AIAA Meeting Papers on Disc, January 1996
A9618009, AIAA Paper 96-0032

An object-oriented approach to hybrid structured/unstructured grid generation

D. Banks
California, Univ., Davis; von Karman Inst. for Fluid Dynamics, Rhode-St.-Genese, Belgium

J.-D. Mueller
Michigan, Univ., Ann Arbor; von Karman Inst. for Fluid Dynamics, Rhode-St.-Genese, Belgium

P. Vankeirsbilck
von Karman Inst. for Fluid Dynamics, Rhode-St.-Genese, Belgium

AIAA, Aerospace Sciences Meeting and Exhibit, 34th, Reno, NV, Jan. 15-18, 1996

A flexible strategy for combining unstructured isotropic grids with structured anisotropic layers based on an object-oriented framework is proposed. The boundaries between the two regions are controlled by the cell aspect-ratio in the structured part. A new matching algorithm is presented that adapts the angles in the hyperbolically generated structured part from pi/2 at the wall towards pi/3 at the isotropic limit. In this way a smooth and regular transition to the unstructured part is achieved. Preliminary results demonstrate the high quality of the hybrid meshes generated by this approach. (Author)

AN OBJECT-ORIENTED APPROACH TO HYBRID STRUCTURED/UNSTRUCTURED GRID GENERATION

D. Banks* J.-D. Müller† P. Vankeirsbilck‡
CFD-Group - von Karman Institute for Fluid Dynamics
Steenweg op Waterloo 72
B-1640 Sint-Genesius-Rode
Belgium

Abstract

A flexible strategy for combining unstructured isotropic grids with structured anisotropic layers based on an object-oriented framework is proposed. The boundaries between the two regions are controlled by the cell aspect-ratio in the structured part. A new matching algorithm is presented that adapts the angles in the hyperbolically generated structured part from $\pi/2$ at the wall towards $\pi/3$ at the isotropic limit. In this way a smooth and regular transition to the unstructured part is achieved. Preliminary results demonstrate the high quality of the hybrid meshes generated by this approach.

1. Introduction

Despite advances in recent years on the frontiers of both structured and unstructured grid generation, an algorithm that can reliably produce high quality viscous grids for complex geometries without an excessive amount of human intervention remains elusive. A possible solution to this problem may be found in hybrid methods that seek to exploit the strengths of each method where appropriate while avoiding the weaknesses that slow down the generation process.

A discussion of the various grid generation methods used is put forward in a historical perspective that starts with structured meshes in section 2 where the hyperbolic method is described in some detail as it will be incorporated into our hybrid algorithm. Section 3 presents the unstructured approach invented to remedy the excessive pre-processing time of structured multi-blocked mesh systems when dealing with complex geometries. The section details the Delaunay algorithm and reviews variations of the algorithm in order to incorporate stretching into isotropic, unstructured meshes. Section 4 details the requirements imposed on a truely flexible mesh generation method. A Discussion of object-oriented design and our motives for using it are given in section 5. Finally, some preliminary results will be presented in section 6 that show the high quality grids that can be generated with the methods developed so far.

*MAE Dep., U.C. Davis - von Karman Institute
†Univ. of Michigan - von Karman Institute
‡von Karman Institute

2. Structured Grid Generation

The three primary options available in structured gridding are algebraic, elliptic and hyperbolic methods. The first two lend themselves well to generating multi-block grids for complex configurations while hyperbolic generators have mostly been used to generate overset grids. A thorough review of structured gridding methods (as well as unstructured) is available in [1]. In the present work, the technique used to generate high aspect ratio structured cells in the viscous region is a hyperbolic method based on the work of [2, 3]. Grid points are placed according to the specification of the cell volumes and angles

$$\begin{aligned} \frac{\vec{r}_\xi \cdot \vec{r}_\eta}{|\vec{r}_\xi||\vec{r}_\eta|} &= \cos\theta, \\ \vec{r}_\xi \times \vec{r}_\eta &= \Delta V. \end{aligned}$$

After linearization about a known state $\vec{r}_o$, these equations become

$$A_o (\vec{r} - \vec{r}_o)_\xi + B_o (\vec{r} - \vec{r}_o)_\eta = \vec{f}.$$

where,

$$A_o = \begin{bmatrix} \frac{x_{o_\eta}}{|\vec{r}_{o_\xi}||\vec{r}_{o_\eta}|} - \frac{\cos\theta_o x_{o_\xi}}{|\vec{r}_{o_\xi}|^2} & \frac{y_{o_\eta}}{|\vec{r}_{o_\xi}||\vec{r}_{o_\eta}|} - \frac{\cos\theta_o y_{o_\xi}}{|\vec{r}_{o_\xi}|^2} \\ y_{o_\eta} & -x_{o_\xi} \end{bmatrix},$$

$$B_o = \begin{bmatrix} \frac{x_{o_\xi}}{|\vec{r}_{o_\xi}||\vec{r}_{o_\eta}|} - \frac{\cos\theta_o x_{o_\eta}}{|\vec{r}_{o_\eta}|^2} & \frac{y_{o_\xi}}{|\vec{r}_{o_\xi}||\vec{r}_{o_\eta}|} - \frac{\cos\theta_o y_{o_\eta}}{|\vec{r}_{o_\eta}|^2} \\ -y_{o_\eta} & y_{o_\eta} \end{bmatrix},$$

$$\vec{f} = \begin{bmatrix} cos\theta + cos\theta_o \\ \Delta V - \Delta V_o \end{bmatrix}.$$

