Engineering with Computers (2001) 17: 287–298
© 2001 Springer-Verlag London Limited

Engineering with
Computers

Approximate Shape Quality Mesh Generation

B. Simpson[1], N. Hitschfeld[2,3] and M.-C. Rivara[3]

[1]Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada; [2]Integrated Systems Laboratory, ETH-Zürich, Zürich, Switzerland; [3]Department of Computer Science, University of Chile, Chile

Abstract. *We present two techniques for simplifying the list processing required in standard iterative refinement approaches to shape quality mesh generation. The goal of these techniques is to gain simplicity of programming, efficiency in execution, and robustness of termination. 'Shape quality' for a mesh generation method usually means that, under suitable conditions, a mesh with all angles exceeding a prescribed tolerance is generated. The methods introduced in this paper are truncated versions of such methods. They depend on the shape improvement properties of the terminal-edge LEPP-Delaunay refinement technique; we refer to them as approximate shape quality methods. They are intended for geometry-based preconditioning of coarse initial meshes for subsequent refinement to meet data representation needs. One technique is an algorithm re-organization to avoid maintaining a global list of triangles to be refined. The re-organization uses a recursive triangle processing strategy. Truncating the recursion depth results in an approximate method. Based on this, we argue that the refinement process can be carried out using a static list of the triangles to be refined that can be identified in the initial mesh. Comparisons of approximate to full shape quality meshes are provided.*

Keywords. Quality triangulation; Mesh generation; Pre-conditioning

1. Introduction

In general terms, (planar) quality mesh generation (qmg) has come to mean generating a mesh for a given domain in the plane, possibly including internal features, which meets specified size and triangle shape requirements. These two types of requirements are quite different. The size requirement depends very strongly on the specifics of the application for the target mesh. It is typically variable throughout the given domain and, if the requirement is stated as an upper bound on triangle size measures is relatively easily met by refinement algorithms. The shape requirement is predominantly motivated by the benefits for most applications of having triangles that are roughly equilateral (albeit possibly in some non-Euclidean metric in the case of anisotropic meshes [1]). These motivations are connected to optimality properties and to computational robustness. Hence, the shape criterion has typically been a global lower bound on the minimal angle of each triangle, and it is not usually related to the specifics of an application. It has not been simple to ensure that refinement methods meet this criterion.

The study of how to meet a global shape criterion has been undertaken particularly in computational geometry [2–8]. We will refer to the problem of refining a Constrained Delaunay Triangulation (CDT) on a given Planar Straight Line Graph (PSLG) to meet a minimum angle bound, while respecting the edges of the PSLG as a shape quality mesh generation problem. Algorithms that solve this problem are reviewed below in section 2. Viewed slightly differently, a histogram of shape quality for mesh generated by these methods would show a sharp cut-off at some prescribed minimum quality parameter. One context for shape quality mesh generation problem is as an intermediate step in the task of generating a mesh to serve data representation for an application. In this context, shape quality is, in effect a form of geometry based pre-conditioning for the application driven mesh which is usually much larger. As such, it may not be necessary to have a mesh whose shape quality metric histogram has an absolute cut-off at a prescribed minimum quality parameter; but a mesh with a 'good' shape quality histogram may be good enough.

Using this premise, in this paper we discuss methods that generate meshes that approach good

Correspondence and offprint requests to: B. Simpson, Department of Computer Science, University of Waterloo, 200 Waterloo Avenue W, Waterloo, Ontario N2L 3G1, Canada. E-mail: rbsimpson@uwaterloo.ca

quality but are simpler to implement, more efficient to execute, and have robust termination conditions. The methods exploit the shape improvement properties of the terminal edge LEPP-Delaunay technique, reviewed in Section 5, for selecting the vertex for Delaunay insertion in the refinement process. We discuss two main topics for developing such methods. The more general one describes replacing the iterative refinement organization of conventional qmg methods as discussed in Section 2 by a recursive one as discussed in Section 3. Central to this reorganization is the identification of small edges in the input PSLG. It is then proposed to identify as an approximate method the procedure formed by truncating the recursion at a fixed arbitrary, but small, depth. We argue that, due to the shape improvement properties of the refinement method, the largest gains in shape improvement occur at the smaller levels of refinement.

The second technique involves simplifying the list of triangles to be refined at the top level of recursive refinement. At this top level, the triangles to be refined are all identifiable with small edges in the initial CDT of the PSLG, possibly with some preprocessing required to introduce some additional small constrained edges. In Sections 5 and 6, we show that by identifying these triangles with their small constrained edges, and relaxing the application of the small angle criterion appropriately, we can obtain a simple static list representation of the triangles requiring refinement. The approximate method then makes a single scan down this list, applying recursive refinement to each triangle. A comparison of meshes generated by the approximate and full iterative versions of terminal-edge LEPP Delaunay refinement are presented in Section 6.

2. Iterative Refinement for Shape Quality Mesh Generation

We formulate the shape qmg problem and review some of the features of Delaunay refinement algorithms for it. The region to be meshed is described by a PSLG, $\mathcal{P}$ [2]. This is presented to algorithms via the constrained Delaunay triangulation of $\mathcal{P}$, which we denote $\tau^{(0)}$ ([9, p. 85, 10]). A refinement of $\tau^{(0)}$ is a mesh, τ, on the same region as $\tau^{(0)}$ such that the edges of τ cover those of $\mathcal{P}$. An edge of τ that lies on an edge of $\mathcal{P}$ is a boundary edge. It may be an internal boundary edge (two neighboring triangles in τ,) or an external one. A Delaunay refinement of $\tau^{(0)}$ is a refinement that is a constrained Delaunay triangulation of its boundary edges. By the term 'shape quality mesh for angle tolerance α_{tol}', then, we will mean a Delaunay refinement of $\tau^{(0)}$, $\tau(\alpha_{tol})$, such that the smallest angle in $\tau(\alpha_{tol})$ at a vertex that is not incident on two boundary edges is not less than α_{tol}.

We will refer to:

- a triangle with its smallest angle $<$ than α_{tol} as a bad triangle;
- a triangle with exactly one angle $<$ than α_{tol} as a 1-small-angle triangle;
- a triangle with two angles $<$ than α_{tol} as a 2-small-angle triangle.

