COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING, Vol. 12, 683–702 (1996)

EXTENSIONS AND IMPROVEMENTS OF THE ADVANCING FRONT GRID GENERATION TECHNIQUE

RAINALD LÖHNER

GMU/CSI, The George Mason University, Fairfax, VA 22030-4444, U.S.A.

SUMMARY

We describe extensions and improvements to the advancing front grid generation technique that have proven useful over the years. The following areas are treated in detail: situations with thin or crossing surfaces, meshing of surfaces defined by triangulations, and ease of user input to define the desired element size in space. The first extension is important if one considers the generation of volumetric grids around shells, membranes, fabrics, or CAD-data that exhibit cusps. Traditional advancing front generators are likely to fail in these situations. We propose the introduction of a crossing environment variable attached to faces and points in order to filter out undesired or incorrect information during the grid generation process. The second extension is required for situations where the surfaces to be gridded are not defined analytically, but via a triangulation. Typical cases where such triangulations are used to define the domain are geophysical problems, climate modelling and medical problems. The third topic deals with the reduction of manual labour to specify element size in space. Sources, element size attached directly to CAD-data, and adaptive background grids are discussed. Adaptive background grids, in combination with surface deviation tolerances, are used to obtain surface triangulations that represent the geometry faithfully, and at the same time enable a smooth transition to volumetric meshes.

KEY WORDS unstructured grid generation; finite elements; advancing front

1. INTRODUCTION

Unstructured grid generators based on the advancing front technique have gained widespread distribution due to their versatility and speed.[1-12] The basic technique 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 ungridded one is called the front. The algorithm may be summarized as follows:

F1. Define the boundaries of the domain to be gridded.

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, as well as 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.

CCC 1069–8299/96/100683–20

Received 6 August 1994
Revised 1 May 1995

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 F7.2).
  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 F7.3).
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. If there are any faces left in the front, go to F6.

The complexity of the advancing front algorithm 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.[1,3] The procedure has been used extensively to grid large-scale complex geometry domains,[7,10,13–16] within adaptive remeshing procedures,[5,9,10] has been ported to parallel machines,[6,12] and extended for Navier–Stokes problems.[8,11] Sustained speeds in excess of 50,000 tetrahedra/min have been achieved on the CRAY-YMP.[7,13,14]

The present paper discusses extensions and improvements to the basic technique that have proven useful over the years. In particular, three areas are treated in detail: situations with thin or crossing surfaces, meshing of discretely defined surfaces, and ease of user input to define the desired element size in space.

2. THIN/CROSSING SURFACES

In some practical applications, portions of the boundaries to be gridded will come very close together or even cross. Examples where this occurs are the external meshing of thin-walled structures, such as shells (e.g. roofs, walls, etc.), membranes or fabrics (e.g. parachutes, sails, parasols, airbags, etc.), or CAD-data that exhibit cusps (e.g. trailing edges of aerofoils and wings). For these cases, the initial surface triangulation will in all probability exhibit crossing faces and/or duplicate points. The application of the usual advancing front technique to this class of problems is not possible, as there is no mechanism to distinguish between the points that may or may not lead to a crossed front. Suppose that face A in Figure 1 is to be eliminated from the front. Point P will be the point chosen when eliminating this face to form a new element, implying that the 'outside' of the domain to be gridded now contains an element. The occurrence of faces that are extremely close or crossing can very quickly lead to a failure of the advancing front technique, making it impossible to treat these problems on a routine basis.

Possible ways to circumvent this limitation are the definition of multiple domains, or the doubling of faces and points lying on surfaces separating volumes. These two possibilities have been sketched in Figure 2. The obvious disadvantage of the first approach is the increased amount of labour required to specify the domains. This effort can become considerable if

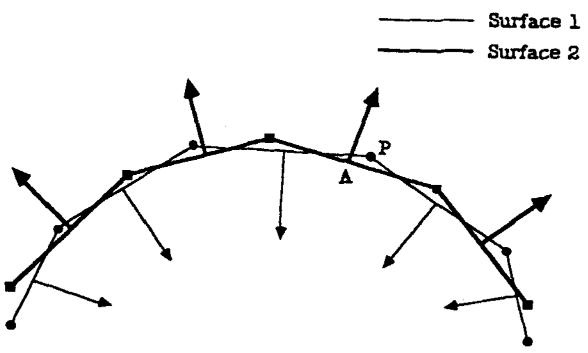


Figure 1. Crossing of faces for thin surface

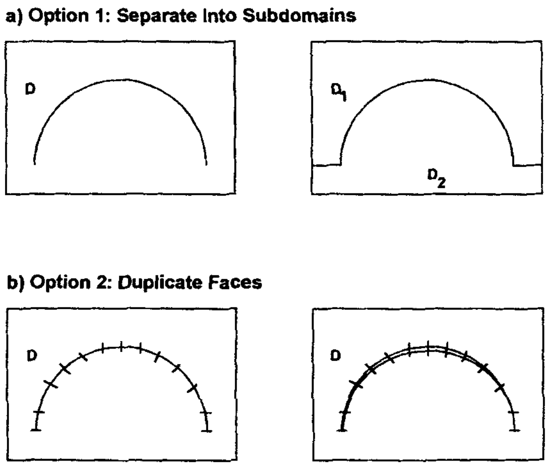

a) Option 1: Separate Into Subdomains
D
D1
D2
b) Option 2: Duplicate Faces
D
D

Figure 2. Other options to treat thin surfaces

additional 'internal fake surfaces' have to be introduced in order to arrive at proper multidomain subdivisions, or original CAD-surfaces have to be split. The second approach, although applicable to a large class of problems, will not solve the more general case shown schematically in Figure 3. In this case, the close or crossing surfaces are not the same. As before, a multidomain approach would involve a significant increase of labour.

The method advocated here is to mark the faces of the initial front with a so-called **crossing environment variable** LFACR(1 : NFACE). This crossing environment variable can either be obtained by checking the initial front for crossing faces, or by marking the different surface patches that comprise a thin structure, fabric, or are crossing, before the surface grid is generated. In the latter case, the surface faces inherit the crossing environment variable from the surface patch they belong to. As an example, the surfaces shown in Figure 4 have been marked with such variables. Notice that not every surface patch has its own environment variable, but that all the surfaces lying on one side of a potentially troublesome region have been given the same environment variable. As will be shown, marking the surface environments enables the advancing front technique to cope even with crossing surface patches.

