Finite Element Triangulation of Complex Regions Using Computational Geometry

Barry Joe

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

CS-84-19

July 1984

FINITE ELEMENT TRIANGULATION OF COMPLEX REGIONS USING COMPUTATIONAL GEOMETRY

by

Barry Joe

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, 1984

ABSTRACT

Mesh generation is an important, resource consuming part of finite element analysis and should be carried out automatically as much as possible. For this purpose, we present a method for producing a triangulation of a complex polygonal region of the plane. The method allows the desired number of triangles and a mesh smoothness parameter to be specified along with the polygonal curves of the region's boundary. Computational geometry techniques are used to decompose and triangulate the region.

The method proceeds in three stages. In the first stage, the region is decomposed into convex polygons such that small interior angles are avoided. In the second stage, a mesh distribution function is defined based on the boundary length scales and the region is further decomposed into convex polygons such that the variation of the function in each subregion is limited, since we aim to produce a triangulation which is approximately equidistributing with respect to the function. In the third stage, a Delaunay triangulation is constructed in each subregion using a quasi-uniform grid whose spacing is determined from the mesh distribution function. The computational complexity and correctness of algorithms in the three stages are discussed.

We have implemented an experimental prototype of the method in PASCAL and carried out tests of triangulations for a variety of regions. We report on the performance of the method for major computational experiments on two complex test regions.

ACKNOWLEDGEMENTS

The author is especially grateful to his supervisor, Professor R. Bruce Simpson, for the suggestion of the thesis topic and many helpful discussions. His guidance and encouragement during the research for this thesis were invaluable.

The author wishes to thank Professor John C. Beatty for the use of the Computer Graphics Laboratory graphics package and plotter to produce plots of triangulations.

The author wishes to acknowledge the Canada Centre for Inland Waters, Environment Canada for providing the outline of Lake Superior, and Professor Randolph E. Bank for providing the CMOS region and the PLTMG package.

Finally, the author wishes to acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada through a Postgraduate Scholarship.

TABLE OF CONTENTS

ABSTRACT ii

ACKNOWLEDGEMENTS iii

LIST OF TABLES vi

LIST OF FIGURES vii

CHAPTER 1 INTRODUCTION 1

1.1 Overview 1

1.2 Survey of previous works on finite element triangulation 2

1.3 Outline of thesis 7

CHAPTER 2 PRELIMINARIES 10

2.1 Objectives 10

2.2 Outline of the method 13

2.3 Computational geometry problems 15

2.4 Data structure for polygons in the decomposition 18

CHAPTER 3 INITIAL DECOMPOSITION INTO CONVEX POLYGONS 22

3.1 Simply connected regions 22

3.2 Selection of separators 23

3.3 Multiply connected regions 31

3.4 Summary and time complexity 34

CHAPTER 4 FURTHER DECOMPOSITION TO BOUNDARY SCALES 37

4.1 Definition of mesh distribution function 37

4.2 Subdivision of convex polygon 40

4.3 Computation of triangle sizes 46

4.4 Summary and time complexity 48

CHAPTER 5 TRIANGULATION OF CONVEX POLYGONS 50

5.1 Valid and Delaunay triangulations 51

5.2 Shrinking a convex polygon 53

5.3 Triangulation of the interior of P 59

5.4 Triangulation near the boundary of P 65

5.5 Validity of triangulation VT(P) 75

5.6 Converting VT(P) into a Delaunay triangulation 82
5.7 Summary and time complexity 87
CHAPTER 6 PERFORMANCE OF THE METHOD 90
6.1 Internal parameters and performance measurements 93
6.2 Effects of varying the parameters 98
6.3 Performance on two test regions 103
CHAPTER 7 CONCLUDING REMARKS 129
APPENDIX A EXAMPLES OF TRIANGULATIONS 131
APPENDIX B DEFINITIONS 137
B.1 Geometric terms 137
B.2 Parameters of method 138
REFERENCES 139

LIST OF TABLES

6.1 Results from varying σ for Lake Superior 99

6.2 Results from varying τ and $\lambda = 2\tau$ for Lake Superior 100

6.3 Results from varying $d_{\min}$ for Lake Superior 100

6.4 Results from varying $n_{\min}$ for Lake Superior 101

6.5 Results from varying κ for Lake Superior 102

6.6 Results from varying $d_{\min}$ for CMOS region 103

6.7 Results from the second stage for CMOS region 117

6.8 Results from the second stage for Lake Superior 117

6.9 Results from the third stage for CMOS region 119

6.10 Results from the third stage for Lake Superior 119

6.11 Distribution of triangle angles for CMOS region 120

6.12 Distribution of triangle angles for Lake Superior 120

6.13 Distribution of q_i for CMOS region 121

6.14 Distribution of q_i for Lake Superior 122

6.15 Distribution of s_i for CMOS region 123

6.16 Distribution of s_i for Lake Superior 123

6.17 CPU times in seconds for the three stages for CMOS region 124

6.18 CPU times in seconds for the three stages for Lake Superior 124

LIST OF FIGURES

2.1 Illustration of three types of interfaces 11

2.2 Illustration of the three lists of data structure for polygons 20

3.1 Definition of cones RC, IC, LC 24

3.2a Definition of angles α_0, β_0, α_1, β_1 for one separator 25

3.2b Definition of angles α_0, β_0, γ_0, α_1, β_1, α_2, β_2 for two separators 25

3.3 Restriction of separator endpoints to V_1 causes small angles 27

3.4 Restriction of separator endpoints to V avoids narrowings 27

3.5 Illustration of VP for Lemma 3.1 30

3.6 Illustration of 'spiral' polygon with $a = 20°$, $b = 40°$, $c = 120°$ 30

3.7a Reduction of R to simply connected region R' after Step 1 33

3.7b Decomposition of R into simple polygons R_1, R_2, R_3 after Step 2 33

4.1a Illustration of sector S and possible endpoints for a separator 43

4.1b Example where sector S does not identify a separator 43

4.2 Selection of separator based on shape of polygon 44

4.3 Illustration of $D(P_i)$ 45

4.4a v is on $\partial D(P_i)$ 45

4.4b v is in the interior of $D(P_i)$ 45

5.1 Illustration of convex polygon Q 55

5.2 ∂Q_{i-1} and L_i intersect at point t 55

5.3 ∂Q_{s-1} and L_s intersect at points t_1 and t_2 57

5.4 Three cases of position of L_i with respect to Q_{i-1} 57

5.5 Generation of mesh vertices in $int(P)$ 60

5.6 Triangulation of mesh vertices in $int(P)$ 60

5.7a Edge v_1v_2 is in E_1 63

5.7b Edge v_1v_2 is in E_2 63

5.8 Illustration of closed walk C, cycle B, and strip A 66

5.9 Four cases of quadrilateral Q 67

5.10 Illustration of $T(P)$ and $DT(P)$ 69

5.11 Illustration of invalid triangulation $T(P)$ 70

5.12 Illustration of chains B_L, B_R and $VT(P)$ in the case of no interior vertices 71
5.13a Example where degenerate triangle is formed by procedure MERGE 76
5.13b Perturbation of chain $u_1u_2u_3$ to obtain valid triangulation 76
5.14 $\Delta v_{j+1}v_ju_{r(j)}$ intersects walk $u_0v_0v_1\cdots v_j$ 79
5.15 $\Delta v_jv_{j-1}u_{r(j-1)}$ intersects edge v_jv_{j+1} 79
6.1 CMOS region with six subregions determined by internal interfaces 91
6.2 Outline of Lake Superior with six islands 92
6.3 First stage decomposition of CMOS region 105
6.4 Second stage decomposition of CMOS region for $N_t = 1000$, $\kappa = 0.25$ 106
6.5 Triangulation of CMOS region for $N_t = 1000$, $\kappa = 0.0$ 107
6.6 Triangulation of CMOS region for $N_t = 1000$, $\kappa = 0.25$ 108
6.7 Triangulation of CMOS region for $N_t = 1000$, $\kappa = 0.5$ 109
6.8 Right half of triangulation of CMOS region for $N_t = 3000$, $\kappa = 0.25$ 110
6.9 First stage decomposition of Lake Superior 111
6.10 Second stage decomposition of Lake Superior for $N_t = 2000$, $\kappa = 0.25$ 112
6.11 Triangulation of Lake Superior for $N_t = 2000$, $\kappa = 0.0$ 113
6.12 Triangulation of Lake Superior for $N_t = 2000$, $\kappa = 0.25$ 114
6.13 Triangulation of Lake Superior for $N_t = 2000$, $\kappa = 0.5$ 115
6.14 Western end of triangulation of Lake Superior for $N_t = 5000$, $\kappa = 0.25$ 116
6.15 Graph of CPU times versus N_t for CMOS region 126
6.16 Graph of CPU times versus N_t for Lake Superior 127
6.17 Triangulation of CMOS region with 1213 triangles by subroutine TRIGEN 128
A.1 Triangulation of L-shaped region for $N_t = 300$, $\kappa = 0.25$ 132
A.2 Triangulation of pistol for $N_t = 200$, $\kappa = 0.25$ 133
A.3 Triangulation of machine element for $N_t = 600$, $\kappa = 0.25$ 134
A.4 Triangulation of Lake Okeechobee for $N_t = 800$, $\kappa = 0.25$ 135
A.5 Triangulation of bracket for $N_t = 800$, $\kappa = 0.25$ 136

CHAPTER 1

INTRODUCTION

1.1. Overview

The finite element method is used to approximate the solution of a partial differential equation with boundary conditions defined on a domain. In the mesh generation step in two dimensions, the domain or planar region is subdivided into finite elements, e.g. triangles or quadrilaterals. This step is an important, resource consuming part of finite element analysis and should be carried out automatically by the computer as much as possible. A variety of mesh generation techniques for partial differential equations have been proposed in the literature (cf. surveys by Simpson (1979), Thacker (1980), Thompson (1983)). Meshes in this context have a fairly broad interpretation including triangulations, coverings by convex quadrilaterals, and curvilinear coordinate systems. Common general techniques include local mesh refinement (Babuska and Rheinboldt (1978), Bank and Sherman (1980)) and region mapping (Thompson (1982)).

For complex regions of the plane, triangles are better than quadrilaterals at approximating a complicated boundary, so we concentrate on triangular meshes in this thesis. Triangulation to produce a finite element mesh differs from the standard triangulation problems of computational geometry in that the vertices of the triangulation are largely free to be selected by the method, but the distribution of triangle sizes should be controlled. Other applications of triangulation of planar regions include surface interpolation (Lawson (1977)) and contour plotting (Lewis and Robinson (1978)).

Approaches for triangulation of complex regions include the techniques of boundary contraction (Bykat (1976)), modified quadtree representation (Yerry and Shephard (1983)), and triangulation of sets of vertices (Cavendish (1974)), this latter topic relating to the large literature on Voronoi diagrams and Delaunay triangulations. A survey of several triangulation methods is provided in Section 1.2. Generally, these techniques have been described at the method level, with little or no demonstration of algorithm correctness or complexity analysis, and no quantitative reporting of

experiments, so that comparisons are largely speculative. Many of these methods require a lot of user input for complex regions, e.g. a partitioning of the region into simpler subregions. Few of these methods are available as generally distributed software; some are embedded in finite element packages, e.g. subroutine TRIGEN of the PLTMG package (Bank (1982)). In these methods there seem to be difficulties in understanding how to handle general regions.

To overcome these difficulties, we have explored the application of computational geometry ideas to finite element triangulation in this thesis, since computational geometry is concerned with efficient and correct algorithms for geometric problems. We have developed and implemented a method for triangulating general polygonal regions of the plane which is based on recently developed techniques for polygon decomposition and triangulation. It is our speculation that basing the method on computational geometry techniques makes it robust and competitive. In Section 1.3 we provide an outline of the thesis.

1.2. Survey of previous works on finite element triangulation

In this section we provide a brief survey of methods for triangulating a (complex) region, R. The goals of an automatic triangulation method are:

(a) minimal user input should be required,

(b) it should be capable of generating triangular meshes for general regions including multiply connected regions and regions with complicated boundary geometry,

(c) it should control the distribution of triangle sizes within the region so that the triangle sizes adequately represent the region and the solution of the partial differential equation.

The following eight methods are representative of the various approaches for finite element triangulation (see the references of Simpson (1979) and Thacker (1980) for other methods). For each method we briefly describe (i) the input required by the user, (ii) how the distribution of triangle sizes is determined, and (iii) how the region R is triangulated.

A. Cavendish (1974) - modification of Fukuda and Suhara (1972)

(i) The input includes mesh vertices on polygonal boundaries of R in counterclockwise order, fixed interior mesh vertices, vertices of simply connected polygonal zones Z_i which are disjoint and cover R, density factor $r_i > 0$ for each zone Z_i, and shrinking distance parameter $\epsilon > 0$.

(ii) The distribution of triangle sizes is determined by the different density factors r_i in the zones and the mesh vertices on the boundaries of R.

(iii) The boundary of R is shrunk by distance ϵ to get subregion $\tilde{R}$. For each zone Z_i, the subregion $R_i = \tilde{R} \cap Z_i$ is determined. A square grid of spacing r_i is superimposed over R_i. An attempt is made to randomly generate one mesh vertex in each subsquare in R_i such that there are no mesh vertices in the circle of radius r_i centred at this vertex. A heuristic procedure is used to triangulate all the vertices so that triangles do not have a large deviation from being equilateral. Finally this triangulation is improved by a smoothing technique: several passes are made in which each interior vertex is moved to the centroid of the polygon formed from the triangles incident on that vertex.

B. Shaw and Pitchen (1978) - modification of Fukuda and Suhara (1972)

(i) The input includes vertices of polygonal boundaries of simply connected subregions R_i of R and density factor $r_i > 0$ for each subregion R_i.

(ii) The distribution of triangle sizes is determined by the different density factors r_i in the subregions.

(iii) Mesh vertices are generated on the boundary of R_i at spacing of approximately r_i. If subregions R_i and R_j share a boundary segment, then density factor $\sqrt{r_i r_j}$ is used to generate equally spaced mesh vertices on the segment. Each subregion R_i is shrunk by distance $r_i/2$ to get $\tilde{R}_i$. A rectangular grid of spacing r_i by $\sqrt{3}r_i$ is superimposed over $\tilde{R}_i$. At most two mesh vertices are generated at fixed relative positions in each subrectangle in $\tilde{R}_i$ so that equilateral triangles can be formed from these vertices. Triangles in the strip between ∂R_i (the boundary of R_i) and $\partial \tilde{R}_i$ are then formed using a heuristic procedure. Finally the smoothing technique of Cavendish is applied to the triangulation in each subregion R_i.

C. Bykat (1976)

(i) The input consists of mesh vertices on the polygonal boundary of R in counterclockwise order. If R is multiply connected, then 'cut' lines are introduced to make it simply connected.

(ii) The distribution of triangle sizes is determined by the boundary geometry and the mesh vertices on the boundary. Additional mesh vertices are inserted on the boundary so that two consecutive segments on the boundary do not differ in length by more than a factor of two.

(iii) Region R is subdivided into convex subregions using line segments which originate at reflex vertices and do not create small angles at the boundary. Each convex subregion is triangulated using boundary contraction, i.e. triangles are successively removed from the boundary of the untriangulated part of the subregion, such that triangles with small angles are avoided. Automatic grading is performed near 're-entrant corners'. In Bykat (1982), each convex subregion is triangulated by recursive bisection of convex polygons such that triangles with small angles are avoided.

D. Sadek (1980)

(i) The input includes vertices of polygonal subregions (which may be multiply connected) of R and either the mesh vertices on each boundary edge or the number of triangles and gradient along each boundary edge. In the latter case, mesh vertices are generated on the boundary edges using the given information.

(ii) The distribution of triangle sizes is determined by the boundary geometry and the mesh vertices on the boundary of the subregions.

(iii) Each subregion is triangulated using boundary contraction such that triangles with small angles are avoided. The boundary contraction technique is different from Bykat (1976) and can be used to triangulate nonconvex and multiply connected subregions.

E. Reid (1974)

(i) The input includes a subroutine which gives each smooth section of boundary or internal interface in parametric form and a local step-size function for the region R.

(ii) The distribution of triangle sizes is determined by the local step-size function and the boundary geometry.

(iii) Starting with an equilateral triangle which circumscribes R, a mesh consisting of equilateral triangles and bisected equilateral triangles is created by local refinement based on the step-size function, i.e. a triangle may be subdivided into four equilateral triangles or two bisected equilateral triangles depending on the values of the step-size function in the triangle and any adjacent triangles and this procedure is repeated for any new triangles. Then the mesh is distorted to fit the boundary of R. This may cause more local refinement of triangles due to complicated boundary sections.

F. Thacker, Gonzalez, and Putland (1980)

(i) The input includes vertices of polygonal boundaries of R, a discrete mesh density function, and a parameter for the spacing of the initial triangular grid.

(ii) The distribution of triangle sizes is determined by the initial spacing parameter and the mesh density function.

(iii) An equilateral triangular grid of spacing given by the input parameter is superimposed on R. A zigzag grid is obtained by eliminating points of the grid which are outside R. The zigzag grid is transformed into a curvilinear grid with a better representation of the boundary. Rows of points are removed from the curvilinear grid so that the density of points conforms better to the mesh density function. The remaining points of the grid are smoothed as in Cavendish's method so that the triangles are as equilateral as possible. This process of removing rows of points and smoothing the grid may be repeated several times.

G. Subroutine TRIGEN of PLTMG package (Bank (1982))

(i) The input includes vertices of the boundaries of simply connected subregions of R and a parameter h for the maximum triangle side length. The boundary of a subregion may consist of straight edges and curved edges. A curved edge has two endpoints and a midpoint and is approximated by a circular arc passing through the three points.

(ii) The distribution of triangle sizes is determined by the boundary geometry and the parameter h.

(iii) Each subregion (polygon, possibly with curved edges) is triangulated using three heuristics such that triangles with small angles are avoided. The first strategy tries to 'chop' off triangles from vertices of the boundary with small interior angles to reduce the order of the polygon. If the remaining polygon is convex with eight or fewer sides, the second strategy tries to triangulate the entire remaining subregion by adding the centroid as a mesh vertex and connecting it to each boundary vertex. The third strategy tries to break the polygon into two smaller polygons by connecting two nonadjacent vertices by a line segment which does not create small angles at the boundary. If all three strategies fail, then TRIGEN sets an error flag and returns. If h is small enough, TRIGEN should produce a triangulation. If two or more subregions are congruent, then only one subregion is triangulated using the above process and the other subregions are triangulated using an affine mapping.

H. Yerry and Shephard (1983)

(i) An interactive graphics front-end program is used to input the boundary curves and interfaces of region R (which do not have to be polygonal). Tolerances (or tree levels) for different portions of the boundary may also be input. The tolerances allow the creation of a user-defined graded mesh.

(ii) The distribution of triangle sizes is determined by the boundary geometry and the tolerances.

(iii) Starting with a square which contains R in an integer coordinate system, the region is subdivided into complete quadrants (squares) or cut quadrants (squares with corners cut off) by recursive subdivisions of quadrants into four subquadrants and is represented in a modified quadtree data structure. The creation of the modified quadtree begins with the discretization of the boundary by complete and cut quadrants. Then the rest of the modified quadtree is generated such that at most a one-level difference exists between any two neighbouring quadrants. Each quadrant in the modified quadtree is triangulated using a transitional mesh of one to six triangles

which depends on the quadrant type and its neighbouring quadrants. Then the boundary vertices of this mesh are pulled to the boundaries of the region and the triangulation is smoothed as in Cavendish's method.

Most of these methods are semi-automatic since a lot of user input is required for complex regions. In all these methods, there is no time complexity analysis given and not enough algorithmic details given for us to determine how parts of the method work. The capabilities and effectiveness of these methods are demonstrated by pictures of triangulations of various regions. There is no quantitative reporting of experiments to measure how 'good' the triangulations are or the time required for the triangulations.

1.3. Outline of thesis

The purpose of this thesis is to study applications of computational geometry to triangular mesh generation for the finite element method, and to develop and implement an efficient and robust method for triangulating general polygonal regions of the plane which may contain holes and/or internal interfaces. The research presented in this thesis is made up of the following components:

(1) design of the method to take advantage of basic computational geometry algorithms and to introduce the concept of boundary length scales and the related mesh distribution function,

(2) development and modification of algorithms for finite element triangulation,

(3) establishment of correctness and discussion of computational complexity of these algorithms,

(4) implementation of a prototype version of the method and experimentation on test regions.

The basic strategy of our method is to decompose the region into convex subregions in which triangles of uniform size can be used. In Chapter 2 we discuss the objectives and the outline of our method, the computational geometry problems encountered in developing the method, and the

data structure for representing the subregions in the decomposition of the region. In Chapter 3 we describe our algorithm for decomposing a region into convex polygons such that small interior angles are avoided, since we want to avoid triangles with small angles in the triangulation of the region. Our approach is based on the method of Schachter (1978) with some features from Bykat (1976).

Let T be a triangulation of a region R and $\psi(x,y)$ be a positive weight function defined in R. We define T to be *equidistributing* with respect to ψ in R if $\iint_{\Delta} \psi \, dA$ is constant for all triangles, Δ, in T. In our method we want the distribution of triangle sizes to conform to the distribution of length scales implied by the boundary. To accomplish this, we define a heuristic mesh distribution function, $\psi(x,y)$, based on the boundary length scales (see Chapter 4) and aim to produce a triangulation, T, of R which is approximately equidistributing with respect to ψ, i.e. $\iint_{\Delta} \psi \, dA$ is approximately constant for all triangles, Δ, in T.

There are no algorithms in the literature for constructing (approximately) equidistributing triangulations. The term 'equidistributing' is used to describe one-dimensional meshes for boundary value problems in which an interval is partitioned into subintervals such that the integral of a positive weight function is constant in each subinterval (Kautsky and Nichols (1980)). In two dimensions, $\psi(x,y)$ is usually related to the error estimate in the finite element solution and an equidistributing mesh is often called an optimal mesh. Shephard, Gallagher, and Abel (1980) construct near-optimal finite element meshes interactively. Ladeveze and Leguillon (1983) also construct near-optimal finite element meshes, but no algorithm is given in their paper. Our approach for constructing an approximately equidistributing triangulation is described in Chapter 4. Briefly, the region is decomposed into convex polygons of limited variation of ψ and a uniform triangle size is assigned to each subregion.

We construct a Delaunay triangulation in each subregion using a quasi-uniform grid. A Delaunay triangulation satisfies a max-min angle criterion, i.e. it is the optimal triangulation of a set of vertices for avoiding triangles with small angles. In Chapter 5, we use information about

the location of vertices to develop an algorithm for constructing a Delaunay triangulation in a convex polygon in expected $O(n_t)$ time where n_t is the number of triangles. In the standard triangulation problem of computational geometry in which the vertices are given as input and a (Delaunay) triangulation is constructed to cover the convex hull of the vertices, the time required to construct the triangulation is $O(n_t \log n_t)$ (Shamos and Hoey (1975), Lee and Schachter (1979)).

We have implemented an experimental prototype of the method in PASCAL and carried out tests of triangulations for a variety of regions on a VAX 11/780 running the UNIX operating system. Pictures of some triangulations are shown in Appendix A. In Chapter 6 we report in some detail on the performance of the method and code for major experiments on two complex test regions obtained from 'real' data; one arises in semiconductor device simulation and the other is a geographical region.

CHAPTER 2

PRELIMINARIES

2.1. Objectives

The regions of the plane to be triangulated by our method are simply connected or multiply connected polygonal regions, i.e. they are defined by piecewise linear simple closed boundary curves. The regions may contain piecewise linear internal interfaces. Internal interfaces are used to force triangle edges along particular polygonal curves inside the region (e.g. there are boundary conditions defined on the interfaces in the partial differential equation problem). Interfaces can also be used to force particular subregions in the decomposition of the region. There are three types of interfaces (see Figure 2.1):

(a) A 'separator' interface is an open nonintersecting polygonal curve which connects two different points on the same boundary curve and partitions a region into two subregions.

(b) A 'cut' interface is an open nonintersecting polygonal curve which connects a point on each of two different boundary curves (one is the boundary of a hole and the other is the outer boundary of the region containing the hole) and reduces the number of holes in the region by one.

(c) A 'hole' interface is a simple closed polygonal curve inside a region. It partitions the region into two subregions; one is the simple subregion inside the closed curve and the other is the original region with one more hole in it.

The line segments on the boundary curves and internal interfaces of a region will be referred to as boundary edges. What qualifies a region to be considered complex seems to be largely intuitive; however several properties of the boundary can probably be identified as contributing to 'complexity'. One is a significant variation in the lengths of the boundary edges. We assume that the lengths of these boundary edges represent relevant length scales which are to be reflected in the sizes of the triangles in their vicinity; i.e. we assume that extraneously small boundary edges

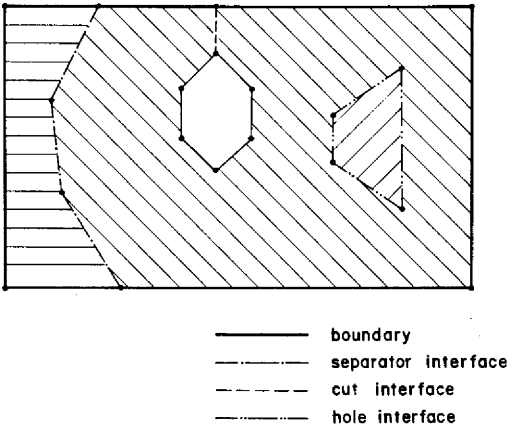


Figure 2.1 Illustration of three types of interfaces; shadings indicate different subregions

have been removed by a 'preprocessing' simplification step (e.g. Williams (1981)).

It is clear that convex regions are simpler than nonconvex regions. We call a boundary vertex at which the region occupies an angle of more than 180° a reflex vertex. A non-reflex vertex will also be referred to as a convex vertex. The proportion of reflex vertices to all vertices could be considered a measure of the complexity of the region. A more specific type of complication associated with nonconvexity occurs when a concave angle on the boundary cuts deeply into the region, bringing its reflex vertex, v, close to another boundary edge. This introduces a small length scale for triangulations that is not reflected in the boundary edge sizes, and we will say that the region is narrow at v. Most of the aspects of complexity that multiply-connectedness implies for a region from our perspective are covered by the measures discussed above, i.e. an increase in the proportion of reflex vertices, and the occurrence of narrowing in parts of the

region. In our method we use a common technique of adding 'cuts' to the boundary curves of multiply connected regions to reduce them to simply connected regions. In summary we view regions as increasingly complex as the proportion of reflex vertices rises, and the variation in length scales pertinent to triangulation increases.

Our triangulation procedure has been designed to meet the following objectives:

(a) specification in advance of the number of triangles in the mesh,

(b) avoidance of triangles with small angles,

(c) conformance of the distribution of triangle sizes to the distribution of length scales implied by the boundary, (2.1)

(d) control over the rate of change of the triangle sizes.

We will elaborate on these criteria. Objective (a) is motivated by the fact that the cost and resource requirements of finite element computations are directly related to the number of triangles in the mesh. Objective (b) is a conventional goal of finite element triangulation which appears to have its roots in the error analysis of piecewise polynomial approximation on triangulations, e.g. Strang and Fix (1973, p.138). This role is unclear in view of the results of Babuska and Aziz (1976), showing that the related avoidance of large angles is both necessary and sufficient for error control. For our procedure the main role of (b) is to ensure that there is only one length scale represented by a triangle.

As the above discussion of region complexity indicates, we view the boundary as defining a series of length scales for the triangulation, in the vicinity of the boundary. The distribution of triangle sizes near the boundary should follow this distribution of length scales, in a way made more explicit in Chapter 4. It is natural to extend this idea to allow as input to the procedure a function describing a distribution of length scales for the interior of the region as well, i.e. a user specified or adaptively determined mesh distribution function. We will discuss this extension in Chapter 7.

The rate of change of the triangle sizes as one moves through the triangulated region must

be limited if small angles in the triangles are to be avoided, i.e. objectives (b) and (d) are not independent. However, within this limitation, there is still considerable variation possible. Given two triangulations with different rates of change of triangle sizes, we will regard the lower rate of change as being the smoother triangulation. Our procedure allows as input, a parameter which controls the smoothness of the resulting triangulation.

Of course, the four objectives may conflict. For example, one cannot expect gradual changes in triangle sizes, with no small angle triangles, in a triangulation with only a few triangles for a region with a large variation in boundary edge size. So objectives (b), (d) may conflict with objective (a). In Chapter 6, we provide an empirical report on the degrees to which these objectives are served by the method for two test regions.

2.2. Outline of the method

In this section we provide an outline of our triangulation method. The input to our method is minimal and consists of the polygonal boundary curves of the region, R, the desired number of triangles, N_t, and a mesh smoothness parameter, κ. The input of piecewise linear internal interfaces in R is optional. The basic strategy of our method is to decompose R into convex polygons such that each subregion can be triangulated using triangles of uniform size. The method proceeds in three stages. The first two stages decompose the region into convex subregions such that the variation of triangle size in each subregion is limited, and a single length scale can be assigned to each subregion. The first stage is a purely geometric decomposition taking account of objective (2.1b) and the second stage further subdivides the convex subregions of the first stage to try to isolate subregions of different length scale. In the third stage, a Delaunay triangulation is constructed for each subregion using a quasi-uniform grid whose spacing is the length scale for that subregion, and the resulting triangulations are merged across subregion boundaries. Details of the decomposition stages are provided in Chapters 3 and 4 and of the triangulation stage in Chapter 5.

In the first stage, the input consists of the vertices (vertex coordinates) of the polygonal curves of the boundaries and interfaces of region R in counterclockwise order. If R contains any

holes or hole interfaces then R is first decomposed into simply connected subregions by introducing a 'cut' line segment from the boundary of each hole or hole interface to the outer boundary of the subregion containing the hole or hole interface. The simply connected subregions of R are decomposed into convex polygonal subregions using an approach based on Schachter (1978) with the additional constraints that small interior angles in the convex polygons and artificial narrowings in the subregions are avoided so that objectives (2.1b) and (2.1c) are satisfied respectively (cf. Bykat (1976)). To satisfy these constraints, points in addition to the boundary vertices are allowed as endpoints of the edges of the decomposition. The outputs of the first stage are the convex polygons $P_1, P_2, \ldots, P_{N_p}$ in the decomposition of R as well as the adjacency relations between the polygons (see Section 2.4 for the data structure used to represent the polygons and their adjacency relations).

In the second stage, the input consists of the output of the first stage and the parameters, N_t and κ. The length scales defined by the boundary and interface edges and the narrowings of R are present in the lengths of the edges of the P_i. We define a mesh distribution function, $\psi(x,y)$, using the lengths of the edges of the P_i which, in conjunction with the desired number of triangles N_t and the mesh smoothness parameter κ, indicates the triangle size to be used in a neighbourhood of (x,y). We aim to produce a triangulation which is approximately equidistributing (see Section 1.3) of ψ in R and contains approximately N_t triangles, i.e.

$$\iint_{\Delta} \psi \, dA \Big/ \iint_{R} \psi \, dA \approx 1/N_t$$

for all triangles, Δ, in the triangulation, so that objectives (2.1a) and (2.1c) are satisfied. To accomplish this, the P_i of the first stage are further subdivided into convex polygons such that the variation in ψ in each polygon is limited. Then a uniform triangle size is assigned to each convex polygon based on N_t and the mean of $\psi(x,y)$ in the polygon, with the constraint that the triangle sizes in adjacent polygons do not differ by a factor greater than two, so that a gradual change in triangle sizes occurs between adjacent polygons and objective (2.1d) is satisfied. The outputs of the second stage are the convex polygons $P_1, P_2, \ldots, P_{N_S}$ in the further decomposition of R

($N_s \geq N_p$ and the P_i have been relabelled from the first stage) and the uniform triangle sizes, h_i, in the P_i.

In the third stage, the input consists of the output of the second stage. For each polygon P_i, mesh vertices are generated on a quasi-uniform grid of spacing h_i in the interior of P_i. For each edge e of P_i, mesh vertices are generated on it at an equal spacing of approximately $\bar{h}_i$ where $\bar{h}_i = h_i$ if e is an edge of ∂R (the boundary of R) and $\bar{h}_i = \sqrt{h_i h_j}$ if e is a common edge of P_i and P_j so that a gradual change in triangle sizes occurs near e. The mesh vertices in the interior of P_i are triangulated to produce triangles of area $h_i^2/2$. The mesh vertices on ∂P_i and the border of the interior triangulation are merged to produce triangles in the strip next to ∂P_i. Then the triangulation in this strip is modified to obtain a Delaunay triangulation in P_i so that objective (2.1b) is satisfied since a Delaunay triangulation satisfies the max-min angle criterion (see Section 5.1). The output of the third stage is a triangulation of region R.

We use a common data structure for representing the triangulation, a vertex coordinate list and a triangle (element) incidence list (Simpson (1979)). The vertex coordinate list consists of two one-dimensional real arrays, X and Y, for storing the coordinates of the vertices of the triangulation, i.e. $(X(i),Y(i))$ are the coordinates of the vertex of index i. The triangle incidence list consists of a two-dimensional integer array with three columns, $VERT$, for storing the vertices of the triangles. The triangle of index i is represented by $(VERT(i,1),VERT(i,2),VERT(i,3))$ where $VERT(i,j)$, $1 \leq j \leq 3$, is the index of a vertex in the vertex coordinate list, i.e. the coordinates of the jth vertex of the ith triangle are $(X(VERT(i,j)),Y(VERT(i,j)))$. The vertices of a triangle are ordered so that they are in the counterclockwise direction.

2.3. Computational geometry problems

In this section we discuss the computational geometry problems encountered in developing our method. In Appendix B we define the geometric terms used below. In the first stage, a polygonal region which may contain holes is decomposed into convex polygons. A common computational geometry problem is the decomposition of a simple polygon into convex polygons. There

are various methods for solving this problem with different requirements. Some methods restrict the endpoints of the edges of the decomposition to be vertices of the simple polygon, i.e. a restricted decomposition is produced; other methods allow additional points to be introduced as endpoints of decomposition edges, i.e. an unrestricted decomposition is produced. The decomposition produced may or may not contain a minimum number of convex polygons.

Pavlidis (1968) describes a method for the unique unrestricted decomposition of a simple polygon into a minimum number of convex primary sets, where the convex sets may overlap. Chazelle and Dobkin (1979) give an algorithm for producing an unrestricted decomposition of a simple polygon into a minimum number of (pairwise disjoint) convex polygons. Keil (1983) presents dynamic programming algorithms for producing restricted decompositions of a simple polygon into a minimum number of convex polygons and into convex polygons in which the length of the internal edges used to form the decomposition is minimized.

Feng and Pavlidis (1975) give an algorithm for decomposing a simple polygon into convex polygons, in which the decomposition is restricted, not minimal, and depends on the ordering of the vertices of the polygon. In the algorithm of Bykat (1976), the endpoints of decomposition edges may also be mesh vertices on the boundary of the polygon (see C in Section 1.2). His algorithm produces a non-minimal decomposition of a simple polygon into convex polygons where small interior angles are avoided in creating the convex polygons.

Schachter (1978) presents an algorithm for a non-minimal restricted decomposition of a simple polygon into convex polygons. His algorithm is based on a Delaunay triangulation of the polygon; only those Delaunay edges originating at reflex vertices are needed in the decomposition. Garey, Johnson, Preparata, and Tarjan (1978) present an algorithm for triangulating a simple polygon. By removing internal edges which join two convex vertices, their algorithm can be used to produce a non-minimal restricted decomposition into convex polygons.

Let m and n be the number of reflex vertices and vertices, respectively, of a simple polygon. Schachter's algorithm requires a worst case time of $O(mn)$. The algorithm of Garey et al. requires $O(n \log n)$ time. The other algorithms have time complexity greater than $O(mn)$.

In our method, we do not require a minimal decomposition and we allow additional points in the decomposition so that small interior angles in the convex polygons and artificial narrowings in the subregions are avoided. We have based our first stage decomposition algorithm on Schachter's method since it is more efficient than all the other algorithms except for the algorithm of Garey et al.; it provides a way to avoid creating artificial narrowings; it can be modified to avoid creating small interior angles in the convex polygons; and it can be extended to decompose a multiply connected polygonal region (simple polygon with polygonal holes) into convex polygons.

In our decomposition algorithm, there are two computational geometry subproblems. We need to construct the visibility polygon from a point in a simple polygon and construct the Voronoi polygon around a vertex, v, in a set of vertices which are in increasing polar angle order about v. Linear time algorithms for solving the first problem are presented by Lee (1983) and El Gindy and Avis (1981). We use our implementation of Lee's algorithm. We develop a linear time algorithm for solving the second problem by exploiting the ordering of the half-planes corresponding to the perpendicular bisectors of the line segments joining v and the other vertices. This algorithm is similar to the algorithm for the last problem discussed in this section, and both use the same approach as the algorithm of Lee and Preparata (1979) for finding the kernel of a simple polygon. They exploit the ordering of the half-planes corresponding to the polygon edges to obtain a linear time algorithm.

In the second and third stages, we need to compute the diameter (i.e. maximum breadth) of a convex polygon. Shamos (1975) presents a linear time algorithm for solving this problem. In the second stage, we use the minimum breadth (Lyusternik (1963, p. 26)) of a convex polygon to indicate the amount of subregion narrowing in a convex polygon. We compute the minimum breadth in linear time using a modification of Shamos's algorithm for computing the diameter.

In the third stage, we construct a Delaunay triangulation in each convex polygon using a quasi-uniform grid. Some properties of the Delaunay triangulation are stated in Section 5.1. Lawson (1977) presents an algorithm for constructing the Delaunay triangulation of n vertices in an estimated time of $O(n^{4/3})$. Lee and Schachter (1979) present two algorithms for constructing the

Delaunay triangulation. Their first algorithm runs in $O(n \log n)$ time which is asymptotically optimal. Their second algorithm requires $O(n^2)$ time in the worst case but requires roughly $O(n^{3/2})$ time, empirically. These algorithms are given the n vertices as input and construct the Delaunay triangulation of the vertices. In our triangulation method, we generate the vertices as well as the triangles of the triangulation. By using information about the location of the generated vertices, we develop an algorithm for constructing a Delaunay triangulation of n vertices in a convex polygon in expected $O(n)$ time.

