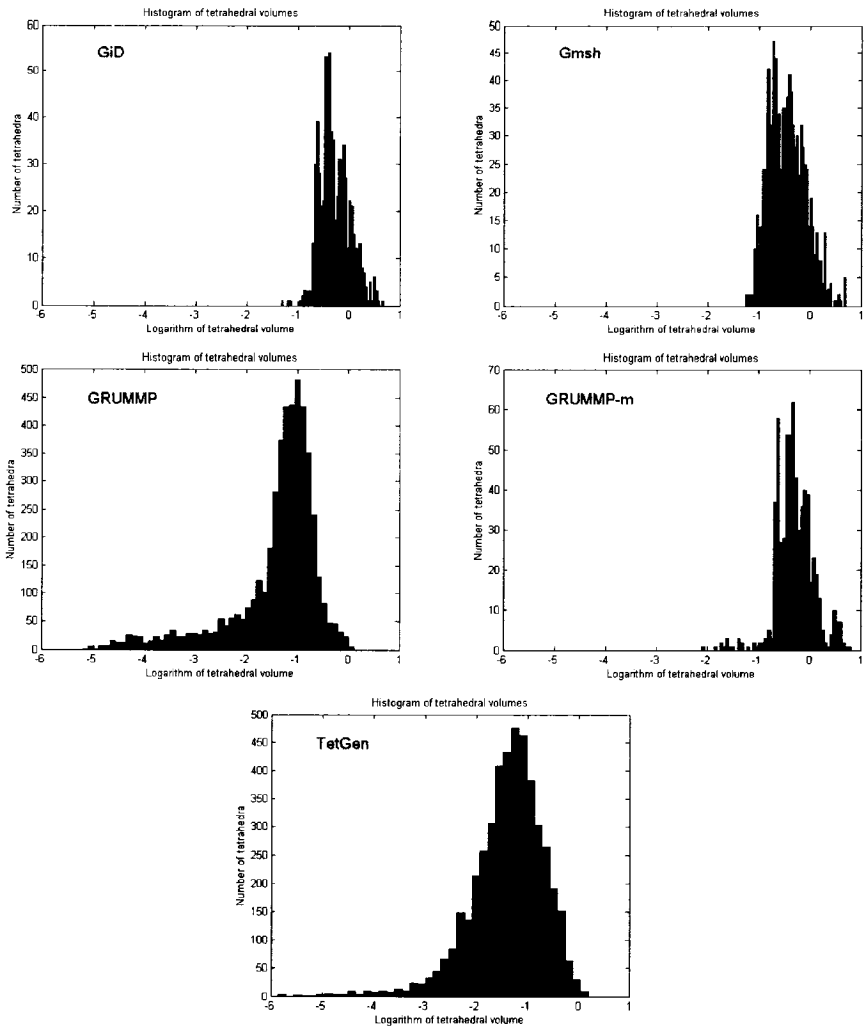


Figure 6.15 Histograms of v scaled by 10^{12} for the pillat model

Table 6.7 Statistical information about v scaled by 10^{12} for the pillat model

Tetrahedron		GiD	Gmsh	GRUMMP	GRUMMP-m	TetGen
v	**max**	4.866	5.049	1.322	6.546	1.572
	min	0.048	0.057	6.42e-6	0.008	4.27e-8
	mean	0.689	0.520	0.101	0.731	0.104
	std	0.590	0.583	0.127	0.784	0.152
	skewness	2.637	3.839	3.553	3.237	3.288

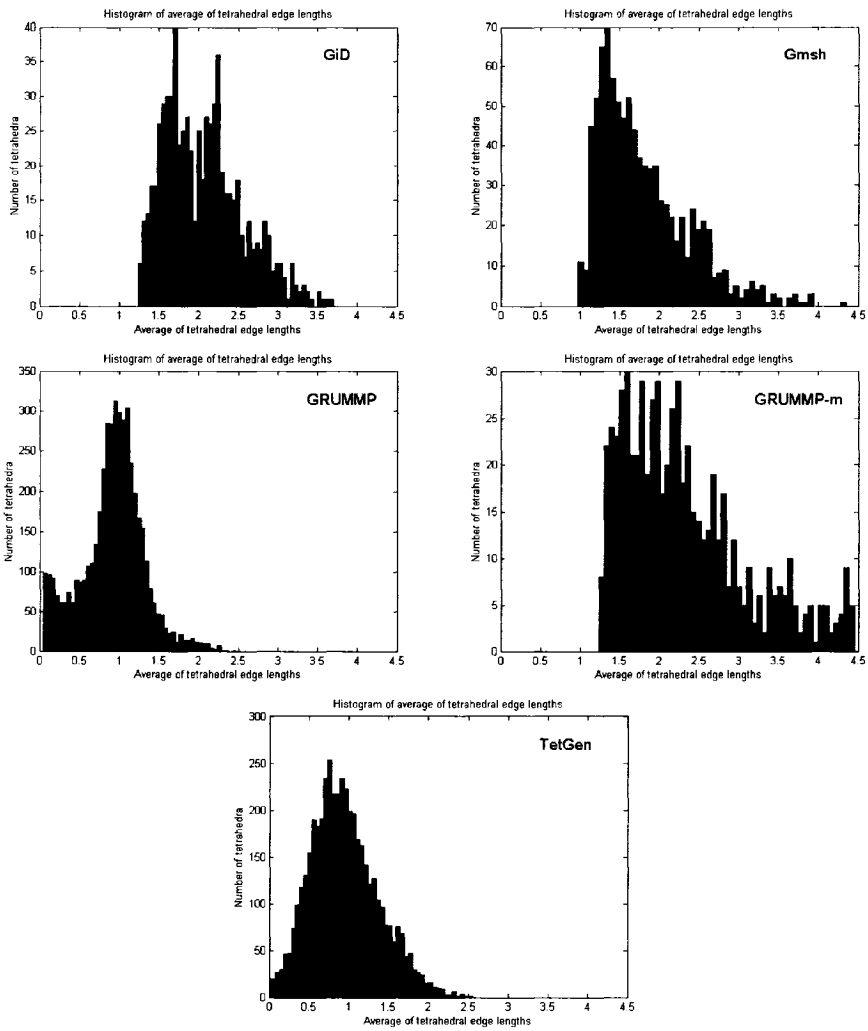


Figure 6.16 Histograms of l_{avg} scaled by 10^4 for the pillat model

Table 6.8 Statistical information about l_{avg} scaled by 10^4 for the pillat model

Tetrahedron		GiD	Gmsh	GRUMMP	GRUMMP-m	TetGen
l_{avg}	**max**	3.694	4.348	2.482	4.450	2.553
	min	1.236	0.970	0.050	1.236	0.028
	mean	2.061	1.795	0.921	2.319	0.946
	std	0.494	0.571	0.394	0.781	0.424
	skewness	0.668	1.161	0.021	0.899	0.512

6.3.2.2 Incus model

Similar to the characteristics of the histograms of the shape measures for the pillat model, the histograms of the three shape measures are again skewed, as illustrated in Figures 6.17 to 6.19. The histograms of θ_{min} are again all skewed to the right. The histograms of ρ and η for all programmes except GRUMMP-m are again skewed to the left, and they are similar to each other for each programme.

From Tables 6.9 to 6.11, GRUMMP generates the largest maximum values for all three shape measures. GRUMMP-m generates the smallest minimum ρ and TetGen generates the smallest minimum θ_{min} and η. The order GRUMMP>Gmsh>GiD>TetGen>GRUMMP-m is obtained for the mean values of θ_{min} as shown in Table 6.9. The order, Gmsh>GRUMMP>GiD>TetGen>GRUMMP-m, which is different from the above order in the first two places, is obtained for both ρ and η from Tables 6.10 and 6.11. The values for Gmsh and GRUMMP are similar in all three cases. Different orders are obtained when considering the standard deviations across all three measures. One obtains the order GiD<Gmsh<GRUMMMP<TetGen<GRUMMP-m when considering the skewness values for both ρ and η. Considering the percentage of tetrahedra with values less than 0.1 for the three shape measures, one obtains the order GRUMMP<Gmsh<TetGen<GiD<GRUMMP-m with GRUMMP and Gmsh being very similar. Considering the percentage of tetrahedra with values larger than 0.9, one obtains the order Gmsh>GRUMMP>GiD>TetGen>GRUMMP-m for both ρ and η, but a different order GRUMMP>Gmsh>TetGen>GiD>GRUMMP-m for θ_{min}.

GRUMMP and TetGen generate smaller mean values for the two size measures than the programmes because of their high mesh densities. It can be seen in Tables 6.12 and 6.13 that GRUMMP-m generates the largest maximum size for the two size measures while GRUMMP and TetGen generate the smallest minimum sizes. The order GRUMMP<TetGen<GiD<Gmsh<GRUMMP-m is obtained for the standard deviations of the two size measures. The results based on the standard deviation for both size measures imply that GRUMMP generates the most uniform meshes and GRUMMP-m generates the least uniform meshes. Considering the skewness values, the order TetGen>GRUMMP>GiD>GRUMMP-m>Gmsh is obtained for the two size measures,

which indicate that the histograms for Gmsh are least skewed to the right and the histograms for TetGen are most skewed to the right.

GRUMMP and TetGen are excluded since they are unable to preserve the boundary-surface mesh. Gmsh is again the best programme when evaluating the three shape measures for the incus model. Although Gmsh is after GiD in the rankings for the standard deviations of the two size measures, the values generated by Gmsh and GiD are very close for l_{avg} (0.885 and 0.918, respectively) and practically identical for v (2.942 and 2.943, respectively). As was found for the pillat model, GRUMMP-m is the worst candidate programme. Finally, Gmsh is the best programme in terms of both shape and size measures.