After marking all the surface faces appropriately, the points are marked according to the environment variable of the faces surrounding them. The points that belong to more than one

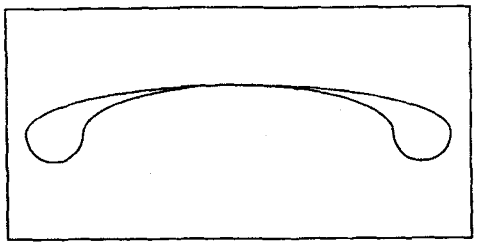


Figure 3. Example face duplication is not applicable

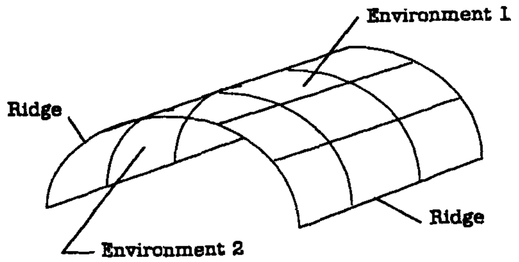


Figure 4. Marking of environments for thin surfaces

environment, such as may occur along the ridges that separate two such regions (see Figure 4), are marked as LPOCR(IPOIN) = −1.

The marked points and faces can be used in a variety of ways to make sure that only the proper points and faces are considered when introducing new elements for a face marked as belonging to a particular environment. The two most important are:

(a) No **points** belonging to any other positive environment are considered. This avoids most of the possible logic mistakes that would lead to failure. For the situation shown in Figure 1, this would eliminate all potentially troublesome points from the list of possible points for face IFOUT, including point P.
(b) No **faces** belonging to any other environment are considered for the face-crossing checks. Keeping the faces that belong to the 'other side' of a thin surface situation would render it impossible to introduce new faces on any of the two sides. This is because some of these faces will always be close enough or crossing the newly formed element, thereby triggering a rejection. For this reason, only the faces belonging to the present environment or no environment (LFACR(IFACE = 0) are kept for front-crossing checks. New (domain) **points** are always assigned the value LPOCR(IPOIN) = 0. For the new **faces**, LFACR(IFACE) is set to the maximum value of LPOCR encountered over the three points IP1, IP2, IP3 belonging to it:

LFACR(IFACE)=MAX(0, LPOCR(IP1), LPOCR(IP2), LPOCR(IP3)).

In this way, all faces touching the surfaces marked as belonging to an environment are marked as well.

In order to grid as straightforwardly as possible the regions immediately adjacent to troublesome surfaces, the faces marked as belonging to an environment are given the highest priority for deletion from the active front. For advancing front generators that choose the face forming the smallest new element as the one to be deleted next, it is a simple matter to prioritize the queue artificially for the marked faces, in order to process them first.[3,4]

3. SURFACE GRIDDING FROM DISCRETE DATA

In many instances, the boundaries of the domains to be gridded are not defined in terms of analytical functions, such as splines, b-splines, or Coon's patches, but in terms of discrete faces and points. Applications where this is frequently the case are:

(a) climate modelling, where the surface of the earth is available from remote sensing data,
(b) groundwater and seepage modelling, where the geological layers have been obtained from drill data or seismic analysis,
(c) medical problems, where the patient data have been obtained from CAT scans,
(d) structural response simulations with large deformation, such as forging simulations, where remeshing or rezoning is required to regularize the grid,
(e) fluid-structure interaction problems, where the surface of the fluid domain is given by the structural surface grid.

Given these discrete data, one may either approximate them via analytical functions, or work directly with them. We prefer the second choice, as the proper approximation via analytical functions becomes cumbersome and problematic for complex geometries. The basic steps required to form a new face on a discretely defined surface are as follows:

S1. Select the next side to be deleted from the front; in order to avoid large faces crossing over regions of small faces, the side forming the smallest new face is selected as the next side to be deleted from the front.
S2. Determine the discrete surface face IFADS that contains or is close to the midpoint of the side to be deleted.
S3. Obtain the unit surface normal $\mathbf{n}_{fds}$ for IFADS.
S4. With the information of the desired element size and shape, and $\mathbf{n}_{fds}$, select a 'best point' position for the introduction of a new point IPNEW (see Figure 5).
S5. 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.
S6. Determine whether the face formed with the selected point IPNEW crosses any given sides. If it does, select a new point as IPNEW and try again.
S7. Add the new face, point, and sides to their respective lists.

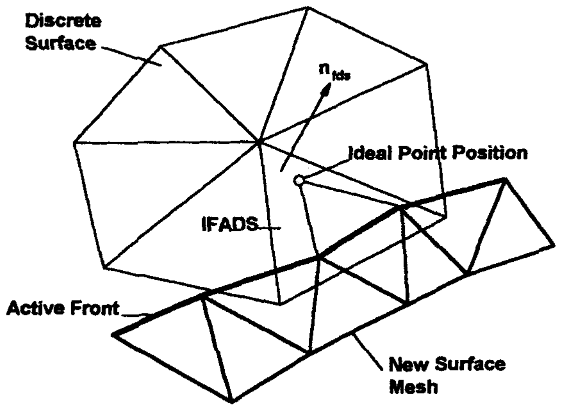


Figure 5. Generation of surface triangulation on discrete surfaces

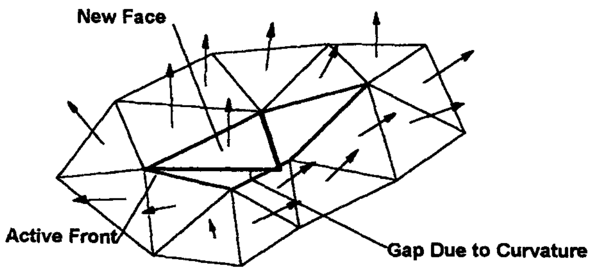


Figure 6. Non-uniqueness of front crossing for curved surfaces

S8. If a new point was added: reposition it on the discretely defined surface.
S9. Find the generation parameters for the new sides from the background grid and the sources.
S10. Delete the known sides from the list of sides.
S11. If there are any sides left in the front, go to S1.

As compared to the surface gridding of analytically defined surfaces, the only differences are:

– the search for the discrete surface face containing or close to a point (Steps S2, S8)
– the introduction of a normal vector $\mathbf{n}_{fds}$ for each face in order to determine the ideal point position (Step S4)
– the repositioning of new points on the discretely defined surface (S8).