In the third stage, we shrink a convex polygon to determine where mesh vertices are generated in the interior of the polygon. In Section 5.2, we present a linear time algorithm for shrinking a convex polygon by a distance $r > 0$. As mentioned earlier, our algorithm uses the same approach as Lee and Preparata (1979).

2.4. Data structure for polygons in the decomposition

In this section we describe a data structure for representing the polygons and their adjacency relations in the decomposition of polygonal region R in the first two stages. Starting from polygons determined by the outer boundary, interfaces, and holes of R, the number of polygons and vertices increase as polygons are subdivided, so we use arrays to represent the list of polygons and vertices, adding new polygons or vertices at the end. We use a circular doubly linked list to store the vertices of a polygon so that it is easy to insert a vertex on an edge and subdivide a polygon into two subpolygons. For each polygon we need to determine the adjacent polygon along each edge and the interior angle of each vertex.

Our data structure consists of a vertex coordinate list, a head vertex list, and a polygon vertex list. The vertex coordinate list is used to stored the coordinates of the polygon vertices and is described at the end of Section 2.2. Note that all polygon vertices become vertices of the triangulation. The head vertex list consists of a one-dimensional integer array, *HEAD*, for storing the index in the polygon vertex list of the head vertex of each polygon.

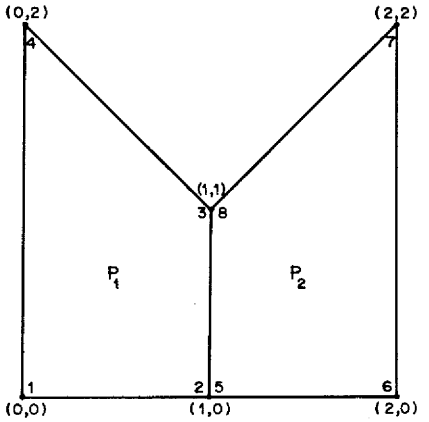
The polygon vertex list consists of six one-dimensional arrays: *LOC*, *POLY*, *SUCC*, *PRED*,

EDGV, and *INTANG*. The first five are integer arrays and the last is a real array. The ith position in the arrays is used to represent a polygon vertex which we call vertex i. $LOC(i)$ is the location of vertex i in the vertex coordinate list, i.e. $(X(LOC(i)), Y(LOC(i)))$ are the coordinates of vertex i. $POLY(i)$ is the index in the head vertex list of the polygon containing vertex i. $INTANG(i)$ is the interior angle of vertex i. $SUCC(i)$ and $PRED(i)$ are the successor and predecessor vertices of vertex i in the counterclockwise direction and are indices in the polygon vertex list. $EDGV(i)$ is used to determine the adjacent polygon along the edge e with vertices i and $SUCC(i)$. If e is a boundary edge of R, then $EDGV(i) = 0$. Otherwise e is an internal edge of R and is the common edge of the polygon with index $POLY(i)$ and one other polygon. $EDGV(i)$ is the index in the polygon vertex list of the vertex $SUCC(i)$ as represented in this other polygon, i.e. $LOC(SUCC(i)) = LOC(EDGV(i))$, $SUCC(i) \neq EDGV(i)$, and $POLY(i)$ and $POLY(EDGV(i))$ are the indices of the two polygon sharing edge e. An example illustrating the three lists is given in Figure 2.2.

In the data structure, polygon k has vertices $i = HEAD(k)$, $SUCC(HEAD(k))$, $SUCC(SUCC(HEAD(k)))$, $\ldots$, $PRED(HEAD(k))$ in counterclockwise order. For each of these vertices i, $POLY(i) = k$ and if $EDGV(i) \neq 0$ then $POLY(EDGV(i))$ is the polygon adjacent to polygon k along the edge joining vertices i and $SUCC(i)$.

Now we describe the initialization and update of the data structure. On input, the interfaces (if any) partition region R into $ns \geq 1$ subregions with $nh \geq 0$ holes. Initially the polygons determined by the outer boundaries of the subregions are given the first ns indices in the head vertex list, and the polygons determined by the boundaries of the holes are temporarily given the next nh indices. The vertices of the outer boundaries are stored in counterclockwise order and the vertices of the holes are stored in clockwise order so that the interior of the subregion is to the left as the boundary is traversed. The interfaces determine adjacency relations between the subregions and these relations are recorded in the *EDGV* array. The interior angle at each vertex is computed from the vertex and its successor and predecessor vertices.

Region R is first decomposed by the method into ns simply connected subregions by intro-



Vertex Coordinate List

j	X(j)	Y(j)
1	0.0	0.0
2	1.0	0.0
3	2.0	0.0
4	2.0	2.0
5	1.0	1.0
6	0.0	2.0

Head Vertex List

k	HEAD(k)
1	1
2	5

Polygon Vertex List

i	LOC(i)	POLY(i)	SUCC(i)	PRED(i)	EDGV(i)	INTANG(i)
1	1	1	2	4	0	90°
2	2	1	3	1	8	90°
3	5	1	4	2	0	135°
4	6	1	1	3	0	45°
5	2	2	6	8	0	90°
6	3	2	7	5	0	90°
7	4	2	8	6	0	45°
8	5	2	5	7	2	135°

Figure 2.2 Illustration of the three lists of data structure for polygons

ducing 'cut' line segments from the boundary of each hole to the outer boundary of the subregion containing the hole (see Section 3.3). In the data structure, the (circular doubly) linked list of vertices of the hole is inserted into the linked list of vertices for the outer boundary. Then the simply connected subregions are decomposed into convex polygons by recursively subdividing polygons into two subpolygons. When a polygon is subdivided, the linked list of vertices for the polygon is split into two linked lists, the adjacency relation between the two subpolygons is recorded in *EDGV*, and one of the subpolygons is given the next index in the head vertex list. If an endpoint of the line segment subdividing a polygon is not a vertex then the endpoint is inserted into the linked list of vertices for the polygon. When the linked lists are updated, any new vertex coordinates or vertices are added at the end of the vertex coordinate list or polygon vertex list, respectively. Therefore the update of the data structure during the decomposition process involves simple operations on linked lists and the addition of elements at the end of the three lists.

CHAPTER 3

INITIAL DECOMPOSITION INTO CONVEX POLYGONS

In this chapter we discuss the first stage of our triangulation method. The initial decomposition is done solely on a geometric basis and the only data required is the description of the polygonal boundary curves of the region, R. We shall first discuss in Sections 3.1 and 3.2 the case of R simply connected, and then, in Section 3.3, indicate our reduction of a multiply connected region R to a simply connected region. To simplify the discussion, we will assume in this chapter that R contains no internal interfaces, since interfaces just reduce R to more than one simply or multiply connected subregions and the data structure described in Section 2.4 records the adjacency relations between the subregions. Algorithms for the decomposition of a simple polygon into convex polygons are discussed in Section 2.3. We have basically used the approach of Schachter (1978), with some features from Bykat (1976). The time complexity of our algorithm is discussed in Section 3.4.

3.1. Simply connected regions

The class of regions treated in this section is broadened from the class of simple polygons to include the simply connected regions obtained by introducing cuts into multiply connected regions as discussed in Section 3.3 below. Consider a subregion R_i in the decomposition of R which has a reflex vertex v on its boundary. The vertex v is *resolved* by decomposing R_i into two or three subregions $R_{i,j}$ $j = 1,2$ and possibly 3, using one or two line segments, i.e. *separators,* which lie in R_i and have v as one endpoint. The separators are chosen so that v is a convex vertex of the resulting subregions, and the $R_{i,j}$ replace R_i in the decomposition of R. The introduction of separators will not produce any new reflex vertices in the $R_{i,j}$; but it is possible that a separator introduced into R_i may resolve a reflex vertex at the other endpoint from v, so the effect of the decomposition of R_i is to reduce the total number of reflex vertices in the decomposition of R by at least one. The decomposition of R then proceeds by a scan down the list of reflex vertices of R. When the scan reaches a reflex vertex, v, the (unique) subregion, R_i, to which v belongs is

identified, and decomposed as described above. Any other reflex vertices resolved by this decomposition are removed from the list and the scan advances to the next (unresolved) reflex vertex. When the scan is complete, a decomposition of R into convex polygons is obtained.

For finite element triangulation, a major criterion in the selection of separators is that they not introduce angles that are small (relative to existing angles) into ∂R_i. In the standard decomposition problem, the separators are restricted to join v to other vertices of ∂R, i.e. it is assumed that all the necessary vertices of ∂R are known a priori. This assumption is overly restrictive for our purposes, so we allow separators to end at nonvertex points on edges of ∂R; but, of course, these points are restricted so that they do not divide an edge into a short and long subedge, thereby introducing extraneously short subedges. In the next section, we discuss our procedure for selecting separators which satisfy the above criteria.

3.2. Selection of separators

Let R be a simply connected polygonal region with vertices $v_0, v_1, \ldots, v_n$ in counterclockwise order, and let v_0 be a reflex vertex of R. Let the interior angle at v_0 be θ_0 ($180° < \theta_0 < 360°$). Define the right cone (RC), inner cone (IC), and left cone (LC), to contain points with polar angles, measured from 0° at v_0v_1, in the intervals $(0°, \theta_0 - 180°)$, $[\theta_0 - 180°, 180°]$, $(180°, \theta_0)$, respectively (see Figure 3.1). Define VP to be the visibility polygon from v_0, i.e. $VP = \{v \mid v \in R \text{ and } v_0v \cap R = v_0v\}$. In a counterclockwise traversal of ∂VP from v_1 to v_n, the points of ∂VP are in nondecreasing polar angle order. Define VS to be the sections of ∂VP (or ∂R) visible from v_0, i.e. the set of boundary points, v, for which the line segment v_0v lies in the interior of R, except for v_0 and v. Linear time algorithms for computing VP (or VS) have been published by El Gindy and Avis (1981), and by Lee (1983); our method uses our implementation of Lee's algorithm.

The reflex vertex v_0 can be resolved by a separator joining v_0 to $w_I \in IC \cap VS$, or by separators joining v_0 to $w_R \in RC \cap VS$ and $w_L \in LC \cap VS$ if $\text{angle}(w_Rv_0w_L) \leq 180°$. For $w \in VS$, let the interior angles determined by separator v_0w be α_0, β_0, α_1, β_1 as indicated in Fig-

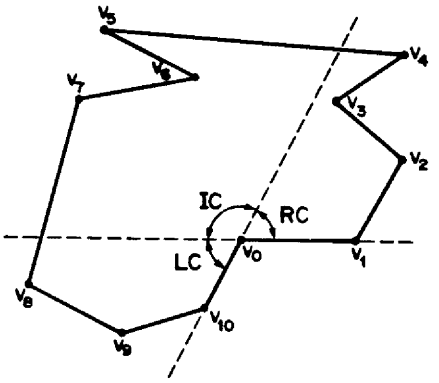


Figure 3.1 Definition of cones *RC*, *IC*, *LC*

ure 3.2a and define $g_1(w) = \min_{i=0,1} \{\alpha_i, \beta_i\}$ (cf. Bykat (1976)). For a pair of separators v_0w_R and v_0w_L define the interior angles $\alpha_0, \beta_0, \gamma_0, \alpha_1, \beta_1, \alpha_2, \beta_2$ as indicated in Figure 3.2b and define $g_2(w_R, w_L) = \min_{i=0,1,2} \{\alpha_i, \beta_i, \gamma_0\}$. In principle then, good candidates for the endpoints of separators from v_0 would be the $w_I \in VS \cap IC$ or $w_R \in VS \cap RC$ and $w_L \in VS \cap LC$, whichever produces $\max\{\theta_1, \theta_2\}$ where

$$\begin{aligned} \theta_1 &= \max\{g_1(w_I) \mid w_I \in VS \cap IC\}, \\ \theta_2 &= \max\{g_2(w_R, w_L) \mid w_R \in VS \cap RC, w_L \in VS \cap LC, \gamma_0 \leq 180^\circ\}. \end{aligned} \tag{3.1}$$

However, since VS is a continuum it is not practical to try to determine these maxima, so we define a suitable subset of VS to search for these maxima.

Let $V_1 = VS \cap \{v_i \mid 2 \leq i < n\}$ be the vertices of R visible from v_0. Clearly, restricting the search in (3.1) to V_1 can introduce small angles into the decomposition as indicated in Figure 3.3. We take advantage of the freedom to add vertices to the triangulation for our application to

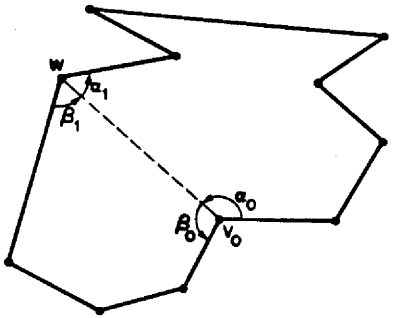


Figure 3.2a Definition of angles α_0, β_0, α_1, β_1 for one separator

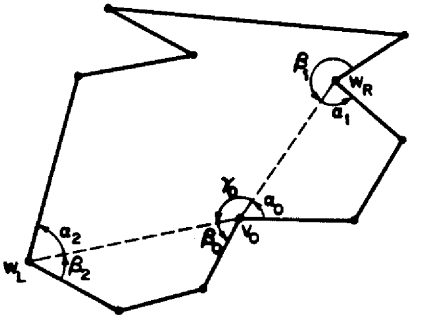


Figure 3.2b Definition of angles α_0, β_0, γ_0, α_1, β_1, α_2, β_2 for two separators

extend the search to a set of points, V_2, situated on edges that subtend large angles at v_0. These edges are identified using a parameter σ in the following way. Let uv be an edge of VS which subtends an angle $\alpha \geq 2\sigma$ at v_0 and let $k = \left\lfloor \alpha/\sigma \right\rfloor$ and $\sigma' = \alpha/k \in [\sigma, 2\sigma)$. Then add to V_2 the $k - 1$ points w_i, $1 \leq i < k$, on uv with equal angular spacing σ' at v_0, i.e. angle$(w_i v_0 u) = i\,\sigma'$ (see Figure 3.3). From elementary geometry, it can be seen that

$$g_1(w_i) \geq \sigma. \tag{3.2}$$

The points of $V_1 \cup V_2$ are found in increasing polar angle order from VP in linear time.

The search for a maximum in (3.1) is carried out over $V_1 \cup V_2$ and is guided by a number of heuristics. If separators are constructed by choosing w_I or w_R and w_L solely on the basis of their angular properties, it is possible for a separator to pass close to a visible vertex which would result in an artificial narrowing in one of the subregions. This is indicated in Figure 3.4; choosing v_0v_8 as a separator would resolve both reflex vertices, but create a narrowing at v_6. In the method of Schachter (1978), the choice of separator endpoints is restricted to Voronoi neighbours of v_0 in V_1, where v_0 and $v \in V_1$ are Voronoi neighbours if their Voronoi polygons are adjacent in the Voronoi tessellation of $V_1 \cup \{v_0\}$. This avoids the phenomenon of creating an artificial narrowing since the Voronoi neighbours are the vertices which are 'closer' to v_0. Therefore we restrict our initial search for separator endpoints w_I or w_L, w_R in (3.1) to the set V of Voronoi neighbours of v_0 in $V_1 \cup V_2$. Figure 3.4 illustrates these sets of boundary points. The points of V are determined in increasing polar angle order from $V_1 \cup V_2$ by constructing the Voronoi polygon around v_0. Each point of V corresponds to an edge of the Voronoi polygon. The Voronoi polygon is constructed in linear time using an algorithm which is similar to that in Lee and Preparata (1979) and in Section 5.2.

We prefer to use one separator v_0w_I rather than two separators v_0w_R and v_0w_L so that fewer subregions are obtained, and we are prepared to accept a separator or pair of separators that make $g_1(w_I)$ or $g_2(w_R, w_L)$ adequately large, rather than search the entire eligible vertex set, $V_1 \cup V_2$, for a maximum as in (3.1). For this purpose let parameter λ indicate the lower bound

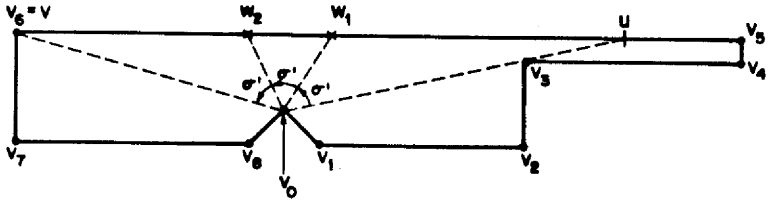


Figure 3.3 Restriction of separator endpoints to $V_1 = \{v_2, v_3, v_6, v_7\}$ causes small angles; allowing separator endpoints to be in $V_2 = \{w_1, w_2\}$ gives better angles

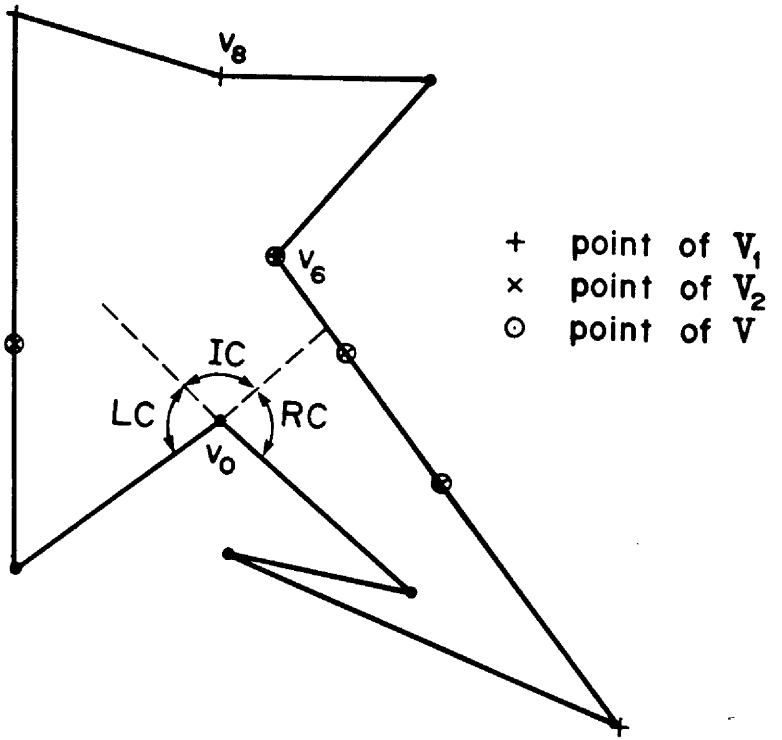


Figure 3.4 Choosing v_0v_8 as a separator creates a narrowing at v_6; restriction of separator endpoints to V avoids narrowings

on $\max_{w_I} g_1(w_I)$ which is acceptable without checking $\max_{w_R, w_L} g_2(w_R, w_L)$, and parameter τ indicate the lower bound on $\max_{V} \{g_1(w_I), g_2(w_R, w_L)\}$ which is acceptable without checking $\max_{V_1 \cup V_2} \{g_1(w_I), g_2(w_R, w_L)\}$.

The following procedure RESOLVE finds w_I or w_R, w_L to resolve v_0 using the angle parameters σ, λ, τ. In the experiments reported in Chapter 6, we have mostly used $\sigma = 30°$, $\lambda = 40°$, and $\tau = 20°$. This procedure returns from the for loop that controls the searches over V and $V_1 \cup V_2$ if acceptable separator(s) is (are) found. Otherwise it returns upon completion of the loop with the best separator(s).

```
Procedure RESOLVE(R, v_0, w_1, w_2);
# Input: simply connected polygonal region R and reflex vertex v_0
# Output: w_1 = w_I, w_2 = nil or w_1 = w_R, w_2 = w_L
   Compute V_1 ∪ V_2 and V using parameter σ;
   for i := 1 to 2 do
      if i = 1 then S := V else S := V_1 ∪ V_2;
      w_1 := nil; w_2 := nil; μ := 0;
      if S ∩ IC ≠ φ then
         Find w_1 such that g_1(w_1) = max {g_1(w_I) | w_I ∈ S ∩ IC};
         if g_1(w_1) ≥ λ then return else μ := g_1(w_1);
      w_I := w_1;
      if there exist w_R ∈ S ∩ RC, w_L ∈ S ∩ LC such that γ_0 ≤ 180° then
         Find w_1, w_2 such that g_2(w_1, w_2) = max {g_2(w_R, w_L) |
            w_R ∈ S ∩ RC, w_L ∈ S ∩ LC, γ_0 ≤ 180°};
         if g_2(w_1, w_2) > μ then μ := g_2(w_1, w_2)
            else w_1 := w_I; w_2 := nil;
      if μ ≥ τ then return;
   return;
```

The following lemma implies that procedure RESOLVE will find either separator v_0w_1 in IC or separators v_0w_1 and v_0w_2 in RC and LC, respectively, to resolve reflex vertex v_0.

Lemma 3.1 : If $V_1 \cap IC = \phi$ then there exist $w_R \in V_1 \cap RC$ and $w_L \in V_1 \cap LC$ such that angle$(w_L v_0 w_R) \leq 180°$.

Proof : Suppose $V_1 \cap IC = \phi$. Then there exists an edge $u_R u_L$ of ∂VP which goes through IC with the two endpoints u_R and u_L in RC and LC, respectively (see Figure 3.5). Let θ_R and θ_L be the polar angles of u_R and u_L, respectively. $\theta_L - \theta_R$ is the angle subtended by edge $u_R u_L$ at v_0 so it must be less than 180°. Let w_R (w_L) be the point of ∂VP closest to v_0 with polar angle

θ_R (θ_L). w_R and w_L are clearly in VS. They are also vertices of R since $VP \subseteq R$. Therefore $w_R \in V_1 \cap RC$, $w_L \in V_1 \cap LC$, and angle$(w_L v_0 w_R) = \theta_L - \theta_R \leq 180°$. □

Now we show that the lower bound for the angles created at the boundary by separators selected by procedure RESOLVE is 0°. Consider the class of 'spiral' polygons illustrated in Figure 3.6 where the polygon is the union of similar triangles determined by the angles a, b, and c, and v_0 is a reflex vertex. If a spiral polygon has angles $a < 2\sigma$ and $b \leq c$ then procedure RESOLVE selects either separator v_0w_1 in IC or separators v_0w_1, v_0w_2 in RC, LC to resolve v_0 where $g_1(w_1) \leq b$ or $g_2(w_1,w_2) \leq b$, respectively. If b is made arbitrarily small, then $g_1(w_1)$ or $g_2(w_1,w_2)$ are also arbitrarily small, so the lower bound for the angles created at the boundary is 0°. This 'worst-case' polygon is not likely to occur in regions for partial differential equations since $|v_0v_n| / |v_0v_1| = |\sin(a+b)/\sin(b)|^{n-1}$ gets very large as b approaches 0°, where $|v_0v|$ denotes the length of edge v_0v, $n = \theta_0/a$, and θ_0 is the interior angle at v_0. For example, if $a = 20°$, $b = 1°$, $n = 11$, then $|v_0v_n| / |v_0v_1| = 1.33 \times 10^{13}$.

However, if the point w_1 or pair of points w_1, w_2 selected by procedure RESOLVE are in V_2, then the following lemma shows that the lower bound for the angles created at the boundary by the separator(s) is σ.

Lemma 3.2 :

(a) If separator v_0w_1 with $w_1 \in V_2 \cap IC$ is selected by procedure RESOLVE to resolve reflex vertex v_0, then $g_1(w_1) \geq \sigma$.

(b) If separators v_0w_1, v_0w_2 with $w_1 \in V_2 \cap RC$, $w_2 \in V_2 \cap LC$ are selected by procedure RESOLVE to resolve v_0, then $g_2(w_1,w_2) \geq \sigma$.

Proof : (a) $g_1(w_1) \geq \sigma$ by (3.2) since $w_1 \in V_2$.

(b) $w_1 \in V_2$ and $w_2 \in V_2$ imply that the angles α_0, β_0, α_1, β_1, α_2, β_2 (see Figure 3.2b) are all at least σ by (3.2). Now we show that $\gamma_0 \geq \sigma$. There are two cases: (i) w_1 and w_2 are on the same edge of ∂VP, or (ii) w_1 and w_2 are on different edges of ∂VP. In case (i), $\gamma_0 \geq \sigma$ by the way points are added to V_2. In case (ii), there exist points u_1, u_2 on ∂VP such that

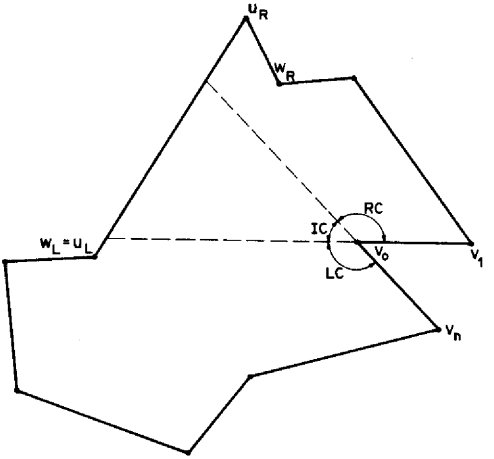


Figure 3.5 Illustration of *VP* for Lemma 3.1

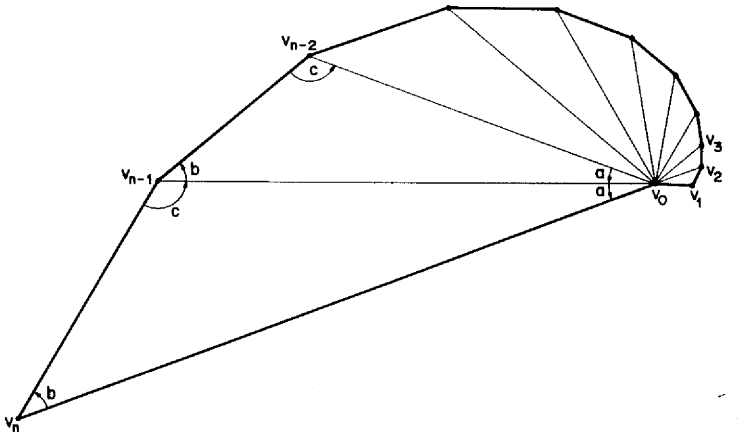


Figure 3.6 Illustration of 'spiral' polygon with $a = 20°$, $b = 40°$, $c = 120°$

$\theta(w_1)+\sigma \leq \theta(u_1) \leq \theta(u_2) \leq \theta(w_2)-\sigma$ where $\theta(v)$ denotes the polar angle of point v, since w_1 and w_2 are introduced on the interior of different edges of ∂VP; $\gamma_0 = \theta(w_2)-\theta(w_1) \geq 2\sigma$. Therefore in both cases $g_2(w_1,w_2) \geq \sigma$. □

3.3. Multiply connected regions

In this section we describe our technique for reducing a multiply connected region R to a simply connected region R' and decomposing R' into simple polygons. Our technique can be intuitively motivated by considering the case of a region R with one hole. Let P and H be simple polygons with boundaries ∂P and ∂H, let H lie in the interior of P, and let $R = P -$ (interior of H), so that $\partial R = \partial P \cup \partial H$. Observe that the top and bottom vertices of H, t and b, are reflex vertices of R. If separators are found from t to a vertex of P above t and from b to a vertex of P below b, then R will be decomposed into two simple polygons. The separator, S, which joins t to ∂P connects two previously disconnected boundary curves of R. It forms a cut in R such that the interior of $R - S$ is a simply connected region with a boundary curve that is not simple because the edge S is traversed twice, once in each direction in a traversal of the boundary of $R - S$. It is the class of simply connected regions with possibly a series of such cuts to which procedure RESOLVE of the preceding section applies.

To formalize these ideas for a general multiply connected region R, let P be a simple polygon, $H_1, H_2, \ldots, H_m$, $m \geq 1$, be nonintersecting simple polygons (holes) in the interior of P, and R be the region defined by $P - \bigcup_{1 \leq i \leq m}$ (interior of H_i). Define the following ordering of points in the plane: $(x_1, y_1) < (x_2, y_2)$ if $y_1 < y_2$ or $y_1 = y_2$ and $x_1 < x_2$. Define the top vertex (bottom vertex) of hole H_i to be the vertex of H_i with largest (smallest) coordinates using this ordering. Let t_i and b_i be the top and bottom vertex, respectively, of H_i. Note that t_i and b_i are reflex vertices of R.

Let Q be a simply connected region and v be a point in the interior of Q. Define subregions $Q_T(v)$ and $Q_B(v)$ as follows. Let l be the horizontal line through v. Let v_L (v_R) be the point on l to the left (right) of v which is on ∂Q and is closest to v. The line segment $v_L v_R$ subdivides Q

into two simply connected subregions $Q_T(v)$ and $Q_B(v)$ such that $Q_T(v)$ is above $v_L v_R$, $Q_B(v)$ is below $v_L v_R$, $Q_T(v) \cup Q_B(v) = Q$, and (interior of $Q_T(v)$) $\cap$ (interior of $Q_B(v)$) $= \phi$. $Q_T(v)$ and $Q_B(v)$ can be similarly defined if v is a reflex vertex of Q such that the y-coordinates of the vertices preceding and succeeding v are both greater or less than the y-coordinate of v.

The decomposition of R into simple polygons is done in two steps (see procedure DECOMPM below). In the first step R is converted into a simply connected region R' by using m separators (cuts) to join ∂H_i, $1 \leq i \leq m$, to ∂P (see Figure 3.7a). In the second step R' is decomposed into k simple polygons by $k-1$ separators where $2 \leq k \leq m+1$ (see Figure 3.7b). Procedure RESOLVE is used to find the separators in both steps. The second step is not required in order to decompose R into convex polygons since the procedure described in Section 3.1 can be applied to R'. We have included it in our implementation since in practice it tends to reduce the size of the polygons which are input to procedure RESOLVE when resolving reflex vertices of R'.

```
Procedure DECOMPM(R, m, Plist, k);
# Input: multiply connected polygonal region R which consists of
#        an outer polygon P and m holes H_1, H_2, ..., H_m
# Output: list of simple polygons Plist = [P_1, P_2, ..., P_k] where k ≤ m+1
# Step 1: convert R into simply connected region R' = P_1
   Order the holes H_i so that t_1 > t_2 > ··· > t_m;
   Q := P;
   for i := 1 to m do
      RESOLVE (Q_T(t_i), t_i, w_i1, w_2);
      Update Q by adding the edges of H_i and separator
         t_i w_i1 twice to ∂Q;
   P_1 := Q;
   k := 1;
# Step 2: decompose R' into simple polygons
   Order the holes H_i so that b_1 < b_2 < ··· < b_m;
   for i := 1 to m do
      if the edges of H_i are all in the same subregion then
         P_j := subregion containing vertex b_i;
         Q := P_j;
         k := k + 1;
         RESOLVE (Q_B(b_i), b_i, w̄_i1, w_2);
         Decompose Q into two subregions, labelled P_j and P_k,
            using separator b_i w̄_i1;
   return;
```

When procedure RESOLVE is used to resolve $v_0 = t_i$ or $v_0 = b_i$ in $Q_T(t_i)$ or $Q_B(b_i)$, respectively, vertex v_0 has an interior angle of 180° so that the left and right cones for v_0 are

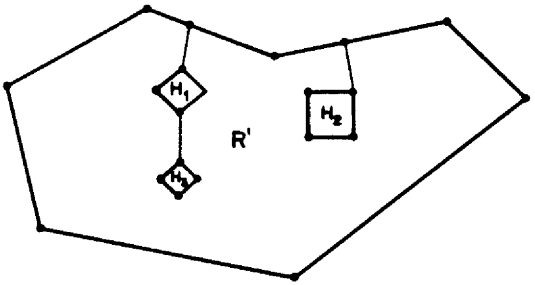


Figure 3.7a Reduction of R to simply connected region R' after Step 1

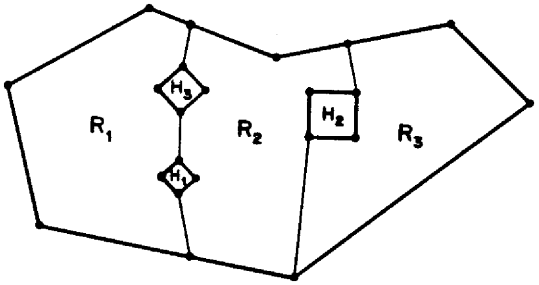


Figure 3.7b Decomposition of R into simple polygons R_1, R_2, R_3 after Step 2

empty and w_2 = nil. In Step 1 of procedure DECOMPM, the holes are ordered so that $H_i, H_{i+1}, \ldots, H_m$ are not in $Q_T(t_i)$ which implies that the separators $t_i w_{i1}$ do not go through any holes and the update of Q is accomplished correctly to get R' at the end of the step. In Step 2, all the edges of H_1 (in the new ordering) lie on $\partial R'$ so separator $b_1 \overline{w}_{11}$ is introduced to decompose R' into two subregions, each containing some edges of H_1. Some of the remaining holes may have their edges divided between the two new subregions as well (e.g. H_3 in Figure 3.7b). However, if there is a hole remaining with all its edges on the boundary of one subregion, this process is repeated with the next such hole in the ordering.

3.4. Summary and time complexity

The decomposition algorithm described in this chapter is summarized in procedure DECOMP. In our implementation, the order in which the reflex vertices are resolved is determined by the order in which they appear in the polygon vertex list (see Section 2.4). The data structure described in Section 2.4 is updated when a 'cut' line segment is added to a subregion or a subregion is decomposed into two or three subregions.

```
Procedure DECOMP(R, m, Plist, N_p);
# Input: simply or multiply connected polygonal region R with m ≥ 0 holes
# Output: list of convex polygons Plist = [P_1, P_2, ..., P_{N_p}]
   if m > 0 then
        DECOMPM(R, m, Plist, N_p);
   else P_1 := R; N_p := 1;
   while there remains an unresolved reflex vertex do
        v_0 := next reflex vertex;
        P_j := subregion containing v_0;
        RESOLVE(P_j, v_0, w_1, w_2);
        if w_2 = nil then
           N_p := N_p + 1;
           Decompose P_j into two subregions, labelled P_j and P_{N_p},
              using separator v_0 w_1;
        else
           N_p := N_p + 2;
           Decompose P_j into three subregions, labelled P_j, P_{N_p-1}, and P_{N_p},
              using separators v_0 w_1 and v_0 w_2;
   return;
```

In the rest of this section, we discuss the time complexity for procedures RESOLVE and DECOMP. We first derive the time complexity for procedure RESOLVE to resolve a reflex vertex

v_0 in a subregion containing n vertices. $V_1 \cup V_2$ contains $O(n)$ vertices and is found in increasing polar angle order in $O(n)$ time (Lee (1983)), provided parameter σ is bounded below. (If σ is bounded below, then the number of points added to V_2 from an edge subtending a large angle at v_0 is bounded by a constant.) V is determined from $V_1 \cup V_2$ in $O(n)$ time (Lee and Preparata (1979)). It requires $O(n)$ time to find the best separator in IC, i.e. w_1 such that $g_1(w_1) = \max\{g_1(w_I) \mid w_I \in S \cap IC\}$ where S is either V or $V_1 \cup V_2$. Let $|S \cap RC|$ ($|S \cap LC|$) denote the number of vertices in $S \cap RC$ ($S \cap LC$), and let $m = |S \cap RC| \cdot |S \cap LC|$. It requires $O(m)$ time to find the best pair of separators in RC and LC, i.e. w_1, w_2 such that

$$g_2(w_1, w_2) = \max\{g_2(w_R, w_L) \mid w_R \in S \cap RC,\ w_L \in S \cap LC,\ \gamma_0 \leq 180°\}. \tag{3.3}$$

(In practice few reflex vertices are resolved with two separators due to the use of parameter λ.) In the worst case $|S \cap RC| = |S \cap LC| = O(n)$ and $m = O(n^2)$. Therefore, in the worst case, procedure RESOLVE requires $O(n^2)$ time to find the vertices w_1 and w_2. The following modification can be made to procedure RESOLVE to reduce the time complexity to $O(n)$. The function g_1 can be used to select the 'best' $O(\sqrt{n})$ vertices from $S \cap RC$ and $S \cap LC$. Then an approximation to (3.3) and w_1, w_2 can be found in $O(n)$ time using these vertices. In the sequel, we will assume that this modification has been made to procedure RESOLVE.

Now we derive the time complexity for procedure DECOMP to decompose a polygonal region R with N vertices, r reflex vertices, and m holes into convex polygons. Each hole contains at least two reflex vertices (the top and bottom vertices), so

$$2m \leq r \leq N. \tag{3.4}$$

The sorting of the holes in procedure DECOMPM requires $O(m \log m)$ time. Each subregion $Q_T(t_i)$ or $Q_B(b_i)$ in procedure DECOMPM is found in $O(N)$ time since R contains N vertices, so it requires $O(mN)$ time to determine at most $2m$ of these subregions. Procedure RESOLVE is invoked at most $2m$ times in procedure DECOMPM and at most r times in the while loop of procedure DECOMP. Let $q \leq 2m + r$ be the number of times procedure RESOLVE is invoked, Q_i be the subregion of the ith invocation, and n_i be the number of vertices in Q_i, $1 \leq i \leq q$. The ith invocation of procedure RESOLVE and the subsequent addition of the separator(s) to the

decomposition (i.e. the update of the data structure described in Section 2.4) requires $O(n_i)$ time. Therefore the time required for all the invocations of procedure RESOLVE is $O(T)$ where $T = \sum_{i=1}^{q} n_i$. In the worst case, each subregion Q_i is decomposed by one separator into a triangle and a subregion containing n_i vertices; $n_i = O(N)$ for all i and $T = O(qN)$. Therefore the time complexity of procedure DECOMP, in the worst case, is $O(rN)$ by (3.4), which is the same as Schachter's method.

