Computing in Civil and Building Engineering, Pahl & Werner (eds) © 1995 Balkema, Rotterdam. ISBN 90 5410 556 9

Automatic mesh generation for plates and shells

Ernst Rank, Martin Rücker & Manfred Schweingruber
Numerische Methoden und Informationsverarbeitung, University of Dortmund, Germany

ABSTRACT: The paper will present a method to generate triangular and quadrilateral meshes on curved surfaces, using message passing between geometric modeller and a generator for meshes in the parameter plane of surface patches. The coupling is based on standard interprocess communication with a minimal set of message types. Initial mesh distortion on the surface is removed using a local Delaunay triangulation in the tangential plane. Complex examples show the efficiency of the method.

1 INTRODUCTION

Considerable progress has been made during the past few years in the development of algorithms for an automatic mesh generation of spatial structures. Free meshing, which will be discussed in this paper, is most frequently based on Delaunay methods (e.g. [George,1991]), quad-tree or oct-tree methods (e.g. [Shephard et.al., 1991]) or methods using recursive region splitting (e.g. [Bank, 1990]). Many variants of these basic methods are known, most of these algorithms generating triangular or mixed triangular-quadrilateral meshes. Especially in plate and shell computations however, pure quadrilateral meshes are known to be in general much less stiff and much more accurate than corresponding triangular elements. Therefore a very simple algorithm for the conversion of any triangular mesh into a purely quadrilateral mesh, which was first published by the authors [Rank et.al.,1992] was taken up by many other researches and further developed. Although this technique is not restricted to plane structures (see e.g. [Okstad et.al.,1994], [Ramm et.al.,1994],[Kreiner et.al.,1994] for related mesh generation techniques on surfaces), there appear still many problems when arbitrary shell structures have to be meshed automatically. Some of these problems, especially the question of mesh distortion and of interfacing mesh generation and geometric modelling, shall be addressed in this paper. Section 2 will outline a concept for an interprocess communication between mesh generator and modeller, strictly hiding the methods of the two processes and thus guaranteeing full geometric flexibility of the structures to be meshed. Section 3 will present a method to remove mesh distortion resulting from the parametrization of non-revolutable surfaces and section 4 will give several examples.

2 INTERFACING MESH GENERATION AND GEOMETRIC MODELLING

There are two common ways of interfacing modelling systems and mesh generation. In the first approach, meshing is integrated as a task in the geometric modeller, using all geometric entities of the modelling program (e.g. [ANSYS,1992]). A major disadvantage of this approach is that it is obviously necessary to have (at least part of) the source code of the modelling program available and to be able to recompile the functional interface of the two programs. The second approach strictly seperates modelling and mesh generation and makes all geometric information available to the mesh generator in some file-based interface. Besides all problems of definition of data exchange formats this