Analysis of the matrix ${B_o}^{-1}A_o$ reveals that this system is hyperbolic which indicates that efficient marching schemes can be used to generate the grid starting from an initial point distribution on the geometry surface. With this in mind, the equations are put in delta form and descretized forward in η and central in ξ to give

$$\left[I + \left({B_o}^{-1}A_o\right)\delta_\xi\right](\vec{r}_{i,j+1} - \vec{r}_{i,j}) = \vec{r}_{o_\eta}.$$

As per Chan and Steger [2], a second order explicit (ϵ_{2e}) and implicit (ϵ_{2i}) viscosity and an implicit factor (α) are

then added to the equations to enhance the stability of the marching scheme

$$\left[I + (1+\alpha)\left(B_o{}^{-1}A_o\right)\delta_\xi + \epsilon_{2i}\Delta\nabla_\xi\right]\left(\vec{r}_{i,j+1} - \vec{r}_{i,j}\right) = \vec{r}_{o_\eta} + \left(\epsilon_{2e}\Delta\nabla_\xi\right)\vec{r}_{i,j}.$$

The value of ϵ_{2e} is based locally on the η distance from the wall, the aspect ratio, a spacing convergence factor, and an angular convergence factor. The implicit viscosity ϵ_{2i} is a constant times the explicit viscosity ϵ_{2e}. In addition, the implicit factor α, which is a user defined constant, effectively introduces an artificial viscosity in the η direction. Further details, including the specification of the $\vec{r}_{o_\eta}$ term in the right hand side can be found in reference [2].

The choice to use a hyperbolic generator for the structured generation tool was based on the robustness and speed of which the algorithm is capable. As a demonstration of these qualities a grid has been generated for a box geometry with both sharp convex and concave corners (see figure 1.) Hyperbolic grid generation has been particularly successful with overset grid flow solvers. However, these require the interpolation of data between overlapping grids and thus there is a performance hit in CPU time in addition to the extra memory required for the points in overlapped regions. Through the use of a shooting algorithm, as was advocated in [3], the angle and volume terms can be adjusted iteratively to match the outer boundaries. Therefore, the hyperbolic method can also be used to generate multiblocked grids. Of course, elliptic grid generation is well suited for this application as well. However, the blocking process can take on the order of man months to complete for complex geometries and so far has proven difficult to automate.

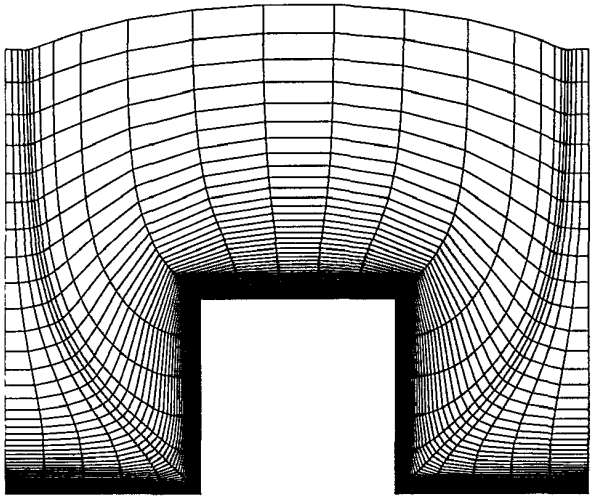

Fig. 1. Grid over a box showing the robustness of the hyperbolic generator.

3. Unstructured Triangular Mesh Generation

Unstructured mesh methods based on triangles or tetrahedra have become an important tool in industrial design for solving the Euler equations on complex geometries. They offer unrivaled simplicity and efficacy for the initial generation and the solution adaptation of the mesh. A wealth of advanced Euler solvers has been developed for unstructured meshes, such as upwind methods and multigrid acceleration that together with solution adaptation produce very robust methods with a high degree of automation.

However, the solution of the Navier-Stokes equations has remained more elusive. The demands on the mesh generation in this case, namely cells with aspect ratios of up to 1:10000 aligned with the shear layer, cells with bounded maximum angles and bounded variations in cell size between neighboring cells, leads to an overconstrained problem for purely unstructured methods. Several ways have been proposed to incorporate principles of structured, grid generation into the unstructured process, all with certain limitations.

In the following we will briefly review the basic principle of the Delaunay triangulation and describe a few of the variants to incorporate stretching. The latter part of this section will outline the limitations encountered with these algorithms.

The Delaunay Triangulation

Two triangulation algorithms have emerged as the most popular for grid generation purposes: the heuristically derived Advancing Front Method [4] and the Delaunay triangulation [5]. The mathematical basis of the latter makes it more efficient and robust, besides providing several very interesting properties of the triangulation [6].

The Delaunay triangulation prescribes a unique connectivity for a given set of grid vertices. Each vertex is surrounded by a convex polygon, the Voronoi region, that contains all points of the plane closer to this vertex than to any other. The set of edges of the polygons on which points are equidistant to two vertices is called the Voronoi diagram. A unique triangulation is obtained when joining all vertices that share an edge in the Voronoi diagram, i.e. their Voronoi regions are bordering each other. A closer look reveals that for each triangle formed between three vertices there exists an associated node of the Voronoi diagram. Since the edges of the Voronoi diagram joined at that node are the medians of the edges of the triangle, the Voronoi node is the circumcenter of the triangle. Moreover, since the three regions meeting at the node are convex, there cannot be another vertex within the circle (fig. 2). This property also holds in three dimensions.