CHAPTER 4

FURTHER DECOMPOSITION TO BOUNDARY SCALES

In the decomposition of region R into convex polygons $P_1, P_2, \ldots, P_{N_p}$ of the preceding chapter, the length scales defined by the boundary and interface edges and narrowings of R are present in the lengths of edges of the P_i. In this chapter, we define a mesh distribution function, $\psi(x,y)$, using the lengths of edges of the P_i which, in conjunction with the desired number of triangles N_t and the mesh smoothness parameter κ, indicates the triangle size to be used in a neighbourhood of (x,y). To satisfy objectives (2.1c) and (2.1a), we aim to produce a triangulation which is approximately equidistributing (see Section 1.3) of ψ in R and contains approximately N_t triangles, i.e.

$$\iint_{\Delta} \psi \, dA \approx \iint_{R} \psi \, dA / N_t \tag{4.1}$$

for all triangles, Δ, in the triangulation. Our approach is to further decompose the P_i into convex polygons such that the variation of ψ in each is limited, and triangulate each subregion with triangles of approximately constant area (or uniform size). The triangulation stage is discussed in the next chapter. In Section 4.1 we discuss the definition and evaluation of $\psi(x,y)$. In Section 4.2 we describe the further subdivision of the P_i into convex polygons of limited variation of ψ. In Section 4.3 we describe the computation of an appropriate uniform triangle size to use on each polygon. In Section 4.4 we discuss the time complexity of the second stage.

4.1. Definition of mesh distribution function

The mesh distribution function, $\psi(x,y)$, is defined in terms of the smoothness parameter κ and a 'preliminary' distribution function, $\eta(x,y)$, which is defined in each polygon, P_i, of the decomposition of R using the length scales of P_i and its neighbours. From (4.1), it can be seen that in an equidistributing triangulation the value of ψ in a triangle is inversely proportional to the triangle area (or square of triangle size). Thus we define $\eta(x,y)$ to be inversely proportional to the square of $l(x,y)$, where $l(x,y)$ represents the length scale (or relative triangle size) at the point

(x,y).

Consider a polygon P_i with m vertices v_j and edges e_j, $0 \leq j \leq m-1$, in counterclockwise order and let e_j have vertices v_j and v_{j+1}. We determine the length scales for points of P_i as follows. Let w_i be the width (i.e. minimum breadth, Lyusternik (1963, p.26)) of polygon P_i. We use w_i to measure the amount of narrowing in P_i and it is computed in linear time using a modification of the algorithm of Shamos (1975) for computing the diameter (i.e. maximum breadth) of a convex polygon. A length scale s_j is defined for each edge e_j of P_i by

$$s_j = \min(|e_j|, w_i, w_k) \tag{4.2}$$

where $|e_j|$ denotes the length of edge e_j, and $w_k = \infty$ if e_j is a boundary edge of R, otherwise w_k is the width of the polygon, P_k, adjacent to P_i on e_j. For each vertex v_j of P_i define t_j to be the minimum of the length scales (as computed in (4.2)) of all the edges of the decomposition which have v_j as one endpoint, i.e.

$$t_j = \min\{|e|, w_k \mid \text{edge } e \text{ or polygon } P_k \text{ contains vertex } v_j\}. \tag{4.3}$$

For an arbitrary point (x,y) of P_i, we define for each vertex v_j of P_i

$$d_j(x,y) = \max\{(\text{distance of } (x,y) \text{ to } v_j)/2,\ t_j\}$$

and for each edge e_j of P_i

$$f_j(x,y) = \max\{(\text{distance of } (x,y) \text{ to } e_j)/2,\ s_j\}.$$

We define a length scale for (x,y) in terms of these truncated distances as

$$l(x,y) = \min\{w_i,\ \min_{0 \leq j < m} d_j(x,y),\ \min_{0 \leq j < m} f_j(x,y)\} \tag{4.4}$$

and define

$$\eta(x,y) = 1/l^2(x,y).$$

We have defined $l(x,y)$ so that it has smaller values near edges and vertices of P_i with s_j and t_j values smaller than w_i (e.g. near short edges) and it gradually increases to a maximum possible value of w_i towards the centre and the other edges of P_i at a rate determined by $1/2$ of the distance of (x,y) to the parts of ∂P_i with small values of s_j and t_j. Let

$$\mu = \min\{w_i, \min_{0 \le j < m} s_j, \min_{0 \le j < m} t_j\}.$$

Then

$$\mu \le l(x,y) \le w_i \tag{4.5}$$

for all (x,y) in P_i.

From $\eta(x,y)$ we define the mesh distribution function, $\psi(x,y)$, in terms of the smoothing parameter κ, $0 \le \kappa \le 1$, as

$$\psi(x,y) = (1-\kappa)(\eta(x,y)/\bar{\eta}A) + \kappa/A \tag{4.6}$$

where A is the area of R and $\bar{\eta}$ is the mean of η over R. A is computed as the sum of the areas of the P_i and $\bar{\eta}$ is computed using numerical quadrature over the P_i as described in the next section. Note that $\psi(x,y)$ has been defined with the normalization $\iint_R \psi \, dA = 1$ and it is piecewise continuous in R (it is continuous in each P_i). Also, $\psi(x,y)$ has the most variation when $\kappa = 0$ and becomes smoother (i.e. has less variation) as κ increases; when $\kappa = 1$, $\psi(x,y)$ is a constant function.

The most time consuming part of the second stage is the evaluation of $\psi(x,y)$ for determining its variation in the polygons (see Sections 4.2 and 4.4), so we want to compute $\psi(x,y)$, or equivalently $l(x,y)$, in P_i as efficiently as possible. We make the following observations about the evaluation of $l(x,y)$. If $s_j \ge w_i$ then $f_j(x,y) \ge w_i$ and since $l(x,y) \le w_i$ by (4.5), $f_j(x,y)$ does not have to be computed in (4.4). Similarly if $t_j \ge w_i$ then $d_j(x,y)$ does not have to be computed in (4.4). If $t_j = \min\{s_{j-1}, s_j\}$ then $d_j(x,y) \ge \min\{f_{j-1}(x,y), f_j(x,y)\}$ so $d_j(x,y)$ does not have to be computed in (4.4). Therefore only a subset of the $d_j(x,y)$ and $f_j(x,y)$ have to be computed in the evaluation of $l(x,y)$ as indicated in the following pseudo-code.

```
function l(x,y);
   val := w_i;
   for j := 0 to m − 1 do
      if s_j < w_i then
         val := min(val, f_j(x,y));
      if j > 0 then k := j − 1 else k := m − 1;
      if t_j < min(w_i, s_j, s_k) then
         val := min(val, d_j(x,y));
   l := val;
```

Further efficiency is obtained in the implementation of our method by a preprocessing step to determine the subset of the $d_j(x,y)$ and $f_j(x,y)$ that have to be evaluated for (x,y) in P_i; if this subset is empty then $l(x,y)=w_i$ is constant in P_i. Also, the squares of the constants s_j, t_j, w_i and the functions $d_j(x,y)$, $f_j(x,y)$, $l(x,y)$ are computed so that square root calculations are avoided.

4.2. Subdivision of convex polygon

In this section we discuss when and how a convex polygon P_i is further subdivided to get subregions of limited variation of ψ. To estimate the variation of ψ, we compute its mean, $\overline{\psi}_i$ and standard deviation, σ_i, over P_i using numerical integration as follows. We triangulate P_i with triangles whose vertices are the centroid, c_i, of P_i and vertices on the boundary edges of P_i, either at the endpoints or introduced on the edges that subtend large angles at c_i as in Section 3.2 (i.e. if an edge subtends an angle of $\geq 2\sigma$ then it is subdivided into k segments subtending equal angles at c_i). Let the resulting vertices of P_i be relabelled v_j, $0 \leq j \leq M-1$, in counterclockwise order, let the edges of P_i be v_jv_{j+1}, and let the triangles of P_i be $T_j = \Delta v_jv_{j+1}c_i$, where all indices of v_j and T_j are taken modulo M in this section. Let $A_{i,j}$ be the area of T_j, $A_i = \sum_{j=0}^{M-1} A_{i,j}$ be the area of P_i, and $A = \sum_{i=1}^{N_p} A_i$ be the area of R. Let $\overline{\eta}_{i,j}$ be the mean of η over T_j, $\overline{\eta}_i$ be the mean of η over P_i, and $\overline{\eta}$ be the mean of η over R. Using numerical quadrature over the triangles T_j (Strang and Fix (1973, p.184)) we can compute:

$$\bar{\eta}_{i,j} = \iint_{T_j} \eta \, dA \, / A_{i,j}$$
$$\bar{\eta}_i = \sum_{j=0}^{M-1} A_{i,j} \bar{\eta}_{i,j} \, / A_i \tag{4.7}$$
$$\bar{\eta} = \sum_{i=1}^{N_p} A_i \bar{\eta}_i \, / A.$$

From (4.6) and (4.7), $\bar{\psi}_i$ can be computed as

$$\bar{\psi}_i = (1 - \kappa)(\bar{\eta}_i / \bar{\eta} A) + \kappa / A.$$

Similarly

$$\sigma_i = [\iint_{P_i} (\psi(x,y) - \bar{\psi}_i)^2 \, dA / A_i]^{1/2}$$

can be computed from the standard deviation of η over P_i.

We can obtain a preliminary estimate for the number of triangles, n_i, to be generated in P_i if we wish to have N_t triangles in total from

$$n_i = N_t \iint_{P_i} \psi \, dA = N_t \, (A_i / A)((1 - \kappa) \, \bar{\eta}_i / \bar{\eta} + \kappa). \tag{4.8}$$

However, ψ may be modified by a further procedure described in Section 4.3 to prevent large differences in triangle sizes between adjacent subregions.

Further subdivision of P_i takes place if the estimated variation of ψ is sufficiently high, and the estimated number of triangles for P_i is sufficiently large. We parameterize the meaning of 'sufficiently high' and 'sufficiently large' using parameters $d_{\min}$ and $n_{\min}$, i.e. the criteria for further subdivision of P_i are that

$$\sigma_i / \bar{\psi}_i > d_{\min} \quad \text{and} \quad n_i > n_{\min}. \tag{4.9}$$

In the experiments reported in Chapter 6, we have mostly used $d_{\min} = 0.5$ and $n_{\min} = 10$. In the event that criteria (4.9) call for P_i to be subdivided, we select a suitable separator with endpoints from $\{v_j \mid 0 \leq j < M\}$ which does not create relatively small angles with the boundary of P_i. We attempt to choose a separator that isolates the higher values of η (or equivalently ψ) on one side so as to reduce the variation of η (or ψ) in each of the two resulting subpolygons. Failing this, we choose a separator on the basis of the shape of P_i.

To identify a separator that partitions P_i into regions of higher and lower η, we determine an index l such that $\overline{\eta}_{i,l} = \max_{0 \le j < M} \overline{\eta}_{i,j}$, and build up a sector, S, which has area less than $A_i/2$, by adding adjacent triangles with larger than average values of $\overline{\eta}_{i,j}$, starting with T_l, i.e.

$$S = \cup\{T_{l-p}, T_{l-p+1}, \ldots, T_{l+q-1}\}$$

with $p \ge 0$, $q \ge 1$, and $T_k \in S$ implying $\overline{\eta}_{i,k} \ge \overline{\eta}_i$. We now attempt to select a separator from the four possibilities arising from picking one of v_{l-p-1}, v_{l-p} for one endpoint and one of v_{l+q}, v_{l+q+1} for the other (see Figures 4.1a and 4.1b). The selection is based on maximizing the minimum angle created at the boundary as in Section 3.2.

It may happen, however, that this process fails to identify a separator (as illustrated in Figure 4.1b) or that the minimum angle chosen from the process is unacceptably small (i.e. smaller than $\tau = 20°$ in our experiments). In this case, we default to a selection of a separator based on the shape of P_i. Let $v_b v_t$ be a diameter of P_i (see Shamos (1975) for efficient computation of a diameter) and let q be the perpendicular bisector of $v_b v_t$ as in Figure 4.2. The directed line segment $v_b v_t$ partitions ∂P_i into a left and right side (if $v_b v_t$ is on ∂P_i then it is one of the two sides). Let w_1 and w_2 be the endpoints of the edge intersected by q on the left side of ∂P_i ($w_1 = w_2 = v$ if q intersects ∂P_i at a vertex v). Similarly let z_1 and z_2 be the endpoints of the edge intersected by q on the right side of ∂P_i. We select as the separator one of the up to four line segments $w_j z_k$, $1 \le j,k \le 2$, using a maximum minimum angle criterion as above.

The following lemmas establish that the second strategy is guaranteed to identify a separator, and that it is likely that this separator will not create small angles with ∂P_i. Let $d = |v_b v_t|$ and $D(P_i)$ be the region formed from the intersection of the two circles of radius d centred at v_b and v_t (see Figure 4.3). Since d is the diameter of P_i, $D(P_i)$ must contain P_i.

Lemma 4.1 : Let v be a point of $D(P_i)$ which is not v_b or v_t. Then angle$(v_b v v_t) \ge 60°$.

Proof : Without loss of generality, assume v is to the right of $v_b v_t$ and in the top half of $D(P_i)$. First suppose v is on $\partial D(P_i)$ (see Figure 4.4a). Then $\Delta v_b v v_t$ is an isosceles triangle. Let α be the two equal angles and β be the third angle. Then $\beta \le$ angle$(v_b v_r v_t) = 60°$ where v_r is the

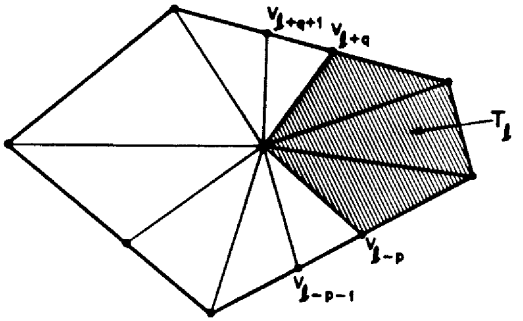


Figure 4.1a Illustration of sector S (shaded) and possible endpoints for a separator

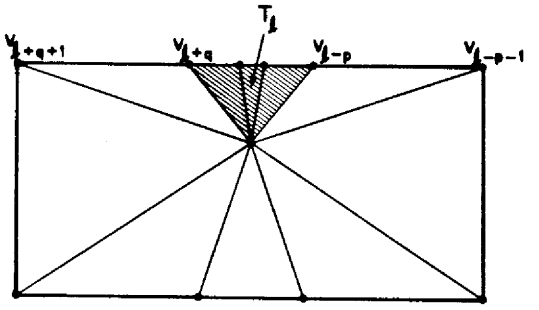


Figure 4.1b Example where sector S (shaded) does not identify a separator

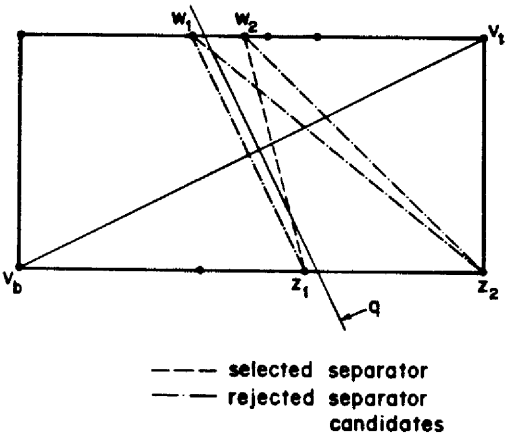


Figure 4.2 Selection of separator based on shape of polygon

point such that $\Delta v_b v_r v_t$ is an equilateral triangle. Therefore angle$(v_b v v_t) = \alpha \geq 60°$. Now suppose v is in the interior of $D(P_i)$ (see Figure 4.4b). Let v' be the intersection of the line through $v_b v$ and $\partial D(P_i)$. Let $\alpha = \text{angle}(v_b v v_t)$, $\alpha' = \text{angle}(v_b v' v_t) = \text{angle}(v' v_t v_b)$, $\beta = \text{angle}(v_t v_b v)$, and $\gamma = \text{angle}(v v_t v_b)$. As above, $\beta \leq 60°$ and $\alpha' \geq 60°$. Therefore $\alpha > \alpha' \geq 60°$ since $\gamma < \alpha'$. □

Lemma 4.2 :

(a) The interior angle at all vertices of P_i, except possibly at v_b and v_t, are $\geq 60°$.

(b) Let angle$(v_b c_i v_t)$ and angle$(v_t c_i v_b)$ be the two angles at the centroid c_i made by edges $v_b c_i$ and $v_t c_i$. Then the minimum of these two angles is $> 60°$.

(c) Let v_m be the midpoint of $v_b v_t$ and $v \neq v_m$ be a point of $D(P_i)$ on q, the perpendicular bisector of $v_b v_t$. Then angle$(v_b v v_m)$ and angle$(v_m v v_t)$ are both $\geq 30°$.

Proof : (a) and (b) follow from Lemma 4.1. (c) follows from Lemma 4.1 by considering the

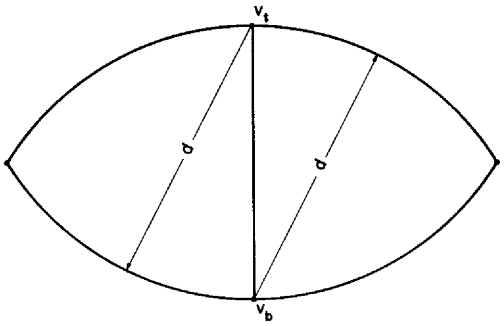


Figure 4.3 Illustration of $D(P_i)$

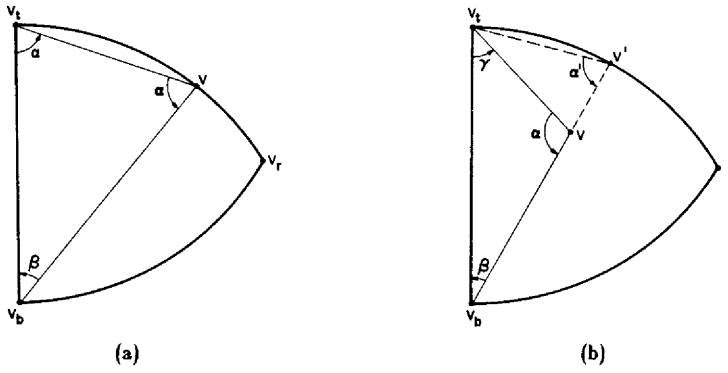


Figure 4.4 (a) v is on $\partial D(P_i)$, (b) v is in the interior of $D(P_i)$

congruent triangles $\Delta v_b v v_m$ and $\Delta v_m v v_t$. □

Suppose $\sigma \leq 30°$. All edges of P_i subtend an angle of less than 2σ at c_i, so at least one of the vertices w_1, w_2 (or similarly z_1, z_2) is not v_b or v_t by Lemma 4.2(b). Hence at least one of the line segments $w_j z_k$ is a separator of P_i. If the line segment which is the intersection of line q and P_i were used as a separator, then the minimum angle at ∂P_i would be $\geq 30°$ by Lemma 4.2(c). Since we are choosing a separator which is near this line segment and the vertices w_1, w_2, z_1, z_2 are not likely to be v_b or v_t and thus likely to have interior angles $\geq 60°$ by Lemma 4.2(a), it is likely that the minimum angle at ∂P_i from using this separator is acceptable (i.e. larger than $\tau = 20°$).

The selected separator subdivides P_i into two convex subpolygons and these subpolygons can also be further subdivided using criteria (4.9). Note that the factors used in the first criterion of (4.9) depend on P_i and its neighbours but not on N_t directly, whereas n_i grows with N_t at a rate that depends on P_i (see (4.8)). If P_i passes the first test for subdivision in (4.9) but not the second, it is deemed to have significant variation in ψ, but N_t is not large enough to sustain subdividing it; it will be referred to as 'underpopulated' in Chapter 6. When N_t is large enough that no subregion is underpopulated, then the second stage decomposition is constant for all larger N_t, and we refer to the decomposition as 'saturated'.

4.3. Computation of triangle sizes

When the decompositions of Chapter 3 and Section 4.2 have been carried out, we have a set of convex polygons $P_1, P_2, \ldots, P_{N_S}$ (relabelled from the first stage) each of which fails to satisfy one of criteria (4.9) for further subdivision. We now wish to determine the number and size of triangles to generate in each P_i so as to obtain a triangulation of R of approximately N_t triangles. Let h_i be the uniform triangle size to be used in P_i. h_i^2 will be inversely proportional to $\bar{\psi}_i$, the mean value of ψ over P_i. However, we do not want an abrupt change in triangle size from one polygon to a neighbouring one, so we impose the constraint that

$$1/2 \leq h_i / h_j \leq 2 \tag{4.10}$$

for all adjacent polygons P_i, P_j. To satisfy (4.10) we modify the $\bar{\psi}_i$ to $\bar{\psi}_{mod,i}$ so that

$$1/4 \leq \bar{\psi}_{mod,i} / \bar{\psi}_{mod,j} \leq 4 \tag{4.11}$$

for all adjacent polygons P_i, P_j. This modification is done by the following algorithm, using a heap to keep track of the maximum $\bar{\psi}_{mod,i}$ value at each step.

$$\begin{array}{l} I := \{1,2,\ldots,N_s\}; \\ \text{for } i := 1 \text{ to } N_s \text{ do } \bar{\psi}_{mod,i} := \bar{\psi}_i; \\ \text{for } l := 1 \text{ to } N_s - 1 \text{ do} \\ \quad \text{Find } i \in I \text{ such that } \bar{\psi}_{mod,i} = \max_{k \in I} \bar{\psi}_{mod,k}; \\ \quad I := I - \{i\}; \\ \quad \text{for } P_j \text{ adjacent to } P_i \text{ do} \\ \quad\quad \text{if } \bar{\psi}_{mod,j} < \bar{\psi}_{mod,i}/4 \text{ then } \bar{\psi}_{mod,j} := \bar{\psi}_{mod,i}/4; \end{array} \tag{4.12}$$

From (4.12), $\bar{\psi}_{mod,i} \geq \bar{\psi}_i$ so this modification removes the normalization of the $\bar{\psi}_i$, (i.e. $\sum_{i=1}^{N_s} \bar{\psi}_i A_i = 1$), so we compute

$$\Psi = \sum_{i=1}^{N_s} \bar{\psi}_{mod,i} A_i \tag{4.13}$$

and set $\psi_{mod,i} = \bar{\psi}_{mod,i} / \Psi$ to restore it. We now approximately equidistribute N_t triangles over the P_i using the $\psi_{mod,i}$ distribution, i.e. we attempt to generate n_i triangles in P_i for

$$n_i = N_t \, \psi_{mod,i} \, A_i. \tag{4.14}$$

Our triangulation technique for P_i, described in Chapter 5, generates triangles of constant area $h_i^2/2$ in the interior of P_i and triangles of approximately constant area $h_i^2/2$ near ∂P_i. The appropriate triangle size can hence be approximately determined by $n_i \, h_i^2/2 = A_i$, i.e.

$$h_i = \sqrt{2/(N_t \, \psi_{mod,i})} \tag{4.15}$$

using (4.14). Since the number of triangles in the interior of each P_i increases as N_t increases, (4.15) determines the triangle sizes more accurately and the number of triangles generated in R is closer to N_t for larger values of N_t.

4.4. Summary and time complexity

The second stage of our triangulation method is summarized in procedure EQDIST.

Procedure EQDIST($Plist, N_p, N_t, \kappa, hlist, N_s$);
Input: list of convex polygons $Plist = [P_1, \ldots, P_{N_p}]$,
desired number of triangles N_t, and mesh smoothness parameter κ
Output: further decomposition into convex polygons $Plist = [P_1, \ldots, P_{N_S}]$
and list of uniform triangle sizes $hlist = [h_1, \ldots, h_{N_S}]$
where $N_s \geq N_p$ and the polygons are relabelled from input
Define $\psi(x,y)$ as described in Section 4.1;
$N_s := N_p$;
for $j := 1$ to N_p do
$top := 1$;
$stack(top) := j$;
while $top \geq 1$ do
$i := stack(top)$;
$top := top - 1$;
Compute $\bar{\psi}_i$, σ_i, A_i, n_i;
if $\sigma_i / \bar{\psi}_i > d_{\min}$ and $n_i > n_{\min}$ then
$N_s := N_s + 1$;
Subdivide P_i into two convex subpolygons, labelled P_i and P_{N_S},
as described in Section 4.2;
$top := top + 2$;
$stack(top-1) := N_s$;
$stack(top) := i$;
Execute (4.12);
$\Psi := \sum_{i=1}^{N_s} \bar{\psi}_{mod,i} A_i$;
for $i := 1$ to N_s do
$\psi_{mod,i} := \bar{\psi}_{mod,i} / \Psi$;
$h_i := \sqrt{2/(N_i \psi_{mod,i})}$;
return;

In the rest of this section, we discuss the time complexity of the second stage. We first derive the time complexity for computing the constants required in the definition of $\psi(x,y)$ in Section 4.1 and for the evaluation of ψ at a point (x,y). Let m be the maximum number of edges of a polygon P_i from the first stage. The width and area of a convex polygon can each be computed in linear time (Shamos (1975)) so the w_i, $1 \leq i \leq N_p$, and the area of R are computed in $O(mN_p)$ time. The length scales s_j and t_j (see (4.2) and (4.3)) in all the P_i can be computed in $O(mN_p)$ time. The integral of $\eta(x,y)$ in P_i is approximated by numerical quadrature over triangles using $O(m)$ function evaluations, so the computation of $\bar{\eta}$ for (4.6) requires $O(mN_p)$ evaluations of $\eta(x,y)$. Each evaluation of $\eta(x,y)$ requires at most $2m$ evaluations of the functions $d_j(x,y)$ and

$f_j(x,y)$ (see discussion at the end of Section 4.1). $d_j(x,y)$ and $f_j(x,y)$ can each be computed in constant time so each evaluation of $\eta(x,y)$ or $\psi(x,y)$ requires a worst case time of $O(m)$.

Now we derive the time complexity for the further decomposition into convex polygons $P_1, \ldots, P_{N_S}$ as described in Section 4.2. The computation of the integrals for $\overline{\psi}_i$ and σ_i in P_i require $O(m)$ evaluations of $\psi(x,y)$ using numerical quadrature (provided parameter σ is bounded below). There are N_p polygons at the end of the first stage and $N_s - N_p$ subdivisions are made in the second stage to obtain N_s polygons. Integrals are first computed in the N_p polygons from the first stage. Each subdivision results in two more polygons in which integrals are computed. Therefore integrals are computed in a total of $N_p + 2(N_s - N_p)$ polygons and the total number of evaluations of $\psi(x,y)$ is $O(mN_s)$. The time required for finding a separator in P_i, subdividing P_i, and computing the areas of the two subpolygons is $O(m)$, so $O(mN_s)$ time is required for processing the polygons (excluding function evaluations).

Now we derive the time complexity for the computation of the triangle sizes as described in Section 4.3. In (4.12), we store the $\overline{\psi}_i$ values in a heap which is built in $O(N_s)$ time. The removal of the maximum $\overline{\psi}_{mod,i}$ value from the heap requires $O(\log N_s)$ time. Each $\overline{\psi}_{mod,i}$ value is modified at most once and the update of the heap after a modification requires $O(\log N_s)$ time. Therefore $O(N_s \log N_s)$ time is required to maintain the heap in (4.12). The time required to determine the neighbours of polygons in (4.12) is $O(mN_s)$. Then the triangle sizes h_i are computed in $O(N_s)$ time.

Therefore the time complexity of the second stage is $O(mN_s)$ evaluations of ψ or η, plus $O(mN_s + N_s \log N_s)$ for other computations. Since each evaluation of ψ or η requires a worst case time of $O(m)$, the worst case time complexity for the function evaluations is $O(m^2 N_s)$. In practice, most of the CPU time in the second stage is spent on the function evaluations.

CHAPTER 5

TRIANGULATION OF CONVEX POLYGONS

In Chapters 3 and 4, we have described our method for decomposing the region R into convex polygons $P_1, P_2, \ldots, P_{N_S}$, with an appropriate triangle size, h_i, determined for each P_i. In this chapter we describe our method for triangulating P_i with triangles of size h_i (area $h_i^2/2$) in its interior, and triangles which are graded in size to merge with those of the neighbours of P_i near its boundary. To accomplish this grading, we assign a triangle size to each edge of the decomposition of R. An edge, e, of this decomposition is either (a) a boundary edge of R and an edge of one convex polygon, P_i, or (b) an internal edge of R and the common edge of two convex polygons, P_i and P_j. Let $\bar{h}(e) = h_i$ in case (a) and $\bar{h}(e) = \sqrt{h_i h_j}$ in case (b). Then the triangle size assigned to edge e is $h(e)$, computed as the nearest length to $\bar{h}(e)$ which is an integral submultiple of $|e|$ (the length of e), i.e.

$$\begin{aligned} &k := \text{trunc}(|e|/\bar{h}(e)); \\ &r := |e|/\bar{h}(e) - k; \\ &\text{if } r > k/(2k+1) \text{ then } k := k + 1; \\ &h(e) := |e|/k; \end{aligned} \tag{5.1}$$

From (4.10) and (5.1),

$$\min(|e|, \sqrt{2}h_i/3) \le h(e) \le 4\sqrt{2}h_i/3 \tag{5.2}$$

for any edge, e, of P_i.

In the rest of this chapter, we will discuss a single polygon, P, with triangle sizes specified for each edge, $h(e)$, and for the interior, h. In our algorithm, mesh vertices are generated in the interior of P using a quasi-uniform grid of spacing h and on each edge, e, of P at an equal spacing of $h(e)$ (cf. Shaw and Pitchen (1978)); and a Delaunay triangulation of these vertices is constructed in P (Lawson (1977), Lee and Schachter (1979)). In the standard triangulation problem of computational geometry, in which the vertices are given as input and a (Delaunay) triangulation is constructed to cover the convex hull of the vertices, the construction of a Delaunay triangulation (or any triangulation) requires $O(n_t \log n_t)$ time, where n_t is the number of triangles in

the triangulation (Shamos and Hoey (1975), Lee and Schachter (1979)). However, in finite element triangulation, the vertices are generated as well as the triangles. By using information about the location of the vertices relative to each other, our algorithm constructs a Delaunay triangulation of the vertices in an expected time of $O(n_t)$.

In Section 5.1, we define a valid triangulation and a Delaunay triangulation. In Section 5.2, we describe an algorithm for shrinking a convex polygon which we use to determine a convex polygon, $int(P)$, in the interior of P. In our triangulation algorithm, we construct a preliminary triangulation, $VT(P)$, by generating triangles in $int(P)$ and then in the strip near ∂P. These procedures are described in Sections 5.3 and 5.4, respectively. The validity of $VT(P)$ is discussed in Section 5.5. In Section 5.6, we describe how $VT(P)$ is converted into a Delaunay triangulation, $DT(P)$. The time complexity of the triangulation algorithm is discussed in Section 5.7.

5.1. Valid and Delaunay triangulations

In this section we state conditions for a valid triangulation in a region and state some properties of a Delaunay triangulation of a set of vertices. We say that a collection of triangles is a valid triangulation of a polygonal region of the plane if the triangles form a 'tiling' of the region without overlaps or gaps. Let w_1, w_2, w_3 be distinct vertices. We define triangle $\Delta w_1 w_2 w_3$ to be a counterclockwise (CCW) triangle if the interior of the triangle is to the left of the three directed edges w_1w_2, w_2w_3, w_3w_1; otherwise $\Delta w_1 w_2 w_3$ is a clockwise (CW) triangle. If w_1, w_2, w_3 are collinear, we consider $\Delta w_1 w_2 w_3$ to be a CW triangle. Note that the ordering of the vertices of a triangle determine whether it is a CCW or CW triangle. Four conditions that ensure that a collection of triangles, $\Delta_1, \Delta_2, \ldots, \Delta_N$, is a valid triangulation of a region have been established by Simpson (1981). They can be described in geometric terms as follows:

(a) Δ_i is a CCW triangle for all i.

(b) Each edge e of Δ_i is either the only edge joining its vertices or there is one other triangle, Δ_j, having an edge joining these vertices. In the latter case, the direction of e as an edge of Δ_i is opposite to its direction as an edge of Δ_j. In the former case, e is a boundary edge and Δ_i is its boundary triangle. (5.3)

(c) No interval of a boundary edge intersects a triangle other than its boundary triangle.

(d) A vertex can have at most one boundary edge directed away from it.

Let V be a set of vertices in the plane such that they are not all collinear. A Delaunay triangulation of V is a (valid) triangulation in the convex hull of V which satisfies the max-min angle criterion: for any two triangles in the triangulation that share a common edge, if the quadrilateral formed from the two triangles with the common edge as its diagonal is strictly convex, the replacement of the diagonal by the alternative one does not increase the minimum of the six angles in the two triangles making up the quadrilateral (Sibson (1978)). In other words, the Delaunay triangulation is the triangulation which maximizes the minimum angle in the triangles globally as well as locally in any two adjacent triangles which form a strictly convex quadrilateral.

A Delaunay triangulation also satisfies the circle criterion: the circumcircle of any triangle in the triangulation contains no vertex of V in its interior. The Delaunay triangulation is unique if no four vertices are co-circular. An edge uv, where u and v are vertices of V, is a Delaunay edge if it is an edge in a Delaunay triangulation of V.

Lemma 5.1 (Lee and Schachter (1979)) : An edge uv is a Delaunay edge if and only if there exists a point c such that the circle centred at c and passing through u and v does not contain any other vertex of V in its interior.

The following local optimization procedure (LOP) of Lawson (1977) can be used to convert a triangulation of V into a Delaunay triangulation. Let e be an internal edge (i.e. an edge not on

the boundary of the convex hull) of a triangulation of V and Q be the quadrilateral formed by the two triangles having e as a common edge. If the circumcircle of one of the triangles contains the fourth vertex of Q in its interior, then e is replaced by the other diagonal of Q (so that the minimum of the angles in the two triangles is increased). Edge e is said to be *locally optimal* if an application of LOP would not swap it. Note that e is locally optimal if Q is a nonconvex quadrilateral, and that the validity conditions (5.3) are unaffected by the LOP.

Theorem 5.1 (Lawson (1977)) : All internal edges of a triangulation T of V are locally optimal if and only if T is a Delaunay triangulation of V.

In our algorithm, we generate a preliminary valid triangulation, $VT(P)$, in P which satisfies the four conditions of (5.3). Then $VT(P)$ is converted into a Delaunay triangulation, $DT(P)$, by applying LOP to the internal edges of $VT(P)$. We have constructed $VT(P)$ so that LOP only has to be applied to a subset of the internal edges and few swaps are expected to be made to obtain $DT(P)$.

5.2. Shrinking a convex polygon

Let $p_0, p_1, \ldots, p_{m-1}, p_m$ be the vertices of convex polygon P in counterclockwise order, where $p_m = p_0$ and all interior angles are less than 180°. Let Q be obtained by shrinking P by a distance of $r > 0$, i.e. if $v \in Q$ then the distance from v to $\partial P \geq r$. In this section, we present an algorithm for constructing Q which uses the same approach as the algorithm of Lee and Preparata (1979) for finding the kernel of a simple polygon. They exploit the ordering of the half-planes corresponding to the polygon edges to obtain a linear time algorithm.

First we give some notation for representing Q and ∂Q. Let l_i be the directed line from p_i to p_{i+1}, L_i be the directed line parallel to l_i and at distance r to the left of l_i, and H_i be the half-plane to the left of and including L_i. Let qL_iq' denote the directed line segment from q to q' where q and q' are two points on L_i, and let $qL_i\infty$ and ∞L_iq denote the directed half-lines starting and ending at q, respectively, on L_i. Q is the intersection of the half-planes $H_0, H_1, \ldots, H_{m-1}$. If r is sufficiently small then Q is a convex polygon otherwise Q is degen-

erate, i.e. Q is either empty or a single point or a line segment. In the degenerate case, we treat Q as empty. If Q is a convex polygon then ∂Q consists of line segments from $n \geq 3$ of the lines L_i (if $\bigcap_{i \neq j} H_i = Q$ then no line segment of L_j is part of ∂Q). Using the above notation

$$\partial Q = q_0 L_{k_0} q_1 L_{k_1} q_2 \cdots q_{n-1} L_{k_{n-1}} q_n, \quad 0 \leq k_0 < k_1 < \cdots < k_{n-1} \leq m-1,$$

where $q_n = q_0$ is the intersection of L_{k_0} and $L_{k_{n-1}}$ and q_i is the intersection of $L_{k_{i-1}}$ and L_{k_i}, $1 \leq i \leq n-1$ (see Figure 5.1).

Our algorithm for constructing Q scans in order the edges $p_i p_{i+1}$ of P and constructs a sequence of convex polygons $Q_1, Q_2, \ldots, Q_{m-1}$ such that Q_i is the intersection of half-planes $H_0, H_1, \ldots, H_i$. Since P is a convex polygon, the polar angles, θ_i, of the vectors $p_{i+1} - p_i$ (or lines l_i) are ordered. Without loss of generality assume $\theta_0 = 0°$, i.e. l_0 is a horizontal line directed from left to right, so that $0° = \theta_0 < \theta_1 < \cdots < \theta_{m-1} < 360°$. Let s be the smallest index such that $\theta_s > 180°$ (note that $2 \leq s \leq m-1$). If $\theta_{s-1} = 180°$ then let $s' = s-2$ otherwise let $s' = s-1$. The following facts can be seen from elementary geometry:

(a) For $1 \leq i \leq s'$, Q_i is an unbounded convex polygon.

(b) If $\theta_{s-1} = 180°$, Q_{s-1} is either an unbounded convex polygon or degenerate. (5.4)

(c) For $s \leq i \leq m-1$, Q_i is either a bounded convex polygon or degenerate.

