COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING, Vol. 9, 121–129 (1993)

ADAPTIVE MESH GENERATION AND TRANSFORMATION OF TRIANGULAR TO QUADRILATERAL MESHES

ERNST RANK, MANFRED SCHWEINGRUBER AND MARKUS SOMMER

Numerische Methoden und Informationsverarbeitung, FB Bauwesen, Universität Dortmund, Postfach 500 500, D-4600 Dortmund 50, Germany

SUMMARY

In this paper an algorithm for generating pure triangular or pure quadrilateral meshes for arbitrary polygonal domains is presented. The first part shows a triangulation procedure being as suitable for local mesh refinement as for local mesh coarsening. In the second part an algorithm for transformation of triangular grids to quadrilateral meshes of the same relative mesh density is introduced. Two simple strategies are shown which can easily be implemented into any existing triangulation algorithm.

1. INTRODUCTION

Mesh generation has been a topic of research since the beginning of the practical use of finite-element methods. Following the excellent survey paper of Ho–Le,[1] most mesh generators can be classified into two groups. In the *macro-element technique* the user has to divide the structure into simple basic regions, which are then meshed automatically into a more or less regular element pattern. These basic regions are described most frequently by 3- to 8-node isoparametric mappings. One main advantage of these mapping methods is the possibility to easily generate purely quadrilateral meshes – a favourable result, as 4-noded elements show very often higher accuracy than the corresponding triangular elements. The big disadvantage of macro-element meshing is that it is nearly impossible to generate strongly graded meshes on arbitrary domains.

In a completely different approach *free-meshing techniques* are capable to generate automatically unstructured grids for arbitrary polygonal domains. Various algorithms (e.g. References 2 and 3 and references listed there) have been published which use mesh density functions to generate strongly graded meshes necessary for adaptive finite-element computations. The main disadvantage of these codes is that they all result in triangular meshes. Until now, there have been only very few publications for pure quadrilateral mesh generation.[4,5] A different but obvious idea is the transformation of triangular grids into pure quadrilateral meshes. Recently, Johnston *et al.*[6] published an algorithm working for arbitrary domains and graded meshes. A similar successful approach has been reported in Reference 7.

We present in this paper a different, very simple idea, robust and easy to implement, resulting in excellent meshes even for rapidly varying mesh density functions. In the following Section we introduce our general concept and the triangular meshing. The conversion from triangular to quadrilateral meshes is explained in Section 3, followed by various examples in Section 4.

0748–8025/93/020121–09$09.50

Received 23 September 1991
Revised 13 February 1992

2. GRADED TRIANGULAR MESHING

Let a domain be described by a union of simply connected, non-overlapping regions, each defined by a polygonal boundary. To be able to introduce a function for the desired mesh density $d(x)$ let us assume that an initial triangular mesh M_1 with a node list N_1 has already been defined. This first finite-element mesh is created assuming a constant mesh density function over the whole domain. Since this is only a special case of an arbitrary continuous mesh density we do not need to consider this initial stage separately.

Let now $d(x)$ be defined by its nodal values, assuming an interpolation with linear triangular shape functions within each element of M_1.$d(x)$ may result from an *a posteriori* error estimation after a previous finite-element computation on M_1 as well as from *a priori* criteria or from user interaction. Obviously $d(x)$ is equivalent to a function $h(x)$ defining the local distance of nodes at point x. A possible quality measure for good triangular meshes is the ratio of diameters of adjacent elements. This ratio shall be limited by a user-definable number *RATIO*. If this limit conflicts with the density function, then $d(x)$, respectively $h(x)$, has to be smoothed. This is done by the following algorithm:

Step 1: Sort all nodes according to their local distance function.
The smallest value gets the first position in LIST.

WHILE LIST is not empty

Step 2: Take the first Node n_i from LIST
Step 3: *FOR EACH* neighbour n_j of n_i:
Step 3a: Compute distance l_j of n_i and n_j

Step 3b: Compute $k = \text{int}\left[\dfrac{\log(l_j*(RATIO-1))}{\log\ RATIO} - 1\right]$

Step 3c: IF($RATIO^k*h_i < h_j$)*THEN*

- Remove n_j from LIST and set $h_j = RATIO^k*h_i$,
- Insert n_j into LIST according to the new local distance function h_j

END IF
END WHILE

With the smoothed density function the triangulation of the domain can start. As our basic concept we use the idea of *recursive region splitting* which was published, for example, in Reference 8 or 9. The basic principle is shown for a domain with one region in Figure 1.

