ELSEVIER Finite Elements in Analysis and Design 25 (1997) 111-134

FINITE ELEMENTS IN ANALYSIS AND DESIGN

Automatic unstructured grid generators

Rainald Löhner

GMU/CSI, The George Mason University, Fairfax, VA 22030-4444, USA

Abstract

A review of automatic unstructured grid generators is given. These types of grids have found widespread use in computational fluid dynamics, computational structural dynamics, computational electro-magnetics and computational thermodynamics. The following topics are treated: the methods most commonly used, the specification of desired element size/shape and surface definition/meshing. Finally, the use of automatic grid generators as an enabling technology for moving body simulations and adaptive remeshing techniques is discussed.

Keywords: Unstructured grid generation; Finite elements; Triangular and tetrahedral meshes

1. Introduction

The advent of numerical methods based on the spatial subdivision of a domain into polyhedra or elements immediately gave rise to the task of generating a mesh. This is especially true for the case of numerical techniques based on the so-called unstructured grids. Typical grids of this kind consist of triangular or tetrahedral elements that may have an arbitrary number of neighbours. Other element types include trapezoids, bricks or prisms. With the availability of more versatile field solvers and powerful computers, analysts attempted the simulation of ever increasing geometrical and physical complexity. At some point (probably around 1985), the main bottleneck in the analysis process became the grid generation itself. The last decade has seen a considerable amount of effort devoted to automatic grid generation [1-5], resulting in two very powerful and by now mature techniques: the Advancing Front and the Delauney Triangulation. Typical unstructured grids commonly used in CFD today consist of 1 Mpts for simple parts (e.g. a wing with an Euler solver), and up to 25 Mpts for complete configurations. It is estimated that before the end of the decade grids in excess of 100 Mpts will be common for Reynolds-Averaged Navier Stokes (RANS) flow simulations of complete configurations.

Any automatic grid generator requires as input the following information:

(a) *Surface definition*, i.e. a description of the bounding surfaces of the domain to be gridded;

0168-874X/97/$17.00 © 1997 Elsevier Science B.V. All rights reserved
PII S0168-874X(96)00038-8

(b) *Mesh size and shape*, i.e. a description of how the element size, shape and orientation should be in space;

(c) *Element type*, i.e. tetrahedron, brick, etc.; and

(d) *Grid generation technique*, i.e. the choice of a suitable method to achieve the generation of the desired mesh given the first three pieces of information.

Historically, the work progressed in the opposite order to the list given above. This is not surprising as the same happened when solvers were being developed. In the same way that the need of grid generation was only realized after field solvers were sufficiently efficient and versatile, surface definition and the specification of element size and shape only became issues once sufficiently versatile grid generators were at hand.

The present paper attempts a review of automatic grid generators for triangles and tetrahedra. The intended use of these grids is for computational fluid dynamics (CFD), computational structural dynamics (CSD), computational electro-magnetics (CEM) and computational thermodynamics (CTD), although they may also be used for visualization and data interpolation. The topics treated follow the historical development, i.e. the methods are first treated in Section 2, the specification of element size/shape follows in Section 3, and surface definition is dealt with in Section 4. The paper then focuses in Section 5 on the use of automatic grid generators as an enabling technology for adaptive remeshing techniques.

2. Unstructured grid generation methods

If we consider the task of filling a given domain with elements, there appear to be only two basic ways of accomplishing it:

- Filling the 'empty', i.e. as yet ungridded region with elements; or
- Modifying an existing grid that already covers the domain to be gridded.

The first class of techniques denotes the advancing front methods (AFMs) [6–21], the front being defined as the boundary between the gridded and ungridded region. The key algorithmic step that must be addressed for Advancing Front Methods is the proper *introduction of new elements* to the ungridded region. For triangular and tetrahedral grids the elements are introduced sequentially one at a time. For quadrilateral and hexahedral elements, this technique is known as paving or plastering [22, 23]. The second class of techniques is known as Voronoi or Delaunay triangulation methods. Here, the key algorithmic step is the proper *introduction of new points* to the given grid. This class of techniques has been used only for the construction of triangular or tetrahedral grids. The name Voronoi or Delaunay that is associated with these techniques stems from the element reconnection technique most often employed [24–46]. We remark that the modified or finite octree [47, 48] techniques represent just one possible realization of a Delaunay triangulation. Given the known distribution of points from the octree, the mesh connectivity is obtained by applying the circumcircle or Delaunay criterion.

2.1. The advancing front method

The advancing front method [6–11, 14–19, 21], consists in marching into as yet ungridded space by adding one element at a time. The region separating the gridded portion of space from the as yet

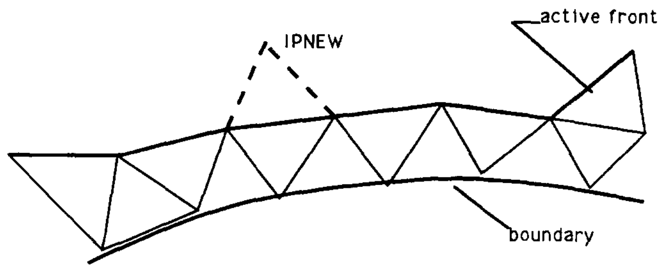


Fig. 1. Advancing front method (2-D).

ungridded one is called the *front*. The algorithm, which is sketched in Fig. 1, may be summarized as follows:

F1. Define the boundaries of the domain to be gridded. Without going into further detail, we will assume some general form of hierarchical surface definition consisting of patches, the lines that surround or delimit them, and points at the intersections of lines.

F2. Define the spatial variation of element size, stretchings, and stretching directions for the elements to be created.

F3. Using the information given for the distribution of element size and shape in space and the line-definitions: generate sides along the lines that connect surface patches. These sides form an initial front for the triangulation of the surface patches.

F4. Using the information given for the distribution of element size and shape in space, the sides already generated, and the surface definition: triangulate the surfaces. This yields the initial front of faces.

F5. Find the generation parameters (element size, element stretchings and stretching directions) for these faces.

F6. Select the next face to be deleted from the front; in order to avoid large elements crossing over regions of small elements, the face forming the smallest new element is selected as the next face to be deleted from the list of faces.

F7. For the face to be deleted:

F7.1 Select a 'best point' position for the introduction of a new point IPNEW.

F7.2 Determine whether a point exists in the already generated grid that should be used in lieu of the new point. If there is such a point, set this point to IPNEW and continue searching (go to F.7.2).

F7.3 Determine whether the element formed with the selected point IPNEW crosses any given faces. If it does, select a new point, as IPNEW and try again (go to F.7.3).

F8. Add the new element, point, and faces to their respective lists.

F9. Find the generation parameters for the new faces from the background grid and the sources.

F10. Delete the known faces from the list of faces.

F11. Add the new faces to the front.

F12. If there are any faces left in the front, go to F6.

A recent thesis by Frykestig [19] gives a good overview of the different possibilities explored to date for selecting the 'best point position', when to use given points vs. the introduction of a new point, specialized data structures for storing and retrieving mesh data, etc. The complexity of the advancing front method is of $O(N\log(N))$, where N denotes the number of elements. Over the years, optimal data structures have been implemented to realize such a favourable scaling [10, 11, 14, 49, 17, 19]. The procedure has been used extensively to grid large-scale complex geometry domains [50–56] and within adaptive remeshing procedures [8, 57–63, 14]. Sustained speeds in excess of 50 Ktet/min have been achieved on the CRAY-YMP and SGI Power Challenge [58, 17], and the procedure has been ported to parallel machines [17, 64]. Recent extensions include special gridding procedures for the highly stretched grids required when solving the Reynolds–Averaged Navier–Stokes equations [65–68].

2.2. *Delaunay technique*

The Delaunay triangulation technique has a long history in mathematics, geophysics and engineering. Given a set of points $\mathscr{P} := \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_n$, one may define a set of regions or volumes $\mathscr{V} := v_1, v_2, \ldots v_n$ assigned to each of the points, that satisfy the following property: any location within v_i is closer to $\boldsymbol{x}_i$ than to any other of the points:

$$v_i := \mathscr{P} : \| \boldsymbol{x} - \boldsymbol{x}_i \| < \| \boldsymbol{x} - \boldsymbol{x}_j \| \quad \forall j \neq i. \tag{1}$$

This set of volumes $\mathscr{V}$, which covers the domain completely, is known as the *Dirichlet tesselation*. The volumes v_i are convex polyhedra, and are referred to as Voronoi regions. Joining all the pairs of points $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ across polyhedral boundaries results in a triangulation of the convex hull of $\mathscr{P}$. It is this triangulation that is commonly know as the *Delaunay triangulation*. The set of triangles (tetrahedra) that form the Delaunay triangulation satisfy the property that no other point is contained within the circumcircle (circumsphere) formed by the nodes of the triangle (tetrahedron). Although this and other properties of Delaunay triangulations have been studied for some time, practical triangulation procedures have only appeared in the last decade [25–46, 69–72]. Most of these are based on Bowyer's algorithm [26], which is summarized here for 3-D applications (see Fig. 2):

B1. Define the convex hull within which all points will lie. This can either be a single large tetrahedron (4 points), or a brick (8 points) subdivided into five tetrahedra.