In both instances, the face of the discretely defined surface that contains or is closest to a point must be found, i.e. we are faced with a surface-to-surface interpolation problem. By storing this 'host face' for every point, an efficient neighbour-to-neighbour search procedure can be constructed. We have found that the extra cost of this search is negligible.

In regions of colliding fronts on curved 3-D surfaces, the face/side-crossing check is not uniquely defined (see Figure 6). This difficulty is best circumvented by transforming all the points required, i.e. those in the list of close points and those attached to close sides, to the plane defined by the midpoint of the side to be deleted and the normal vector $\mathbf{n}_{fds}$. Thereafter, the face/side-crossing check is performed in 2-D.

In order to avoid gridding over 'edges' or 'corners', it is advisable to first identify these. A simple way to determine edges is by comparing the surface normals of adjacent faces. If the scalar product of them lies below a certain tolerance, a ridge is defined. Corners are defined as points where more than two edges meet. The definition of edges and corners, together with the orientation of the triangulation of the discretely defined surface, leads naturally to the definition of discrete lines and surface patches. The best way to avoid gridding over edges or corners is to first construct sides along the discrete lines, and then grid each discrete surface patch in turn.

4. REDUCTION OF INPUT REQUIREMENTS TO SPECIFY ELEMENT SIZE

The specification of the desired element size and shape in space has been a recurring problem for most grid generators. This is because the requirements for simplicity (low user input) and flexibility (complex geometries) are conflicting. The earliest advancing front grid generators employed a background grid to specify the desired element size and shape in space. This worked well for simple geometries, and 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.[17,18] 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 regions away from surfaces (e.g. a heat-source, an oblique shock in supersonic flow, etc.). Therefore, alternative ways to prescribe element size and shape in space, that combine generality and low user input, are required.

4.1. 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.[7,11] The element size for an arbitrary location $\mathbf{x}$ in space is given as a function of the closest distance to the source $r(\mathbf{x})$. Consider first the line source given by the points $\mathbf{x}_1$, $\mathbf{x}_2$ shown in Figure 7. The vector $\mathbf{x}$ can be decomposed into a portion lying along the line and the normal to it. With the notation of Figure 7, we have

$$\mathbf{x} = \mathbf{x}_1 + \xi \mathbf{g}_1 + \alpha \mathbf{n} \tag{1}$$

The ξ-co-ordinate may be obtained by scalar multiplication with $\mathbf{g}_1$, and is given by

$$\xi = \frac{(\mathbf{x} - \mathbf{x}_1) \cdot \mathbf{g}_1}{\mathbf{g}_1 \cdot \mathbf{g}_1} \tag{1}$$

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

$$\xi' = \max(0, \min(1, \xi)) \tag{3}$$

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

$$\delta(\mathbf{x}) = |\mathbf{x} - \mathbf{x}_1 - \xi' \mathbf{g}_1| \tag{4}$$

Point-sources can be constructed by collapsing both points into one. Consider next the surface source element given by the points $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ shown in Figure 8. The vector $\mathbf{x}$ 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 Figure 8, we have

$$\mathbf{x} = \mathbf{x}_1 + \xi \mathbf{g}_1 + \eta \mathbf{g}_2 + \gamma \mathbf{g}_3 \tag{5}$$

where

$$\mathbf{g}_3 = \frac{\mathbf{g}_1 \times \mathbf{g}_2}{|\mathbf{g}_1 \times \mathbf{g}_2|} \tag{6}$$

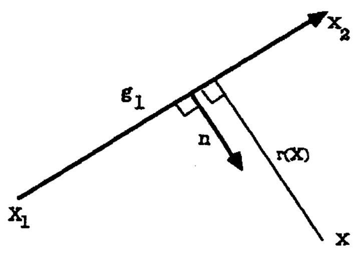


Figure 7. Line source

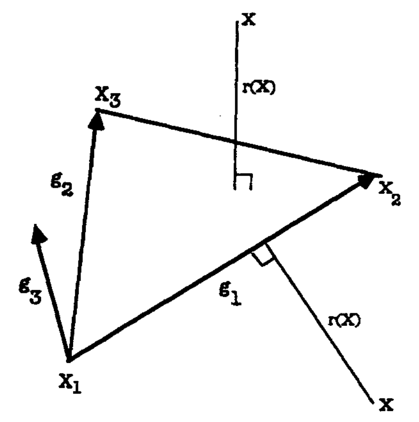


Figure 8. Surface source

By using the contravariant vectors $\mathbf{g}^1$, $\mathbf{g}^2$, where $\mathbf{g}_i \cdot \mathbf{g}^j = \delta_i^j$, we have

$$\xi = (\mathbf{x} - \mathbf{x}_1) \cdot \mathbf{g}^1 \qquad \eta = (\mathbf{x} - \mathbf{x}_1) \cdot \mathbf{g}^2 \qquad \zeta = 1 - \xi - \eta \tag{7}$$

Whether the point **x** lies 'on the surface' can be determined by the conditions:

$$0 \leqslant \xi, \eta, \zeta \leqslant 1 \tag{8}$$

If this condition is violated, the point **x** is closest to any of the given edges, and the distance to the surface is evaluated by checking the equivalent line-sources associated with the edges. If, on the other hand, equation (8) is satisfied, the closest distance between the surface and the point is given by

$$\delta(\mathbf{x}) = |(1 - \xi - \eta)\mathbf{x}_1 + \xi\mathbf{x}_2 + \eta\mathbf{x}_3 - \mathbf{x}| \tag{9}$$

As one can see, the number of operations required to determine $\delta(\mathbf{x})$ is not considerable if one precomputes and stores the geometrical parameters of the sources ($\mathbf{g}_i$, $\mathbf{g}^i$, 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

$$\rho = \max\left(0, \frac{r(\mathbf{x}) - r_0}{r_1}\right) \tag{10}$$

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[7]

$$\delta(\mathbf{x}) = \delta_0[1 + \rho^\gamma] \tag{11}$$

with the four input parameters δ_0, r_0, r_1, γ; typically, $1{\cdot}0 \leqslant \gamma \leqslant 2{\cdot}0$.

(b) *Exponential functions*, which are of the form[19]

$$\delta(\mathbf{x}) = \delta_0 e^{\gamma\rho} \tag{12}$$

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

$$\delta(\mathbf{x}) = \delta_0 \left[1 + \sum_{i=1}^{n} a_i \rho^i \right] \tag{13}$$

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. Indeed, 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:

$$\delta(\mathbf{x}) = \min(\delta_1, \delta_2, \ldots, \delta_m) \tag{14}$$

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 is 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 class of problems.[5] 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 (10–15), it is very difficult to 'localize' the sources in space in order to filter out the relevant ones. On the other hand, 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.