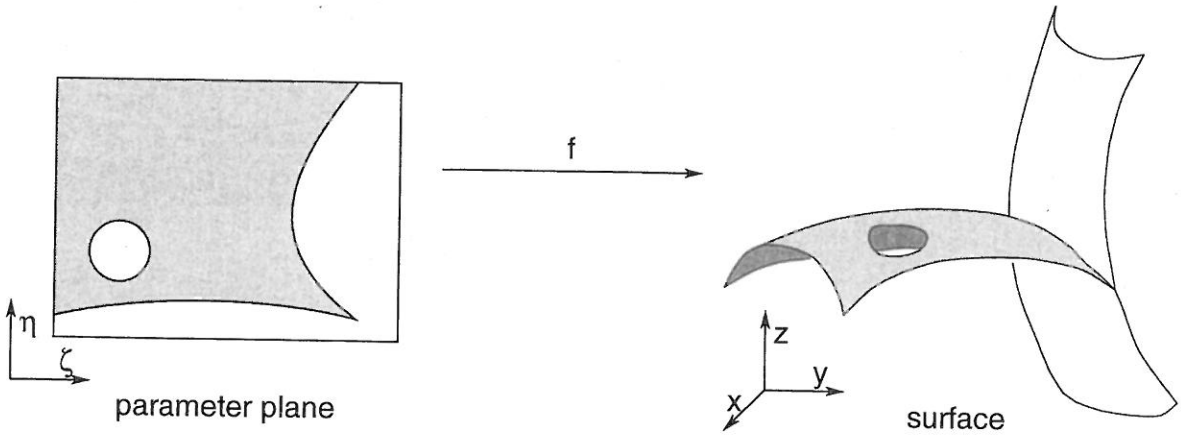

Figure 1: $(\zeta_0, \eta_0) \rightarrow f(\zeta_0, \eta_0) = (x_0, y_0, z_0)$

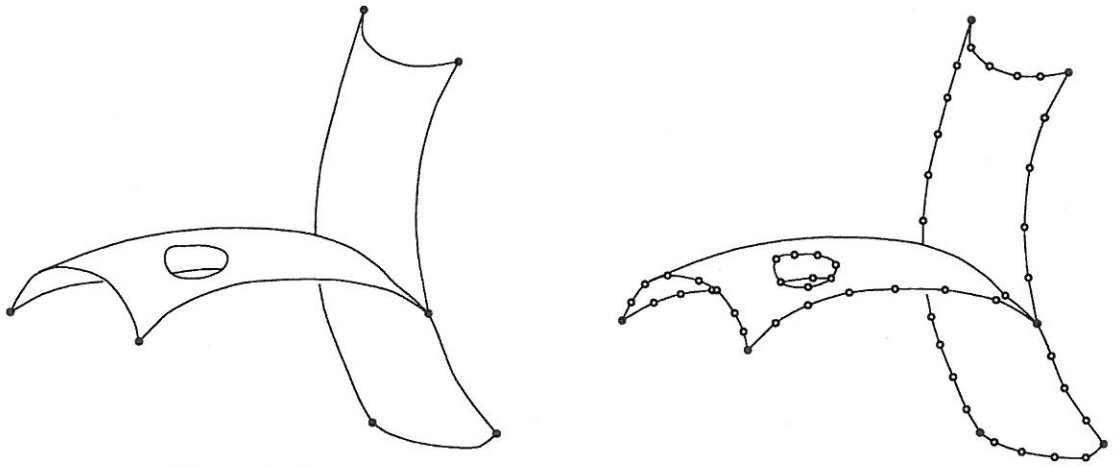

Figure 2: Key-points and nodes created on intersection segments

approach hinges on the problem, that any kind of surface being available in the modelling program has to be coded in a consistent way in the mesh generator again, making this sort of coupling very inflexible. In the following we will discuss a third possibility of coupling, strictly seperating methods for modelling and meshing, with only very limited message passing in an interprocess communication.

Let us assume, that a spatial structure to be meshed is composed of n parametric patches, being connected by curve segments. In each of these patches the surface is defined by some functional description, which may be a global polynominal, a B-spline, a Bezier surface or any other mapping of a parameter plane $[a, b] \times [c, d], a < b, c < d$ to the surface. Note that in general the origin of the patch will only be a part of the parameter plane (see Figure 1).

We will now only assume, that the geometric modeller used is able to provide a functional interface for f and $g = f^{-1}$:

f:: get-global-coordinates-for (ζ, η)

g:: get-closest-local-coordinates-for (x, y, z)

The first function will return x, y, z-coordinates, the second function a distance to the point of projection and ζ, η-coordinates in the parameter plane of the projected point x_p, y_p, z_p. We will further assume, that the second function is called only for points *on* a patch or *close* to a patch, so that the projection is unique. It is very important to note that it will *not* be necessary to have any knowledge on the type of the surface, wether it is a polynominal, a spline, a cylinder, or a sphere etc..

As outlined in [Rank et al.,1994], the surface model shall be defined by the hierarchy of objects

points — line segments — patches

each associated with topologic and geometric data.

We will now explain the major steps to a mesh generation with a uniform element size h on a sur-