B2. Introduce a new point $\boldsymbol{x}_{n+1}$.

B3. Find all tetrahedra LEDEL (1:NEDEL) whose circumsphere contains $\boldsymbol{x}_{n+1}$; these are the tetrahedra that will be deleted.

B4. Find all the points belonging to these tetrahedra.

B5. Find all the external faces LFEXT (1:NFEXT) of the void that results when deleting these tetrahedra.

B6. Form new tetrahedra by connecting the NFEXT external faces to the new point $\boldsymbol{x}_{n+1}$.

B7. Add the new elements and the points to their respective lists.

B8. Update all data structures.

B9. If more points are to be introduced, go to B.2.

The algorithm described above assumes a given point distribution. The specification of a point distribution (in itself a tedious task) may be replaced by a general specification of desired element size and shape in space. The key idea, apparently proposed independently by Frey [73] and Holmes [36], is to check the discrepancy between the desired and the actual element shape and size

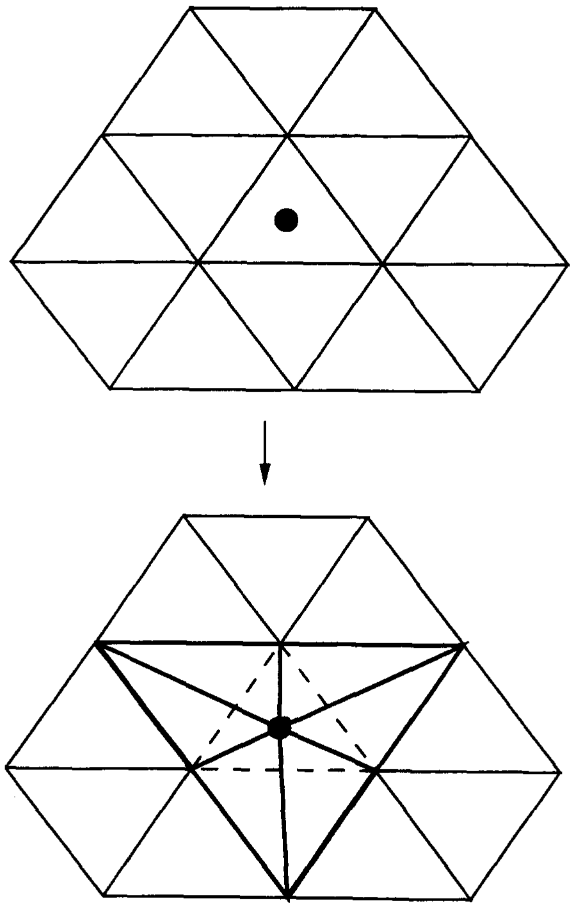


Fig. 2. Delaunay triangulation (2-D).

of the current mesh. Points are then introduced in those regions where the discrepancy exceeds a user-defined tolerance. An automatic Delaunay mesh generator of this kind proceeds as follows:

D1. Assume given a boundary point distribution.

D2. Generate a Delauney triangulation of the boundary points.

D3. Using the information stored on the background grid and the sources, compute the desired element size and shape for the points of the current mesh.

D4. Introduce new points as required.

D5. If new points were introduced:

- Perform a Delaunay Triangulation for the new points;
- Go to D.3

D6. Recover the surface mesh.

Important aspects of Delauney triangulators that require special care include:

- Circumsphere calculations, particularly if roundoff is a problem [33];
- Avoidance of 'sliver elements' [30, 41, 45, 46]; and
- Boundary recovery [33, 69, 39, 74, 41, 42, 75].

The complexity of the Delaunay triangulation technique is of $O(N\log(N))$, where N denotes the number of elements. Over the years, optimal data structures have been implemented to realize such a favourable scaling [41–43, 45, 46]. The procedure has been used extensively to grid large-scale complex geometry domains [76, 41–43, 72, 45, 46] and within adaptive remeshing procedures [37, 72]. Sustained speeds in excess of 250 Ktet/min have been achieved on the CRAY-YMP [41–43, 45, 46], and the procedure has been ported to parallel machines [43]. In some cases, the Delaunay circumsphere criterion is replaced by or combined with a max(min) solid angle criterion [70, 71, 78, 45, 46], which has been shown to improve the quality of the elements generated. For these techniques, a 3-D edge-swapping technique is used to speed up the generation process.

2.2.1. *Front-based point introduction*

When comparing 2-D grids generated by the advancing front or the Delauney technique, the most striking difference lies in the appearance of the grids. The Delauney grids always look more 'ragged' than the advancing front grids. This is because the grid connectivity obtained from Delauney triangulations is completely free, and the introduction of points in elements does not allow a precise control. In order to improve this situation, several authors [79–81, 44–46] have tried to combine the two methods. These methods are called advancing front Delaunay, and can produce extremely good grids that satisfy the Delaunay or max(min) criterion.

3. Specification of element size and shape

As the versatility, speed and generality of unstructured grid generators increased, the specification of the desired element size and shape in space became a problem. This is because the requirements for simplicity (low user input) and flexibility (complex geometries) are conflicting. The earliest automatic unstructured grid generators employed a background grid for simple geometries. This was particularly suited for adaptive remeshing procedures. For CAD-based surface descriptions, the modified quad- and octree techniques provide an automatic way of refining the mesh in regions of high surface curvature [47, 83, 48]. This works well for problems that require a fine mesh in regions of high surface curvature, and a coarser mesh away from surfaces. While this is indeed the case for many elliptic problems, a user may still wish to refine the mesh in some spatial region away from surfaces (e.g. a heat-source, an oblique shock in supersonic flow, etc.). Therefore, alternate ways to prescribe element size and shape in space, that combine generality and low user input, are required.

3.1. *Sources*

A flexible way that combines the smoothness of functions with the generality of boxes or other discrete elements is to define sources. Indeed, a number of authors have proposed the use of sources in recent years [83, 41, 67, 68]. The element size for an arbitrary location $\boldsymbol{x}$ in space is given as a function of the closest distance to the source $r(\boldsymbol{x})$. Consider first the line source given by the points $\boldsymbol{x}_1, \boldsymbol{x}_2$ shown in Fig 3. The vector $\boldsymbol{x}$ can be decomposed into a portion lying along the line, and the normal to it. With the notation of Fig. 3, we have

$$\boldsymbol{x} = \boldsymbol{x}_1 + \xi \boldsymbol{g}_1 + \alpha \boldsymbol{n}, \qquad (2)$$

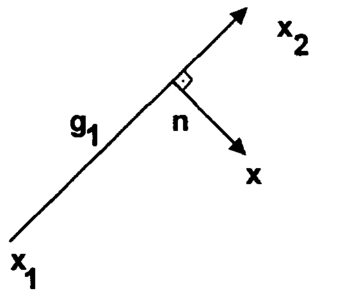


Fig. 3. Line source.

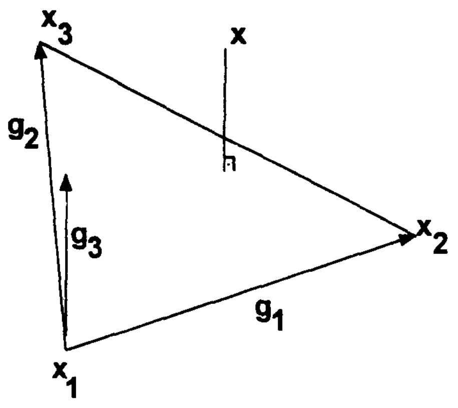


Fig. 4. Surface source.

The ζ-coordinate may be obtained by scalar multiplication with $\boldsymbol{g}_1$, and given by

$$\xi = \frac{(\boldsymbol{x} - \boldsymbol{x}_1)\cdot\boldsymbol{g}_1}{\boldsymbol{g}_1\cdot\boldsymbol{g}_1}. \tag{3}$$

By delimiting the value of ζ to be on the line:

$$\xi' = \max(0, \min(1, \zeta)), \tag{4}$$

the distance between point $\boldsymbol{x}$ and the closest point on the line source is given by

$$\delta(\boldsymbol{x}) = |\boldsymbol{x} - \boldsymbol{x}_1 - \xi'\boldsymbol{g}_1|. \tag{5}$$

Point-sources can be constructed by collapsing both into one. Consider next the surface source element given by the point $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$ shown in Fig. 4. The vector $\boldsymbol{x}$ can be decomposed into a portion lying in the plane given by the surface source-points, and the normal to it. With the notation of Fig. 4, we have

$$\boldsymbol{x} = \boldsymbol{x}_1 + \xi\boldsymbol{g}_1 + \eta\boldsymbol{g}_2 + \gamma\boldsymbol{g}_3, \tag{6}$$

where

$$\boldsymbol{g}_3 = \frac{\boldsymbol{g}_1 \times \boldsymbol{g}_2}{|\boldsymbol{g}_1 \times \boldsymbol{g}_2|}, \tag{7}$$

