Adaptive Mesh Generation for Triangular or Quadrilateral Elements

Manfred Schweingruber, Ernst Rank*

1 Introduction

Since the beginning of practical use of finite element methods automatic mesh generation has become more and more important. Automatic mesh generators can be classified into two groups, in the macro element technique and in the free meshing techniques. In generators of the first class the user has to devide the structure into simple basic regions, described most frequently by 3 to 8 noded isoparametric mappings. The main advantage of these mapping methods is the possibility to generate well structured meshes. The main disadvantage is, that it is nearly impossible to generate strongly graded meshes on arbitrary domains. Various free meshing algorithms (e.g. [1] [2] [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 is, that most codes result in triangular meshes, and there are only few publications on pure quadilateral meshing [4] [5] [6].

In the second section of this paper we will present an adaptive meshing algorithm based on an idea of Bank [7], being able to generate strongly graded triangular meshes. The conversion of triangular to quadrilateral meshes is described in section 3, extending an algorithm presented in [8] to domains with curved boundaries.

In the last section we will introduce a basic relaxation method to get optimized element shapes. Numerical examples will demonstrate the quality of resulting meshes.

2 Adaptive triangular meshing

Our mesh generator is based on the idea of recursive region splitting. Let a domain be described by a union of simply connected, non-overlapping regions, each defined by a sequence of boundary segments. These can be composed of straight lines and curves. First of all on each segment of the boundary new nodes are generated according to a nodal distance value H. The value is computed by

$$H = \sqrt{\frac{2 \cdot A}{N_T}} \tag{1}$$

where A is the area of the domain and N_T the desired number of triangles.

Now the algorithm starts with the first generated node on the first segment and splits the region into two parts R_1 and R_2, so that acceptable angles close to 60^o at the endpoints of the new line are obtained. Then new nodes are generated on this line. If R_2 has more than 3 nodes, the splitting is repeated starting with the next generated node. If R_n has three nodes, a triangle is defined and splitting is continued in region R_{n-1}. With this algorithm it is possible to generate a quasi-uniform mesh. To be able to introduce a function for a 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.

*Universität Dortmund, FB Bauwesen, NMI, D-4600 Dortmund 50, Germany. The work of the first author was partially supported by SIEMENS AG, Corporate Research and Development

$d(x)$ 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 desired local distance of nodes at point x. A quality measure to be observed for 'good' triangular meshes is the ratio of diametres of adjacent elements. This ratio has to be limited by a user definable number RATIO. If this limit conflicts with the local distance function, $h(x)$ has to be smoothed.
We generate a second mesh M_2 in a way similar to that described above, observing now the mesh density defined by the first mesh M_1 on the generated new splitting lines. We obtain a function $f(t)$,being the distance function $h(x)$ restricted to the splitting line (Fig. 1). The number of nodes to be generated on each new line is

$$N_g = \sum_{i=1}^{k-1} N(k_i, k_{i+1}) \qquad (2)$$

with $N(k_i, k_{i+1})$ =(Number of new nodes between intersection point k_i and k_{i+1} of the new splitting line with the old mesh). Then $F(x)$ is defined by integration of $f(t)$

$$y = F(x) = \int_{x_a}^{x} f(t)\,dt \quad . \qquad (3)$$

As $f(t)$ is strictly positive, $F(x)$ is a monotonously increasing function of x. Taking N_g intervals of equidistant length on the y-axis and corresponding points on the x-axis (Fig. 2), we finally get the desired density-controlled distribution of nodal points on the intersection line.
The efficency of this method is shown in Fig. 3 for an L-shaped domain with a local refinement at the reentrant corner.

3 Transformation of triangular to quadrilateral meshes

Johnston [9] has published a transformation method which combines, in a first step, suitable neighboring triangles to quadrilaterals. Obviously this strategy alone can not create a pure quadrilateral mesh for an odd number of triangles, as well as an even number may produce 'islands' of isolated triangles. Therefore in a second step these islands are resolved by edge flipping techniques or by introduction of 'split propagation' which often has to be defined up to the boundary of the domain.
Instead of combining two neighboring triangles to *one* quadrilateral, our strategy is to split two adjacent triangles into *four* quadrilaterals as shown in Fig. 4. If P_1 to P_4 are chosen to be midside points, a compatible quadrilateral mesh will be generated except for possible islands of one or more triangles. As shown in Fig. 5 every remaining triangle can be split into 3 quadrilaterals, connecting points on the sides with an interior point, finally resulting in a purely quadrilateral mesh. Obviously this procedure is not unique. As our goal is to obtain well-shaped quadrilaterals, first a list of all *possible* combinations of neighboring triangles is generated and sorted according to the following angle-criterion:

$$G = \sum_{i=1}^{4} |\alpha_i - 90^\circ| \qquad (4)$$

A combination is forbidden if one angle of the quadrilateral is greater than an acceptable angle less then 90^o.*Actual* combinations are then generated in the sequence of this list. This algorithm is very easy to implement and, due to its strictly *local* nature (except for the sorting of list (4)) only of *linear* time complexity. An adaptive, purely quadrilateral mesh is shown in Fig. 6.

4 Nodal Relaxation

The desired interior angle for triangular elements is 60^o and so, after conversion of a 'good' triangular to a quadrilateral mesh, an accumulation of angles near 60^o and 120^o is observed. The optimal interior angle for quadrilaterals is yet 90^o, so the converted mesh has still to be improved. In our mesh generator a relaxation according to the nodal distance function is performed. After generation there is a difference between the *actual* distance d_{ik} of node i to each of its neighbors k=1,... n_i with relative coordinates (x_{ik}, y_{ik}) and the mean *desired* distance $\bar{h}_{ik}$ in node i and node k.

$$\bar{h}_{ik} = \frac{h_i + h_k}{2} \qquad (5)$$

The relaxed coordinates of node i are defined by

$$x_i = (\sum_{k=1}^{n_i} \frac{x_{ik} \cdot d_{ik}}{\bar{h}_{ik}}) \;/\; (\sum_{k=1}^{n_i} \frac{d_{ik}}{\bar{h}_{ik}}) \qquad (6)$$

$$y_i = (\sum_{k=1}^{n_i} \frac{y_{ik} \cdot d_{ik}}{\bar{h}_{ik}}) \;/\; (\sum_{k=1}^{n_i} \frac{d_{ik}}{\bar{h}_{ik}}) \qquad (7)$$

Numerical experiences show that three relaxation sweeps are enough for sufficiently smoothing the mesh. Fig. 7 shows the mesh of Fig. 6 after 3 relaxation sweeps.

References

[1] P.L. BAEHMANN AND M.S. SHEPHARD: *Adaptive multiple-level h-refinement in automated finite element analyses*, Engineering with Computers, **5**, 235-247, (1989)

[2] J.C. CAVENDISH: *Automatic triangulation of arbitrary planar domains for the finite element method*, Int. j. numer. methods eng., **8**, 679-696, (1974)

[3] H.JIN AND N.-E. WIBERG: *Two-dimensional mesh generation, adaptive remeshing and refinement*, Int. j. numer. methods eng., **29**, 1501-1526, (1990)

[4] P.L. BAEHMANN, S.L. WITTCHEN, M.S. SHEPHARD, K.R.G. RICE AND M.A. YERRY : *Robust, geometrically based, automatic twodimensional mesh generation*, Int. j. numer. methods eng.,**24**,1043-1078(1987)

[5] J.A. TALBERT AND A.R. PARKINSON: *Development of an automatic, twodimensional finite element mesh generator using quadrilateral elements and bezier curve boundary definition*, Int. j. numer. methods eng.,**29**,1551-1567(1990)

[6] 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)

[7] R.E. BANK : *PLTMG: A software package for solving elliptic partial differential equations, Vol. 6*,Society for Industrial and applied Mathematics, Philadelphia 1990

[8] E. RANK, M. SCHWEINGRUBER, M. SOMMER: *Adaptive mesh generation and transformation of triangular to quadrilateral meshes*, To be published in Comm. Appl. Num. Meth.