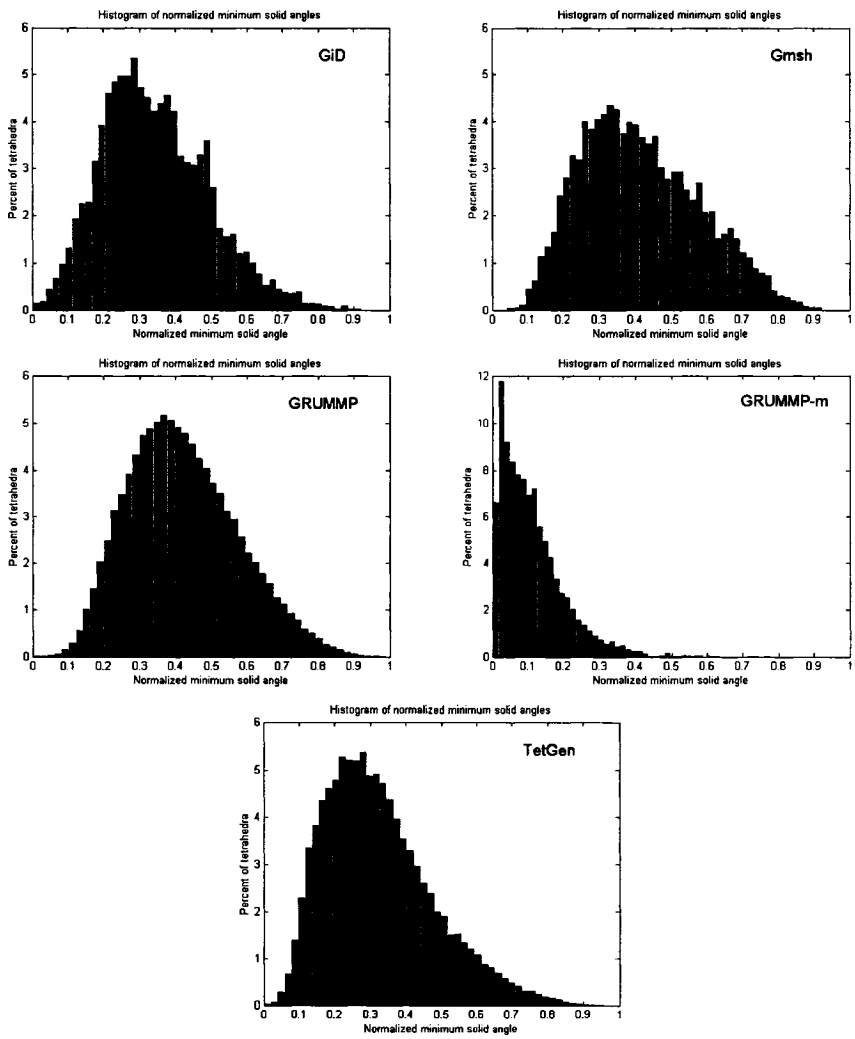


Figure 6.17 Histograms of θ_{min} for the incus model

Table 6.9 Statistical information about θ_{min} for the incus model

θ_{min}	GiD	Gmsh	GRUMMP	GRUMMP-m	TetGen
max	0.919	0.922	0.990	0.774	0.957
min	0.001	0.041	0.002	0.002	9.870e-4
mean	0.344	0.415	0.420	0.111	0.324
std	0.149	0.164	0.152	0.088	0.153
skewness	0.518	0.403	0.407	1.582	0.749
< 0.1	255 (2.84%)	21 (0.28%)	270 (0.25%)	3278 (54.09%)	1344 (2.80%)
> 0.9	2 (0.02%)	5 (0.06%)	91 (0.08%)	0 (0%)	14 (0.03%)

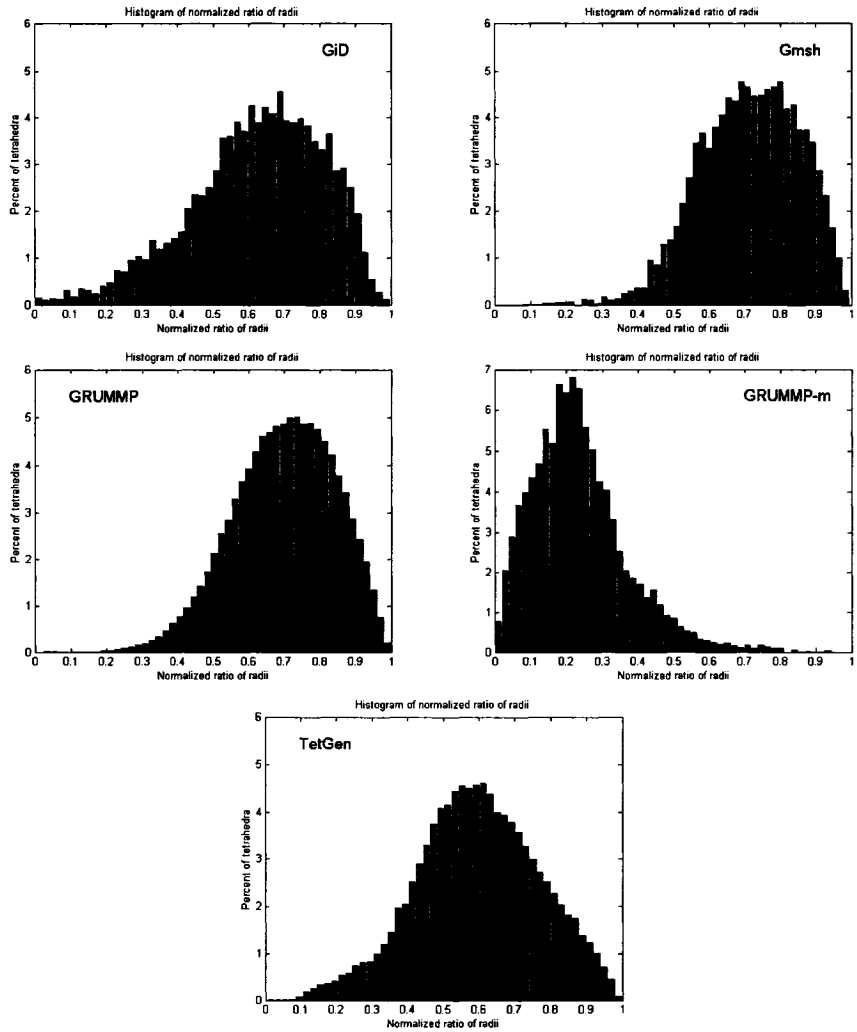


Figure 6.18 Histograms of ρ for the incus model

Table 6.10 Statistical information about ρ for the incus model

ρ	GiD	Gmsh	GRUMMP	GRUMMP-m	TetGen
max	0.997	0.994	1.000	0.942	0.999
min	8.516e-4	0.076	0.025	1.509e-4	0.006
mean	0.624	0.713	0.698	0.236	0.595
std	0.185	0.140	0.143	0.135	0.172
skewness	-0.582	-0.422	-0.370	1.060	-0.163
< 0.1	71 (0.79%)	1 (0.01%)	11 (0.01%)	892 (14.72%)	45 (0.09%)
> 0.9	324 (3.60%)	654 (8.57%)	7655 (6.99%)	2 (0.03%)	1672 (3.48%)

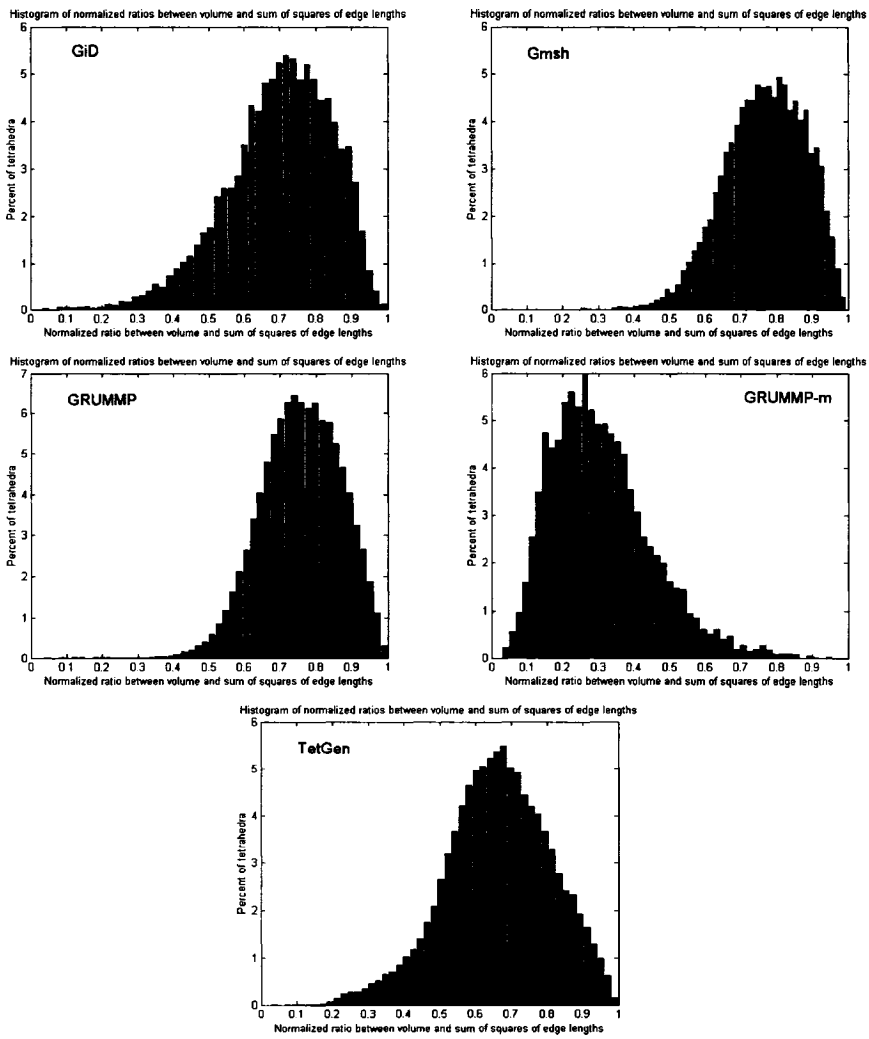


Figure 6.19 Histograms of η for Incus model

Table 6.11 Statistical information about η for Incus model

η	GiD	Gmsh	GRUMMP	GRUMMP-m	TetGen
max	0.997	0.994	1.000	0.955	0.999
min	0.032	0.248	0.036	0.030	0.026
mean	0.696	0.765	0.756	0.307	0.662
std	0.147	0.113	0.110	0.143	0.147
skewness	-0.658	-0.404	-0.304	0.842	-0.300
< 0.1	8 (0.09%)	0(0%)	11 (0.01%)	181 (2.30%)	6 (0.01%)
> 0.9	527 (5.86%)	938 (12.30%)	10786(9.85%)	3 (0.05%)	2332(4.86%)