4.2. *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 Figure 9, may require many line-sources. Similarly, the specification of gridding parameters for surfaces with high curvature may require many surface-sources. The net effect is that for complex geometries one 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:

$$\delta = \min(\delta_{bg}, \delta_s, c_i \cdot \min(\delta_A, \delta_B, \delta_C)) \tag{15}$$

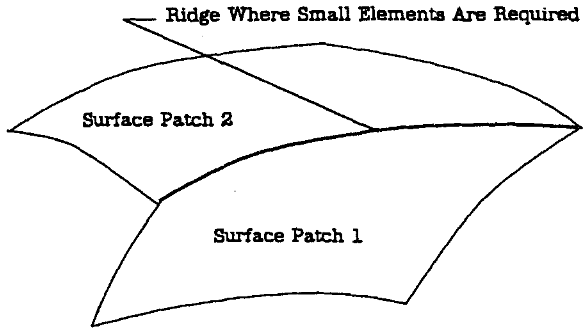


Figure 9. Element size attached to CAD-Data

Typical values for c_i are $1{\cdot}0 \leq c_i \leq 1{\cdot}5$. The first value yields a mesh of uniform element size, whereas the latter gives rise to grids with element 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 Figure 10, 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 is 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.

4.3. 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. In order to reduce the amount of user intervention to a minimum, we propose the use of adaptive background grid refinement. As with any other mesh refinement scheme, we have to define **where** to refine and **how** to refine. Because of its very high speed, we prefer to refine the background grid via classic h-refinement.[20] In this way the possible interactivity on current workstations is maintained.

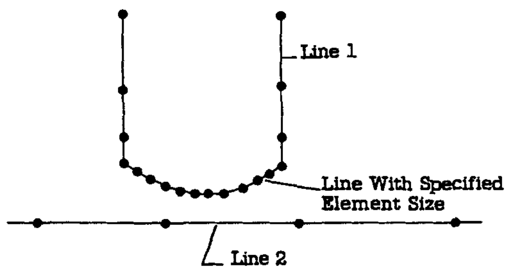


Figure 10. Potential problems with incompatible sizes

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:

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 co-ordinates $\mathbf{x}_1$, $\mathbf{x}_2$, 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:

$$s_n = |\mathbf{x}_2 - \mathbf{x}_1| = \delta_1 \frac{c_i^{n+1} - 1}{c_i - 1} \tag{16}$$

implying

$$\delta_2^* = \delta_1 c_i^n = \frac{s_n(c_i - 1) + \delta_1}{c_i} \tag{17}$$

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.

$$\delta_{bg} = \min(\delta_{bg}, \delta_p^*(\mathbf{x}_{bg} - \mathbf{x}_p)) \tag{18}$$

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

$$\min_{a,b,c,d}(\delta_{bg}) > c_f \delta_p \Rightarrow \textit{refine} \tag{19}$$

where, typically, $1{\cdot}5 \leq c_f \leq 3$. All elements flagged by this last criterion are subdivided further via classic h-refinement,[20] and the background grid variables are interpolated linearly for the newly introduced points.

4.4. Background grid adaptation for accurate surface representation

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 Figure 11, the angle between two faces is given by

$$\frac{l}{2r} = \tan\left(\frac{\alpha}{2}\right) \tag{20}$$

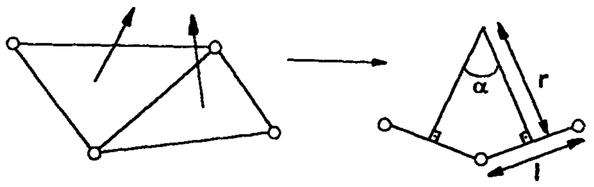


Figure 11. Measuring surface curvature

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

$$l_p = l_g \frac{\tan\left(\frac{\alpha_p}{2}\right)}{\tan\left(\frac{\alpha_g}{2}\right)} \tag{21}$$

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 (21) and reduced appropriately:

$$\delta_i = \min(\delta_i, l_p) \tag{22}$$

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.

5. EXAMPLES

The applicability of the described improvements is shown on four examples. The first two involve thin shells or fabrics. The third case shows how the data input reduction techniques can be used advantageously for complex configurations.

(a) *Dome*: The first case is a dome, characteristic of roofs, parachutes or parasols. The configuration is shown in Figure 12(a). Due to symmetry, only one-quarter of the domain needs to be gridded. The inner and outer surfaces of the dome are separated by only 1/1000th of the radius. A background grid of only five elements (essentially a parallelepiped split into tetrahedra), enclosing the domain, was used to set a limit on the element size. This limit was set to δ=0·8. A point-source was prescribed at the origin, with parameters $\delta_0 = 0{\cdot}15$, $r_0 = 1{\cdot}0$, $r_1 = 0{\cdot}2$, $a_1 = 1{\cdot}0$, $a_2 = 0{\cdot}0$. The surface grid obtained is shown in Figure 12(b). The close-up given in Figure 12(c) reveals crossing faces. The final mesh generated consisted of 4133 elements and 1038 points. A cut through the volume grid at $\Theta = 45°$, which reveals the proximity or crossing of the inner and outer surfaces, is shown in Figure 12(d).

(b) *Sails*: The second example illustrates an application of the proposed technique for sail optimization studies. The configuration is shown in Figure 13(a). Note the inclination of the sails, as well as the very narrow gap between the front and back sails. The sails, which measure approximately 30 m, were modelled as having a thickness of 1 mm. For all practical purposes, this is equivalent to a vanishing thickness. As before, a background grid of only five elements enclosing the domain was used to set a limit on the element