The iterative Delaunay refinement approach to shape qmg involves the repeated use of the Delaunay insertion operation to remove each bad triangle t in the current mesh, τ. This operation has two major subtasks: selecting a point, P to be inserted as a new vertex in τ and updating the mesh to include P. The use of powerful techniques for improving the mesh in these two subtasks is crucial to the success of this approach both theoretically and practically.

The second task is the familiar Delaunay vertex insertion to update the old mesh, τ, to a new mesh, τ', which is a Delaunay refinement of τ. Algorithmically, the update can be accomplished by a simple insertion of P followed by a series of edge swaps [5], or equivalently, as the Delaunay kernel operation [9], which involves the explicit removal of the cavity of P in τ, denoted $Cav(P)$[1], and the insertion of the ball of P in τ', denoted $B(P)$[2]. This can be expressed by the formula (the Delaunay kernal [11], p. 55]):

$$\tau' = \tau - Cav(P) + B(P) \qquad (1)$$

This update might be implemented by modifying the memory space of a deleted bad triangle to represent a newly added one. Whatever the implementation is, it will reflect these set operations. We use such set notation and operations to abstract representation dependent details in our following discussion.

An iterative Delaunay refinement algorithm computes a sequence of Delaunay refinements, $\tau^{(n)}$, $n = 0, 1, 2, \ldots$ and it maintains a representation of the set, S, of bad triangles in $\tau^{(n)}$. So, in addition to updating τ the Delaunay refinement algorithm must update S from being the set of bad triangles in τ to S', the set of bad triangles in τ'. We introduce

[1] $Cav(P)$ is the set of triangles in τ with P in their circumcircle.
[2] $B(P)$ is the set of triangles in τ' incident on P.

subsets $S_{del} = S \cap Cav(P)$ and $S_{add} = S' \cap B(P)$, and express this update as

$$S' = S - S_{del} + S_{add} \qquad (2)$$

Iterative Delaunay refinement methods for shape qmg typically take the form of the following *Quality-Mesh-Generation* procedure. It calls a procedure *Delaunay-Insertion*, which takes as input a triangle t to be refined in mesh τ and the minimum angle tolerance. It performs the two tasks of Delaunay insertion referred to above, and returns the two sets S_{add} and S_{del} as well as updating τ:

```
    **Quality-Mesh-Generation** ($\mathcal{P}$, $\alpha_{tol}$)
    Input: a PSLG $\mathcal{P}$ (defined by a set of vertices and edges) and a tolerance parameter $\alpha_{tol}$ ($\alpha_{tol} \leq 30°$)
    initialize $\tau = \tau^{(0)}$
    initialize $S = \{t \in \tau \mid t \text{ is a bad triangle}\}$
for $t$ in $S$ do
    ($S_{add}$,$S_{del}$) = Delaunay-Insertion ($t$, $\tau$,$\alpha_{tol}$)
    $S = S - S_{del} + S_{add}$
    end for
```

An algorithm based on this outline would have to specify:

- some data structure for the set S. This would be a dynamic data structure supporting insertions and deletions as per the set update operation (2);
- the order of access to the elements of S for the **for** loop.

The approximate qmg methods that we describe below permit implementations which can use a small array of size computable from the initial CDT, $\tau^{(0)}$. This array can be processed in a sorted order with no modifications required of it during refinement.

Chew [3] laid the theoretical basis for Delaunay refinement based on circumcircle insertion. Quality meshes are assured with angles between 30° and 120° for some limits on the angles present in the initial CDT, and termination is assured by a lower limit on the sizes of triangle in the mesh. A comprehensive treatment of an iterative form of the shape quality algorithm for this approach was presented by Ruppert [7], as well as the variable size aspect of the algorithm. Termination is rigorously established for $\alpha_{tol} < 20°$ under restrictions on the boundary edge angles. It is observed that termination is to be expected for a wider range of α_{tol}, and preprocessing to handle difficult boundary configurations is discussed. This algorithm development stream culminated in the implementation of Triangle by Shewchuk [8]. This paper includes a discussion of the list structures used to represent the set S of bad triangles, and a discussion of the difficulty in developing robust heuristics for treatment of edges in the initial $\tau^{(0)}$ that meet at small angles.

The Longest Edge Propagation Path (LEPP) technique for identifying insertion points for refinement was initially adapted to the quality mesh generation problem using Delaunay insertion by Rivara and coauthors [4,12,13]. It has subsequently been developed to the form presented as the terminal-edge Delaunay algorithm [5,14]. The terminal-edge LEPP-Delaunay algorithms are guaranteed to produce quality meshes for α_{tol} up to 30° [14], with an adequate preprocessing of infrequent boundary related configurations which appear in the initial CDT. In Section 5, we discuss approximation modifications of the terminal-edge LEPP(t) technique.

3. Approximate Shape Quality Mesh Generation

Mesh generation by refinement generally proceeds from a very coarse initial triangulation of the domain to be meshed (e.g. its CDT), to a large mesh whose triangle sizes and shapes are computed to control the error in some data representation. The initial mesh is essentially a data structure for the geometry of the region. The shape quality mesh generation task we have just described increases the mesh size somewhat to improve the triangle shape solely on the basis of the geometry. It in effect preconditions the mesh for the task of generating a discretization suitable to an application. This geometrically based shape improvement may be interleaved with size distribution motivated refinement, or carried out as a separate preprocessing step. But in any case, the effect is an intermediate one in the total process of producing a final mesh. It seems appropriate, then, to look for simpler and more efficient preconditioning methods, which we are referring to as approximate shape quality mesh generation algorithms.

We turn to formalizing the requirements of approximate qmg.

Definition 1. *A small edge detail* (*sed*) *of the input PSLG* $\mathcal{P}$, *is a directed edge*, $\vec{e}$, *of its CDT*, $\tau^{(0)}$, *with a* 1*-small-angle triangle*, t_{neigh}, *on its right and the small angle of* t_{neigh} *opposite* $\vec{e}$.

Note that an undirected, internal edge of the augmented input PSLG can be a sed for one of its directions, and not for the other; alternatively it could be a sed for both directions. We assume that an arbitrary input CDT has been 'preprocessed', i.e. extended so that it:

(a) contains only 1-small-angle triangles,
(b) the shortest edge of every 1-small-angle triangle is a constrained edge, i.e. belongs to $\mathcal{P}$.