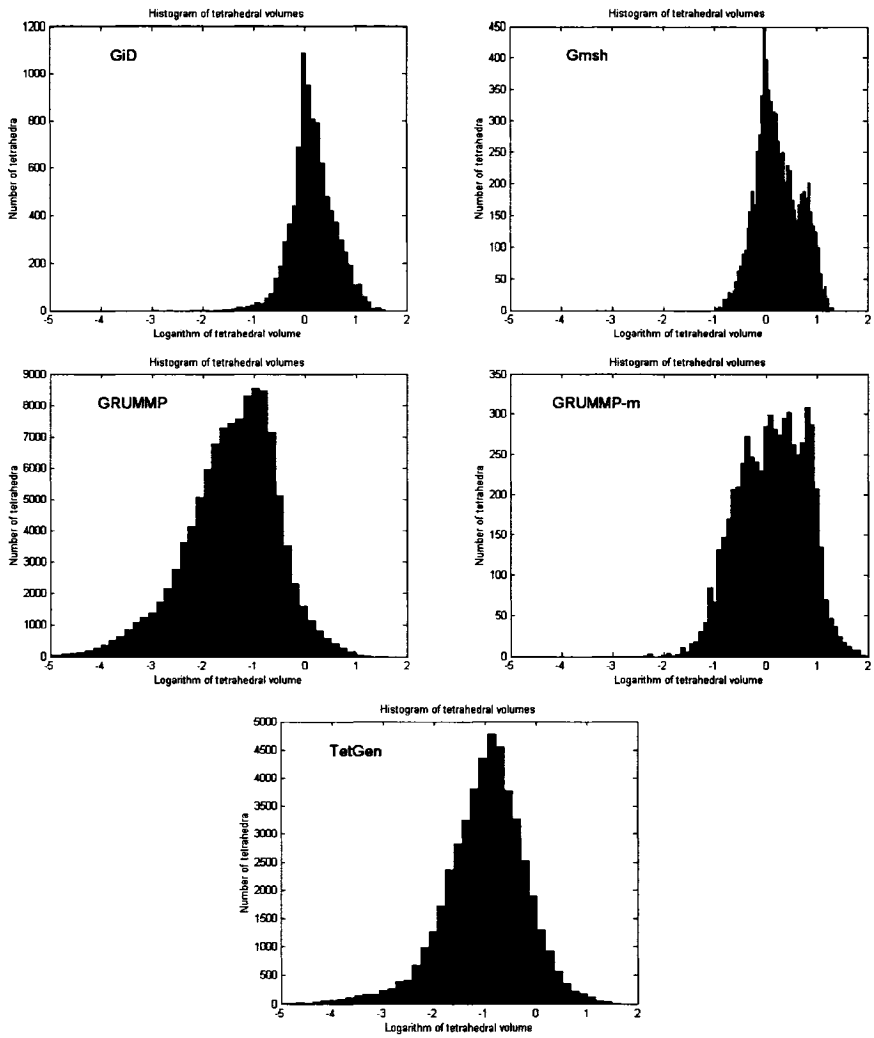


Figure 6.20 Histograms of v scaled by 10^{12} for the incus model

Table 6.12 Statistical information about v scaled by 10^{12} for the incus model

Tetrahedron		GiD	Gmsh	GRUMMP	GRUMMP-m	TetGen
v	**max**	37.782	21.385	19.390	132.190	45.863
	min	0.001	0.108	3.194e-7	0.004	3.142e-7
	mean	2.338	2.755	0.192	3.469	0.438
	std	2.942	2.943	0.614	6.008	1.444
	skewness	3.930	1.963	10.257	6.493	11.149

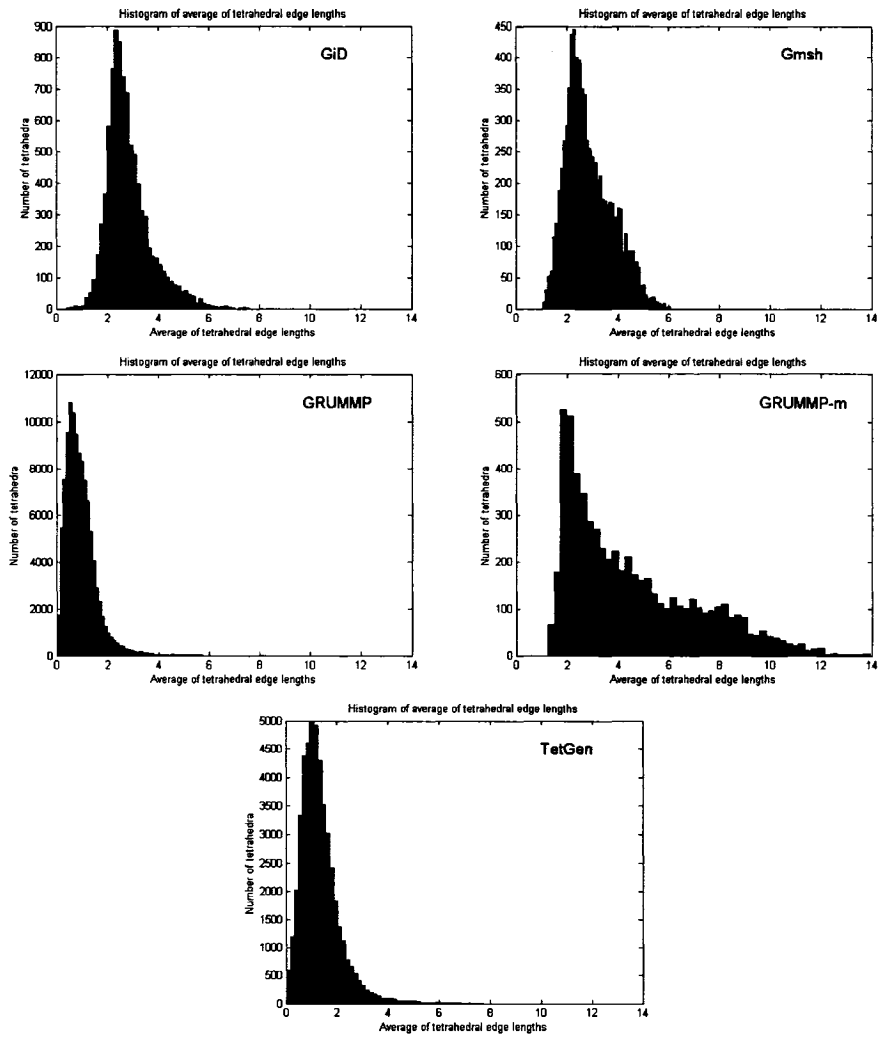


Figure 6.21 Histograms of l_{avg} scaled by 10^4 for the incus model

Table 6.13 Statistical information about l_{avg} scaled by 10^4 for the incus model

Tetrahedron		GiD	Gmsh	GRUMMP	GRUMMP-m	TetGen
l_{avg}	**max**	7.627	6.068	5.766	13.918	7.734
	min	0.403	1.044	0.026	1.233	0.039
	mean	2.838	2.875	0.941	4.537	1.351
	std	0.885	0.918	0.609	2.561	0.801
	skewness	1.254	0.648	1.799	0.9233	1.948

6.3.2.3 Malleus model

GiD fails to generate a volume mesh for the malleus model. Figures 6.22 to 6.24 show that the histograms of the three shape measures are still skewed and have the same characteristics as those for the pillat model and the incus model.

The order Gmsh>GRUMMP>TetGen>GRUMMP-m is obtained from Tables 6.14 to 6.16 for the three shape measures in terms of the mean values and the percentages of tetrahedra with values larger than 0.9. The orders are not the same when considering the percentages of tetrahedra with values smaller than 0.1, the standard deviations, or the skewness values. GRUMMP-m has the largest percentage of tetrahedra with values smaller than 0.1 and the largest skewness for all the shape measures, which indicates that there are more element qualities close to 0 than for the other candidates.

Similar to the results of the two size measures for the pillat model, TetGen generates the smallest minimum sizes and GRUMMP-m generates the largest maximum sizes for the two size measures, as shown in Tables 6.17 and 6.18. One obtains the order GRUMMP<TetGen<Gmsh<GRUMMP-m in terms of the standard deviations for the two size measures.