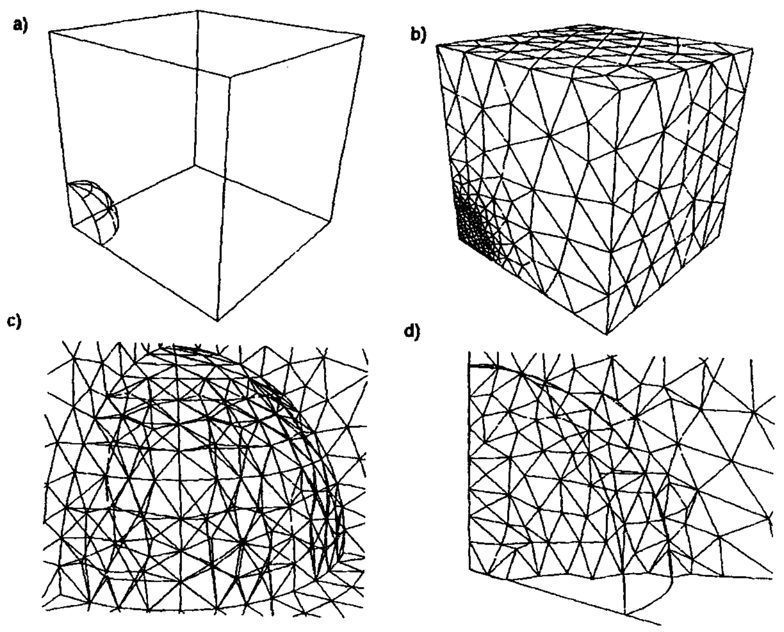


Figure 12. Dome: (a) CAD definition of the domain to be gridded; (b) surface grid; (c) surface grid (close-up); (d) cut through the generated mesh

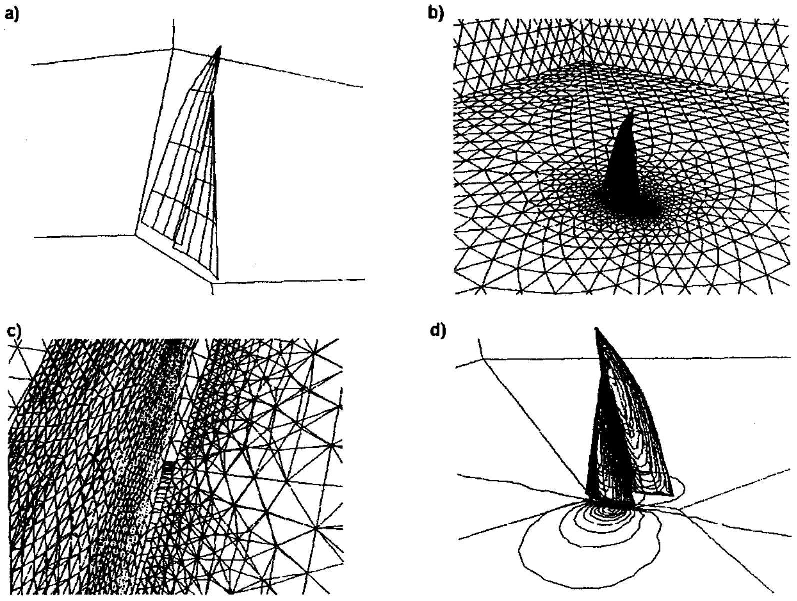
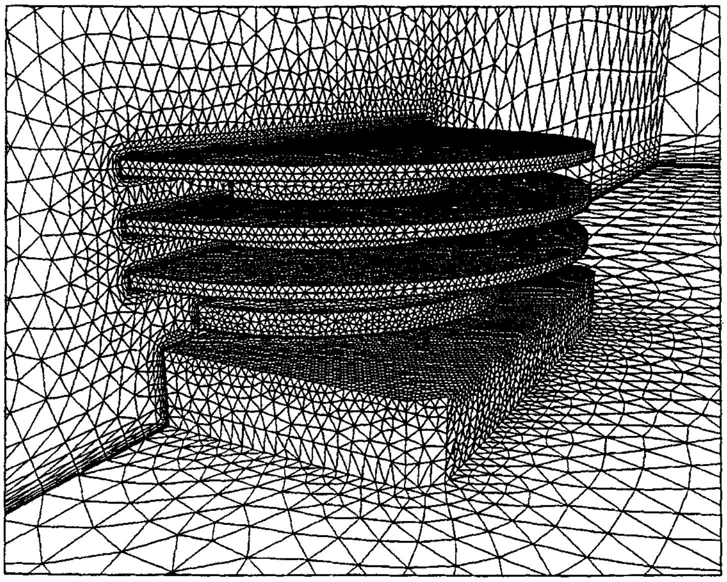


Figure 13. Sails: (a) CAD definition of the domain to be gridded; (b) surface grid; (c) surface grid; (d) pressure distribution on the surface

size. A linear variation of element size in the vertical direction was prescribed, with $\delta_0 = 10$ m and $\delta_1 = 30$ m. In addition, two line-sources and six surface-sources were employed to concentrate smaller elements on the sails and between them. Furthermore, element size attached to CAD-data was used for the sails. Two views of the surface mesh are shown in Figures 13(b,c). As before, the surfaces of the sails cross. The final mesh


a) Navier-Stokes Mesh

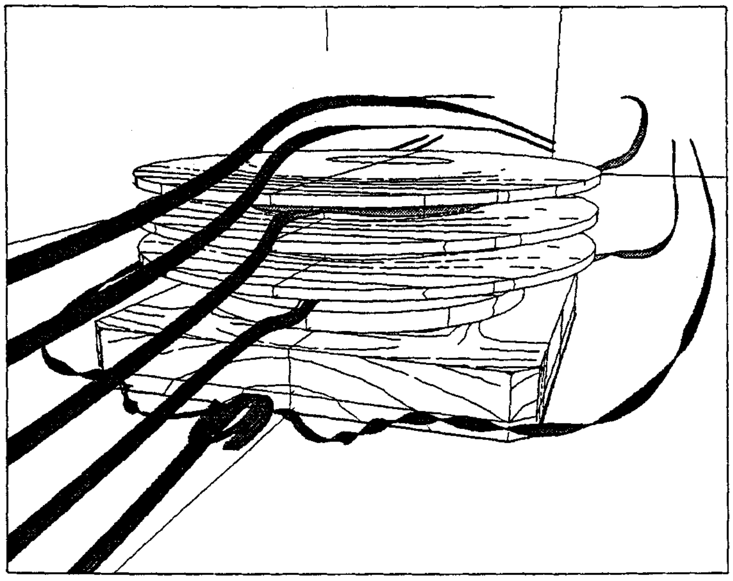

b) Chip Iso-Temperature Contours & Streamlines

Figure 14. Chip: (a) Surface grid; (b) solution

consisted of 200,277 elements and 20,383 points. The contours of the surface pressures obtained on this mesh for an inviscid flow simulation are shown in Figure 13(d).

