Acta Numerica (1996), *pp.* 121–148

Automatic grid generation

William D. Henshaw
Scientific Computing Group
Computing, Information and Communications Division
Los Alamos National Laboratory
Los Alamos, NM 87545, USA
E-mail: henshaw@lanl.gov

Current methods for the automatic generation of grids are reviewed. The approaches to grid generation that are discussed include Cartesian, multi-block-structured, overlapping and unstructured. Emphasis is placed on those methods that can create high-quality grids appropriate for the solution of equations of a hyperbolic nature, such as those that arise in fluid dynamics. Numerous figures illustrate the different grid generation techniques.

CONTENTS

1. Introduction

The intent of this paper is to give a brief review of current methods for the automatic generation of grids for the solution of problems from computational fluid dynamics (CFD), computational electromagnetics and other fields where the solutions are hyperbolic in nature. These applications require the generation of high quality grids with a large number of grid points. It is often the case that the geometry may change with time or it may be necessary to refine the mesh adaptively. It is thus essential that the grid generation algorithms be fast, since the grid may have to be regenerated at every step of a time-dependent simulation. Various popular methods for structured and unstructured grid generation will be described. Figures will illustrate the

current state of the technology. Grid generation capabilities have improved greatly in recent years. However, it is perhaps not an exaggeration to say that the construction of a grid is currently the most difficult and time-consuming aspect of determining an accurate solution to a problem on a complicated domain. Indeed, starting from scratch, with some description of the geometry, the time to generate a grid is measured in *weeks* rather than hours.

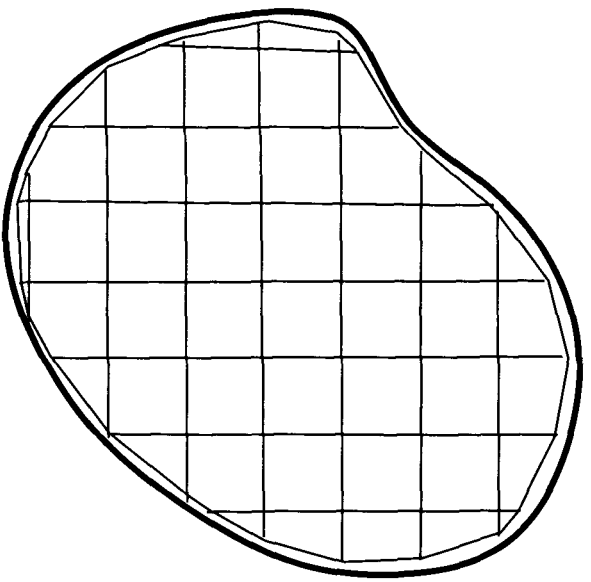

Fig. 1. A Cartesian grid.

Early computational grids were often *Cartesian grids*, or cut-out grids (Fig. 1), whereby the region was covered by a single rectangular grid and the portions of the grid lying outside the region were *cut out*, leaving some irregular cells. This approach was replaced by boundary-conforming grids, whereby a rectangular grid was mapped onto the region, with boundaries corresponding to a coordinate line. Such curvilinear grids that are transformations of a rectangular grid will be called *logically rectangular grids.* Grids that conformed to boundaries improved solution accuracy and made it easier to apply boundary conditions. As computers became faster and more complicated problems were attempted, it became apparent that this single-block approach was not flexible enough to handle complicated geometries. This led to the introduction of the multi-block approach, where the domain was partitioned into blocks and within each block a logically rectangular grid was constructed (Fig. 2). In time, however, it became apparent that this approach was still not sufficiently flexible, and was difficult to automate. Therefore some other approach was needed. One way to add additional flexibility, while still

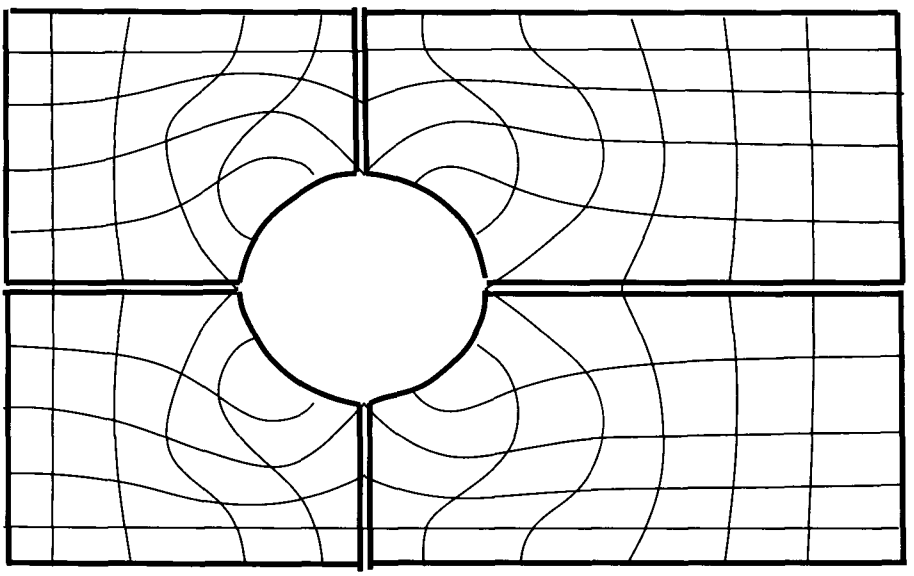

Fig. 2. A multi-block-structured grid divides the region into logically rectangular blocks.

retaining the logically rectangular structure, was the use of overlapping grids in which the component grids are allowed to overlap. There has also been renewed interest in the Cartesian grid approach, using adaptive mesh refinement to improve boundary resolution. Recently, the main interest and focus of research has been in unstructured meshes, which allow complete freedom in grid point placement, although at the expense of speed and memory usage. With little doubt, unstructured meshes offer the best hope for a completely automatic mesh generation program. Completely unstructured grids are not without their difficulties for CFD, and perhaps a hybrid method, combining the unstructured approach with locally structured grids (to resolve boundary layers, for example), will turn out to be the most effective.

The purpose of grid generation is to create a discrete representation for a domain. This entails distributing points throughout the domain. There are two main classes of grids, structured and unstructured. In a structured grid the points covering the domain result from the transformation of a logically rectangular square (or cube in three dimensions*). The grid points can be stored as an array $\mathbf{x}(i_1, i_2)$ and the neighbours of a given grid point are simply found as the neighbours in index space, $\mathbf{x}(i_1 \pm 1, i_2 \pm 1)$. In an unstructured grid, on the other hand, the points are connected to one another

* Throughout this paper, the terminology will be for two-dimensional grids (quadrilaterals, triangles), but the remarks will usually apply equally well to three-dimensional grids (hexahedra, tetrahedra).

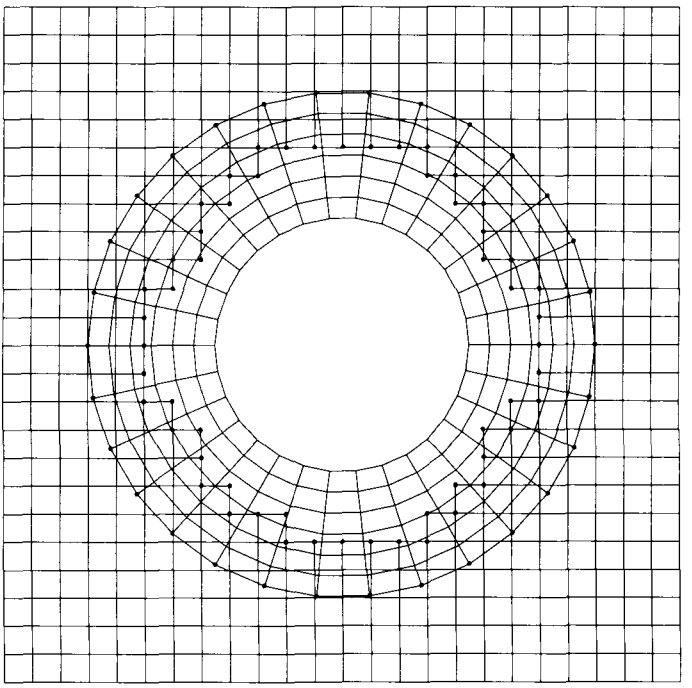

Fig. 3. An overlapping grid consists of logically rectangular blocks that overlap; some blocks have cut-out regions.

in a general manner; the connectivity information must be explicitly saved. The grid points might be saved as a list $\mathbf{x}_i$, and there would be other lists giving information about neighbours. Of course, the partition of grid types into structured and unstructured is not entirely appropriate, since some grids consist of a *set* of structured grids and other *hybrid* grids have both unstructured and structured parts.