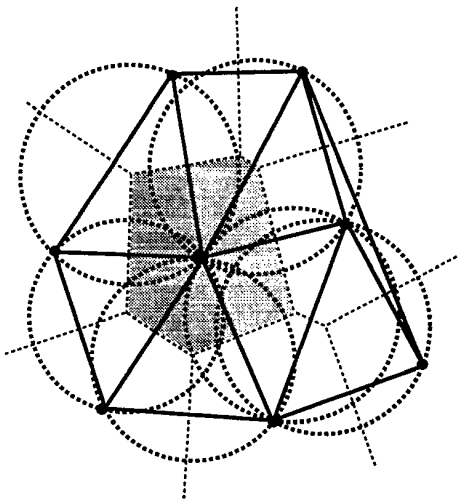

Fig. 2. A set of vertices with the Voronoi region (shaded), the Voronoi diagram (dashed lines), the circumcircles (dashed) and the Delaunay Triangulation (solid lines).

This "circumcircle" property is exploited in the Bowyer-Watson algorithm [7, 8] to construct a Delaunay triangulation by recursive insertion of vertices. Alternatively one can show that a Delaunay triangulation is the MaxMin triangulation. That is, it is the unique triangulation that maximizes the minimum angles in the grid.

Many strategies have been proposed for the generation of the interior point cloud [9, 10, 11, 12, 13] that all start with a triangulation of the boundary vertices and a successive refinement by insertion according to certain quality measures. Most of them allow to show certain bounds of quality measures as the insertion is linked to the circumcircle property. the hybrid grid generator of the present work uses the frontal Delaunay algorithm of [14]. An example grid about a three element airfoil generated with FroD is shown in figures 3 and 4.

Stretching in Delaunay Triangulations

Resolving shear layers in Navier-Stokes calculations requires cell aspect ratios of e.g. up to 1:10000 for an airplane wing section. An important problem encountered when stretching a mesh is the control of large angles in the grid in order to bound the error in the solution [15]. That is, we seek the MinMax triangulation that minimizes the maximum angle rather than the Delaunay triangulation (figure 5).

A straightforward way to remedy this situation has been proposed in [16]. The first step is to generate structured collar meshes around all components that can possibly overlap and overturn. In a second step elements are clipped from these meshes if they a) overlap

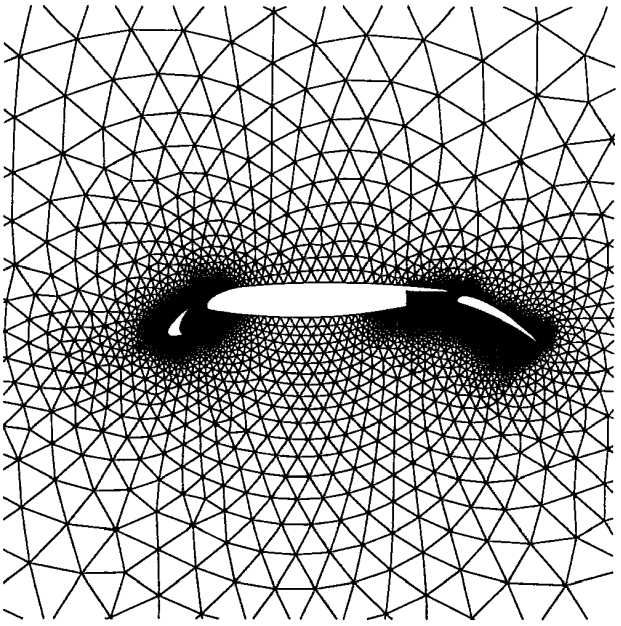

Fig. 3. Grid over a three element airfoil generated with the frontal delaunay algorithm.

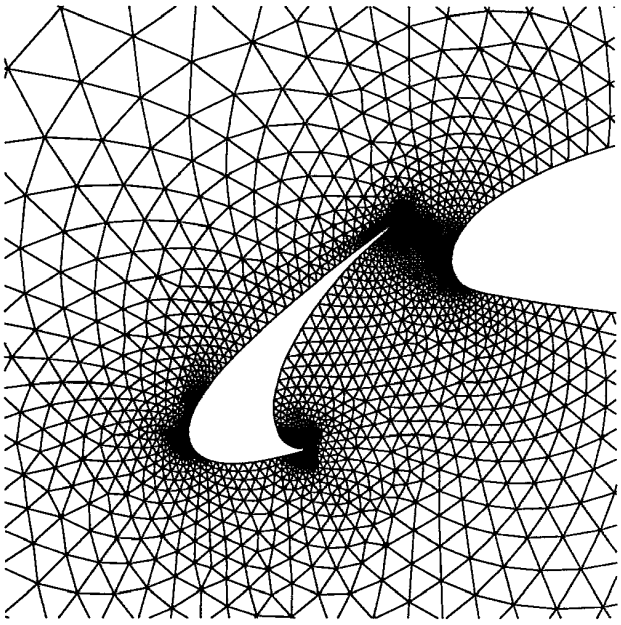

Fig. 4. Close up of the leading edge slat.

with another element, b) are overturned and exhibit a negative element surface or c) have an aspect ratio > 1. The open regions of the domain are triangulated with an isotropic process in a third step.

This concept has been taken a step further by incorporating the structured grid generation part into the unstructured method. One achieves this by modifying the point placement strategy and the triangulation algorithm in the viscous region. Rather than being placed

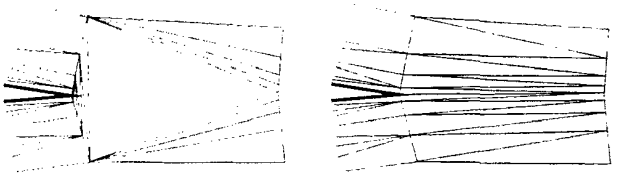

Fig. 5. Delaunay (left) and MinMax-triangulation (right) of the trailing edge of an airfoil. The Delaunay triangulation produces very obtuse cells.

at circumcenters [9, 10, 11] or edge-medians [12, 13], vertices in a stretched layer have to be "stacked" as in a structured grid. The connectivity in the viscous region can either be taken as MinMax [17] or some constraint on the Delaunay criterion that prohibits large angles [18, 14].