(c) *Chip*: In this particular instance, the proper discretization of the solid domain inside a chip, as well as the surrounding fluid medium, was required. The definition of the surfaces is shown in Figure 14. The complete domain had five different materials, including the fluid. In this instance, no sources were required. Instead, the option to attach element size directly to CAD-data was used extensively. Most of the chip surfaces had an element size attached to them. For the fluid region, a Navier–Stokes grid was required. This special option has been described elsewhere,[8] and is therefore omitted here. The surface of the grid, which consisted of approximately 1·5 million elements, is shown in Figure 14(a). Results from an incompressible flow solution with conjugate heat transfer at a Reynolds number of Re = 1000 based on the diameter of the chip are included in Figure 14(b) to show that these grids can indeed be used for industrial applications. Due to the ability to attach grid size directly to CAD-data, the preprocessing time was kept to a minimum. Less than 4 h (one morning) were required to input all the surface defining information, as well as the grid generation parameters.

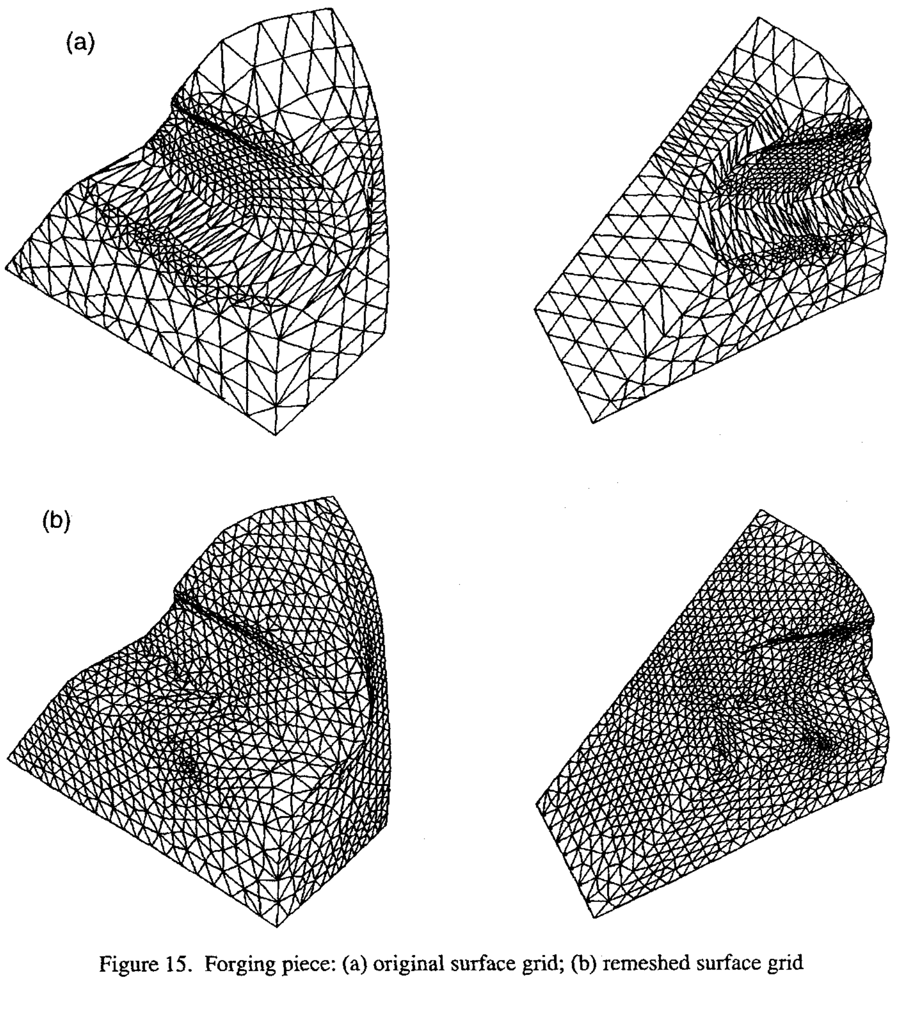


Figure 15. Forging piece: (a) original surface grid; (b) remeshed surface grid

(d) *Forging piece*: This case demonstrates the use of surface remeshing for discretely defined domains. The surface of the original, highly distorted mesh, is shown in Figure 15(a). The edge-detection algorithm then forms the discrete line and surface patch definition. An adaptive background grid is generated automatically, starting from a cube. With this background mesh, a new surface triangulation, shown in Figure 15(b), is generated. The whole process takes less than 5 min on the workstation, indicating that it is feasible to port these automatic remeshing techniques into production codes.

(e) *Cylinder on wall*: This example demonstrates the adaptation of the background grid to surface features. The maximum allowable angle change between neighbouring surface triangles on the cylinder surface was set to $\alpha = 10°$. In order to generate an initial mesh, a line-source along the cylinder was specified, together with an initial coarse background grid consisting of a parallelepiped split into five tetrahedra. The surface triangulation obtained in this way is shown in Figure 16(a). After five adaptation passes, the corresponding background grid and surface grid are shown in Figures 16(b, c). The final grid for this configuration consisted of 2600 boundary points, 15,176 points and 81,501 elements.