For the theoretical discussion, this means that the CDT of a general PSLG must be preprocessed to refine all its 2-small-angle triangles [15], and any new edges opposite small angles added to $\mathcal{P}$. Some comments on our experience with implementing such preprocessing are given in Section 6. See that 'small' for an edge in this context means relative to some length of its immediate neighbors in $\tau^{(0)}$. It follows trivially from the definition that:

Lemma 1. *A sed is the smallest edge of its defining neighboring triangle.*

A sed is removed by a refinement of τ that results in a modification of its neighboring triangle so that it ceased to be a 1-small-angle triangle.

We note that, at least in principle, a boundary edge that is a sed in one or both of its directions could be split by a refinement that created two smaller seds. This would introduce algorithmic complications, as well as the undesirable effect of making seds even smaller. We discuss in Section 5 how to avoid this situation.

Our approach to approximate quality mesh generation uses a re-organization of the *Quality-Mesh-Generation* outline of Section 2 to incorporate a recursive form of Delaunay insertion that we will designate **Delaunay-Insertion-R. Delaunay-Insertion-R** uses a general procedure **Select-Insertion-Point** (t, τ), which is an abstraction for the method of refinement. It includes the semi-global methods based on LEPP and circumcircle insertion referred to above. **Select-Insertion-Point** takes as input a triangle, t, to be refined that belongs to the mesh, τ. It returns the new vertex. P^*, to be inserted in τ and a triangle, t^*, used to locate P^* in τ for the Delaunay insertion. Its input parameters are unchanged.

Delaunay-Insertion-R also calls procedure **UpDate-Mesh** ($P,t,\tau,S^{cum}{}_{del}$, α_{tol}) to perform a Delaunay insertion (1), and keep track of the changes to the set S of bad triangles in τ. The input to **UpDate-Mesh** is:

- a vertex, P, not in mesh τ,
- a triangle t to locate P in τ,
- a set, $S^{cum}{}_{del}$, of bad triangles no longer in τ,
- the small angle tolerance, α_{tol}.

As its output, **UpDate-Mesh** returns the set S_{add}, as described in Eq. (2). It modifies τ as mentioned, and modifies S_{del}^{cum} by adding the set of bad triangles removed from τ by the update. P, t and α_{tol} are unchanged.

The inputs t, τ, and α_{tol} for **Delaunay-Insertion-R** are as previously described; it also takes a set of bad triangles, S_{del}^{cum}, not in τ. This set is updated by **UpDate-Mesh** to accumulate the triangles removed during a call to **Delaunay-Insertion-R**. When this procedure returns, it has updated S_{del}^{cum} and τ. The superscript *cum* on S_{del}^{cum} is a reminder that this set accumulates bad triangles over all recursive subcalls by the parent procedure **Delaunay-Insertion-R**.

Note that the loop for removing t from the input mesh τ has been incorporated into **Delaunay-Insertion-R**, so, unlike the **Delaunay-Insertion** of Section 2, which inserts one vertex per call, it can insert more than one vertex even without the recursive quality enhancement loop being executed.

```
  Delaunay-Insertion-R ($t$, $S_{del}^{cum}$, $\tau$, $\alpha_{tol}$)
      Note: no $S_{add}$ in parameter list}
  while $t$ is not in $S_{del}^{cum}$ do
    ($P^*$, $t^*$) = Select-Insertion-Point($t$, $\tau$)
    $S_{add}$ = UpDate-Mesh ($P^*$, $t^*$, $\tau$, $S_{del}^{cum}$, $\alpha_{tol}$)
{$S_{add}$ is a temporary set local to Delaunay-Insertion-R}
  for $s$ in $S_{add} - S_{del}^{cum}$ do
  Delaunay-Insertion-R($S_{del}^{cum}$,$\alpha$)
{refer to this as the recursive quality enhancement}
    $S_{add} = S_{add} - \{s\}$
  end for
  end while
return
```

Theorem 1. *On a return from **Delaunay-Insertion-R**, triangle t has been removed from τ and no new bad triangles have been added to τ.*

Proof by induction. There are two looping structures that must terminate if **Delaunay-Insertion-R** is to return. The recursion occurs in the inner **for** loop which we have called the recursive enhancement process. The import of the theorem is then, that if the inner **for** loop terminates all its instances, and the outer **while** terminates, then the procedure returns as stated. The recursion ends only if all calls to **UpDate-Mesh** do not return any bad triangles, (S_{add} = NULL), during the execution of the outer **while**. In this case, a return from **Delaunay-Insertion-R** implies the outer **while** exits with $t \in S_{del}^{cum}$, i.e. τ is as stated in the theorem.

We now assume that the theorem is true for the recursive quality enhancement calls. If the **for** loop terminates, then particular at each pass through the loop **Delaunay-Insertion-R** returns. By the induction assumption, at the bottom of the **for** loop the current triangle s has been remove from and no new bad ones added to it, and S_{add} is reduced in size. The **for** loop terminates, having removed all the bad triangles created by the Delaunay insertion of P^* into τ. Since we are assuming that **Delaunay-Insertion-R**, returns, we must have $t \in S_{del}^{cum}$, and hence τ satisfies the statement of the theorem.

This completes the proof, but we also note that the outer loop, the **while** t remains a bad triangle loop, has been shown to terminate in Rivara and Hitschfeld [5], assuming the inner **for** loop always terminates.

The modified form of **Quality-Mesh-Generation** then looks like this:

```
Quality-Mesh-Generation-R ($\mathcal{P}$, $\alpha_{tol}$)
  Input: a PSLG, $\mathcal{P}$ (defined by a set of vertices and edges) and a tolerance parameter $\alpha_{tol}$
  Construct $\tau$ the CDT of $\mathcal{P}$, including preprocessing
$S = \{t \in \tau \mid t \text{ is a bad triangle}\}$
  for $t$ in $S$ do
  $S_{del}^{cum} = \phi$;
  Delaunay-Insertion-R($\tau$,$t$,$S_{del}^{cum}$,$\alpha_{tol}$) $S = S - S_{del}^{cum}$
  end for
```

The maintenance of set S in this top level module is simplified, since no additions to S take place at this level as indicated by Theorem 1. We note that this organization for **Quality-Mesh-Generation** partially specifies the order of access to the bad triangle set, S.