By using the contravariant vectors $\boldsymbol{g}^2, \boldsymbol{g}^2$, where $\boldsymbol{g}_1 \cdot \boldsymbol{g}^j = \delta_i^j$, we have

$$\xi = (\boldsymbol{x} - \boldsymbol{x}_1)\cdot\boldsymbol{g}^1, \quad \eta = (\boldsymbol{x} - \boldsymbol{x}_1)\cdot\boldsymbol{g}^2, \quad \zeta = 1 - \xi - \eta. \tag{8}$$

Whether the point $\boldsymbol{x}$ lies 'on the surface' can be determined by the conditions:

$$0 \leqslant \xi, \eta, \zeta \leqslant 1. \tag{9}$$

If this condition is violated, point $\boldsymbol{x}$ is closest to any of the given edges, and the distance to the surface is evaluated by checking the equivalent line-source associated with the edges. If, on the other hand, Eq. (8) is satisfied, the closest distance between the surface and the point is given by

$$\delta(\boldsymbol{x}) = |(1 - \xi - \eta)\boldsymbol{x}_1 + \xi\boldsymbol{x}_2 + \eta\boldsymbol{x}_3 - \boldsymbol{x}|. \tag{10}$$

As one can see, the number of operations required to determine $\delta(\boldsymbol{x})$ is not considerable if one pre-computes and stores the geometrical parameters of the sources ($\boldsymbol{g}_i, \boldsymbol{g}^i$, etc.). In order to reduce the internal complexity of a code, it is advisable to only work with one type of source. Given that the most general source is the surface source, line- and point-sources are prescribed as surface sources, leaving a small distance between the points to avoid numerical problems (e.g. divisions by zero).

Having defined the distance from the source, the next step is to select a function that is general yet requires a minimum amount of input to define the element size as a function of distance. Typically, the user desires a small element size close to the source, and a large element size away from it. Moreover, the element size should, in many instances, be constant (and small) in the vicinity $r < r_0$ of the source. An elegant way to satisfy these requirements is to work with functions of the transformed variable

$$\rho = \max\left(0, \frac{r(\boldsymbol{x}) - r_0}{r_1}\right). \tag{11}$$

For obvious reasons, the parameter r_1 is called the scaling length. Commonly used functions of ρ used to define the element size in space are:

(a) *Power laws*: given by expressions of the form [83]

$$\delta(\boldsymbol{x}) = \delta_0[1 + \rho^{\gamma}], \tag{12}$$

with the four input parameters $\delta_0, r_0, r_i \gamma$; typically, $1.0 \leqslant \gamma \leqslant 2.0$.

(b) *Exponential functions*: which are of the form [41–43]

$$\delta(\boldsymbol{x}) = \delta_0 e^{\gamma\rho}, \tag{13}$$

with the four parameters δ_0, r_0, r_1, γ.

(c) *Polynomial expressions*: which avoid the high cost of exponents and logarithms by employing expressions of the form:

$$\delta(\boldsymbol{x}) = \delta_0\left[1 + \sum_{i=1}^{n} a_i\rho^i\right], \tag{14}$$

with the $n + 3$ parameters δ_0, r_0, r_1, a_i. We have found that in practice quadratic polynomials are sufficient, i.e. $n = 2$.

Obviously, many other functions may be considered. As a matter of fact, in the hands of experienced users all of them will lead to similar grids. Given a set of m sources, the minimum element size computed for each of them is taken whenever an element is to be generated:

$$\delta(\boldsymbol{x}) = \min(\delta_1, \delta_2, \ldots \delta_m). \tag{15}$$

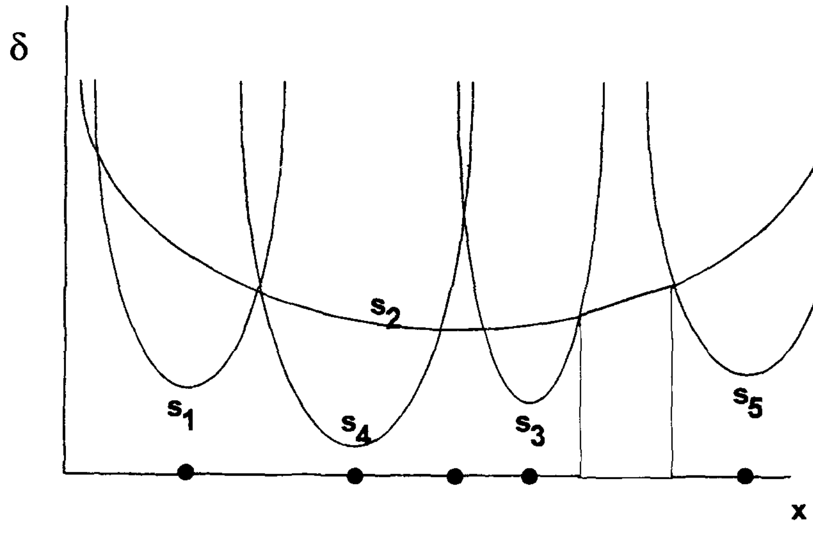


Fig. 5. Localization with sources: Closest source(s) may not be the one(s) Determining Element sizes.

Sources offer a convenient and general way to define the desired element size in space. They may be introduced rapidly on a workstation in interactive mode with a mouse-driven menu once the surface data are available. For moving or tumbling bodies (e.g. store separation), the points defining the sources relevant to them may be synchronized in their movement to the movement of the respective body. This allows high quality remeshing when required for this case of problems [58, 51, 55]. On the other hand, sources suffer from one major disadvantage: at every instance, the generation parameters of *all* sources need to be evaluated. For a distance distribution given by Eqs. (10)–(15), it is very difficult to 'localize' the sources in space in order to filter out the relevant ones. As an example, consider the 1-D situation shown in Fig. 5. Although sources S_3, S_5 are closer to the shaded region than any other source, the source that yields the smallest element size δ is not in this region. The evaluation of the minimum distance obtained over the sources may be vectorized in a straightforward way. Nevertheless, a high number of sources ($N_s > 100$) will have a marked impact on CPU-times, even on a vector-machine. It has been our experience that the high number of sources dictated by some complex geometries can lead to situations where the dominant, CPU cost is given by the element-size evaluations of the sources, not the advancing front method itself.

3.2. Element size attached to CAD-data

For problems that require gridding complex geometries, the specification of proper element sizes can become a tedious process. Conventional background grids would involve many tetrahedra, whose generation is a labour-intensive, tedious task. Point, line, or surface-sources are not always appropriate either. Curved 'ridges' between surface patches, as sketched in Fig. 6, may require many line-sources. Similarly, the specification of gridding parameters for surfaces with high curvature may require the many surface-sources. The net effect is that for complex geometries one

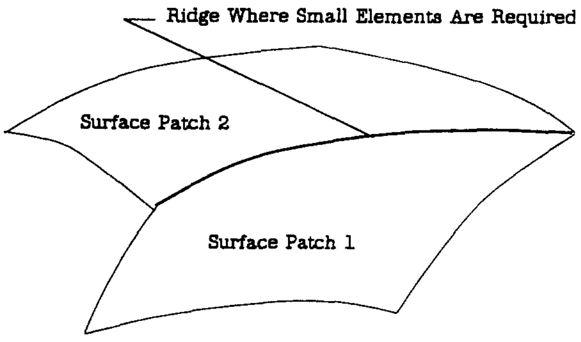


Fig. 6. Element size attached to CAD-data.

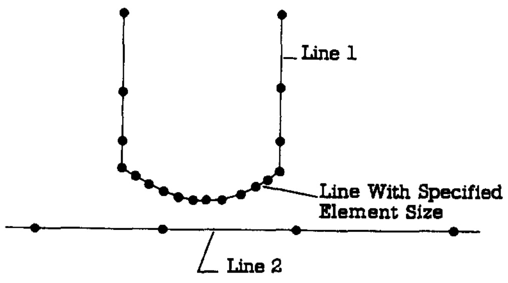


Fig. 7. Potential problems with incompatible element sizes.

is faced with excessive labour-costs (background grids, many sources), and/or CPU requirements during mesh generation (many sources).

A better way to address these problems is to attach element size (or other gridding parameters) directly to CAD-data. For many problems, the smallest elements are required close to the boundary. Therefore, if the element size for the points of the current front is stored, the next element size may be obtained by multiplying it with a user-specified increase factor c_i. The element size for each new point introduced is then taken as the minimum obtained from the background grid δ_{bg}, the sources δ_s and the minimum of the point sizes corresponding to the face being deleted, multiplied by a user-specified increase factor c_i:

$$\delta = \min(\delta_{bg}, \delta_s, c_i \cdot \min(\delta_A, \delta_B, \delta_C)). \tag{16}$$

Typical values for c_i are $1.0 \leqslant c_i \leqslant 1.5$. The first value yields a mesh of uniform element size, whereas the latter gives rise to grids with elements that grow rapidly in size away from the surface. Specifying or attaching element sizes to CAD-data can lead to incompatibilities if surfaces are close to each other. For example, in the situation shown in Fig. 7, specifying a small distance for line L1 without proper modification of the distance-parameter for line L2 can lead to size incompatibilities and badly shaped elements. This happens because this type of specification of element size is ‘hyperbolic’ by nature, starting from the surfaces and marching blindly into the domain. This limitation may be circumvented by the use of adaptive background gridding (see below). A considerable reduction in input requirements for complex geometries may be realized by assigning to the end-points of lines a point-size that is related to the minimum length of the lines surrounding it. As these end-points of lines are ‘inherited’ by the final grid, this is a natural way to assign proper element sizes to regions where small lines occur.