First, nodes are created according to the density function on the boundary of the region. Then the region is divided recursively into smaller domains. Dissection lines are chosen so that the resulting angles are close to 60 degrees. On each dissection line nodes are created in a distance suggested by the local distance function. Note that mesh M_1 and node list N_1 is used only for the definition of $d(x)$ and $h(x)$ and that the newly created node list N_2 of mesh M_2 does not necessarily include N_1. Thus it is easily possible to create meshes M_2 which are *locally coarser* than M_1. This is especially important for transient computations where the mesh density has to vary over time.

Recursive region splitting results in simple standard regions with three or four nodes. Finally quadrilaterals are divided along their shorter diagonal. A following optimisation of the mesh is performed by a nodal relaxation procedure. As our goal is not to generate uniform but graded meshes, the simple relaxation method by standard Laplace smoothing is not adequate.

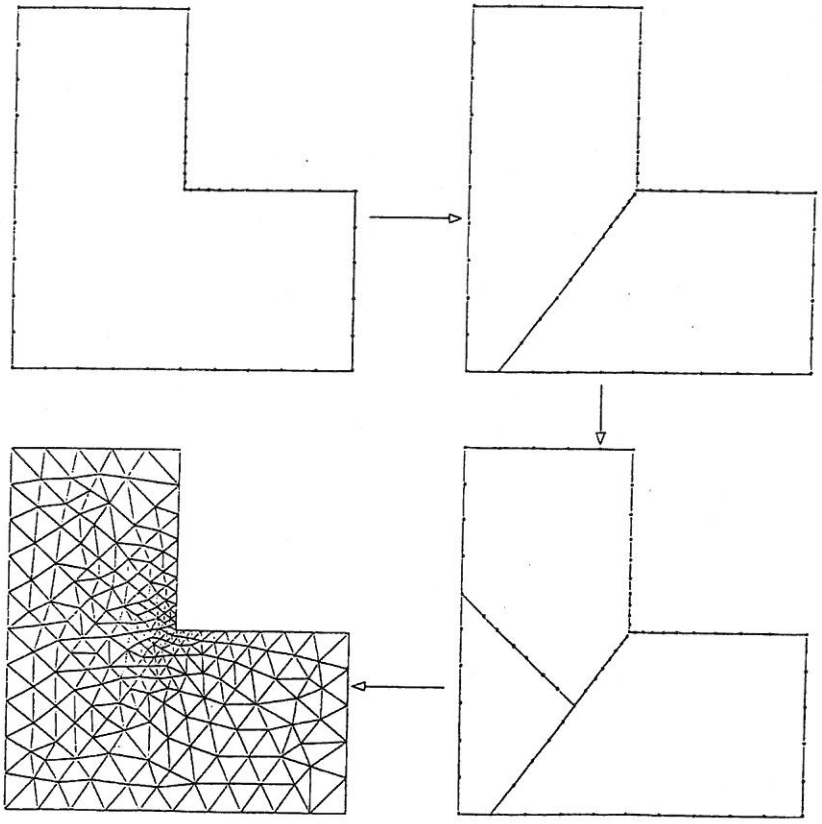

Figure 1. Triangulation procedure

We have to consider the desired mesh density at each node. To explain our relaxation procedure, consider an element patch around a node i with neighbouring nodes $i_1, \ldots, i_k$. The nodal distance function (i.e. the *desired* distance to its neighbours) shall be denoted by h_i, and $h_{i1}, \ldots, h_{ik}$, respectively. For each edge $e_{i1}, \ldots, e_{ik}$ from node i to its neighbours we can define a mean desired distance as

$$\bar{h}_{ij} = \frac{h_i + h_j}{2} \tag{1}$$

Let the relative co-ordinates of $i_1, \ldots, i_k$ with respect to node i be given by $(x_{i1}, y_{i1}) \ldots (x_{ik}, y_{ik})$ and let $l_{i1}, \ldots, l_{ik}$ be the *true* distance to the neighbours. Then the 'relaxed' co-ordinates of node i are given by the weighted centre of gravity

$$\begin{aligned} x_i &= \left(\sum_{j=1}^{k} \frac{x_{ij} * e_{ij}}{\bar{h}_{ij}} \right) \Big/ \left(\sum_{j=1}^{k} \frac{e_{ij}}{\bar{h}_{ij}} \right) \\ y_i &= \left(\sum_{j=1}^{k} \frac{y_{ij} * e_{ij}}{\bar{h}_{ij}} \right) \Big/ \left(\sum_{j=1}^{k} \frac{e_{ij}}{\bar{h}_{ij}} \right) \end{aligned} \tag{2}$$

Our numerical experiences show that three relaxation sweeps are enough for sufficiently smoothing the mesh.

