GRID GENERATION AND SURFACE MODELING FOR CFD

Stuart D. Connell, Janet S. Sober and Scott H. Lamson
GE Research and Development Center
Schenectady, NY 12301

SUMMARY

When computing the flow around complex three dimensional configurations, the generation of the mesh is the most time consuming part of any calculation. With some meshing technologies this can take of the order of a man month or more. The requirement for a number of design iterations coupled with ever decreasing time allocated for design leads to the need for a significant acceleration of this process. Of the two competing approaches, block-structured and unstructured, only the unstructured approach will allow fully automatic mesh generation directly from a CAD model. Using this approach coupled with the techniques described in this paper, it is possible to reduce the mesh generation time from man months to a few hours on a workstation.

The desire to closely couple a CFD code with a design or optimization algorithm requires that the changes to the geometry be performed quickly and in a smooth manner. This need for smoothness necessitates the use of Bezier polynomials in place of the more usual NURBS or cubic splines. A two dimensional Bezier polynomial based design system is described.

1 INTRODUCTION

The aerodynamic design of components is by and large achieved through "design by analysis". An engineer will begin with an approximation to the shape of a part. This is then analyzed utilizing a suitable CFD tool. Based on these results, changes to the geometry are postulated which will improve the part subject to certain constraints. These improvements, for example, may aim to increase the efficiency or decrease the manufacturing cost of the part. Having made the changes, the new design is reanalyzed and the results evaluated. This process may be repeated a large number of times during the design process. In general a limited amount of time and money are allocated for the design of a given component, hence the CFD code must allow a number of design iterations within these restrictions. To achieve this geometrical approximations are made and certain features are ignored because it is too costly to include them in the simulation. For some complex three dimensional configurations where geometrical simplifications cannot be made the use of a wind tunnel proves to be more cost effective.

With many currently available CFD technologies there is little room for improvement in component design within the constraints of time and money. Further improvements can only be achieved by including more geometric fidelity, reducing turnaround times and improving the physical models within the simulation. In this paper methods for improving the former two items will be investigated.

Many impressive calculations have been performed on complex three dimensional geometries employing block-structured techniques. However turnaround time makes these calculations impractical for routine design work. In some cases it may be possible to use these techniques in a design environment if mesh generation software is tailored for one particular topology [1]. This approach however requires a significant investment of time and money to develop such software which makes it too costly for a large number of problems. In addition

any design change that results in a change in topology would require much rewriting of the mesh generation software.

The use of an unstructured mesh presents a possible solution. Here mesh generation can be made automatic for arbitrary geometries thereby significantly reducing turnaround times. However the geometry specification (also common to the structured approach) and the generation of a mesh suitable for a viscous calculation still represent formidable tasks. These areas are addressed in detail in sections 2, 3 and 4.

Of the available unstructured mesh generators there is no accepted "best" method. It is possible using any of these methods to generate a mesh for complex configurations. The relative merits of the competing unstructured and structured approaches together with examples are presented in Section 2.

Within industry there is a strong trend towards representing all geometries by one single CAD model which is used and modified by a variety of disciplines during the design process. The model is ultimately used to manufacture the part. This approach has many advantages, for example, the elimination of the need for conversion programs, and their associated errors, to move between one geometry format and another. With powerful CAD packages available there seems little point in developing "in-house" software to represent and manipulate geometry. The obvious next step is to use these CAD models as the basis for a CFD calculation. Any geometry modification required by the design process can be efficiently performed using the CAD package. In section 3 the approach used to produce a computational mesh directly from a CAD model is described.

The desire to perform viscous calculations adds further constraints to the mesh generation algorithm. For these calculations high aspect ratio cells aligned with the flow gradient are required in the boundary layer and wake regions. Existing unstructured mesh generation algorithms aim to produce tetrahedra close to equilateral, or at best provide a limited ability to stretch elements. The use of equilateral cells in viscous regions would result in a prohibitively expensive algorithm. In addition there will ultimately be a requirement to adapt the mesh in viscous regions. To be efficient this adaption must be performed in an anisotropic manner, i.e. increasing the resolution normal to the wall while maintaining the streamwise spacing. The generation of meshes meeting the above requirements is the subject of much current research. An "inflation" technique to efficiently produce a near wall prismatic mesh about complex geometries is described in Section 4.

Unstructured meshes lend themselves well to adaption by reinforcement where new nodes are added into an existing mesh. In addition it is possible to use the refinement levels as the basis of a multigrid algorithm. Such a scheme is described and compared to other multigrid schemes in section 5.

The exact mathematical formulation of the surfaces used in the CAD model of the component to be meshed has been ignored and left to the CAD program. When it is desired to automate a CFD based design system the mathematical form of these surfaces becomes important. The use of Bezier polynomials as a basis for such a system is described in section 6.

Finally in section 7 recommendations are made for an ideal CFD system. These recommendations include the use of available technologies and the development of new ones.

2 MESH GENERATION

One of the aims when developing mesh generation software is to minimize the cost, both machine and human, of generating a mesh. As a designer's time is more valuable than CPU cycles this inevitably means minimising the amount of user interaction required to produce a mesh. In two dimensions mesh generation is essentially a solved problem. There are a variety of structured and unstructured algorithms which will produce a high quality mesh around complex geometries. For example the unstructured method described in [2] requires a few minutes of CPU time on a workstation and just four user inputs to generate a mesh: the file name for the geometry, the maximum element size, the minimum element size and a boundary curvature sensitivity. The procedure begins by placing points on the boundary to reflect surface curvature. These points are triangulated using a Delaunay

procedure. New nodes are now repeatedly inserted into the mesh to remove poor quality elements. The process is terminated when all elements satisfy the required quality measure. Figure 1 shows a mesh generated by this approach.

