AIAA JOURNAL
Vol. 34, No. 6, June 1996

# Automatic Grid Generation and Flow Solution for Complex Geometries

Richard J. Smith* and Leslie J. Johnston†
*UMIST, Manchester M60 1QD, England, United Kingdom*

**The development of a new method for the automatic generation of computational grids around complex aerodynamic configurations is described. The grid-generation procedure uses a Cartesian cell, embedded, unstructured approach that enables the efficient placement of computational cells to resolve important flow regions close to the surface. The main difference between the present method and most existing grid-generation techniques is the avoidance of the need to explicitly generate a surface grid. This new approach is easily automated and is capable of producing grids around arbitrarily complex geometries. The capabilities of the method are demonstrated with a selection of inviscid transonic flows solutions.**

## Introduction

COMPUTATIONAL fluid dynamics (CFD) analysis is finding increasingly widespread use within the aerospace industry as recent published results highlight.[1,2] The application of CFD to complex configurations for design purposes, however, is still a long way from being considered routine. In particular, the number of existing grid-generation systems is testament to the fact that no one technique can universally meet the current demands of the aerospace industry. Broadly speaking there are four main types of grid-generation techniques: multiblock, unstructured, overlapping, and unaligned.

Multiblock[3] is the most well established of the current grid-generation techniques, and typically it can treat complex geometries with high accuracy, however, the time taken to generate the associated grids is not compatible with current design time scales. Also multiblock requires a high level of user expertise in order to achieve optimal usage. Having identified the weaknesses of multiblock methods, other approaches have evolved, such as unstructured[4] and overlapping[5] grids. Although quite general in their ability to treat complex geometries, both techniques require the explicit generation of a surface grid. Surface grids for complex geometries can again require a high level of user expertise and interaction, thus compromising the turn-around time. The advantage of these unstructured methods, however, is that once a surface grid has been obtained the field grid can be generated automatically.

The final grouping of grid generation techniques is that of unaligned.[6] The removal of the requirement for a surface conforming grid greatly simplifies the overall process and reduces the turn-around time for complex geometries, but the accuracy of such techniques is dependant on the user placing grids manually in areas of interest before starting the calculation. The treatment of high Reynolds number viscous flows does not appear to fit in with the unaligned approach, and this is a major drawback for obtaining essential design information, such as skin friction, and predicting the influence of shock/boundary-layer interactions.

The aim of the present work is to develop a computational grid-generation technique for complex configurations requiring minimal user expertise and minimal interaction, in order to produce a turn-around time compatible with the engineering design environment, while providing acceptable accuracy levels in the flow solutions. The technique generates the whole computational grid in one stage, with the surface grid produced as the final byproduct of the method. The grid-generation method uses an embedded unstructured Cartesian cell approach with a clustering of cells, initially close to the surface geometry. This approach is similar to that of Baehmann et al.,[7] with two main differences: the first is the generation of the interior cells before the surface cells, and the second is the vertices of the surface cells are pulled onto the actual surface, thus keeping the quadrilateral cell structure, rather than allowing cut cells. The generation of the interior cells before the surface cells is driven by the application of the current method to external flowfields around aerodynamic geometries. Also the Cartesian grid is then reconnected to form a dual grid (covered later), not a wholly triangular or quadrilateral grid, upon which the flow solutions are then obtained. The grid generation method is highly automated and is capable of producing a grid around any number of geometry elements.

A brief outline of the method has been presented previously[8]; thus, the aim of the present paper is to provide more details, with a description of the inviscid flow solver. The results section includes an evaluation of the dependence of the flow solution on the embedded level of the grid for a NACA 0012 aerofoil. Also there is a comparison between the present method and experiment for a two-element aerofoil geometry at a transonic flow condition, and a further example for stores release is shown in order to demonstrate the capabilities of the method.

## Grid-Generation Procedure

### Introduction

The basic requirement of any grid-generation method is to provide a discretized flow domain on which to implement the physical flow model and associated boundary conditions. The final accuracy of the flow solution is greatly influenced by the surface and the field definition of the flow domain provided by the grid-generation method. The resolution of the surface within the grid-generation method is important on two accounts. First, an adequate number of points is required to implement accurately the surface boundary conditions. Second, clustering points on the surface, especially in regions of high curvature and shock waves, is required to resolve the large flow gradients that exist there. The field grid resolution is a compromise between having a globally fine grid that resolves all flowfield gradients sufficiently accurately but is still coarse enough to allow an acceptable computation time.