3.3. Adaptive background gridding

As was seen from the previous sections, the specification of proper element size and shape in space can be a tedious, labour-intensive task. Adaptive background grid refinement may be employed in order to reduce the amount of user intervention to a minimum. As with any other mesh refinement scheme, one has to define *where* to refine and *how* to refine. Because of its very

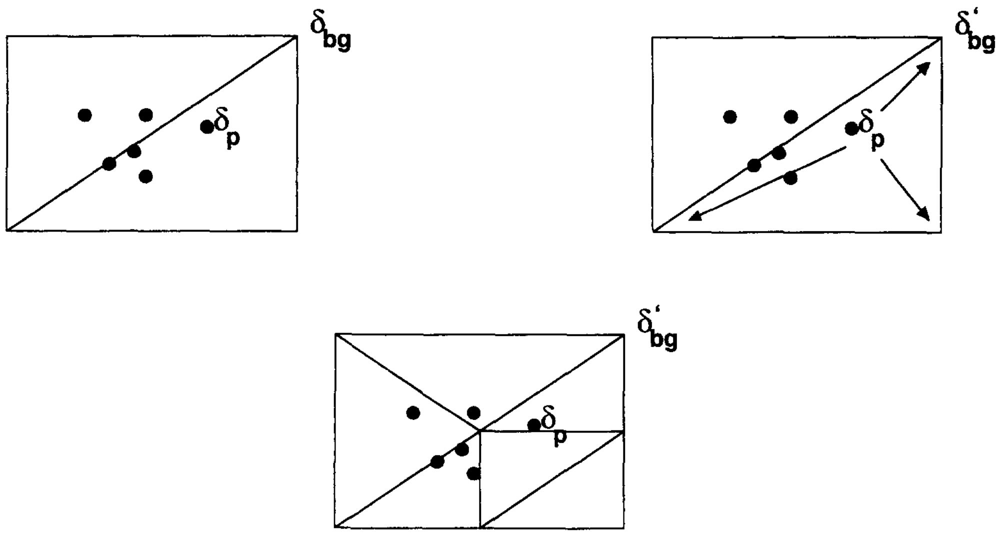


Fig. 8. Background grid adjustment/refinement.

high speed we prefer to refine the background grid via classic h-refinement [84]. In this way the possible interactivity on current workstations is maintained.

The selection as to where to refine the background mesh is made with the following assumptions: (a) points have already been generated;

(b) at each of these points, a value for the characteristic or desired element size δ is given;

(c) for each of these points, the background grid element containing it is known;

(d) a desired increase factor c_i between elements is known.

The refinement selection is then made in two passes (see Fig. 8):

Pass 1: *Background grid adjustment.* Suppose a background grid element has very large element sizes defined at its nodes. If it contains a generated point with characteristic length that is much smaller, an incompatibility is present. The aim of this first pass is to prevent these incompatibilities by comparing lengths. Given a set of two points with coordinates $\boldsymbol{x}_1,\boldsymbol{x}_2$, as well as an element length parameter δ_1, the maximum allowable element length at the second point may be determined from the geometric progression formula:

$$s_n = |\boldsymbol{x}_2 - \boldsymbol{x}_1| = \delta_1 \frac{c_i^{n+1} - 1}{c_i - 1} \tag{17}$$

implying that

$$\delta_2^* = \delta_1 c_i^n = \frac{s_n(c_i - 1) + \delta_1}{c_i}. \tag{18}$$

Given this constraint, we compare for each of the generated points its characteristic length to the element size defined at each of the background grid nodes of the background grid element containing it.

$$\delta_{bg} = \min(\delta_{bg}, \delta_p^*(x_{bg} - x_p)). \tag{19}$$

Pass 2: Selection of elements to be refined. After the element lengths at the points of the background grid have been made compatible with the lengths of the actual points, the next step is to decide where to refine the background grid. The argument used here is that if there is a significant difference between the element size at generated points and the points of the background grid, the element should be refined. This simple criterion is expressed as:

$$\min_{a,b,c,d}(\delta_{bg}) > c_f \delta_p \Rightarrow \text{refine}, \tag{20}$$

where, typically $1.5 \leqslant c_f \leqslant 3$. All elements flagged by this last criterion are subdivided further via classic h-refinement [84], and the background grid variables are interpolated linearly for the newly introduced points.

3.4. Surface gridding with adaptive background grids

The described background grid adaptation may be used to automatically generate grids that represent the surface within a required or prescribed accuracy. Consider, as a measure for surface accuracy, the angle variation between two adjacent faces. With the notation defined in Fig. 9, the angle between two faces is given by

$$\frac{h}{2r} = \tan\left(\frac{\alpha}{2}\right). \tag{21}$$

This implies that for a given element size h_g and angle α_g, the element size for a prescribed angle α_p should be

$$h_p = h_g \frac{\tan\left(\frac{\alpha_p}{2}\right)}{\tan\left(\frac{\alpha_g}{2}\right)}. \tag{22}$$

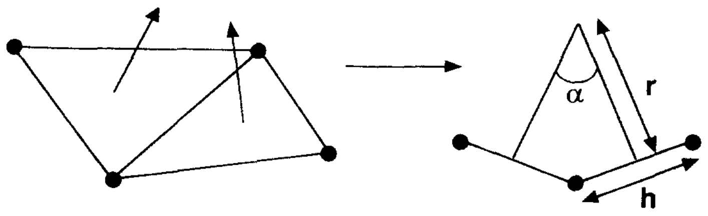


Fig. 9. Measuring surface fidelity/curvature.

For other measures of surface accuracy, similar formulae will be encountered. Given a prescribed angle α_p, the point-distances for the given surface triangulation are compared to those obtained from Eq. (21) and reduced appropriately:

$$\delta_i = \min(\delta_i, h_p). \tag{23}$$

These new point-distances are then used to adjust and/or refine the background grid further, and a new surface triangulation is generated. This process is repeated until the surface representation is accurate enough.

Adaptive background grids combine a number of advantages:
- possible use in combination with CAD-based element size specification and/or background sources;
- automatic gridding to specified surface deviation tolerance;
- smooth transition between surface-faces of different size;
- smooth transition from surface-faces volume-elements.

Thus, they may be used to arrive at minimal-input grid generators, an outstanding goal over the last decade.

4. Surface definition

Having solved the automatic grid generation problem as such, as well as the minimization of labour for element size and shape specification, the final problem to tackle is the rapid definition of geometries and boundary conditions desired for the problem at hand. There are two possible ways of describing the surface of a computation domain:
(a) using analytic functions, and
(b) via discrete data.

The first way is the preferred choice if a CAD–CAM data base exists for the description of the domain, and has been used almost exclusively to date. Splines, B-Splines, NURBS Surfaces [85] or other types of functions are used to define the surface of the domain. An important characteristic of this approach is that the surface is continuous, ie. there exist no 'holes' in the information. While generating elements on the surface, the desired element size and shape is taken into consideration via mappings [11–13, 20, 21]. The second possibility, i.e. a definition of surfaces via discrete data, becomes attractive when no CAD–CAM data base exists. The data may be defined simply by a cloud of points (e.g.remote sensing data, data from digitizers [79, 86], or medical data sets [87]) or via an existing triangulation (e.g. geodesic data, multi-disciplinary simulation models [86]). If a cloud of points is provided as the starting data, the surface definition is completed by triangulating it. This triangulation process is far from trivial and has been the subject of major research and development efforts (see, e.g. [34, 87, 89]). The re-triangulation required for surfaces defined in this way has been treated in [89, 90].

The present incompatibility of formats from the description of surfaces currently makes the surface definition by far the most man-hour intensive task of the CFD, CEM, CSD or CTD analysis process. Grid generation, field solvers and visualization are processes that have been automated to a high degree. This is not the case with surface definition, and may not be so for a long time. It may also have to do with the nature of analysis: for a typical field solver run a vast portion of CAD–CAM data also has to be filtered out (e.g. nuts, bolts, rivets, etc. are not required for the surface definition required of a standard CFD run), and the surface patches used by the designers seldomly match, leaving gaps or overlap regions that have to be treated manually.

5. Adaptive remeshing

Having reviewed some of the aspects required for automatic unstructured grid generators, one may exploit this enabling technology further. Obvious candidates include remeshing for field simulations with several moving or deforming objects, and remeshing of existing data sets, in particular adaptive remeshing. The key idea is to use the existing grid and solution, combined with a proper error indicator or estimator, to generate a mesh that is more suited to the problem at hand. The current mesh becomes the background mesh for the new (adapted) mesh. The steps required for one *adaptive remeshing* are as follows:

- Obtain the error indicator matrix for the gridpoint of the present grid.
- Given the error indicator matrix, get the element size, element stretching and stretching direction for the new grid.
- Using the old grid as the background grid, remesh the computational domain using an unstructured grid generator.
- If further levels of global h-refinement are desired, refine the new grid globally.
- Interpolate the solution from the old grid to the new one.