In three dimensions automatic mesh generation for complex geometries represents a formidable, though not intractable task. The block structured approach requires the specification of an initial blocking topology prior to generating the mesh in the individual blocks. The specification of the initial blocking topology is a global operation requiring knowledge of all the boundary surfaces. For the human brain this is a trivial task in two dimensions requiring a few mouse clicks in a suitable user interface to specify the block topology. The problem is significantly more difficult in three dimensions. Progress has been made in reducing the level of user interaction [3,4] to some degree, but due to the need for complex global operations it is unlikely that this will ever be fully automated.

In contrast the unstructured approach can be made automatic for arbitrary geometries. With these approaches there is no need for complex global operations. All operations are simple and local, requiring only local information. This fact alone makes it possible to automate the procedure.

There are three basic techniques used for automatic unstructured mesh generation:

(1) The octree approach [5,6] is based on successive subdivision of the domain to produce a Cartesian mesh. At boundaries many compute-intensive line surface intersection operations are required. This results in a high quality mesh in the interior with the worst mesh at boundaries. Smoothing operations are then used in an attempt to improve the near-wall mesh. Octree approaches do not lend themselves well to the generation of high aspect ratio cells needed for viscous simulations which will be described in Section 4.

(2) The Advancing front algorithm[7,8] is a two step procedure. First each surface is triangulated. This is done by placing nodes around the boundary of each surface in the model. These nodes are connected by edges to form an initial two dimensional front, this front is then advanced into the surface by building triangles on each edge of the front. In the second stage the triangulated surfaces form an initial front for the volume mesh. Using a similar algorithm to that used for the surfaces the domain is filled by recursively building tetrahedra on each face of the front.

(3) Delaunay based methods as described in [9,10,11] are the natural extension of the two dimensional algorithm described in [2] into three dimensions. As with the advancing front algorithm a surface triangulation is first generated and a constrained tetrahedralization is formed of these boundary triangles. The tetrahedral mesh is now generated by repeatedly inserting points into this mesh.

Of the above algorithms the advancing front and Delaunay methods have been found to produce smoother and more regular meshes than the somewhat irregular meshes typically produced by octree approaches. A comparison between the surface meshes produced by the advancing front and octree algorithms for a nacelle is shown in figure 2. In addition the formation of an initial surface mesh provides a natural framework for the viscous mesh generation procedure described in Section 4. Delaunay based methods have also been found to be significantly faster than the advancing front and octree methods.

The current drawback of the advancing front and Delaunay methods is the need to provide a background mesh. This background mesh is required to specify the mesh density throughout the flow domain. The automatic generation of this file, using for example surface curvature to drive the placement of source terms, would further accelerate the mesh generation process. An alternative approach suitable for the Delaunay algorithms is to abandon the the background mesh altogether. The surface mesh can be generated as in 2D [2] with the additional constraint that points be inserted into the mesh to resolve surface curvature. The 3D tetrahedral mesh can then be generated in a similar manner again by inserting points based on element quality.

Another point to note here is that much of the speed and ease of use of an automatic unstructured mesh generator can quickly be destroyed by the addition of a graphical user interface (GUI). These GUIs are of little or no use to a designer who needs to repeatedly mesh and solve on a series of similar geometries. Having to

repeatedly enter similar data is error prone and significantly hinders the mesh generation process. Much of the effort expended in writing such interfaces would be better utilized in further automation of the mesh generation procedure. Ideally any graphics in a mesh generator should be used for viewing the mesh and not for interactively generating the mesh.

3 LINK TO CAD SYSTEM

Computer Aided Design (CAD) systems can represent a component in three forms: solid, surface and wireframe models. A solid model contains information at two levels, topology and geometry. The topological entities are vertex, edge and face. Each face is surrounded by a number of edges, and a vertex lies at the ends of each edge. For example a solid model of a cube would comprise six faces, 12 edges and 8 vertices. The underlying geometrical entities used to define the actual shape of the object are referred to as point, curve and surface. These can be of various types within the model, for example, the geometry for a surface could be based on Non-Uniform Rational b-splines (NURBS) or Bezier polynomials. A surface model merely contains geometrical information for the individual surfaces. All topological information on how the surfaces fit together is lost. In addition there is no requirement that the bounding surfaces be closed. Wireframe models are a further simplification which only contain information on the outlines of the surfaces.

The use of a solid model to represent components is becoming more and more prevalent throughout industry, and are utilized by all disciplines from design to manufacturing. The models contain enough information to manufacture the part and hence provide a logical starting point for mesh generation. Generally with these models the solid part is the body, i.e. the metal, whereas to be of use for a CFD calculation the model needs to be inverted so the solid is the gas path around the metal. Fortunately this a relatively straightforward procedure within a CAD system.

An alternative approach is to base the mesh generation on a common surface model format such as IGES (or NIGES). This approach initially appears attractive as CAD systems can output IGES files, hence if a mesh generator can read IGES files it can work with all CAD systems. However with the surface model the topological information required to automatically build the mesh is lost and must be input by the user during the mesh generation process. Hence mesh generation cannot be made automatic and much of the advantage and elegance of unstructured mesh generation is immediately destroyed. In addition this approach necessitates the development and support of computer codes employing complex Graphical User Interfaces (GUI) to aid in the input of this information.

The strategy being developed at GE CRD is to generate an unstructured mesh by interrogating the solid model directly via the vendor supplied subroutine library in order to obtain information necessary to build the mesh. This is done as the subroutine package and solid model form a self consistent entity. Converting to some intermediate format such as some as yet undefined extension to IGES opens up many issues, for example tolerancing; while the model may be a perfectly valid solid model in one CAD package, it may fail some tests on the common file format. Coupling unstructured mesh generation with the native read of the solid model allows the mesh to be built automatically from the solid model with no further intervention from the user. At GE, solid models from the PARASOLID [12] based UNIGRAPHICS CAD system are used.