3. TRANSFORMATION TO A QUADRILATERAL MESH

We will present two strategies to transform triangles into quadrilaterals. Both are extremely easy but can be combined to a simple and efficient algorithm. As shown in Figure 2 every triangle can be split into three quadrilaterals, connecting points on the sides of the triangle with an interior point. Usually the mid-points of the sides and the centre of gravity are good choices for these new nodes. Obviously it is possible to transform any triangular into a quadrilateral mesh only by this strategy, yet resulting element shapes are odd, as shown in Figure 3.

A different idea has been used in Reference 6 where neighbouring triangles are combined to quadrilaterals, as shown in Figure 4 for the same basic triangular mesh. Obviously this strategy

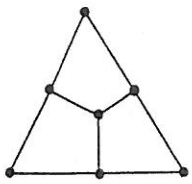

Figure 2. Triangle split into three quadrilaterals

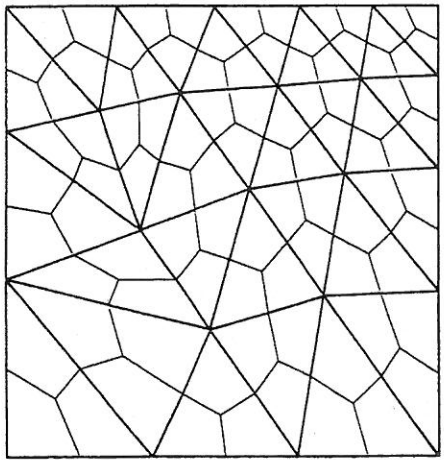

Figure 3. Quadrilateral mesh derived from triangular mesh (thick lines) by strategy 1

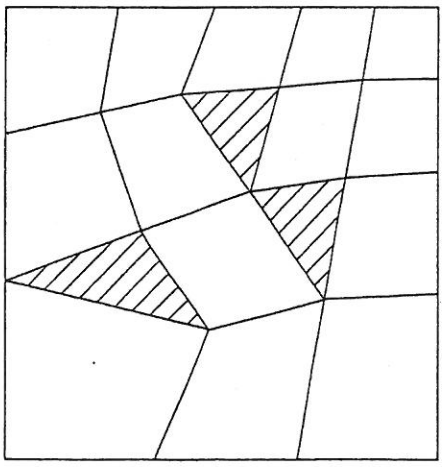

Figure 4. Quadrilateral mesh by combination of neighbouring triangles

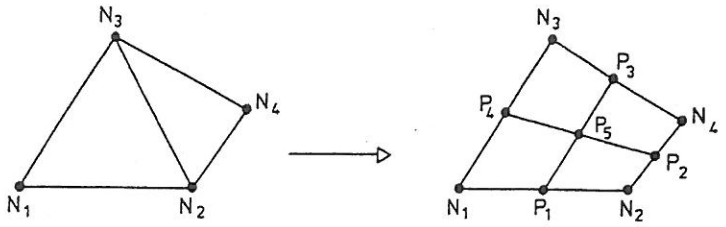

Figure 5. Quadrilaterals created by splitting of two neighbouring triangles

cannot create a pure quadrilateral mesh for an odd number of triangles, and an even number may produce 'islands' of isolated triangles. These islands are resolved by edge-flipping techniques or by introduction of 'split propagation',[6] which often have to be defined up to the boundary of the domain.

Instead of only *combining* two neighbouring triangles, *our* second strategy is to *split* two adjacent triangles into *four* quadrilaterals as shown in Figure 5. If mid-side points are chosen for P_1 to P_4, a compatible quadrilateral mesh will be generated except for possible islands (Figure 6). But these isolated triangles can now be split into three quadrilaterals, as demonstrated at the beginning of this Section. The important difference to the algorithm of Reference 6 is the *strictly local* resolution of triangular islands. This can be performed very fast in a computer program and does not disturb element shapes of already generated quadrilaterals.

A question still to be answered is the *sequence* of combination of triangles to quadrilaterals. Since our goal is to construct meshes which are as 'smooth' as possible we can define quality measures for resulting elements and introduce heuristic strategies for the combination sequence in order to meet these criteria as well as possible. One possible measure is the ratio of the radii of inscribed and circumscribed circles, and another one the difference between largest and smallest interior angle. Before actually combining neighbouring triangles to quadrilaterals, the quality criterion is computed for every possible pair of triangles and stored in a list. This list (with length less than $3 \times$ Number of triangles) is then sorted with respect to the quality criterion and actual element pairs are created in the sequence of their position in the list.