For adaptive remeshing using the advancing from method, see [8, 9, 11, 14, 17, 51. 52, 59–63, 91–98]; using Delaunay triangulations, see [37, 77, 45, 46]; and using the modified quad/octree approach, see [47, 48].

6. Examples

This section gives a cross-section of gridding examples. It is not meant to be an exhaustive sampling. Given the author's background, all the grids were generated using the Advancing Front Method, and were used for CFD simulations. For some large-scale CFD applications using unstructured grids, the reader is referred to [50–52, 56, 60–104].

(a) *Generic hypersonic vehicle*: This case shows the combination of discrete and analytically defined surfaces to obtain rapid turnaround in preliminary design calculations. The airplane fuselage is given from a structural dynamics calculation and shown in Fig. 10(a). The recovered discrete surface patches, together with the added outer box and some further analytical patches for nozzle entry and exit planes, is shown in Fig 10(b). The new surface discretization is shown in Fig 10(b). The new surface discretization, suitable for preliminary aerodynamic design calculations, is shown in Fig. 10(c). The generated grid had approximately 1 Mtet.

(b) *Truck*: This configuration shows how techniques described above may be combined in order to grid geometrically complex regions. The problem considered is the generation of a tetrahedral grid surrounding a 5 ton army truck. This mesh should be adequate for aerodynamic analysis simulating a shock impacting on the truck. The CAD description, shown in Fig. 11(a), consisted of 6306 points, 3718 lines and 1604 surface segments. An initial coarse background grid, as well as

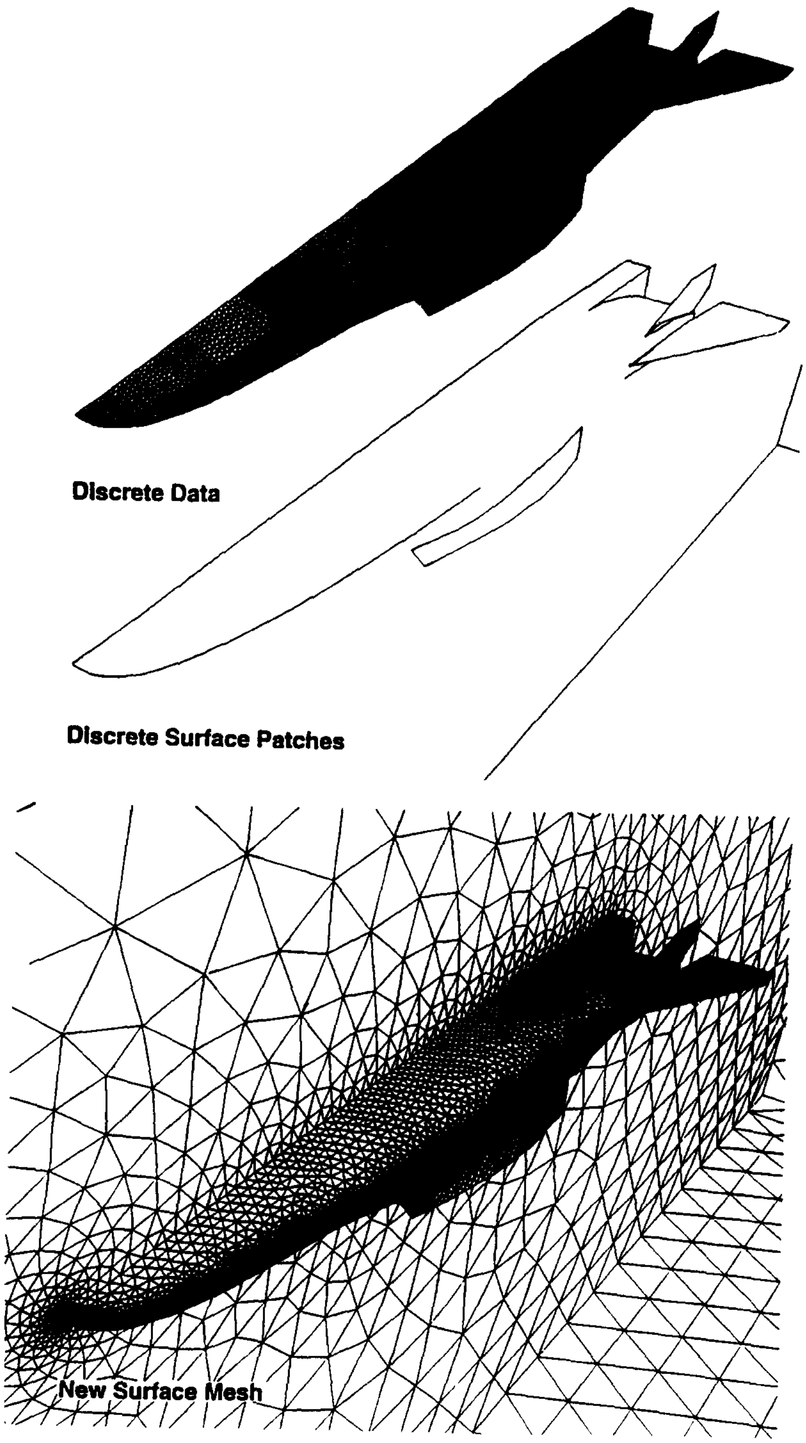
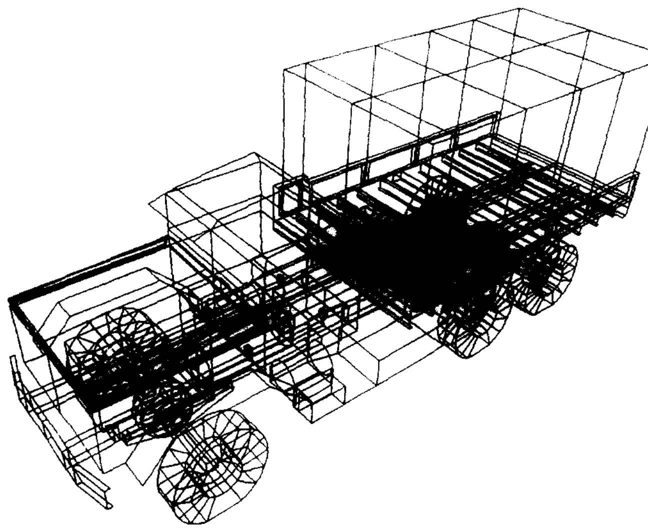


Fig. 10 Generic hypersonic vehicle.



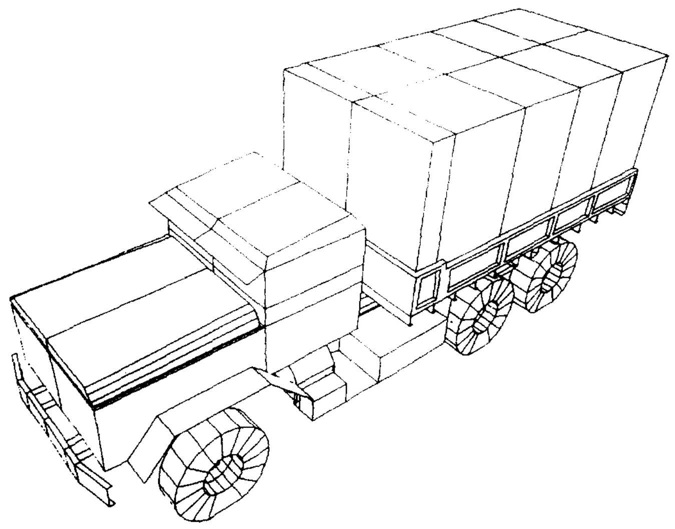
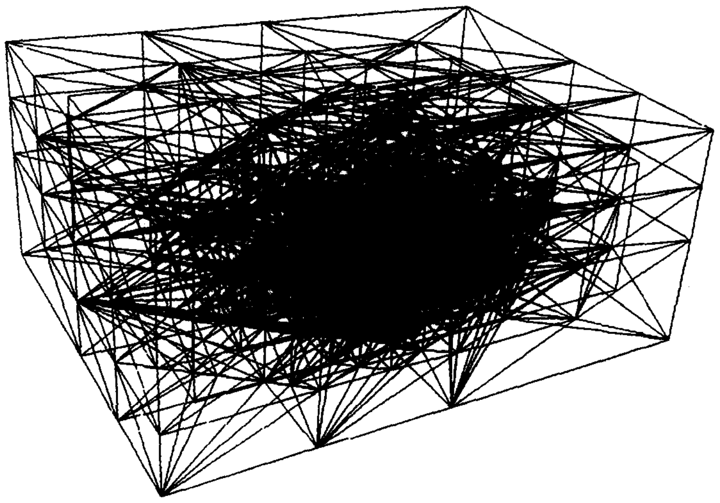


Fig. 11. Truck.