Grids are used to solve equations, typically partial differential equations (PDEs) and integral equations. The computer programs that solve these equations, hereafter referred to as *solvers*, typically discretize a continuous equation with finite-difference, finite-element, finite-volume, spectral-element or boundary-integral methods. At the grid generation level, it is usually not important which particular solver will use the grid: rather the *style* of the grid is most relevant. That is, it is important whether the grid is structured or unstructured, whether the grid elements are triangles or quadrilaterals, or whether the grid is multi-block-structured or overlapping (see Fig. 3). The unstructured triangular grid (see Fig. 4) can be used either with a finite-element solver or a finite-volume solver, just as a multi-block-structured grid can be used by a finite-difference solver or by finite-element solver for quad-

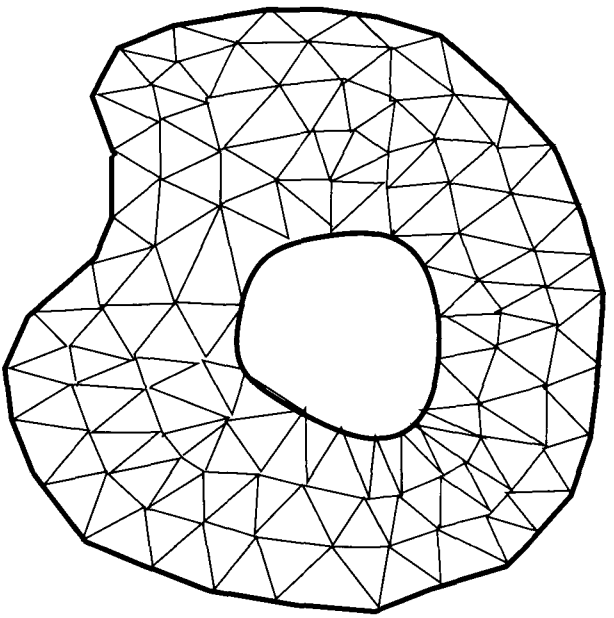

Fig. 4. An unstructured triangular grid is very flexible for representing geometry.

rilaterals. Despite these remarks, it is not uncommon to see references to a *finite-element mesh* (which usually just means an unstructured grid).

The errors in solving a PDE on a grid depend strongly on the quality of the grid. The *quality* of a grid is a relative concept and depends on the actual equations that will be solved, as well as the numerical method that will be used. In principle, a given problem could be repeatedly solved with different grids (with the same number of grid points) and the error in the numerical solution could be measured as a function of the grid. The smaller the error, the better the quality of the grid. Some adaptive methods do indeed redistribute points to try and minimize the error. When creating a grid initially, however, the grid is usually generated with some general principles in mind, such as keeping the cell size varying smoothly, and resolving boundary layers, if appropriate. Generally speaking, the solution of equations with wave-like behaviour (hyperbolic) require *smoother* grids than the solution of elliptic equations. The smoothness of a grid is hard to define in general but relates to the local variation of the cells. An elliptic problem can be accurately solved on a relatively poor-quality grid since the effects of any non-smoothness in the grid will be smoothed out by the elliptic operator. In contrast, hyperbolic operators provide no smoothing effects. To understand this further, note that the properties of the grid, such as the variations in the grid point positions, appear, implicitly or explicitly, in the discrete equations

used in the solver. Consider the solution of the one-dimensional wave equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0.$$

If the grid points are allowed to vary according to the parameterization $x = X(r)$ (that is the grid points are equally spaced in r), then the equation for $v(r,t) = u(X(r),t)$ becomes

$$\frac{\partial v}{\partial t} + \frac{1}{X_r}\frac{\partial v}{\partial r} = 0.$$

It is now clear that, if the parameterization is not smooth, then X_r will not be smooth, and this will be reflected in the discrete solution. A grid that is not smooth can distort waves and cause spurious reflections, rather similar to the effect of a wave passing through a non-uniform medium. Higher-order accurate methods are also popular for the solution of wave-like problems, for both efficiency and accuracy reasons. Higher-order methods will in general require higher-quality grids than lower-order methods.

It is important to realize that solvers written for one type of grid will typically not work on other types of grids. Although a structured grid can always be turned into an unstructured grid and used with an unstructured solver, the unstructured solver would not usually take advantage of the structured nature of the grid. There is, however, increasing interest in *hybrid* grids and hybrid grid solvers. Hybrid grids range from those that are primarily unstructured triangles, with some structured quadrilaterals to resolve a boundary layer, to those that are primarily structured blocks, with triangles used to merge the blocks.

The number of grid points required for many three-dimensional problems is extremely large. For typical big simulations there are on the order of 10^6 grid points, this number being limited only by computer memory and speed. Viscous fluid flow computations over an entire aircraft could easily use orders of magnitude more grid points. Points are required not only to represent complicated geometries (as illustrated by some of the figures in this paper) but also to resolve rapidly varying features of the solution (shocks, boundary layers, vortex shedding).

The following grid generation methods will be discussed in more detail in the rest of this paper:

- Cartesian
- multi-block-structured
- overlapping
- unstructured.

The area of adaptive mesh refinement, a large and active field in itself, will only be briefly mentioned here. Further information can be found in many of

the references. There are a number of issues that must be considered when evaluating the appropriateness of a given type of grid or grid generator:

- the speed of generating a grid
- the robustness of grid generation
- the quality of the generated grid
- the ability to construct grids from standard computer-aided-design specifications
- the level to which the grid generation is automatic – how much user intervention is required and how many tuning parameters are there?
- the support for adaptivity and moving geometries – can grids be regenerated quickly?
- the speed of the solver on the resulting grid
- the effectiveness of the approach on both parallel and serial architectures.

Generally, unstructured grid generators tend to be more robust and automatic, while structured grid generators create higher quality grids for which faster and more efficient solvers can be written. Further remarks on these issues will be made when the different approaches are discussed.

The field of grid generation is expanding rapidly. Many excellent references have been unavoidably omitted from this review and apologies are due to the authors. For further information, the reader is referred to the books by Thompson, Warsi and Mastin (1985), George (1991), Knupp and Steinberg (1993), and Castillo (1991); the conference proceedings edited by Weatherhill et al. (1994), Arcilla, Häuser, Eiseman and Thompson (1991), and Babuška, Flaherty, Henshaw, Hopcroft, Oliger and Tezduyar (1995); and the review papers by Löhner (1987), and Eiseman (1985). Some other excellent sources of information are Robert Schneiders' *Finite Element Mesh Generation* site on the World Wide Web:

`http://www-users.informatik.rwth-aachen.de/~roberts/meshgeneration.html`

and Steven Owen's *Meshing Research Corner* site:

`http://www.ce.cmu.edu:8000/user/sowen/www/mesh.html`

These sites include information about both unstructured and structured mesh generation, and pointers to a variety of public-domain and commercial grid generation packages.

2. Basic steps in grid generation

There are some basic steps in constructing a grid that are common to many of the grid generation approaches.

- As a first step in the grid generation process, the geometry of the region to be discretized must be defined, that is, the surfaces that make

up the boundary of the region must be described. The geometry can be represented in many ways, such as with analytic shapes (spheres, cylinders), splines, NURBS (non-uniform rational b-splines), and interpolation methods. The geometry may be constructed within a computer-aided-design (CAD) system or within the grid generation system itself. Many CAD systems emphasize solid modelling using analytic shapes and do not cater particularly well to the creation of grids for flow problems. As a result, many grid-generation packages provide some level of CAD support.

- Given the representation of the surface (as a NURB, for example), it is often necessary to reparameterize the surface. This step is referred to as *constructing a surface grid.* (The Cartesian grid approach would not require this step.) Given a smooth surface, the most widely used CAD representations of this surface are only guaranteed to be *geometrically* smooth – they are often not parametrically smooth. Thus, if grid lines are drawn on the surface, equally spaced in parameter space, the lines will not vary smoothly. Typically the parametric derivatives of the surface will not even be continuous. By relaxing the requirements of parametric smoothness, it is easier for the CAD system to represent the surface, but unfortunately such a representation causes major difficulties for the grid generation system. Furthermore, CAD programs often represent complicated surfaces by multiple patches and these patches may not join properly (there may be gaps between patches, or the patches may overlap). Grid generators must carefully examine the surfaces and fix such defects. This is a difficult task and one that in principle should not be necessary.

  Grid generators would like to have parametrically smooth surfaces so that the grid points vary smoothly over the surface. The smoothing of the surface parameterization typically involves solving an elliptic-like equation on the surface or, in the case of triangles, shifting vertices according to some averaging procedure. This step will also involve clustering of grid points, such as in regions of high curvature. Surface grid generation techniques are usually quite similar to volume grid generation methods.
- The third step is the generation of a volume grid. The procedure followed at this stage differs significantly between the various grid types, and will be described in the following sections.