All methods above are reasonably successful in generating viscous grids around smooth shapes, but have problems around sharp corners and in regions where slip lines lay close to each other. In both cases the aspect ratio of the cells at the outer edge of the structured layer does not decrease to unity and the continuing isotropic process produces a grid with poor quality. It must be expected that the problems become even more pronounced in three dimensions.

4. Hybrid Grid Generation

From the previous sections it becomes clear that more flexibility is needed in matching structured and/or unstructured grid parts; a simple unstructured filling of holes in between structured grid collisions is insufficient for complex flow situations. Conversely, using an unstructured method in most of the domain and covering only the viscous layer with a structured collar grid is not satisfactory either. The algorithms reviewed above fail to satisfy the following requirements:

- to allow the matching of two structured patches without an unstructured buffer in between,
- to assure isotropy where a structured patch borders an unstructured one,
- to allow the separation of patches of different types in order to run different kinds of solvers on them,
- to permit multigrid coarsening across patch interfaces, and
- to perform these tasks with little or no user intervention.

To attain these goals, a hybrid grid generation tool should be generic and flexible enough to enable a wide variety of future extensions. Such extensions would include the coupling of 2D with 3D meshes, the coupling of unstructured and structured grid generators, the coupling of boundary element meshes with finite element meshes. In other words, we want to interpret the term "hybrid" in the widest possible sense with no one method being considered as the "base" which is to be modified or extended.

At the moment the complete hybrid algorithm is as follows:

1. March structured grids off of solid bodies with the hyperbolic method.
2. Clip cells from grids which have aspect ratios of less than one.
3. Block together grids that collide while still in the anisotropic regions via the shooting method.
4. Generate unstructured grids in the isotropic regions with the FroD method.
5. Smooth the entire domain with a laplacian operator.

The inclusion of the source terms in the hyperbolic generator has proven useful in facilitating the linking of structured and unstructured subdomains. The final grids produced have the structured quads cut into triangles. So, to enhance the smoothness of transition, the angle θ is coupled to the aspect ratio of the cell by

$$\cos\theta = 1/2 \min(\vec{r}_\xi/\vec{r}_\eta, \vec{r}_\eta/\vec{r}_\xi)$$

so that roughly equaliteral triangles are produced at the point where the structured grid becomes isotropic. For the unstructured grid tool we use the FroD algorithm in reference. Three views of a viscous grid about a NACA0012 airfoil with aspect ratio of 10^4 are shown (figures 6, 7, 8 which highlight the border between the structured and unstructured regions with a thick line. In addition, the source terms can be used in a shooting algorithm as presented in [3] to link together sections of structured grids that collide before they have become isotropic. After the cells beyond the isotropic layer and the collision of two grids have been clipped, the source terms are iteratively adjusted such that the outer boundary of the grids match. A grid for a bi-NACA0012 airfoil configuration generated with this approach is shown in figure 9.

5. Object-Oriented Programming Topics

Motivation

Object-oriented analysis, design and implementation were adopted to develop the hybrid grid generation package. Several reasons lie behind this choice which will be discussed below.

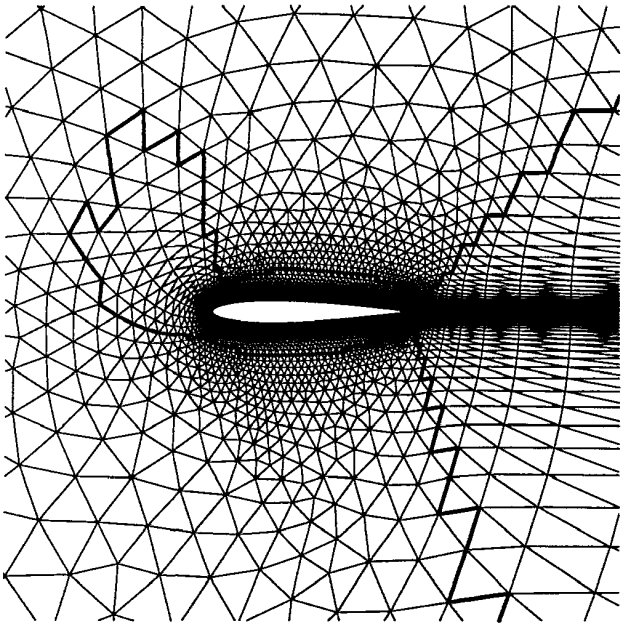

Fig. 6. Hybrid grid about a NACA0012 airfoil with aspect ratios of up to 10^4 at the boundary. The thick line highlights the transition between the structured and unstructured regions.