The majority of existing grid-generation techniques, be they unstructured or block structured, have two distinct stages. First, a surface grid is generated, and then the field grid is generated, using the surface grid as a boundary condition. By the use of a Cartesian cell embedding technique in the present work, a method has been developed that reduces grid generation to a single automatic stage. The technique starts with an initial single outer boundary cell. This cell and subsequent cells are then subdivided if they contain or intersect the geometry. Following this procedure, cells are clustered around the geometry surface, thus satisfying surface definition

Received Dec. 6, 1994; presented as Paper 95-0214 at the AIAA 33rd Aerospace Sciences Meeting, Reno, NV, Jan. 9–12, 1995; revision received June 16, 1995; accepted for publication June 23, 1995. 

*Research Engineer. Department of Mechanical Engineering. Student Member AIAA.
†Lecturer, Department of Mechanical Engineering. Member AIAA.

requirements. To ensure adequate field definition, a constraint is imposed that prevents any two neighboring cells differing by more than one level of subdivision. This also has the benefit of preventing abrupt changes in cell area, which is deemed to be desirable in the interests of flow solution accuracy.

The computational cells produced by the present method have no implicit connectivity. This is overcome by imposing a binary tree data structure on the cells as they are generated, which is then the basis for an efficient search algorithm used during the grid-generation procedure itself. A major benefit of this unstructured approach is the relative ease and generality for the addition and deletion of cells.

### Grid-Generation Algorithm

Generation of the computational field grid is achieved using the following simple algorithm:

1) Define an outer boundary box for the flow domain.

2) Interrogate the CAD representation of the surface geometry.

3) Subdivide the box and subsequent boxes if they contain or intersect the geometry.

4) Detect and remove cells inside the geometry.

5) Return to 2 and repeat the procedure until the required level of embedded grid is reached.

At this point, the resulting grid is unaligned to the actual surface geometry but is aligned to a stepped representation of the solid surface. Upon further investigation the method was found to require two further steps before acceptable flow solutions were obtained.

6) Move the surface cell vertices onto the geometry surface.

7) Finally, smooth the grid.

The next section covers in detail the various features of the grid-generation routine.

## Features of the Procedure

### Cell Connectivity

The cell-generation algorithm necessitates the searching of both the input CAD geometry elements and the generated cells. Because of the nature of the process, cells are generated in a nonsequential (unstructured) manner. This requires a form of connectivity to be imposed on the cells. In this method, connectivity is provided by a binary tree, which is the basis for an efficient searching routine covered in a later section. A feature of a binary tree data structure is the relative ease of addition and deletion of data. Each data node (parent) can be connected to either one or two nodes (children). The connectivity held by the parent node is as follows: its own address and pointers to left and right children. If a child is not present, it is indicated by the respective pointer in the parent being zero. Relating the described tree structure to the generated cells allows a cell to be interpreted as a node. By first subdividing a cell horizontally, two new cells are produced. If the new left and right cells are now each in turn subdivided vertically, the result is an embedded region inside the original cell, as shown in Fig. 1. In this manner the connectivity of the generated cells is created from the original single outer boundary cell.

The input CAD geometry is broken up into elements (lines in two dimensions and patches in three dimensions) to allow a full description of a complex geometry. For the geometry elements, a binary tree is constructed in the opposite sense to the cell tree, i.e., from the final level upwards. Each geometry element is enclosed by a box covering the extent of that element. Adjacent boxes are then paired together and enclosed by new boxes and so on, until