Now we describe our algorithm for constructing Q. Initially Q_1 is the intersection of H_0 and H_1 and $\partial Q_1 = \infty L_0 q_1 L_1 \infty$ where q_1 is the intersection of L_0 and L_1. For $2 \leq i \leq s'$, Q_i is the intersection of Q_{i-1} and H_i and ∂Q_i is obtained by modifying ∂Q_{i-1}. Suppose

$$\partial Q_{i-1} = q_0 L_{k_0} q_1 \cdots q_n L_{k_n} q_{n+1}, \; q_0 = q_{n+1} = \infty, \; 0 = k_0 < k_1 < \cdots < k_n = i-1. \tag{5.5}$$

Since Q_{i-1} is convex and $\theta_j < \theta_i < 180°$ for $j < i$, ∂Q_{i-1} and L_i intersect at exactly one point, t (see Figure 5.2). Let $j \geq 0$ be the index such that t is on the line segment $q_j L_{k_j} q_{j+1}$ and $t \neq q_j$. j can be found by scanning backwards from n for the first q_j to the left of L_i where $q_0 = \infty$ is considered to be to the left of L_i. The polygonal curve $t L_{k_j} q_{j+1} \cdots q_{n+1}$ is to the right of L_i so it is not part of ∂Q_i. It is replaced by $t L_i \infty$ to obtain $\partial Q_i = q_0 \cdots q_j L_{k_j} t L_i \infty$ which can be

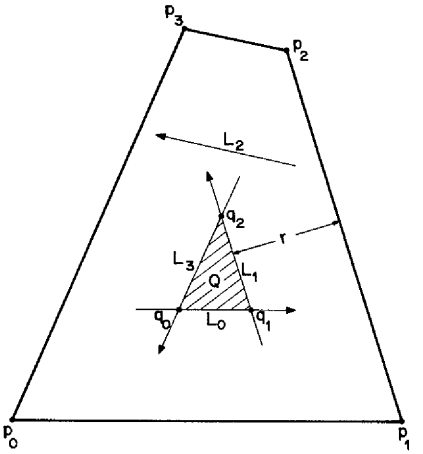


Figure 5.1 Illustration of convex polygon Q

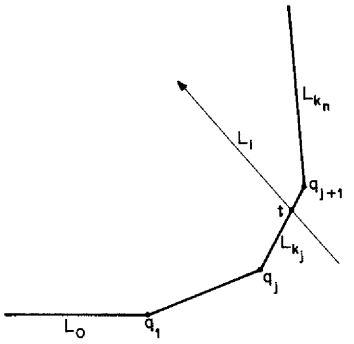


Figure 5.2 ∂Q_{i-1} and L_i intersect at point t

expressed in the same form as (5.5). If $\theta_{s-1} = 180°$ then if all points of ∂Q_{s-2} (in particular $q_1, q_2, \ldots, q_n$) are to the right of or on L_{s-1} then Q_{s-1} is degenerate and the algorithm terminates otherwise ∂Q_{s-1} is obtained by modifying ∂Q_{s-2} as above.

Suppose Q_{s-1} is not degenerate. Q_s is either a bounded convex polygon or degenerate by (5.4c) and ∂Q_s is obtained by modifying ∂Q_{s-1}. If all points of ∂Q_{s-1} are to the right of or on L_s then Q_s is degenerate and the algorithm terminates. Otherwise L_s intersects ∂Q_{s-1} at exactly two points, t_1 and t_2, since Q_{s-1} is convex and $\theta_s > 180°$. Suppose ∂Q_{s-1} is in the form of (5.5) with $i = s$ and t_1 occurs before t_2 in the polygonal curve (see Figure 5.3). Suppose j and l are the indices $(j < l)$ such that t_1 is on line segment $q_j L_{k_j} q_{j+1}$ and $t_1 \neq q_{j+1}$ and t_2 is on line segment $q_l L_{k_l} q_{l+1}$ and $t_2 \neq q_l$. l can be found by scanning backwards from n for the first q_l to the left of L_s and j can be found by scanning forward from 0 for the first q_{j+1} to the left of L_s. The polygonal curves $t_2 L_{k_l} q_{l+1} \cdots q_{n+1}$ and $q_0 \cdots q_j L_{k_j} t_1$ are to the right of L_s so they are not part of ∂Q_s. They are replaced by $t_2 L_s t_1$ to obtain $\partial Q_s = t_1 L_{k_j} q_{j+1} \cdots q_l L_{k_l} t_2 L_s t_1$ which can be expressed in the same form as (5.6) below.

Suppose $s+1 \leq i \leq m-1$ and Q_{i-1} is not degenerate. ∂Q_i is obtained by modifying

$$\partial Q_{i-1} = q_0 L_{k_0} q_1 \cdots q_n L_{k_n} q_{n+1},\ q_0 = q_{n+1},\ 0 \leq k_0 < k_1 < \cdots < k_n \leq i-1. \tag{5.6}$$

Consider the position of L_i with respect to Q_{i-1}. There are three cases (see Figure 5.4):

(a) Q_{i-1} is to the right of or on L_i,

(b) Q_{i-1} is to the left of or on L_i,

(c) ∂Q_{i-1} and L_i intersect at exactly two points, t_1 and t_2.

In case (a), $q_0, q_1, \ldots, q_n$ are to the right of or on L_i, Q_i is degenerate, and the algorithm terminates. In case (b), $Q_i = Q_{i-1}$ and the condition that q_0 is to the left of or on L_i is sufficient to determine this case since $\theta_{k_0} \geq 0°$ and $\theta_{k_n} < \theta_i$, i.e. the slope of L_i lies between the slopes of L_{k_n} and L_{k_0}. In case (c), ∂Q_i is determined in the same way as ∂Q_s above. If Q_i is not degenerate for $s'+1 \leq i \leq m-1$, then the algorithm terminates with $\partial Q = \partial Q_{m-1}$.

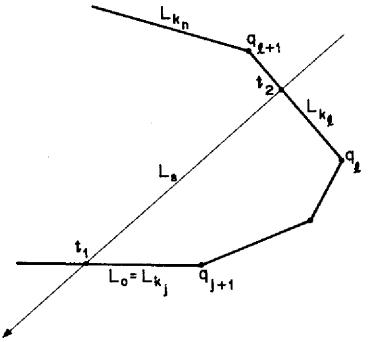


Figure 5.3 ∂Q_{s-1} and L_s intersect at points t_1 and t_2

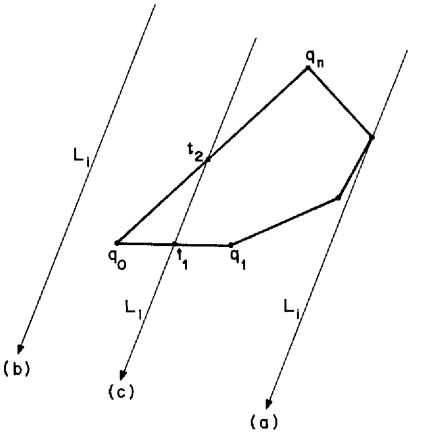


Figure 5.4 Three cases of position of L_i with respect to Q_{i-1}

The pseudo-code for our algorithm is given in procedure SHRINK.

Procedure SHRINK(P, m, r, Q, n);
Input: convex polygon $P = p_0 p_1 \cdots p_{m-1} p_m$ and shrinking distance r
Output: $Q = \phi$ and $n = 0$ or convex polygon $Q = q_0 q_1 \cdots q_{n-1} q_n$ and $3 \leq n \leq m$
  $\alpha :=$ polar angle of $p_1 - p_0$;
  $q_1 :=$ intersection of L_0 and L_1;
  $k_0 := 0$; $k_1 := 1$;
  $i := 2$; $n := 1$;
  $\theta :=$ (polar angle of $p_3 - p_2$) $- \alpha$;
  Translate angle θ to the interval $[0°, 360°)$;
  while $\theta \leq 180°$ do
    # polygonal boundary curve is $\infty L_{k_0} q_1 \cdots q_n L_{k_n} \infty$ and next line is L_i
    while $n \geq 1$ and q_n is to the right of or on L_i do $n := n - 1$;
    if $\theta = 180°$ and $n = 0$ then return;
    $t :=$ intersection of L_{k_n} and L_i;
    $n := n + 1$; $q_n := t$; $k_n := i$;
    $i := i + 1$; $\theta :=$ (polar angle of $p_{i+1} - p_i$) $- \alpha$;
    Translate angle θ to the interval $[0°, 360°)$;
  $j := 0$; $q_j := \infty$; $q_{n+1} := \infty$;
  while $i \leq m - 1$ do
    # polygonal boundary curve is $q_j L_{k_j} q_{j+1} \cdots q_n L_{k_n} q_{n+1}$ and next line is L_i
    if $q_j = \infty$ or q_j is to the right of L_i then
      while $n \geq j + 1$ and q_n is to the right of or on L_i do $n := n - 1$;
      if $n = j$ then $n := 0$; return;
      $t_2 :=$ intersection of L_{k_n} and L_i;
      $n := n + 1$; $q_n := t_2$; $k_n := i$;
      while q_{j+1} is to the right of or on L_i do $j := j + 1$;
      $t_1 :=$ intersection of L_{k_j} and L_i;
      $q_j := t_1$; $q_{n+1} := t_1$;
    $i := i + 1$;
  for $i := 0$ to $n + 1 - j$ do $q_i := q_{i+j}$;
  $n := n + 1 - j$;
  return;

We now derive the time complexity for the algorithm. For each $i \geq 2$, ∂Q_i is obtained from ∂Q_{i-1} by removing zero or more line segments from the beginning and end of ∂Q_{i-1}. Each line segment is parallel to an edge of ∂P and when it is removed from ∂Q_{i-1} it is not involved in any further computation. Therefore at most m line segments are removed from the ∂Q_i, $1 \leq i \leq m-2$. All other computations clearly require $O(m)$ time. Therefore we have shown

Theorem 5.2 : Procedure SHRINK determines Q in $O(m)$ time.

5.3. Triangulation of the interior of P

Let $int(P)$ be obtained by shrinking P by a distance of $h/\sqrt{2}$, i.e.

$$int(P) = \{v \mid v \in P \text{ and distance from } v \text{ to } \partial P \geq h/\sqrt{2}\}. \tag{5.7}$$

If $h/\sqrt{2}$ is sufficiently small then $int(P)$ is a convex polygon, otherwise $int(P)$ is degenerate, i.e. it is empty, a point, or a line segment. In the degenerate case, no interior triangulation is done, so we assume that $int(P)$ is a convex polygon in this section.

Let v_b and v_t be the endpoints of a diameter of $int(P)$. The algorithm of Shamos (1975) is used to compute the diameter in linear time. We rotate the coordinate system so that line segment $v_b v_t$ is parallel to the y-axis with $y(v_t) > y(v_b)$, where $y(v)$ denotes the y-coordinate of vertex v. We introduce $n+1$ horizontal lines, $y = y_i$, through $int(P)$, evenly spaced h apart with $n = \left\lfloor (y(v_t) - y(v_b))/h \right\rfloor$; $dy = (y(v_t) - y(v_b) - n\,h)/2$; and $y_i = y(v_t) - dy - i\,h$, $0 \leq i \leq n$. Let a_i and b_i be the x-coordinates of the left and right endpoints of $y = y_i$ in $int(P)$ and let $m(i) = \left\lfloor (b_i - a_i)/h \right\rfloor$; $dx_i = (b_i - a_i - m(i)h)/2$; and $x_{i,j} = a_i + dx_i + jh$, $0 \leq j \leq m(i)$. On each line, we introduce a sequence of mesh vertices $(x_{i,j}, y_i)$ for $0 \leq j \leq m(i)$, spaced h apart. Note that at least one vertex is generated on each line and that the vertex which is nearest to ∂P is between $h/\sqrt{2}$ and $h/2 + h/\sqrt{2}$ distance from ∂P (see Figure 5.5). These vertices do not lie on a uniform square grid of spacing h because those on $y = y_{i+1}$ may be shifted horizontally with respect to those on $y = y_i$ or $y = y_{i+2}$. Consequently we have referred to them as being on a *quasi-uniform* grid.

A subset (possibly empty) of these vertices are then triangulated in a scan down the strips $y_{i+1} \leq y \leq y_i$ (see Figure 5.6). For each pair of lines, let $a = \max(x_{i,0}, x_{i+1,0})$ and $b = \min(x_{i,m(i)}, x_{i+1,m(i+1)})$. (Note that it is possible that $b < a$. In this case $a - b \leq h$ since on lines $y = y_i$ and $y = y_{i+1}$ there exists a vertex at distance $\leq h/2$ from diameter $v_b v_t$.) The vertices on the two lines for which the x-coordinate is in the interval $[a - h, b + h]$ are then connected up to form a sequence of similar triangles in which the area is $h^2/2$ and the angles are between 45° and 90° inclusive. This process is carried out for each pair of lines in which $a \leq b$

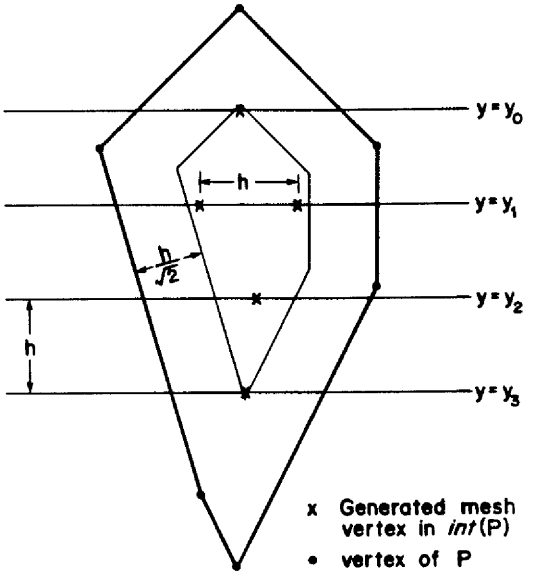


Figure 5.5 Generation of mesh vertices in $int(P)$

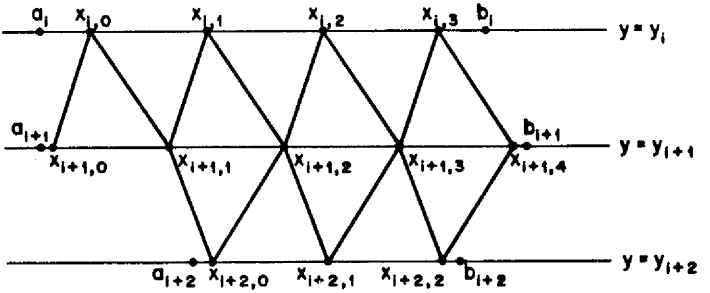


Figure 5.6 Triangulation of mesh vertices in $int(P)$

and $m(i) + m(i+1) > 0$. If $b < a$ or $m(i) = m(i+1) = 0$, then no triangles are formed between $y = y_i$ and $y = y_{i+1}$. In this case, the shortest edge joining a vertex on each line is introduced.

We need to identify a boundary of this set of vertices, so that we can triangulate the strip between it and ∂P. Let V_I be the set of vertices, and E_I be the following set of edges:

(a) edges of triangles formed in the scans described above,

(b) additional edges joining consecutive vertices at the ends of lines $y = y_i$, (5.8)

(c) the shortest edge joining a vertex on line $y = y_i$ to a vertex on line $y = y_{i+1}$, in the case that no triangles are formed between these lines.

Then $G_I = (V_I, E_I)$ is a connected planar graph, and if it contains at least two vertices, then a counterclockwise closed walk, C, can be formed to identify the boundary of G_I by including an edge e of E_I k times in C if e occurs in $2-k$ triangles, $0 \leq k \leq 2$. If an edge e occurs twice in C, the direction is opposite in its two occurrences (see Figure 5.8). If V_I contains only one vertex then let C be this single vertex.

C consists of a walk, C_L, down the left side of G_I followed by the sequence of edges on the line $y = y_n$ from left to right, and a walk, C_R, up the right side of G_I followed by the sequence of edges on the line $y = y_0$ from right to left. C_L consists of paths from $(x_{i,0}, y_i)$ to $(x_{i+1,0}, y_{i+1})$, $i = 0,1, \ldots, n-1$, which are constructed as follows. The leftmost edge between lines $y = y_i$ and $y = y_{i+1}$ is either part of the triangulation or the edge formed in (5.8c). This edge joins vertices $(x_{i,j}, y_i)$ and $(x_{i+1,k}, y_{i+1})$ with either $j = 0$ or $k = 0$ (or both). If $j = 0$ then the path formed by the vertices $(x_{i,0}, y_i)$, $(x_{i+1,k}, y_{i+1})$, $(x_{i+1,k-1}, y_{i+1})$, $\ldots$, $(x_{i+1,0}, y_{i+1})$ is a subwalk of C_L, otherwise the path formed by the vertices $(x_{i,0}, y_i)$, $(x_{i,1}, y_i)$, $\ldots$, $(x_{i,j}, y_i)$, $(x_{i+1,0}, y_{i+1})$ is a subwalk of C_L. For example, $(x_{i,0}, y_i)$, $(x_{i+1,0}, y_{i+1})$, $(x_{i+1,1}, y_{i+1})$, $(x_{i+2,0}, y_{i+2})$ is a subwalk of C_L in Figure 5.6. Similarly C_R consists of paths from $(x_{i+1,m(i+1)}, y_{i+1})$ to $(x_{i,m(i)}, y_i)$, $i = n-1, n-2, \ldots, 0$. C_L and the reverse of C_R are constructed during the scan of lines to form the triangles of $int(P)$.

The following lemma implies that the edges of E_I (see (5.8)) are locally optimal in any triangulation of the mesh vertices (including those on ∂P) which contains these edges.

Lemma 5.2 : If $e \in E_I$, then e is a Delaunay edge.

Proof : E_I can be partitioned into two disjoint sets of edges E_1 and E_2 where E_1 contains the horizontal edges and E_2 contains the edges which join vertices on two consecutive lines $y = y_i$ and $y = y_{i+1}$. First suppose $e \in E_1$. Let the vertices of edge e be $v_1 = (x_{i,j}, y_i)$ and $v_2 = (x_{i,j} + h, y_i)$. Let S be the circle of radius $h/2$ with centre at $(x_{i,j} + h/2, y_i)$ (see Figure 5.7a). Clearly no vertices of V_I can be in the interior of S. Mesh vertices on ∂P lie a distance $\geq h/\sqrt{2}$ from the centre of S by (5.7), so they are not in the interior of S. By Lemma 5.1, e is a Delaunay edge.

Now suppose $e \in E_2$. Let the vertices of edge e be $v_1 = (x_{i,j}, y_i)$ and $v_2 = (x_{i+1,k}, y_{i+1})$, and let $d = |x_{i,j} - x_{i+1,k}|$. Let S be the circle whose diameter is e. It is apparent from Figure 5.7b that no vertices of V_I can be in the interior of S if $d \leq h$. To establish this, we note that e may have been formed by either (5.8a) or (5.8c). In the first case, it is clear that $d \leq h$ as a consequence of the triangulation process in $int(P)$ (see Figure 5.6). In the second case, $d = a - b \leq h$ as discussed in the paragraph above (5.8). The radius of S is $\sqrt{h^2 + d^2}/2 \leq h/\sqrt{2}$. As above, mesh vertices on ∂P are not in the interior of S. By Lemma 5.1, e is a Delaunay edge. □

The pseudo-code for the generation of mesh vertices, triangles, and closed walk C in $int(P)$ is given in procedure INTTRIANG. In this procedure, a triangle is represented by a list of three vertex coordinates in counterclockwise order, the function $append(list1, list2)$ appends the elements of the second list at the end of the first list, and the function $reverse(list)$ reverses the elements of the list.

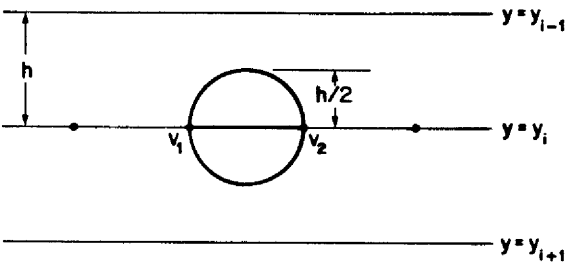


Figure 5.7a Edge v_1v_2 is in E_1

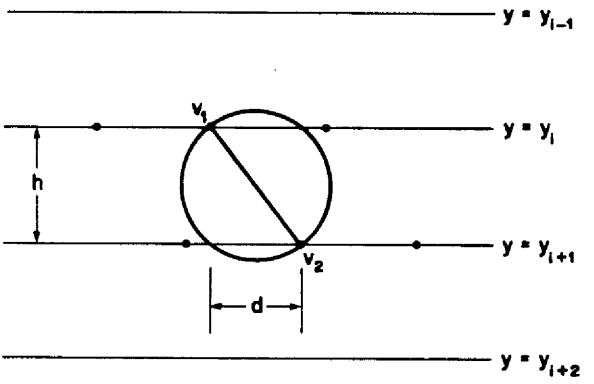


Figure 5.7b Edge v_1v_2 is in E_2

Procedure INTTRIANG($int(P), h, TI, nt, C, nc$);
Input: convex polygon $int(P)$ with diameter parallel to y-axis and triangle size parameter h
Output: list of triangles TI, number of triangles nt, closed walk C, number of edges nc in C
  # Generate mesh vertices in $int(P)$
  $ymax$:= maximum y-coordinate of vertices of $int(P)$;
  $ymin$:= minimum y-coordinate of vertices of $int(P)$;
  n := trunc($(ymax - ymin)/h$);
  dy := $(ymax - ymin - n\,h)/2$;
  for i := 0 to n do y_i := $ymax - dy - i\,h$;
  Scan down the left and right sides of $\partial int(P)$ to determine the points (a_i, y_i) and (b_i, y_i);
  for i := 0 to n do
    $m(i)$:= trunc($(b_i - a_i)/h$);
    dx_i := $(b_i - a_i - m(i)h)/2$;
    for j := 0 to $m(i)$ do $x_{i,j}$:= $a_i + dx_i + j\,h$;
  nt := 0; TI := [ ];
  if n = 0 then nc := 0; C := $[(x_{0,0}, y_0)]$; return;
  C_L := $[(x_{0,0}, y_0)]$; C_R := $[(x_{0,0}, y_0), (x_{0,1}, y_0), \ldots, (x_{0,m(0)}, y_0)]$;
  for i := 0 to $n-1$ do
    a := $\max(x_{i,0}, x_{i+1,0})$;
    b := $\min(x_{i,m(i)}, x_{i+1,m(i+1)})$;
    $l0$:= smallest integer such that $x_{i,l0} \geq a - h$;
    $l1$:= smallest integer such that $x_{i+1,l1} \geq a - h$;
    $r0$:= largest integer such that $x_{i,r0} \leq b + h$;
    $r1$:= largest integer such that $x_{i+1,r1} \leq b + h$;
    # Generate the triangles between $y = y_i$ and $y = y_{i+1}$
    if $l0 < r0$ or $l1 < r1$ then
      j := $l0$; k := $l1$;
      while $j < r0$ and $k < r1$ do
        if $x_{i+1,k} \leq x_{i,j}$ then
          nt := $nt + 1$; $TI(nt)$:= $\Delta[(x_{i+1,k}, y_{i+1}), (x_{i+1,k+1}, y_{i+1}), (x_{i,j}, y_i)]$;
          k := $k + 1$;
        else
          nt := $nt + 1$; $TI(nt)$:= $\Delta[(x_{i,j+1}, y_i), (x_{i,j}, y_i), (x_{i+1,k}, y_{i+1})]$;
          j := $j + 1$;
      if j = $r0$ then while $k < r1$ do
          nt := $nt + 1$; $TI(nt)$:= $\Delta[(x_{i+1,k}, y_{i+1}), (x_{i+1,k+1}, y_{i+1}), (x_{i,j}, y_i)]$;
          k := $k + 1$;
      else while $j < r0$ do
          nt := $nt + 1$; $TI(nt)$:= $\Delta[(x_{i,j+1}, y_i), (x_{i,j}, y_i), (x_{i+1,k}, y_{i+1})]$;
          j := $j + 1$;
    # Generate the subwalks of C_L and C_R between $y = y_i$ and $y = y_{i+1}$
    if $x_{i,l0} \leq x_{i+1,l1}$ then
      C_L := append(C_L, $[(x_{i,1}, y_i), (x_{i,2}, y_i), \ldots, (x_{i,l0}, y_i), (x_{i+1,0}, y_{i+1})]$)
    else C_L := append(C_L, $[(x_{i+1,l1}, y_{i+1}), (x_{i+1,l1-1}, y_{i+1}), \ldots, (x_{i+1,0}, y_{i+1})]$);
    if $x_{i,r0} \leq x_{i+1,r1}$ then
      C_R := append(C_R, $[(x_{i+1,r1}, y_{i+1}), (x_{i+1,r1+1}, y_{i+1}), \ldots, (x_{i+1,m(i+1)}, y_{i+1})]$)
    else C_R := append(C_R, $[(x_{i,m(i)-1}, y_i), (x_{i,m(i)-2}, y_i), \ldots, (x_{i,r0}, y_i), (x_{i+1,m(i+1)}, y_{i+1})]$);
  C_L := append(C_L, $[(x_{n,1}, y_n), (x_{n,2}, y_n), \ldots, (x_{n,m(n)-1}, y_n)]$);
  C := append(C_L, reverse(C_R));
  nc := number of edges in C;
  return;

5.4. Triangulation near the boundary of P

We describe a procedure for triangulating the strip, A, between ∂P and C in the case of at least one vertex in the interior of P. The case in which $int(P)$ is degenerate and there are no vertices in the interior of P will be discussed later in this section. Mesh vertices are generated on each edge e of ∂P at an equal spacing of $h(e)$ (see (5.1)). We believe that the spacing of mesh vertices on ∂P given by (5.1), (5.2) is small enough for the following procedure to generate a valid triangulation in A. However, if the spacing of vertices on ∂P is too large then an invalid triangulation can be formed. We show how the procedure can be modified to produce a valid triangulation in this case. In Section 5.6 we describe how to modify the triangulation of A so that the total triangulation of P is a Delaunay triangulation.

If there are at least two vertices in $int(P)$, then let the closed walk C determined in the preceding section be represented by the list of vertices $C = [v_0, v_1, \ldots, v_{nc}]$ where $v_0 = v_{nc} = (x_{0,0}, y_0)$ and $nc \geq 2$ is the number of edges $v_j v_{j+1}$ in C; otherwise let $C = [v_0]$ and $nc = 0$. Let the counterclockwise cycle of edges on ∂P be represented by the list of mesh vertices $B = [u_0, u_1, \ldots, u_{nb}]$ where $u_0 = u_{nb}$, $nb \geq 3$ is the number of edges $u_i u_{i+1}$ in B, and the numbering of the u_i is done as follows (see Figure 5.8). If there are at least two vertices in $int(P)$ then we number the u_i so that u_0 is closest to v_0 among the vertices on ∂P with y-coordinate greater than y_0 (note that edge $u_0 v_0$ is entirely in A). Otherwise we number the u_i so that u_0 is the vertex on ∂P which is closest to v_0. In the former case the selection of u_0 is a heuristic for finding a Delaunay edge $u_0 v_0$. In the latter case,

Lemma 5.3 : If there is only one vertex v_0 in $int(P)$ and u_0 is the vertex on ∂P closest to v_0 then $u_0 v_0$ is a Delaunay edge.

Proof : Let S be the circle of radius $|u_0 v_0|$ centred at v_0. S contains no vertices in its interior since u_0 is the closest vertex to v_0. Let S' be the circle of radius $|u_0 v_0|/2$ whose diameter is $u_0 v_0$. S' is contained inside S therefore S' contains no vertices in its interior. By Lemma 5.1, $u_0 v_0$ is a Delaunay edge. □

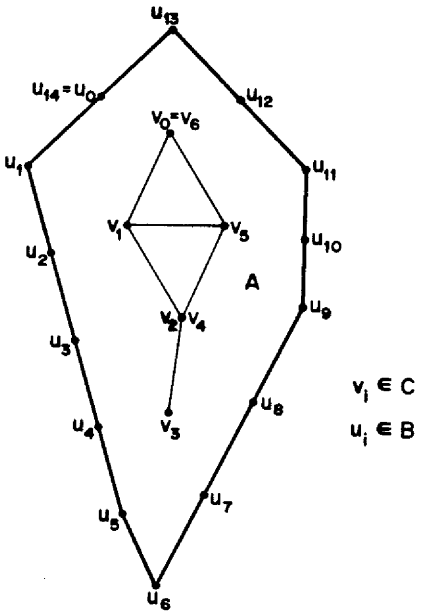


Figure 5.8 Illustration of closed walk C, cycle B, and strip A

We now describe how to generate edges of the type $u_i v_j$ and triangles of the types $\Delta u_i u_{i+1} v_j$ and $\Delta v_{j+1} v_j u_i$ in the strip A by 'merging' B and C and 'zigzagging' counterclockwise around A starting and ending at edge $u_0 v_0$. Note that A is to the left of B and to the right of C so that for a valid triangulation of A (see Section 5.1), CCW triangles $\Delta u_i u_{i+1} v_j$ and $\Delta v_{j+1} v_j u_i$ must be generated. Suppose $u_i v_j$ is the last edge generated and $i < nb$ or $j < nc$ (initially $i = 0$ and $j = 0$). The direction of edge $u_i v_j$ is from u_i to v_j in the last triangle and its direction in the next triangle will be from v_j to u_i. If $i = nb$ then the next edge and triangle generated are $u_{nb} v_{j+1}$ and $\Delta v_{j+1} v_j u_{nb}$. If $j = nc$ then the next edge and triangle generated are $u_{i+1} v_{nc}$ and $\Delta u_i u_{i+1} v_{nc}$. Otherwise either edge $u_i v_{j+1}$ and triangle $\Delta v_{j+1} v_j u_i$ or edge $u_{i+1} v_j$ and triangle $\Delta u_i u_{i+1} v_j$ are generated next based on the following test.

Consider the quadrilateral $Q = u_i u_{i+1} v_{j+1} v_j$. v_j and v_{j+1} are to the left of the directed line from u_i to u_{i+1}, but the positions of u_i and u_{i+1} relative to the directed line, $l(v_j v_{j+1})$, from v_j to v_{j+1} may be one of the following four cases (see Figure 5.9):

(a) u_i and u_{i+1} are both to the right of $l(v_j v_{j+1})$,

(b) u_i (u_{i+1}) is to the left of or on (right of) $l(v_j v_{j+1})$,

(c) u_i (u_{i+1}) is to the right of (left of or on) $l(v_j v_{j+1})$, (5.9)

(d) u_i and u_{i+1} are both to the left of or on $l(v_j v_{j+1})$.

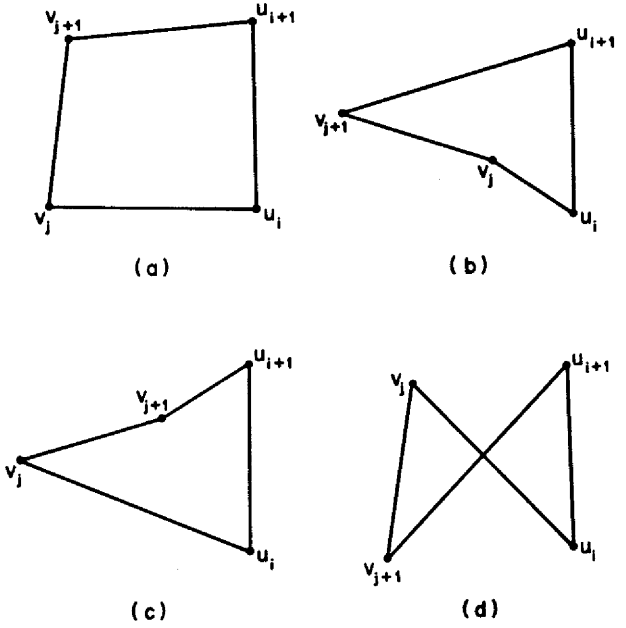


Figure 5.9 Four cases of quadrilateral Q

In case (a), Q is a strictly convex quadrilateral and if the circle through the vertices u_i, u_{i+1}, v_j does not contain vertex v_{j+1} in its interior then edge $u_{i+1}v_j$ and triangle $\Delta u_i u_{i+1} v_j$ are chosen else edge $u_i v_{j+1}$ and triangle $\Delta v_{j+1} v_j u_i$ are chosen. In case (b), Q is a nonconvex quadrilateral and edge $u_{i+1}v_j$ and triangle $\Delta u_i u_{i+1} v_j$ are chosen. In case (c), Q is a nonconvex quadrilateral and edge $u_i v_{j+1}$ and triangle $\Delta v_{j+1} v_j u_i$ are chosen. In case (d), Q is a nonsimple quadrilateral and edge $u_{i+1}v_j$ and triangle $\Delta u_i u_{i+1} v_j$ are chosen; the other triangle $\Delta v_{j+1} v_j u_i$ is oriented in the wrong (clockwise) direction (see (5.3a)). In cases (a), (b), and (c), the chosen edge is locally optimal in Q (see Section 5.1).

Let $T(P)$ be the collection of triangles formed in A and $int(P)$ as described above and in Section 5.3. $T(P)$ is illustrated in Figure 5.10. It is not clear that the triangles in $T(P)$ do not overlap, i.e. that they produce a valid triangulation. If $|u_i u_{i+1}|$ is too large for some i, then $T(P)$ can be an invalid triangulation (e.g. see Figure 5.11). After discussing the case of no interior vertices, we indicate how this merge procedure can be modified to generate a valid triangulation, $VT(P)$, even if the $|u_i u_{i+1}|$ are large.

Now we consider the case of no vertices in the interior of P. Let v_b and v_t be the endpoints of a diameter of P and suppose the coordinate system is rotated so that line segment $v_b v_t$ is parallel to the y-axis with $y(v_t) > y(v_b)$ (as was done for $int(P)$ in Section 5.3). Let $B = [u_0, u_1, \ldots, u_{mb}, \ldots, u_{nb-1}, u_{nb}]$ be the counterclockwise cycle of vertices on ∂P where $u_0 = u_{nb} = v_t$ and $u_{mb} = v_b$. B can be split into two chains $B_L = [u_0, u_1, \ldots, u_{mb}]$ and $B_R = [u_{nb}, u_{nb-1}, \ldots, u_{mb}] = [v_0, v_1, \ldots, v_{mc}]$ on the left and right sides of ∂P, respectively, such that the y-coordinates of the vertices strictly decrease from $y(v_t)$ to $y(v_b)$. Note that mb and mc are both at least one. B_L and B_R can be merged to produce a valid triangulation, $VT(P)$, in P in a way similar to the procedure described above (see Figure 5.12), and $VT(P)$ can be converted into a Delaunay triangulation in a way similar to that described in Section 5.6. The validity of $VT(P)$ is discussed in Section 5.5.