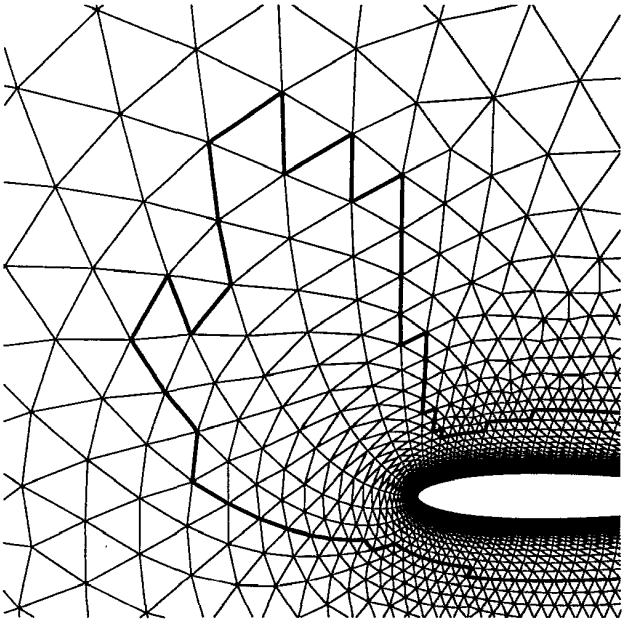

Fig. 7. Closeup of leading edge of the NACA0012 airfoil grid.

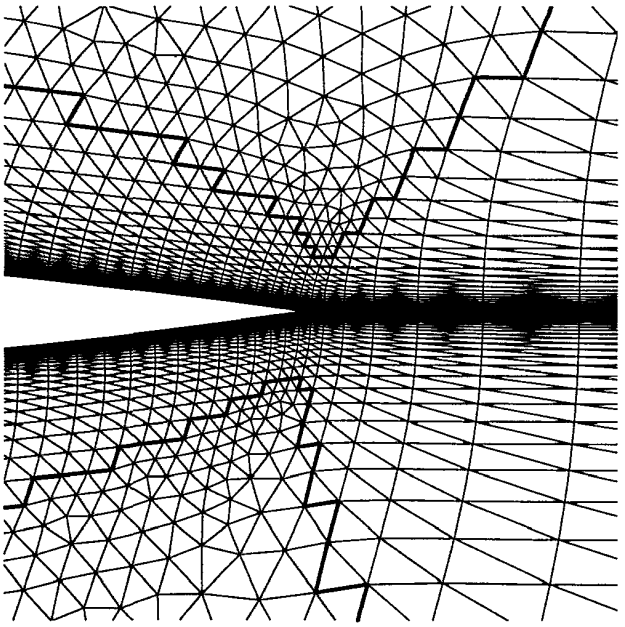

Fig. 8. Closeup of trailing edge of the NACA0012 airfoil grid.

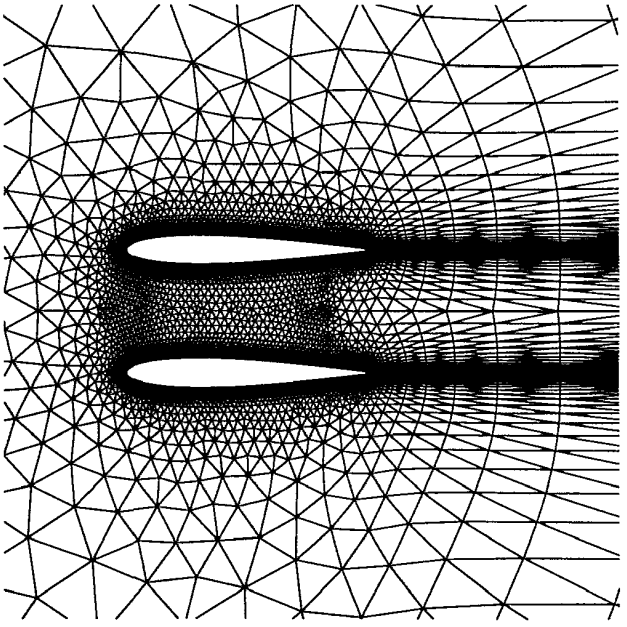

Fig. 9. Hybrid grid for a bi-NACA0012 airfoil confinguration using the shooting method to patch structured grids in the viscous region

As was mentioned in the previous section, the goal of this project is to implement a grid generation algorithm that is hybrid in the most general sense of the word. Hence, the basic components of our grid generation software system must be characterized by a high degree of *reusability*. This means that the basic components of the tool should form a kind of generic programming framework from which more specialized components can be developed through derivation (*inheritance*) or *parametrization* [19].

The fact that a hybrid grid generation tool has to cope with different grid generation algorithms whose behaviour might differ considerably also imposes more restrictive requirements related to the *modularity* of the

resulting software system. Modularity means here that the different algorithms not only be as independent as possible but also that the algorithms can be related to each other in some hierarchical manner. Algorithms can be made independent by hiding their internal data from other components via *encapsulation*.

Grid generation is an ever evolving technology leading to new or changing algorithms. Therefore, the proposed grid generation tool needs a high degree of *maintainability*. This means that when introducing a new algorithm, the implementation should only change locally within the system even though the global behaviour of the system will have changed. For instance, consider a subdomain to be a part of the complete domain which will be discretized by a single grid. The grid in each subdomain is then to be sewn together with the grids in neighboring subdomains. The data and functionality associated with a subdomain should be able to use a new grid generation algorithm without changing the implementation of the subdomain itself. In other words, grid generation algorithms must exhibit a *polymorphic* behaviour [20].

It is believed within the software engineering community that an object-oriented development process satisfies most the requirements of reusability, modularity and maintainability. The C++ implementation language [21] was adopted because of its run-time efficiency (*inlining*), the fact that a huge set of existing C-libraries can be reused and the fact that it fully supports inheritance, parametrization, encapsulation and polymorphism, [19]. For these reasons, C++ has become a de facto standard in the object-oriented scientific computing community [22].