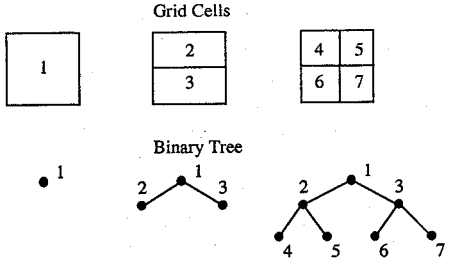


**Fig. 1 Construction of the grid cell binary tree.**

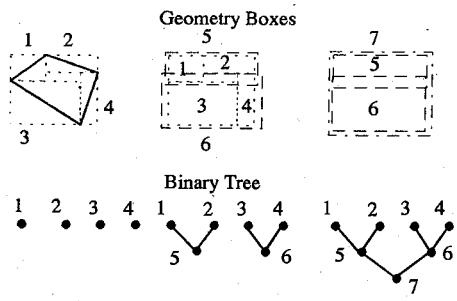


**Fig. 2 Construction of the geometry box binary tree.**

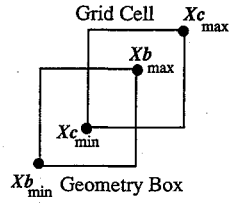


**Fig. 3 Intersection test.**

finally one box (the root) covers the whole extent of the geometry, as shown in Fig. 2. This then provides the connectivity between geometry elements for an efficient searching algorithm covered in the next section.

### Intersection of a Given Cell with a Geometry Box

A major feature of the present scheme is the detection of a geometry element intersection with a given grid cell. One way to test for an intersection is to test every geometry element with the cell, but this is clearly an inefficient method. A more efficient approach, suggested by Bonet and Peraire,[9] which makes use of the geometry box binary tree, is outlined as follows:

1) Test the root geometry box and subsequent boxes for an intersection with the grid cell.

2) Move down the geometry tree if test 1 is positive, and ignore a branch if test 1 is negative.

3) If a geometry box has no children, then the geometry element level of the tree has been reached and the cell is tested for an intersection with the subsequent geometry element.

4) If test 3 is positive the cell is targeted for subdivision and the routine ends.

5) If test 3 is negative return to test 1, until all of the local geometry boxes have been tested.

The method for the detection of an intersection between a grid cell and a geometry box is based on the diagonal vertices of each, as shown in Fig. 3. The procedure requires the placement of a hypercube around the box and the cell, which is then scaled to give 0, 1 as the two extent vertices. The criteria to be satisfied for an intersection are

$$0 \le Xc_{\min} \le Xb_{\max}$$
$$Xb_{\min} \le Xc_{\max} \le 1 \qquad (1)$$

along with the equivalent criteria for the $Y$ (and $Z$ in three dimensions) coordinate direction. A new list of cells to be tested for intersections is generated by keeping a record of all of the new cells formed, at the current level, due to the intersection test.

### Nearest Neighbor Constraint

To prevent unacceptably large cells in the field, a nearest neighbor constraint has been imposed on cells targeted for subdivision. The adjacent cells must satisfy the following condition: upon subdivision no neighboring cells will be more than one level of subdivision different from the cell under test. To test for this condition a search through the grid cell tree is required. If the condition is not satisfied, the illegal cell is also targeted for subdivision. These extra cells targeted for subdivision will, in turn, have their nearest neighbors checked. The test continues until no more illegal cells are detected.

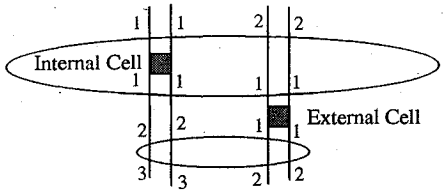


**Fig. 4 Internal cell detection.**

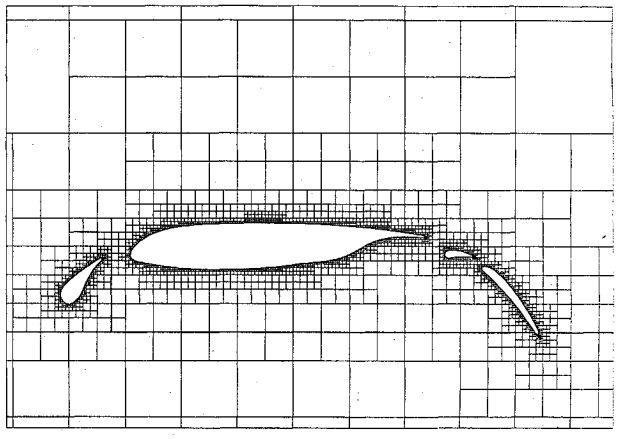

**Fig. 5 Cartesian hanging node grid.**

#### Detection of Cells Inside the Geometry

Cells that are completely inside a closed geometry are not relevant to the external flow solution and are, therefore, discarded at this stage. To determine whether a cell is inside the geometry, a method of projecting vertical lines from the cell vertices has been adopted. The projections are tested for intersections with any geometry elements, and for each positive test a counter for the projection is incremented. The cell is then internal if each projection cuts the geometry an odd number of times, as shown in Fig. 4.