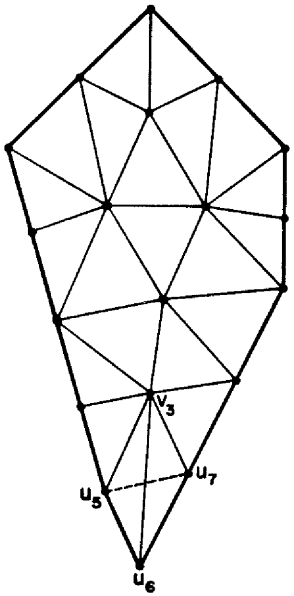


Figure 5.10 Illustration of $T(P)$ and $DT(P)$; u_6v_3 is in $T(P)$; u_5u_7 is in $DT(P)$

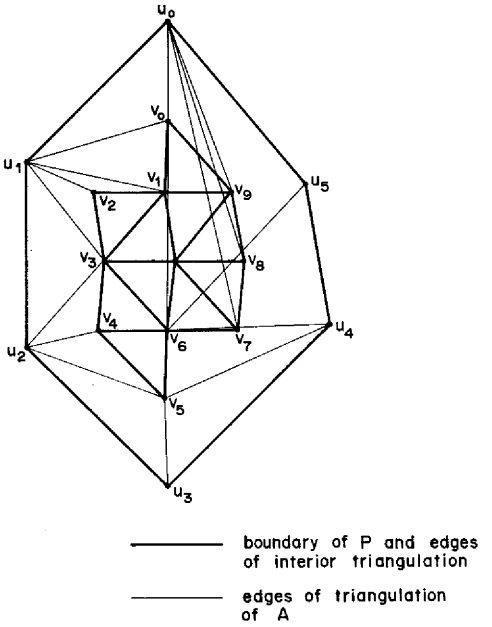


Figure 5.11 Illustration of invalid triangulation $T(P)$; note overlap by triangles $\Delta v_6 v_5 u_4$ and $\Delta v_9 v_8 u_0$

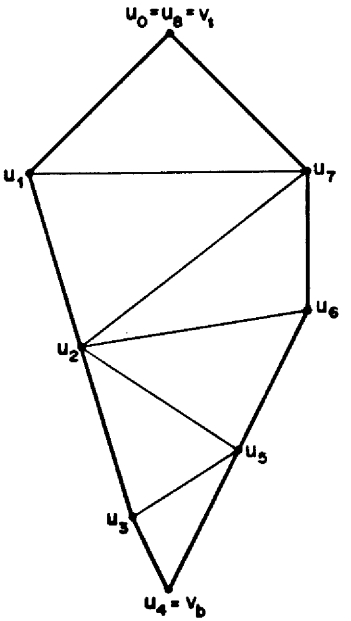


Figure 5.12 Illustration of chains B_L, B_R and $VT(P)$ in the case of no interior vertices

The pseudo-code for the merge of B and C or B_L and B_R to generate triangles in the strip A or polygon P, respectively, is given in procedure MERGE. The list of triangles, TM, produced by this procedure is modified by the procedure of Section 5.6 and then, in the case of at least one vertex in the interior of P, appended to the list of triangles, TI, produced by procedure INTTRIANG.

```
Procedure MERGE(inter, L, nl, M, nm, TM, EM);
# Input: if inter = true then L = B, nl = nb, M = C, nm = nc
#            else L = B_L, nl = mb, M = B_R, nm = mc
# Output: list of triangles TM and list of edges EM;
#           if inter = true then TM and EM each contain nb + nc elements
#              else TM and EM each contain mb + mc - 2 elements
   nll := nl; nmm := nm;
   nt := 0;
   if not inter then
        nt := nt + 1;
        TM(nt) := Δu_0 u_1 v_1; EM(nt) := u_1 v_1;
        if nl + nm = 3 then return;
        nl := nl - 1; nm := nm - 1;
        i := 1; j := 1;
   else i := 0; j := 0;
   while i < nl and j < nm do
        if (u_i and u_{i+1} are both to the left of or on l(v_j v_{j+1})) or
           (v_{j+1} is not in the circle through u_i, u_{i+1}, v_j) then
           nt := nt + 1;
           TM(nt) := Δu_i u_{i+1} v_j; EM(nt) := u_{i+1} v_j;
           i := i + 1;
        else
           nt := nt + 1;
           TM(nt) := Δv_{j+1} v_j u_i; EM(nt) := u_i v_{j+1};
           j := j + 1;
   if i < nl then
        if not inter and j = nm then nl := nl + 1;
        while i < nl do
           nt := nt + 1;
           TM(nt) := Δu_i u_{i+1} v_j; EM(nt) := u_{i+1} v_j;
           i := i + 1;
   else  # j < nm
        if not inter and i = nl then nm := nm + 1;
        while j < nm do
           nt := nt + 1;
           TM(nt) := Δv_{j+1} v_j u_i; EM(nt) := u_i v_{j+1};
           j := j + 1;
   nl := nll; nm := nmm;
   return;
```

As mentioned above, procedure MERGE can fail to produce a valid triangulation of P in the case of at least one interior vertex if the spacing of the u_i on ∂P is too large. In Section 5.5, we examine procedure MERGE to see how it can be modified to always produce a valid triangulation. We will show that an invalid triangulation $T(P)$ is due to an overlapping triangle $\Delta v_j v_{j-1} u_i$ in which v_{j+1} is to the right of $l(v_{j-1}v_j)$ and u_i is to the left of or on $l(v_j v_{j+1})$. Intuitively, we can think of this as resulting from there being too few vertices in B for the number of vertices in C. The modification consists of generating CCW triangle $\Delta v_{j+1} v_j v_{j-1}$ instead of $\Delta v_j v_{j-1} u_i$, replacing subwalk $v_{j-1}v_j v_{j+1}$ of C with edge $v_{j-1}v_{j+1}$, and restarting the merge of B and the new shortened C from edge $u_r v_{j-1}$ where $\Delta v_{j-1} v_{j-2} u_r$ is formed by the merge procedure. The pseudo-code for the modification is given in procedure MMERGE. Let $VT(P)$ be the triangulation produced by procedures INTTRIANG and MMERGE. Procedure MMERGE and the validity of $VT(P)$ will be discussed in Section 5.5. In procedure DELTRIANG of Section 5.7, procedure MMERGE is used to produce a valid triangulation of A in the case of at least one interior vertex, and procedure MERGE is used to produce a valid triangulation of P in the case of no interior vertices.

```
Procedure MMERGE(B,nb,C,nc,TM,EM,ne);
# Input: boundary cycle B, closed walk C, number of edges nb and nc in B and C
# Output: list of triangles TM and edges EM in A, each containing nb + nc elements,
#       and the number of edges ne of type v_j v_{j+m}, m ≥ 2, generated in A.
#       The edges of type v_j v_{j+m} are stored at the end of EM in the reverse order
#       that they are generated. The edges of type u_i v_j are stored at the front of EM
#       in the order that they are generated. The triangles are stored in a similar way.
# The working arrays s, p, r, and n are used as follows:
#       v_{s(j)} and v_{p(j)} are the successor and predecessor vertices of v_j in C.
#       s and p are updated when a vertex is removed from C.
#       u_{r(j)} is the vertex of B in the triangle Δ v_{s(j)} v_j u_{r(j)} and
#       n(j) is the index of this triangle in list TM.
#       r and n are used to determine how far to backtrack the merge of
#       B and C when a vertex is removed from C.
  for j := 0 to nc − 1 do
    s(j) := j + 1; p(j+1) := j; r(j) := 0; n(j) := 0;
  nt := 0; ne := 0;
  i := 0; j := 0;
  while i ≤ nb and j ≤ nc and i + j < nb + nc do
    if (j = nc) or (u_i and u_{i+1} are both to the left of or on l(v_j v_{s(j)}))
      or (v_{s(j)} is not in the circle through u_i, u_{i+1}, v_j) then
      nt := nt + 1;
      TM(nt) := Δ u_i u_{i+1} v_j; EM(nt) := u_{i+1} v_j;
      i := i + 1;
    else
      if (s(j) = nc) or (v_{s(s(j))} is to the left of or on l(v_j v_{s(j)}))
        or (u_i is to the right of or on l(v_{s(j)} v_{s(s(j))})) then
        nt := nt + 1;
        TM(nt) := Δ v_{s(j)} v_j u_i; EM(nt) := u_i v_{s(j)};
        r(j) := i; n(j) := nt;
        j := s(j);
      else
        flag := false;
        repeat  # remove v_{s(j)} from C
          TM(nb + nc − ne) := Δ v_{s(s(j))} v_{s(j)} v_j;
          EM(nb + nc − ne) := v_j v_{s(s(j))};
          ne := ne + 1;
          s(j) := s(s(j)); p(s(j)) := j;
          if j = 0 then i := 0; nt := 0;
            else i := r(p(j)); nt := n(p(j));
          if (j > 0) and (v_{s(j)} is to the right of l(v_{p(j)} v_j))
            and (u_i is to the left of or on l(v_j v_{s(j)})) then
            j := p(j)
          else flag := true;
        until flag;
  return;
```

5.5. Validity of triangulation VT(P)

$VT(P)$ is a valid triangulation of P if the triangles form a 'tiling' of P without overlaps or gaps. We will show that $VT(P)$ is valid for the three cases of (i) no interior vertices, (ii) one interior vertex, and (iii) two or more interior vertices. In case (ii), $VT(P)$ (or $T(P)$) is valid since the triangles $\Delta u_i u_{i+1} v_0$, $0 \leq i \leq nb-1$, are formed and they clearly tile P without overlaps or gaps.

In case (i), each edge added to $VT(P)$ in procedure MERGE subdivides an untriangulated convex subregion of P into two convex subregions - a triangle and a smaller untriangulated subregion. For example, in Figure 5.12, edge $u_1 u_7$ subdivides P into triangle $\Delta u_0 u_1 u_7$ and convex subpolygon $u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_1$. If all the untriangulated convex subregions are nondegenerate (i.e. not a line segment) then the triangles of $VT(P)$ clearly tile P without overlaps or gaps so $VT(P)$ is valid. An untriangulated convex subregion can be degenerate if one side of ∂P contains only two vertices and the other side contains three or more collinear vertices at the bottom. In this case degenerate triangles (i.e. with three collinear vertices) are formed. For example, in Figure 5.13a, triangles $\Delta u_0 u_1 u_3$ and $\Delta u_2 u_3 u_1$ are formed by procedure MERGE and the latter triangle is degenerate. If the collinear vertices at the bottom are perturbed slightly so that no three vertices are collinear and the polygon remains convex then all the untriangulated subregions are nondegenerate and the triangulation is valid (e.g. see Figure 5.13b). Therefore we still consider the triangulations with the degenerate triangles to be 'valid'. The procedure described in the next section will convert these triangulations into Delaunay triangulations.

In the rest of this section we will consider the case of two or more interior vertices. We will show that $VT(P)$ is valid by showing how $T(P)$ (the triangulation produced by procedures INTTRIANG and MERGE) can sometimes be invalid and how a modification can be made to procedure MERGE to always produce a valid triangulation. In Section 5.1, we mentioned that a triangulation of a region is valid if it satisfies the four conditions of (5.3).

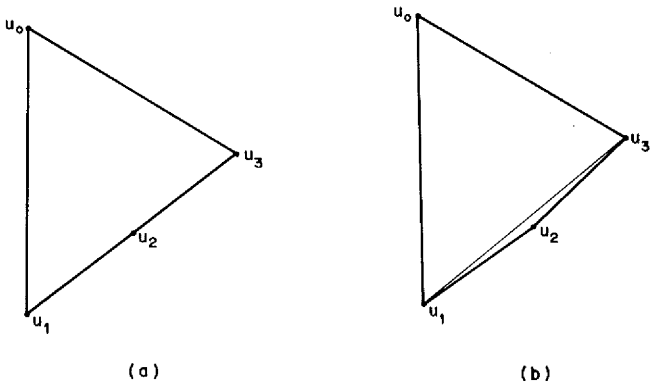


Figure 5.13 (a) Example where degenerate triangle $\Delta u_2 u_3 u_1$ is formed by procedure MERGE (b) Perturbation of chain $u_1 u_2 u_3$ to obtain valid triangulation

Lemma 5.4 :

(a) $T(P)$ satisfies conditions (b), (c), and (d) of (5.3).

(b) If $\Delta v_{j+1} v_j u_i$ is produced by procedure MERGE and $i < nb$, then $\Delta v_{j+1} v_j u_i$ is CCW.

(c) If all triangles of type $\Delta v_{j+1} v_j u_i$ produced by procedure MERGE are CCW, then $T(P)$ is a valid triangulation of P.

(d) $T(P)$ is not a valid triangulation of P if and only if there is a CW triangle of type $\Delta v_{j+1} v_j u_{nb}$ produced by procedure MERGE.

Proof : The boundary edges of P are the edges $u_i u_{i+1}$ of B. Conditions (c) and (d) of (5.3) are clearly satisfied by $T(P)$ since the triangles formed in A by procedure MERGE are of the types $\Delta u_i u_{i+1} v_j$ and $\Delta v_{j+1} v_j u_i$. We now show that condition (5.3b) is satisfied by all edges of $T(P)$. The edges of E_I (see (5.8)) which are not edges of C clearly satisfy condition (5.3b) (see

Figure 5.6). An edge $e = v_j v_{j+1}$ of C occurs either once or twice in C. In the latter case, $v_j = v_{k+1}$ and $v_{j+1} = v_k$ for some $k \neq j$ and e occurs in opposite directions in the triangles $\Delta v_{j+1} v_j u_i$ and $\Delta v_{k+1} v_k u_l$ formed in A. In the former case, e occurs in opposite directions in the triangles $\Delta v_{j+1} v_j u_i$ in A and $\Delta v_j v_{j+1} w$ in $int(P)$. A boundary edge $u_i u_{i+1}$ of B occurs only once in the triangle $\Delta u_i u_{i+1} v_j$ formed in A. The other edges of $T(P)$ are of the type $u_i v_j$, $u_i \in B$ and $v_j \in C$, formed in A. One of the triangles with edge $u_i v_j$ is either $\Delta u_{i-1} u_i v_j$ or $\Delta v_j v_{j-1} u_i$; the other triangle with edge $u_i v_j$ is either $\Delta u_i u_{i+1} v_j$ or $\Delta v_{j+1} v_j u_i$, where the indices of u_i and v_j are taken modulo nb and nc respectively. In all four possibilities, edge $u_i v_j$ occurs in opposite directions in the two triangles. Therefore condition (5.3b) is satisfied by all edges of $T(P)$.

Now we determine when condition (5.3a) is satisfied or not satisfied. The triangles generated by procedure INTTRIANG are all CCW. The triangles generated by procedure MERGE are of the types $\Delta u_i u_{i+1} v_j$ or $\Delta v_{j+1} v_j u_i$. $\Delta u_i u_{i+1} v_j$ is a CCW triangle since v_j is to the left of directed edge $u_i u_{i+1}$. $\Delta v_{j+1} v_j u_i$ is a CCW triangle if it is formed from case (a) or (c) of (5.9) (see Figure 5.9). It cannot be formed from case (b) or (d) of (5.9). So $\Delta v_{j+1} v_j u_i$ may be a CW triangle only when there are no more quadrilaterals $u_i u_{i+1} v_{j+1} v_j$ and $i = nb$. By condition (5.3a), $T(P)$ is a valid triangulation if the triangles of type $\Delta v_{j+1} v_j u_{nb}$ are all CCW.

If $T(P)$ is not a valid triangulation then condition (5.3a) is not satisfied and there is a CW triangle of the type $\Delta v_{j+1} v_j u_{nb}$. To show that the converse is also true, suppose $\Delta v_{j+1} v_j u_{nb}$ is a CW triangle produced by procedure MERGE. Edge $v_j v_{j+1}$ occurs either once or twice in C. In the former case, CCW triangle $\Delta v_j v_{j+1} w$ is formed by procedure INTTRIANG, and $\Delta v_{j+1} v_j u_{nb}$ and $\Delta v_j v_{j+1} w$ overlap. In the latter case, $v_j = v_{k+1}$ and $v_{j+1} = v_k$ for some $k \neq j$ and $\Delta v_{k+1} v_k u_l$ is formed by procedure MERGE; either $\Delta v_{k+1} v_k u_l$ is CCW if $l < nb$ or $\Delta v_{k+1} v_k u_l = \Delta v_{j+1} v_j u_{nb}$ if $l = nb$; in both subcases $\Delta v_{j+1} v_j u_{nb}$ and $\Delta v_{k+1} v_k u_l$ overlap. Therefore $T(P)$ is not a valid triangulation. □

We now examine the triangles produced in A by procedure MERGE to see how CW trian-

gles $\Delta v_{j+1}v_ju_{nb}$ can be formed and how the procedure can be modified to produce a valid triangulation in this case. Since the triangles of type $\Delta u_iu_{i+1}v_j$ are always CCW, we concentrate on the triangles of type $\Delta v_{j+1}v_ju_i$. Let $\Delta v_{j+1}v_ju_{r(j)}$, $0 \leq j \leq nc-1$, be the triangles of type $\Delta v_{j+1}v_ju_i$ produced by procedure MERGE, where $0 \leq r(0) \leq r(1) \leq \cdots \leq r(nc-1) \leq nb$.

We first make some definitions which will be used in the following lemmas. We define v_j to be a reflex vertex of C if v_{j+1} is to the right of $l(v_{j-1}v_j)$ and define v_j to be a convex vertex of C otherwise. Note that v_0 is a convex vertex of C since it is the leftmost interior mesh vertex on the top line $y = y_0$. Let v_{mid} be the rightmost interior mesh vertex on the bottom line $y = y_n$. v_{mid} is also a convex vertex of C. Let C_L be the subwalk of C from v_0 to v_{mid} and C_R be the subwalk of C from v_{mid} to v_{nc}. Let $[w,z]$ denote the 'interval' of ∂P going counterclockwise from point w to point z inclusive. A parenthesis will be used in place of the bracket if the endpoint w or z is not included, e.g. (w,z), $(w,z]$, $[w,z)$. Let w_j and z_j be the intersections of line $l(v_jv_{j+1})$ with ∂P where w_j is the intersection closer to v_j and z_j is the intersection closer to v_{j+1}. (w_j,z_j) is the interval of ∂P to the right of $l(v_jv_{j+1})$. Define $I_j = (w_j,z_j)$ if u_0 is not in (w_j,z_j), $I_j = [u_0,z_j)$ if u_0 is in (w_j,z_j) and $v_jv_{j+1} \in C_L$, and $I_j = (w_j,u_{nb}]$ if $u_0 = u_{nb}$ is in (w_j,z_j) and $v_jv_{j+1} \in C_R$. Recall that $u_{r(j)}$ is a vertex of the triangle $\Delta v_{j+1}v_ju_{r(j)}$ produced by procedure MERGE.

Lemma 5.5 : If $u_{r(j)}$ is not in I_j for some j, then $T(P)$ is not a valid triangulation.

Proof : If $u_{r(j)}$ is not in (w_j,z_j) then $\Delta v_{j+1}v_ju_{r(j)}$ is CW so $r(j) = nb$ and $T(P)$ is not valid by Lemma 5.4. Suppose u_0 is in (w_j,z_j) so that $I_j \neq (w_j,z_j)$ and suppose $u_{r(j)}$ is in (w_j,z_j) but not in I_j. First suppose $v_jv_{j+1} \in C_L$. Then $u_{r(j)}$ is in (w_j,u_{nb}) and CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$ must intersect the walk $u_0v_0v_1 \cdots v_j$ (see Figure 5.14). Now suppose $v_jv_{j+1} \in C_R$. Then $u_{r(j)}$ is in (u_0,z_j) and CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$ must intersect the walk $v_{j+1}v_{j+2} \cdots v_{nc}u_{nb}$. In both cases, $\Delta v_{j+1}v_ju_{r(j)}$ overlaps other triangles of $T(P)$ so $T(P)$ is not valid. □

Lemma 5.6 : If v_j is a reflex vertex of C and $\Delta v_jv_{j-1}u_{r(j-1)}$ is CCW with $u_{r(j-1)}$ to the left of or on $l(v_jv_{j+1})$, then $\Delta v_jv_{j-1}u_{r(j-1)}$ is an overlapping triangle and $T(P)$ is not a valid tri-

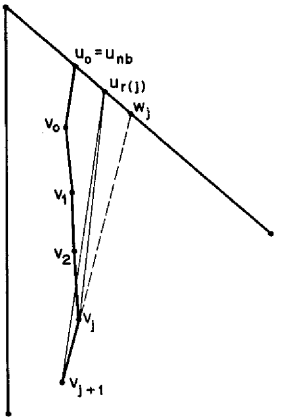


Figure 5.14 $\Delta v_{j+1} v_j u_{r(j)}$ intersects walk $u_0 v_0 v_1 \cdots v_j$

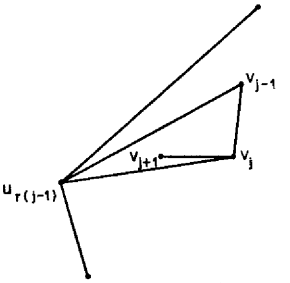


Figure 5.15 $\Delta v_j v_{j-1} u_{r(j-1)}$ intersects edge $v_j v_{j+1}$

angulation.

Proof : If $u_{r(j-1)}$ is to the left of or on $l(v_jv_{j+1})$ then part of edge v_jv_{j+1} lies in the interior or on the boundary of $\Delta v_jv_{j-1}u_{r(j-1)}$ (see Figure 5.15). Therefore $\Delta v_jv_{j-1}u_{r(j-1)}$ and $\Delta v_{j+1}v_ju_{r(j)}$ overlap and $T(P)$ is not a valid triangulation. □

Lemma 5.7 :

(a) If v_j is a reflex vertex of C and $\Delta v_jv_{j-1}u_{r(j-1)}$ is CCW with $u_{r(j-1)}$ in I_{j-1} and to the right of $l(v_jv_{j+1})$ then $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j.

(b) If v_j is a convex vertex of C, $0<j<nc$, and $\Delta v_jv_{j-1}u_{r(j-1)}$ is CCW with $u_{r(j-1)}$ in I_{j-1} then $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j.

(c) If $j=0$ then $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j.

Proof : Define $u_{t(j)}$ to be the vertex of largest index in I_j.

(a) Let u_l, $r(j-1)\le l\le nb$, be the vertex of largest index to the right of $l(v_jv_{j+1})$ such that u_i, $i=r(j-1),\ldots,l$, are to the right of $l(v_jv_{j+1})$. If $l>r(j-1)$ then quadrilaterals $u_iu_{i+1}v_{j+1}v_j$, $i=r(j-1),\ldots,l-1$, are case (a) of (5.9) and CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$ may be formed from one of these quadrilaterals. If not, then CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$, $r(j)=l$, is formed because either $l=nb$ or quadrilateral $u_lu_{l+1}v_{j+1}v_j$ is case (c) of (5.9). v_j is a reflex vertex and $u_{r(j-1)}$ is in I_{j-1} and to the right of $l(v_jv_{j+1})$ imply that $u_{r(j-1)}$ is in I_j and $r(j-1)\le r(j)\le t(j)=l$. Therefore $u_{r(j)}$ is in I_j.

(b) v_j is a convex vertex and u_0 is chosen so that $y(u_0)>y(v_0)$ imply that $t(j)\ge t(j-1)$. $u_{r(j-1)}$ is in I_{j-1} implies that $r(j-1)\le t(j-1)$. Let u_k, $r(j-1)\le k\le t(j)$, be the vertex of smallest index to the right of $l(v_jv_{j+1})$ such that u_i, $i=k,\ldots,t(j)$, are to the right of $l(v_jv_{j+1})$. If $k>r(j-1)$ then $\Delta u_iu_{i+1}v_j$, $i=r(j-1),\ldots,k-1$ are formed from case (b) or (d) of (5.9). If $k<t(j)$ then quadrilaterals $u_iu_{i+1}v_{j+1}v_j$, $i=k,\ldots,t(j)-1$, are case (a) of (5.9) and CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$ may be formed from one of these quadrilaterals. If not, then CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$, $r(j)=t(j)$, is formed because either $t(j)=nb$ or quadrilateral

$u_{t(j)}u_{t(j)+1}v_{j+1}v_j$ is case (c) of (5.9). $u_{r(j-1)}$ is in I_{j-1} and $k \leq r(j) \leq t(j)$ imply that $u_{r(j)}$ is in I_j.

(c) Let u_k, $0 \leq k \leq t(j)$, be the vertex of smallest index to the right of $l(v_jv_{j+1})$ such that u_i, $i = k, \ldots, t(j)$, are to the right of $l(v_jv_{j+1})$. The rest of the proof is the same as (b) with $r(j-1)$ replaced by 0 (also omit '$u_{r(j-1)}$ is in I_{j-1}' from the last sentence). □

Lemma 5.8 : If all vertices v_j of C are convex then $T(P)$ is a valid triangulation.

Proof : By parts (c) and (b) of Lemma 5.7 and induction, $\Delta v_{j+1}v_ju_{r(j)}$, $j = 0, 1, \ldots, nc-1$, are CCW triangles. By Lemma 5.4, $T(P)$ is a valid triangulation. □

Lemma 5.9 : If for all reflex vertices v_j of C, $u_{r(j-1)}$ is to the right of $l(v_jv_{j+1})$, then $T(P)$ is a valid triangulation.

Proof : We will show by induction that for all j, $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j. By Lemma 5.7(c), $\Delta v_1v_0u_{r(0)}$ is CCW with $u_{r(0)}$ in I_0. Suppose $0 < j < nc$ and $\Delta v_jv_{j-1}u_{r(j-1)}$ is CCW with $u_{r(j-1)}$ in I_{j-1}. If v_j is a convex vertex then $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j by Lemma 5.7(b). If v_j is a reflex vertex then it is given that $u_{r(j-1)}$ is to the right of $l(v_jv_{j+1})$ so $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j by Lemma 5.7(a). Therefore $\Delta v_{j+1}v_ju_{r(j)}$, $j = 0, 1, \ldots, nc-1$, are CCW triangles. By Lemma 5.4, $T(P)$ is a valid triangulation. □

Conjecture 5.1 : If $|u_iu_{i+1}|$ is sufficiently small for all i (e.g. as produced by the spacing of (5.1), (5.2)), then $T(P)$ is a valid triangulation.

Rationale : If $|u_iu_{i+1}|$ is sufficiently small for all i then for all reflex vertices v_j of C there exists a vertex u_k in I_{j-1} such that u_k is to the right of $l(v_jv_{j+1})$; in order for CCW triangle $\Delta v_jv_{j-1}u_k$ to be generated by procedure MERGE, either $u_k = u_{nb}$ or u_{k+1} must not be in the interior of the circle through the vertices v_{j-1}, v_j, u_k. It appears from the way that C is constructed from $int(P)$ that such a vertex u_k exists for all reflex vertices v_j and the hypothesis of Lemma 5.9 is satisfied, so $T(P)$ is a valid triangulation. □

If $|u_iu_{i+1}|$ is too large for some i, then procedure MERGE can produce an invalid triangulation $T(P)$ (see Figure 5.11). Lemmas 5.6, 5.7, 5.8, and 5.9 suggest how procedure MERGE can

be modified to produce a valid triangulation in this case. By Lemma 5.9, if $T(P)$ is not valid then there exists a reflex vertex v_j such that $u_{r(j-1)}$ is to the left of or on $l(v_jv_{j+1})$. Suppose v_j is the reflex vertex of smallest index such that $u_{r(j-1)}$ is to the left of or on $l(v_jv_{j+1})$ (e.g. in Figure 5.11, $v_j = v_6$). Then $\Delta v_{k+1}v_ku_{r(k)}$, $k = 0, \ldots, j-1$, are CCW with $u_{r(k)}$ in I_k by Lemma 5.7 but $\Delta v_jv_{j-1}u_{r(j-1)}$ is an overlapping triangle by Lemma 5.6. Therefore instead of generating $\Delta v_jv_{j-1}u_{r(j-1)}$; generate CCW triangle $\Delta v_{j+1}v_jv_{j-1}$ which does not overlap any triangles in $C \cup$ (interior of C), replace subwalk $v_{j-1}v_jv_{j+1}$ of C with edge $v_{j-1}v_{j+1}$, and rerun procedure MERGE with B and the new shortened C which still lies in $int(P)$. If this process is repeated enough times when overlapping triangles $\Delta v_jv_{j-1}u_{r(j-1)}$ are detected, then the resulting triangulation is valid because either the hypothesis of Lemma 5.9 is satisfied or $C \cup$ (interior of C) eventually becomes a convex set so the hypothesis of Lemma 5.8 is satisfied. Procedure MMERGE of Section 5.4 contains this modification to procedure MERGE in which the merge is restarted from edge $u_{r(j-2)}v_{j-1}$ instead of edge u_0v_0 (if $j \geq 2$) when it is determined that a CCW triangle $\Delta v_{j+1}v_jv_{j-1}$ must be added to the triangulation, since the triangles in A between u_0v_0 and $u_{r(j-2)}v_{j-1}$ would remain the same. We have shown in the above discussion that

Theorem 5.3 : The triangulation, $VT(P)$, produced by procedures INTTRIANG and MMERGE is a valid triangulation.

5.6. Converting VT(P) into a Delaunay triangulation

We describe a procedure for converting $VT(P)$ into a Delaunay triangulation, $DT(P)$. We discuss only the case of at least one interior vertex, since the case of no interior vertices is similar. From Lemma 5.2, the edges of the interior triangulation and C (as defined in Section 5.3) are locally optimal in any triangulation of the mesh vertices which contains these edges. From Theorem 5.1, $VT(P)$ can be converted into a Delaunay triangulation by applying LOP to the internal edges in the triangulation of A until they are all locally optimal.

Procedure MMERGE produces $nt = nb + nc$ triangles, $TM(1), TM(2), \ldots, TM(nt)$, and nt edges, $EM(1), EM(2), \ldots, EM(nt)$, in A, where $ne \geq 0$ edges are of the type v_jv_{j+m}, $m \geq 2$,

and the triangles and edges are ordered as follows. Edges $EM(k)$, $k = 1, \ldots, nt-ne$, are of the type $u_i v_j$ and are in the order that they are generated. Edges $EM(k)$, $k = nt-ne+1, \ldots, nt$, are of the type $v_j v_{j+m}$, $m \geq 2$, and are in the reverse order that they are generated (these edges are added to ensure the validity of the triangulation of A). For $k = 1, \ldots, nt-ne-1$, $EM(k)$ is the common edge of $TM(k)$ and $TM(k+1)$. $EM(nt-ne) = u_0 v_0 = u_{nb} v_{nc}$ is the common edge of $TM(nt-ne)$ and $TM(1)$. For $k = nt-ne+1, \ldots, nt$, $EM(k)$ is the common edge of $TM(k)$ and $TM(l)$ for some $l < k$. For $1 \leq k \leq nt$, let A_k be the region formed by the adjacent triangles $TM(1), \ldots, TM(k)$. We define an *internal* edge in a triangulation of A_k to be an edge of the type $u_i v_j$, $u_i u_{i+m}$, or $v_j v_{j+m}$, $m \geq 2$, which is a common edge of two triangles in A_k.

The following step is performed for $k = 1, 2, \ldots, nt-1$ to obtain locally optimal internal edges in the triangulation of A. Suppose the internal edges in the triangulation of A_k are locally optimal (this is trivially true for $k = 1$). Then the internal edges in the triangulation of A_{k+1} are made locally optimal as follows (with the exception mentioned in the next paragraph when $k = nt-ne-1$). The triangulation of A_{k+1} consists of the triangles in A_k plus the triangle TM(k+1). Let $e = EM(k)$ if $k \leq nt-ne-1$ and $e = EM(k+1)$ if $k \geq nt-ne$. Then e is the only internal edge in the triangulation of A_{k+1} which may not be locally optimal. Therefore apply LOP to e. If it is swapped for the other diagonal edge of the quadrilateral, Q, formed by the two triangles having e as a common edge, then the edges of Q which are internal edges of A_{k+1} may no longer be locally optimal so they are placed in a stack of edges for which LOP must be applied. If the stack is not empty, then the top edge is popped from the stack and LOP is applied to this edge. This may cause another swap and more internal edges of A_{k+1} to be pushed onto the stack. This process of applying LOP is repeated until the stack is empty which implies that the internal edges in the triangulation of A_{k+1} are locally optimal. Lawson (1977) shows that this process must always terminate. (There are only a finite number of possible triangulations of A_{k+1}; a linear ordering of these triangulations can be defined using the sorted vector of the smallest angle in each triangle; and each swap produced by LOP causes a strict advance through this linear ordering of triangulations.)

The above step must be performed twice when $k = nt - ne - 1$ (i.e. the triangulation of A 'closes') since $EM(k)$ and $EM(k+1) = u_0v_0$ may not be locally optimal in the triangulation of A_{k+1}. First the above step is performed for $e = EM(k)$ and then it is performed for $e = EM(k+1)$. We have chosen u_0 so that u_0v_0 is likely to be a Delaunay edge (see Section 5.4) so u_0v_0 is not likely to be swapped when LOP is applied to it. After the above step is performed for $k = nt - 1$, the internal edges in the triangulation of $A = A_{nt}$ and P are locally optimal and a Delaunay triangulation, $DT(P)$, is obtained by Lemma 5.2 and Theorem 5.1. Figure 5.10 illustrates a Delaunay triangulation $DT(P)$ obtained from $VT(P)$ by making one diagonal edge swap: u_5u_7 replaces u_6v_3 in the quadrilateral $u_5u_6u_7v_3$.

The pseudo-code for converting the triangles of $VT(P)$ in A into Delaunay triangles is given in procedure CONVERT. This procedure can also be used in the case of no interior vertices.

```
Procedure CONVERT(inter, TM, EM, nt, ne);
# Input: if inter = true then TM, EM, and ne are output from procedure MMERGE
#              and nt = nb + nc
#        else TM and EM are output from procedure MERGE, nt = mb + mc - 2,
#              and ne = 0
# Output: updated list of triangles TM such that all triangles are Delaunay
#         and updated list of edges EM such that all edges are Delaunay
   for k := 1 to nt do
      if not inter and k = nt then return;
      if k ≤ nt - ne - 1 then kk := k + 1 else kk := k;
      top := 1;
      stack(top) := k;
      while top ≥ 1 do
            l := stack(top);
            top := top - 1;
            Search TM(kk), TM(kk-1), ..., TM(1) sequentially until the two
                 triangles TM(m) and TM(n) containing edge EM(l) are found;
            Q := quadrilateral w1w2w3w4 where TM(m) = Δw1w2w3,
                 TM(n) = Δw1w3w4, and EM(l) = w1w3;
            if w4 is in the circle through w1, w2, w3 then
                 TM(m) := Δw1w2w4;
                 TM(n) := Δw2w3w4;
                 EM(l) := w2w4;
                 w5 := w1;
                 for q := 1 to 4 do
                    if 1 ≤ index(wqwq+1) ≤ k then
                          top := top + 1;
                          stack(top) := index(wqwq+1);
   return;
```

In this procedure, the function $index(ww')$ is defined to be 0 if ww' is an edge of the type

u_iu_{i+1} or v_jv_{j+1} otherwise it is the index of edge ww' in the list EM. The function is used to determine whether an edge is an internal edge in the triangulation of A_{k+1}. For each triangle $\Delta w_1w_2w_3$ in the list TM, we store the values $index(w_1w_2)$, $index(w_2w_3)$, and $index(w_3w_1)$ in addition to the vertices of the triangle, so that it is easy to determine $index(w_qw_{q+1})$ in the innermost for loop. When LOP is applied to an internal edge e, the list TM must be searched for the two triangles containing e. The reason for the sequential search starting backwards from $TM(kk)$ will be discussed below.

We now estimate the number of edge swaps required for converting $VT(P)$ into $DT(P)$ since the efficiency of the procedure depends on the number of swaps. In $DT(P)$, strip A may contain edges of the types u_iv_j, u_iu_{i+m}, and v_jv_{j+m}, $m \geq 2$. By Conjecture 5.1, $VT(P)$ is expected to contain only edges of type u_iv_j in A so edge swaps are needed to obtain Delaunay edges of types u_iu_{i+m} and v_jv_{j+m}. If u_iv_j is a Delaunay edge then it is likely to be an edge of $VT(P)$ since u_0v_0 is likely to be a Delaunay edge and in cases (a), (b), and (c) of (5.9) the next edge $u_{i+1}v_j$ or u_iv_{j+1} is chosen to be locally optimal in the quadrilateral $u_iu_{i+1}v_{j+1}v_j$. The following theorem indicates when no swaps are required.

Theorem 5.4 : If u_0v_0 is a Delaunay edge and there are no Delaunay edges of types u_iu_{i+2} or v_jv_{j+2}, then $VT(P) = T(P)$ is a Delaunay triangulation.

Proof : First we show that under the hypothesis of the theorem there are no Delaunay edges of types u_iu_{i+m} or v_jv_{j+m}, $m \geq 3$. Suppose u_lu_{l+m}, $m \geq 3$, is a Delaunay edge. Then $u_lu_{l+1} \cdots u_{l+m}u_l$ must form a simple subpolygon of P which contains no vertices in its interior. In a Delaunay triangulation of the mesh vertices, this subpolygon contains $m-1$ triangles involving only the vertices $u_l, u_{l+1}, \ldots, u_{l+m}$. Therefore there must be a Delaunay edge u_iu_{i+2} where $l \leq i \leq l+m-2$. This contradicts the hypothesis therefore there are no Delaunay edges of type u_iu_{i+m}, $m \geq 3$. Similarly there are no Delaunay edges of type v_jv_{j+m}, $m \geq 3$. Therefore the Delaunay edges in A are all of the type u_iv_j.

We show by induction that the edges produced by procedure MERGE (which are the same

as those produced by procedure MMERGE) are Delaunay edges. Suppose $u_i v_j$ is a Delaunay edge (initially $u_0 v_0$ is a Delaunay edge). Then $\Delta u_i u_{i+1} v_j$ or $\Delta v_{j+1} v_j u_i$ must be a Delaunay triangle and $u_{i+1} v_j$ or $u_i v_{j+1}$ must be a Delaunay edge, respectively. If $i = nb$, then $\Delta v_{j+1} v_j u_i$ is the only possible triangle therefore $u_i v_{j+1}$ is a Delaunay edge. Similarly if $j = nc$, then $u_{i+1} v_j$ is a Delaunay edge. Suppose $i < nb$ and $j < nc$. Consider quadrilateral $Q = u_i u_{i+1} v_{j+1} v_j$ in (5.9). In case (a), the circle test chooses the next Delaunay edge. In cases (b), (c), and (d) the chosen triangle is a Delaunay triangle since the other triangle is not a 'valid' triangle - it is a CW or overlapping triangle (see Figure 5.9). Therefore procedure MERGE generates the next Delaunay edge, either $u_{i+1} v_j$ or $u_i v_{j+1}$. By induction, procedure MERGE produces Delaunay edges in A. The edges generated by procedure INTTRIANG are Delaunay edges by Lemma 5.2. Therefore $VT(P) = T(P)$ is a Delaunay triangulation. □

In general, $DT(P)$ contains edges of types $u_i u_{i+m}$ and $v_j v_{j+m}$, $m \geq 2$. An edge $v_j v_{j+m}$ may be a Delaunay edge if it lies entirely in A and angle$(v_{j+m} v_{j+l} v_j)$ for some l, $0 < l < m$, is smaller than approximately 120°, e.g. the edge joining $v_j = (x_{i+1,0}, y_{i+1})$ and $v_{j+2} = (x_{i+2,0}, y_{i+2})$ in Figure 5.6. With the spacing of the u_i on ∂P restricted by (5.1) and (5.2), it is unlikely that $m > 2$ and the number of Delaunay edges of type $v_j v_{j+2}$ and the number of swaps to obtain these edges are small relative to nc. An edge $u_i u_{i+m}$ may be a Delaunay edge if it is near a vertex of P or $int(P)$ with an interior angle smaller than approximately 90°, e.g. $u_5 u_7$ in Figure 5.10. If P does not contain 'small' interior angles (e.g. $\leq 20°$) then the number of Delaunay edges of type $u_i u_{i+m}$, $m \geq 2$, and the number of swaps to obtain these edges are expected to be small relative to nb. From Lemma 4.2(a), at most two interior angles of a convex polygon can be less than 60°. For polygons P in which the smallest interior angle at a vertex decreases, the maximal value m of a Delaunay edge $u_i u_{i+m}$ near the vertex increases, so the number of swaps required to obtain the Delaunay edges in the subpolygon $u_i u_{i+1} \cdots u_{i+m} u_i$ increases.

Conjecture 5.2 : For a fixed convex polygon P (which may contain at most two small interior angles), as the triangle size parameter h decreases and the number of triangles in A, $nb + nc$,

increases, the number of edge swaps for converting $VT(P)$ into $DT(P)$ is at most $O(nb + nc)$.