The key element in object-oriented development is a *class*. A class is a general description of a set of data and the functionality related to that data. For instance, the data of the class `Grid` would contain an array of grid points. This data is typically hidden away and can only be accessed through specific functions. The functionality would include the computation of the cell volumes, angles, etc...

An *object* is a specific instance of a class. For instance, the object with name `grid_1` could be the viscous grid for the flap in a multi-element airfoil. Several objects can be instances of a single class.

Code Architecture

The present grid generation software system is a subsystem of a larger system called ELEMD [22]. The very abstract layer of the ELEMD system is inherited by the grid generation software system which is otherwise independent of the ELEMD system. The system has been equipped with a graphical interface since in the past this has proven to be a neccassary tool of most useful grid generation packages.

The classes that do the actual work of generating grids consist of two main parts: the GridTree part and the Generation part.

The GridTree Part

The relationship between classes in the GridTree part are mainly based on *aggregation*; i.e. a given class contains one or more instances of several other classes (*compositional hierarchy*), as is shown in fig. 10.

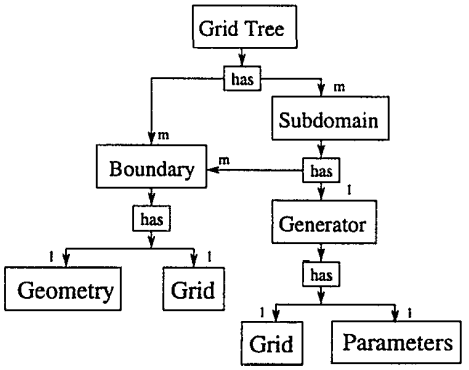

Fig. 10. The relationships in the GridTree. An 'm' indicates multiple objects.

The class `GridTree` reflects the concept in the `Tree` class of the ELEMD system. The `Tree` class contains several grid levels so as to enable multigrid acceleration when solving systems of PDE's. The `GridTree` always represents one grid level because grid generation is assumed to always occur on a single grid level. However, the class `GridTree` can contain several instances of the `Subdomain` class. Each of these instances represents a given subdomain of the total domain being gridded. At this stage, it is not specified whether these subdomains may overlap or not for reasons of generality. The class `Subdomain` also contains a set of instances of the `Boundary` class representing a part of the complete boundary enclosing the current subdomain. Each `Subdomain` object further stores a reference to a `Generator` class which will, in turn, store the grid generated inside the subdomain and the user specified parameters needed by the `Generator`. Only one grid per subdomain can exist.

The GridGen Part

The relationships between the classes in the Generation part of the tool are based on class derivation (inheritance), see fig. 11. Deriving a class `HyperbolicGenerator` from a class `Generator` means that `HyperbolicGenerator` is a kind of `Generator` and, hence, inherits all the proper-

ties of the `Generator` class. However, class `HyperbolicGenerator` has something more added to it. It can have more data and enhanced functionality.

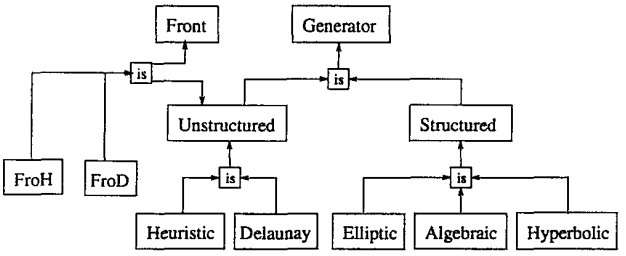

Fig. 11. The inheritance hierarchy in the Generator part.

The Generation part has a single abstract base class `Generator` from which all other types of grid generators (1D, 2D, 3D, unstructured, structured, etc...) are or can be derived. The class `Generator` itself does not have functionality but declares the common interface to all of its derived classes. The base class `Generator` is the only class that is visible in other parts of the grid generation tool. In particular the class `Subdomain` will interact with `Generator` objects but, as this class only sees the abstract base class of the Generation part, its implementation does not need to be changed whenever a new grid generation class is derived. Nevertheless, the overall look of the grid stored in the subdomain will change as will as the parameters which need to be specified by the user.

6. Results

As an illustration of the merit of hybrid grids we provide a comparison of solutions obtained on structured vs. hybrid grids for separating flow over a NACA0012 at Mach 0.5 and Re 5,000. The two grids have the same level of refinement (223 points) on the airfoil and wake surfaces. However, the hybrid grid covers a domain extending out 14 cords with 5590 nodes while the structured grid needs 11180 nodes to cover a domain with the far field at 7 cords. Figures 12 and 13 show the two grids. Figure 14 shows a comparison of the separation bubble at the trailing edge. Despite having half the nodes, the hybrid grid does a significantly better job at capturing the separation bubble location than the structured grid does. The flow separates at 0.835 vs. 0.85 chord which agrees better with previously existing solutions. Also, the separation bubble on the hybrid grid persists roughly twice as far downstream which is in better agreement as well.