Theorem 2. *If **Quality-Mesh-Generation-R** terminates, then the resulting mesh meets the shape quality criterion.*

Clearly, the termination of **Quality-Mesh-Generation-R** is tied to the termination of **Quality-Mesh-Generation**; it is only to be expected under the restrictions on α_{tol} and the angles present in the initial mesh, $\tau^{(0)}$. We can get an approximate quality mesh generation algorithm by truncating the recursion at some depth. The intuitive justification for limiting the depth of the recursive quality enhancement is that for a good choice of insertion point, and Delaunay insertion, the quality of the new triangles should be improving with each level of recursion. Evidently,

Theorem 3. *A truncated version of **Quality-Mesh-Generation-R** terminates, for any α_{tol}.*

The price of definitive termination is, of course, that the result of Theorem 2 cannot be expected to extend to a truncated version of **Quality-Mesh-Generation-R**. In fact, while we expect that a truncated version of **Quality-Mesh-Generation-R** would produce a good triangle as the the neighbor of every sed of the initial CDT, $\tau^{(0)}$, this does not seem proveable in a direct way, if at all.

For simplifying implementations, an important feature of **Quality-Mesh-Generation-R** is that no additions to S take place during the processing of triangles in S at the top level of recursion. So, at this level, all the bad triangles to be processed are known from the initial (preprocessed) CDT. Moreover, each bad triangle can be identified with a sed, which lies on a constrained edge of the CDT. The implications of this for simplifying an implementation are described below in Section 6.

We intend the introduction of **Delaunay-Insertion-R** to be seen as primarily a conceptual organization rather than a template for a recursive implementation. Tests with terminal edge LEPP-Delaunay base refinement (described in the next section and in Section 6) indicate that significant shape improvement is already usually achieved with one level of recursion. So an efficient implementation would be to simply unroll the recursion for one level, rather that implementing recursive procedures, literally.

4. Comparing these Methods: An Example

To demonstrate these ideas, we present some meshes generated for the ‘A’ shaped domain distributed with the Triangle quality mesh generation program [8]. Figure 1 shows the initial CDT

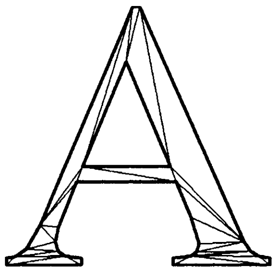
Fig. 1. Initial mesh.

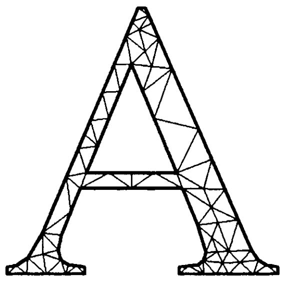

Fig. 2. Iterative refinement (no recursion).

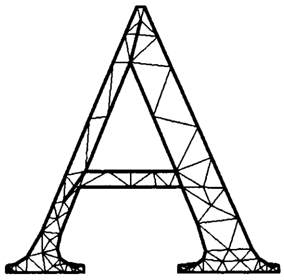

Fig. 3. Approximate shape quality: one recursion level.

for this domain; and Fig. 2 shows a shape quality mesh generated by iterative refinements using the terminal edge LEPP Delaunay method (see Section 5.) The angle tolerance is $\alpha_{tol} = 30°$ for the meshes shown here. Figure 3 and 4 show the approximate shape quality meshes generated by **Quality-Mesh-Generation-R** truncated at 1 and 2 levels of recursion, respectively.

The iterative refinement mesh is generated with the set of bad triangles, S, ordered largest first. We note that the order imposed on the processing of bad triangles by the recursive organization is not largest first. This is likely why there are more triangles in the meshes generated by the truncated recursive methods.

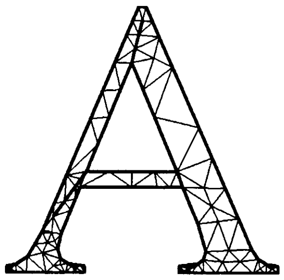

Fig. 4. Approximate shape quality: two recursion level.

5. Simplifying the Use of the Terminal-edge LEPP-Delaunay Method

To present a further simplification, we first present some specifics of the terminal-edge LEPP-Delaunay algorithm [5,14], in the approximate quality mesh generation framework just presented. For any bad quality triangle t, the longest edge propagation path of t, LEPP (t), is found as the finite sequence of increasing neighbor triangles, LEPP(t)=$\{t_k, k = 1, m\}$, where t_j is the neighbor of t_j-1 by the longest edge of t_j-1 for $j = 2, m$ and $t_1 = t$, and the associated terminal-edge is the common longest edge l of both terminal triangles t_m-1 and t_m if this edge exists. In the case that LEPP(t) is an interior sequence of triangles (without involving constrained edges), the midpoint of the terminal-edge is selected for point insertion. Some boundary related edge point insertions must be also considered in the case where LEPP(t) includes constrained edges or vertices. Thus, the terminal-edge Delaunay algorithm can be described as the following Select-Insertion-Point procedure. It calls a procedure named **Risky-Vertex-Boundary-Terminal-Triangle**, which returns *true* or *false* depending on a boundary related configuration of terminal triangles of the LEPP (see elsewhere [5,14] for details).

```
($P^*$,$t^*$) = Select-Insertion-Point($t$,$\tau$)
  Compute the sequence, $\{t_k, k = 1, m\}$, of triangles in LEPP($t$)
  for $k = 1$ to $m$; accept the first instance of do
      case = 1 $\Leftrightarrow$ $t_k$ has a longest edge on the boundary
      case = 2 $\Leftrightarrow$ (not case =1) and largest angle ($t_k$) $\leq 120°$ and second longest edge on the boundary
      case = 3 $\Leftrightarrow$ (not case = 1,2) and $k = m-1$ or $m$ and Risky-Vertex-Boundary-Terminal-Triangle ($t_k$)
      case = 4 $\Leftrightarrow$ (not case = 1,2,3) and $k = m$ i.e. $t_m-1$, $t_m$ are a pair of terminal triangles sharing a terminal-edge
end for

Select insertion vertex as:

Switch (case):
case = 1: $P^*$ = midpoint of longest edge of $t^* = t_k$ – boundary insertion
case = 2: $P^*$ = midpoint of second longest edge of of $t^* = t_k$ – boundary insertion
case = 3: $P^*$ = midpoint of second longest edge of of $t^* = t_k$ – internal insertion
```

case = 4: P^* = midpoint of longest edge of $t^* = t_m$ – internal insertion
end Switch

1. Since this procedure always selects the longest, or second longest, edge of a triangle, and a sed is necessarily the shortest edge of its defining triangle, we have the following:

   Theorem 4. *Approximate quality mesh generation based on* ***Select-Insertion-Point*** *never selects a sed for the directed edge to be refined.*
2. Case 3, in which **Risky-Vertex-Boundary-Terminal-Triangle** is invoked, is required in the terminal-edge method to ensure termination of the iterative refinement algorithm in a case that otherwise can result in fractal refinement and non-convergence. However, if an approximate method is designed by truncating **Delaunay-Insertion-R**, then the method is guaranteed to converge. In this case, the simplification of dropping case 3 is possible.
3. The criterion of case 2 that the largest angle $(t_k) \leq 120°$ is not essential. It is included because it is known [14], that triangles with angles in excess of 120° are modified by the terminal edge Delaunay insertion method automatically.