a pair of surface sources were provided initially to specify element size in space. The CAD-data were analysed, and the point-sizes specified were set to be commesurate with the length of the individual lines. This was of great help in gridding the undercarriage section of the truck, and without this capability, the equivalent of several dozen sources would have been required to obtain an adequate mesh. The background grid was adapted six times. The surface of the final mesh, which consisted of 38 K boundary points, 192 Kpts and approximately 1 Mtet, is shown in Fig 11(b). Grids consisting of quad-shells and beams were also generated for the structural members. As



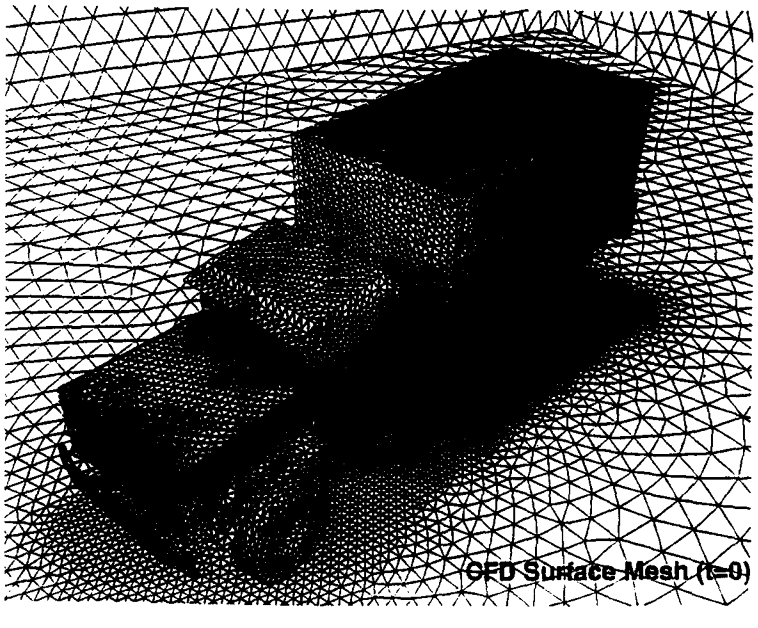


Fig. 11. Continued.

before, the element size was set to be commesurate with the length of the individual lines. The surface triangulation obtained in this way (Fig. 11(c)) was modified into a mixed quad-triangle mesh (Fig. 11(d)). With one level of global h-refinement, an all-quad mesh is obtained (Fig. 11(e)). For more details, see [100].

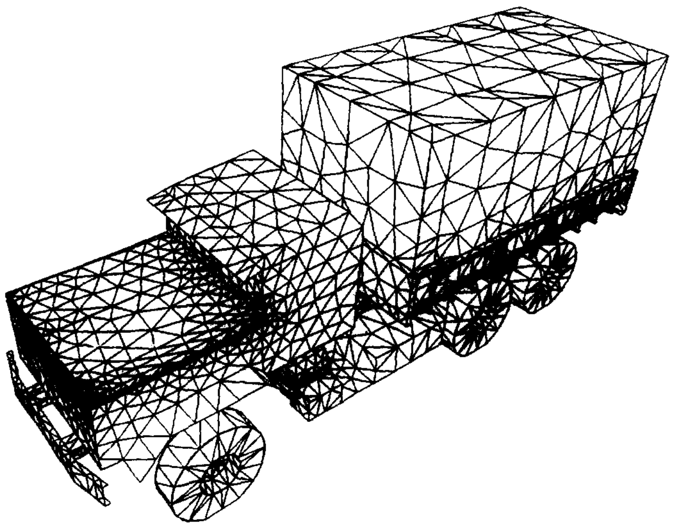
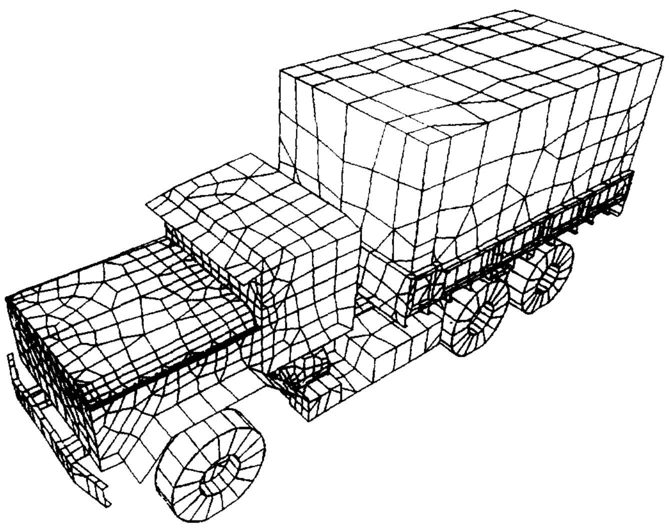
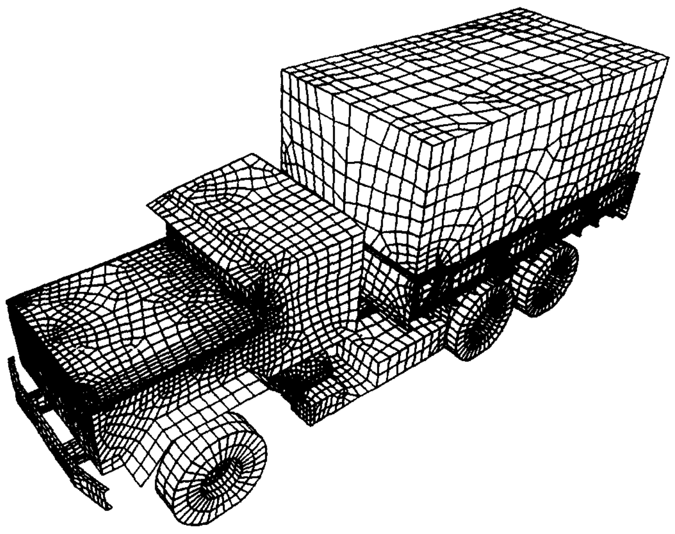



Fig. 11. Continued.

7. Conclusion

Automatic unstructured grid generators have advanced rapidly in the last decade. Both the advancing front method and the Delauney triangulation technique have yielded reliable and versatile gridding tools that may be used routinely in CFD, CSD, CEM and CTD applications. The combination of CAD-attached element size, sources, and adaptive background grids has led to a dramatic reduction of input requirements, speeding up the analysis cycle considerably. Automatic



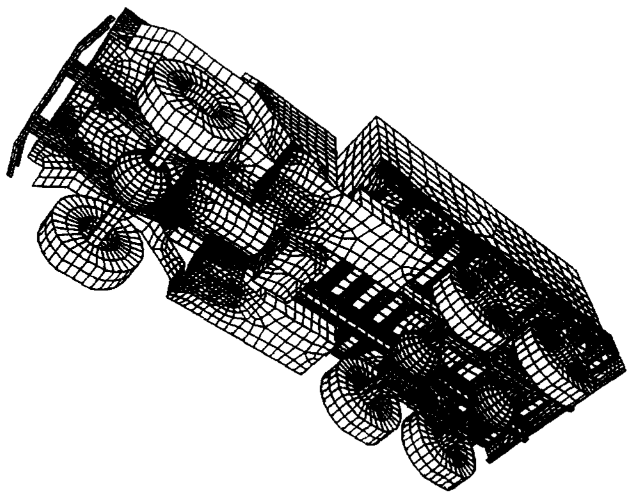


Fig. 11. Continued.

grid generators have been used for simulations that require remeshing (e.g. those with moving bodies), as well as adaptive remeshing procedures.

Current research is focusing on:

- further reductions of input requirements;
- automatic gridding for Navier–Stokes applications;
- wake gridding and adaptive wake remeshing; and
- better post-processing techniques for unstructured grids.

Acknowledgements

I would to acknowledge many fruitful and stimulating discussions during the course of the years with Profs. O.C. Zienkiewicz and K. Morgan from Swansea, Prof. J. Peraire from MIT, as well as Drs. J.P. Boris, D.L. Book, S.T. Zalesak, R. Ramamurti from NRL-LCP&FD, Dr. M.D. Salas from NASA LARC/ICASE, Drs. J.D. Baum and H. Luo from SAIC, and the former and present members of my research team: Drs. Jean Cabello, B. Petitjean, D. Martin, E. Mestreau, Chi Yang and J. Cebral.

The support of both CRAY Research in the form of many free hours on a CRAY-2 over the years, and IBM for providing a RISC-6000/550 for private use is also gratefully acknowledged.

References

[1] S. Sengupta, J. Häuser, P.R. Eiseman and J.F. Thompson (eds.), *Proc. 2nd Int. Conf. Numerical Grid Generation in Computational Fluid Dynamics*, Pineridge Press, Swansea, 1988.

[2] A.S. Archilla, J. Häuser, P.R. Eiseman and J.F. Thompson (eds.) *Proc. 3rd Int. Conf. Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, North-Holland, Amsterdam, 1991

[3] N. Weatherill, P.R. Eiseman and J.F. Thompson (eds.), *Proc. 4th Int. Conf. Numerical Grid Generation in Computaional Fluid Dynamics and Related Fields*, Pineridge Press, Swansea, 1993.