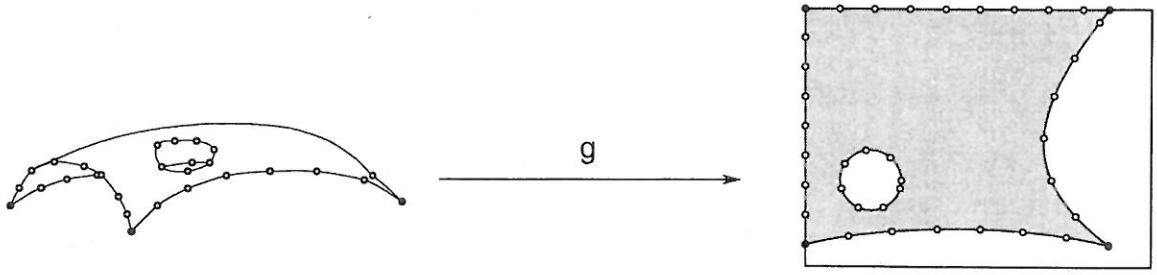

Figure 3: Surface patch and meshing region in parameter plane

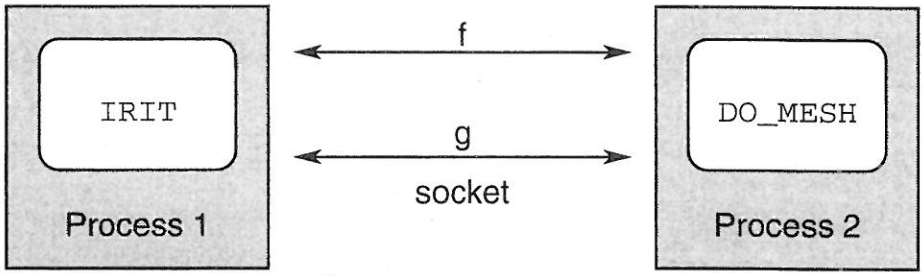

Figure 4: Interprocess communication via UNIX-'socket'

face composed of parametric patches. Generalization to non-uniform meshes can be performed using a 'background mesh' as described in [Rank et.al.,1994].

Step 1:Generate nodes on all line segments connecting patches on the surface at distance h.

As this step can in general not be performed exactly, nodes are distributed equidistantly on each segment, with the closest possible distance to h. This task can usually be performed within the modelling system itself. The result of this first step is still the 'exact' surface, with additional nodes on the boundaries of the patches (Figure 2).

Step 2 to 5 are looped over all patches of the structure:

Step 2: For all points (x_i, y_i, z_i) on the boundaries of a patch: Find parameter (ζ_i, η_i) in the parameter plane.

This step uses function g and marks the first communication between modeller and mesh generator (see Figure 3).

Note that line segments between boundary points in the parameter plane are still *curves*, but it will be seen in the following that they can be replaced by straight line segments without loss of generality:

Step 3: Approximate patch in the parameter plane by a polygonal defined by the points found in step 2.

The next step is now performed by the mesh generator in the parameter plane:

Step 4: Generate a (triangular or quadrilateral) mesh for the polygonal domain of step 3, without creating additional boundary nodes.

Next, a first mesh is created on the surface:

Step 5: Map the mesh from the ζ, η-plane onto the surface patch, using interface function f.

After looping Steps 2 to 5 over all patches, the mesh on the surface is compatible by construction, as no additional nodes are created on the interfaces between patches. In general, curve segments of the interfaces will now have to be replaced by geometric objects (in the most simple case by straight lines) which can be handled by the finite element program to be used. Yet this *geometric* information needs not be communicated to the mesh generator, meshing of interfaces needs only the *topological* information of neighboring patches.

In our implementation, we used the public domain geometric modeller IRIT [IRIT,1993], interfaced to our generator DO_MESH for meshes in the parameter plane by standard UNIX™-interprocess communication, one process running the geometric modeller, the other one the mesh generator. Messages between processes (via functions f and g) are exchanged very frequently during the meshing process, yet each individual message is very short, as only local and global

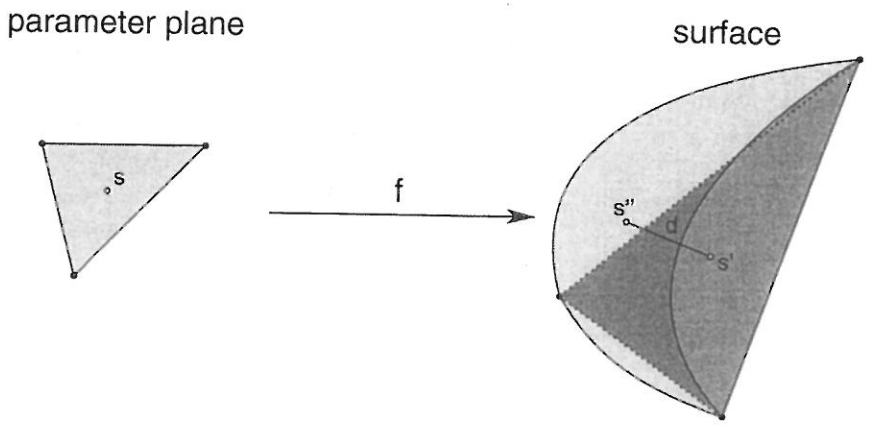

Figure 5: Definition of curvature criterion

coordinates are communicated. Therefore an interprocess communication using message passing software like PVM or MPI, which opens and closes a communication channel for each message, turned out to be very inefficient. Using UNIX-'sockets', i.e. opening a channel only once, proved to be much more adequate for the communication used in our application (Figure 4).