We now look at a simplification that can be made in the top level of **Quality-Mesh-Generation-R**. If we could be sure that an edge which is a sed in one, or both, of its directions would not be selected for splitting by **Select-Insertion-Point** selection, then the maintenance of the set of current bad triangles), S, at the top level of the recusion would be substantially simplified. Unfortunately, this is **not** guaranteed by Theorem 4.

Figure 5 shows how, in general, processing the defining neighbour of one sed can lead to bisection another. It shows a triangle, t_1, that defines a sed on edge AB. LEPP(t_1) leads to triangle t_a, which is well shaped and has second longest edge BC which is a constrained edge, i.e. an internal boundary edge. The neighbouring triangle, t_b, defines CB as a sed. But case 2 of **Select-Insertion-Point** requires the bisection of BC.

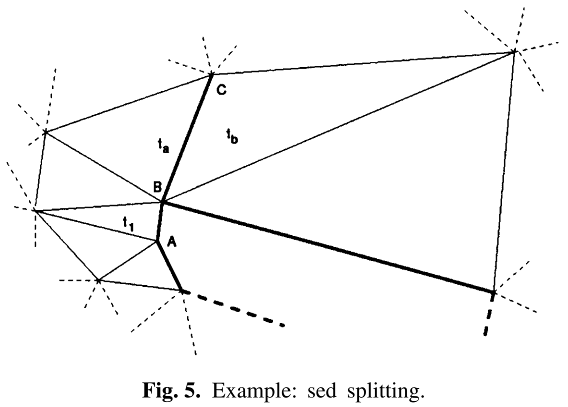

Fig. 5. Example: sed splitting.

We can, however, reduce the likelihood of the processing of one sed in S leading to the splitting of another by processing S on a largest defining triangle first basis. For processing triangle t_1, of longest edge l_{max} (t_1), the shortest edge that can be chosen for bisection has length greater than l_{max} (t_1)/2. This is because:

1. the sizes of longest edges in the sequence LEPP (t_1) are increasing;
2. **Select-Insertion-Point** can select either the longest, or second longest edge for bisection.

We can see qualitatively that if in Fig. 5 the longest edge of t_b were significantly shorter than l_{max} (t_1), then edge CB could not be the second longest edge of a triangle in LEPP (t_1). This leads us to the common idea to access S by a largest defining triangle first ordering.

To be more quantitative about this, we need to develop a limit on the maximum size of a sed in terms of the longest edge of its defining triangle and the angle tolerance, α_{tol}. This we do in Theorem 5:

Theorem 5. *Let t be a* 1*-small-angled triangle with longest edge of length* l_{max}. *Let its shortest edge have length* l_{min} *and be opposite angle* θ_{min}. *Then*

1. $l_{min} \leq l_{max}/\cos(\alpha_{tol}) + \text{cotan}\,(\theta_{min})\sin(\alpha_{tol}))$ (3)
2. *Let* f^* *be the (smallest positive) solution of*
$\cos(\alpha_{tol}) + \text{cotan}\,(f^*\alpha_{tol})\sin(\alpha_{tol}) = 2$ (4)
and let $\sigma(\alpha_{tol}) = \sin(\alpha_{tol})/(2 - \cos(\alpha_{tol}))$. *Then* f^* *depends on* α_{tol} *according to*

$$f^*(\alpha_{tol}) = \arctan^*\sigma(\alpha_{tol}))/\alpha_{tol} \approx 1/(1 + \alpha_{tol}^2/2) \quad (5)$$

3. *If* $\theta_{min} \leq f^*\,(\alpha_{tol})\alpha_{tol}$ *then*

$$l_{min} \leq l_{max}/2 \quad (6)$$

Proof (Part 1). Let the intermediate and maximal angles of t be θ_{int} and θ_{max}, respectively, and let l_{max} be the length of the longest edge. Then $\theta_{int} \geq \alpha_{tol}$, and

$$\theta_{max} = \pi - (\theta_{int} + \theta_{min}) \geq \pi - (\alpha_{tol} + \theta_{min}) \quad (7)$$

The sine law for triangles and Eq. (7) indicate that

$$l_{min} = (\sin(\theta_{min})/\sin(\theta_{max}))l_{max} \quad (8)$$

$$\leq (\sin(\theta_{min})/\sin(\theta_{min} + \alpha_{tol}))l_{max} \quad (9)$$

But

$$\sin(\theta_{min} + \alpha_{tol})/\sin(\theta_{min}) \quad (10)$$
$$= [\sin(\theta_{min})\cos(\alpha_{tol}) + \cos(\theta_{min})\sin(\alpha_{tol})]/\sin(\theta_{min})$$

from which Eq. (3) follows.

Proof (Part 2). Let the vertices of t be labeled ABC, with AB being the longest edge, of length l_{max}, and BC the shortest edge, of length l_{min}. Figure 6 shows the edge AB of t lying along the positive x-axis (i.e. in longest edge coordinates [1].) This figure shows as dashed lines the region in which C can lie if t is a 1-small-angle triangle.

The point Y is located at $x = l_{max}/2$ such that angle YAB $= \alpha_{tol}$. Since the smallest angle, θ_{min}, is at A and is less than α_{tol}, C must lie below AY. The angle ABY is also equal to α_{tol}. Since t has only one small angle, C must lie above BY. The dashed circular arc through B is part of the circle of radius, l_{max} centered on A. Since AC is shorter than l_{max}, C must lie inside this circle. So C must lie in the sector XYB.