3. Cartesian grid generation

Lately, there has been renewed interest in the Cartesian grid approach, due to its simplicity and ease of automatic grid generation. By combining the Cartesian approach with adaptive mesh refinement, several of the drawbacks

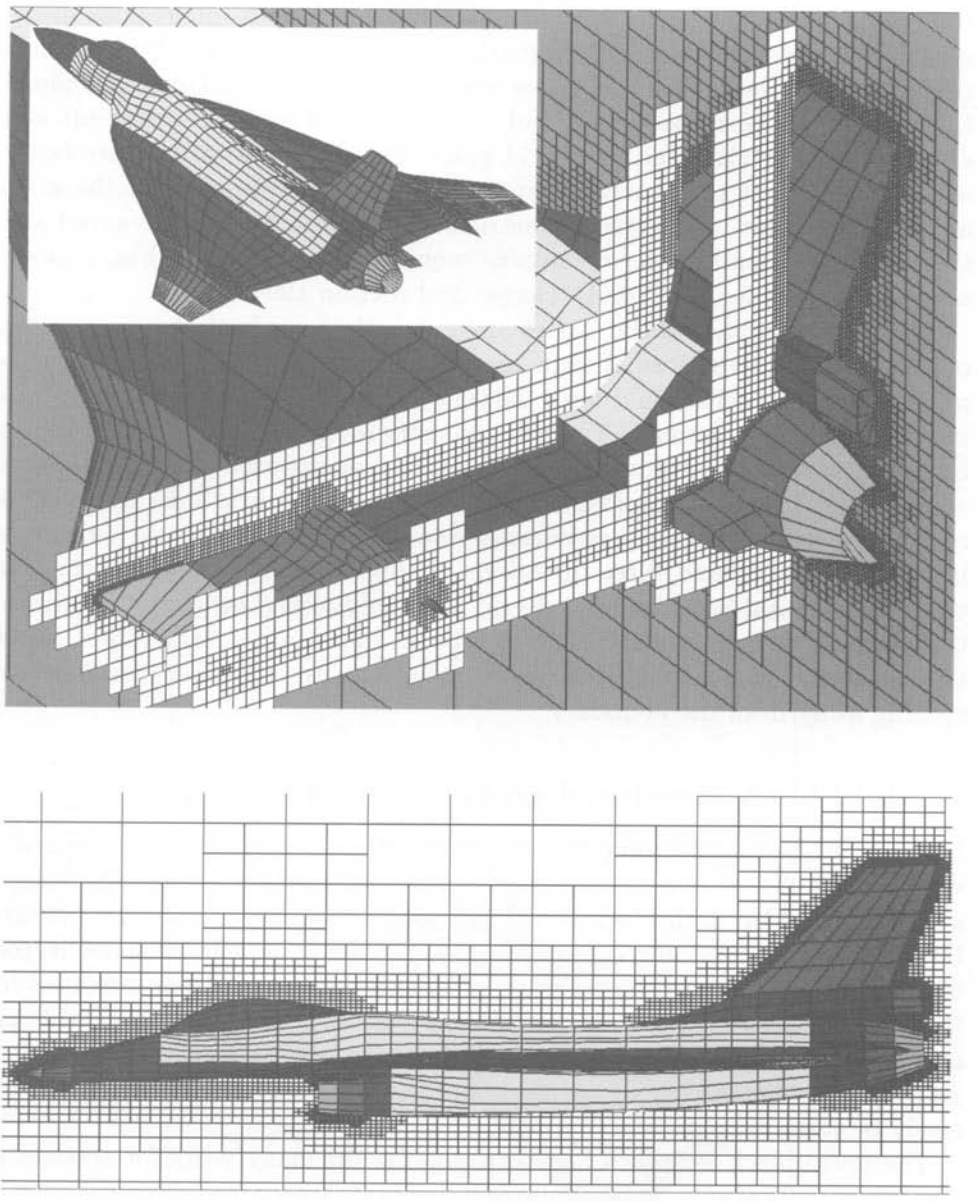

Fig. 5. Cartesian grid for the F16XL.

of the technique have been eased; see, for example, Berger and Melton (1994), and Coirier and Powell (1995). In the Cartesian grid approach, the region is covered by a rectangular grid. Domain boundaries cut out regions of the grid. The boundaries are not covered by boundary-fitted grids, but adaptive refinement can be used to improve surface resolution. Adaptively refined Cartesian grids combine elements of structured and unstructured grids and are perhaps best classified as hybrid grids. Cartesian grid solvers are faster and more efficient than more general unstructured solvers. Since the grids are all rectangular, much less geometrical information needs to be saved and there are significantly fewer operations required per grid point. Fig. 5 shows a Cartesian grid for an F16XL (Berger and Melton 1994).

The main drawback of the Cartesian-grid method lies in the representation of the boundary, where small cells are often formed. Without special treatment these small cells would force the time-step of a time-dependent solver to become prohibitively small. Typical applications only solve problems without boundary layers (Euler equations, for example, as opposed to Navier–Stokes equations). Since the boundary is not aligned with a grid line, in order to resolve a boundary layer it is necessary to refine the grid in two directions in two dimensions and three directions in three dimensions. In contrast, a three-dimensional boundary-fitted grid need only refine the grid in the direction normal to the boundary. This can be an important consideration, since the boundary layer grid spacing can be more than 10^3 times smaller than the spacing away from the boundary.

4. Multi-block-structured grid generation

In the multi-block-structured grid approach, the computational volume is divided into a set of non-overlapping logically rectangular *blocks*. A volume grid is created on each block; see Thompson (1988) and Spekreijse (1995). Usually, global smoothing is performed on the blocks to achieve some degree of continuity in the grid metrics at the block boundaries. Discontinuities in the grid spacing at block boundaries can result in poor solutions. Grid lines may or may not join across blocks; if not, the grid is sometimes called a *patched grid*. Patched grids require more general interpolation, but this can easily be made conservative.

The multi-block approach has been popular for many years for aerospace and other applications. It has improved flexibility over a single logically rectangular patch. High-quality grids can be created, and solvers are fast and efficient. Efficient numerical methods such as implicit methods and multigrid methods work well. Good-quality, highly stretched boundary-layer grids can be created. The main disadvantage with the method is that it is difficult to automate the decomposition of a region into non-overlapping blocks, especially in three dimensions. There is some difficulty with moving geometries

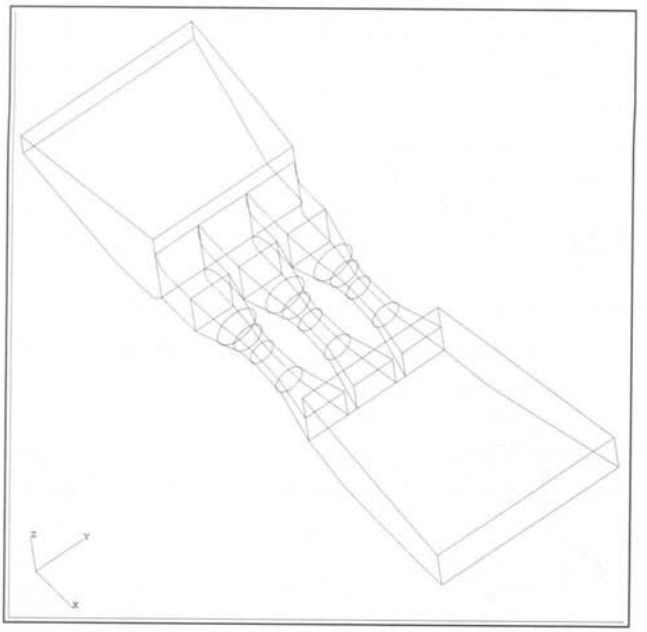

Fig. 6. Block decomposition of a hydroelectric power station.

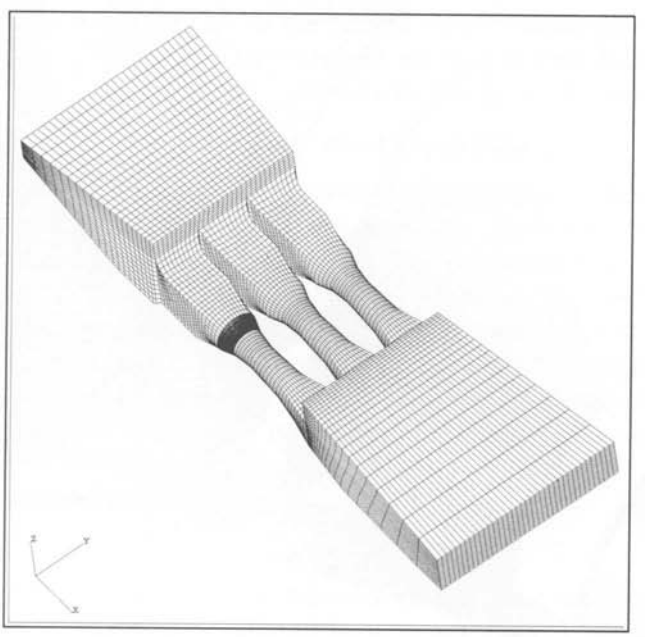

Fig. 7. Corresponding grid.

since the block decomposition may have to change. Generating a multi-block grid for a complicated three-dimensional region usually requires significant human intervention.

Figs 6 and 7 show a block structure grid for a hydroelectric power station (Spekreijse, Boerstoel, Vitagliano and Kuyvenhoven 1992). Fig. 8, showing a multi-block-structured grid for the space shuttle, is reproduced courtesy of Steven Alter, Lockheed Engineering and Sciences Company.

4.1. Structured component grid generation approaches

One of the most important parts of structured grid generation (whether multi-block or overlapping) is the creation of the individual blocks. The blocks will generally have some or all bounding surfaces specified, and the aim is to create a smooth volume-filling grid with appropriate grid spacing and orthogonality. The most common techniques fall into the following categories:

- algebraic
- elliptic and variational
- hyperbolic.