[9] 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)

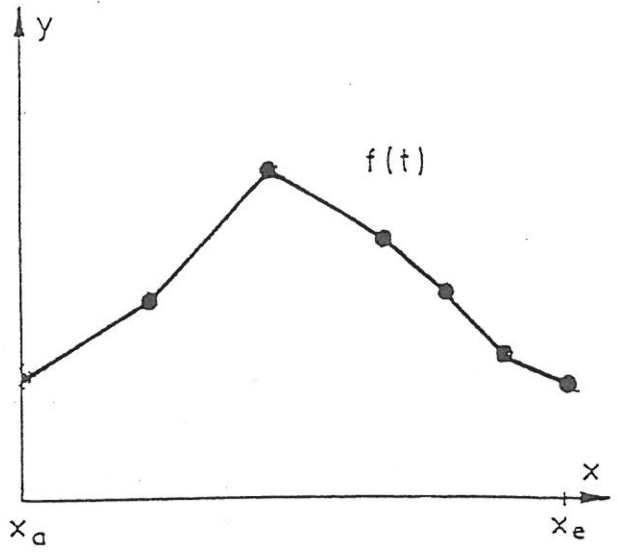

Fig.1: Distance function $f(t)$ over new splitting line

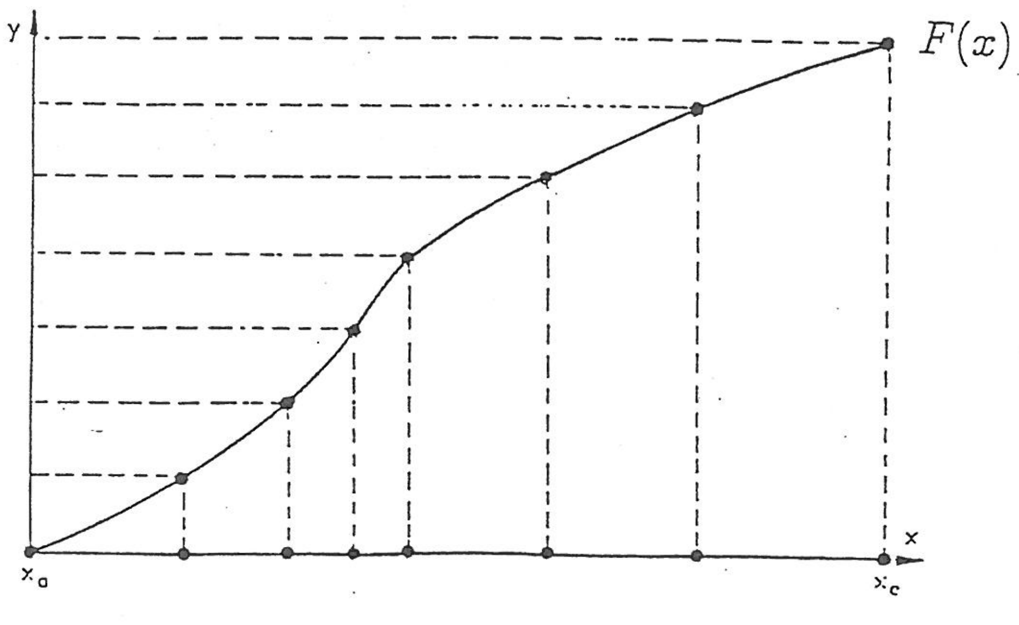

Fig.2: Definition of new nodal points on splitting line