Rationale : As nc increases, the number of Delaunay edges of type $v_j v_{j+2}$ increases but is small relative to nc. As nb increases, the number of Delaunay edges of type $u_i u_{i+m}$, $m \geq 2$, does not increase since the spacing of vertices u_i on ∂P relative to h and the distance of each vertex of P to the nearest interior mesh vertex relative to h remain approximately the same. Therefore the number of edge swaps required to convert $VT(P)$ into $DT(P)$ increases at a slower rate than $nb + nc$ and is at most $O(nb + nc)$. □

When LOP is applied to edge $EM(k)$ in A_{k+1} for $k \leq nt - ne - 1$, edge swaps, if any, occur for triangles $TM(n)$ near $TM(k+1)$ with indices $n \leq k+1$. Therefore we store the triangles of A in a linear list and when LOP is applied to an edge e in the triangulation of A_{k+1}, a search is made sequentially backwards in this list starting from $TM(k+1)$ until the two triangles containing edge e are found. The number of triangles searched is expected to be a small constant if P does not contain small interior angles. When LOP is applied to edge $EM(nt - ne) = u_0 v_0$ in A_{nt-ne}, one of the triangles containing edge $u_0 v_0$ is near the end of list TM and the other is near the front of TM. By using this information, few triangles are searched when LOP is applied to $u_0 v_0$. If $VT(P)$ contains an edge of type $v_j v_{j+m}$, $m \geq 2$, then $O(nt)$ triangles are expected to be searched when LOP is applied to this edge since the two triangles containing this edge may not be close together in TM. By Conjecture 5.1, ne is expected to be zero. Therefore if P does not contain small interior angles, the number of triangles searched in the applications of LOP is expected to be $O(nb + nc)$. The number of applications of LOP can be reduced by noting that if consecutive triangles $TM(k) = \Delta u_{i-1} u_i v_j$ and $TM(k+1) = \Delta v_{j+1} v_j u_i$ (or $TM(k) = \Delta v_j v_{j-1} u_i$ and $TM(k+1) = \Delta u_i u_{i+1} v_j$) are formed by procedure MMERGE, then $EM(k) = u_i v_j$ is locally optimal in A_{k+1} if $\Delta u_{i-1} u_i v_j$ (or $\Delta v_j v_{j-1} u_i$) has not been swapped in A_k.

5.7. Summary and time complexity

The pseudo-code for our triangulation algorithm is summarized in procedure DELTRIANG.

```
Procedure DELTRIANG(P, m, h, T, n_t);
# Input: convex polygon P with m vertices and triangle size parameter h
#        (the triangle sizes, h(e), for the edges of ∂P are computed by (5.1)
#         using h and the triangle size parameters in adjacent polygons)
# Output: list of triangles T and number of triangles n_t
  SHRINK(P, m, h/√2, int(P), ni);
  if ni > 0 then inter := true else inter := false;
  if inter then
     Rotate coordinate system so that diameter of int(P) is parallel to y-axis;
     INTTRIANG(int(P), h, TI, nt, C, nc);
  else
     Rotate coordinate system so that diameter of P is parallel to y-axis;
     TI := [ ]; C := [ ]; nt := 0; nc := −2;
  for each edge e of ∂P do
     Generate mesh vertices on e at an equal spacing of h(e);
  if inter then
     Determine B = [u_0, u_1, ..., u_nb];
     MMERGE(B, nb, C, nc, TM, EM, ne);
  else
     Determine B_L = [u_0, u_1, ..., u_mb], B_R = [u_nb, u_nb−1, ..., u_mb];
     MERGE(inter, B_L, mb, B_R, nb − mb, TM, EM);
     ne := 0;
  CONVERT(inter, TM, EM, nb + nc, ne);
  T := append(TI, TM);
  n_t := nt + nb + nc;
  return;
```

We now derive the time complexity for procedure DELTRIANG to construct the Delaunay triangulation, $DT(P)$, in P. Let n_t and n_v be the number of triangles and mesh vertices, respectively, generated in P by procedure DELTRIANG. Let nb be the number of mesh vertices on ∂P and m be the number of vertices of P. n_t, n_v, nb, and m are related by the formulas (Lawson (1977))

$$n_t = 2n_v - nb - 2 \text{ and } n_t \geq nb - 2 \geq m - 2. \tag{5.10}$$

By Theorem 5.2, $int(P)$ is determined from P in $O(m)$ time by procedure SHRINK. The diameter of $int(P)$ or P is found in $O(m)$ time (Shamos (1975)). In procedure INTTRIANG, the generation of mesh vertices in $int(P)$ requires $O(n_v)$ time and the generation of triangles and closed walk C requires $O(n_t)$ time. The generation of mesh vertices on ∂P requires $O(nb)$ time. The generation of triangles in A by procedure MMERGE requires $O(n_t)$ time if ne, the number of edges of type $v_j v_{j+k}$, $k \geq 2$, is zero or bounded by a small constant (ne is expected to be zero by Conjecture 5.1). In the case of no interior vertices, the generation of triangles in P by procedure

MERGE requires $O(n_t)$ time. Therefore the time complexity for constructing $VT(P)$ (procedure DELTRIANG minus procedure CONVERT) is $O(n_t) = O(n_v)$ using (5.10). From the discussion at the end of Section 5.6, the number of edge swaps and the number of triangle searches in the applications of LOP in procedure CONVERT are expected to be $O(n_t)$ if P contains no small interior angles. Therefore the time complexity for constructing $DT(P)$ by procedure DELTRIANG is expected to be $O(n_t)$ if P contains no small interior angles. (A goal of the first two stages of our triangulation method is to avoid creating convex polygons with small interior angles in the decomposition of the region R.) By Conjecture 5.2, for a fixed P (which may contain small interior angles), the time complexity of procedure DELTRIANG is expected to be $O(n_t)$ as the triangle size parameter h varies.

CHAPTER 6

PERFORMANCE OF THE METHOD

We have implemented an experimental prototype of the triangulation method described in Chapters 3, 4, and 5 in PASCAL and carried out tests of triangulations for a variety of regions to establish the effectiveness of the method. The implementation and tests were done on a VAX 11/780 running the UNIX operating system and triangulations of regions were plotted using the Computer Graphics Laboratory graphics package. In Appendix A we show examples of triangulations of some regions from the literature. In this chapter we report in some detail on the performance of the method and code for major computational experiments on two complex test regions.

The first test region arises in semiconductor device simulation (Fichtner, Rose, and Bank (1983)). It was provided by R. E. Bank and will be referred to as the CMOS region. Internal interfaces partition the region into six subregions which correspond to the various parts of the semiconductor device as shown in Figure 6.1. The piecewise linear description of the boundary has 30 vertex coordinates, 35 edges, and 50 polygon vertices. Referring to the measures for region complexity mentioned in Section 2.1, this region has edge lengths varying from 0.17 to 15.0 with a mean edge length of 2.59, and 9 out of 50 vertices are reflex. In Figure 6.1, narrow subregions can be seen at the top of the device.

The second test region is the outline of Lake Superior as provided by the Canada Centre for Inland Waters, Environment Canada. The region is made up of the outer boundary of the lake plus six major islands; the piecewise linear description of the boundary has 303 edges and vertices. This region has edge lengths varying from 2.6 to 163.5 (in units of 350 metres) with a mean edge length of 25.1. Over half of its vertices are reflex; 160 out of 303. In the outline of the lake shown in Figure 6.2, several narrow channels can be seen, particularly near the group of islands at the western end and near the northernmost island.

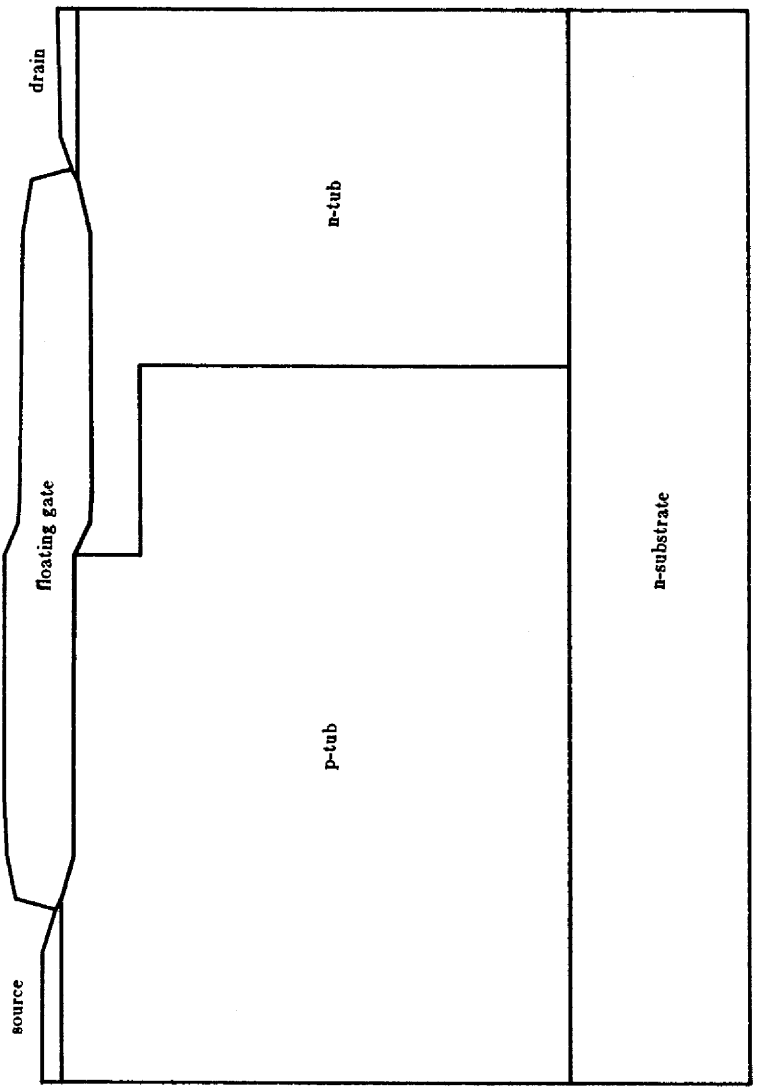


Figure 6.1 CMOS region with six subregions determined by internal interfaces

Figure 6.2 Outline of Lake Superior with six islands

In the method there are two input parameters N_t and κ and five 'internal' parameters σ, τ, λ, $d_{\min}$, and $n_{\min}$. These parameters are defined again in Appendix B. In Section 6.1 we discuss the internal parameters and performance measurements for the method. In Section 6.2 we report on the effects of varying the internal parameters and κ for the two test regions with a fixed value of N_t. In Section 6.3 we report on the performance of the method for the two test regions as N_t and κ vary and the internal parameters are fixed. The validity of all the triangulations of the two test regions was checked using the mesh verification algorithm of Simpson (1981).

6.1. Internal parameters and performance measurements

In this section we discuss the choice of 'default' values for the internal parameters and the information used to measure the performance of the method for the triangulation of a region. The internal parameters consist of the angle parameters σ, τ, and λ and the parameters $d_{\min}$ and $n_{\min}$ used in criteria (4.9) to determine when a polygon is further subdivided in the second stage.

Parameter σ is used to insert extra vertices on edges which subtend a large angle at a reflex vertex and at the centroid of a polygon in the first and second stages, respectively. These extra vertices allow more choices for the endpoints of a separator so that it is more likely that a separator can be found which does not create small angles at the boundary (see Sections 3.2 and 4.2).

Parameters τ and λ are used to indicate a 'small' angle and an 'acceptable' angle at the boundary, respectively. If $\theta < \tau$ then θ is a small angle. If $\theta \geq \lambda$ then θ is an acceptable angle. In procedure RESOLVE of the first stage, if the best separator(s) with endpoint(s) from the Voronoi neighbours V results in an angle smaller than τ at the boundary, then the entire visible vertex set $V_1 \cup V_2$ is used to find the best separator(s) to resolve the reflex vertex. Also, parameter λ is used as a preference for one separator over two separators. If the best separator in the inner cone results in all angles at the boundary being at least λ then this separator is used to resolve the reflex vertex (see Section 3.2). In the second stage, if the best separator chosen from the strategy of isolating the higher values of ψ on one side results in an angle smaller than τ at the boundary, then a separator chosen on the basis of the shape of the polygon is used to subdivide the polygon

(see Section 4.2).

Parameter $d_{\min}$ is used to indicate 'sufficiently high' variation of ψ in a polygon P_i and parameter $n_{\min}$ is used to indicate 'sufficiently large' number of triangles in P_i. If the variation of ψ in P_i determined by the ratio of the standard deviation and mean of ψ in P_i is greater than $d_{\min}$ and the estimated number of triangles in P_i is greater than $n_{\min}$ then P_i is further subdivided in the second stage.

We have chosen to use the default values of $\sigma = 30°$, $\tau = 20°$, $\lambda = 2\tau = 40°$, $d_{\min} = 0.5$, and $n_{\min} = 10$ for the internal parameters. These values have given satisfactory results in preliminary testing of the method. We now elaborate on why we have chosen these values. In the paragraph following Lemma 4.2, it is mentioned that σ must not exceed 30° in order to guarantee that a separator is found for the subdivision of a polygon in the second stage. If σ is smaller than 30°, then there will be more function evaluations done in the second stage to find separators since the number of triangles T_j in which numerical quadrature is applied increases as σ decreases (see Section 4.2). We have chosen $\tau = 20°$ since we want to avoid triangles with angles smaller than 20° in the triangulation if possible. Bykat (1976) also uses 20° as a tolerance for small angles at the boundary in his decomposition algorithm. The angles of the triangles produced by procedure INTTRIANG are between 45° and 90° (see Section 5.3). We have chosen $\lambda = 40°$ to be slightly smaller than the smallest possible angle produced by procedure INTTRIANG.

Parameters $d_{\min}$ and $n_{\min}$ determine how well a triangulation is approximately equidistributing with respect to ψ. By (4.9), polygon P_i is not further subdivided if $\sigma_i / \bar{\psi}_i \leq d_{\min}$, i.e. the values of ψ in P_i within one standard deviation of $\bar{\psi}_i$ differ by a factor of at most $(1 + d_{\min})/(1 - d_{\min})$. For $d_{\min} = 0.5$, $(1 + d_{\min})/(1 - d_{\min}) = 3$ which is less than the factor of four required for the difference of means of ψ in adjacent polygons (see (4.11)). If $d_{\min}$ is smaller than 0.5, then the triangulation will be more approximately equidistributing of ψ due to the smaller variation of ψ in each polygon, but there will be more polygons and function evaluations in the second stage. If the number of triangles in a polygon is small (e.g. $\leq n_{\min} = 10$) then the triangle sizes are partially determined by the boundary geometry so that subdividing the polygon

further may not result in a more approximately equidistributing triangulation.

Now we discuss performance measurements for the triangulation of a region. We compute the following information in order to measure the effects of the variation of the internal parameters, how well the four objectives of (2.1) are satisfied, and how efficient the method is. In the first stage the following values are used to measure the effect of the angle parameters σ, τ, and λ:

N_p = number of convex polygons in the decomposition of the region,

N_{scp2} = number of reflex vertices resolved with two separators,

N_{vis} = number of reflex vertices resolved using the entire visible vertex set,

θ_{min1} = minimum angle created at the boundary by a separator in the first stage.

In the second stage the following values are used to measure the effects of the internal parameters:

N_s = number of convex polygons in the further decomposition,

N_{up} = number of underpopulated polygons (see last paragraph of Section 4.2),

Ψ = the sum in (4.13) which indicates how much the $\overline{\psi}_i$ are modified to satisfy (4.11),

N_f = number of function evaluations of η or ψ,

N_{shap} = number of separators based on the shape of the polygon,

θ_{min2} = minimum angle created at the boundary by a separator in the second stage.

From Section 4.4, N_s and N_f also measure the cost of the further decomposition. We note that $\Psi \geq 1$ with $\Psi = 1$ only if no $\overline{\psi}_i$ are modified. Ψ is further discussed in the last paragraph of this section.

In the third stage the following values are computed:

N_Δ = actual number of triangles in the triangulation,

N_{swap} = number of edge swaps in converting the $VT(P_i)$ to $DT(P_i)$,

N_{LOP} = number of applications of LOP in all the polygons,

N_{comp} = number of triangle comparisons in the searches of triangles in all the applications of LOP.

N_Δ indicates how well objective (2.1a) is satisfied. From Sections 5.6 and 5.7, N_{swap}, N_{LOP}, and N_{comp} measure the cost of constructing the Delaunay triangulations in the polygons. In our implementation of procedure CONVERT, we make it more efficient by not applying LOP to edge $EM(k)$ in the kth iteration of the for loop if it is known that $EM(k)$ is locally optimal (see last sentence of Section 5.6). Also, for edge $EM(k)$, it is known that $TM(kk)$ is one of the triangles containing this edge so a search is made only for the other triangle containing this edge (starting sequentially backwards from $TM(kk-1)$), where kk is either k or $k+1$. With these two additions to procedure CONVERT, the values of N_{LOP} and N_{comp} are smaller.

To measure the empirical time complexity of the method, the CPU times of the three stages are determined:

t_1 = CPU time for the first stage,

t_2 = CPU time for the second stage,

t_3 = CPU time for the third stage.

Timings on the UNIX operating system appear to be subject to a variation of up to 20% for the same program and data at different times. Nevertheless, the gross features of the CPU times for the three stages can be seen from t_1, t_2, and t_3.

To give an indication of the performance of the method with respect to objective (2.1b), i.e. avoiding generating triangles with small angles, we compute the frequency distribution of triangle angles in a triangulation in intervals of 15° and the minimum and maximum angles in the triangulation.

The distribution of length scales implied by the boundary has been quantified for our method in the mesh distribution function, ψ, parameterized by the smoothing parameter κ. The method attempts to generate triangles which approximately equidistribute ψ (see Chapter 4). Let $p_i = \iint_{\Delta_i} \psi \, dA$ for a triangle, Δ_i, and let $\bar{p}$ be the mean value of the p_i for a triangulation. If we

set $q_i = p_i / \bar{p}$, then we see that a triangulation which equidistributes ψ would have $q_i = 1$ for all i. To quantify how well a triangulation approximately equidistributes ψ (i.e. how well objective (2.1c) is satisfied), we estimate the p_i values for its triangles by numerical integration and compute the frequency distribution of the corresponding q_i values in intervals of 0.4. We also compute the minimum, maximum, and standard deviation of the q_i values. For two triangulations based on the same ψ, the one with the 'tighter' distribution of q_i values about the mean value of 1.0 is better.

Objective (2.1d) is to control the rate of change of the triangle sizes in a triangulation. This is partially done by the constraint (4.10) that the triangle sizes, h_i, must differ by a factor of at most two in adjacent polygons. The mesh smoothness parameter κ is also used to control the rate of change of the triangle sizes. To see how the variation of the triangle sizes (i.e. the 'smoothness' of the triangulation) is affected by κ, we compute the square root of the areas of the triangles in a triangulation and compute some statistics for this distribution. Let s_i be the scaled square root of the area of a triangle, Δ_i, where the s_i values are scaled so that the mean value is 1.0 as for the q_i values above. We compute the minimum, maximum, and standard deviation of the s_i values. For two triangulations with different rates of change of triangle sizes, we regard the one with a smaller standard deviation of s_i values as being the smoother triangulation.

The sum $\Psi = \sum \overline{\psi}_{mod,i} A_i$ in (4.13) measures the conflict between objectives (2.1b), (2.1d) and how well a triangulation approximately equidistribute ψ. In order to avoid triangles with small angles, the triangle sizes, h_i, in adjacent polygons are constrained to differ by a factor of at most two. This control over the rate of change of the triangle sizes in adjacent polygons is done by modifying the means $\overline{\psi}_i$ to $\overline{\psi}_{mod,i}$ in (4.12) so that they differ by a factor of at most four in adjacent polygons. This means that the triangle sizes in some polygons may not conform to the mesh distribution function ψ, and for a 'large' value of Ψ the triangulation of the region will be less approximately equidistributing.

6.2. Effects of varying the parameters

In this section we report on an experiment to examine the effects of varying the internal parameters and input parameter κ for the Lake Superior region, the more complicated of the two test regions. (We briefly report on a similar experiment for the CMOS region at the end of this section.) In this experiment, λ is set to 2τ for all the triangulations, the default values of the internal parameters are $\sigma = 30°$, $\tau = 20°$, $d_{\min} = 0.5$, and $n_{\min} = 10$ as mentioned in the preceding section, the default value of κ is 0.25, and N_t is fixed at 2000. To examine the effects of varying a particular parameter, either σ, τ, $d_{\min}$, $n_{\min}$, or κ, we fix the remaining parameters at their default values and generate triangulations for five different values of the parameter which is varied. The values for the variation of each parameter are:

σ = 20°, 25°, 30°, 35°, 40°;

τ = 10°, 15°, 20°, 25°, 30°, $\lambda = 2\tau$;

$d_{\min}$ = 0.3, 0.4, 0.5, 0.6, 0.7;

$n_{\min}$ = 1, 5, 10, 15, 20;

κ = 0.0, 0.125, 0.25, 0.375, 0.5;

The triangulation of the Lake Superior region using the default values of the parameters is shown in Figure 6.12 in the next section. There are twenty other triangulations in which one of the parameters does not have its default value. The results for varying the parameters are shown in Tables 6.1 to 6.5. The middle column in these tables contains the results for the default values of the parameters. In these tables we report on some of the performance measurements mentioned in the preceding section. Those measurements in which the variation of parameters have little or no effect are not tabulated. The CPU times, t_2 and t_3, increase as the number of subregions, N_s, increases. This will be illustrated in the next section. The values N_p, N_{sep2}, N_{vis}, N_{shap}, θ_{min1}, and θ_{min2} are affected only by the angle parameters σ and τ so they appear only in the first two tables.

σ	20	25	30	35	40
N_p	131	130	128	126	123
N_{sep2}	9	6	5	5	5
N_{vis}	0	1	1	2	4
N_{shap}	0	3	2	1	0
θ_{min1}	26.3	25.8	24.5	21.1	17.7
θ_{min2}	25.2	20.0	23.7	21.3	20.5
N_s	210	215	224	233	224
N_{up}	15	24	24	22	30
Ψ	1.10	1.13	1.14	1.09	1.08
N_f/N_s	55.9	47.2	42.0	38.1	34.1
s.d. of q_i	.43	.48	.48	.47	.51
s.d. of s_i	.51	.47	.48	.47	.49

Table 6.1
Results from varying σ for Lake Superior

The results from varying σ are shown in Table 6.1. As σ increases, the effects are:

(a) N_{sep2} decreases since there are fewer Voronoi neighbours in the right and left cones. This probably causes the decrease in N_p.

(b) N_{vis} increases and θ_{min1} decreases since there are fewer candidates for endpoints of separators in the first stage.

(c) N_s increases since there are fewer candidates for endpoints of separators in the second stage which means that the separators which isolate the higher values of ψ to one side are not as 'good'.

(d) N_f/N_s decreases since there are fewer triangles in which the numerical quadrature rule is applied.

We observe that most of the reflex vertices are resolved using a Voronoi neighbour and one separator in the first stage (there are 160 reflex vertices in the Lake Superior region). Also most of the separators in the second stage are chosen from the strategy of isolating the higher values of ψ to one side.

τ	10	15	20	25	30
N_p	124	125	128	129	130
N_{sep2}	1	2	5	6	6
N_{vis}	1	1	1	2	3
N_{shap}	0	0	2	4	5
θ_{min1}	22.0	25.8	24.5	25.8	27.8
θ_{min2}	15.4	15.4	23.7	25.3	33.5
N_s	205	211	224	229	229
N_{up}	28	30	24	27	28
Ψ	1.11	1.15	1.14	1.09	1.09
N_f/N_s	41.1	41.2	42.0	42.3	42.4
s.d. of q_i	.54	.54	.48	.47	.48
s.d. of s_i	.46	.45	.48	.49	.49

Table 6.2
Results from varying τ and $\lambda = 2\tau$ for Lake Superior

The results from varying τ are shown in Table 6.2. As τ and $\lambda = 2\tau$ increase, the effects are:

(a) N_{sep2} increases because of the increase in λ. This results in an increase in N_p.

(b) N_{vis}, N_{shap}, θ_{min1}, and θ_{min2} increase because of the increase in τ.

(c) N_s increases due to the increase in N_p and N_{shap}.

$d_{\min}$	0.3	0.4	0.5	0.6	0.7
N_s	275	246	224	192	167
N_{up}	111	59	24	6	2
Ψ	1.04	1.15	1.14	1.17	1.57
N_f/N_s	46.2	44.1	42.0	37.0	31.1
s.d. of q_i	.40	.45	.48	.53	.83
s.d. of s_i	.51	.48	.48	.46	.48

Table 6.3
Results from varying $d_{\min}$ for Lake Superior

The results from varying $d_{\min}$ are shown in Table 6.3. As $d_{\min}$ increases, the effects are:

(a) N_s and N_{up} decrease due to the first criterion in (4.9) for further subdivision.

(b) The standard deviation of the q_i increases (i.e. the triangulation is less approximately equidistributing) since there are fewer subregions with higher variation of ψ.

(c) The second stage decomposition would become saturated (i.e. $N_{up} = 0$) for a smaller value of

N_t due to the decrease in N_{up}.

(d) N_f / N_s decreases due to the decrease in N_s as explained below.

We note that $d_{\min} = 0.7$ results in a large value of Ψ due to not enough subregions in the decomposition. This causes a much larger value for the standard deviation of the q_i. The effect of a large value of Ψ on the standard deviation of the q_i is also illustrated in Tables 6.5 and 6.6.

Now we explain why N_f / N_s increases as N_s increases in Table 6.3. This effect can also be seen in Tables 6.4 and 6.5. In the implementation of our method, $\eta(x,y)$ is first evaluated at quadrature points in the N_p polygons from the first stage decomposition to obtain an estimate for $\overline{\eta}$ in (4.6); then $\psi(x,y)$ is evaluated at quadrature points in each polygon which is subdivided or results from a subdivision in the second stage. To reduce the number of evaluations of η in the first N_p polygons, no evaluations are done if η is constant in a polygon (see end of Section 4.1), and parameter σ is not used to subdivide edges subtending a large angle at the centroid so that there are fewer triangles T_j in which numerical quadrature is applied. For the remaining polygons, in which ψ is evaluated, parameter σ is used to subdivide edges subtending a large angle at the centroid so that there are extra candidates for the endpoints of a separator. Therefore N_f / N_s increases as N_s (and the number of subdivisions) increases. However, N_f / N_s should approach a constant as N_s gets sufficiently large.

$n_{\min}$	1	5	10	15	20
N_s	253	244	224	188	173
N_{up}	0	6	24	40	48
Ψ	1.16	1.17	1.14	1.11	1.11
N_f/N_s	45.4	44.7	42.0	36.5	33.3
s.d. of q_i	.45	.46	.48	.55	.60
s.d. of s_i	.48	.48	.48	.47	.46

Table 6.4
Results from varying $n_{\min}$ for Lake Superior

The results from varying $n_{\min}$ are shown in Table 6.4. As $n_{\min}$ increases, the effects are:

(a) N_s decreases and N_{up} increases due to the second criterion in (4.9) for further subdivision.

(b) The standard deviation of the q_i increases but the effect is less significant than for $d_{\min}$.

(c) The second stage decomposition would become saturated for a larger value of N_t due to the increase in N_{up}.

κ	0.0	0.125	0.25	0.375	0.5
N_s	265	240	224	189	157
N_{up}	49	27	24	19	15
Ψ	1.15	1.21	1.14	1.39	1.42
N_f/N_s	45.7	43.8	42.0	36.6	28.0
s.d. of q_i	.50	.52	.48	.67	.64
s.d. of s_i	.63	.52	.48	.45	.35

Table 6.5
Results from varying κ for Lake Superior

The results from varying κ are shown in Table 6.5. As κ increases, the effects are:

(a) N_s and N_{up} decrease since there is less variation in ψ, which is parameterized by κ.

(b) The standard deviation of the s_i decreases since there is less variation in ψ.

(c) The second stage decomposition would become saturated for a smaller value of N_t due to the decrease in N_{up}.

A similar experiment was performed on the CMOS region in which N_t is fixed at 1000 and the parameters are varied as for the Lake Superior region. The effects of varying the parameters for this region are similar to that for the Lake Superior region except that N_p, N_{sep2}, N_{vis}, and N_{shap} are not affected since the CMOS region is less complicated (there are $N_p = 14$ subregions after the first stage decomposition). However, the value of Ψ for the CMOS region (and the standard deviation of the q_i) is more sensitive to the second stage decomposition obtained. This is likely due to the large variation of ψ in going from the top to the bottom of the region. The sensitivity of Ψ is illustrated in Table 6.6 for the variation of $d_{\min}$.

$d_{\min}$	0.3	0.4	0.5	0.6	0.7
N_s	61	34	28	21	18
N_{up}	5	0	0	0	0
Ψ	1.12	1.24	1.40	1.80	1.24
N_f/N_s	56.6	49.8	47.3	40.9	33.8
s.d. of q_i	.35	.45	.51	.76	.61
s.d. of s_i	.58	.57	.53	.40	.51

Table 6.6
Results from varying $d_{\min}$ for CMOS region

From the tables in this section we conclude that

(1) the default values $\sigma = 30°$, $\tau = 20°$, $\lambda = 40°$, and $n_{\min} = 10$ give satisfactory results,

(2) internal parameter $d_{\min}$ should be set small enough so that Ψ is not large and the standard deviation of the q_i is sufficiently small,

(3) input parameter κ can be used to obtain the desired 'smoothness' of the triangulation, i.e. variation in triangle sizes.

6.3. Performance on two test regions

In this section we report on the performance of the method for major experiments on the two test regions. For each test region, we generate three series of meshes using smoothing parameter values $\kappa = 0.0$, $\kappa = 0.25$, and $\kappa = 0.5$.

For the CMOS region we generate meshes for desired number of triangles N_t = 500, 1000, 1500, 2000, 2500, and 3000 in each series. Increasing N_t from 500 to 3000 corresponds roughly to decreasing the triangle sizes by a factor of 0.4 and going from a coarse to a fine mesh from the viewpoint of finite element analysis. The internal parameters, except for $d_{\min}$, are held fixed at their default values, i.e. $\sigma = 30°$, $\tau = 20°$, $\lambda = 40°$, and $n_{\min} = 10$. $d_{\min}$ is fixed at 0.3 instead of the default value of 0.5 to get triangulations which are more approximately equidistributing since $d_{\min} = 0.4$ and $d_{\min} = 0.5$ result in larger values of Ψ (see Table 6.6). The decomposition of the CMOS region into 14 convex polygons after the first stage is shown in Figure 6.3. The decomposition after the second stage for $N_t = 1000$ and $\kappa = 0.25$ is shown in Figure 6.4. Observe that most

of the convex polygons are at the top of the region where the narrow subregions cause a large variation of ψ. The general character of the triangulations can be seen from the case $N_t = 1000$ shown in Figure 6.5 for $\kappa = 0.0$, Figure 6.6 for $\kappa = 0.25$, and Figure 6.7 for $\kappa = 0.5$. To illustrate the details at the finer mesh levels, the triangulation of the right half of the region is shown in Figure 6.8 for $N_t = 3000$ and $\kappa = 0.25$.

For the Lake Superior region, we generate meshes for desired number of triangles N_t = 500, 1000, 2000, 3000, 4000, and 5000 in each series. The region is too complex for a reasonable triangulation of only 500 triangles, so the case $N_t = 500$ examines the behaviour of the method when N_t is too small. Increasing N_t from 1000 to 5000 corresponds roughly to decreasing the triangle sizes by a factor of 0.45 and going from a coarse to a medium sized mesh from the viewpoint of finite element analysis. The internal parameters are held fixed at their default values, i.e. $\sigma = 30°$, $\tau = 20°$, $\lambda = 40°$, $d_{\min} = 0.5$, and $n_{\min} = 10$. The decomposition of the Lake Superior region into 128 convex polygons after the first stage is shown in Figure 6.9. The decomposition after the second stage for $N_t = 2000$ and $\kappa = 0.25$ is shown in Figure 6.10. The general character of the triangulations can be seen from the case $N_t = 2000$ shown in Figure 6.11 for $\kappa = 0.0$, Figure 6.12 for $\kappa = 0.25$, and Figure 6.13 for $\kappa = 0.5$. To illustrate the details at the finer mesh levels, the triangulation of the western end of the lake is shown in Figure 6.14 for $N_t = 5000$ and $\kappa = 0.25$.