Algebraic grid generation methods create grids for the interior of a domain by algebraically combining the representations of the boundary surfaces. The transfinite interpolation procedure uses polynomials to interpolate the interior grid from the boundaries; see Thompson et al. (1985). For example, a two-dimensional grid bounded by the two curves $\mathbf{C}_1(s)$ and $\mathbf{C}_2(s)$, can be created using the simple shearing transformation

$$\mathbf{G}(r,s) = r\mathbf{C}_1(s) + (1-r)\mathbf{C}_2(s).$$

Whether the grid is useful depends strongly on the shape and parameterization of the curves. Algebraic methods are not as flexible as some of the other methods but their simplicity and speed of generation makes them popular.

Elliptic generation methods, pioneered by Thompson and co-workers, can handle more general cases. They can be used to construct high-quality grids on rather complicated domains; see, for example, Thompson (1987), Sorenson (1986), and Spekreijse (1995). A Poisson equation is solved to determine the location of the grid points. This equation commonly takes the form (in two dimensions):

$$\frac{\partial^2 r_i}{\partial x_1^2} + \frac{\partial^2 r_i}{\partial x_2^2} = P_i, \quad i = 1, 2,$$

where $\{r_i\}$ are the unit square coordinates, $\{x_i\}$ are the physical domain coordinates and $\{P_i\}$ are the *control functions*. In practice, these equations are transformed so that $\{r_i\}$ are the independent variables,

$$\sum_{\mu,\nu} g_{\mu,\nu} \frac{\partial^2 x_i}{\partial r_\mu r_\nu} + \sum_{\mu} \frac{\partial x_i}{\partial r_\mu} P_\mu = 0, \quad i = 1, 2.$$

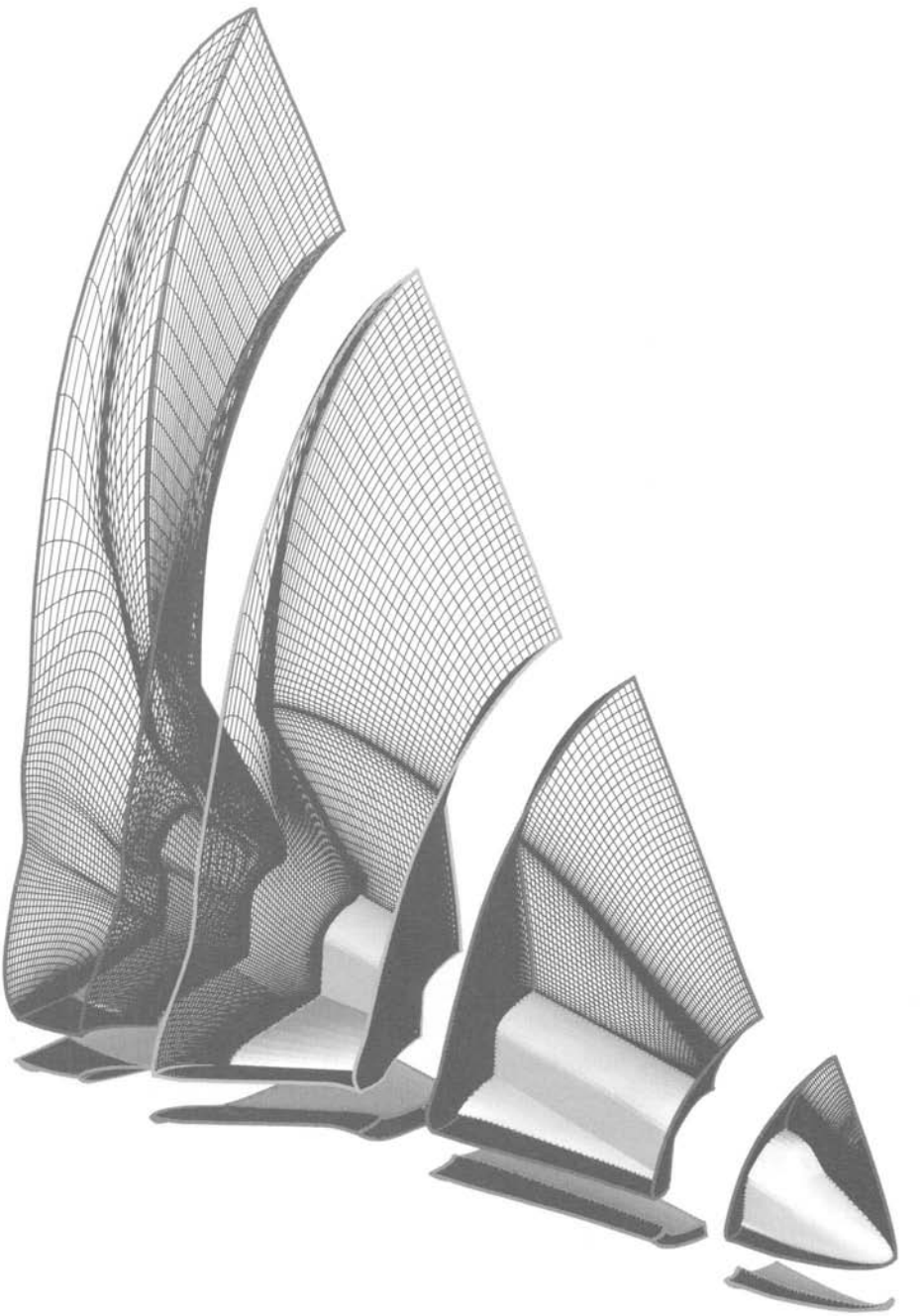

Fig. 8. Multi-block grid for the space shuttle.

Here,

$$g_{\mu,\nu} = \frac{\partial \mathbf{x}}{\partial r_\mu} \cdot \frac{\partial \mathbf{x}}{\partial r_\nu}$$

are the coefficients of the metric tensor. The equations are elliptic in nature and this means that the resulting grid has desirable smoothness properties. One of the keys to elliptic grid generation is the choice of control functions that determine the grid point spacing and grid orthogonality. The Poisson system that needs to be solved can be highly nonlinear and is difficult and time-consuming to solve. The solution to the system is generally not guaranteed to produce a single-valued grid, so care must be taken to prevent the grid from becoming multi-valued (folding).

The variational approach also produces an elliptic equation whose solution determines the locations of the grid points; see Brackbill and Saltzman (1982)† and Knupp and Steinberg (1993). The equations determining the grid point locations are derived by forming the Euler–Lagrange (variational) equations of a functional that measures properties of the grid such as orthogonality, cell area and smoothness. By weighting these different properties it is usually possible to obtain a grid with the desired features, although care must be taken to prevent folding grids.

Hyperbolic grid generation methods solve a hyperbolic set of equations to grow a grid from a boundary; see Starius (1977) and Chan and Steger (1992). Fig. 9 shows a grid generated in this way (Chan and Steger 1992).

Typically, the hyperbolic system is defined by requiring that the grid lines be orthogonal,

$$\frac{\partial \mathbf{x}}{\partial r_\mu} \cdot \frac{\partial \mathbf{x}}{\partial r_\nu} = 0, \quad \mu \neq \nu,$$

and that the cell area is specified

$$\left| \frac{\partial \mathbf{x}}{\partial \mathbf{r}} \right| = \Delta.$$

Hyperbolic methods usually always add smoothing to prevent grid lines from crossing prematurely. The outer boundary of the grid is determined as the equations are solved, and thus this method is of limited use for block-structured grids. It is, however, an extremely useful technique in the context of overlapping grids. The method is much faster than an elliptic method since the grid is constructed by marching.

5. Overlapping grid generation

The overlapping (overlaid, overset or Chimera) grid approach is similar to the block-structured approach except that the component grids are allowed

† It doesn't hurt to cite your manager whenever possible.

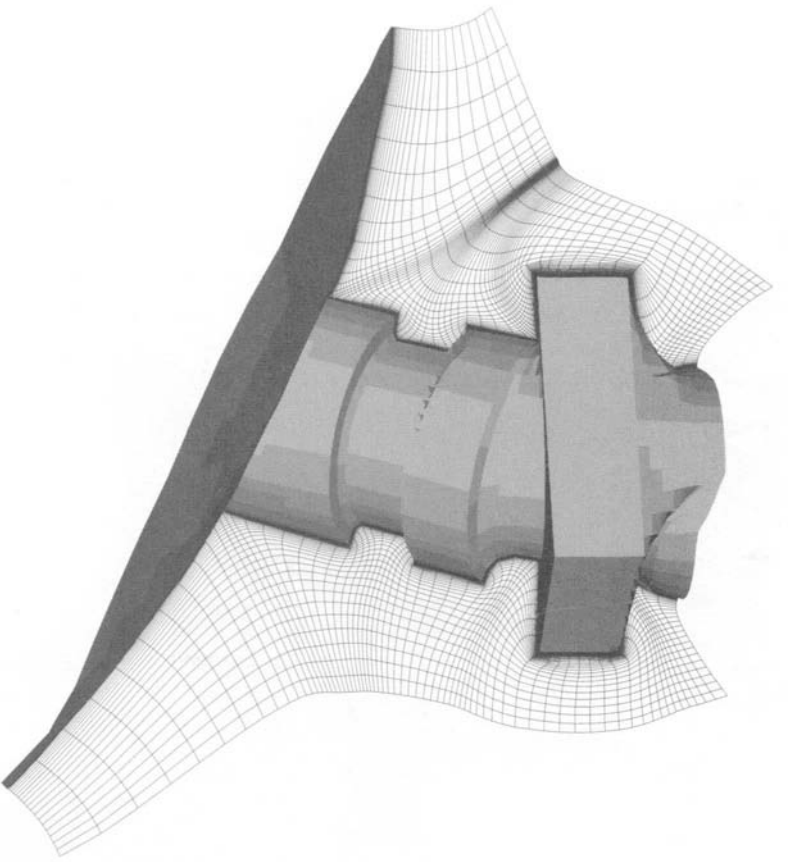

Fig. 9. Some sections of a three-dimensional grid for the liquid hydrogen feedline of the space shuttle, created with hyperbolic grid generation methods.

to overlap, instead of aligning along block boundaries; see Steger and Benek (1987), Chesshire and Henshaw (1990), Meakin (1995), Tu and Fuchs (1995). This approach has added flexibility over the block-structured technique while still retaining the efficiency of a set of logically rectangular grids. The great strength of overlapping grids is that component grids can be created in a manner that is relatively independent from the other component grids. New features can be added to the composite grid in an incremental fashion and the grid only changes locally. Fig. 10 shows part of a detailed overlapping grid for the space shuttle (Gomez and Ma 1994). The method is also attractive for moving geometries. Fig. 11 shows the overlapping grid used for a moving grid computation (Meakin 1995).