Unstructured mesh generators typically require a limited number of low level geometry and topology interrogation operations in order to build the mesh. It would be extremely useful if all the CAD vendors agreed on a standard in order to extract this information. Calls to these standard routines could then be made from the grid generation code thus allowing meshes to be generated from solid models from a variety of vendors.

As a standard set of geometry interrogation routines is not currently available an approach in a similar vein has been adopted. A set of low level interrogation routines to be used by the mesh generation package has been defined. Within these subroutines are the calls to the vendor specific routines which provide the required information. Examples of the sort of topological and geometric functionality needed are:

1) The total number of faces, edges and vertices
2) The (x,y,z) coordinates from face id number and parametric coordinates (u,v)
3) The id number of edges surrounding a face
4) The nearest point on a face to a given (x,y,z) location

Figure 3 shows the surface mesh for a three dimensional geometry generated directly from a solid model in 10 minutes on an HP735 workstation using a modified version of the advancing front program of [7].

4 VISCOUS MESH GENERATION

The requirements for a viscous mesh differ significantly from an inviscid mesh as it is no longer sufficient to fill the domain with equilateral tetrahedra. For efficient simulations high aspect ratio elements must be generated and aligned with viscous gradients. A recent paper [13] demonstrated such a capability in two dimensions. Here a layer of structured quadrilateral cells was wrapped around the body and extended into wake regions. The interior was then filled with triangular cells. An example of such a mesh is shown in figure 4. The quadrilateral mesh gives very efficient resolution of boundary layers and wakes. In addition it permits the directional refinement necessary in viscous regions. This is illustrated in figure 5. A natural extension of this approach to three dimensions is to first place structured layers of triangular based prismatic cells on an existing surface triangulation. This process can be thought of as "inflating" the surface triangulation. The interior is then filled with a tetrahedral mesh. The prismatic cells can have a high aspect ratio which will permit efficient resolution of the boundary layers. They also possess quadrilateral faces which will ultimately permit the directional refinement illustrated in figure 5.

There are a variety of methods currently being developed to produce near-wall prismatic meshes [14,15,16,17,18]. All the methods begin with a surface triangulation which is then marched or inflated towards the interior in a series of steps.

In [15] this marching is achieved through representing the initial surface triangulation by a number of non-intersecting hexahedral elements (voxels). The triangulation is contained within these voxels. The outer surface of these voxels is then smoothed to form the first inflated surface. Computing the intersection of normals from the original surface with the inflated surface forms the first prismatic layer. The process is then repeated to form the complete prismatic mesh. This method has the disadvantage (also common to Octree based methods) that the mesh will change if the geometry is rotated with respect to the coordinate system.

The unstructured hyperbolic mesh generation technique [16] was also investigated. This method was found to be prone to crossovers at sharp internal and external corners unless many explicit steps of the algorithm were taken. For some relatively simple test cases even with many steps a valid mesh remained impossible to obtain. With this method there is also little control over the spacing away from the wall. The specification of this spacing is crucial for turbulent calculations.

The advancing layers method [17,18] inflates the surface along quasi-normal directions with a modified advancing front type algorithm. When certain geometrical criteria are satisfied, such as distance from a wall or element quality, the algorithm reverts to the standard method. Numerical experiments indicate that this method can be made less prone to crossovers at sharp corners than the hyperbolic technique. It also has the advantage that near-wall spacing may be specified directly for turbulent flows.

The algorithm described in [14] also uses quasi-normal directions as a starting point for generating the prismatic mesh. New nodes are placed along the normal directions to form a structured mesh of prismatic elements which wrap around the viscous surfaces. The volume mesh generator uses this inflated triangulation as its initial front rather than the initial triangulation. This algorithm will now be described in detail.

To describe the combined algorithm of surface inflation and advancing front, a rectangular box with a cylindrical hole will be used as an example. This geometry is illustrated in figure 6. The cylindrical surface and lower wall are marked as viscous and are to be inflated. All other boundaries are either inlet surfaces, exit surfaces or inviscid walls and will not be inflated.

The procedure begins by triangulating the viscous walls using the surface meshing part of the advancing front algorithm. This surface mesh is shown in figure 7(a). The next stage in the procedure is to compute a quasi-normal direction at each node on the surface triangulation. The algorithm used to compute these normal directions is described in [14]. The initial surface is now inflated a specified distance δ along these quasi-normal directions. This gives the first layer of prismatic cells. This inflation process is repeated a number of times using different values of δ but the same normal direction. Typically δ varies between one layer and the next by a fixed geometrical factor ranging between 1.0 for a uniform mesh and 5.0 for a highly stretched mesh. The resultant prismatic mesh is shown in figure 7(b). The total thickness of this prismatic region is adjusted to encompass the expected boundary layer.

The remaining non-viscous surfaces are now triangulated. Any new nodes and edges generated by the inflation algorithm which lie on these surfaces are required to form part of the initial front for these surfaces. The newly triangulated surfaces and the inflated triangulation are combined to form a closed front. This becomes the initial front for the interior mesh generation algorithm. The resultant combined mesh is shown in figure 7(c).

For many cases this scheme is found to work well. However for some complex geometries a suitable algorithm to produce a set of normals, which in turn lead to a valid prismatic mesh of acceptable quality remains elusive. In addition, for boundaries which are close to one another, it may be possible for the prismatic meshes to overlap. To produce a robust mesh generator a fallback position is adopted. After the prismatic mesh is generated a number of checks for cell quality are made. These checks ensure that:

1) Each prismatic cell has positive volume,
2) No cell intersects any other and
3) Cells are of reasonable quality.

Any cell failing these tests is tagged for deletion from the mesh.

When cells are removed triangular and quadrilateral faces of elements below the inflated surface are exposed. To form a front the quadrilateral faces are divided into triangles. It is possible for these triangular faces to posess a high aspect ratio, especially near the wall. A front containing such faces has proven problematical to the volume mesh generator. The algorithm for removing these high aspect ratio faces is described in [14].