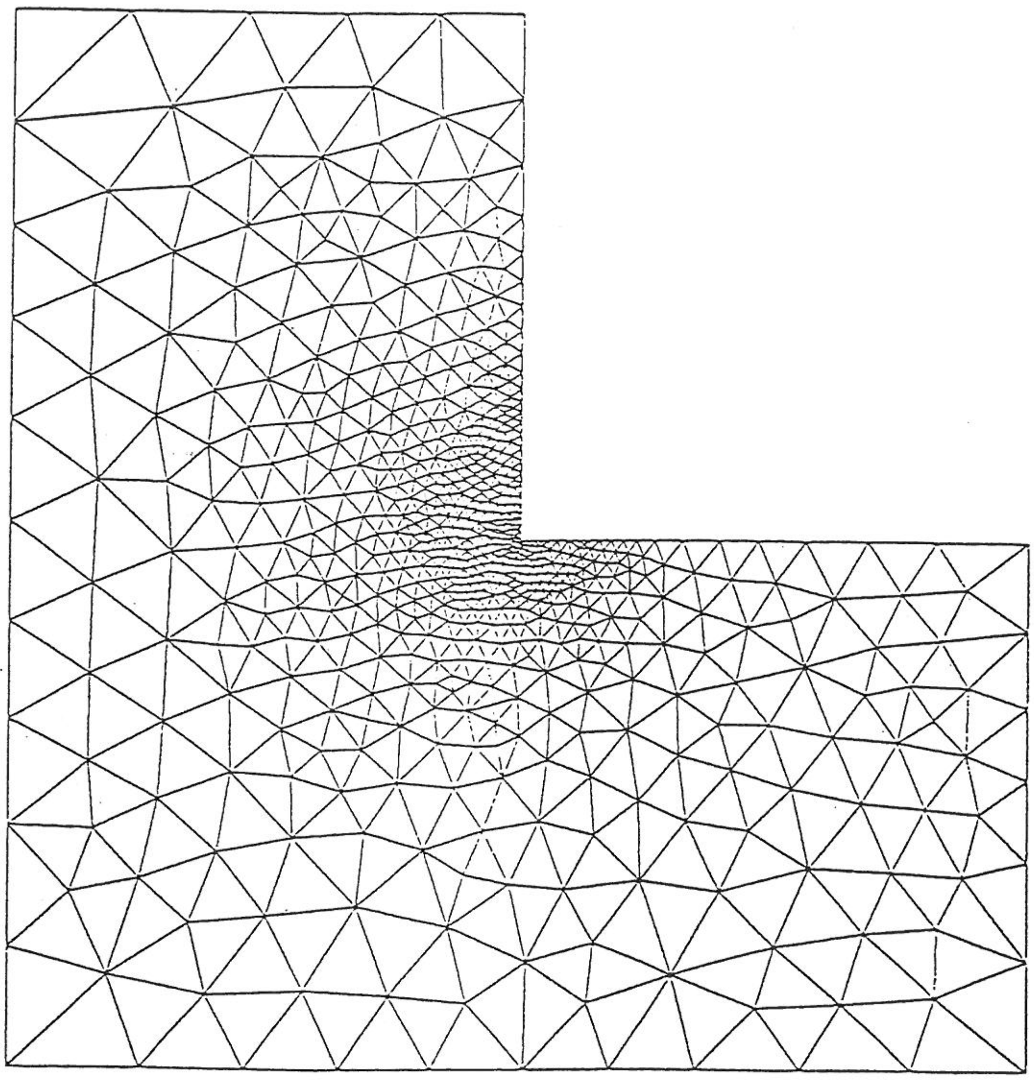

Fig.3: Adaptive triangulated mesh

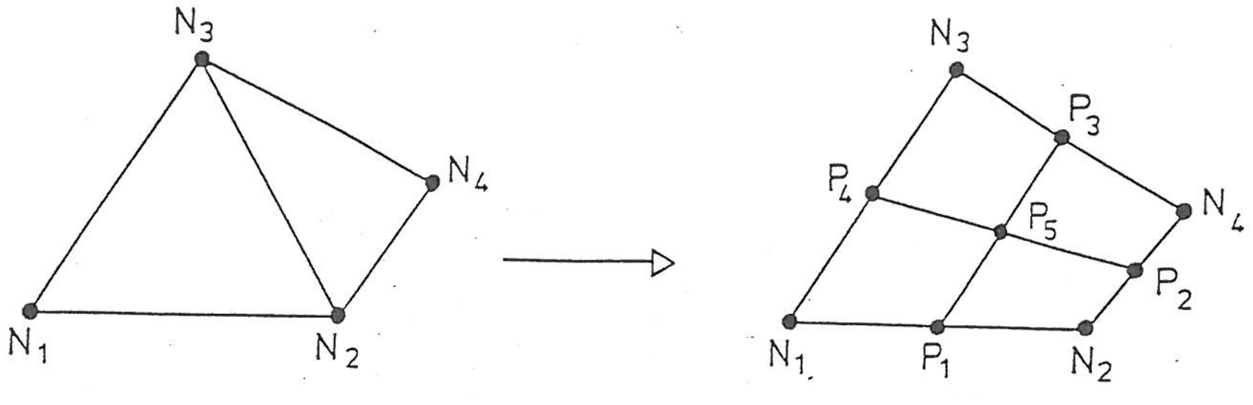

Fig.4: Quadrilaterals created by splitting of two neighboring triangles

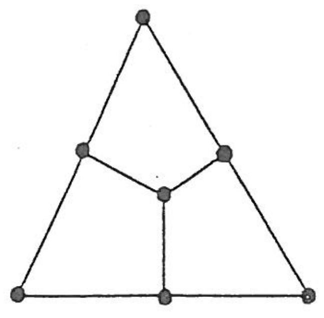