## Application to a Flow Solver

#### Hanging Node Treatment

The final grid of Cartesian cells is characterized by hanging nodes, as shown in Fig. 5. A hanging node occurs at the interface between cells of different levels. One way of dealing with hanging nodes is to treat each hanging node as a special case[10] within the flow solver. An alternative method is adopted in the present work, which employs the dual grid, constructed by connecting the cell centers of all of the Cartesian cells surrounding each vertex. This then produces a grid without hanging nodes, as shown in Fig. 6. Note that the resulting computational cells are a mixture of quadrilaterals and triangles, but this requires no special treatment within the flow solver, as a triangle is a degenerate form of a quadrilateral; however, efficiency is compromised by carrying out a calculation for a collapsed edge.

#### Surface Alignment

The procedure outlined in the preceding section produces a computational grid that is unaligned with the actual surface geometry but is aligned to an approximate representation of this geometry. The original idea was to use this approximation and apply the exact surface flow boundary conditions to it. Upon further investigation, this approach was found to be unacceptable, even though converged solutions were possible. However, by aligning the surface representation with the actual geometry surface, thus producing a body conforming grid, acceptable results are possible. The alignment is achieved by pulling vertices of surface cells onto the actual geometry surface.

#### Smoothing

A further enhancement to the basic grid-generation method is the addition of a grid smoothing routine, based on a Laplacian operator. This is beneficial in reducing excessive wiggles in the final converged surface pressure distribution and also improves the convergence rate. A final smoothed dual grid is shown in Fig. 7, which can be compared with the original unsmoothed grid of Fig. 6.