Meshes generated by this algorithm are shown in figure 8 for a turbine blade and figure 9 for a wing/pylon/nacelle. The mesh is illustrated by making various cuts through the prismatic region. As can be seen the mesh is of high quality and provides a good starting point for a viscous calculation.

Meshes generated by the above approach will contain a variety of element types. While it is possible to divide these elements and produce a purely tetrahedral mesh this is not the most efficient method. Modifying the flow solver to work directly on a mixed mesh is a better option.

5 ADAPTION AND MULTIGRID

Adaption forms an integral part of any unstructured CFD calculation, since using an unstructured mesh without adaption utilizes only half the power of the method. The use of the various meshes generated by an adaptive procedure in a multigrid algorithm is an obvious step and has been demonstrated in [19,20,21,22]. There are two basic methods used for adaption: reinforcement, where new nodes are added into an existing mesh, and remeshing where the entire geometry is remeshed.

Multigrid schemes based on adaptive remeshing have three disadvantages. The first is that the mesh generation algorithm needs to be robust and fast. If the scheme is not fast, the mesh generation time can quickly exceed the solution time and a lot of the benefit of multigrid is lost. The second disadvantage relates to the difficulty in computing transfer operators for the unstructured mesh. As the meshes bear no relation to each other this is a non trivial task. Complex data structures are required to avoid an $O(n^2)$ search. Finally, unless the meshes are generated through solution adaption then a large percentage of the nodes on the finer meshes may serve no useful purpose.

The multigrid scheme presented in [19] overcomes the above disadvantages. The process begins with an initial coarse grid. An initial solution is obtained on this grid. This solution is then examined to determine if the grid is sufficiently fine to resolve features of interest. In regions where more resolution is required the mesh is enhanced through reinforcement. A solution is obtained on this new grid. The solve and refine process is repeated until some desired level of accuracy is achieved. When using this approach the multigrid levels are formed naturally by the refinement procedure. The construction of the transfer operators is a simple task requiring a single pass through the data. Full details of the multigrid scheme are presented in [19].

The reinforcement procedure can be significantly faster than remeshing. Typically, the reinforcement procedure generates tetrahedra at the rate of 25,000 per cpu second on an HP735 workstation. In contrast the advancing front algorithm of [7] produces tetrahedra at the rate of 66 per cpu second. The newer Delaunay based meshing algorithms [9] are significantly faster than advancing front though still slower than reinforcement.

The initial grid for the reinforced procedure meshes has to be fine enough to resolve all the features of the geometry. If this is not so, it is possible for the node snapping procedure to produce a crossed over mesh, as illustrated in figure 10. This restriction prevents the full benefit of multigrid being realized on complex geometries as a relatively fine initial mesh has to be used. A possible solution would be to combine multigrid by agglomeration [23] with the reinforcement based multigrid. Coarse multigrid levels could then be generated from the initial mesh by agglomeration and finer levels by reinforcement.

6 SURFACE MODELING

More complex geometric models of internal and external systems require a greater degree of rigor in geometric definition. This requirement together with pressure for standardization and cost effective engineering are driving design of aerodynamic devices to be either based on standard CAD systems or to produce geometric models compatible with those packages. By basing the analysis on generic CAD systems, it becomes feasible to analyze aerodynamic devices with increased detail included, such as wings with flaps deployed and nacelle/pylons attached instead of analyzing an idealized wing. Similarly, turbine blade models may include shrouds, cooling holes, and endwall gaps previously ignored. With increasing focus on system level integration and optimization, all engineering disciplines require access to a consistent geometric model, often refered to as a master model.

The geometric model for aerodynamic devices should provide good support not merely for representing the final geometry but for the process of reaching that design. Aerodynamic surfaces are frequently free form sculptured surfaces. While low order polynomial (cubic) splines (or cubic NURBS in 3D) are quite suitable to represent geometry of this type, they provide poor support for the design process [24,25,26,27]. As a designer (either human or an optimizer) modifies a single cubic spline control point to find a better design, the surface curvature develops large oscillations undesirable to the fluid flow (see figure 11). It has been found that using high order Bezier curves instead of cubic splines has a much more desirable response to control point manipulation, as shown in figure 12. This technology has been developed into a quasi-3D turbine blade design tool which finds application in both interactive design of blades and automated design where it is driven by an optimization system [28].

For the master model concept and automated grid generation to deliver the required benefits in streamlining the engineering process, the geometric processing required must proceed smoothly and reliably. Unfortunately, the current state of commercial solid modelling software has not yet attained this level of robustness. Automation and speed of the design process are dependent on the correct reliable performance of solid modelling software.

Particularly in this age of virtual corporations, outsourcing, and strategic alliances, it cannot be expected that a single CAD system will be selected as the universal supplier of geometry. While it is possible to set up a standardized interface to geometry packages that grid generation software could use, as described in section 3, experience in two-dimensions indicates that once you go beyond static CFD solutions and include analytic sensitivites, design optimization or inverse design, it is required that the CFD tools have an explicit geometric capability built in to them. It is quite possible to have a generic geometric interface for static geometry, but it is not so clear that the design and sensitivity calculation capability in the CFD tools could get required geometric operations from a general standardized interface to a variety of solid modellers.