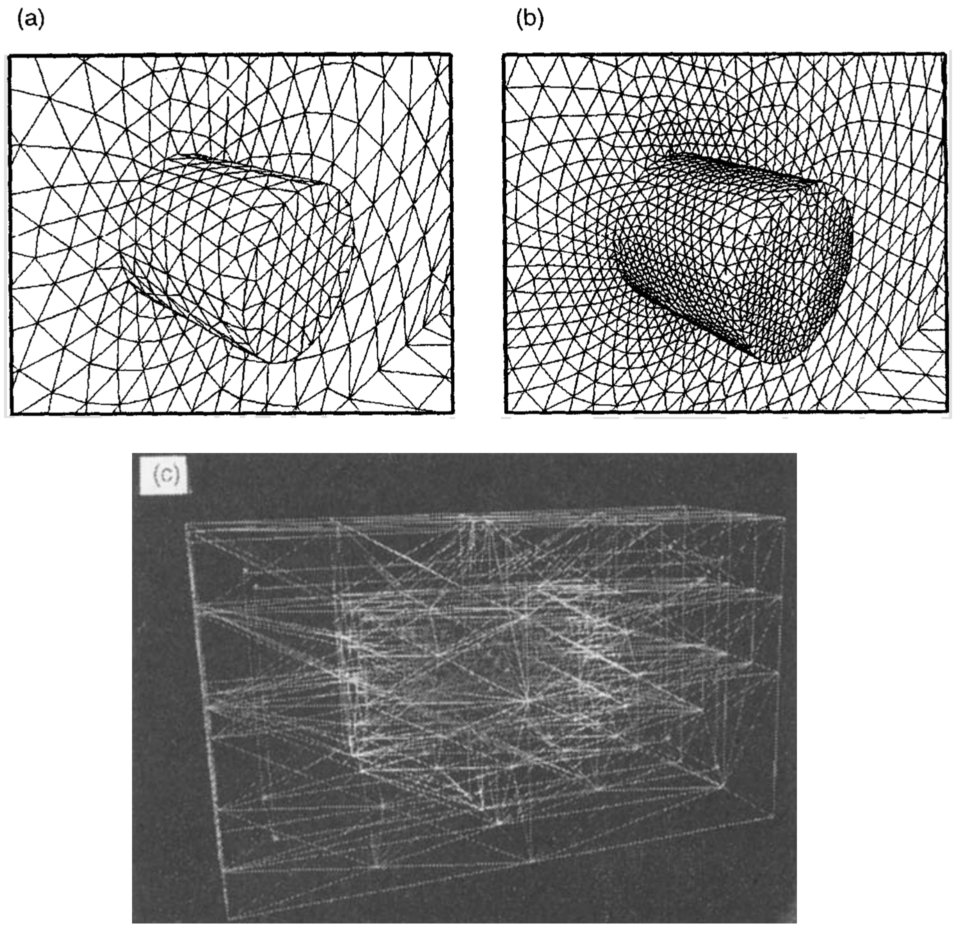
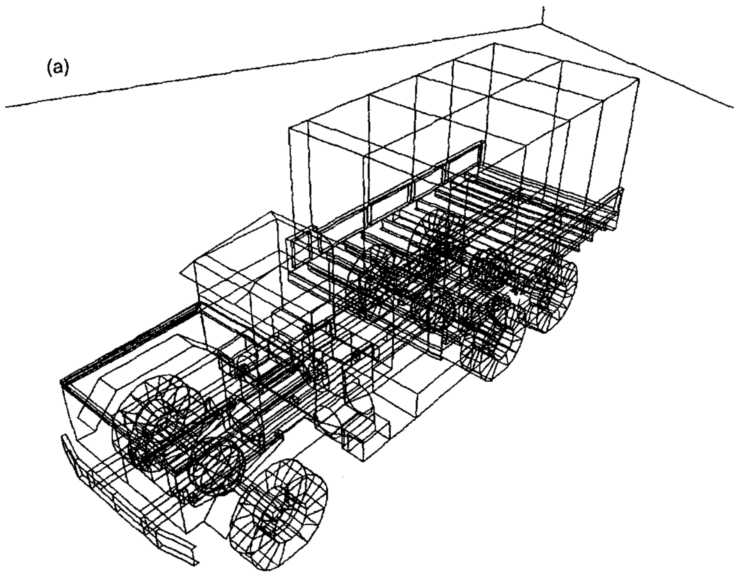


Figure 16. Cylinder on wall: (a) original surface mesh; (b) adaptively remeshed surface mesh (automatic surface refinement (angle criterion 10°)); (c) refined background grid

(f) *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 Figures 17(a, b), consisted of 6306 points, 3718 lines and 1604 surface segments. An initial coarse background, consisting of a parallelepiped split into five tetrahedra, as well as a pair of surface sources, were provided initially to specify element size in space. The CAD-data were analyzed, and the point-sizes specified were set to be commensurate with the

(a)



(b)

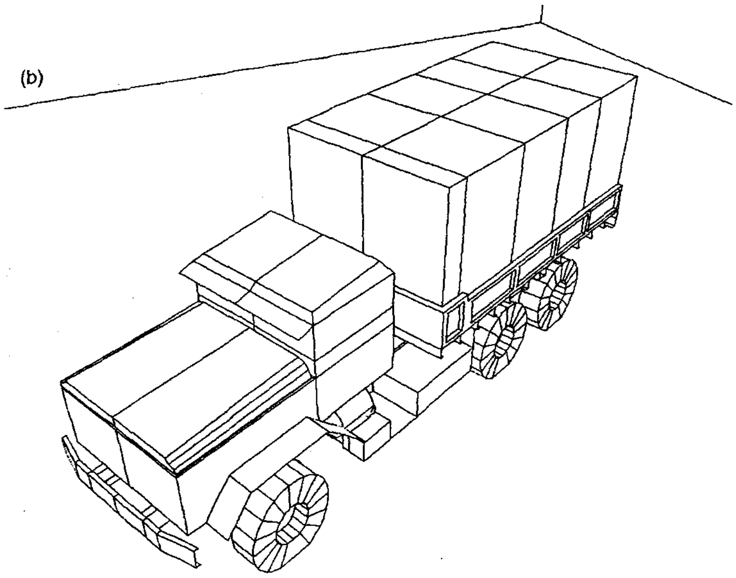
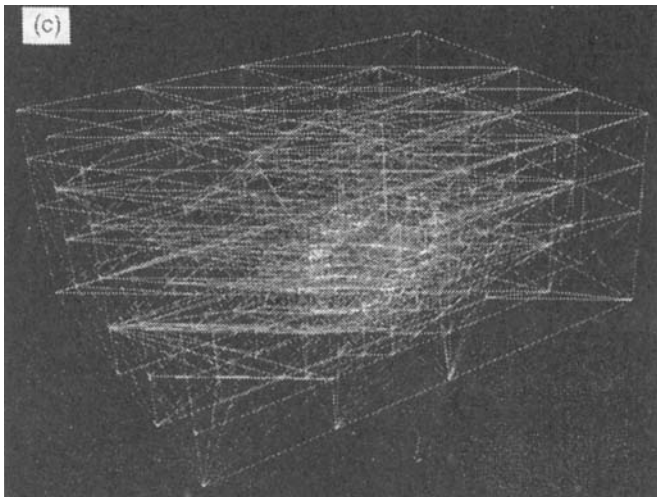


Figure 17. Truck: (a) CAD data; (b) CAD data; (c) adapted background grid; (d) surface triangulation