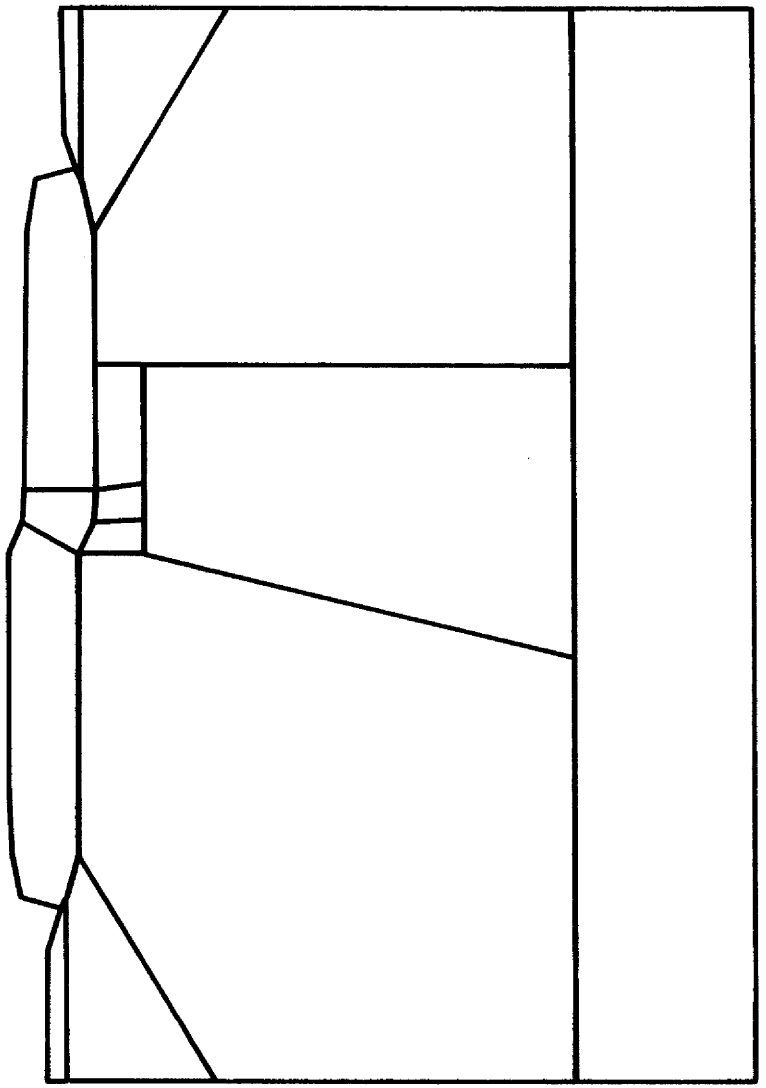


Figure 6.3 First stage decomposition of CMOS region

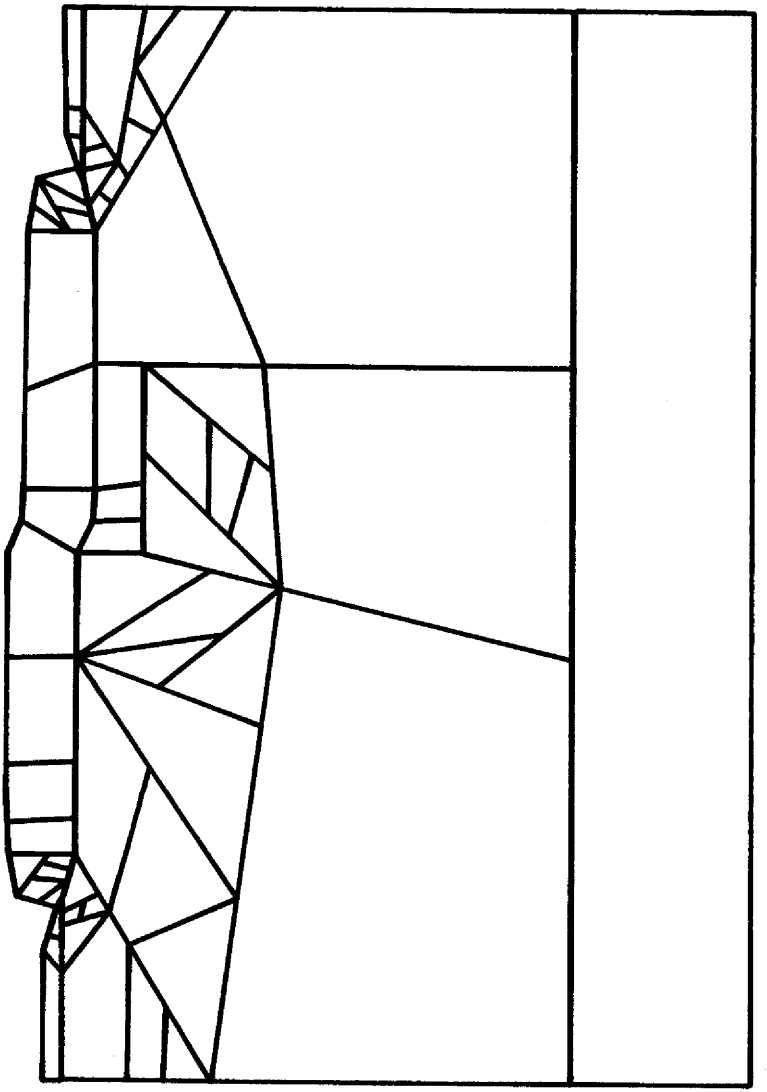


Figure 6.4 Second stage decomposition of CMOS region for $N_t = 1000$, $\kappa = 0.25$

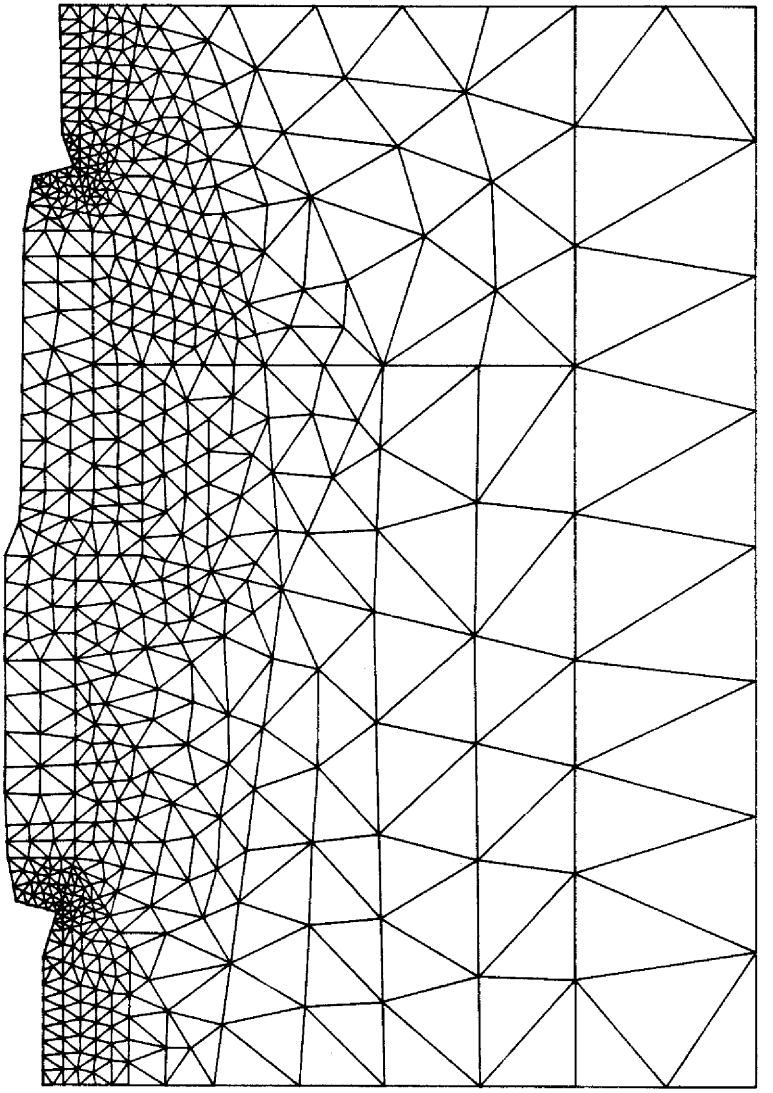


Figure 6.5 Triangulation of CMOS region for $N_t = 1000$, $\kappa = 0.0$

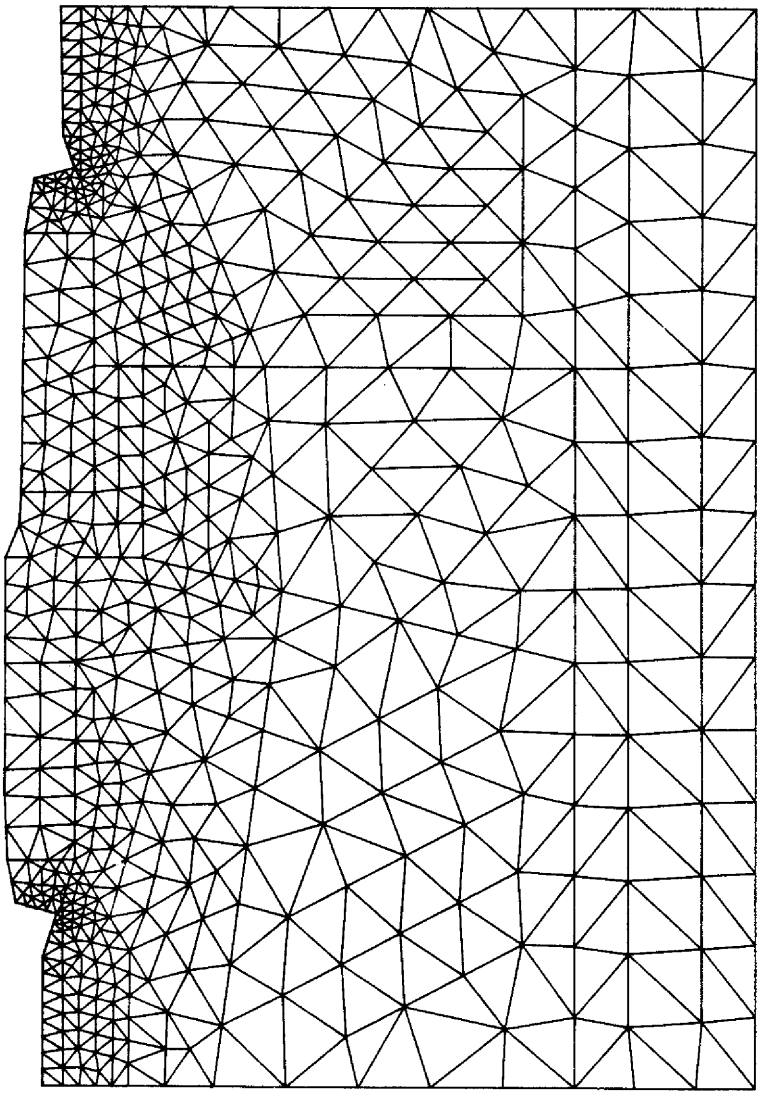


Figure 6.6 Triangulation of CMOS region for $N_t = 1000$, $\kappa = 0.25$

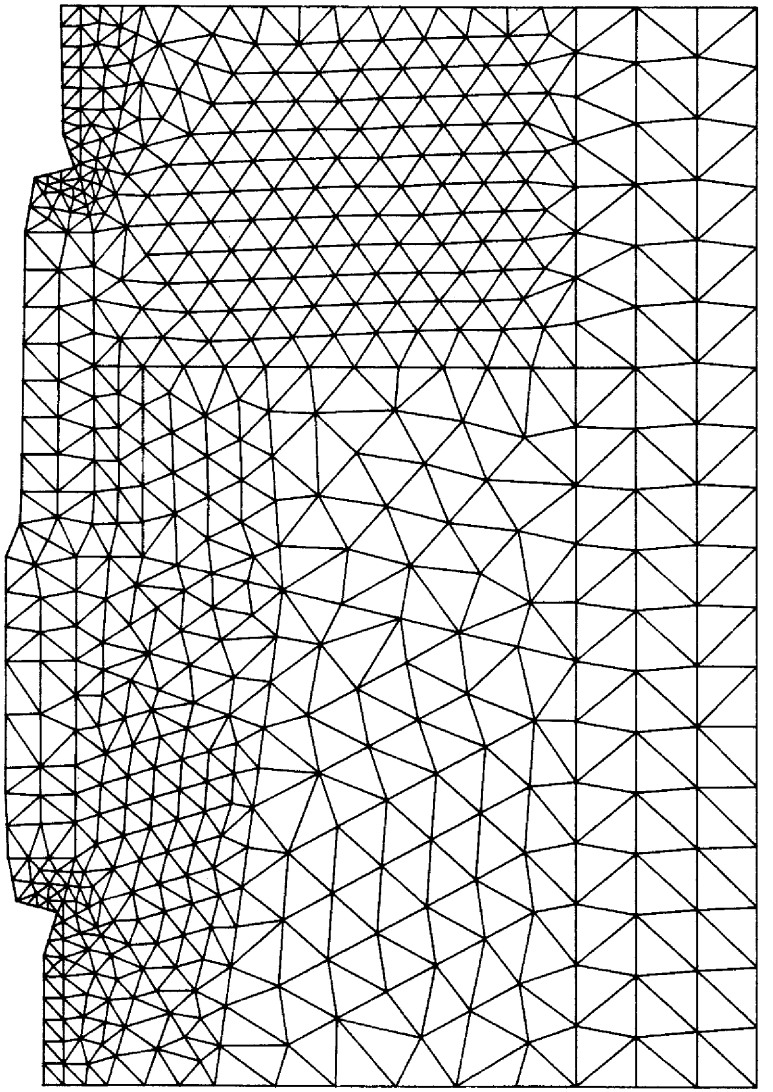


Figure 6.7 Triangulation of CMOS region for $N_i = 1000$, $\kappa = 0.5$

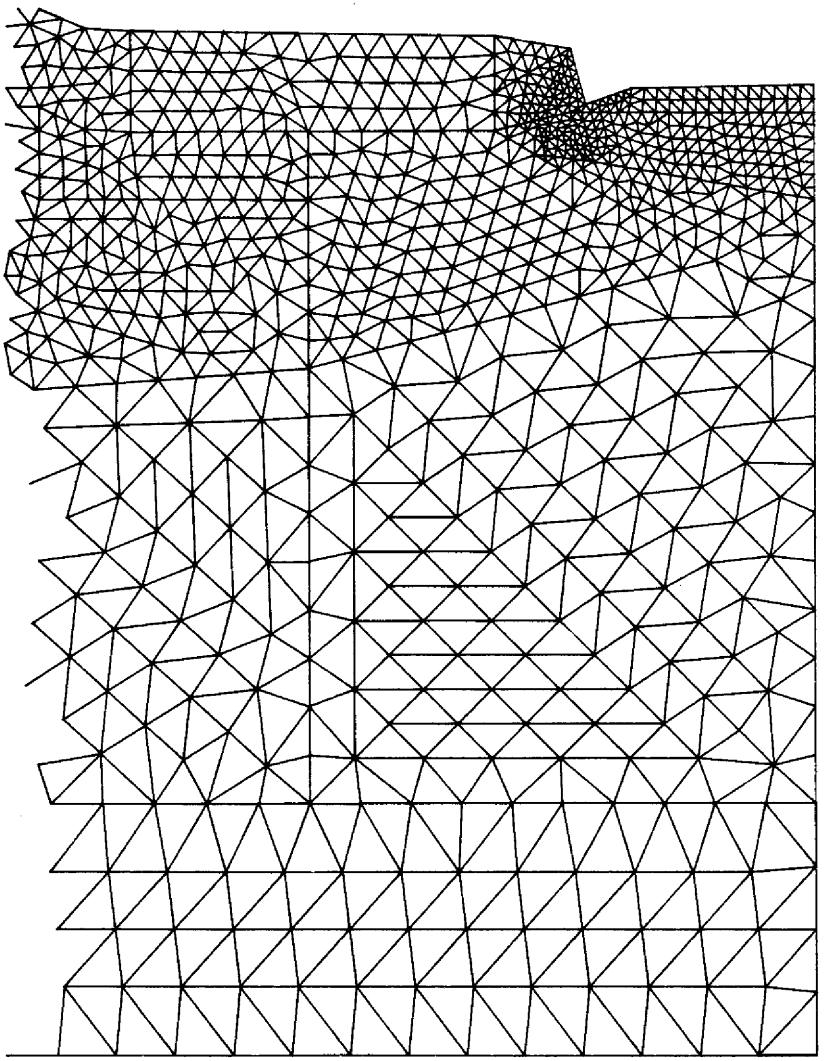


Figure 6.8 Right half of triangulation of CMOS region for $N_t = 3000$, $\kappa = 0.25$

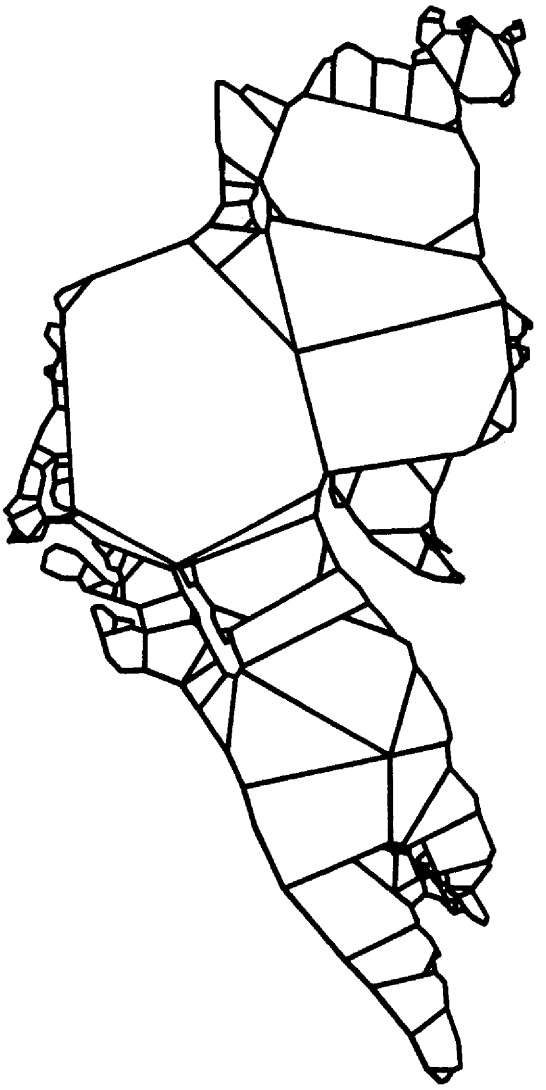


Figure 6.9 First stage decomposition of Lake Superior

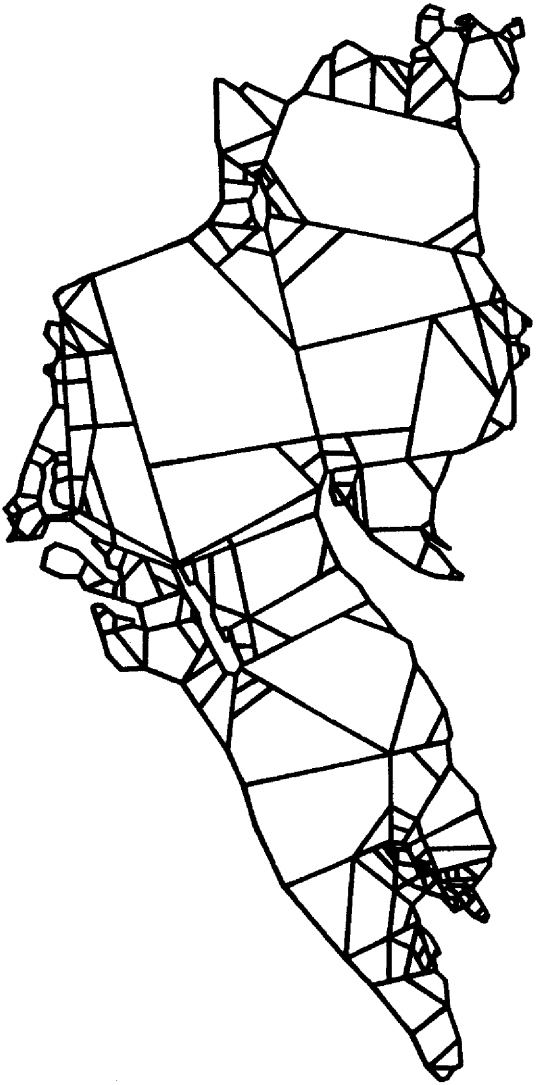


Figure 6.10 Second stage decomposition of Lake Superior for $N_t = 2000$, $\kappa = 0.25$

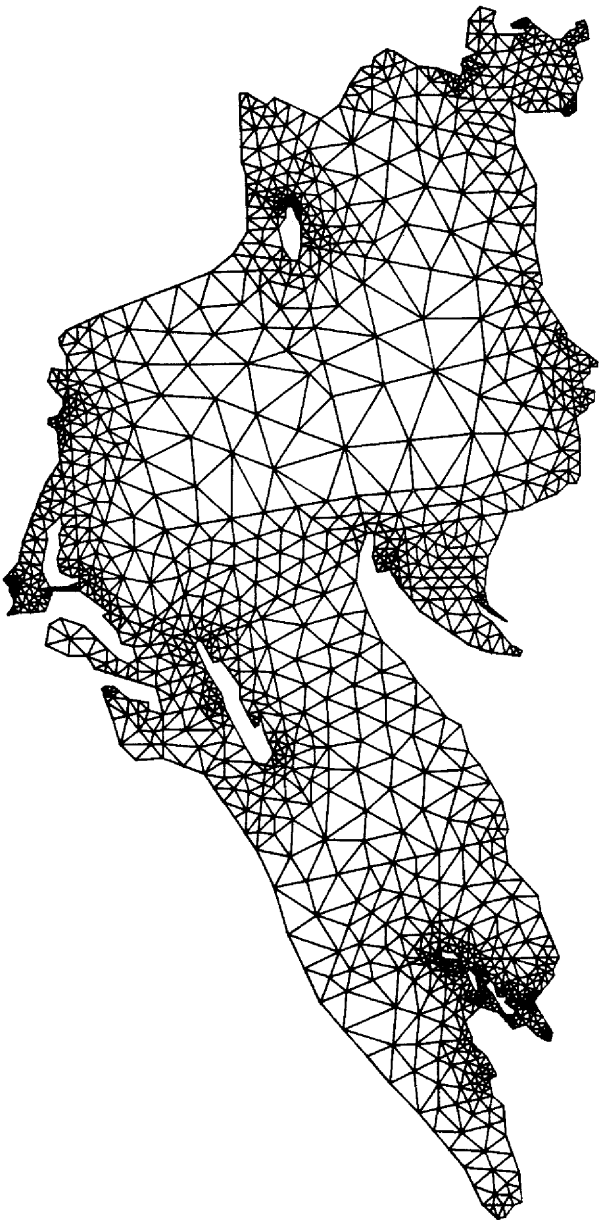


Figure 6.11 Triangulation of Lake Superior for $N_t = 2000$, $\kappa = 0.0$

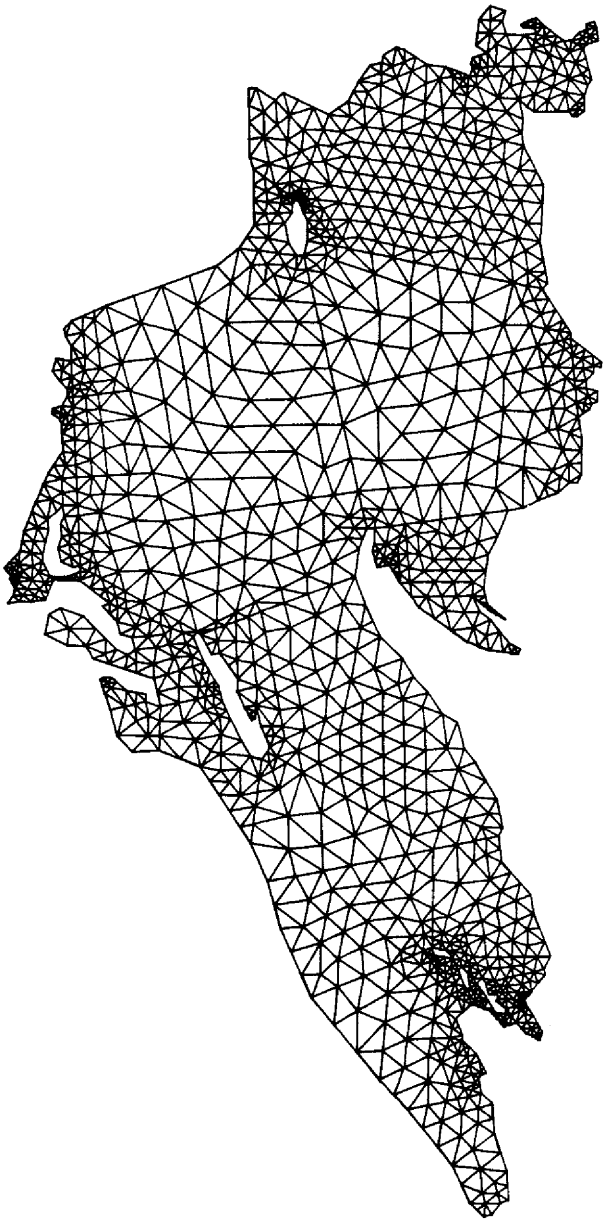


Figure 6.12 Triangulation of Lake Superior for $N_t = 2000$, $\kappa = 0.25$

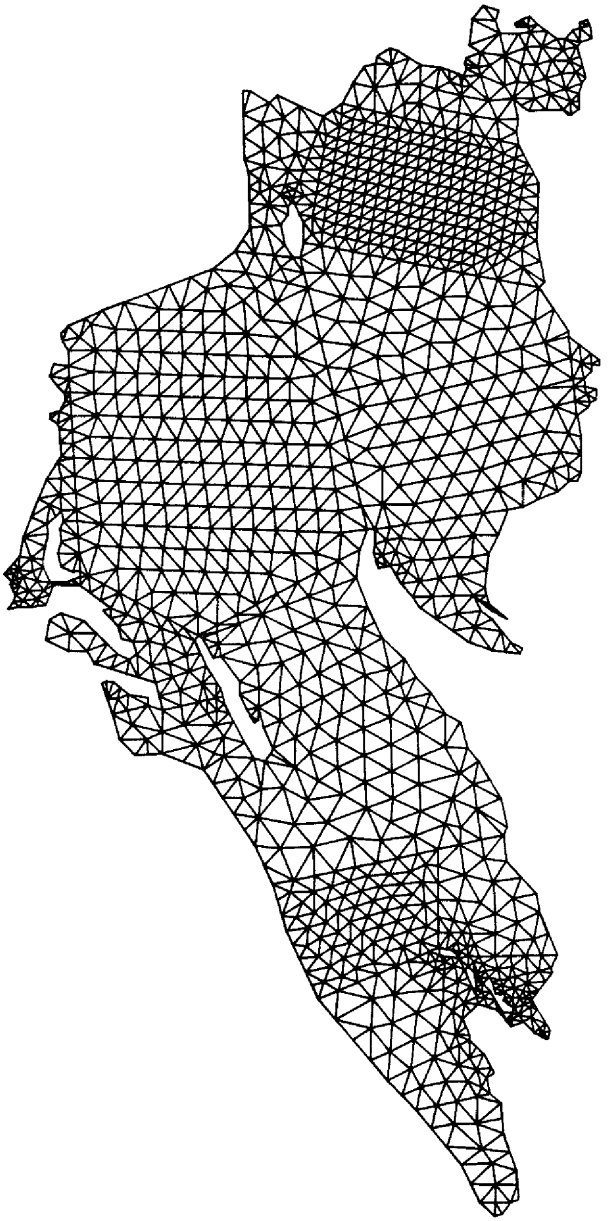


Figure 6.13 Triangulation of Lake Superior for $N_t = 2000$, $\kappa = 0.5$

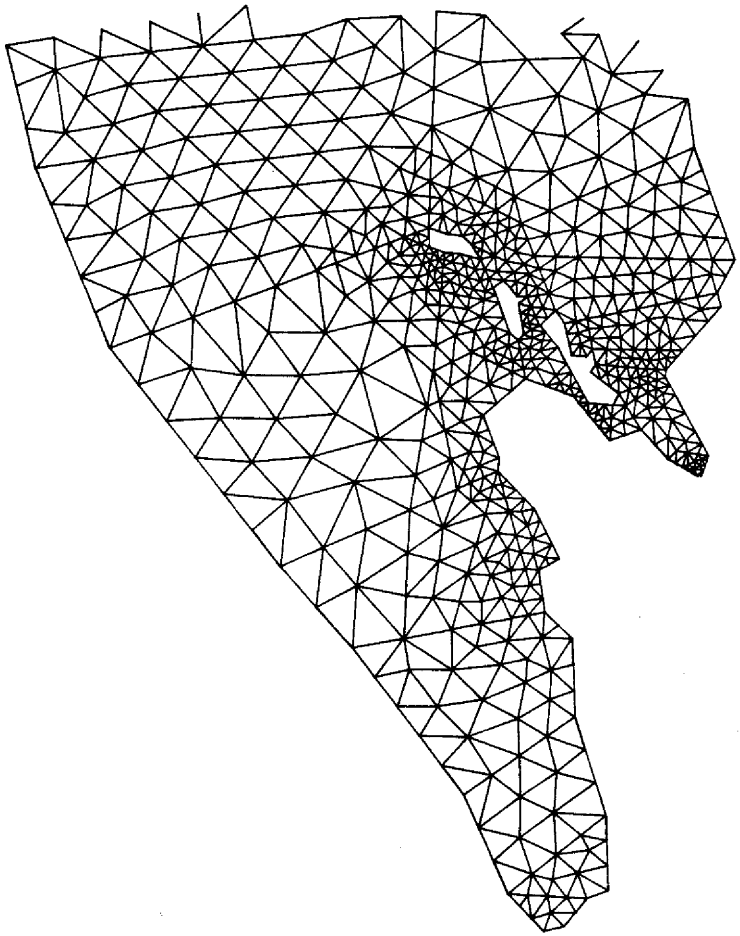


Figure 6.14 Western end of triangulation of Lake Superior for $N_t = 5000$, $\kappa = 0.25$

N_t		500	1000	1500	2000	2500	3000
N_s		56	73	79	80	80	80
N_{up}	$\kappa = 0.0$	19	7	1	0	0	0
Ψ		1.24	1.21	1.12	1.12	1.12	1.12
N_f/N_s		55.9	58.1	58.4	58.5	58.5	58.5
N_s		47	61	64	66	66	66
N_{up}	$\kappa = .25$	14	5	2	0	0	0
Ψ		1.18	1.12	1.17	1.07	1.07	1.07
N_f/N_s		54.5	56.6	56.9	57.1	57.1	57.1
N_s		36	44	48	49	49	50
N_{up}	$\kappa = .5$	11	6	2	1	1	0
Ψ		1.46	1.30	1.32	1.38	1.38	1.22
N_f/N_s		51.6	53.7	54.6	54.7	54.7	54.9

Table 6.7
Results from the second stage for CMOS region
($N_p = 14$ after the first stage)

N_t		500	1000	2000	3000	4000	5000
N_s		148	195	265	290	307	321
N_{up}	$\kappa = 0.0$	66	73	49	29	15	4
Ψ		1.42	1.22	1.15	1.10	1.11	1.10
N_f/N_s		25.1	37.2	45.7	47.5	48.5	49.2
N_s		142	173	224	240	244	248
N_{up}	$\kappa = .25$	51	48	24	10	6	3
Ψ		1.18	1.11	1.14	1.16	1.17	1.17
N_f/N_s		22.2	33.3	42.0	44.0	44.7	45.0
N_s		129	141	157	165	168	171
N_{up}	$\kappa = .5$	26	23	15	8	4	2
Ψ		1.44	1.42	1.42	1.42	1.42	1.42
N_f/N_s		15.6	22.3	28.0	30.2	31.1	31.7

Table 6.8
Results from the second stage for Lake Superior
($N_p = 128$ after the first stage)

The first stage decomposition does not depend on κ or N_t so no results are tabulated from the first stage. In Tables 6.7 and 6.8 we provide some results from the second stage for the CMOS and Lake Superior regions, respectively. As N_t increases, the number of subregions, N_s, increases and the number of underpopulated polygons, N_{up}, decreases due to the second criterion in (4.9) for further subdivision. For sufficiently large N_t, the number of convex polygons in the total decomposition becomes independent of N_t (i.e. $N_{up} = 0$), and increasing N_t further raises the

number of triangles per polygon, i.e. the decomposition becomes saturated. For the CMOS region, the decomposition becomes saturated (or nearly saturated in the case $\kappa = 0.5$) for N_t between 1500 and 2000. For the Lake Superior region, the decomposition is nearly saturated at $N_t = 5000$.

For both regions, the values of Ψ are larger for $\kappa = 0.5$ than for $\kappa = 0.0$ and $\kappa = 0.25$. This indicates that mesh distribution function $\psi(x,y)$ parameterized by $\kappa = 0.5$ contains too much 'smoothing' since the mean values of ψ in adjacent polygons differ by a factor of greater than four for some large polygons in which the variation of ψ is small. This suggests to us that $\kappa = 0.25$ is close to being an upper limit on the amount of smoothing that is useful for these two regions (also see Table 6.5).

We have measured the CPU time spent in the routine which does the numerical integration of ψ or η and usually over 80% of the second stage CPU time is spent in this routine. Therefore the number of evaluations of ψ or η, N_f, is a measure of the cost of the further decomposition to boundary scales. In Table 6.7, it can be seen that the ratio N_f / N_s is between 50 and 60 for all eighteen triangulations. In Table 6.8, it can be seen that N_f / N_s approaches a constant as N_s increases. This agrees with the result in Section 4.4 which states that the number of function evaluations is $O(mN_s)$ where m is the maximum number of edges of a polygon from the first stage decomposition. From Figures 6.3 and 6.9 it can be seen that m is a small constant (less than 20) for the two regions so that N_f should be proportional to N_s.

In Tables 6.9 and 6.10 we provide a comparison of the actual number of triangles generated, N_Δ, for each mesh to the requested number, N_t, and measurements of the empirical time complexity of the third stage for the CMOS and Lake Superior regions, respectively. For the CMOS region N_Δ is within 10% of N_t for $N_t \geq 1000$ and for the Lake Superior region N_Δ is within 10% of N_t for $N_t \geq 4000$. For both regions, the ratio N_Δ / N_t gets closer to one as N_t increases. This is due to more triangles per polygon as N_t increases which means that (4.15) is more accurate in determining the triangle size (and number of triangles) for each polygon.

N_t		500	1000	1500	2000	2500	3000
N_Δ		578	1096	1616	2141	2598	3131
N_{swap}/N_s	$\kappa = 0.0$	1.6	2.3	2.8	3.2	3.5	3.9
N_{LOP}/N_{swap}		4.8	4.4	4.3	4.4	4.3	4.3
N_{comp}/N_{LOP}		1.7	1.8	1.9	1.9	1.9	1.9
N_Δ		575	1079	1627	2122	2597	3069
N_{swap}/N_s	$\kappa = .25$	1.8	2.2	3.3	3.5	3.7	3.8
N_{LOP}/N_{swap}		4.7	4.6	4.3	4.3	4.4	4.4
N_{comp}/N_{LOP}		1.7	1.8	2.0	1.9	1.9	1.8
N_Δ		532	1052	1557	2044	2593	3079
N_{swap}/N_s	$\kappa = .5$	1.2	1.3	1.8	2.4	3.2	3.1
N_{LOP}/N_{swap}		5.5	6.2	5.3	5.0	4.6	4.8
N_{comp}/N_{LOP}		1.6	1.5	1.6	1.6	1.9	1.7

Table 6.9
Results from the third stage for CMOS region

N_t		500	1000	2000	3000	4000	5000
N_Δ		779	1259	2396	3411	4398	5419
N_{swap}/N_s	$\kappa = 0.0$	1.0	1.0	1.7	1.9	2.1	2.2
N_{LOP}/N_{swap}		4.4	5.4	4.8	4.9	4.8	4.9
N_{comp}/N_{LOP}		1.7	1.6	1.7	1.7	1.7	1.7
N_Δ		768	1255	2285	3340	4338	5331
N_{swap}/N_s	$\kappa = .25$	1.0	1.1	1.5	2.0	2.1	2.4
N_{LOP}/N_{swap}		4.5	5.4	5.0	4.8	5.0	4.9
N_{comp}/N_{LOP}		1.7	1.6	1.7	1.7	1.7	1.7
N_Δ		779	1223	2208	3207	4212	5243
N_{swap}/N_s	$\kappa = .5$	0.9	1.1	1.2	1.6	2.0	2.1
N_{LOP}/N_{swap}		4.4	4.7	5.3	5.0	5.0	5.0
N_{comp}/N_{LOP}		1.7	1.8	1.6	1.8	1.7	1.7

Table 6.10
Results from the third stage for Lake Superior

We observe that in all 36 triangulations of the two regions, procedures INTTRIANG and MERGE produced a valid triangulation in every convex polygon, i.e. procedure MMERGE was not needed, and the first edge, u_0v_0, produced by procedure MERGE was a Delaunay edge in every polygon which contained at least one interior mesh vertex (see Section 5.4). Also there were few or no interior angles smaller than $\tau = 20°$ in the convex polygons produced by the decomposition of the first two stages. From Section 5.7, the time complexity of procedure DELTRIANG for constructing a Delaunay triangulation in a convex polygon depends on the number of edge swaps and

triangle comparisons made in the applications of LOP in procedure CONVERT. These two tables show that the average number of swaps per polygon, N_{swap}/N_s, is small and increases at a very slow rate as N_t increases, and the ratios N_{LOP}/N_{swap} and N_{comp}/N_{LOP} are small and independent of N_t and κ. Therefore, for the two test regions, the empirical time complexity of procedure DELTRIANG for constructing a Delaunay triangulation in a convex polygon is linear in the number of triangles in the polygon, and the empirical time complexity of the third stage triangulation of the region is $O(N_\Delta) = O(N_t)$.

N_t		500	1000	1500	2000	2500	3000
min		17.6	22.1	22.1	21.8	22.0	22.1
max	$\kappa = 0.0$	118.5	129.0	131.0	132.6	124.7	119.8
[a] 45−90		71.9	76.1	78.2	80.7	81.6	82.2
min		16.7	19.4	19.4	23.2	23.2	23.5
max	$\kappa = .25$	121.4	126.6	120.1	132.6	132.6	119.8
[a] 45−90		72.6	76.7	80.4	82.5	83.9	83.5
min		16.7	23.7	23.7	23.7	22.4	23.2
max	$\kappa = .5$	119.5	119.5	126.6	126.6	122.4	120.2
[a] 45−90		78.9	83.1	84.0	87.0	87.7	87.5

Table 6.11
Distribution of triangle angles for CMOS region
[a] table entry = percentage of angles in specified range

N_t		500	1000	2000	3000	4000	5000
min		4.8	11.9	12.2	12.2	12.2	12.2
max	$\kappa = 0.0$	149.7	139.2	124.2	135.0	135.0	137.3
[a] 45−90		60.2	66.8	71.0	72.9	74.6	76.1
min		4.8	10.5	11.9	12.2	12.2	12.2
max	$\kappa = .25$	149.7	144.4	124.7	128.8	135.0	135.0
[a] 45−90		60.9	69.2	73.8	76.4	78.3	79.2
min		5.9	7.9	10.9	12.2	12.2	12.2
max	$\kappa = .5$	149.7	143.2	139.2	127.6	121.6	117.2
[a] 45−90		63.1	72.4	80.7	82.8	84.4	85.7

Table 6.12
Distribution of triangle angles for Lake Superior
[a] table entry = percentage of angles in specified range

To give an indication of the performance of the method with respect to objective (2.1b), we give in Tables 6.11 and 6.12 some statistics on the distribution of angles present in the triangulations. These tables show the minimum and maximum angles and the percentage of angles that

fall in the range 45° to 90°. All the angles of the triangles produced by procedure INTTRIANG are between 45° and 90° so the angles outside this range are in the triangles produced by procedures MERGE and CONVERT. As N_t increases, there are more triangles per polygon and a greater percentage of triangles are produced by procedure INTTRIANG, therefore the percentage of angles in the range 45° to 90° increases. For the Lake Superior region, the triangulations with $N_t = 500$ contain too few triangles and result in much smaller minimum angle and larger maximum angle, and the minimum angle of 12.2° for $N_t \geq 3000$ occurs between consecutive boundary edges. The influence of the smoothing parameter κ on the minimum and maximum angles seem to be minor but the percentage of angles in the 45° to 90° range increases as κ increases due to fewer subregions and more triangles per subregion as κ increases.

N_t		500	1000	1500	2000	2500	3000
min		.18	.15	.14	.13	.14	.13
max		2.59	2.54	2.67	2.72	2.47	2.86
s.d.	$\kappa = 0.0$	.47	.44	.36	.36	.36	.36
[a] .8−1.2		26.1	32.4	41.6	38.5	34.8	37.4
[a] .4−1.6		82.5	80.9	91.1	91.7	92.4	92.2
min		.20	.24	.23	.20	.18	.19
max		2.80	2.23	2.60	2.46	2.64	2.42
s.d.	$\kappa = .25$	.40	.35	.37	.29	.29	.29
[a] .8−1.2		38.1	38.6	35.5	56.4	56.0	52.8
[a] .4−1.6		89.4	93.6	91.8	95.1	94.7	95.1
min		.26	.30	.31	.28	.28	.26
max		3.42	2.36	2.80	2.75	2.93	2.47
s.d.	$\kappa = .5$	.54	.41	.45	.49	.48	.36
[a] .8−1.2		21.4	25.2	21.8	18.2	22.1	32.3
[a] .4−1.6		77.6	91.6	89.0	79.5	81.7	94.2

Table 6.13
Distribution of q_i for CMOS region
[a] table entry = percentage of q_i values in specified range

N_t		500	1000	2000	3000	4000	5000
min		.05	.11	.12	.10	.10	.12
max		5.47	4.84	4.23	2.95	3.14	3.42
s.d.	$\kappa = 0.0$	.79	.73	.50	.45	.45	.45
[a] .8−1.2		15.1	22.6	31.6	32.4	34.0	32.9
[a] .4−1.6		56.1	68.6	79.9	83.6	82.9	82.1
min		.11	.20	.17	.20	.23	.15
max		4.20	4.48	7.49	10.75	3.64	3.58
s.d.	$\kappa = .25$	.60	.58	.48	.47	.45	.46
[a] .8−1.2		21.9	29.2	37.2	36.5	34.2	34.6
[a] .4−1.6		72.0	80.5	84.8	84.1	83.3	83.0
min		.09	.13	.20	.17	.17	.17
max		3.43	3.71	5.94	8.34	8.28	8.25
s.d.	$\kappa = .5$	.60	.62	.64	.64	.65	.63
[a] .8−1.2		19.9	28.5	31.4	31.2	30.4	30.5
[a] .4−1.6		69.2	68.1	70.5	70.4	69.7	69.0

Table 6.14
Distribution of q_i for Lake Superior
[a] table entry = percentage of q_i values in specified range

To see how well the triangulations of the two test regions approximately equidistribute ψ, we give in Tables 6.13 and 6.14 some statistics on the distribution of q_i values. We observe that the standard deviation of q_i values is influenced by Ψ (see Tables 6.7 and 6.8) and the distribution of q_i values stabilizes for the larger values of N_t. It should be recognized that ψ, parameterized by κ, is different for each series of triangulations so that the distributions of the q_i for the three different ψ, i.e. for $\kappa = 0.0$, 0.25, and 0.5, are not strictly comparable in the sense that the three ψ represent different choices of how to quantify the length scale data from the boundary curves. As κ increases, the method is less sensitive to these length scales; there is a less variation in ψ and fewer subregions are generated. However, the conclusion we are able to draw from this is simply that it is more difficult (and expensive) to generate meshes that are approximately equidistributing for less smooth ψ. We do not have a methodology for addressing the question of which choice of ψ (i.e. κ) is more appropriate for the applications short of numerical experiments involving the finite element method itself.