A final test case of an axisymmetric hypersonic flow problem over a flared hyperboloid body is presented. The grid is shown in figure 15 and the mach isolines computed are shown in figure 16. The actual grid used to obtain the solution (which is too dense to reproduce well) has 19,516 vertices in the domain

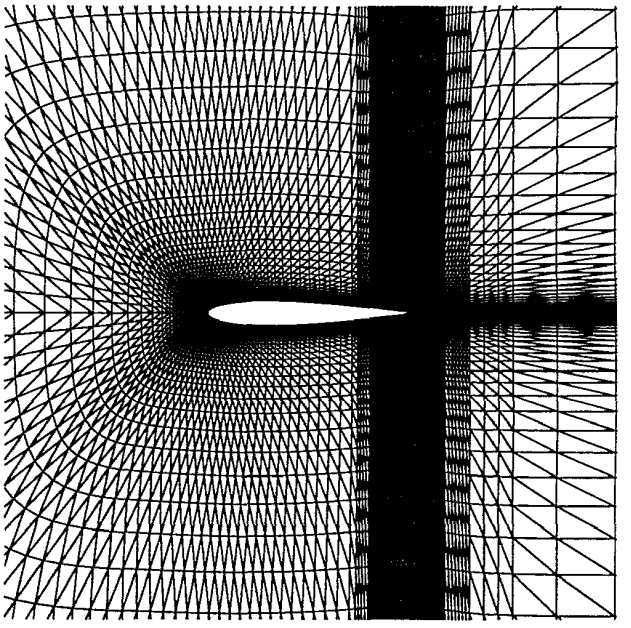

Fig. 12. 'Structured' grid about a NACA0012. The outer boundary is at 7 chords and there are 11180 nodes

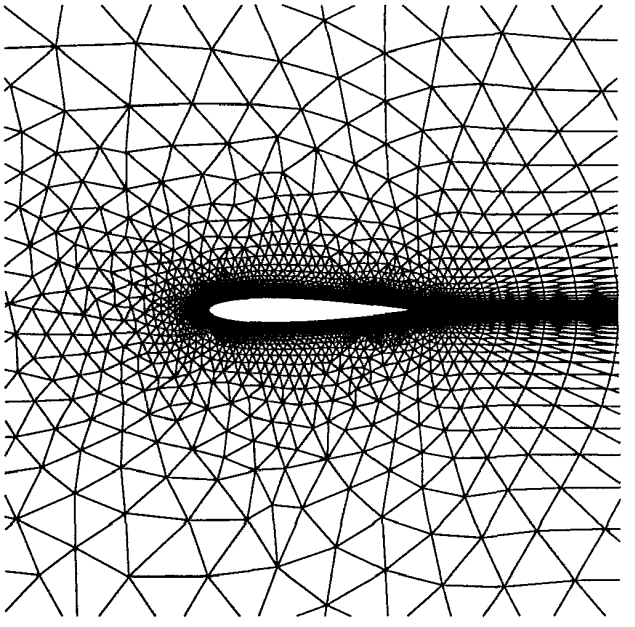

Fig. 13. Hybrid grid about a NACA0012. The outer boundary is at 14 chords and there are 5570 nodes

while the grid shown in figure 15 has only 1500. Reults were calculated for the flow conditions: $M_\infty = 8.7$, $T_\infty = 72.69K$, $T_{wall} = 310$, $Re = 6.25 \times 10^5$, and, $L = 0.05924m$.

7. Conclusion

The present hyperbolic grid generation algorithm combined with the FroD algorithm for isotropic tri-

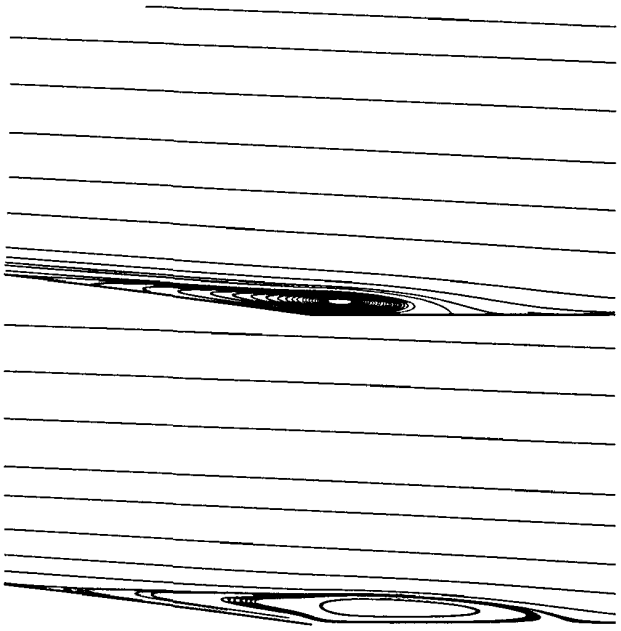

Fig. 14. comparison of separation bubble size about a NACA0012 at $M\,\mathrm{inf} = 0.5$ and $Re = 5000$ calculated on the structured grid (top) vs. the hybrid grid (bottom)

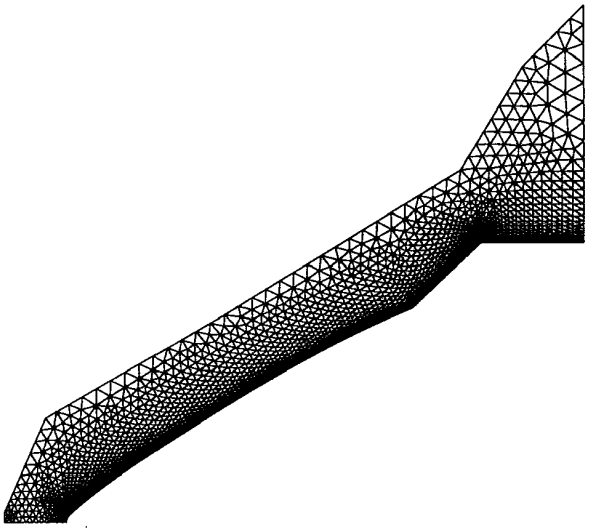

Fig. 15. Coarsenned hybrid grid for axisymmetric hypersonic flow over a flared paraboloid body. Structured grid generated with $\theta = 90°$

angulation forms a robust basis for the generation of highly stretched grids. The transition from the initially stretched structured grid regions to the isotropic unstructured grid zones is in most cases invisible. The higher degree of automation achieved in the present grid generation package enables one to generate meshes for fairly complex geometries with a restricted amount of user-input.