This geometry coupling is the foundation of an innovative approach to interactive turbine blade design using linearized Euler sensitivities developed in [29]. An Adaptive unstructured Euler equation solver produces a non-linear mean flow solution using the Bezier curve geometry as produced by the design tool. Additionally, the flow solver can compute linearized steady perturbation solutions for arbitrary geometry deformations of the blade shape. The perturbations specify the geometric sensitivity coefficients for each flow variable, such as $\partial P/\partial x_i$ for the pressure derivitive with respect to the i-th Bezier control point. Since the design tool and the flow solver both use the Bezier control point geometry to represent the turbine blade profile, it is possible for the flow solver to compute the geometric sensitivity for displacement of any control point. By providing this data to the design tool, as the designer modifies the blade by interactively moving Bezier control points, the blade surface Mach number distribution can be updated in real time using a locally linearized approximation. Given the baseline pressure (P_0) from the non-linear meanflow solution, the pressure P after altering the i-th control point by a displacement dx_i can be computed from

$$P = P_0 + \sum_{i=1}^{n} \frac{\partial P}{\partial x_i} dx_i$$

to within the locally linearized approximation. For blade changes beyond some range, the linear approximation error increases and the meanflow solution plus a new set of sensitivities must be computed. The same geometric sensitivites can also be used to accelerate blade design optimization using a generic engineering optimization package such [24,28].

7 RECOMMENDATIONS

The recommendations made here are intended to define the "ideal" CFD system. Many of the technologies either exist or are under active development. The CFD system should be modular so that if new technology becomes available an existing module can easily be replaced. The modules should be able to communicate with each other via either a common file format, or a particular format and a subroutine library to extract data from the file. There are five basic modules: Geometry definition, mesh generation, flow solution, adaption and post-processing.

7.1 Geometry Definition

The best approach here is to use a solid model from a CAD program to define the geometry of the component to be meshed, then for the reasons outlined in section 3 the grid generation program should read this model directly. To achieve this for a number of CAD packages a standard set of subroutine calls to interrogate a solid model will need to be defined, each CAD vendor will then need to provide an interface using these calls.

7.2 Mesh Generation

For the reasons stated in section 2 mesh generation should be based on unstructured technology. The following proposed algorithm is intended to be as automatic and as efficient as possible. The method begins by triangulating the surface of the body with the mesh density based on surface curvature. This removes the need for any additional input through for example a background mesh file. For efficiency the ability to generate stretched elements aligned with, for example leading edges would be useful. A viscous mesh can then be generated by inflating this surface triangulation. Finally the interior should be tetrahedralized with a Delaunay based algorithm where points are inserted based on mesh quality rather than a background mesh. The majority of components are in place for such an algorithm, the only real need is the removal of the background mesh.

7.3 Flow solver

To converge in a reasonable time any flow solver needs to employ a multigrid or implicit solution algorithm. An attractive and efficient way to generate the multigrid levels is to agglomerate from the initial mesh to get the coarser levels and adaptively refine through point insertion to generate the finer levels. The two technologies exist to do this but to the authors' knowledge they have not yet been combined.

The use of implicit algorithms on unstructured meshes is the subject of much research. As with their structured mesh counterparts, it is unclear which algorithm is the most robust and computationally efficient.

7.4 Refinement

Refinement is most efficiently done by point insertion: this will always be faster than remeshing. The ability to perform one-dimensional refinement described in section 4 is an essential part of any adaptive scheme. In addition this adaption procedure can be used to generate a one-dimensional multigrid scheme to accelerate the solver in viscous regions.

7.5 Post processing

There are a number of flexible post processors available. At the very least the software must be capable of handling different element types and any mix of them. Visual3 and FIELDVIEW are the only general purpose codes that to the authors' knowledge can handle mixed meshes, though others are migrating in that direction. To be of use to designers any post processor must be able to reduce the large three dimensional data sets to simple x-y plots of interest to an engineer, for example pressure around a wing section. 3D views of solutions using color contour plots, while very nice for publicity purposes, have little use to designers.

8 ACKNOWLEDGEMENTS

The authors would like to acknowledge the General Electric Company for permission to publish this paper. In addition the authors wish to acknowledge Dr. Kim Bey of NASA Langley Research Center for supplying the advancing front mesh generation program originally authored by J. Peraire, K Morgan and J. Peiro. Thanks are due to our colleagues Dr. M. E. Braaten and Dr. D. G. Holmes for many helpful discussions.

9 REFERENCES

1 Khune, C. M., Uenishi, K., Leon, R. M. and Abdol-Hamid, K. S. "CFD based 3D Aero Analysis System for High-Speed Mixer-Ejector Exhaust Nozzles" AIAA Paper 94-2941, 1994.

2 Holmes, D. G. and Snyder D. D., "The Generation of Unstructured Triangular Meshes Using Delaunay Triangulation," Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press, 1988.

3 Dannenhoffer, J. F., "Automatic Topology Generation from Block Structured Grids," VKI Lecture Series January 1994.

4 Dawes, W. N. "The Extension of a Solution Adaptive 3D Navier-Stokes Solver Towards Geometries of Arbitrary Complexity" ASME Paper 92-GT-363, 1992.

5 Shepperd, M. S. and Georges M. K., "Automatic Three Dimensional Mesh Generation by the Finite Octree Technique," International Journal for Numerical Methods in Engineering, 32:709-739, 1991.

6 Graichen, C. M., Hathaway, A. F., Finnegan, P. M., Kela, A. and Schroeder, W. J. "A 3-D Fully Automated Geometry Based Finite Element Meshing System" ASME Paper 89-WA/CIE-4, 1989.

7 Peraire, J., Morgan, K. and Peiro, J. "Unstructured Finite Element Mesh Generation and Adaptive Procedures for CFD", AGARD Specialists Meeting, Loen, Norway, 24-25 May 1989.

8 Löhner, R. "Adaptive Remeshing for Transient Problems With Moving Bodies", AIAA Paper 88-3737, 1988

9 Weatheril, N. P., "Adaptive Inviscid Flow Solution for Aerospace Geometries on Efficiently Generated Unstructured Tetrahedral Meshes," AIAA Paper 93-3390, 1993.

10 Blake, K. R., and Spragle, G. S. "Unstructured 3D Delaunay Mesh Generation Applied to Planes, Trains and Automobiles," AIAA Paper 93-0673, 1993.