Overlapping grids are not as flexible as unstructured grids. It is difficult to get very many levels of coarser grids for a multigrid algorithm because the coarsened grids do not overlap enough. Generally, the interpolation between

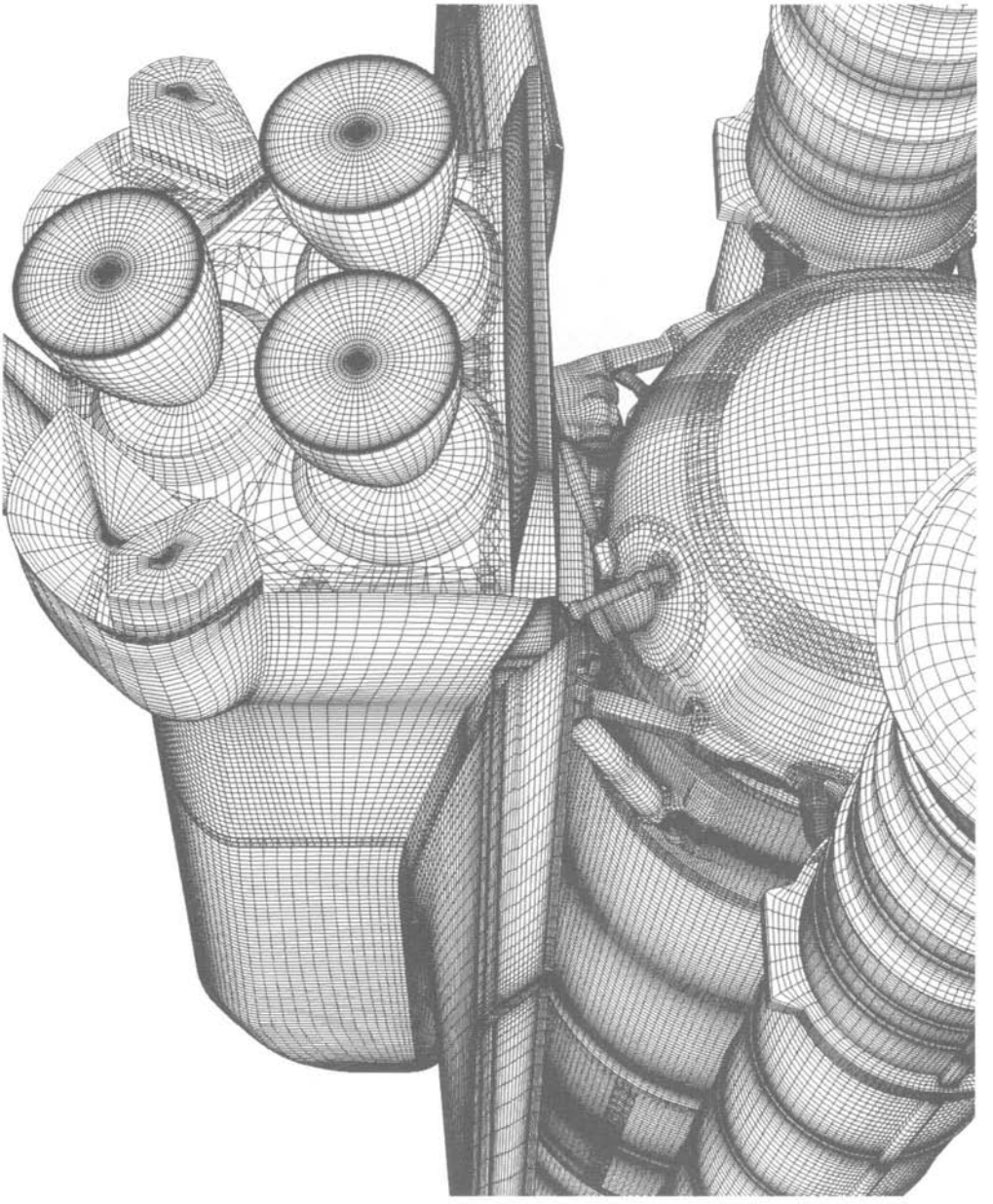

Fig. 10. Overlapping grid for the space shuttle; the three-dimensional grid has over 16 million grid points.

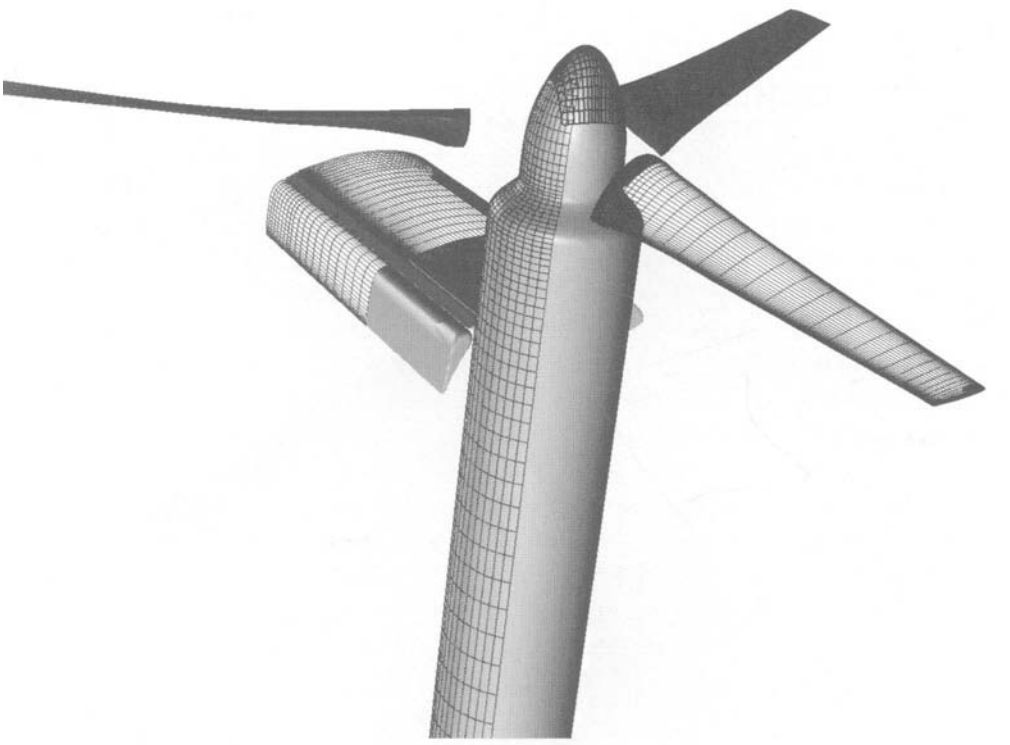

Fig. 11. Overlapping grid for the V-22 rotor and flapped wing, used in a moving grid computation.

component grids is not conservative (Chesshire and Henshaw 1994). In practice this rarely seems to be an issue. Generally, the grid generation proceeds in two steps. First, separate component grids are constructed for the various parts of the geometry, using algebraic, elliptic or hyperbolic methods. Then, given a set of component grids, the grid generation process of determining how the grids overlap can be entirely automatic. The process can fail, however, if there is insufficient overlap between components.

An approach similar to overlapping grids, but one that avoids using non-conservative interpolation, is the hybrid grid technique as shown in Fig. 12, reproduced courtesy of Dr. K.H. Kao at Nasa Lewis Research Center. The region is covered by overlapping blocks but the grid in the overlapping area is replaced by an unstructured grid of triangles.