[4] Sandia National Laboratories, *Proc. 3rd International Meshing Roundtable*, 1994.

[5] Sandia National Laboratories, *Proc. 4th International Meshing Roundtable, 1995.*

[6] N. van Phai, "Automatic mesh generation with tetrahedron elements", *Int. J. Num. Meth. Eng.* **18**, pp. 237–289, 1982.

[7] S.H. Lo, "A new mesh Generation scheme for arbitrary planar domains", *Int. J. Num. Meth. Eng.* **22**, pp. 1403–1426, 1985.

[8] J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz, "Adaptive remeshing for compressible flow computations", *J. Comp. Phys.* **72**, pp. 449–466, 1987.

[9] J. Peraire, J. Peiro, L. Formaggia, K. Morgan and O.C. Zienkiewicz, "Finite element Euler calculations in three dimensions", *Int. J. Num. Eng.* 26, pp. 2135–2159, 1988.

[10] R. Löhner, "Some useful data structures for the generation of unstructured grids", *Comm. Appl. Num. Mech.* **4**, pp. 123–135, 1988.

[11] R. Löhner and P. Parikh, "Three-dimensional grid generation by the advancing front method", *Int. J. Num. Meth. Fluids* **8**, pp. 1135–1149, 1988.

[12] S.H. Lo, "Finite element mesh generation over curved surfaces", *Comp. Sttruc.* **29**, pp. 731–742, 1988.

[13] J. Peiro, J. Peraire and K. Morgan, "The generation of triangular meshes on surfaces", *Proc. POLYMODEL XII Conf.*, Newcastle upon Tyne, May 23–24, 1989.

[14] J. Peraire, K. Morgan and J. Peiro, "Unstructured finite element mesh generation and adaptive procedures for CFD", AGARD-CP-464, pp. 18, 1990.

[15] F. Huet, "Génération de Maillage Automatique dans les Configurations Tridimensionelles Complexes Utilisation d'une Méthode de 'Front'", AGARD-CP-464, **17**, 1990.

[16] J.Z. Zhu, O.C. Zienkiewicz, E. Hinton and J. Wu, "A new approach to the development of automatic quadrilateral grid generation", *Int. J. Num. Meth. Eng.* **361**, pp. 1805–1823, 1993.

[17] R. Löhner, J. Camberos and M. Merriam, "Parallel unstructured grid generation", *Comp. Meth. Appl. Mech. Eng.* **95**, pp. 343–357, 1992.

[18] H. Jin and R.I. Tanner, "Generation of unstructured tetrahedral meshes by the advancing front technique", *Int. J. Num. Meth. Eng.* **31**, pp. 987–997, 1991.

[19] J. Frykestig, "Advancing front mesh generation techniques with application to the finite element method", Pub. 94:10, Chalmers University of Technology; Göteborg, Sweden, 1994.

[20] K. Nakahashi and D. Sharov, "Direct surface triangulation using the advancing front method", AIAA-95-1686-CP, 1995.
[21] C.-J. Woan, "Unstructured surface grid generation on unstructured quilt of patches", AIAA-95-2202, 1995.
[22] T.D. Blacker and M.B. Stephenson, "Paving: A new approach to automated quadrilateral mesh generation", *Int. J. Num. Meth. Eng.* **32**, pp. 811–847, 1992.
[23] T.D. Blacker and R.J. Meyers, "Seams and wedges in plastering: A 3-D hexahedral mesh generation algorithm", *Eng. Comput.* **9**, pp. 83–93, 1993.
[24] J.C. Cavendish, "Automatic triangulation of arbitrary planar domains for the finite element method", *Int. J. Num. Meth. Eng.* **8**, pp. 679–696, 1974.
[25] D.T. Lee and B.J. Schachter, "Two algorithms for constructing a Delaunay triangulation", *Int. J. Comp. Inf. Sci.* **9**, pp. 219–242, 1980.
[26] A. Bowyer, "Computing Dirichlet tesselations", *The Comput. J.* **24**, pp. 162–167, 1981.
[27] D.F. Watson, "Computing the *N*-dimensional Delaunay tesselation with application to Voronoi polytopes", *The Comput. J.* **24**, 2, pp. 167–172, 1981.
[28] M. Tanemura, T. Ogawa and N. Ogita, "A new algorithm for three-dimensional Voronoi tesselation", *J. Comp. Phys.* **51**, pp. 191–207, 1983.
[29] S.W. Sloan and G.T. Houlsby, "An implementation of Watson's algorithm for computing 2-dimensional Delaunay triangulation", *Adv. Eng. Software* **6**, pp. 192–197, 1984.
[30] J.C. Cavendish, D.A. Field and W.H. Frey, "An approach to automatic three-dimensional finite element generation," *Int. J. Num. Meth. Eng.* **21**, pp. 329–347, 1985.
[31] R.C. Kirkpatrick, "Nearest Neighbor Algorithm", in: M.J. Fritts, W.P. Crowley and H. Trease (eds.) *Springer Lecture Notes in Physics* 238, Springer Berlin, (1995) pp. 1037–1057.
[32] D.N. Shenton and Z.J. Cendes, "Three-dimensional finite element mesh generation using Delaunay tesselations", *IEEE Trans. on Magnetics*, MAG-21, pp. 2535–2538 1995.
[33] T.J. Baker, "Three-dimensional mesh generation by triangulation of arbitrary point sets", AIAA-CP-87-1124, *8th CFD Conf.*, Hawaii, 1987.
[34] B.K. Choi, H.Y. Chin, Y.I. Loon and J.W. Lee, "Triangulation of scattered data in 3D space", *Comp. Aided Geom. Des.* 20, pp. 239–248, 1988.
[36] D.G. Holmes and D.D. Snyder, "The generation of unstructured triangular meshes using Delaunay triangulation", in: *Numerical Grid Generation in Computational Fluid Dynamics*, Pineridge Press, Swansea, 1988, Sengupta et al. eds., pp. 643–652.
[37] D.J. Mavriplis, "Adaptive mesh generation for viscous flows using Delaunay triangulation", *J. Comp. Phys.* **90**, pp. 271–291, 1990.
[38] P.L. George, F. Hecht and E. Saltel, "Fully automatic mesh generation for 3D domains of any shape", *Impact of Comput. Sci. Eng.* **2**, pp. 187–218, 1990.
[39] P.L. George, F. Hecht and E. Saltel, "Automatic mesh generator with specified boundary", *Comp. Meth. Appl. Mech. Eng.* **92**, pp. 269–288, 1991.
[40] P.L. George and F. Hermeline, "Delaunay's mesh of a convex polyhedron in dimension d. Application to arbitrary polyhedra", *Int. J. Num. Meth. Eng.* **33**, pp. 975–995, 1992.
[41] N.P. Weatherill, "Delaunay triangulation in computation fluid dynamics", *Comp. Math. Appl.* **24**, 5/6, pp. 129–150, 1992.
[42] N.P. Weatherill and O. Hassan, "Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints", *Int. J. Num. Meth. Eng.* **37**, pp. 2005–2039, 1994.
[43] N.P. Weatherill, "The Delaunay triangulation – from the early work in Princeton", Chapter 6 in *Frontiers of Computational Fluid Dynamics 1994* (D.A. Caughey and M.M. Hafez eds.), J. Wiley & Sons, 1994.
[44] J. Müller, P.L. Roe and H. Deconinck, "A frontal approach for internal node generation in Delaunay triangulations", *Int. J. Num. Meth. Eng.* **17**, pp. 241–256, 1993.
[45] D.L. Marcum, "Generaion of unstructured grids for viscous flow application", AAIA-95-0212, 1995.
[46] D.L. Marcum and N.P. Weatherill, "Unstructured grid generation using iteractive point insertion and local reconnection", *AIAA* J. **33**, 9, pp. 1619–1625, 1995.
[47] M.A. Yerry and M.S. Shepard, "Automatic three-dimensional mesh generation by the modified-octree technique, *Int. J. Num. Meth. Eng.* **20**, pp. 1965–1990, 1984.