11 Barth, T. "Aspects of Unstructured Grids and Finite-Volume Solvers for the Euler and Navier-Stokes Equations," AGARD report 787, March 1992

12 "Overview of PARASOLID, Version 5", EDS Unigraphics, Parasolid Business Unit, Parker's House, 46, Regent Street, Cambridge, UK, 1993.

13 Connell, S. D., Holmes, D. G. and Braaten, M. E., "Adaptive Unstructured 2D Navier-Stokes Solutions on Mixed Quadrilateral/Triangular Meshes" ASME Paper 93 GT-99, 1993

14 Connell, S. D. and Braaten, M. E., "Semi-Structured Mesh Generation for 3D Navier-Stokes Calculations" GE Research and Development Center, Report No. 94CRD154, August 1994.

15 Kallinderis, Y. and Ward, S., "Prismatic Grid Generation with an Efficient Algebraic Method for Aircraft Configurations," AIAA Paper 92-2721, 1992.

16 Melton, J. E., Pandya, S. A. and Steger, J. L., "3D Euler Flow Solutions Using Unstructured Cartesian and Prismatic Grids" AIAA Paper 93-0331, 1993.

17 Löhner, R., "Matching Semistructured and Unstructured Grids for Navier-Stokes Calculations," AIAA Paper 93-3348, 1993.

18 Prizadeh, S., "Unstructured Viscous Grid Generation by Advancing Layers Method," AIAA Paper 93-3453, 1993.

19 Connell, S. D. and Holmes, D. G., "A 3D Unstructured Adaptive Multigrid Scheme for the Euler Equations," AIAA Paper 93-3339, 1993.

20 Holmes, D. G. and Connell, S. D. "Unstructured, Adaptive, Finite Volume Solution Methods for Fluid Dynamics" 7th IMACS International Conference on Computer Methods for Partial Differential Equations (IMACS PDE-7), Rutgers University June 22-24 1992.

21 Mavripilis, D. J. "Three Dimensional Unstructured Multigrid for the Euler Equations", AIAA 10th Computational Fluid Dynamics Conference, AIAA Paper 91-1549-CP, 1991.

22 Braaten, M. E. and Connell, S. D. "A 3-D Unstructured Adaptive Multigrid Scheme for the Navier-Stokes Equations" GE Research & Development Center, Schenectady, NY, Report No. 94CRD146, August 1994.

23 Venkatakrishnan, V. and Mavripilis, D. "Agglomeration Multigrid for the 3D Euler Equations" AIAA Paper 94-0069, 1994.

24 Shelton, M. L., Gregory, B. A., Lamson, S. H., Moses, H.L., Doughty, R. L. and Kiss, T. "Optimization of a Transonic Turbine Airfoil Using Artificial Intelligence, CFD, and Cascade Testing", presented at ASME 1993 Gas Turbine Conference, Cinncinnati Ohio, June 1993.

25 Lamson, S. H. "Bezier Curves, Aerodynamics, and Optimization", presented at SIAM National Meeting, San Diego, July 26, 1994.

26 Korakianitis, T. and Papagiannidis, P. "Surface-Curvature-Distribution Effects on Turbine-Cascade Performance", Journal of Turbomachinery, Vol 115, 334-341, 1993.

27 Burgreen, G. W., Baysal, O. and Eleshaky, M. E. "Improving the Efficiency of Aerodynamic Shape Optimization", AIAA Journal, Vol 32 No. 1, 69-76, 1994.