TetGen and GRUMMP again are excluded because they cannot preserve the boundary-surface mesh. The results of the three shape measures show that Gmsh generates the best meshes and GRUMMP-m generates the worst meshes. Gmsh still generates more uniform meshes than GRUMMP-m does. Overall, Gmsh is the best programme when evaluating the malleus model.

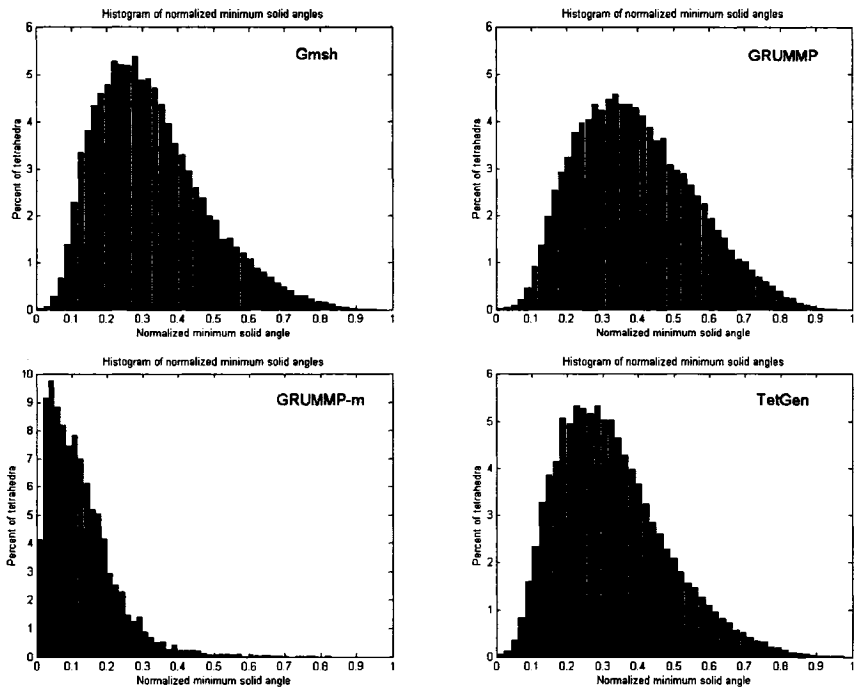


Figure 6.22 Histograms of θ_{min} for the malleus model

*(**GiD** fails in this case)*

Table 6.14 Statistical information about θ_{min} for the malleus model

θ_{min}	Gmsh	GRUMMP	GRUMMP-m	TetGen
max	0.937	0.960	0.827	0.978
min	0.0077	0.002	4.374e-4	0.001
mean	0.4027	0.393	0.125	0.325
std	0.160	0.163	0.096	0.153
skewness	0.371	0.426	1.726	0.725
< 0.1	93 (1.08%)	483 (0.95%)	3254 (47.77%)	1810 (3.05%)
> 0.9	7 (0.08%)	28 (0.06%)	0 (0%)	21 (0.04%)

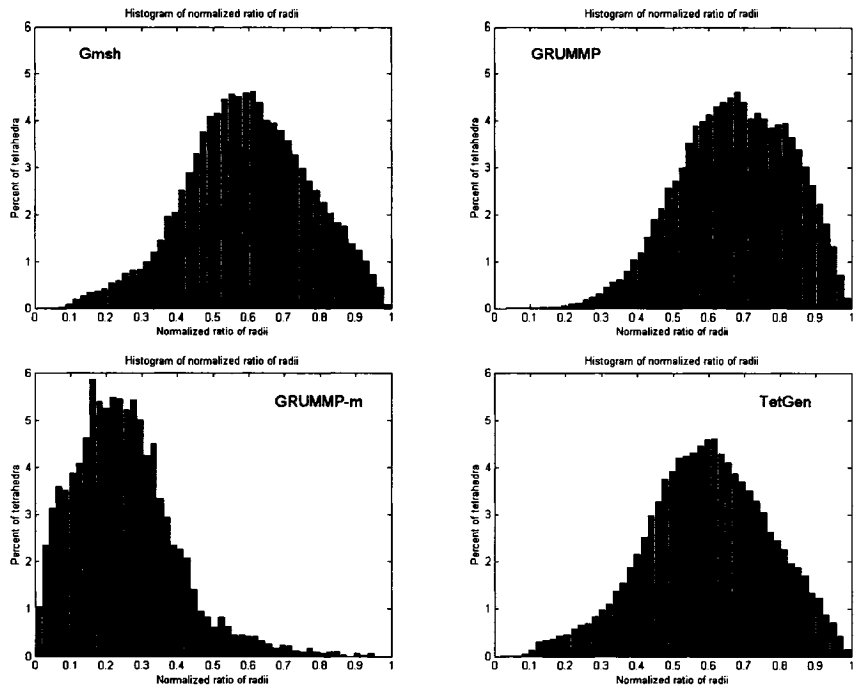


Figure 6.23 Histograms of ρ for the malleus model

*(**GiD** fails in this case)*

Table 6.15 Statistical information about ρ for the malleus model

θ_{min}	Gmsh	GRUMMP	GRUMMP-m	TetGen
max	0.993	0.998	0.951	0.998
min	0.003	0.032	1.018e-5	0.002
mean	0.700	0.672	0.250	0.594
std	0.150	0.156	0.145	0.123
skewness	-0.868	-0.286	0.974	-0.197
< 0.1	47 (0.55%)	63 (0.03%)	999 (14.67%)	56 (0.09%)
> 0.9	577 (6.71%)	3275 (6.46%)	9 (0.13%)	2000 (3.37%)

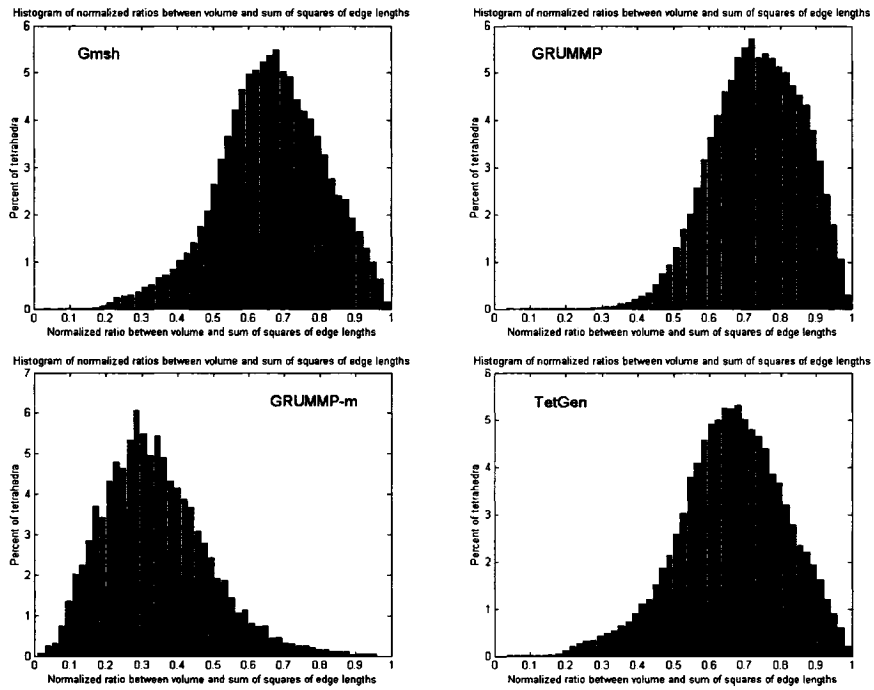


Figure 6.24 Histograms of η for the malleus model

*(**GiD** fails in this case)*

Table 6.16 Statistical information about η for the malleus model

θ_{min}	Gmsh	GRUMMP	GRUMMP-m	TetGen
max	0.994	0.999	0.960	0.999
min	0.069	0.035	0.009	0.034
mean	0.754	0.732	0.338	0.661
std	0.123	0.126	0.147	0.149
skewness	-0.943	-0.331	0.695	-0.352
< 0.1	5 (0.06%)	6 (0.12%)	162 (2.38%)	16 (0.03%)
> 0.9	851 (9.90%)	4607 (9.09%)	10 (0.15%)	2796 (4.72%)

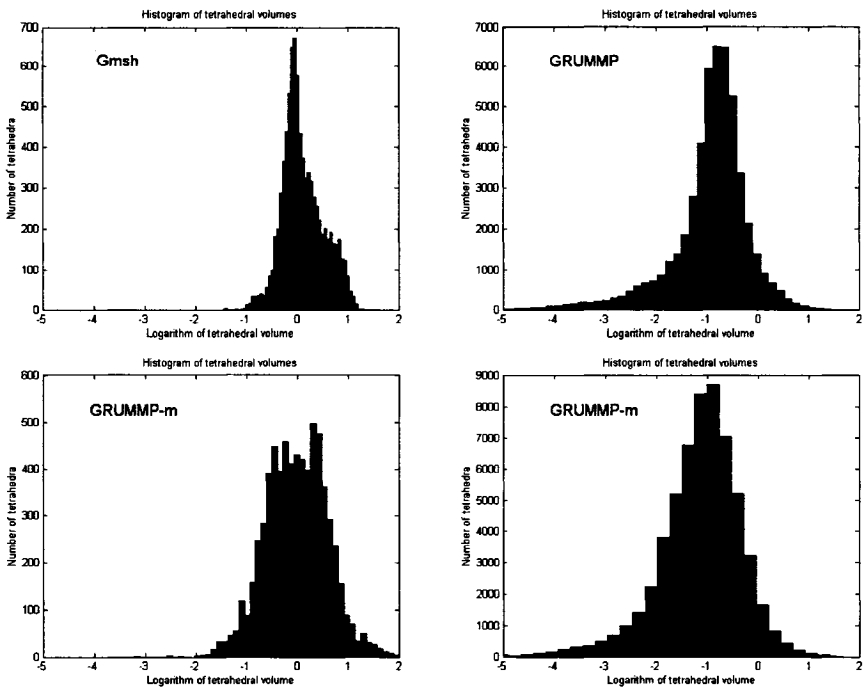


*Figure 6.25 Histograms of v scaled by 10^{12} for the malleus model (**GiD** fails in this case)*