3 REGULARIZATION OF DISTORTED MESHES ON THE SURFACE

Depending on the parametrization of the patches, the transformed mesh will be more or less distorted. A method to regularize triangular meshes will be explained in this section.

Consider an element e mapped from the parameter plane to the patch by the mapping function f. In a first step three quality criteria for e are checked.

1. *Distortion of element e on the surface*: The criterion used is that given by [Bank,1990]:
$g(e) := 4\sqrt{3}a/(h_1^2 + h_2^2 + h_3^2)$
where h_i are hights and a is the area of triangle e.
If $g(e) \leq 0.85$, this criterion fails.

2. *Desired element size h of e:*
If one of the element sides s_i, $i = 1, 2$ is greater than 1.5 times the desired local mesh size, this criterion fails.

3. *Local curvature:*
If the distance d of the triangle's center of gravity s' to the image $s'' = f(s)$ of the center of gravity s of the triangle in parameter space is greater than λ times the elements diameter, with a given parameter λ, this criterion fails. This curvature control is similar to that given by [Rust, 1991].(See Figure 5).

If *one* of these criteria fails, s'' is inserted as a new node into the mesh by a *local Delaunay remeshing*. Neighboring triangles of e are mapped into the tangential plane of e and a local remeshing is performed similarly to [George, 1991, pp. 162–163].

Note that this remeshing only needs the parametric description $f(\zeta, \eta)$ of the surface, i.e. the interface provided by the interprocess communication as described in Section 2. The local remeshing is looped over all elements n times, until all elements pass quality criteria 1 to 3. Typically $n = 5$ is enough even for strongly distorted initial meshes.

Transformation of the resulting triangular mesh and nodal relaxation is performed in a similar way as outlined in [Rank et.al., 1992] and will be described in detail in a forthcoming paper.

4 NUMERICAL EXAMPLES

In the first numerical example the local remeshing to remove distorted elements will be demonstrated. Figure 6 shows a Bezier surface over a quadratic patch, as obtained from the geometric modeller [IRIT, 1993]. This patch is meshed in a first step in the parameter plane into the initial triangular mesh. Mapping the triangular mesh onto the surface yields the distorted mesh of the same Figure 6 which is then locally remeshed as described in Section 3. The corresponding mesh in the parameter plane is now strongly distorted, as can be seen from Figure 7. Transformation of the