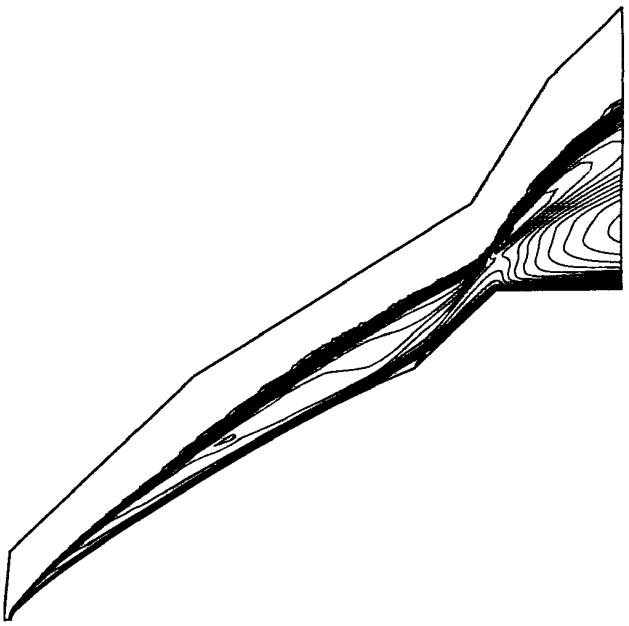

Fig. 16. Mach number distribution for freestream conditions: $M_\infty = 8.7$, $T_\infty = 72.69K$, $T_{wall} = 310K$, $Re = 6.25 \times 10^5$, and, $L = 0.05924m$

The object-oriented approach adopted for the project has led to a comprehensive programming framework for grid generation algorithms. This framework allows to combine different grid generation algorithms on an equal basis and, hence, facilitates the further implementation of hybrid grid generation technology.

References

1. Joe F. Thompson and Nigel P. Weatherill. Aspects of numerical grid generation: Current science and art. In *Proceedings of the 11th AIAA Applied Aerodynamics Conference*, Aug. 1993.

2. W. Chan and J.L. Steger. Enhancements of a three-dimensional hyperbolic grid generation scheme. *Applied Math. and Comput.*, 51:181–205, 1992.

3. J. Q. Cordova and T.J. Barth. Grid generation for general 2-d regions using hyperbolic equations. *AIAA-paper 88-0520*, 1988.

4. J. Peraire, M. Vahdati, K. Morgan, and O.C. Zienkiewicz. Adaptive remeshing for compressible flow computations. *Journal of Computational Physics*, 72, 1987.

5. B. Delaunay. Sur la sphère vide. *Bull. Acad. Science USSR VII: Class. Sci. Mat. Nat. 793-800*, 1934.

6. T.J. Barth. Aspects of unstructured grids and Finite Volume solvers for the Euler and Navier-Stokes equations. Technical Report R-787, AGARD, 1992.

7. A. Bowyer. Computing Dirichlet tessellations. *The Computer Journal*, 24(2):162–166, 1981.

8. D.F. Watson. Computing the n-dimensional Delaunay tesselation with application to Voronoi polytopes. *The Computer Journal*, 24(2):167–171, 1981.

9. D.G. Holmes and D.D. Snyder. The generation of unstructured triangular meshes using Delaunay triangulation. *Proceedings of the Second Conference on Grid Generation in Computational Fluid Dynamics, Pineridge Press, Swansea,*, 1988.

10. P. Chew. Guaranteed-quality triangular meshes. Technical Report TR-89-98993, Cornell University, 1989.

11. J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. Techn. Rep. UCB/CSD 92/694, UCB, June 1992.

12. S. Rebay. Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm. *Journal of Computational Physics*, 106(1):125–38, 1993.

13. J.-D. Müller, P.L. Roe, and H. Deconinck. A frontal approach for internal node generation in Delaunay triangulations. *Int. J. of Num. Meth. in Fluids*, 17(3):241–56, 1993.

14. J.-D. Müller. Quality estimates and stretched meshes based on delaunay triangulations. *AIAA-Journal*, 32(12), December 1994.

15. I. Babuška and A.K. Aziz. On the angle condition in the Finite Element method. *SIAM J. Num. Anal.*, 13(2):214–26, 1976.

16. R. Löhner. Matching semi-structured and unstructured grids for Navier-Stokes calculations. *AIAA-paper*, (93-3348-CP), 1993.

17. D. L. Marcum. Generation of unstructured grids for viscous flow applications. *AIAA-paper*, (95-0212), January 1995.

18. S. Pirzadeh. Unstructured viscous grid generation by advancing-layers method. *AIAA*, 32(8), August 1994.

19. K. Srinivasan and Sundaresan Jayaraman. Comparison of object-oriented programming languages: The enterprise modeling framework perspective. *Journal of Object-Oriented Programming*, 7(3):27–32, June 1994.

20. S.B. Lippman. C++ primer : Objects and datums. *C++ Report*, 6(5):20–26, June 1994.

21. Bjarne Stroustrup Margaret A. Ellis. *The Annotated C++ Reference Manual.* Addison Wesley, 1990.

22. Patrick Vankeirsbilck and Gert Nelissen. ELEMD: An object-oriented software system for grid elements with multiple discretizations for solving PDE's. In SIAM, editor, *Second Annual Object-Oriented Numerics Conf.*, Sunriver, Oregon, April 24-27 1994. Rogue Wave Software.