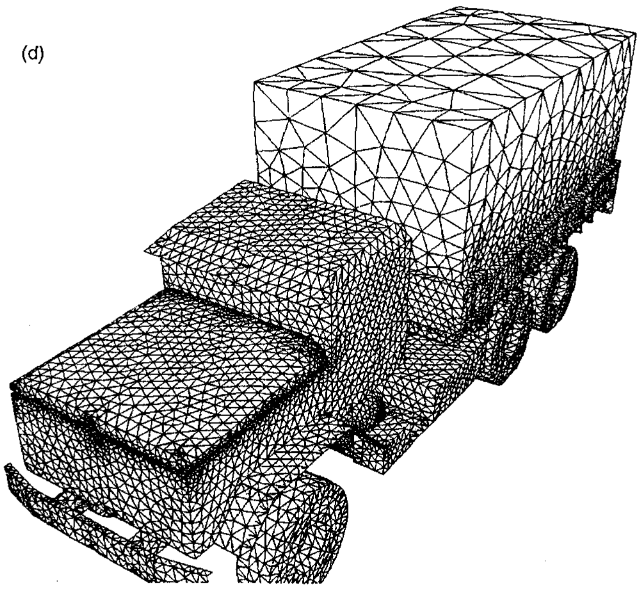


Figure 17. (*continued*)

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, and is shown in Figure 17(c). The surface of the final mesh, which consisted of 38,639 boundary points, 192,473 points and 1,026,349 tetrahedra, is shown in Figures 17(d).

6. CONCLUSIONS

Extensions and improvements to the advancing front grid generation technique have been described.

The first general topic included the generation of volumetric grids for situations with thin surfaces, such as shells, membranes, fabrics or CAD-data that exhibit cusps. The key idea here was the introduction of a crossing environment variable that is attached to faces and points. These environment variables are used to filter out undesired or incorrect information during the grid generation process.

The second topic described the retriangulation of discretely defined surfaces, such as those encountered for geophysical and climate modelling, as well as medical, metal-forming and fluid/structure interaction problems.

The third topic dealt with the reduction of manual labour to specify the desired element size and shape in space. Sources, CAD-attached element size, and adaptive background grids were discussed. Each of these techniques has its merits, and their combined use has proven very effective over the years. They are also very well suited for problems with moving bodies that require remeshing. For these cases, background grids prove ineffective. The co-ordinates of sources, on the other hand, may follow the movement of the bodies they belong to. The same applies, in an obvious way, to CAD-attached element size and shape. Several examples have been included to demonstrate the usefulness of these extensions.

As with every other technique, further improvements and extensions are possible. An important area that deserves further study is the inclusion of arbitrary directional stretching for sources and CAD-attached data, in particular for situations where different directions occur.

ACKNOWLEDGEMENTS

This work was partially supported by AFOSR, with Dr. Leonidas Sakell as the technical monitor, as well as DNA, with Dr. Mike Giltrud as the technical monitor. The help and encouragement of Eric Mestreau, ESI, for the chip cooling case is also gratefully acknowledged.

REFERENCES

1. J. Bonet and J. Peraire, 'An alternate digital tree algorithm for geometric searching and intersection problems', *Int. j. numer. methods eng.*, **31**, 1–17 (1991).
2. S. H. Lo, 'A new mesh generation scheme for arbitrary planar domains', *Int. J. Numer. Methods Eng.*, **21**, 1403–1426 (1985).
3. R. Löhner, 'Some useful data structures for the generation of unstructured grids', *Commun. Appl. Numer. Methods*, **4**, 123–135 (1988).
4. R. Löhner and P. Parikh, 'Three-dimensional grid generation by the advancing front method', *Int. j. numer. methods fluids*, **8**, 1135–1149 (1988).
5. R. Löhner, 'Three-dimensional fluid-structure interaction using a finite element solver and adaptive remeshing', *Comput. Syst. Eng.*, **1**(2–4), 257–272 (1990).
6. R. Löhner, J. Camberos and M. Merriam, 'Parallel unstructured grid generation', *Comput. Methods Appl. Mech. Eng.*, **95**, 343–357 (1992).
7. R. Löhner, 'Finite elements in CFD: Grid generation, adaptivity and parallelization', in *AGARD Rep. –7D–787, Proc. Special Course on Unstructured Grid Methods for Advection Dominated Flows*, VKI, Belgium, May and NASA Ames, Moffet Field, CA, September 1992.
8. R. Löhner, 'Matching semi-structured and unstructured grids for Navier–Stokes calculations', *AIAA-93-3348-CP*, 1993.

9. J. Peraire, M. Vahdati, K. Morgan and O. C. Zienkiewicz, 'Adaptive remeshing for compressible flow computations', *J. Comput. Phys.*, **72**, 449–466 (1987).
10. J. Peraire, K. Morgan and J. Peiro, 'Unstructured finite element mesh generation and adaptive procedures for CFD', *AGARD-CP-464*, 18, 1990.
11. S. Pirzadeh, 'Viscous unstructured three-dimensional grids by the advancing-layers method', *AIAA-94-0417*, 1994.
12. A. Shostko and R. Löhner, 'Three-dimensional parallel unstructured grid generation', *Int. J. Numer. Methods Eng.*, **38**, 905–925 (1995).
13. J. D. Baum and R. Löhner, 'Numerical simulation of shock interaction with a modern main battlefield tank', *AIAA-91-1666*, 1991.
14. J. D. Baum, H. Luo and R. Löhner, 'Numerical simulation of a blast inside a Boeing 747', *AIAA-93-3091*, 1993.
15. H. Luo, J. D. Baum and R. Löhner, 'Edge-based finite element scheme for the Euler equations', *AIAA J.*, **32**(6), 1183–1190 (1994).
16. E. Mestreau and R. Löhner, 'Numerical simulation of chip cooling via large-scale FEM simulations', CSI-GMU Preprint, 1994.
17. M. S. Shepard and M. K. Georges, 'Automatic three-dimensional mesh generation by the finite octree technique', *Int. J. Numer. Methods Eng.*, **32**, 709–749 (1991).
18. M. A. Yerry and M. S. Shepard, 'Automatic three-dimensional mesh generation by the modified-octree technique', *Int. j. numer. methods eng.*, **20**, 1965–1990 (1984).
19. N. P. Weatherill, 'Delaunay triangulation in computational fluid dynamics', *Comput. Math. Appl.*, **24**(5/6), 129–150 (1992).
20. R. Löhner and J. D. Baum, 'Adaptive H-refinement on 3-D unstructured grids for transient problems', *Int. J. Numer. Methods Fluids*, **14**, 1407–1419 (1992).