The dotted circle of radius centered on B belongs to the circle of radius $l_{max}/2$; evidently, $l_{min} < l_{max}/2$ if C lies inside this circle. However, the diagram shows a small shaded region in which C can lie and not be inside this circle. The bottom right corner of this shaded region is denoted Z. We define $f^*(\alpha_{tol})$ by

$$f^*(\alpha_{tol})\alpha_{tol} = \text{angle ZAB} \quad (11)$$

If $\theta_{min} < f^*(\alpha_{tol})\alpha_{tol}$, then C lies inside the dotted circle, which establishes Eq. (6). To establish Eq. (4), see that

$$|B - Z| = l_{max}/2 \quad (12)$$
$$= l_{max}\sin(f^*(\alpha_{tol})\alpha_{tol})/\sin(\theta_{max})$$

If we replace θ_{min} in Eq. (10) by $f^*(\alpha_{tol})\alpha_{tol}$, we get (4).

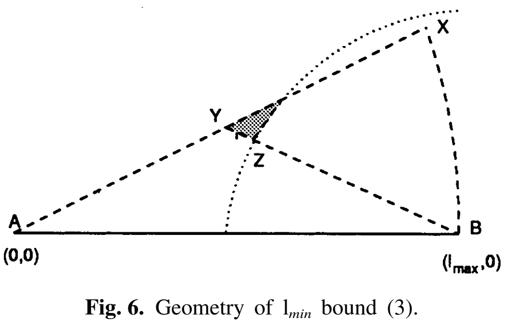

Fig. 6. Geometry of l_{min} bound (3).

The implication of Theorem 5 for the largest first ordering of S for an approx qmg method using LEPP-Delaunay selection is derived from Theorem 6.

Theorem 6. *Let t be the triangle defining the largest sed in S.* **Select-Insertion-Point** *can only select P on an edge that has a sed associated with one of its directions if*

1. *It terminates in case* 2 *of* ***Select-Insertion-Point****, and*
2. *This edge is an internal boundary edge, and, for one of its neighbours*

$$\theta_{min}(t) > f^*(\alpha_{tol})\alpha_{tol} \quad (13)$$

Proof. Select-Insertion-Point will terminate with a triangle t^* and point P that is the midpoint of one of its edges. Let us call this edge the insertion edge and let $\overrightarrow{e}(t^*)$ be its directed edge on which t^* is its neighbor. In cases 1, 3, and 4. **Select-Insertion Point** either selects an internal edge of or a boundary edge of length $= l_{max}(t^*) \geq l_{max}(t)$. In any case, the insertion edge cannot be associated with a sed.

Assume that the selection process terminates in case 2, and that the insertion edge is an internal boundary edge. Let t' be the triangle neighboring on this edge with the opposite direction, i.e. $-\overrightarrow{e}(t^*) = \overrightarrow{e}(t')$. Then

$$l_{min}(t') > l_{max}(t^*)/2 \geq l_{max}(t)/2 \quad (14)$$

By Lemma 1, $\overrightarrow{e}(t^*)$ cannot be a sed. Let us assume that t' is a 1-small-angle triangle, so that $\overrightarrow{e}(t')$ is a sed. Unless $\theta_{min}(t') > f^* \alpha_{tol}$,

$$l_{min}(t') \leq l_{max}(t')/2 \leq l_{max}(t)/2 \quad (15)$$

so $l_{min}(t')$ cannot be processed as a result of LEPP-Delaunay refinement of t.

We would like to have the simplicity of an approximate qmg method which does not split existing seds in S. Theorem 6 shows that the approx qmg method with LEPP-Delaunay point selection can split seds, but only under specific circumstances that are unlikely to be common. Note that the approximation, $f^*(\alpha_{tol}) \approx 1/(1 + \alpha_{tol}^2/2)$ and (13) show that for a bad triangle t to meet these conditions, it must be very close to meeting the angle criterion, at least for small α_{tol}. In fact, for $\alpha_{tol} = 30° = \pi/6$ radians, $1/(1 + \alpha_{tol}^2)/2 = 0.88$, while the exact value of $f^*(\alpha_{tol})$ is 0.80. So the simplest way to gain this simplicity would seem to be to extend the approximate nature of the method by removing the sed that would be bisected from S (but still bisect it as a boundary edge), i.e. referring to Fig. 5, to make t_b an acceptable triangle of the approximate quality mesh, in this case. This has the effect of possibly allowing a triangle that does not meet the angle tolerance as a neighbor of a boundary edge, in some evidently rare circumstances.

6. Implementation and Some Comparisons

We present a comparison between meshes generated by a standard iterative quality mesh generation program organization and those generated by an approximate shape quality mesh generation program based on truncated recursion (Section 3), and a sed based list (Section 5). Both programs use the same vertex selection technique, (terminal-edge LEPP) and use a CDT Delaunay insertion mesh update. The angle tolerance used in all cases is $angTol = 30°$. Consequently, the comparison differences are due to the simplified list processing carried out in the approximate shape qmg program, which we detail below. The iterative implementation is implemented as outlined in **Quality-Mesh-Generation** of Section 2. It uses a dynamic list for the bad triangle set which it repeatedly scans using a heuristic to select the larger triangles first.

In the *sed-based recursive implementation*, triangles are refined by a version of recursive Delaunay refinement as outlined in **Delaunay-Insertion-R** of Section 3, truncated at a preset maximum recursion level. This level is set to one in most of our tests. The program does not keep a list of bad triangles directly. Instead, prior to commencing the refinement process, it builds an array of seds present in the initial (preprocessed) mesh, sorted by decreasing length. We will refer to it as the sed list. The program carries out a single scan of the sed list, processing the (current) neighbouring triangle of each sed by **Delaunay-Insertion-R**.

The point of listing seds instead of bad triangles is that the program can act on a static list. The Delaunay insertions that occur when one bad triangle is being processed typically modify other bad triangles. In terms of the set, S, of bad triangles, a modification of a bad triangle is its deletion from the set and, possibly, the addition of another; this is what necessitates dynamic list representations for S. Because seds are constrained edges of the CDT of the mesh, they are not modified by the Delaunay insertions. Moreover, the arguments of Section 5 show that a sed is not selected for bisection in the refinement process of a triangle connected to another sed of greater size. Although the neighbouring triangle of a sed may be modified during the processing of a sed earlier in the sed list, the sed itself is unchanged. Since triangle modifications by the terminal edge LEPP-Delaunay insertion are shape improving, the new neighbour will have a bigger smallest angle. In fact, it may no longer be a bad triangle by the time the associated sed is reached in the sed list scan. In the set terminology of **Quality-Mesh-Generation-R** of Section 3, the bad triangle that originally defined the sed appears in S_{del}^{cum} for the refinement of an earlier triangle.