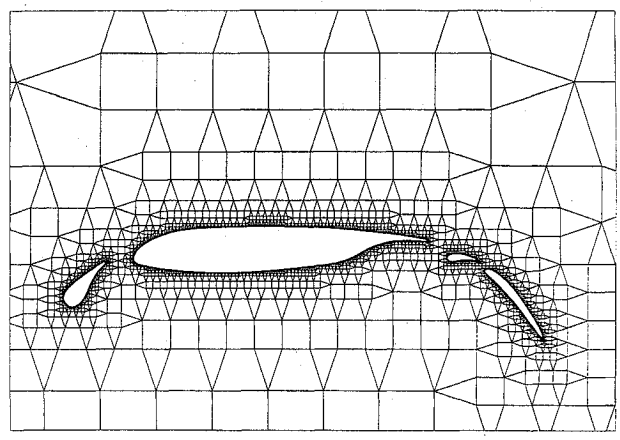

**Fig. 6 Dual grid.**

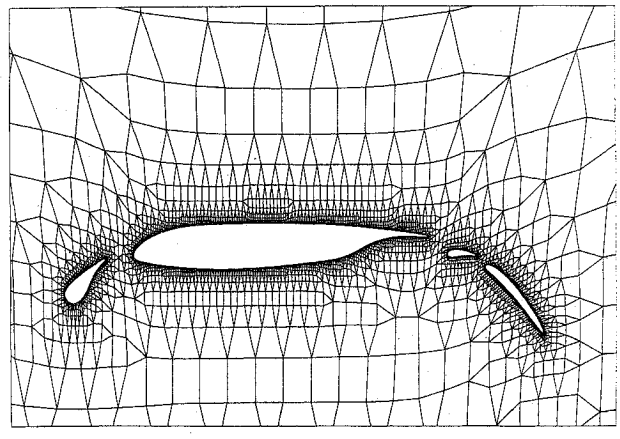

**Fig. 7 Smoothed dual grid.**

## Flow Solver

To demonstrate the feasibility of the present approach, initial computations have been carried out using the two-dimensional Euler equations. The governing inviscid flow equations are solved in time-dependent, integral form. The basis of the solver is a Jameson type, finite volume spatial discretization[11] with multistage time advancement to a steady-state solution. Recognizing that the centered finite volume formulation is nondissipative, artificial dissipation is explicitly added to stabilize the solution procedure and to ensure clean shock wave capture. Local time stepping, enthalpy damping, and an implicit residual smoothing technique are used for convergence acceleration. The implementation of the flow solver follows closely the work of Stolcis and Johnston.[12] The flow algorithm is unstructured in nature and is based on a dual cell pointer scheme.

## Results

The results show an evaluation of the method for a series of successively refined grids around a NACA 0012 aerofoil. Subsequently, two examples of complex configurations are considered. All of the grids generated for the examples by the present method correspond to the fine grid definition covered in the next section.

#### NACA 0012 Aerofoil

An initial evaluation of the present method was carried out using the NACA 0012 aerofoil combined with a grid refinement exercise. Four levels of embedded grid were considered corresponding to a coarse (1142 cells), medium (2041 cells), fine (3808 cells), and very fine density (7219 cells), with each successive grid being a level of subdivision beyond the previous. As an example of the grid generated during this study, Fig. 8 shows the fine grid used for the aerofoil.

The efficiency of the grid-generation method was assessed by plotting the computational time taken to generate the grids (using a 486 DX2 personal computer) against the number of cells, as shown in Fig. 9. This shows that the computational time is linearly proportional to the number of cells for the larger grid sizes, with a small

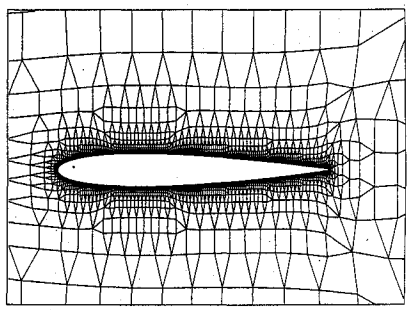

Fig. 8 NACA 0012 fine grid.

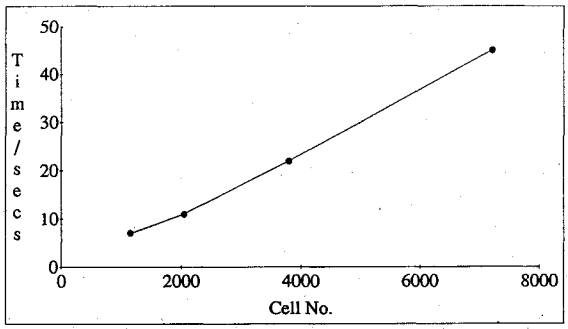


Fig. 9 Graph of grid-generation time against cell number.

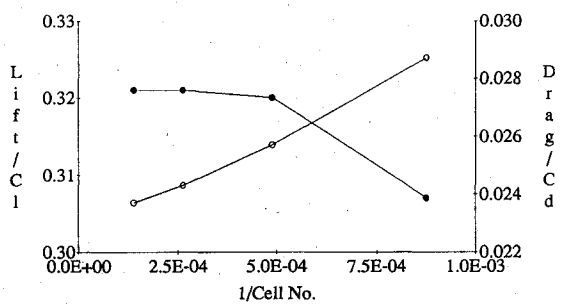


Fig. 10 Graph of lift (•) and drag (○) coefficients against 1/cell number.

deviation for the smallest grid size. The linearity is a good indicator that the method is efficient and predictable for grid cell generation.