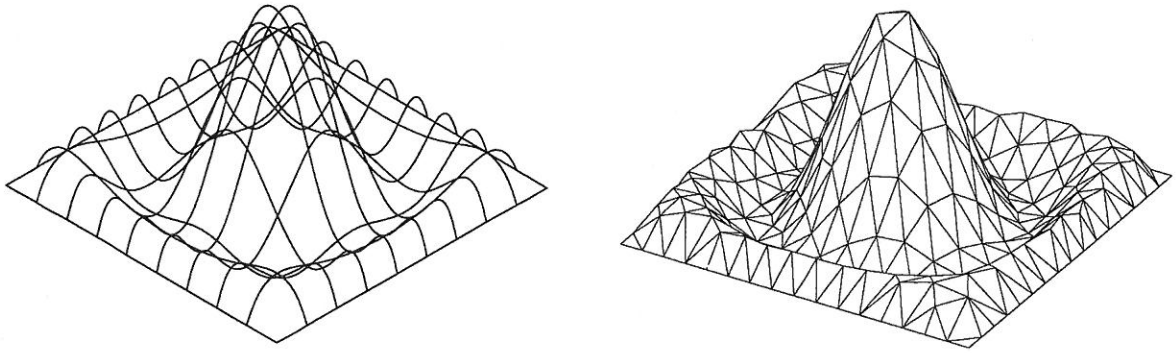

Figure 6: Bezier surface and and distorted mapped mesh on the surface

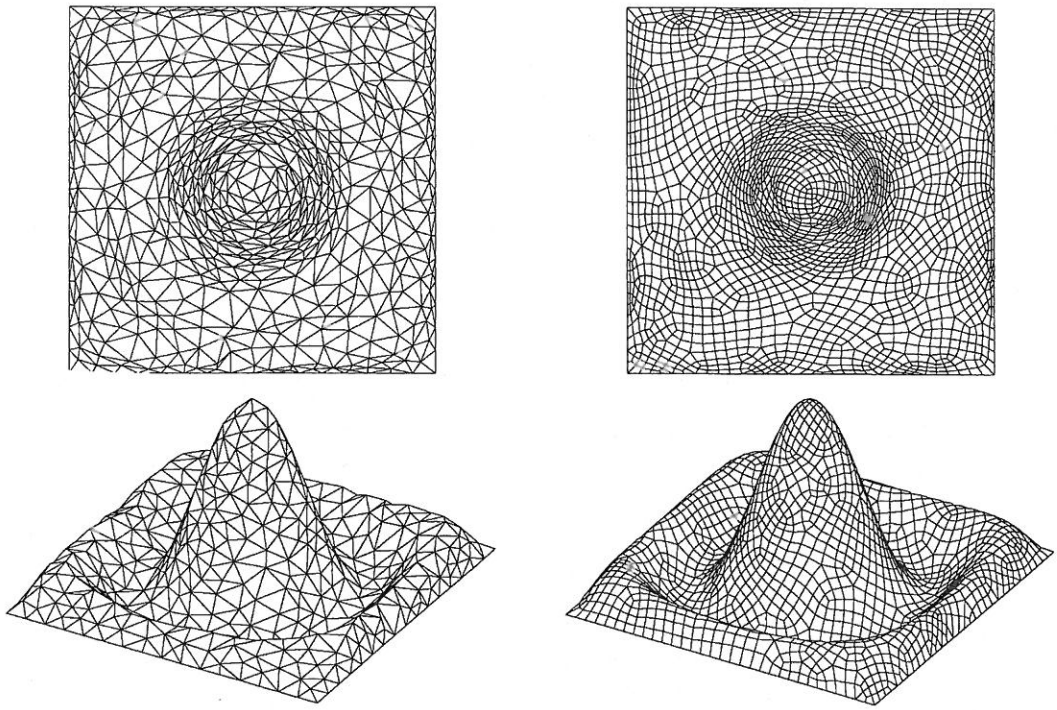

Figure 7: Triangular and quadrilateral meshes after local remeshing in parameter plane and on the surface

triangular mesh [Rank et.al., 1992] finally yields the undistorted quadrilateral mesh on the surface, corresponding to a strongly distorted quadrilateral mesh in the original meshing plane (Figure 7).

It should be noted, that a straightforward variant of the method outlined could be used to generate *anisotropic triangular meshes* in the plane, defining local 'pseudo-surfaces' with a parametrization corresponding to the desired anisotropy. These meshes could be valuable in mesh adaptation for reaction-diffusion type problems or for localization problems [Kornhuber et.al., 1990], [Apel et.al.,1994].

Our final example shows the quadrilateral mesh on a complex structure (Figure 8) being composed of intersecting cylinders and a sphere. In the same Figure a detail near an intersection is shown, demonstrating the mesh compatibility as guaranteed by the presented algorithm.

REFERENCES

T. Apel, G. Lube, 1994. Anisotropic mesh refinement in stabilized Galerkin methods. *Preprint-Reihe der Chemnitzer DFG-Forschergruppe "Scientific Parallel Computing".*

ANSYS *User's Manual*, Revision 5.0, 1992. Swanson Analysis Systems, Inc., Houston.