A final mesh optimization is performed by a nodal relaxation similar to that described for

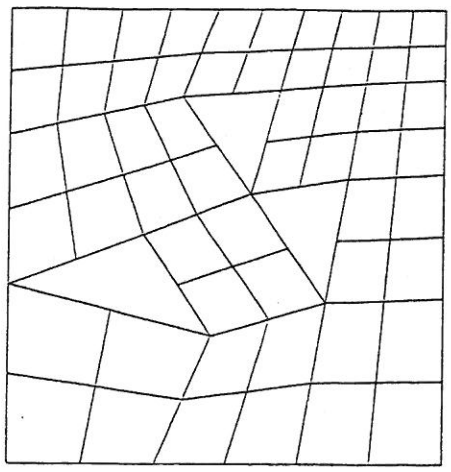

Figure 6. Triangle islands in quadrilateral mesh

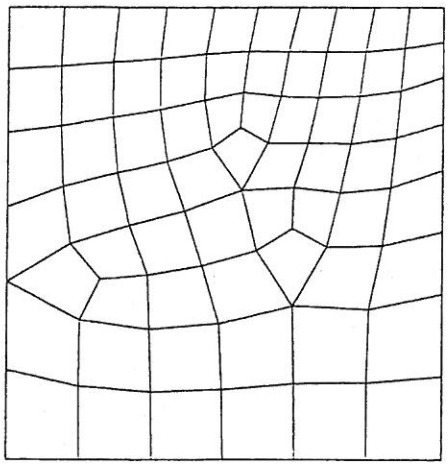

Figure 7. Final mesh after relaxation

triangular meshes in Section 2. Figure 7 shows the relaxed mesh after using as quality criterion the difference of interior angles.

4. NUMERICAL EXAMPLES

In our first example a sequence of structures and meshes is shown, generated in the course of an adaptive finite-element computation. Figure 8 shows a cross-section of a bridge, the defining contours being extracted from CAD data of the structure. In Figure 9 the first triangular and