N_t		500	1000	1500	2000	2500	3000
min		.20	.16	.20	.22	.19	.18
max	$\kappa = 0.0$	5.28	5.77	4.48	4.45	4.97	5.04
s.d.		.89	.89	.90	.88	.90	.87
min		.18	.20	.17	.20	.22	.18
max	$\kappa = .25$	2.66	2.65	2.68	2.77	2.40	2.50
s.d.		.60	.58	.61	.61	.61	.61
min		.16	.26	.23	.27	.21	.22
max	$\kappa = .5$	2.15	1.95	1.99	2.25	2.20	2.25
s.d.		.42	.39	.39	.41	.40	.40

Table 6.15
Distribution of s_i for CMOS region

N_t		500	1000	2000	3000	4000	5000
min		.13	.16	.16	.16	.15	.16
max	$\kappa = 0.0$	3.35	4.03	4.50	4.86	5.00	5.41
s.d.		.59	.59	.63	.66	.68	.68
min		.12	.16	.15	.16	.17	.15
max	$\kappa = .25$	2.71	2.60	2.70	2.62	2.69	2.74
s.d.		.56	.50	.48	.47	.46	.46
min		.12	.15	.20	.19	.19	.19
max	$\kappa = .5$	2.39	2.28	2.09	1.97	2.13	2.15
s.d.		.48	.39	.35	.33	.33	.33

Table 6.16
Distribution of s_i for Lake Superior

To see how the variation of triangle sizes is affected by smoothing parameter κ, we give in Tables 6.15 and 6.16 some statistics on the distribution of s_i values. For a fixed value of κ, the standard deviation of s_i values is approximately constant as N_t increases. As κ increases, the standard deviation of s_i values decreases due to less variation in ψ, i.e. the triangulation is smoother.

N_t		500	1000	1500	2000	2500	3000
t_1		.40	.38	.43	.40	.40	.42
t_2		5.63	7.47	8.38	8.32	8.35	8.30
t_3	$\kappa = 0.0$	1.55	2.48	3.23	3.82	4.25	4.63
total		7.58	10.33	12.04	12.54	13.00	13.35
t_2/N_s		.101	.102	.106	.104	.104	.104
t_1		.42	.37	.42	.40	.38	.42
t_2		4.53	6.02	6.27	6.42	6.42	6.53
t_3	$\kappa = .25$	1.43	2.18	3.00	3.40	3.72	4.02
total		6.38	8.57	9.69	10.22	10.52	10.97
t_2/N_s		.096	.099	.098	.097	.097	.099
t_1		.43	.42	.43	.42	.40	.42
t_2		3.22	4.23	4.55	4.58	4.55	4.62
t_3	$\kappa = .5$	1.02	1.62	2.00	2.27	2.77	3.05
total		4.67	6.27	6.98	7.27	7.72	8.09
t_2/N_s		.089	.096	.095	.093	.093	.092

Table 6.17
CPU times in seconds for the three stages for CMOS region

N_t		500	1000	2000	3000	4000	5000
t_1		8.05	7.78	7.88	7.82	7.98	7.70
t_2		10.72	20.13	32.92	35.87	38.70	40.67
t_3	$\kappa = 0.0$	3.18	4.60	7.77	9.33	11.20	12.58
total		21.95	32.51	48.57	53.02	57.88	60.95
t_2/N_s		.072	.103	.124	.124	.126	.127
t_1		8.00	7.97	8.12	7.82	7.98	8.13
t_2		9.50	15.47	24.63	26.43	27.82	27.98
t_3	$\kappa = .25$	3.10	4.48	6.65	8.47	9.50	10.82
total		20.60	27.92	39.40	42.72	45.30	46.93
t_2/N_s		.067	.089	.110	.110	.114	.113
t_1		7.87	7.88	7.75	7.65	7.77	7.87
t_2		4.57	6.87	9.32	10.37	10.75	11.08
t_3	$\kappa = .5$	2.87	3.48	4.67	5.58	6.62	7.38
total		15.31	18.23	21.74	23.60	25.14	26.33
t_2/N_s		.035	.049	.059	.063	.064	.065

Table 6.18
CPU times in seconds for the three stages for Lake Superior

The CPU times for the three stages of the method, t_1, t_2, t_3, and the total time, $t_1 + t_2 + t_3$ are reported in Tables 6.17 and 6.18 for the CMOS and Lake Superior regions, respectively. The time for the first stage, t_1, should be constant. It is reported in these tables to give an indication of the variability of the timings on the UNIX operating system. These tables show that the

largest fraction of the total time is spent in the second stage, which is due to the many evaluations of ψ. The direct relation between this time, t_2, and the number of subregions for a fixed value of κ as N_t increases is shown by the ratio t_2/N_s in these tables. As κ increases, t_2 and t_3 decrease due to fewer subregions. In Figures 6.15 and 6.16, graphs of CPU times versus N_t are shown for the CMOS and Lake Superior regions, respectively, for $\kappa = 0.0$ and $\kappa = 0.25$. In these graphs, t_1 is the average of the eighteen times reported in Tables 6.17 and 6.18. The slope of the graph of t_2 versus N_t decreases as N_t increases due to the slower increase in N_s. For the CMOS region, t_2 is constant for $N_t \geq 2000$ when the decomposition is saturated. The linear time complexity of the third stage and the method is clearly evident in the graphs of t_3 and total time versus N_t. The slope of these graphs decreases as N_t increases due to the slower increase in N_s.

Finally we provide a comparison of the total CPU time and distribution of triangle angles for our method versus subroutine TRIGEN of the PLTMG package (see G in Section 1.2). Subroutine TRIGEN is written in FORTRAN and was compiled using the F77 compiler on the UNIX operating system. Subroutine TRIGEN also avoids generating triangles with small angles and the number of triangles in the triangulation of a region is determined by the boundary geometry and a parameter h for the maximum triangle side length. Subroutine TRIGEN cannot handle multiply connected regions so we attempted to triangulate the Lake Superior region with no islands. Subroutine TRIGEN failed to produce a triangulation for various values of h. For the CMOS region, we obtained triangulations of 1213 and 2095 triangles from subroutine TRIGEN using $h = 1.0$ and $h = 0.5$, respectively. The first triangulation is shown in Figure 6.17. The CPU times for these two triangulations were 27.08 and 43.45 seconds, respectively. This compares with a total CPU time of 13.35 seconds from our PASCAL implementation for $N_t = 3000$ and $\kappa = 0.0$ (the most expensive of the eighteen triangulations). The distribution of triangle angles from subroutine TRIGEN is similar to that from our method. Both triangulations have a minimum angle of 21.5° and a maximum angle of 123.2°. The two triangulations have 71.9% and 78.4% of the angles in the 45° to 90° range, respectively (cf. Table 6.11).

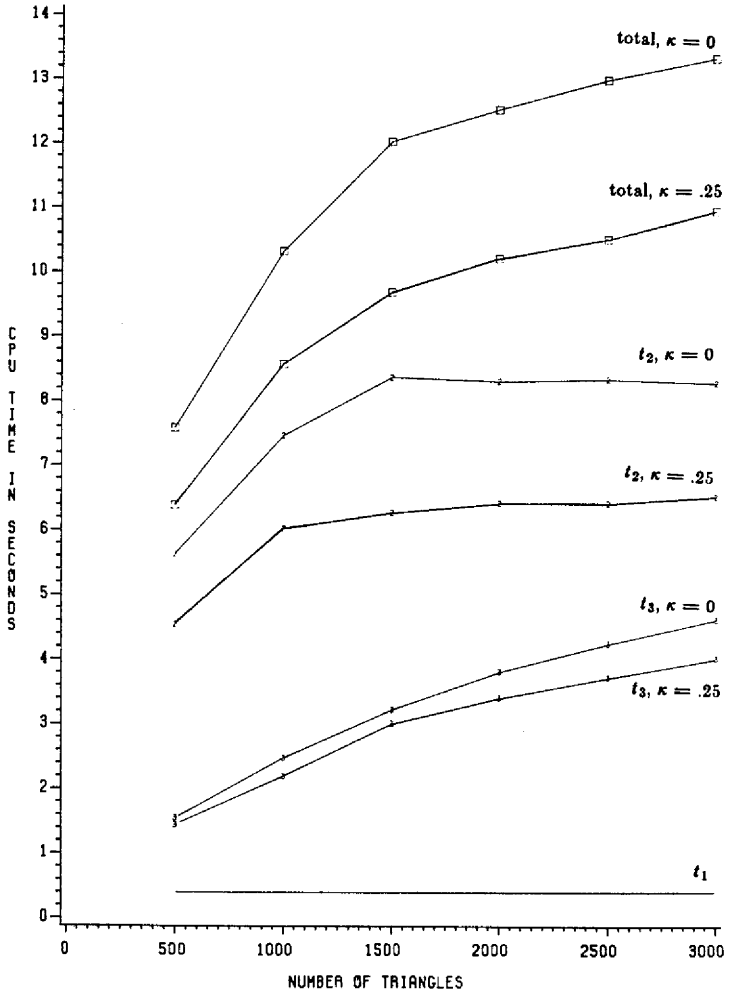


Figure 6.15 Graph of CPU times versus N_t for CMOS region

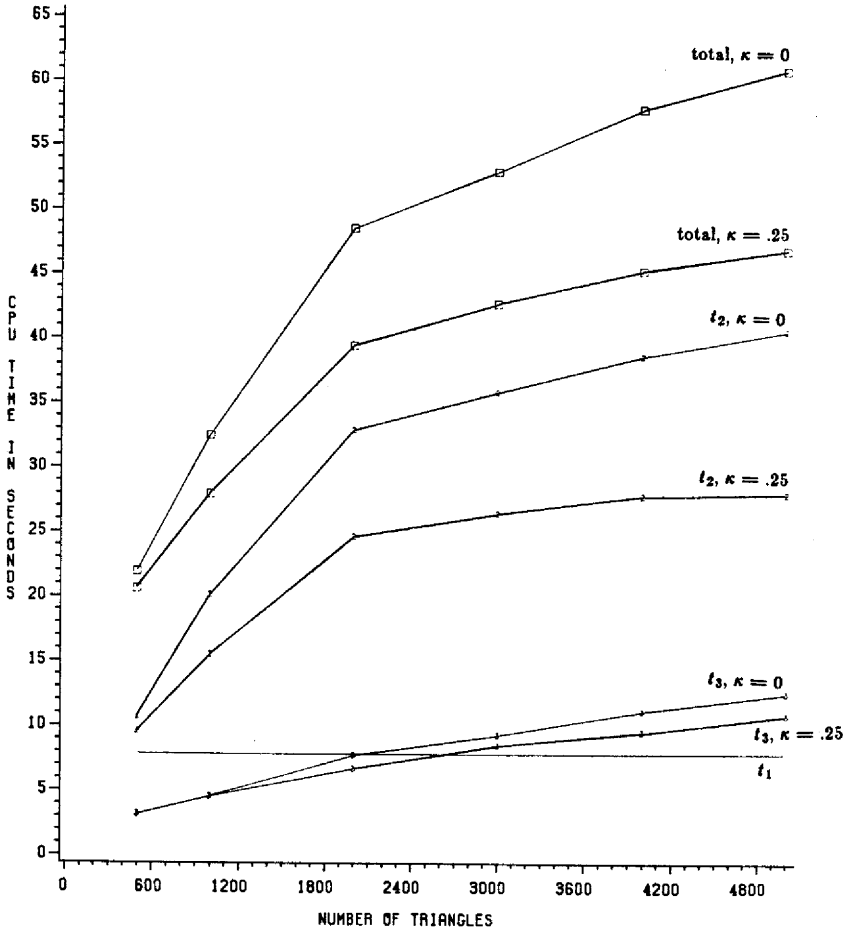


Figure 6.16 Graph of CPU times versus N_t for Lake Superior

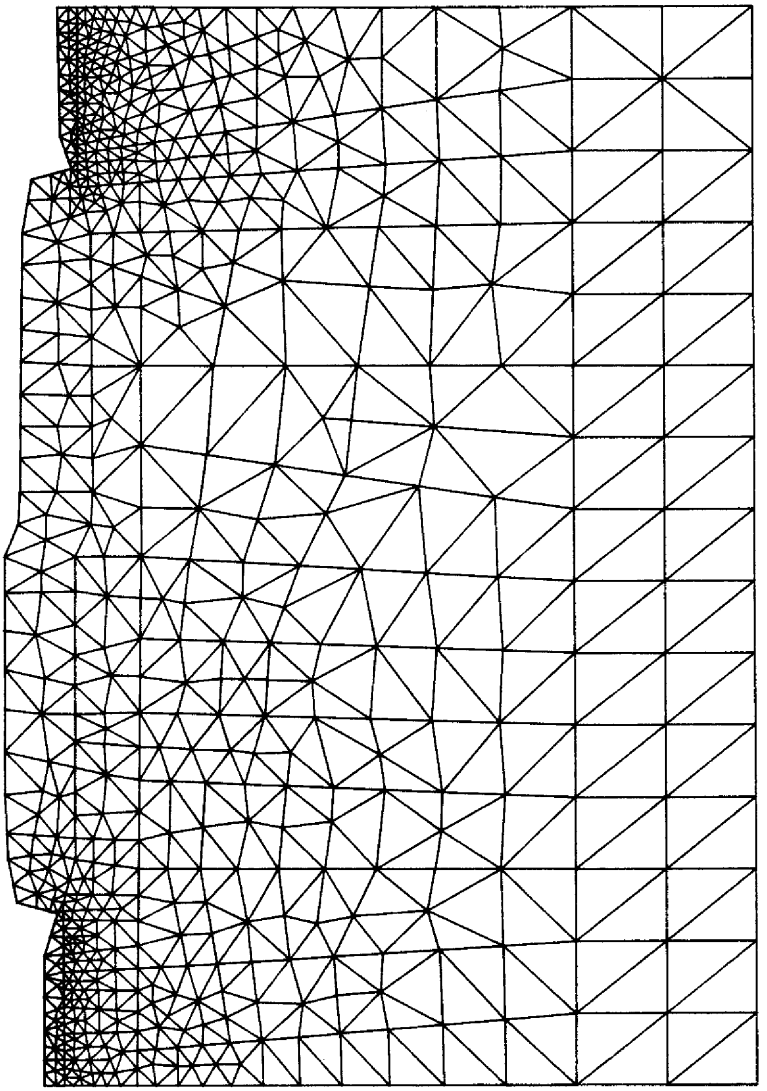


Figure 6.17 Triangulation of CMOS region with 1213 triangles by subroutine TRIGEN

CHAPTER 7

CONCLUDING REMARKS

We conclude this thesis by discussing extensions and open problems in our triangulation method. In Chapter 4, we have defined a heuristic mesh distribution function, $\psi(x,y)$, based on the length scales implied by the boundary and described our approach for constructing an approximately equidistributing triangulation with respect to ψ. Clearly, the method can be extended to construct an approximately equidistributing triangulation with respect to any positive weight function, e.g. a user specified or adaptively determined mesh distribution function.

The user may want to input a mesh distribution function ψ instead of allowing the method to automatically determine ψ if more control is desired over the distribution of triangle sizes or smaller triangles are desired in noncomplicated parts of the region in which the solution of the partial differential equation contains fast variation or singularities. An alternative way to obtain smaller triangles in certain parts of the region is to add extra vertices on the boundary to get short edges and/or add interfaces to get narrow subregions, and use the ψ defined based on boundary length scales.

In adaptive mesh refinement, the error estimate of the finite element solution on a mesh is used to produce a finer mesh, and this process may be repeated several times. Usually, the finer mesh is constructed by the local refinement of the original mesh, i.e. if the error in a triangle is larger than a tolerance parameter then the triangle is further subdivided into smaller triangles (Babuska and Rheinboldt (1978), Bank and Sherman (1980)). Adaptive mesh refinement can be done with our method as follows. The mesh distribution function ψ defined based on boundary length scales can be used to generate an initial coarse mesh. Given a mesh, the error estimate of the finite element solution on this mesh can be used to define a new mesh distribution function ψ. Then, with the decomposition into convex polygons for this mesh and the new ψ as input, the second and third stages of our method can be repeated to produce a finer decomposition into convex polygons and a finer mesh. For efficiency, it may be desirable to repeat the second and third

stages for only the subregions in which the error estimate is too large, i.e. refinement of the mesh is done at the polygon level.

Finally, we mention some problems that we were unable to solve.

(1) An objective of our triangulation method is to have the distribution of triangle sizes conform to the distribution of length scales implied by the boundary. We accomplish this objective by defining a mesh distribution function ψ using the length of edges and the amount of narrowing of the convex polygons in the first stage decomposition (see Section 4.1). But the first stage decomposition is not unique since it depends on the order in which the reflex vertices are resolved. Also, in the testing of our method, artificial narrowings of subregions are occasionally formed in the decomposition. A problem is to find a better heuristic for defining ψ which does not depend on the first stage decomposition but depends only on the length scales implied by the boundary curves and internal interfaces which are input.

(2) In the second stage, an objective is to find separators which do not create small angles at the boundary. Our second strategy is to find a separator which is based on the shape of the polygon (see Section 4.2). In the testing of our method, no angles smaller than $20°$ have been created at the boundary by separators chosen using this strategy when $\sigma = 30°$. A problem is to prove whether $20°$ is a lower bound for the angles resulting from this strategy when $\sigma \leq 30°$.

(3) In the testing of our method, procedure DELTRIANG of the third stage has empirical time complexity which is linear in the number of triangles since the average number of edge swaps per polygon in procedure CONVERT is relatively small. Also, in the case of at least two interior mesh vertices, the first edge u_0v_0 introduced by procedure MERGE is always a Delaunay edge (see beginning of Section 5.4), and the triangulation $T(P)$ produced by procedures INTTRIANG and MERGE is always a valid triangulation. Three unsolved problems are to prove that the heuristic of selecting u_0 results in a Delaunay edge u_0v_0 and to prove Conjectures 5.1 and 5.2, or find counterexamples to these statements.

APPENDIX A

EXAMPLES OF TRIANGULATIONS

In this appendix, we show examples of triangulations of five regions to demonstrate the ability of our method to generate triangulations of complex regions. These regions are not as complicated as the Lake Superior region shown in Chapter 6, and are obtained from papers mentioned in Section 1.2 by manually digitizing the boundaries. For each region, we produce a triangulation for smoothness parameter value $\kappa = 0.25$. Figure A.1 shows a triangulation of an L-shaped region from Shaw and Pitchen (1978) for desired number of triangles $N_t = 300$. Figure A.2 shows a triangulation of a pistol from Bykat (1982) for $N_t = 200$. Figure A.3 shows a triangulation of a machine element from Sadek (1980) for $N_t = 600$. Figure A.4 shows a triangulation of Lake Okeechobee from Thacker, Gonzalez, and Putland (1980) for $N_t = 800$. Figure A.5 shows a triangulation of a bracket from Yerry and Shephard (1983) for $N_t = 800$. The actual number of triangles generated is within 15% of N_t for all these triangulations.

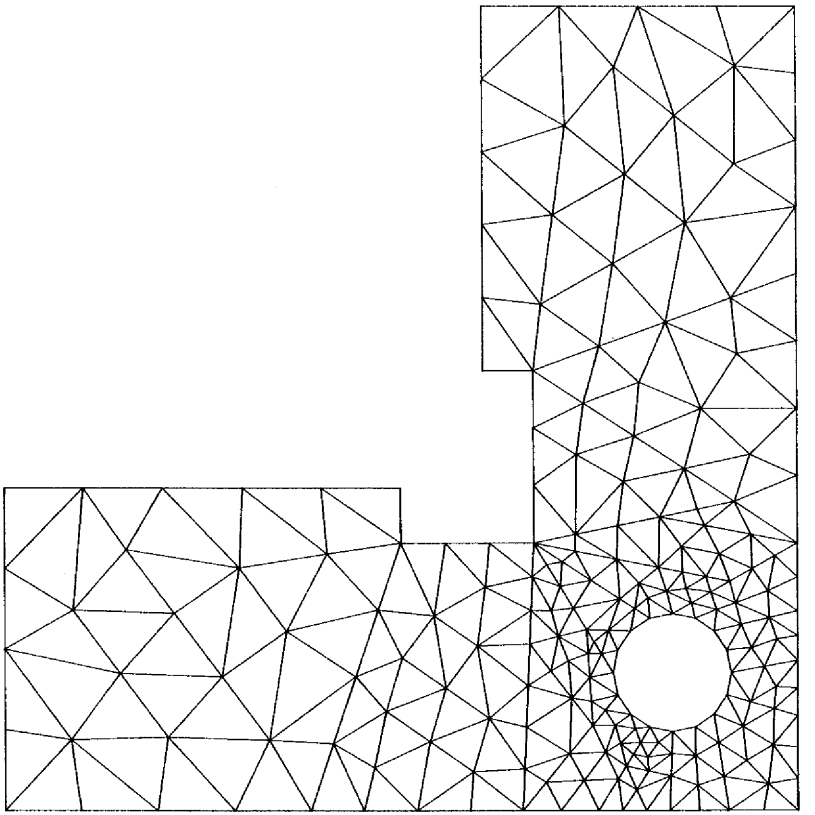


Figure A.1 Triangulation of L-shaped region for $N_t = 300$, $\kappa = 0.25$

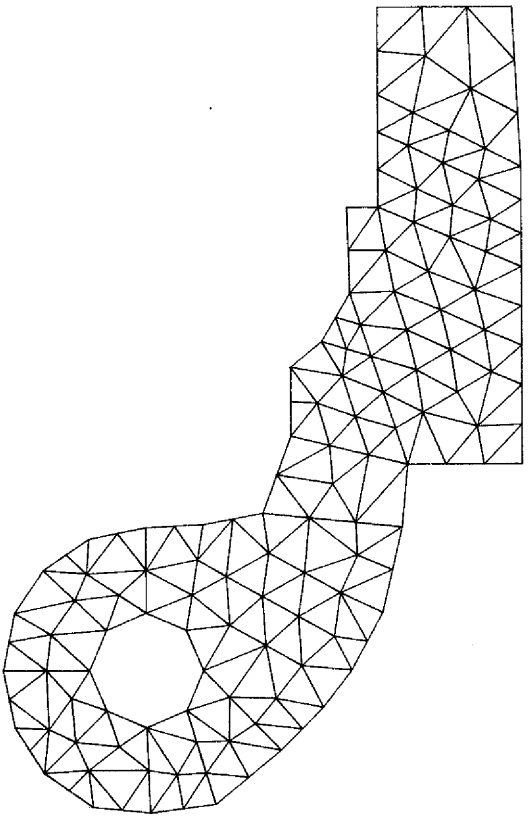


Figure A.2 Triangulation of pistol for $N_t = 200$, $\kappa = 0.25$

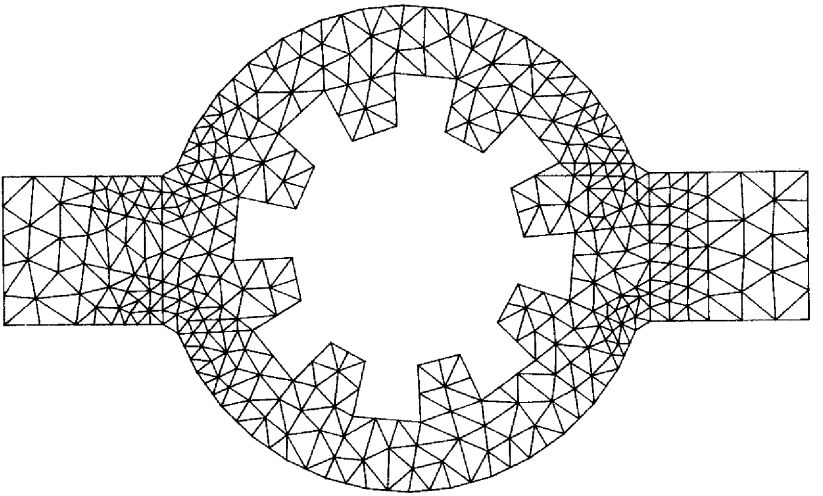


Figure A.3 Triangulation of machine element for $N_t = 600$, $\kappa = 0.25$

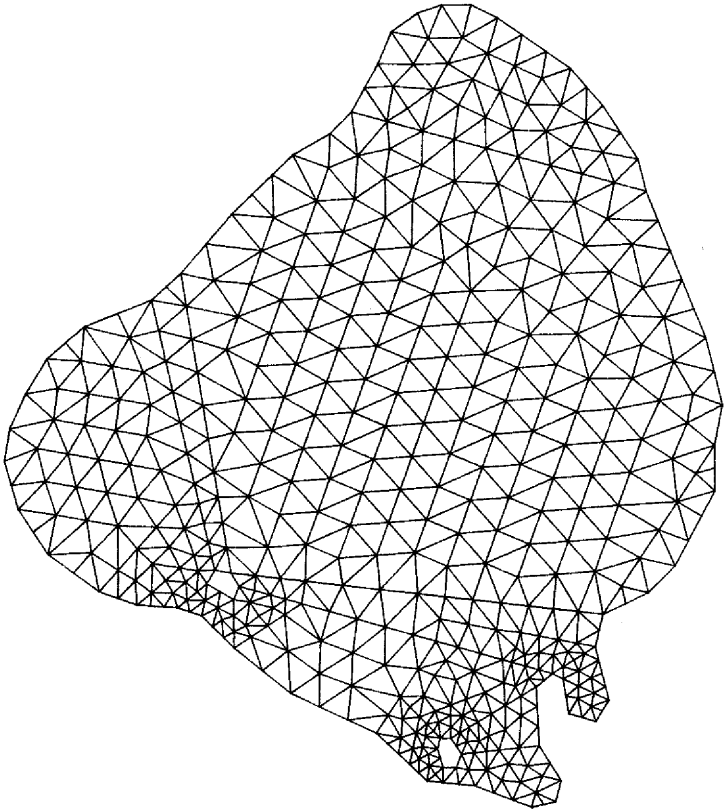


Figure A.4 Triangulation of Lake Okeechobee for $N_t = 800$, $\kappa = 0.25$

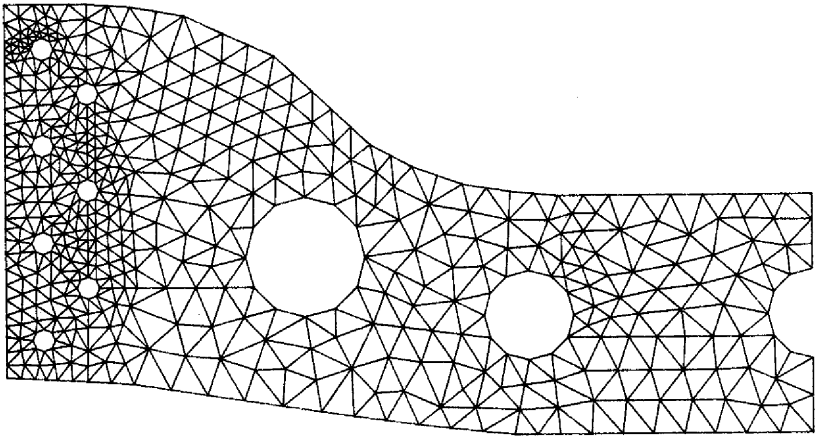


Figure A.5 Triangulation of bracket for $N_t = 800$, $\kappa = 0.25$

APPENDIX B

DEFINITIONS

In this appendix, we provide the definitions of geometric terms used in this thesis and repeat the definitions of the parameters of our triangulation method.

B.1. Geometric terms

Let R be a simple polygon and let ∂R denote the boundary of R. Let u be a point in the interior of R or on ∂R. Point v is said to be *visible* from u if the line segment uv lies entirely in R. The *visibility polygon* $V(R,u)$ from u is the set of points in R visible from u, i.e. $V(R,u) = \{v \mid v \in R \text{ and } uv \cap R = uv\}$ (El Gindy and Avis (1981)).

The *kernel* $K(R)$ of R is the locus of the points in the interior of R from which all vertices of R are visible. Equivalently, if ∂R is considered as a counterclockwise directed cycle, $K(R)$ is the intersection of the half-planes lying to the left of the edges of R (Lee and Preparata (1979)).

Let P be a convex polygon. A *line of support* of P is a line that has at least one point in common with ∂P with the property that all of P lies on one side of the line. The *diameter* or *maximum breadth* of P is the greatest of the distances between two parallel lines of support of P. It is also the greatest distance between two points of P. The *minimum breadth* of P is the minimum of the distances between two parallel lines of support of P (Lyusternik (1963)).

Let V be a set of points in the plane. For each point v of V, there is a convex polygon $P(v)$ surrounding v, called the *Voronoi polygon* associated with v, with the property that v is the closest of the points of V to any point $w \in P(v)$. The Voronoi polygons partition the plane into the *Voronoi diagram*. Points u and v of V are *Voronoi neighbours* if $P(u)$ and $P(v)$ share a common edge. The *Delaunay triangulation* of V is the dual of the Voronoi diagram and is formed by joining all Voronoi neighbours by line segments (Shamos and Hoey (1975)). The Delaunay triangulation also satisfies the max-min angle criterion and the circle criterion (see page 52).

B.2. Parameters of method

In our method, there are two input parameters N_t and κ and five internal parameters σ, τ, λ, $d_{\min}$, and $n_{\min}$.

N_t is the desired number of triangles in the mesh. It is used to determine the uniform triangle size (and number of triangles) for each convex polygon in the second stage decomposition (see page 47).

κ is the mesh smoothness parameter, $0 \leq \kappa \leq 1$. It is used to control the smoothness, i.e. variation in triangle sizes, of the mesh via the mesh distribution function $\psi(x,y)$ defined in the second stage (see page 39).

σ is an angle parameter which is used to insert extra vertices on edges subtending a large angle at a reflex vertex and at the centroid of a polygon in the first and second stages, respectively. These extra vertices allow more choices for the endpoints of a separator so that it is more likely that a separator can be found which does not create small angles at the boundary (see pages 26 and 40).

τ is an angle parameter which is used in the first and second stages to search for separators which do not create 'small' angles at the boundary. Angles less than τ are considered to be small (see pages 28 and 42).

λ is an angle parameter used in the first stage to accept a separator without checking for 'better' separators. A separator is acceptable if it creates angles of at least λ at the boundary (see page 28).

$d_{\min}$ and $n_{\min}$ are parameters used in the second stage to indicate 'sufficiently high' variation of ψ in a polygon P_i and 'sufficiently large' number of triangles in P_i, respectively. If the variation of ψ in P_i determined by the ratio of the standard deviation and mean of ψ in P_i is greater than $d_{\min}$ and the estimated number of triangles in P_i is greater than $n_{\min}$ then P_i is further subdivided in the second stage (see page 41).

REFERENCES

I. Babuska and A. K. Aziz (1976), On the angle condition in the finite element method, *SIAM J. Numer. Anal.*, 13, pp. 214-226.

I. Babuska and W. C. Rheinboldt (1978), Error estimates for adaptive finite element computations, *SIAM J. Numer. Anal.*, 15, pp. 736-754.

R. E. Bank (1982), PLTMG users' guide, Dept. of Mathematics, University of California at San Diego.

R. E. Bank and A. H. Sherman (1980), A refinement algorithm and dynamic data structure for finite element meshes, Rep. CNA TR-166, Center for Numerical Analysis, The University of Texas at Austin.

A. Bykat (1976), Automatic generation of triangular grid: I - subdivision of a general polygon into convex subregions, II - triangulation of convex polygons, *Int. J. for Num. Meth. in Eng.*, 10, pp. 1329-1342.

A. Bykat (1982), Design of a recursive shape controlling mesh generator, Dept. of Mathematics and Computer Science, Emory University.

J. C. Cavendish (1974), Automatic triangulation of arbitrary planar domains for the finite element method, *Int. J. for Num. Meth. in Eng.*, 8, pp. 679-696.

B. Chazelle and D. Dobkin (1979), Decomposing a polygon into its convex parts, *Proc. of the 11th Annual ACM Symp. on Theory of Computing*, pp. 38-48.

H. El Gindy and D. Avis (1981), A linear algorithm for computing the visibility polygon from a point, *J. Algorithms*, 2, pp. 186-197.

H. F. Feng and T. Pavlidis (1975), Decomposition of polygons into simpler components: feature generation for syntactic pattern recognition, *IEEE Trans. on Comp.*, C-24, pp. 636-650.

W. Fichtner, D. J. Rose, and R. E. Bank (1983), Semiconductor device simulation, *SIAM J. Sci. Stat. Comput.*, 4, pp. 391-415.

J. Fukuda and J. Suhara (1972), Automatic mesh generation for finite element analysis, *Advances in Computational Methods in Structural Mechanics and Design*, ed. J. T. Oden, R. W. Clough, and Y. Yamamoto, UAH Press, Huntsville, Alabama.

M. R. Garey, D. S. Johnson, F. P. Preparata, and R. E. Tarjan (1978), Triangulating a simple polygon, *Info. Processing Letters*, 7, pp. 175-179.

J. Kautsky and N. K. Nichols (1980), Equidistributing meshes with constraints, *SIAM J. Sci. Stat. Comput.*, 1, pp. 499-511.

J. M. Keil (1983), Decomposing polygons into simpler components, Ph.D. thesis, Dept. of Computer Science, University of Toronto.

P. Ladeveze and D. Leguillon (1983), Error estimate procedure in the finite element method and applications, *SIAM J. Numer. Anal.*, 20, pp. 485-509.

C. L. Lawson (1977), Software for C^1 surface interpolation, *Mathematical Software III,* ed. J. R. Rice, Academic Press, pp. 161-194.

D. T. Lee (1983), Visibility of a simple polygon, *Computer Vision, Graphics, and Image Processing,* 22, pp. 207-221.

D. T. Lee and F. P. Preparata (1979), An optimal algorithm for finding the kernel of a polygon, *JACM,* 26, pp. 415-421.

D. T. Lee and B. J. Schachter (1979), Two algorithms for constructing a Delaunay triangulation, Rep. 79ASD007, General Electric Co., Daytona Beach, Florida.

B. A. Lewis and J. S. Robinson (1978), Triangulation of planar regions with applications, *Computer Journal,* 21, pp. 324-332.

L. A. Lyusternik (1963), *Convex Figures and Polyhedra,* Dover Publications.

T. Pavlidis (1968), Analysis of set patterns, *Pattern Recognition,* 1, pp. 165-178.

J. K. Reid (1974), Mesh generation, *Finite Element Symposium Proc.,* ed. J. E. Crow, Atlas Computer Lab., Didcot, pp. 129-138.

E. A. Sadek (1980), A scheme for the automatic generation of triangular finite elements, *Int. J. for Num. Meth. in Eng.,* 15, pp. 1813-1822.

B. Schachter (1978), Decomposition of polygons into convex sets, *IEEE Trans. on Comp.,* C-27, pp. 1078-1082.

M. I. Shamos (1975), Geometric complexity, *Proc. of the 7th Annual ACM Symp. on Theory of Computing,* pp. 224-233.

M. I. Shamos and D. Hoey (1975), Closest-point problems, *Proc. 16th Annual Symp. on Foundations of Computer Science,* pp. 151-162.

R. D. Shaw and R. G. Pitchen (1978), Modifications to the Suhara-Fukuda method of network generation, *Int. J. for Num. Meth. in Eng.,* 12, pp. 93-99.

M. S. Shephard, R. H. Gallagher, and J. F. Abel (1980), The synthesis of near-optimum finite element meshes with interactive computer graphics, *Int. J. for Num. Meth. in Eng.,* 16, pp. 1021-1039.

R. Sibson (1978), Locally equiangular triangulations, *Computer Journal,* 21, pp. 243-245.

R. B. Simpson (1979), A survey of two dimensional finite element mesh generation, *Proc. 9th Manitoba Conference on Numerical Math. and Computing,* pp. 49-124.

R. B. Simpson (1981), A two-dimensional mesh verification algorithm, *SIAM J. Sci. Stat. Comput.,* 2, pp. 455-473.

G. Strang and G. J. Fix (1973), *An Analysis of the Finite Element Method,* Prentice Hall.

W. C. Thacker (1980), A brief review of techniques for generating irregular computational grids, *Int. J. for Num. Meth. in Eng.,* 15, pp. 1335-1341.

W. C. Thacker, A. Gonzalez, and G. E. Putland (1980), A method for automating the construction of irregular computational grids for storm surge forecast models, *J. Computational Physics,* 37, pp. 371-387.

J. F. Thompson (1982), *Numerical Grid Generation* (proceedings editor), North Holland.

J. F. Thompson (1983), A survey of grid generation techniques in computational fluid dynamics, *Proc. AIAA 21st Aerospace Sciences Meeting.*

C. M. Williams (1981), Bounded straight-line approximation of digitized planar curves and lines, *Computer Graphics and Image Processing,* 16, pp. 370-381.

M. A. Yerry and M. S. Shephard (1983), A modified quadtree approach to finite element mesh generation, *IEEE Computer Graphics and Applications,* 3, pp. 39-46.