[48] M.S. Shepard and M.K. Georges, "Automatic three-dimensional mesh generation by the finite octree technique", *Int. J. Num. Meth. Eng.* **32**, pp. 709–749, 1991.
[49] J. Bonet and J. Peraire, "An alternate digital tree algorithm for geometric searching and intersection problems", *Int. J. Num. Meth. Eng.* **31**, pp. 1–17, 1991.
[50] J.D. Batum and R. Löhner, "Numerical simulation of shock interaction with a modern main battlefield tank", AIAA-91-1666, 1991.
[51] J.D. Baum and R. Löhner, "Numerical simulation of pilot/seat ejection from an F-16", AIAA-93-0783, 1993.
[52] J.D. Baum, H. Luo and R. Löhner, "Numerical simulation of a blast inside a Boeing 747", AIAA-93-3091, 1993.
[53] H. Luo, J.D. Baum, R. Löhner, "Edge-based finite element scheme for the Euler equations", *AIAA J.* **32**, pp. 1183–1190, 1994.
[54] J.D. Baum, H. Luo and R. Löhner, "Numerical simulation of blast in the world trade center", AIAA-95-0085, 1995.
[55] J.D. Baum, H. Luo and R. Löhner, "Validation of a new ALE, adaptive unstructured moving body methodology for multi-store-simulations", AIAA-95-1792, 1995.
[56] E. Mestreau and R. Löhner, "Numerical simulation of chip cooling via large-scale FEM simulations", CSI-GMU Preprint, 1994.
[57] R. Löhner, "An adaptive finite element scheme for transient problems in CFD", *Comp. Meth. Appl. Mech. Eng.* **61**, pp. 323–338, 1987.
[58] R. Löhner, "Three-dimensional fluid–structure interaction using a finite element solver and adaptive remeshing", *Comp. Sys. in Eng.* **1**, 2-4, pp. 257–272, 1990.
[59] R. Tilch, PhD Thesis, CERFACS, Toulouse, France, 1991.
[60] J. Peraire, K. Morgan, and J. Peiro, "Adaptive remeshing in 3-D", *J. Comp. Phys.*, 1992.
[61] R. Ramamurthi, K, Kailasanath and R. Löhner, "Numerical simulation of unsteady flow in a nozzle", CPIA Publication 580, Vol. I, Chemical Propulsion Information Agency, 1992 JANNAF Propulsion Meeting 1992, pp. 153–163.
[62] E. Rank, M. Schweingruber and M. Sommer, "Adaptive mesh generation and transformation of triangular to quadrilateral meshes", *Comm. Appl. Num. Meth.* **9**, pp. 121–129, 1993.
[63] F. Llinca, D. Pelletier and F. Arnoux-Guisse, "An adaptive finite element method for turbulent free shear flows", AIAA-95-0473, 1995.
[64] A. Shostko and R. Löhner, "Three-dimensional parallel unstructured grid generation", *Int. J. Num. Meth. Eng.* **38**, pp. 905–925, 1995.
[65] Y. Kallinderis and S. Ward, "Prismatic grid generation with an efficient algebraic method for aircraft configurations", AIAA-92-2721, 1992.
[66] R. Löhner, "Matching semi-structured and unstructured grids for Navier–Stokes calculations", AIAA-93-3348-CP, 1993.
[67] S. Pirzadeh, "Viscous unstructured three-dimensional grids by the advancing-layers method", AIAAA-94-0417, 1994.
[68] S. Pirzadeh, "Progress towards a user-oriented unstructured viscous grid generator", AIAA-96-0031, 1996.
[69] T.J. Baker, "Developments and trends in three-dimensional mesh generation", *Appl. Num. Math.* **5**, pp. 275–304, 1989.
[70] B. Joe, "Construction of three-dimensional Delaunay triangulations using local transformations", *Computer-Aided Geometric Des.* **8**, pp. 123–142, 1991.
[71] B. Joe, "Delaunay versus max-min solid angle triangulations for three-dimensional mesh generation", *Int. J. Num. Meth. Eng.* **31**, pp. 987–997, 1991.
[72] N.P. Weatherill, O. Hassan, M.J. Marchant and D.L. Marcum, "Adaptive inviscid flow solutions for aerospace geometries on efficiently generated unstructured tetrahedral meshes", AIAA-93-3390, 1993.
[73] W.H. Frey, " Selective refinement: A new strategy for automatic node placement in graded triangular meshes," *Int. Num. Meth. Eng.* **24**, pp. 2183–2200, 1987.
[74] P.L. George, *Automatic mesh generation*, Wiley, New York, 1991.
[76] A. Jameson, T.J. Baker and N.P. Weatherill, "Calculation of inviscid transonic flow over a complete aircraft", AIAA-86-0103, 1986.
[77] D.J. Mavriplis, "Turbulent flow calculations using unstructured and adaptive meshes", *Int. J. Num. Meth. Fluids* **13**, pp. 1311–1152, 1991.

[78] T. Barth, "Steiner triangulation for isotropic and stretched elements", AIAA-95-0213, 1995.

[79] M. Merriam, "An efficient advancing front algorithm for Delaunay triangulation", AIAA-91-0792, 1991.

[80] D.J. Mavriplis, "An advancing front Delaunay triangulation algorithm designed for robustness", AIAA-93-0671, 1993.

[81] S. Rebay, "Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer–Watson algorithm", *J. Comp. Phys.* **106**, pp. 125–138, 1993.

[82] K. Nakahashi, "FDM-FEM zonal approach for viscous flow computations over multiple bodies", AIAA-87-0604, 1987.

[83] R. Löhner, "Finite elements in CFD: grid generation, adaptivity and parallelization", *AGARD Rep. 787, Proc. Special Course on Unstructured Grid Methods for Advection Dominated Flows*, Chap. 8, VKI, Belgium, May and NASA Ames, Moffet Field, CA, September 1992.

[84] R. Löhner and J.D. Baum, "Adaptive H-refinement on 3-D unstructured grids for transient problems", *Int. J. Num. Meth. Fluids* **14**, pp. 1407–1419, 1992.

[85] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design*; Academic Press, New York, 1990.

[86] M. Merriam and T. Barth, "3D CFD in a day: The laser digitizer project", AIAA-91-1654, 1991.

[87] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Surface reconstruction for unorganized points", *Comput. Graph*, **26**, pp. 71–78, 1992.

[88] S.H. Lo, "Automatic mesh generation over intersecting surfaces", *Int. J. Num. Meth. Eng* **38**, pp. 943–954, 1995.

[89] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Mesh optimization", *Proc. Comp. Graph. Ann. Conf.* pp. 19–26, 1993.

[90] R. Löhner, "Re-gridding surface triangulations", *J. Comp. Phys.*, to appear 1996.

[91] J.Z. Zhu and O.C. Zienkiewicz, "A simple error estimator and adaptive procedure for practical engineering analysis", *Int. J. Num. Meth.* **24**, 31, pp. 337–357, 1987.

[92] R. Löhner, "An adaptive finite element solver for transient problems with moving bodies", *Comp. Struct.* **30**, pp. 303–317, 1988.

[93] T. Struoboulis and J.T. Oden, "A posteriori error estimation for finite element approximations in fluid mechanics", *Comp. Meth. Appl. Mech. Eng.* **78**, pp. 201–242, 1990.

[94] J.-F. Hètu and D.H. Pelletier, "Adaptive remeshing for viscous incompressible flows", *AIAA J.* **30**, pp. 1986–1992, 1992.

[95] E. Loth, J.D. Baum and R. Löhner, "Formation of shocks within axisymmetric nozzles", *AIAA J.* **30**, pp. 268–270, 1992.

[96] C.J. Huang and S.J. Wu, "Global and local remeshing algorithm for compressible flow", *J. Comp. Phys.* **102**, pp. 98–113, 1992.

[97] T. Strouboulis and K.A. Haque, "Recent experiences with error estimation and adaptivity; Part 1: Review of error estimators for scalar elliptic problems", *Comp. Meth. Appl. Mech. Eng.* **97**, pp. 399–436, 1992.

[98] T. Strouboulis and K.A. Haque, "Recent experiences with error estimation and adaptivity; Part 2: Error estimation for H-adaptive approximations on grids of triangles and quadrilaterals", *Comp. Appl. Mech. Eng.* **97**, 1992.

[99] P.L. Baehmann, M.S. Shepard and J.E. Flaherty, "A posteriori error estimation for triangular and tetrahedral quadratic elements using interior residuals", *Int. J. Num. Meth. Eng.* **34**, pp. 979–996, 1992.

[100] J.D. Baum, H. Luo, R. Löhner, C. Yang, D. Pelessone and C. Charman, "A coupled fluid structure modeling of shock interaction with a truck", AIAA-96-0795, 1996.

[101] R. Löhner, C. Yang, J. Cebral, J.D. Baum, H. Luo, D. Pelessone and C. Charman, "Fluid–structure interaction using a loose coupling algorithm and adaptive unstructured grids", AIAA-95-2259, 1995.

[102] E. Mestreau, R. Löhner and S. Aita, "TGV tunnel-entry simulations using a finite element code with automatic remeshing", AIAA-93-0890, 1993.

[103] E. Mestreau and R. Löhner, "Airbag simulation using fluid/structure coupling", AIAA-96-0798, 1996.

[104] A. Kamoulakos, V. Chen, E. Mestreau and R. Löhner, "Finite element modelling of fluid/structure interaction in explosively loaded aircraft fuselage panels using PAM-SHOCK/PAM-FLOW coupling", *Conf. on Spacecraft Structures, Materials and Mechanical Testing*, Noordwijk, The Netherlands, March, 1996.

[105] E. Boender, "Reliable Delaunay-based mesh generation and mesh improvement", *Comm. Appl. Num. Meth. Eng.* **10**, pp. 773–783, 1994.
[106] J. Bloomenthal, "Polygonalization of implicit surfaces", *Comp. Aided Geometric Des.* 1988.
[107] J. Bloomenthal and K. Ferguson, "Polygonalization of non-manifold implicit surfaces", *Proc. SIGGRAPH*, 1995.
[108] K. Kashiyama and T. Okada, "Automatic mesh generation method for shallow water flow analysis", *J. Int. Num. Meth. Fluids* **15**, pp. 1037–1057, 1992.