Table 6.17 Statistical information about v scaled by 10^{12} for the malleus model

Tetrahedron		Gmsh	GRUMMP	GRUMMP-m	TetGen
v	**max**	20.269	28.021	169.768	45.877
	min	0.036	4.765e-7	6.010e-4	3.270e-11
	mean	2.077	0.353	2.622	0.301
	std	2.364	0.935	6.586	1.099
	skewness	2.350	10.928	10.404	14.303

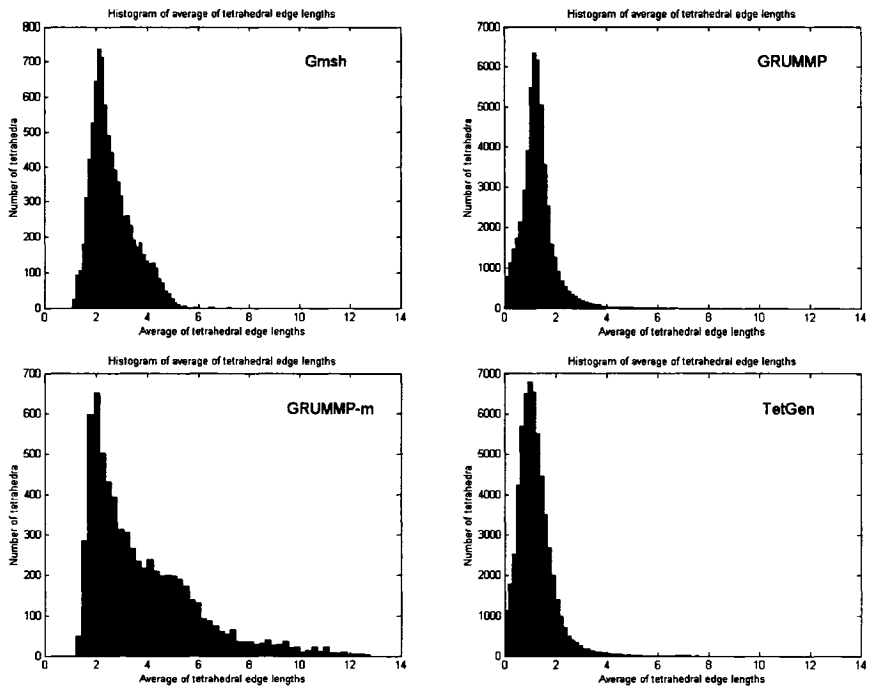


*Figure 6.26 Histograms of l_{avg} scaled by 10^4 for the malleus model (**GiD** fails in this case)*

Table 6.18 Statistical information about l_{avg} scaled by 10^4 for the malleus model

Tetrahedron		Gmsh	GRUMMP	GRUMMP-m	TetGen
l_{avg}	**max**	7.260	6.761	12.805	7.607
	min	1.085	0.043	1.213	0.001
	mean	2.640	1.300	3.835	1.195
	std	0.827	0.645	2.105	0.704
	skewness	0.882	1.622	1.381	1.954

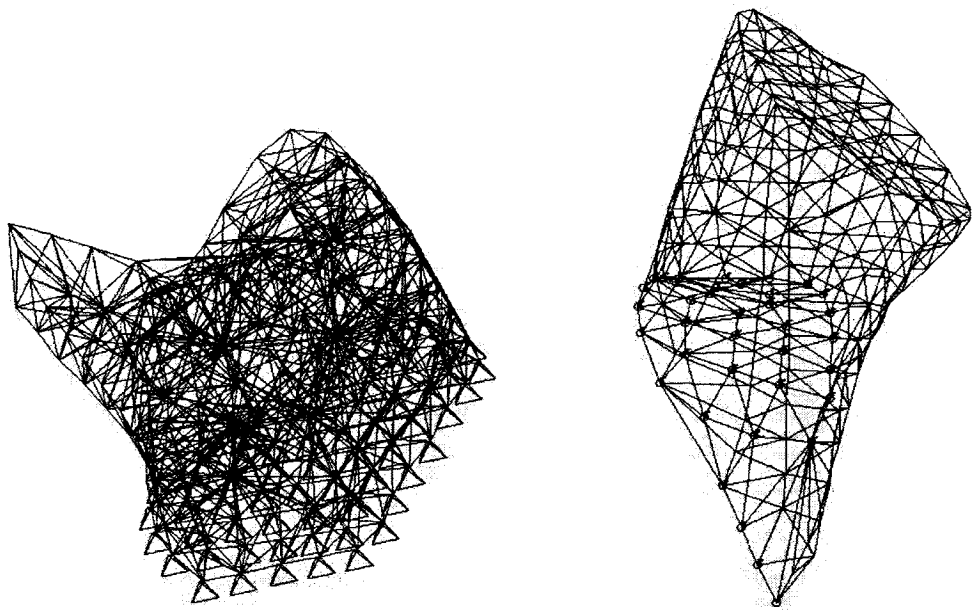
6.4 Solution residuals

Solution residuals, discussed in Chapter 5.6.3, offer a way to evaluate the overall mesh quality that is based on the finite-element solution. The model for this evaluation is the pillat model. The mechanical parameters are defined as follows:

- Boundary conditions: All nodes connecting to the cavity wall are fully clamped, as shown in Figure 6.27(a) where the red tetrahedra represent those

nodes.

- Load conditions: A pressure load of 1.0 Pa is applied on the interface between the pillat model and the incus model, as shown in Figure 6.27(b) where the red circles represent those nodes on the interface.
- Material properties: Young's modulus is 20 MPa and Poisson's ratio is 0.3



(a) *(b)*

Figure 6.27 Boundary conditions (a) and load conditions (b) for the pillat model

The results of the evaluation in the previous section indicate that GiD, Gmsh and GRUMMP-m are able to preserve the boundary-surface mesh for some or all models. In contrast, GRUMMP and TetGen are unable to meet this criterion for any model. Therefore, only GiD, Gmsh and GRUMMP-m are evaluated for the finite-element mesh-quality measures in this section and the following two sections.

As shown in Figure 6.28, most residuals are small. The larger residuals occur on those degrees of freedom that correspond to where the load conditions are applied.

From the root mean squares of residuals in Table 6.19, one obtains the order *Gmsh<GiD<GRUMMP-m*. The finite-element solution using the volume mesh generated by Gmsh results in the smallest root mean square of residuals. In contrast, the finite-element solution using the volume mesh generated by GRUMMP-m results in the largest root mean square of residuals. Therefore, Gmsh is able to generate the best mesh

and GRUMMP-m generates the worst mesh. It should be mentioned that the three root mean squares of residuals differ by a factor of less than 2.

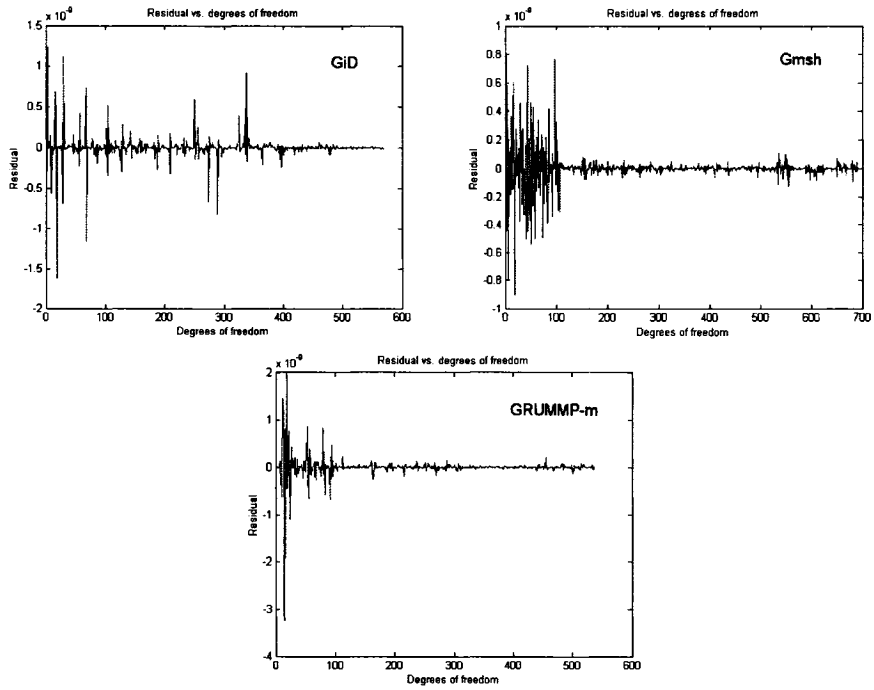


Figure 6.28 Residuals vs. degrees of freedom

Table 6.19 Root mean squares of residuals for the pillat model

Software	Degrees of freedom	RMS(residuals)
GiD	570	0.031
Gmsh	537	0.026
GRUMMP-m	693	0.043

6.5 Condition number

The condition number, discussed in Chapter 5.6.3, is another way to evaluate the overall mesh quality. Its computation requires material properties and boundary conditions in order to formulate the system stiffness matrix.