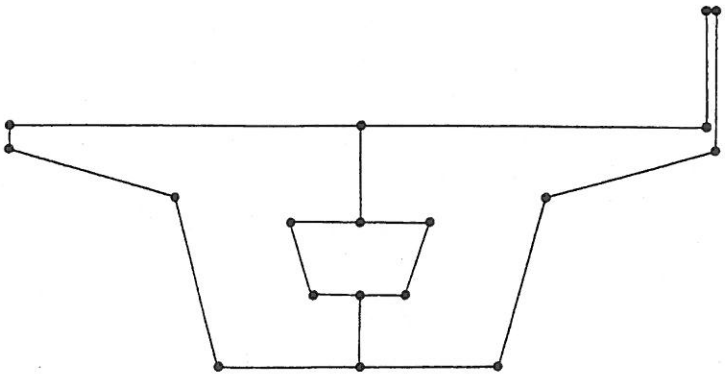

Figure 8. Defining contours of a cross-section of a bridge

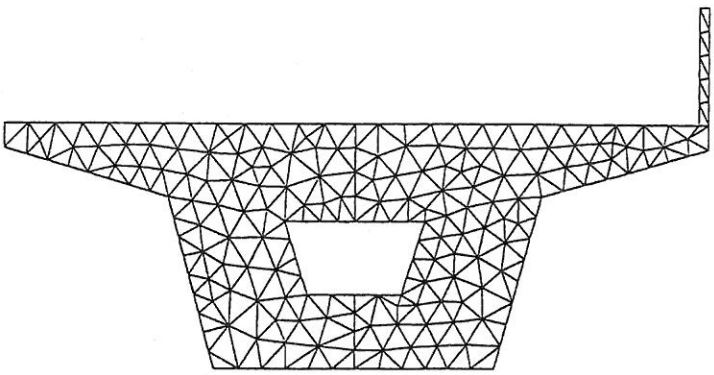

Figure 9. Triangular mesh with uniform density function

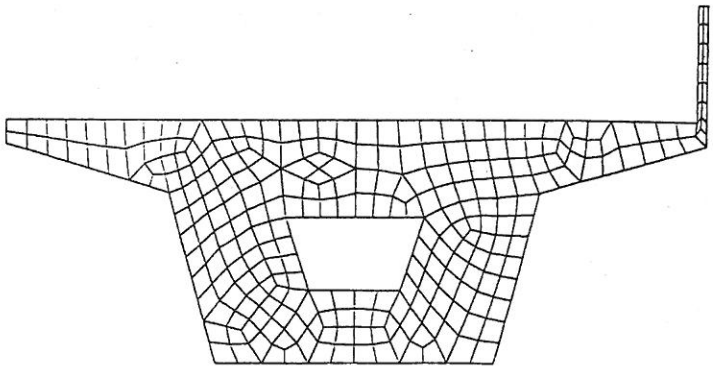

Figure 10. Quadrilateral mesh with uniform density function

in Figure 10 the first quadrilateral mesh with a uniform mesh density function is plotted. The difference between interior angles is used as a quality criterion for the combination of triangles to quadrilaterals.

For the finite-element computation linear elasticity and plain strain conditions are assumed and the structure is loaded by a combination of self-weight and wind load from the right. After the first finite-element computation, error indicators are computed following the ideas of Zienkiewicz and Zhu.[10] Figure 11 shows the mesh density function derived from the distribution of error indicators. For the computation of this mesh density function the triangular discretisation was used.

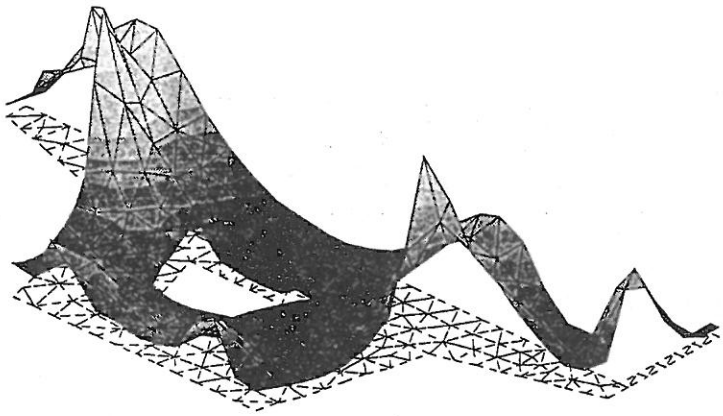

Figure 11. Density function derived from error indicators

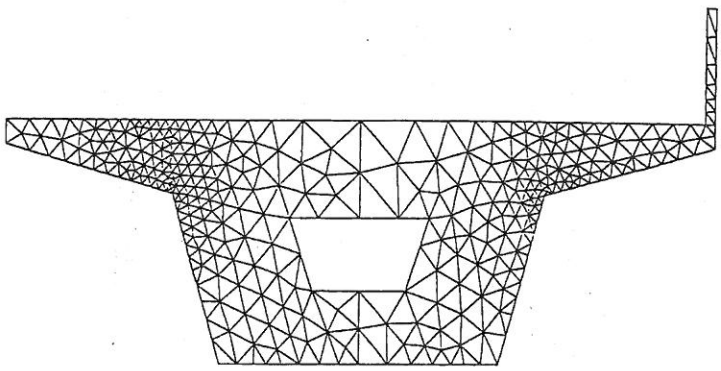

Figure 12. Refined triangular mesh

In Figure 12 the refined triangular mesh is shown, reflecting the desired mesh density very well. The transformation of this triangular mesh into quadrilaterals results in a mesh shown in Figure 13. Well shaped quadrilaterals are generated even in the transition zones of coarse to fine mesh. The computational time on an HP9000-400 with a Motorola 68030 processor was 120 s for the triangular mesh of Figure 12 and an additional 30 s for the transformation to the quadrilateral mesh.

The final two pictures in Figures 14 and 15 show a structure with various refinement zones. A zoomed plot of a strongly refined region in Figure 15 underlines again the capabilities of the algorithm.