Recall that before building the representation of the bad triangle list in **Quality-Mesh-Generation-R**, we indicated that the initial $\tau^{(0)}$ should be preprocessed to ensure that each bad triangle is associated with some sed. We found that our results for the approximate qmg technique were sensitive to the choice of 2-small-angle triangles to preprocess. If one simply preprocesses all of them in the initial mesh, then for some of the test cases, the method generated an unnecessarily large number of triangles. We now restrict the preprocessing to triangles with maximum angle $< 120°$. The terminal edge LEPP-Delaunay insertion process automatically modifies triangles with larger maximum angles; see Rivara and Hitschfeld [5]. These other 2-small-angle triangles are treated as one-small-angle triangles using the sed opposite the smallest angle.

The figures for each case presented come in vertically displayed pairs (with one exception). One figure of the pair displays the mesh generated by either the full iterative shape qmg method or by the approximate shape qmg method. The other figure of the pair shows a histogram of the common shape quality metric, the ratio of the longest edge to the circumradius, normalized to 1 for an equilateral triangle. This metric is denoted Q_K in George and Borouchaki [6].

Figure 7 shows the mesh generated by the interative shape qmg program for a ‘fat comma’

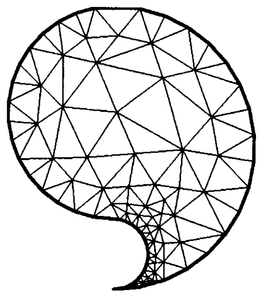
Fig. 7. Iterative Refinement (no recursion) cursion.

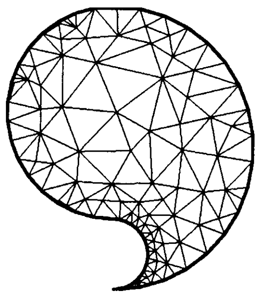

Fig. 8. One level of recursion.

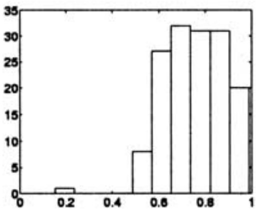

Fig. 9. Iterative Refinement shape quality.

shape. Figure 9 shows the histogram of associated triangle shape metrics. The single poorly shaped triangle in evidence in the histogram is located at the tip of the tail of the comma. The histogram of Q_k for the mesh generated by the approximate method, shown in Fig. 10, is generally quite good, but the distribution shows a tail of five triangles in the 0.15 to 0.4 range

The next test case has a simple geometry that belies its difficulty as a test for shape quality mesh generation. The initial PSLG has a square external boundary and a pair of isolated line segments that meet in the center at a small angle, and are significantly different in length. They can be seen as the heavier line segments in the center of Fig. 11. This configuration was used by Shewchuk [8] in a discussion of the difficulty of ensuring termination of algorithms for isolating small angles of the PSLG as part of iterative shape qmg algorithms.

Figure 11 shows the mesh generated by the iterative qmg program, and Fig. 13 shows the corresponding histogram of shape quality triangle metrics. The program used a reasonably successful heuristic to handle the small internal angles of the $r^{(0)}$. Figures 12 and 14 shows the corresponding plots for the approximate shape qmg program with two levels of recursion. This is our most challenging test for the approximation technique; there are 10 triangles below the 0.4 quality cut-off in the interative distribution and this is an improvement over the mesh generated by one level of recursion. There are two test cases that we have studied in which there is significant additional refinement incurred by a second level of recursion: the A shape shown in Section 3 is the other one. Notice that even with two levels of refinement, substantially fewer triangles are generated by the approximate qmg method in this case. For iterative refinement, heuristics are required to prevent the base method from producing too many triangles in this case, or even not terminating. It appears that the approximate method has the opposite difficulty; without some special treatment of this configuration, it terminates prematurely.

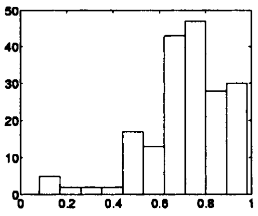

Fig. 10. Approx shape QMG shape quality.

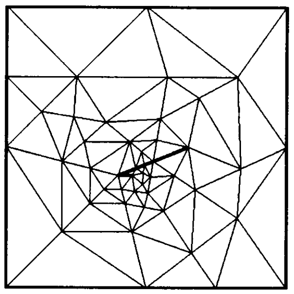

Fig. 11. Iterative refinement.

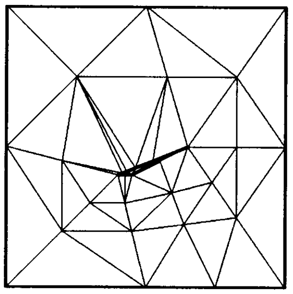

Fig. 12. Two levels of recursion.

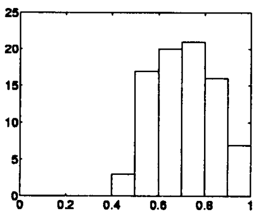

Fig. 13. Triangle quality histogram.

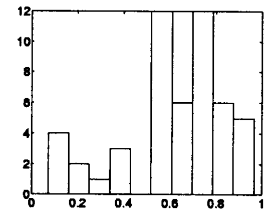

Fig. 14. Triangle quality histogram.

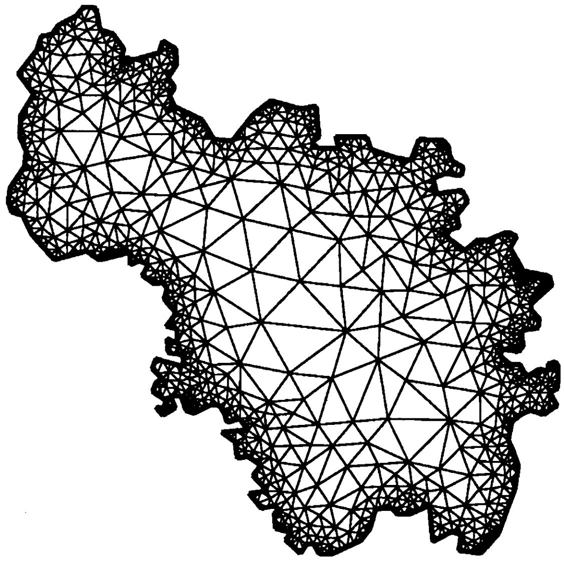

Fig. 15. Iterative refinement.