The test condition chosen was for a Mach number of 0.8 and an angle of incidence of 1.25 deg, with the comparison focusing on the lift and drag coefficients relative to 1/cell number as shown in Fig. 10. It can be seen that the lift coefficient converges to a value of 0.321 with successive grid refinements. The final lift coefficient, however, does not appear to be in good agreement with independent results[13] for this case, which allow a range of $0.346 < C_L < 0.373$. The discrepancy is mainly attributed to the present method imposing a blunt trailing edge on the geometry, whereas the test cases had sharp trailing edges; also it must be borne in mind that the very fine grid produced by the current method is still relatively coarse away from the surface. Figure 10 also shows the drag converging satisfactorily with grid refinement, and the final value of 0.0237 for the very fine grid is within the bounds of $0.0221 < C_D < 0.0244$ quoted by independent results.[13]

### SKF 1.1 Multielement Aerofoil

The first example, chosen to demonstrate the complex geometry capabilities of the present method, was the SKF 1.1 aerofoil/flap configuration. A transonic flow condition with a Mach number of 0.65 and an angle of incidence of 0.02 deg, corresponding to experimental run 229 of Stanewsky and Thibert,[14] is presented. A view of the grid is shown in Fig. 11, with an expanded view of the grid in the flap cove region shown in Fig. 12. The grid consisted of 894 surface cells and a total number of 4218 cells. Figure 13 shows the surface pressure distribution compared with the experimental data. Good agreement was achieved considering the lack of viscous modeling in the current flow solver. The suction peak on the upper surface of the flap appears to be overpredicted, but this can most

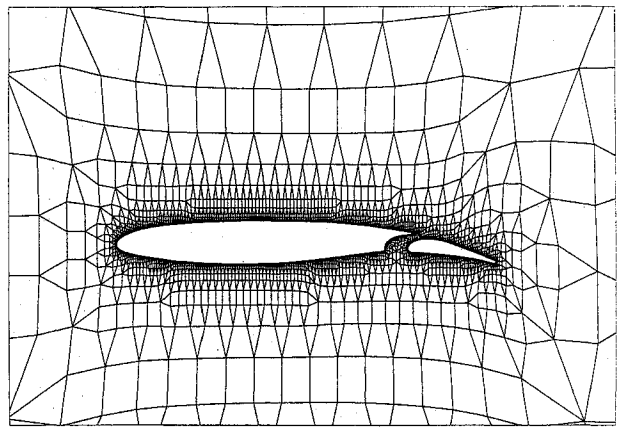

Fig. 11 SKF 1.1 grid.

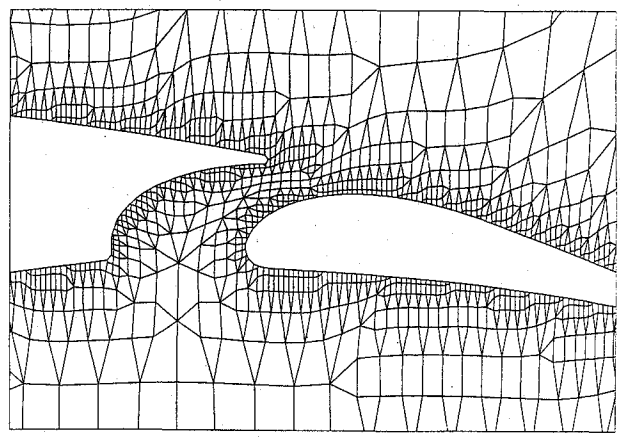

Fig. 12 Grid in the cove region.

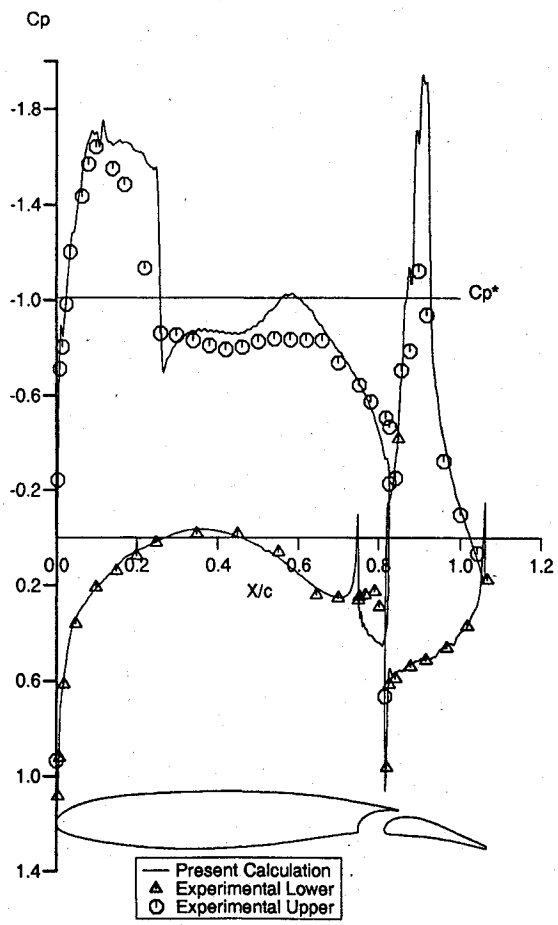


Fig. 13 SKF 1.1 surface pressure distribution.