Fig.5: Triangle split into 3 quadrilaterals

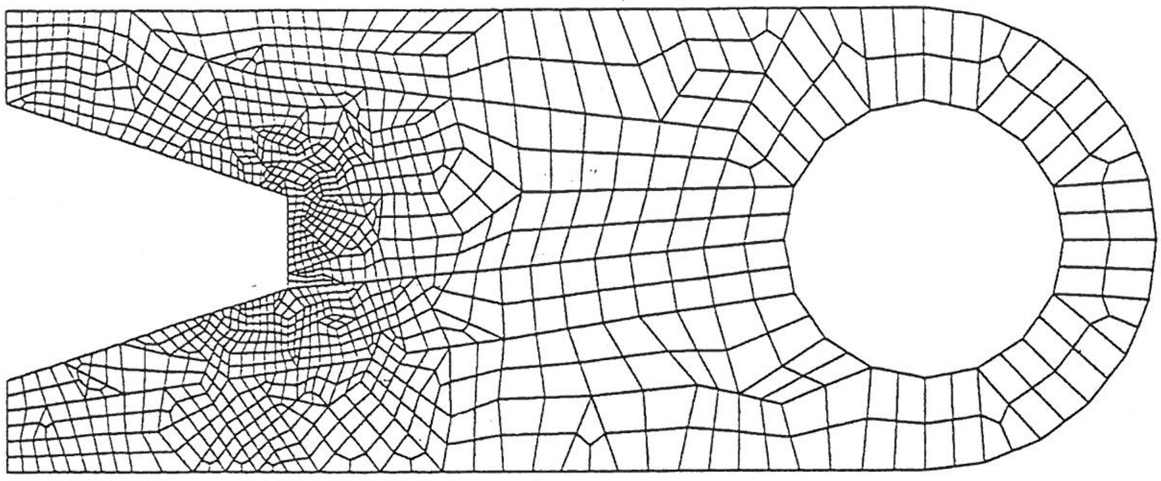

Fig.6: Refined quadrilateral mesh (no relaxation)

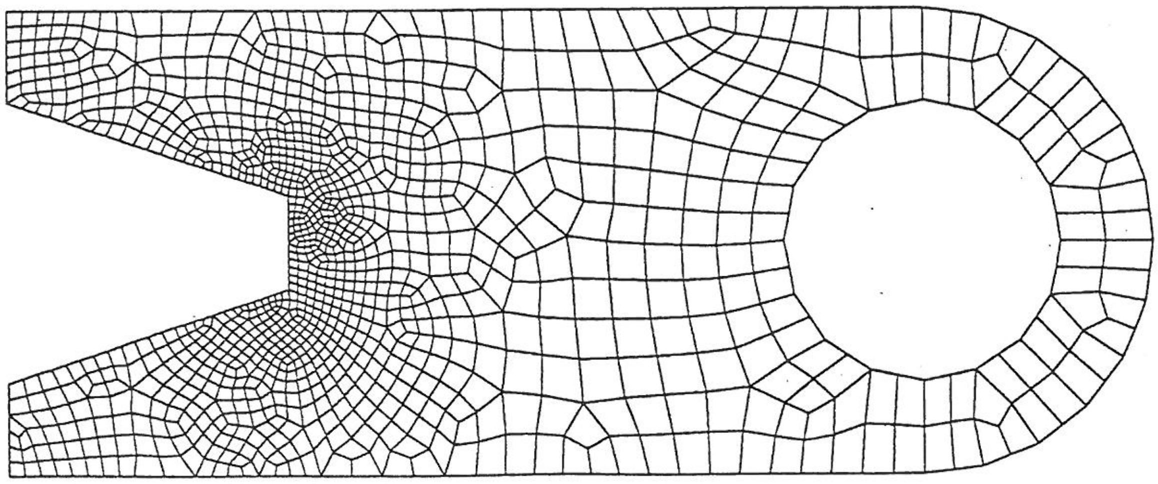

Fig.7: Refined quadrilateral mesh after relaxation