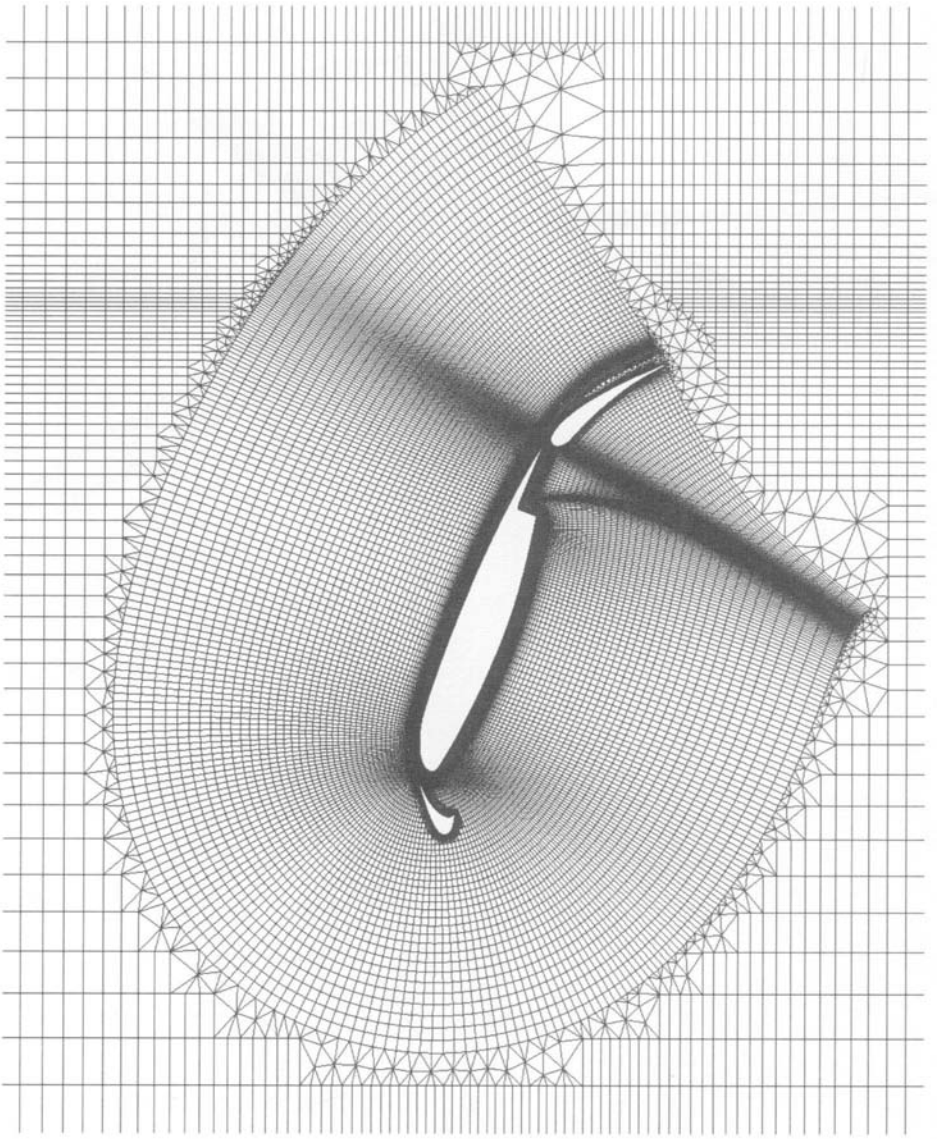

Fig. 12. A hybrid grid consisting of structured component grids joined with a region of triangles.

6. Unstructured grid generation

Unstructured grids have become very popular in recent years, due both to the influence of the finite-element method and to the increase in the power of computers. Unstructured grids and unstructured solvers have successfully demonstrated their capabilities to handle complex geometries in the demanding field of aerospace applications, an area dominated for many years by structured grids. The most flexible and automatic grid generation codes create unstructured grids. They are well suited to point-wise adaptive refinement and to moving mesh methods. See, for example, Shostko and Löhner (1995), Mavriplis (1995), Hasan, Probert, Morgan and Peraire (1995), George and Seveno (1994), Lo (1995), Johnson and Tezduyar (1995).

It is difficult to achieve good performance on unstructured grids; more memory is required and it is quite hard to apply certain fast algorithms such as implicit methods and multigrid. Attaining performance on vector, parallel and cache-based computer architectures is not easy for solvers using unstructured grids because these machines prefer that operations be performed on data that is stored locally in memory. On an unstructured grid, the data belonging to the neighbour of a point may be stored a long distance away. Moreover, triangular (and tetrahedral) meshes inherently require more elements and more computations per grid point; in three dimensions, there are some five to six times more tetrahedra per grid point than on a corresponding mesh of hexahedra. The creation of better-quality grids for hyperbolic problems and forming highly stretched elements in boundary layers continue to be active areas of research.

Fig. 13 shows a three-dimensional unstructured grid refined near the boundary, for use in a viscous flow computation. The figure has been provided by Professor Jaime Peraire.

Fig. 14, showing a cross-section of a three-dimensional grid for Yucca Mountain, is reproduced courtesy of Harold Trease, Los Alamos National Laboratory.

6.1. Un-structured grid generation approaches

Three popular methods for creating unstructured grids are

- Delaunay-based point insertion methods
- advancing front methods
- quadtree (octree) type methods.

Some of the most successful approaches use features of both the Delaunay method and the advancing front method, combining the efficiency of the former approach with the high element quality of the latter. Although quadrilateral (hexahedral) meshes are commonly used for structural problems, meshes for CFD tend to be based on triangles (tetrahedra), with perhaps

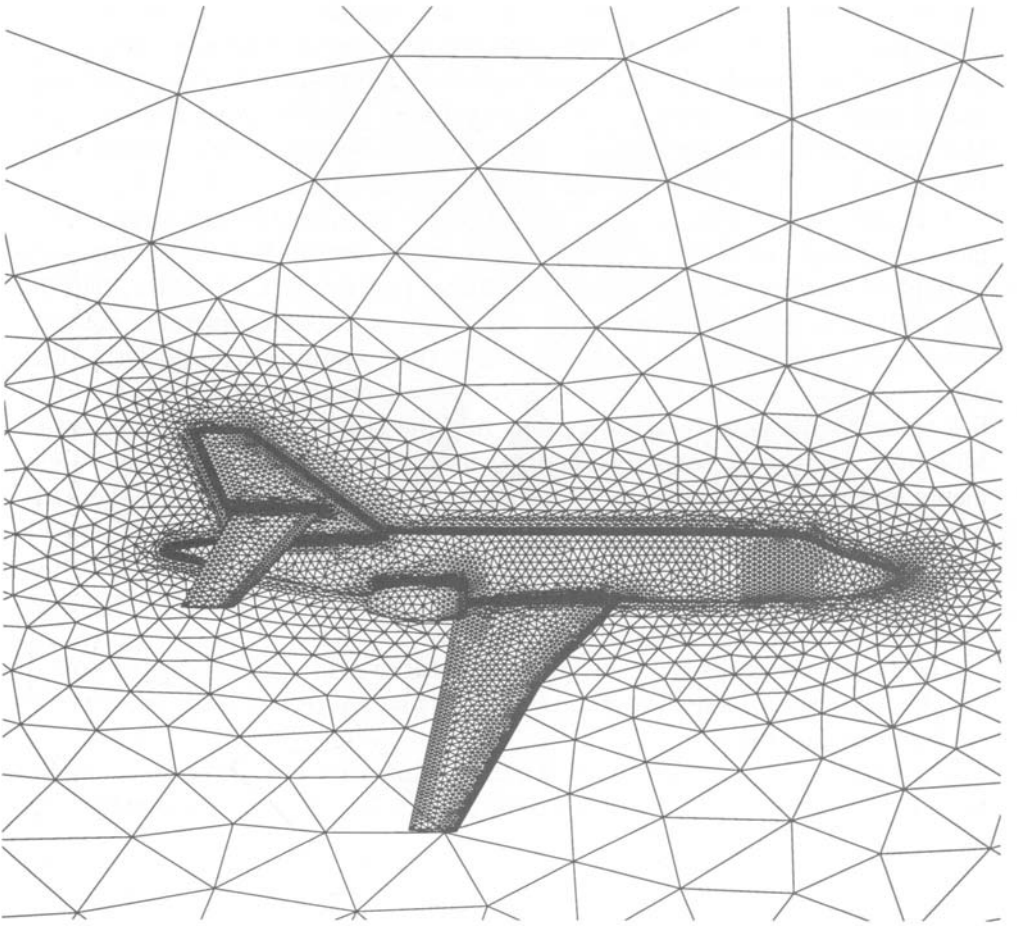

Fig. 13. Three-dimensional unstructured grid for a viscous flow computation.

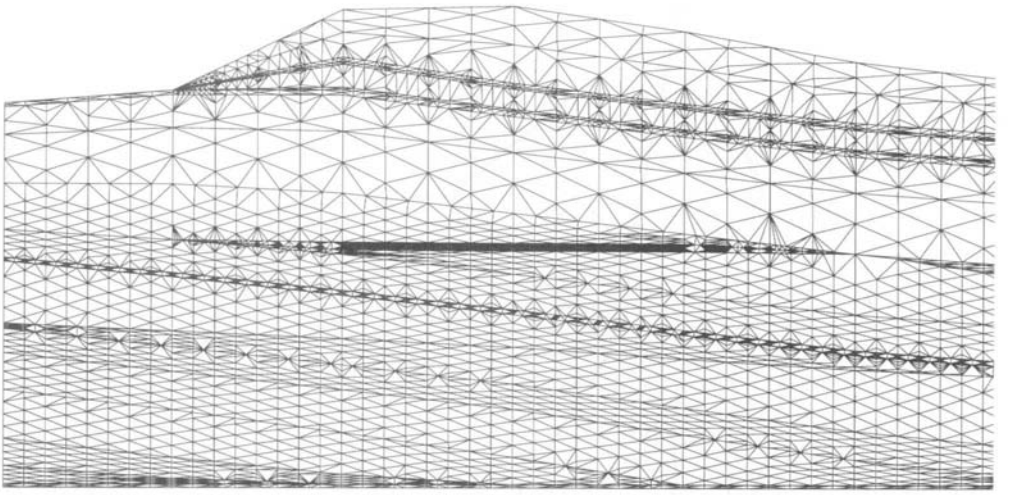

Fig. 14. Three-dimensional unstructured grid for Yucca Mountain

some quadrilateral (prismatic) elements near boundaries. There is some question as to the accuracy of using very thin tetrahedra meshes in a boundary layer; sometimes prismatic elements are used in the boundary layer. See, for example, Kallinderis, Khawaja and McMorris (1995).