28 Ashley, S. "ENGINEOUS Explores the Design Space", Mechanical Engineering, Feb. 1992, Vol. 114, Number 2, p. 49-52.

29 Homes, D. G. Private communication, 1993.

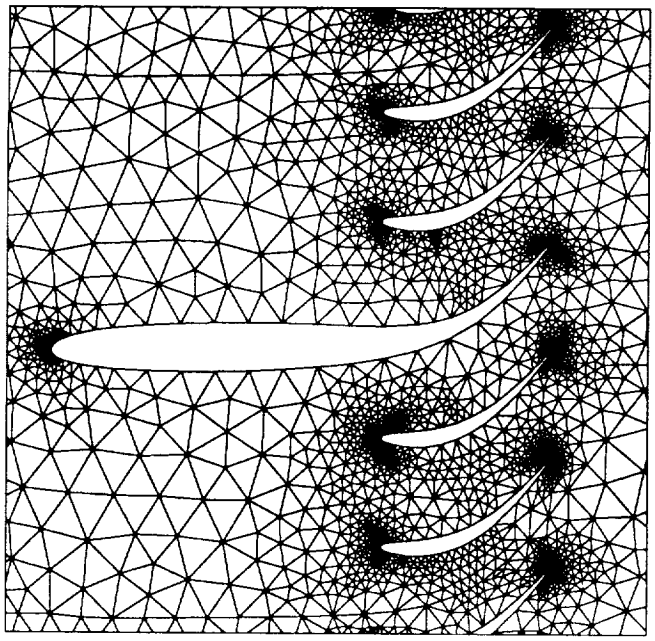


Figure 1 Example inviscid mesh

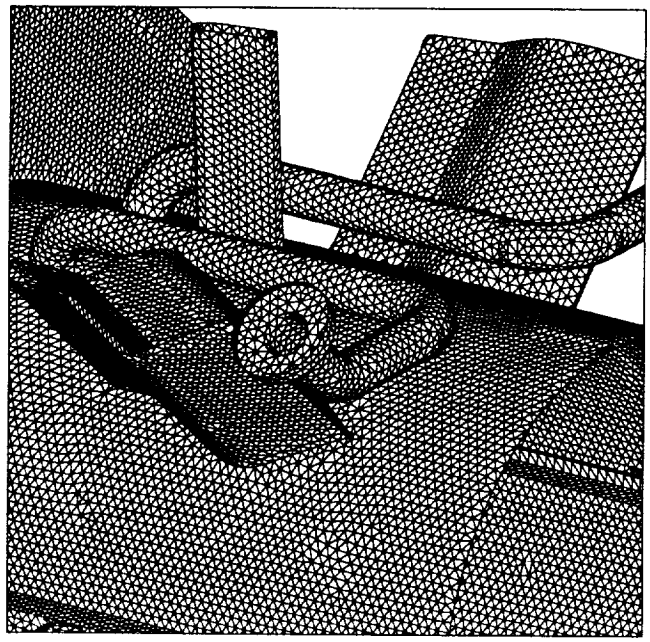
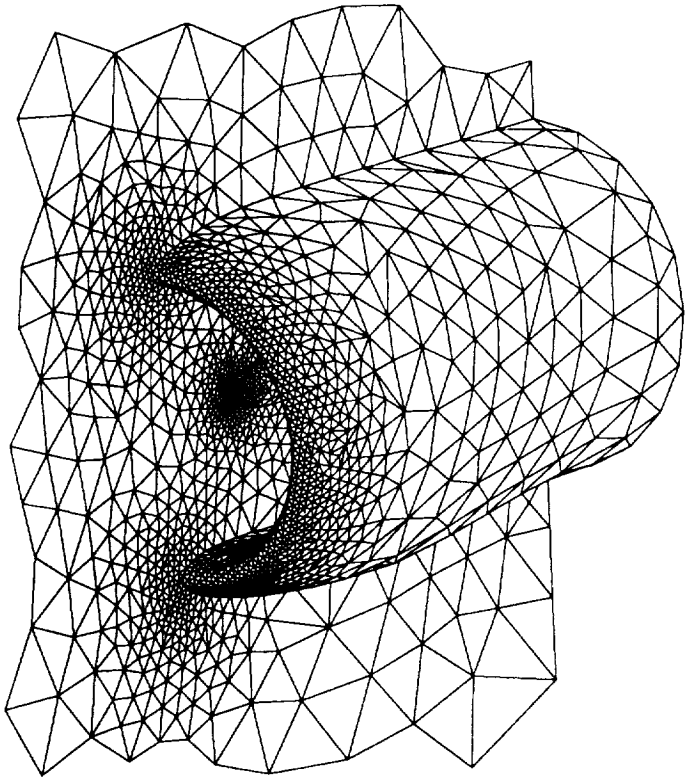
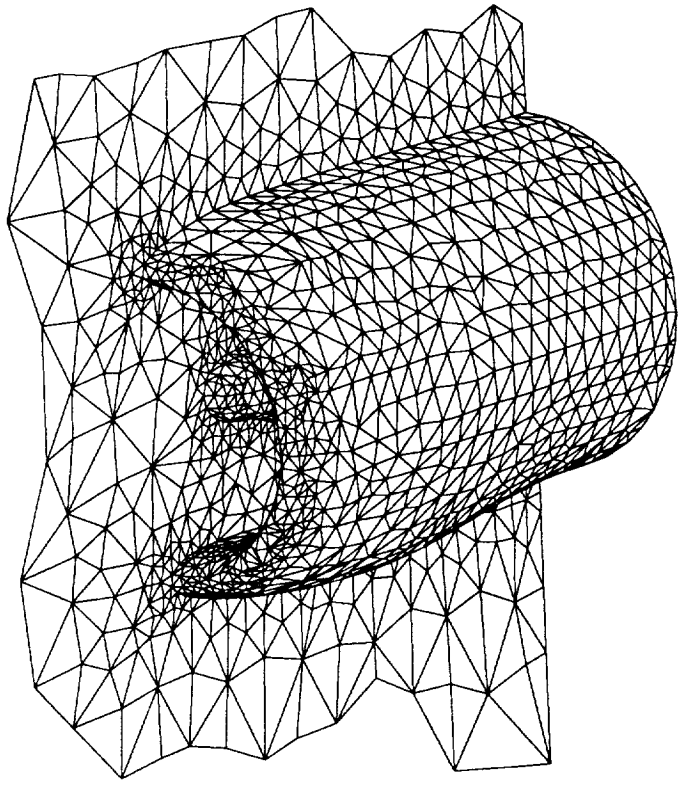