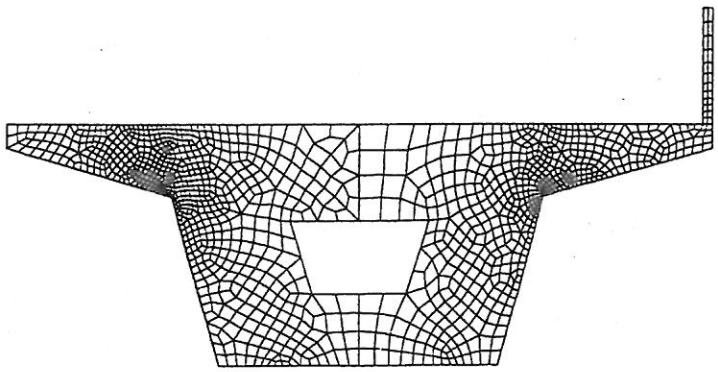

Figure 13. Refined quadrilateral mesh

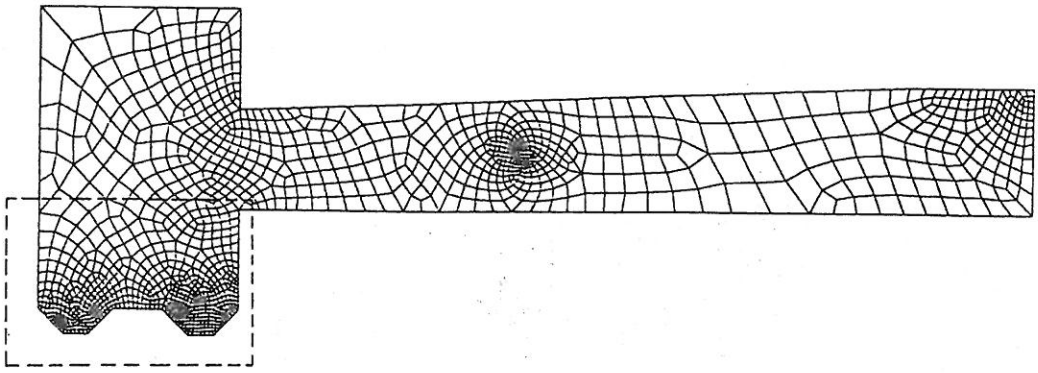

Figure 14. Quadrilateral mesh with various refinement zones

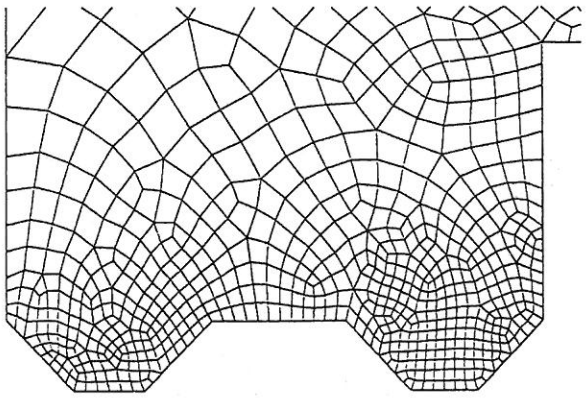

Figure 15. Zoomed plot of strongly refined zones

REFERENCES

1. K. Ho-Le, 'Finite element mesh generation methods: a review and classification', *Computer-Aided Design*, **20**, 27–38 (1988).
2. H. Jin and N.-E. Wiberg, 'Two-dimensional mesh generation, adaptive remeshing and refinement', *Int. j. numer. methods eng.*, **29**, 1501–1526 (1990).
3. P. L. Baemann and M. S. Shephard, 'Adaptive multiple-level h-refinement in automated finite element analyses', *Eng. Comput.*, **5**, 235–247 (1989).
4. P. L. Baemann, S. L. Wittchen, M. S. Shephard, K. R. Grice and M. A. Yerry, 'Robust, geometrically based, automatic two-dimensional mesh generation', *Int. j. numer. methods eng.*, **24**, 1043–1078 (1987).
5. J. A. Talbert and A. R. Parkinson, 'Development of an automatic, two-dimensional finite element mesh generator using quadrilateral elements and bezier curve boundary definition', *Int. j. numer. methods eng.*, **29**, 1551–1567 (1990).
6. B. P. Johnston, J. M. Sullivan and A. Kwasnik, 'Automatic conversion of triangular element meshes to quadrilateral elements', *Int. j. numer. methods eng.*, **31**, 67–84 (1991).
7. J. Z. Zhu, O. C. Zienkiewicz, E. Hinton and J. Wu, 'A new approach to the development of automatic quadrilateral mesh generation', *Int. j. numer. methods eng.*, **32**, 849–866 (1991).
8. A. Bykat, 'Design of a recursive, shape controlling mesh generator', *Int. j. numer. methods eng.*, **19**, 1375–1390 (1983).
9. R. E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations*, Vol. 7, Society for Industrial and Applied Mathematics, Philadelphia 1990.
10. O. C. Zienkiewicz and J. Z. Zhu, 'A simple error estimator and adaptive procedure for practical engineering analysis', *Int. j. numer. methods eng.*, **24**, 337–357 (1987).