The condition numbers listed in Table 6.20 for the three programmes are very close to each other. As discussed in Section 6.3, Gmsh and GiD are able to generate good volume meshes, and GRUMMP-m generates the worst mesh, composed of many long and thin tetrahedra. It is not surprising that the finite-element solutions using the volume

meshes generated by Gmsh and GiD have small condition numbers. The finite-element solution using the volume mesh generated by GRUMMP-m, however, would have been expected to have a larger condition number, which is actually not true.

Table 6.20 Condition numbers for the pillat model

Software	Condition number
GiD	9792
Gmsh	7553
GRUMMP-m	8646

The condition number could not be used to evaluate the overall mesh quality because the finite-element solution using a bad mesh also had a small condition number.

6.6 Closeness to the exact solution

As discussed in Section 4.5.3, the exact solution is generally more accurately approximated as mesh density increases, so the exact solution can be estimated by doing a convergence test. If an approximate solution produced by a particular mesh is closer to that estimated exact solution than are the solutions produced by the other meshes, then that mesh is better than the others.

As shown in Figure 6.29, for this test, the mechanical parameters for the thin-block model are defined as follows:

- Boundary conditions: One face is fully clamped
- Load conditions: A compression load of 1 N/m² is applied on opposite face
- Material properties: Young's modulus is 20 MPa and Poisson's ratio is 0.3

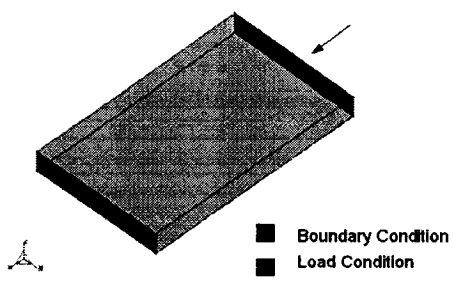


Figure 6.29 Mechanical parameters for the thin-block model

Using COMSOL (COMSOL Inc. 2006) as discussed in Section 5.6.4, the approximate solution monotonically converges to the exact solution as the number of elements increases, as can be seen in Figure 6.30. The simulation results based on the volume meshes generated by the candidate programmes suggest the order Gmsh>GiD>GRUMMP-m. The solution by Gmsh is closer to the exact solution than the others are. The order is actually the same as the order obtained from the three shape measures discussed in Section 6.3.

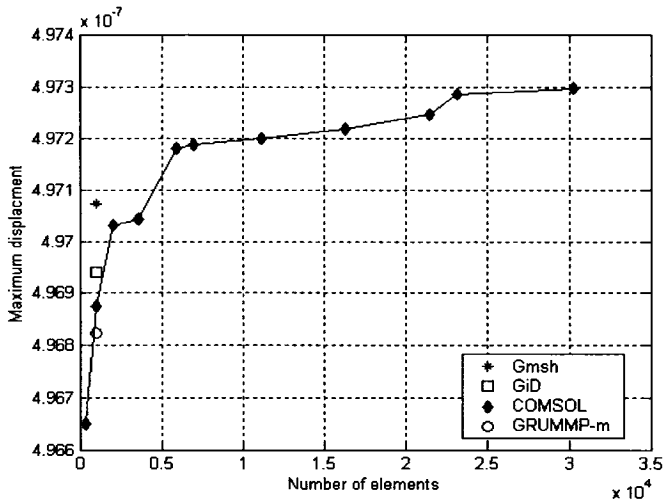


Figure 6.30 Comparing closeness to convergence curve for the thin-block model

6.7 Conclusions

The evaluation involved comparisons in a number of key respects: visual inspections, mesh quality, ability to preserve boundary-surface mesh, robustness, time required for the generation of volume mesh, and cost of software. The overall results of the evaluation are summarized in Table 6.21.

Visual inspections suggest that GRUMMP-m generates bad meshes and the other programmes produce good meshes. According to the histograms of shape measures, Gmsh and GRUMMP are able to generate the best meshes, followed by TetGen, GiD and GRUMMP-m. GiD is the only one that was found not to be robust because it failed for the malleus model, and it is also the only commercial software among the candidate software. Only Gmsh is capable of preserving the boundary-surface mesh for all models.

Table 6.21 Overall results of evaluation

Software	GiD	Gmsh	GRUMMP	GRUMMP-m	TetGen
Visualization	Good	Good	Good	Bad	Good
Mesh quality	Good	Best	Best	Bad	Better
Preserving boundary-surface mesh	Sometimes	Yes	No	Sometimes	No
Robustness	No	Yes	Yes	Yes	Yes
Time	Medium	Slow	Fast	Fast	Fast
Cost	Non-free	Free	Free	Free	Free

TetGen and GRUMMP are similar in that they generate volume meshes with high density, in which the boundary-surface mesh is not preserved. Because of this, TetGen and GRUMMP are excluded as final choices for our project although they may generate good meshes.

Considering the finite-element mesh-quality measures, the analysis of solution residuals could be a useful method to evaluate the overall mesh quality, although the root mean squares of the residuals of GiD, Gmsh and GRUMMP-m varied by less than a factor of 2. The closeness to the convergence curve is also an effective method to evaluate the overall mesh quality, but it is computationally expensive as the convergence curve must be obtained for the comparison. The above two methods reach the same result, that Gmsh generates the best mesh and GRUMMP-m generates the worst mesh. The analysis of the condition number indicates that the condition number is not a good mesh-quality indicator because GRUMMP-m produced an unexpectedly small value of condition number.

Comparing every aspect of the evaluation, GRUMMP-m generates the worst mesh. This conclusion can easily be drawn just on the basis of visual inspections. As mentioned previously, GRUMMP-m is a modification of GRUMMP for the purpose of preserving the boundary-surface mesh, a specific requirement for our project, and the modification violates the design philosophy of GRUMMP. GRUMMP itself produces as high a mesh quality as Gmsh does.

Gmsh is the only candidate that satisfies all of our evaluation criteria as discussed

in Chapter 3. The time required for volume mesh generation is the highest among the candidate software, but it falls within a reasonable range. It should be mentioned that the time spent by Gmsh for 3-D mesh generation includes calling the optimization routine three times, which certainly takes more time than the other candidate programmes.

The results for the three shape measures (θ_{min}, ρ and η) indicate that the mean value of a shape measure is a trustworthy indicator to assess the overall mesh quality. The histograms of the shape measures are skewed, and they present different orders in terms of skewness values for the three measures. The same is true for the standard deviations for the three measures. Meanwhile the percentages of tetrahedra with the normalized measures less than 0.1 or larger than 0.9 offer limited information about the quality of the overall mesh.

The two size measures (v and l_{avg}) are equivalent for characterizing the size of a tetrahedron because the same order was always obtained from their standard deviations for the three models. Since the histograms of the two size measures are also skewed, their standard deviations may not provide reasonable estimates of the extent of mesh uniformity, and they do not predict mesh quality well.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

This project is an attempt to establish a number of criteria for evaluating 3-D mesh-generation software, and to select the 3-D mesh generator that is most suitable for use in our software pipeline for finite-element modelling and simulation of complex natural structures.

The evaluation criteria for our project include the following characteristics: ability to preserve the boundary-surface mesh, volume mesh quality, robustness, time efficiency and cost. Four models are chosen for the evaluation. One model is a simple thin block. The others represent three structures of the human middle ear. One structure is the lateral bundle of the posterior incudal ligament, here referred to simply as pillat. The remaining structures are two ossicles, the incus and malleus.

To import surface triangular meshes generated by our Tr3 programme into the candidate software for 3-D mesh generation, a programme called Fcf was developed to convert the surface definitions describing the models to the native file formats of the mesh-generation programmes, to pre-process the surface triangular meshes to verify that the meshes are closed and consistently oriented, and to post-process the volume tetrahedral meshes to verify that the meshes are topologically correct.

The results of an initial evaluation using a thin-block model provide the guiding information for proper selection of parameters in deciding volume mesh density for the three structures of the middle ear. The boundary-surface meshes were most likely to be preserved in the coarsest volume meshes that were generated by the candidate software.

The results of the evaluation of the preservation of the boundary-surface meshes show that Gmsh is the only programme that is able to preserve the boundary-surface meshes for all models. GiD is able to preserve the boundary-surface meshes for the pillat model and the incus model, but fails to generate a volume mesh at all for the malleus model. GRUMMP and TetGen are unable to preserve the boundary-surface meshes for any of the models. A modified version of GRUMMP, referred to here as GRUMMP-m, is

able to preserve the boundary-surface mesh for the pillat only.

The visual inspection of mesh quality by either the wire-frame viewing method or the cutting-plane viewing method is useful for assessing mesh density. The two methods are not so useful for evaluating mesh quality but may provide qualitative information about mesh quality when the volume mesh has many badly shaped elements.

Three shape-quality measures are investigated in the evaluation. They are θ_{min}, ρ and η, where θ_{min} is the minimum solid angle, ρ is the ratio of the circumscribed sphere radius to the inscribed sphere radius, and η is the ratio between the volume and the sum of squares of the edge lengths, for a tetrahedron. The results from the histograms of the three measures indicate that the three measures are equivalent. The average of any one of these element shape qualities for a volume mesh would be a good indicator for comparing the overall mesh quality.

The results of the histograms of element sizes indicate that the volume and the average edge length of an element are equivalent in characterizing the sizes of elements. The quality order derived from the size histograms is not, however, the same as the order derived from the shape histograms for all models.

The analysis of solution residuals indicates that the root mean square of solution residuals may be a good indicator of mesh quality because it gives the same order as that obtained from the evaluation of the element shape qualities. The condition number of the system stiffness matrix is a very generous indicator of mesh quality and it is computationally very expensive. The analysis of closeness to the exact solution seems to be a good indicator of mesh quality, but it is also computationally very expensive since it requires a convergence test to estimate the exact solution.