Most triangulation algorithms require a function defined over the entire domain that provides the locally suggested value for the triangle size. This *background function* is often defined on a *background grid*, either an existing triangulation for the region or perhaps a rectangular grid that has been refined in a quadtree fashion.

6.2. Delaunay-based methods

The Delaunay triangulation of a set of points has the property that the circumcircle through the vertices of any triangle contains no other points; see Fig. 15. The Delaunay approach tends to create triangles that are regularly proportioned. When a region is already filled with a distribution of points, then either an incremental approach based on the Bowyer–Watson algorithm (Watson 1981, Bowyer 1981), or an advancing-front/Delaunay approach (Tannemura, Ogawa and Ogita 1983, Merriam 1991) can be used.

One of the difficulties of the Delaunay approach is maintaining the integrity of the boundary. The empty circumcircle property of Delaunay triangulations does not hold at the boundary. Care must be taken to prevent the formation of triangles whose edges cross the specified boundary. Sometimes this problem is initially ignored, the boundary being modified at the end by swapping edges and perhaps by adding new points. Another problem is that the Delaunay

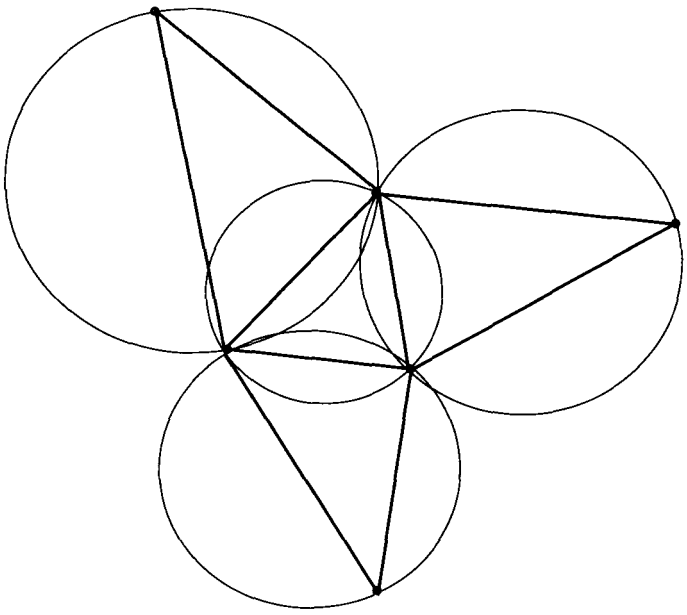

Fig. 15. In a Delaunay triangulation the circumcircles through the triangles are empty of other points.

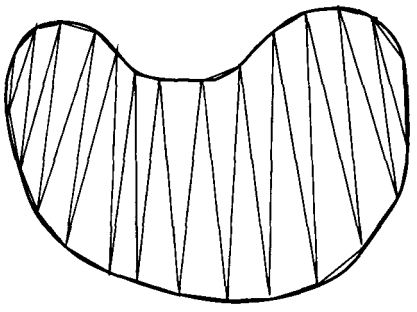

Fig. 16. The incremental Delaunay approach begins from an initial triangulation and progressively adds points; for example, points may be added at the circumcentre of the largest circumradius.

triangulation is not appropriate for creating very thin triangles in a boundary layer: some other method must be used.

In general, the positions of the grid points are not initially specified; they must be determined as part of the grid generation procedure. Incremental Delaunay methods start from a very coarse initial triangulation. Points are added one at a time, and the mesh is locally adjusted so that it remains Delaunay, using the Bowyer–Watson algorithm (Baker 1992). There are a variety of strategies for deciding where to add successive points. This point

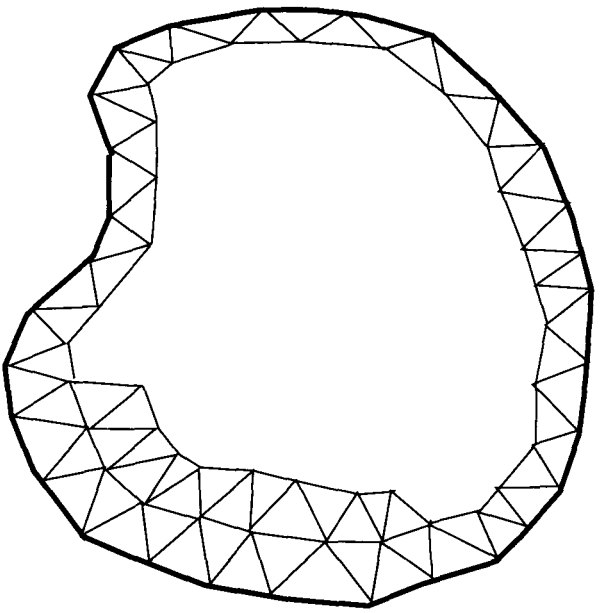

Fig. 17. The advancing-front method grows triangles from the boundaries.

placement strategy can be crucial to the quality of the resulting grid. One simple strategy involves making a list of all triangles that are too large compared to the value indicated by the background function, and incrementally adding new points to these triangles (Holmes and Synder 1988). The resulting grid can depend significantly on the order in which the list is processed. Another approach is to order the list by triangle size and add points to the largest triangle first. An alternative algorithm suggested by Rebay (1993) leads to the triangles being processed along a front that begins at the boundary. It results in a high-quality mesh similar to those produced with the advancing front method, but without some of the difficulties of that method.

6.3. Advancing front

A widely used method that results in high-quality triangulations is the advancing-front method; see for example Löhner and Parikh (1988) and Marcum and Weatherhill (1995). As the name suggests, the advancing-front method starts from the boundaries and progressively adds triangles; see Fig. 17. The triangulated region grows into the interior, forming a propagating front. Since the procedure begins at the boundary, the triangles near the boundary can be constructed to be of high quality; this is an especially important feature for many PDEs. Furthermore, the integrity of the boundary is more easily maintained than with the Delaunay approach. However, significant care must be taken when the *fronts* merge, especially when the elements are of widely

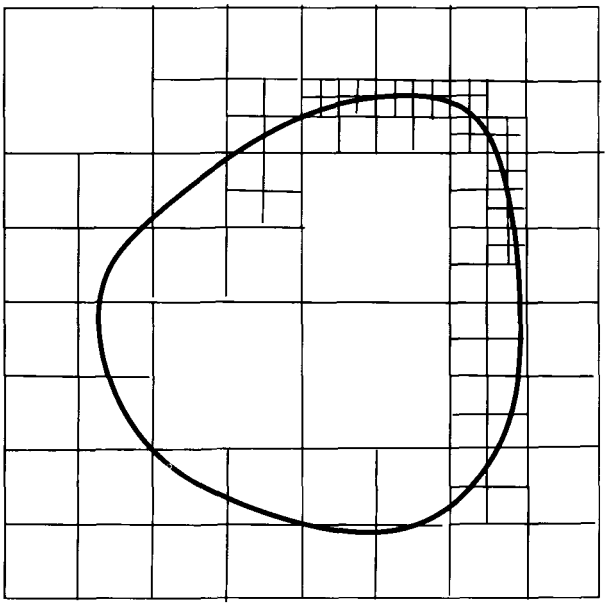

Fig. 18. The quadtree decomposition recursively sub-divides the region.

varying scale; otherwise the triangles may overlap, creating an invalid grid. This requires efficient yet robust search algorithms to determine whether a given point is close to some other part of the front. Sometimes a local Delaunay approach is used when adding new points to the front; see Mavriplis (1995) and Müller, Roe and Deconinck (1993).

There are also advancing front type methods that use quadrilaterals, but these meshes are not usually used for CFD computations; see for example Blacker (1991).

6.4. Quadtree (octree)

In simple terms, the quadtree approach proceeds by dividing the region into four rectangles and then recursively subdividing some of those rectangles into four additional rectangles, see Fig. 18. The cell size is reduced to meet certain criteria and so that the boundary is represented to sufficient resolution. The cells intersecting the boundary are replaced by polygons that follow the boundary. If a triangular mesh is required, the rectangles and polygons can be decomposed into triangles. The quadtree approach is widely used for structural problems; see for example Shephard and George (1991). It is also used to create grids for the Cartesian mesh approach, but is not commonly used to create triangular grids for unstructured flow computations. One disadvantage of the approach is that it cannot be made to conform to a

specified boundary tessellation. Furthermore, it is difficult to control the triangle shape near the boundary.