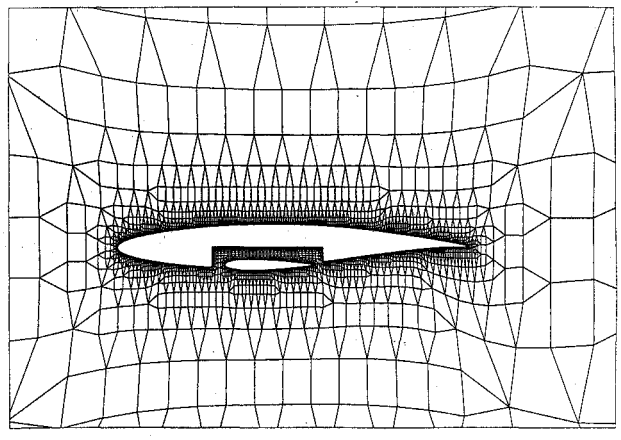

**Fig. 14 Grid for cavity and store position 1.**

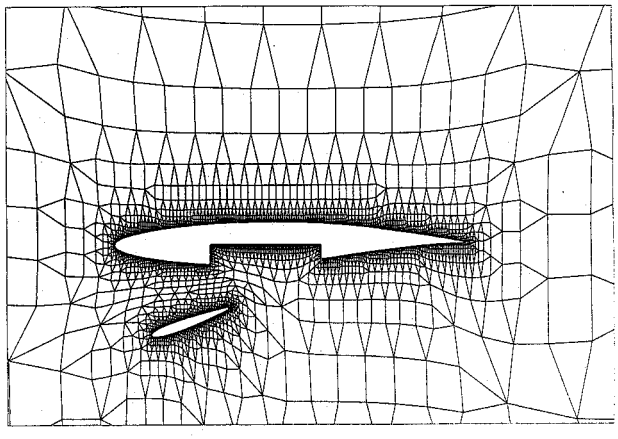

**Fig. 15 Grid for cavity and store position 2.**

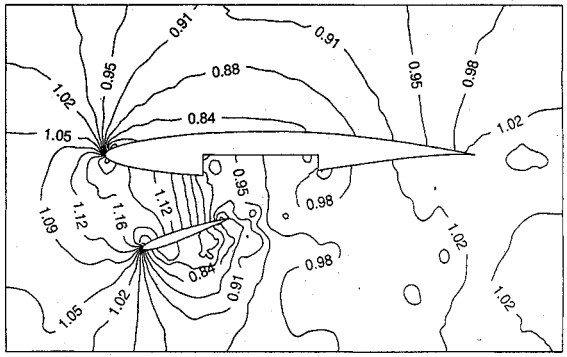


**Fig. 16 Pressure contours for cavity and store position 2.**

likely be attributed to the lack of viscous modeling, especially in the upstream flap cove region.

#### Descending Store

The next demonstration illustrates the ability of the present method to deal with a moving store, descending away from an RAE2822 aerofoil with a cavity. Two positions were chosen, and a steady calculation was carried out at each position, corresponding to a transonic Mach number of 0.6 and an angle of incidence of 0 deg. The remeshing between the two cases took a negligible amount of time, corresponding to 20 s for each on a 486 DX2 personal computer. Each successive grid had approximately the same number of cells as the first store position, that is, −1039 surface cells and a total number of 4576 cells. The grids for each position of the store are shown in Figs. 14 and 15, respectively, and the associated pressure contour for Fig. 15 is shown in Fig. 16. Notice the relative smooth variation of the contours in the field.