The overall result is that Gmsh is the best 3-D mesh generator for use in our finite-element modelling and simulation pipeline.

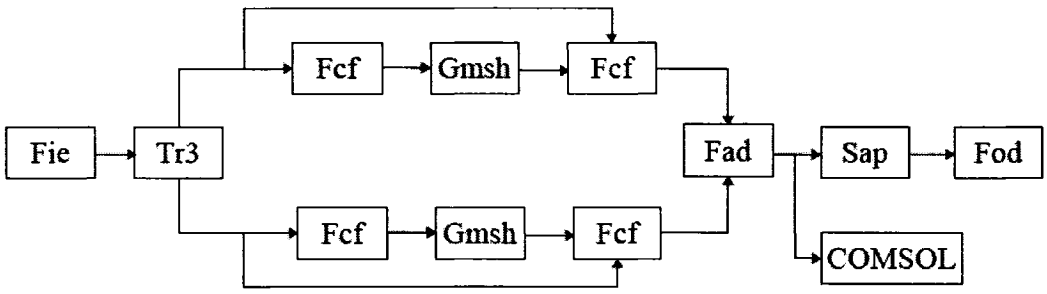


Figure 7.1 Finite-element modeling and simulation pipeline in Auditory Mechanics Laboratory

As shown in Figure 7.1, each simple closed surface mesh of a complex model generated by Tr3 is converted to the native file format of Gmsh using Fcf, and then they are separately imported into Gmsh for 3-D mesh generation. The mechanical parameters (boundary conditions, load conditions and material properties) that were defined on the surface meshes are assigned to the resulting volume meshes by using Fcf again. These models are combined together using Fad to form the complex model. There are two types of output from Fad. One output, with extension .sap, can be imported into Sap for finite-element simulation. For the other output, with extension .nastran, only geometric information about the complex model is preserved and can be imported into COMSOL. Thus, the mechanical parameters must be manually assigned to the resulting volume mesh before finite-element simulation in COMSOL.

7.2 Future work

The current pipeline can be improved in at least four respects. First, an improvement would be to modify Tr3 so that it outputs multiple simple parts at once. With the help of a scripting language, the multiple simple parts could be processed automatically. Second, an improvement would be to integrate the file-format conversion function (Tr3 to Gmsh) of Fcf into Tr3. Third, the pre-processing function of Fcf could be integrated into Tr3. Last, the post-processing function of Fcf could be integrated into Fad. These improvements would simplify the pipeline structure and increase the overall efficiency.

Only three shape measures and two size measures were used and compared in our

evaluation. Since the Jacobian-based mesh-quality measures, as discussed in Section 3.4, can contain not only the shape and size but also the orientation of an element, such measures might be better in evaluating the comprehensive quality of an element than either a shape measure or a size measure alone.

As discussed in Chapter 4, *a priori* finite-element mesh-quality measures, such as solution residuals and the closeness to the exact solution as approximated by a convergence curve, are able to provide only qualitative information about the overall mesh quality. In contrast, *a posteriori* finite-element mesh-quality measures are able to quantitatively characterize the error contribution of each element, which in turn reflects the element quality. In this case, the element quality contains not only the geometrical information but also the information that is closely related to the finite-element formulation. Hence, *a posteriori* measures are very promising measures for estimating the element quality as well as the overall mesh quality.

Material-property discontinuities may result in large errors in finite-element solutions (Muller and Korvink 2003). Most discussions of mesh-quality measures do not consider material properties as possible factors. Further experiments on a complex model with multiple material properties could help in understanding how material properties affect the finite-element solution accuracy.

REFERENCES

ADINA Inc. (2005): *ADINA*. http://www.adina.com/

Ainsworth M and Oden JT (2000): *A posteriori error estimation in finite element analysis*. John Wiley & Sons, New York.

Altair Engineering Inc. (2005): *HyperMesh*. http://www.altair.com/software/hw_hm.htm

ANSYS Inc. (2005): *ANSYS*. http://www.ansys.com/

Babuška I and Rheinboldt WC (1978a): A posteriori error estimates for the finite element method. *Int J Numer Methods Engng*, 12: 1597-1615.

Babuška I and Rheinboldt WC (1978b): Analysis of optimal finite element meshes in R^1. *Math Comput*, 33: 435-463.

Babuška I and Vogelius M (1984): Feedback and adaptive finite element solution of one-dimensional boundary value problems. *Numer Math*, 44: 75-102.

Babuška I and Strouboulis T (2001): *The Finite Element Method and its Reliability*. Oxford Science Publications.

Baker TJ (1989): Element quality in tetrahedral meshes. *Proceedings of the 7th International Conference on Finite Element Methods in Flow Problems*. Huntsville, AL, April, 3-7.

Bathe KJ, Wilson EL and Peterson FE (1974): *SAP IV: a structural analysis program for static and dynamic response of linear systems*. Report UCB/EERC-73/11, Earthquake Engineering Research Center, Berkeley.

Bathe KJ (1996): *Finite element procedures*. Prentice Hall, New Jersey.

Borouchaki H, Hecht F, Saltel E and George PL (1995): Reasonably efficient Delaunay based mesh generator in 3 dimensions. *Proceedings 4th International Meshing Roundtable*, October, 3-14.

Boubez TI, Funnell WRJ, Lowther DA and Pinchuk AR (1986a): Mesh generation for computational analysis: Part I – Electromagnetic and technical considerations for mesh generation. *Comput Aided Engng J*, October, 190-195

Boubez TI, Funnell WRJ, Lowther DA and Pinchuk AR (1986b): Mesh generation for computational analysis: Part II – Geometric and topological considerations for three-dimensional mesh generation. *Comput Aided Engng J*, October, 196-201

Bowyer A (1981): Computing Dirichlet tessellations. *The Comput J*, 24(2): 162-166.

Chae SW and Lee GM (1999): Volume triangulation from planar cross sections. *Comput & Struct*, 72: 93-108.

Cavendish JC, Field DA and Frey WH (1985): An approach to automatic three-dimensional finite element mesh generation. *Int J Numer Meth Engng*, 21: 329-347.

Chellamuthu KC and Ida N (1994): 'A posteriori' element by element local error estimation technique and 2D & 3D adaptive finite element mesh refinement. *IEEE Transactions on Magnetics*, 30(5): 3527-3520.

Cheng SW, Dey T, Edelsbrunner H, Facello M and Teng SH (1999): Sliver exudation. *In ACM Symposium on Computational Geometry*, 1-13.

Chew LP (1989): Constrained Delaunay triangulation. *Algorithmica*, 4: 97-108.

Christiansen HN and Sederberg TW (1978): Conversion of complex contour line definitions into polygonal element mosaics. *Comput Graph*, 12(2): 187-192.

CIMNE Inc. (2005): *GiD*. http://gid.cimne.upc.es/

COMSOL Inc. (2006): *COMSOL*. http://www.comsol.com/

Cougny HL, Shephard MS and George MK (1990): *Explicit node point smoothing within octree*. Report No. 10-1990, SCOREC, RPI, Troy, NY.

Dannelongue HH and Tanguy PA (1991): Three-dimensional adaptive finite element computations and applications for non-Newtonian flows. *Int J Num Meth Fluids*, 13: 145-165.

Delaunay BN (1934): *Sur la sphère vide*. Bulletin of the Academy of Science of the USSR VII. Class Sci Mat Nat, 6: 793-800.

Dompierre J, Labbe P, Guibault P and Camerero R (1998): Proposal of benchmarks for 3D unstructured tetrahedral mesh optimization. *Proceedings of the 7th International Meshing Roundtable,* Dearborn, MI. Sandia Report SAND 98-2250, Sandia National Laboratories, Albuquerque, NM, 459–478.

Ekoule AB, Peyin FC and Odet CL (1991): A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Trans Graph*, 10(2): 182-199.

Ewing RE (1990): A posteriori error estimation. *Comput Methods in Appl Mech Engng*, 82: 59-72.

Field DA (1995): The legacy of automatic mesh generation from solid modelling. *Computer Aided Geometric Design*, 12: 651-673.

Field DA (2000): Qualitative measures for initial meshes. *Int J Numer Meth Engng*, 47: 887-906.

Fortune SJ (1987): A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2: 153-174.

Freitag LA and Ollivier-Gooch CF (1996): A comparison of tetrahedral mesh improvement techniques. *Proceedings of the 5th International Meshing Roundatable*, Sandia National Laboratories, 87-106.

Freitag LA and Knupp PM (1999): Tetrahedral element shape optimization via the Determinant and condition number. *Proceedings of the 8th International Meshing Roundtable*, 247-258.

Freitag LA and Knupp PM (2002): Tetrahedral mesh improvement via the optimization of the element condition number. *Int J Numer Methods Engng,* 53(6): 1377-1391.

Frey PJ, Boroucahki H and George PL (1998): 3D Delaunay mesh generation coupled with an advancing-front approach. *Comput Methods Appl Mech Engng*, 157: 115-131.

Frey PJ and George PL (2000): *Mesh generation: application to finite elements*. Hermes Science Publishing, United Kingdom.

Fuchs H, Kedem ZM and Uselton SP (1977): Optimal surface reconstruction from planar contours. *ACM*, 10: 693-70.

Funnell WRJ (1984): On the choice of a cost function for the reconstruction of surfaces by triangulation between contours. *Comput & Struct*, 18(1): 23-26

Funnell SM and Funnell WRJ (1988): An approach to finite-element modelling of the middle ear. *Proceedings of the 14th Can Med & Biol Eng Conf,* 101-102.