7. Conclusions

Significant advances have been made in the area of automatic grid generation in recent years. The most notable accomplishment is the success of the unstructured grid approach. This flexible approach shows the greatest promise in achieving the goal of a completely automated grid generation procedure for general applications. Structured grid methods, although less automatic (and despite announcements of their death by some in the unstructured community), will continue to be used due to their superior efficiency and accuracy. The author's personal opinion is that overlapping grids, or the hybrid grid approach that replaces the overlapping region by triangles, have great potential for many classes of problems since they are quite flexible and fast. In general, all types of hybrid grids, which combine the best features of structured grids (speed, quality and efficiency) with the best features of unstructured grids (flexibility), will probably be more widely used in the future. One probable reason why hybrid grids are not used more is the complexity of writing solvers for these grids. Improvements in software through the use of better computer languages and object-oriented design should alleviate some of these difficulties.

Despite impressive achievements to date, there is still room for improvement at almost every stage of the grid generation process. For example, the step of taking a CAD description of the geometry and forming smooth surface grids is in general very difficult. Designers of CAD systems need to be more aware of the stringent requirements needed in CFD applications. All grid generation approaches need to be more automatic, more robust, faster, and produce better quality grids. It is perhaps not unfair to say that even the most automatic system of today still requires significant human intervention. Grid generation takes too long and still requires that the person generating the grid not only be an expert in grids but also an expert in CAD and solvers.

REFERENCES

A. S. Arcilla, J. Häuser, P. R. Eiseman and J. F. Thompson (1991), *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, North-Holland, New York.

I. Babuška, J. E. Flaherty, W. D. Henshaw, J. Hopcroft, J. Oliger and T. Tezduyar (1995), *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, Springer, New York.

T. Baker (1992), 'Mesh generation for the computation of flowfields over complex aerodynamic shapes', *Computers Math. Applic.* **24**, 103–127.

M. Berger and J. Melton (1994), An accuracy test of a Cartesian grid method for steady flow in complex geometries, in *Proc. Fifth Intl. Conf. Hyperbolic Problems.*

T. D. Blacker (1991), 'Paving: A new approach to automated quadrilateral mesh generation', *International Journal For Numerical Methods in Engineering* **32**, 811–847.

A. Bowyer (1981), 'Computing Dirichlet tessellations', *Comput. J.* **24**(2), 162–166.

J. U. Brackbill and J. S. Saltzman (1982), 'Adaptive zoning for singular problems in two dimensions', *J. Comp. Phys.* **46**, 342–368.

J. E. Castillo, ed. (1991), *Mathematical Aspects of Numerical Grid Generation*, SIAM.

W. M. Chan and J. L. Steger (1992), 'Enhancements of a three-dimensional hyperbolic grid generation scheme', *Applied Mathematics and Computation* **51**, 181–205.

G. Chesshire and W. D. Henshaw (1990), 'Composite overlapping meshes for the solution of partial differential equations', *J. Comp. Phys.* **90**(1), 1–64.

G. Chesshire and W. D. Henshaw (1994), 'A scheme for conservative interpolation on overlapping grids', *SIAM J. Sci. Comput.* **15**(4), 819–845.

W. J. Coirier and K. G. Powell (1995), 'An accuracy assessment of Cartesian-mesh approaches for the Euler equations', *J. Comp. Phys.* **117**, 121–131.

P. R. Eiseman (1985), 'Grid generation for fluid mechanics computations', *Annual Review of Fluid Mechanics* **17**, 487–522.

P. L. George (1991), *Automatic Mesh Generation. Applications to Finite Element Methods*, Wiley, New York.

P. L. George and E. Seveno (1994), 'The advancing-front mesh generation method revisited', *International Journal For Numerical Methods in Engineering* **37**, 3605–3619.

R. J. Gomez and E. C. Ma (1994), Validation of a large scale chimera grid system for the space shuttle launch vehicle, Technical Report AIAA-94-1859, AIAA 12th Applied Aerodynamics Conference.

O. Hasan, E. J. Probert, K. Morgan and J. Peraire (1995), 'Mesh generation and adaptivity for the solution of compressible viscous high speed flow', *International Journal For Numerical Methods in Engineering* **38**, 1123–1148.

D. Holmes and D. Synder (1988), The generation of unstructured triangular meshes using Delaunay triangulation, in *Proceedings, Second International Conference on Numerical Grid Generation for Computational Fluid Mechanics* (S. Sengupta, J. Hauser, P. Eiseman and J. Thompson, eds), Pineridge Press, Swansea, UK.

A. A. Johnson and T. E. Tezduyar (1995), Mesh generation and update strategies for parallel computation of 3D flow problems, in *Computational Mechanics '95: Theory and Applications, Proceedings of the International Conference on Computational Engineering Science* (S. Sengupta, J. Hauser, P. Eiseman and J. Thompson, eds), Vol. 1, Pineridge Press, Swansea, UK.

Y. Kallinderis, A. Khawaja and H. McMorris (1995), Hybrid prismatic/tetrahedral grid generation for complex geometries, Technical Report AIAA 95-0211, AIAA 33rd Aerospace Sciences Mtg, Reno, RV.

P. Knupp and S. Steinberg (1993), *Fundamentals of Grid Generation*, CRC Press, Boca Raton.

S. H. Lo (1995), 'Automatic mesh generation over intersecting surfaces', *International Journal For Numerical Methods in Engineering* **38**, 943–954.

R. Löhner (1987), 'Finite elements in CFD: What lies ahead', *International Journal For Numerical Methods in Engineering* **24**, 1741–1756.

R. Löhner and P. Parikh (1988), 'Three-dimensional grid generation by the advancing front method', *International Journal For Numerical Methods in Fluids* **8**, 1135–1149.

D. L. Marcum and N. P. Weatherhill (1995), 'Unstructured grid generation using iterative point insertion and local reconnection', *AIAA J.* **33**, 1619–1625.

D. J. Mavriplis (1995), 'An advancing front Delaunay triangulation algorithm designed for robustness', *J. Comp. Phys.* **117**, 90–101.

R. L. Meakin (1995), The chimera method of simulation for unsteady three-dimensional viscous flow, in *CFD Review* (M. Hafez and K. Oshima, eds), Wiley, New York, pp. 70–86.

M. Merriam (1991), An efficient advancing front algorithm for Delaunay triangulation, Technical Report AIAA 91-0792, AIAA 29th Aerospace Sciences Mtg, Reno, NV.

J. D. Müller, P. L. Roe and H. Deconinck (1993), 'A frontal approach for internal node generation in Delaunay triangulations', *International Journal For Numerical Methods in Fluids* **17**(3), 241–256.

S. Rebay (1993), 'Efficient unstructured mesh generation by means of Delaunay triangulation and the Bowyer–Watson algorithm', *J. Comp. Phys.* **106**, 125–138.

M. Shephard and M. George (1991), 'Automatic three-dimensional mesh generation by the finite-octree technique', *International Journal For Numerical Methods in Engineering* **32**(4), 709–747.

A. Shostko and R. Löhner (1995), 'Three-dimensional parallel unstructured grid generation', *International Journal For Numerical Methods in Engineering* **38**, 905–925.

R. L. Sorenson (1986), Three-dimensional elliptic grid generation about fighter aircraft for zonal finite-difference computations, Technical Report AIAA 86-0429, AIAA 24th Aerospace Sciences Mtg, Reno, NV.

S. P. Spekreijse (1995), 'Elliptic grid generation based on Laplace equations and algebraic transformations', *J. Comp. Phys.* **118**, 38–61.

S. P. Spekreijse, J. W. Boerstoel, P. L. Vitagliano and J. L. Kuyvenhoven (1992), Domain modeling and grid generation for multi-block structured grids with application to aerodynamic and hydrodynamic configurations, in *Proceedings, Software Systems for Surface Modeling and Grid Generation* (R. Smith, ed.), NASA Conference Publication 3143, pp. 207–229.

G. Starius (1977), 'Constructing orthogonal curvilinear meshes by solving initial value problems', *Numer. Math.* **28**, 25–48.

J. L. Steger and J. A. Benek (1987), 'On the use of composite grid schemes in computational aerodynamics', *Computer Methods in Applied Mechanics and Engineering* **64**, 301–320.

M. Tannemura, T. Ogawa and N. Ogita (1983), 'A new algorithm for three-dimensional Voronoi tessellation', *J. Comp. Phys.* **51**, 191–207.

J. F. Thompson (1987), 'A general three-dimensional elliptic grid generation system on a composite block structure', *Computer Methods in Applied Mechanics and Engineering* **64**, 377–411.

J. F. Thompson (1988), 'A composite grid generation code for general 3D regions – the Eagle code', *AIAA J.* **26**, 271.

J. F. Thompson, Z. U. A. Warsi and C. W. Mastin (1985), *Numerical Grid Generation*, North-Holland, New York.

J. Y. Tu and L. Fuchs (1995), 'Calculation of flows using three-dimensional overlapping grids and multigrid methods', *International Journal For Numerical Methods in Engineering* **38**, 259–282.

D. F. Watson (1981), 'Computing the n-dimensional Delaunay tesselation with applications to Voronoi Polytopes', *Comput. J.* **24**, 167–172.

N. P. Weatherhill et al. (1994), *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Pineridge Press, Swansea, UK.