We also tested a large complicated polygonal region of geographical data that is the outline of a watershed. The polygonal boundary of this domain consists of 536 vertices. Since the mesh is quite large, only the one generated by the iterative shape qmg program is shown in Fig. 15. In Fig. 16, we show the histogram of triangle quality for the 2204 triangles in the mesh generated by the iterative shape qmg program. In Fig. 17, we show the corresponding histogram for the 2882 triangles in the mesh generated by the approximate qmg method with one level of recursion. A small tail of lower quality triangles are again in evidence in the shape quality metric histogram of the approximate qmg

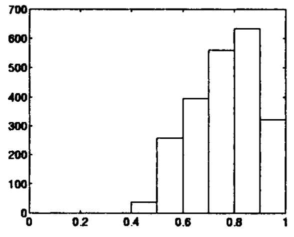

Fig. 16. Iterative refinement, triangle quality.

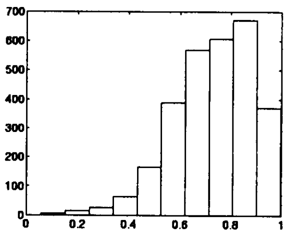

Fig. 17. One level of recursion, triangle quality.

based mesh. But the approximate method produces a perhaps surprisingly good mesh for this relatively complicated domain. It generated about 25% more triangles, however.

The recursive refinement approximate shape qmg method quite generally produces more triangles that the iterative refinement method; 150 versus 248 for the 'fat comma', 88 versus 129 for the A shape shown in Fig. 2. The reverse is true for the second test case above, the square with segment meeting as a small angle in the centre; for this case the ratio is 88 (iterative) to 51. Since our intended goal for the recursive refinement approximate shape qmg method is to produce initial triangulations for further size refinement, these somewhat larger meshes do not seem to be a disadvantage.

7. Conclusions and Further Investigations

We have discussed how some ideas of approximate shape quality mesh generation can simplify the maintenance of the bad triangles set, S, of Delaunay refinement methods. Iterative refinement forms of qmg as typified by **Quality-Mesh-Generation** of Section 2 use a global list of bad triangles in the current mesh that is managed in the top level of the modularization. This is a dynamic data structure that supports insertions and deletions as well as access for selection and modification.

In the recursive organization described in Section 3, the set of bad triangles is distributed into the modularization. The growth of the set is handled in local sets of the recursive **Delaunay-Insert-R** procedure. These sets are small, which makes their list representation processing more efficient. The part of the set handled at the top level does not grow. However, in general its entries may require modification due to the action of **Delaunay-Insert-R**. An approximate method constructed by limiting the levels of recursion is then guaranteed to terminate, since the growth of S has been curtailed.

In Section 5, a further simplification is described that uses specific details of the terminal-edge point selection scheme of the LEPP-Delaunay approach to triangle shape improvement. Using it in conjunction with the recursive organization of refinement an approximate qmg method can be designed that uses a small global array of size computable in advance of refinement plus distributed local storage. This array can be processed in sorted order with no modifications of it required during the refinement.

An important extension of this investigation is to identify how best to incorporate these approximate shape quality techniques with a data representation driven size requirement for full mesh generation. It seems likely to us that the shape criterion can simply be dropped after the pre-conditioning of the initial $\pi^{(0)}$ by an approximate shape quality method. Modified terminal-edge LEPP algorithms for the approximate size and shape qmg problem can be envisaged. Another intriguing challenge is to identify how this idea of simplifying methods through judicious approximations could be used to advantage for space meshing. Formally, the techniques applied in this paper apply directly to three space.

Acknowledgements

The contributions of the first author have been supported by the Natural Science and Engineering Research Council of Canada and the ones of the second and third authors have been partially supported by Fondecyt 1981033

References

1. Simpson, R. B. (1994) Anisotropic mesh transformations and optimal error control. Appl Num Math, 14, 183–198
2. Bern, M., Eppstein, D. (1992) Mesh generation and optimal triangulation. In: F.K. Huang, (editor) Computing in Euclidean Geometry. World Scientific
3. Chew, L. P. (1993) Guaranteed-quality mesh generation for curved surfaces. 9th Annual Symposium on Comp Geometry, San Diego, CA, 274–280
4. Rivara, M. C. (1997) New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations. Int J Num Methods, 40, 3313–3324
5. Rivara, M. C., Hitschfeld, N. (1999) LEPP-Delaunay algorithm: a robust tool for producing size-optimal quality triangulations. Proc 8th Int Meshing Roundtable, 205–220
6. Rivara, M. C., Inostroza, P. (1997) Using longestside bisection techniques for the automatic refinement of delaunay triangulations. Int J Num Meth Eng, 40, 581–597
7. Ruppert, J. (1995) A Delaunay refinement algorithm for quality 2-dimensional mesh generation. J Algorithms, 18, 548–585
8. Shewchuk, J. R. (1996) Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. First Workshop on Applied Computational Geometry. Philadelphia, PN, 124–133
9. George, P. L., Borouchaki, H. (1998) Delaunay Triangulation and Meshing. Hermes
10. Chew, L. P. (1994) Constrained delaunay triangulation. Algorithmica, 4, 97–108
11. Borouchaki, H., George, P. L. (1997) Aspects of 2-d delaunay mesh generation. Int J Num Meth Eng, 40, 1957–1975
12. Rivara, M. C. (1996) New mathematical tools and techniques for the refinement and/or improvement of unstructured triangulations. Proc 5th Int Meshing Roundtable, Pittsburgh, PN, 77–86
13. Rivara, M. C., Palma, M. (1997) New LEPP algorithms for quality polygon and volume triangulation: Implementation issues and practical benhavier. In: S. A. Cannan and S. Saigal (editors), Trends Unstructured Mesh Generation. ASME, 1–8
14. Rivara, M. C., Hitschfeld, N., Simpson, R. B. (2001) Terminal edges Delaunay (small angle based) algorithm for the quality triangulation problem. Computer-Aided Design 33, 263–277
15. Joe, B., Simpson, R. B. (1986) Triangular meshes for regions of complicated shape. Int J Num Meth Eng, 23, 751–778