## Conclusions

A method for automatic generation of a computational flowfield grid around complex aerodynamic geometries has been presented. The grid-generation time is extremely fast, and there is no user interaction required. A validation of the method dealing with successively refined grids has shown a desired improvement of predicted loads and surface pressure distribution. The application of the method to a complex aerodynamic geometry has shown the results were in good agreement with experiment. Also the potential to deal with a moving geometry has been demonstrated.

There are certain deficiencies of the present method, however, which require further consideration. In its present form the method can not resolve sharp trailing edges, and there are an excessive number of cells required to resolve small geometry features. The ability to allow changes of cell size in relation to geometry features is a natural development that the unstructured nature of the cells will allow. Also within the unstructured framework, flow feature adaptation is an extension that will be pursued. To deal with sharp trailing edges in a general, automatic manner requires further development, and at this stage no optimal method is evident.

Another consideration is how to generate and ensure an efficient placement of high aspect ratio cells around the geometry for viscous calculations. Directional subdivision of cells is an option, but this is limited to features that align with a coordinate direction. Further work on high aspect ratio cells is required if the method is to treat viscous flows.

In summary, the present method has beneficial features but also many areas that require further development, if the ultimate goal of treating three-dimensional complex geometries for viscous flows is to be achieved.

## Acknowledgments

The first author would like to acknowledge the joint support of this work by EPSRC and British Aerospace (Warton), United Kingdom, under the new Engineering Doctorate Programme. Special thanks go to the CFD Group at BAe for their suggestions and encouragement.

## References

[1]Busch, R. J., Jr., "Computation Fluid Dynamics in the Design of the Northrop/McDonnel Douglas YF-23 ATF Prototype," AIAA Paper 91-1627, June 1991.

[2]Heiss, A., Eberle, L., Fornasier, L., and Paul, W., "Application of the Euler Method EUFLEX to a Fighter-Type Airplane Configuration at Transonic Speed," AIAA Paper 92-2620, June 1992.

[3]Weatherill, N. P., and Forsey, C. R., "Grid Generation and Flow Calculations for Complex Aircraft Geometries Using A Multi-Block Scheme," AIAA Paper 84-1665, 1984.

[4]Jameson, A., Baker, T. J., and Weatherill, N. P., "Calculation of Inviscid Transonic Flow Over a Complete Aircraft," AIAA Paper 86-0103, Jan. 1986.

[5]Benek, J. A., Buning, P. G., and Steger, J. L., "A 3-D Chimera Grid Embedding Technique," AIAA Paper 85-1523, July 1985.

[6]Epstein, B., Luntz, A. L., and Nachson, A., "Cartesian Euler Method for Arbitrary Aircraft Configurations," *AIAA Journal*, Vol. 30, No. 3, 1992.

[7]Baehmann, P. L., Wittchen, S. L., Shephard, M. S., Grice, K. R., and Yerry, M. A., "Robust, Geometrically Based, Automatic Two-Dimensional Mesh Generation," *International Journal for Numerical Methods in Engineering*, Vol. 24, 1987, pp.1043–1078.

[8]Smith, R. J., and Johnston, L. J., "A Novel Approach to Engineering Computations for Complex Aerodynamic Flows," *Proceedings of the 4th International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, 1994, pp. 271–285.

[9]Bonet, J., and Peraire, J., "An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems," *International Journal for Numerical Methods in Engineering*, Vol. 31, 1991, pp. 1–7.

[10]Schonfeld, T., "Automatic Local Grid Refinement for Vortical Flow Computations Around Delta Wings," *Proceedings of the 4th International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, 1994, pp. 589–602.

[11]Jameson, A., and Baker, T. J., "Multigrid Solution of the Euler Equations for Aircraft Configurations," AIAA Paper 84-0093, 1984.

[12]Stolcis, L., and Johnston, L. J., "Solution of the Euler Equations on Unstructured Grids for Two-Dimensional Compressible Flow," *Aeronautical Journal*, June–July 1990, pp. 181–195.

[13]Anon., "Test Cases for Inviscid Flow Field Methods," AGARD AR 211, 1985.

[14]Stanewsky, E., and Thibert, J. J., "Airfoil SKF 1.1 With Maneuver Flap," AGARD AR 138, A5-1–A5-29, May 1979.