Funnell WRJ (2006): *AudiLab software.* http://audilab.bmed.mcgill.ca/~funnell/AudiLab/sw/

George PL (1971): *Computer implementation of the finite element method.* PhD thesis, Dept of Computer Science, Stanford University.

George PL (1997): Improvement on Delaunay based 3D automatic mesh generator. *Finite Elements in Analysis and Design*, 25: 297-317.

Geuzain C and Remacle J-F (2005): *Gmsh.* http://www.geuz.org/gmsh/

Gratsch T and Bathe KJ (2005): A posteriori error estimation techniques in practical finite element analysis. *Comput & Struct*, 83: 235-265.

Haimes R, Connell SD and Vermeersch SA (1993): Visual grid quality assessment for 3D unstructured meshes. *AIAA paper*, 93-3352, Orlando, FL.

Hang S (2005a): *TetGen.* http://tetgen.berlios.de/

Hang S (2005b): *TetView.* http://tetgen.berlios.de/tetview.html

Henson OW and Henson M (2005): *MRM dataset of human middle ear structure.* http://cbaweb2.med.unc.edu/henson_mrm/pages/Scans_Primates.html

Hermeline F (1982): Triangulation automatique d'un polyhedre en dimension N. *RAIRO Analyse Numerique*, 16(3): 211-242.

Ho-Le K (1988): Finite element mesh generation techniques: a review and classification. *Computer-Aided Design*, 20(1): 27-38.

Homles DG and Snyder DD (1988): The generation of unstructured triangular meshing using Delaunay triangulation. *Numerical grid generation in computational fluid mechanics'88*, Miami, 643-652.

Janicke L and Kost A (1996): Error estimation and adaptive mesh generation in the 2D and 3D finite element method. *IEEE Transactions On Magnetics*, 32(3): 1334-1337.

Jin H and Tanner RI (1993): Generation of unstructured tetrahedral meshes by advancing-front method. *Int J Numer Methods Engng*, 36: 1805-1823.

Kelly DW, Gago OC, Zienkiewicz OC and Babuška I (1983): A posteriori error analysis and adaptive processes in the finite element method: Part I – error analysis. *Int J Numer Methods Engng*, 19: 1953-1619.

Keppel E (1975): Approximating complex surfaces by triangulation of contour lines. *IBM J Res Dev*, 19: 2-11.

Knupp PM (1999): *Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Part II – A framework for volume mesh optimization*. Technical Report SAND 99-0709J, Sandia National Laboratories.

Knupp PM (2000): Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Part I – A framework for surface mesh optimization. *Int J Numer Methods Engng*, 48(3):401-420.

Knupp PM (2001): Algebraic mesh quality metrics. *SIAM J Sci Comput*, 23(1): 193-218.

Lawson CL (1977): Software for C1 Surface Interpolation. Mathematical Software III, *J Rice ed.*, Academic Press, New York,161-194.

Lee DT and Schachter BJ (1980): Two algorithms for constructing a Delaunay triangulation. *Int J Comput Inf Sci*, 9(9): 219-242.

Lewis RW, Zheng Y and Gethin DT (1996): Three-dimensional unstructured mesh generation: Part 3. Volume meshes. *Comput Methods Appl Mech Engng*, 134: 285-310.

Liu A and Joe B (1994a): On the shape of tetrahedra from bisection. *Mathematics of Computation*, 63(207): 141-154.

Liu A and Joe B (1994b): Relationship between tetrahedron shape measures. *BIT*, 34: 268-287.

Lohner R (1996): Extensions and improvements of advancing-front grid generation method. *Commun Numer Methods Eng*, 12: 683-702.

Lo SH (1985): A new mesh generation scheme for arbitrary planar domains. *Int J Numer Methods Engng*, 21: 1403-1426.

Lo SH (2002): Finite element mesh generation and adaptive meshing. *Prog Struct Engng Mater*, 4: 381-399.

Marcum DL and Weatherill NP (1995): Unstructured grid generation using iterative point insertion and local reconnection. *AIAA Journal*, 33(9): 1619-1625.

The MathWorks (2006): *MATLAB*. http://www.mathworks.com/

Mavriplis DJ (1992): *An advancing front Delaunay triangulation algorithm designed for robustness*. IGASE report, 92-99.

Meyers D, Skinner S and Sloan K (1992): Surface from contours. *Compt Graph Image Process,* 11(3): 228-258.

MSU-ERC (2005): *SolidMesh*. http://www.erc.msstate.edu/simcenter/docs/solidmesh/

Muller J and Korvink JG (2003): A general purpose adaptivity driver for FE software. *Softw Pract Exper*, 33:1097-1116.

NetLib (2005): *Eispack*. http://www.netlib.org/eispack/

Nguyen VP (1982): Automatic mesh generation with tetrahedron elements. *Int J Numer Methods Engng*, 18: 273-389.

Noor AK and Babuška I (1987): Quality assessment and control of finite element solutions. *Finite Element in Analysis and Design*, 3: 1-26.

Numerical Engineering & Consulting Service Inc. (2005): *PASTEK*.
http://www.necs.fr/out_divers.php

Oden JT, Strouboulis T, Devloo P and Howe M (1986): Recent advances in error estimation and adaptive improvement of finite element calculations. *Computational Mechanics Advances and Trends*, 75:369-400.

Ollivier-Gooch CF (2005): *GRUMMP*. http://tetra.mech.ubc.ca/GRUMMP/

O'Rourke J (1998): *Computational geometry in C*. Cambridge University Press, Cambridge, UK.

Owen SJ (1998): A survey of unstructured mesh generation technology. *Proceedings of the 7th International Meshing Roundtable*, Dearborn, MI. October.

Owen SJ (2005): *Meshing research corner*.
http://www.andrew.cmu.edu/user/sowen/mesh.html

Parthasarathy VN (1993): A comparison of tetrahedron quality measures. *Finite Elements in Analysis and Design*, 15: 255-261.

Peraire J, Peiro J and Morgan K (1992): Adaptive remeshing for three-dimensional compressible flow computations. *J Comput Phys*, 103: 269-285.

Perrson P-O (2005): *DistMesh*. http://www-math.mit.edu/~persson/mesh/

Remotigue MG and The NGP Team (1994): The national grid project: Making dreams into reality, in: N.P. Weatherill, P.R. Eiseman, J. Hauser and J.F. Thompson (eds.). Numerical Grid Generation in Computational Fluid Dynamics and Related Fields. *Proceedings of the 4th International Conference,* 429-439, Swansea, UK.

Ruppert J (1992): A new simple algorithm for quality 2-dimensional mesh generation. Technical Report UCB/CSD 92/694, University of California, Berkeley, California.

Schmelzer I (2005): *COG*. http://www.wias-berlin.de/software/cog/

Schneiders R (2005): *Software*.
http://www-users.informatik.rwth-aachen.de/~roberts/software.html

Schöberl J (2005): *NETGEN*. http://www.hpfem.jku.at/netgen/

Shantz M (1981): Surface definition for branching contour-defined objects. *Computer Graphics*, 15(2): 242-270.

Shephard MS and Georges MK (1991): Three-dimensional mesh generation by finite octree technique. *Int J Numer Methods Engng*, 32: 709-749.

Shewchuk JR (2002): What is a good linear element? Interpolation, conditioning and quality measures. *Proceedings of the 11th International Meshing Roundtable*, Sandia National Laboratories. September 15-18, 115-126.

Siah TS (2002): *Finite-element modelling of the mechanics of the coupling between the incus and stapes in the middle ear*. M.Eng Thesis, Dept. of Biomedical Engineering, McGill University.

Simulog Technologies Inc. (2005): *TetMesh GSH3D*. http://www.simulog.fr/tetmesh/

Sloan KR and Painter J (1988): Pessimal guess may be optimal. A constitutive search result. *IEEE Trans Pattern Anal Mach Intell*, 10(6): 949-955.

Soroka BI (1981): Generalized cones from serial sections. *Compt Graph Image Process*, 15: 154-166.

Tarnhuvud T, Reichert K and Skoczylas J (1990): Problem-oriented adaptive mesh-generation for accurate finite-element calculation. *IEEE Transactions on Magnetics*, 26(2): 779-782.

Thacker WC, Gonzalez A and Putland GE (1980): A method for automating the construction of irregular computational grids for storm surge forecast models. *J Comput Phys*, 37(3): 37-387.

Van Oosterom A and Strackee J (1983): The solid angle of a plane triangle. *IEEE Trans Biomed Eng*, 30: 125-126.

Verfurth R (1994): A posteriori error estimation and adaptive mesh-refinement techniques. *J Comput Appl Math*, 50: 67-83.

Wang YF and Aggarwal JK (1986): Surface reconstruction and representation of 3-D scenes. *Pattern Recognition*, 19(3): 197-207.

Watson DF (1981): Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *Comput J*, 24(2): 167-172.

Weatherill NP and Hasson O (1994): Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. *Int J Numer Methods Engng*, 37: 2005-2039.

Weatherill NP (1996): The reconstruction of boundary contours and surfaces in arbitrary unstructured triangular and tetrahedral grids. *Engineering Computations*, 13(8): 66-81.

Yerry MA and Shephard MS (1983): A modified quadtree approach to finite element mesh generation. *IEEE Comput Graph Appl,* 1:39-46.

Yerry MA and Shephard MS (1984): Automatic three-dimensional mesh generation by the modified-octree technique. *Int J Numer Methods Engng*, 20:1965-1990.

Zienkiewicz OC and Zhu JZ (1987): A simple error estimator and adaptive procedure for practical engineering analysis. *Int J Numer Methods Engng*, 24: 337-357.

Zienkiewicz OC and Taylor RL (2000): *The Finite Element Method. Volume 1. Basic Formulation and Linear Problems*. McGraw-Hill.