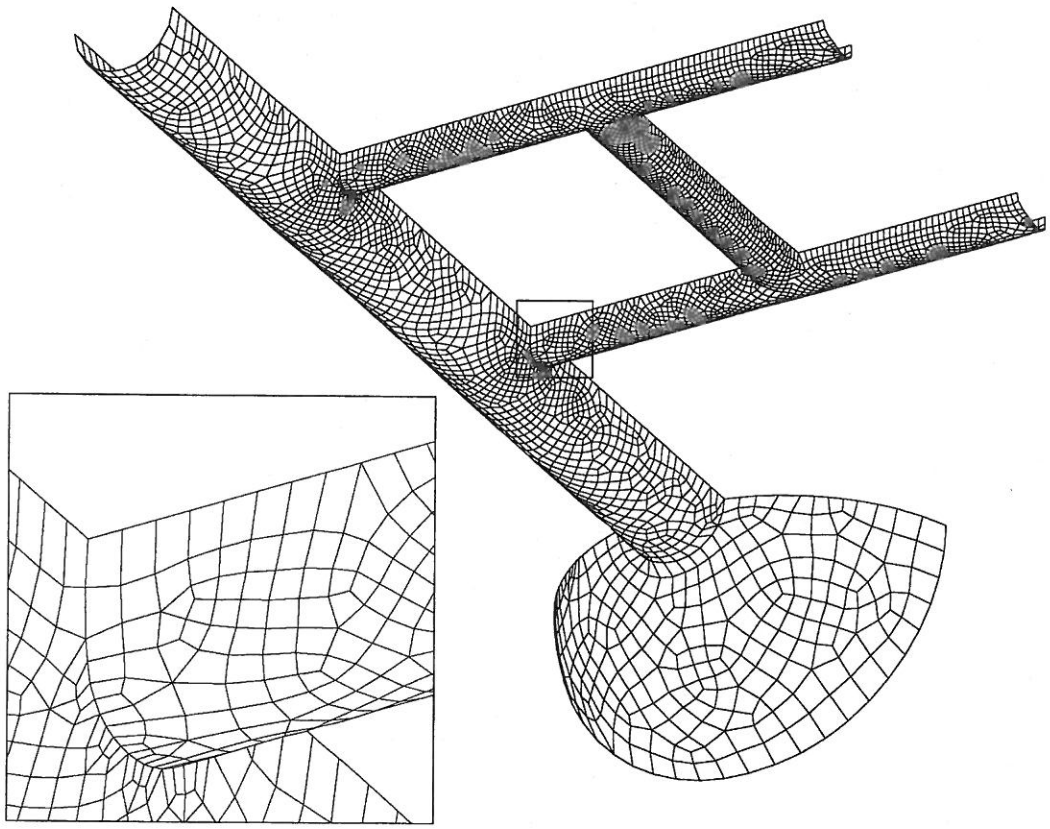

Figure 8: Intersecting surfaces with zommed detail

R.E. Bank, 1990. PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, Users' Guide 6.0. In *Frontiers in Applied Mathematics, Vol. 7.* Philadelphia: SIAM.

P.L. George, 1991. *Automatic Mesh Generation – Application to Finite Element Methods.* Paris: J.Wiley & Sons.

IRIT 4.0 *User's Manual*, 1993. A Solid modeling Program, Gershon Elbers.
EMail: gershon@cs.technion.ac.il

R. Kornhuber, R.Roitzsch,1990. On adaptive grid refinement in the presence of internal and boundary layers. *IMPACT of Comp. in Sci. and Engrg.*2:40–72.

R. Kreiner, B.Kröplin,1994. Unstructured quadrilateral mesh generation on surfaces from CAD. In: *Numerical grid generation in computational fluid dynamics and related fields*, N.P. Weatherhill, ed., Swansea, Pinderidge Press.

E. Ramm, N. Rehle, 1994. Netzgenerierung auf Parameterflächen. *Mitteilungen des Instituts für Baustatik der Universität Stuttgart I/94*

K. M. Okstad, and K. M. Mathisen,1994. Towards Automatic Adaptive Geometrically Non-linear Shell Analysis. Part I: Implementation of an h-adaptive Mesh Refinement Procedure', *Int. Journal for Num. Meth. in Eng.* 37:2657–2678.

E. Rank, M. Schweingruber, M.Sommer, 1992. Adaptive mesh generation and transformation of triangular to quadrilateral meshes. *Comm. in Appl. Num. Meth.* 9:121–129.

E. Rank, M. Rücker, M. Schweingruber, 1994. Automatische Generierung von Finite-Element-Netzen. *Bauingenieur.* 69:373–379.

W. Rust, 1991. *Mehrgitterverfahren und Netzadaption für lineare statische Finite-Elemente-Berechnungen von Flächentragwerken.* Forschungs- und Seminarberichte aus dem Bereich der Mechanik der Universität Hannover.

M. Shephard and M. Georges, 1991. Automatic three-dimensional mesh generation by the finite octree technique. *Int. Journal for Num. Meth. in Eng.* 32:709–749.