Figure 3 Surface mesh for 3D geometry



advancing front, nn=5805, nc4=28407



octree nn=5652, nc4=26175

Figure 2 Comparison of advancing front and octree meshes

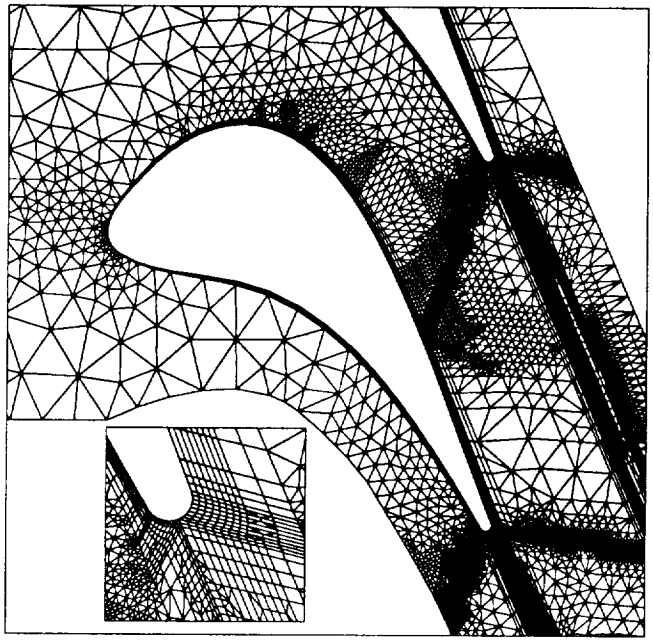


Figure 4 Mixed triangular/quadrilateral mesh

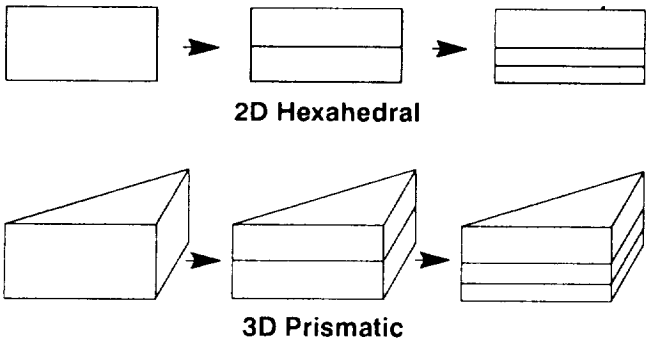


Figure 5 2D and 3D directional cell division

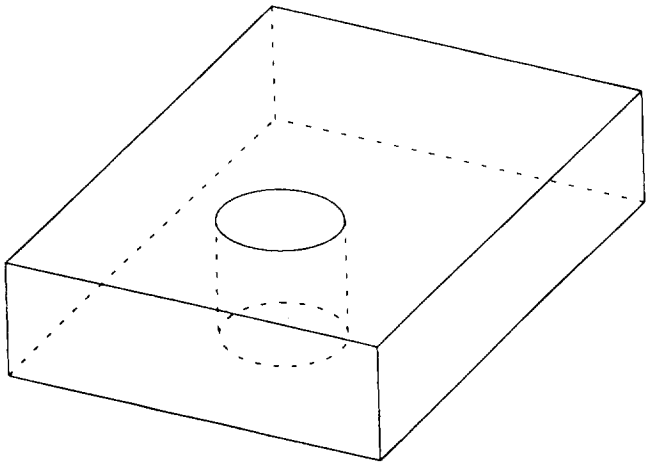


Figure 6 Cylindrical hole in box

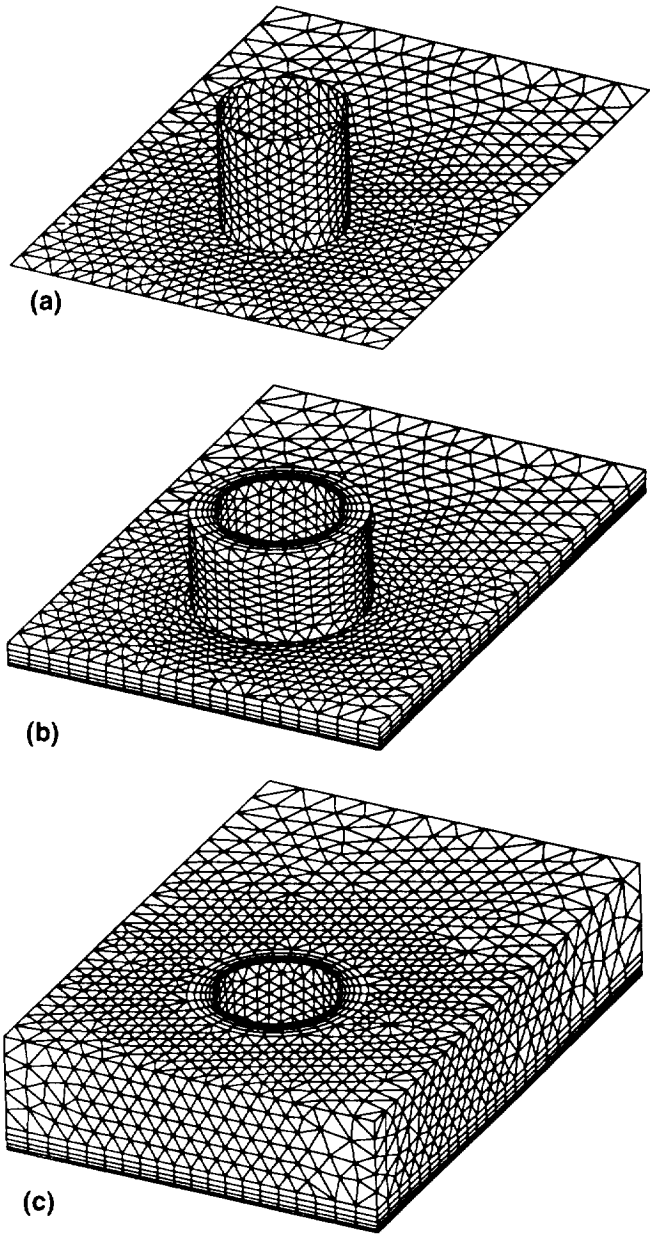


Figure 7 Stages of viscous mesh generation

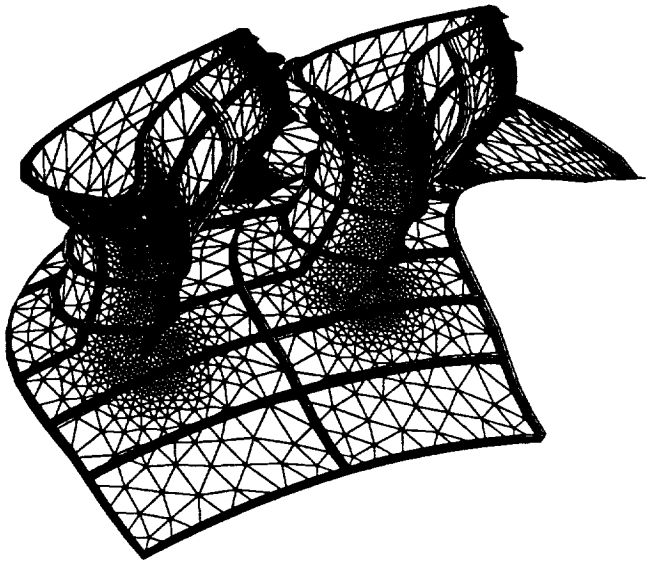


Figure 8 Prismatic mesh for turbine

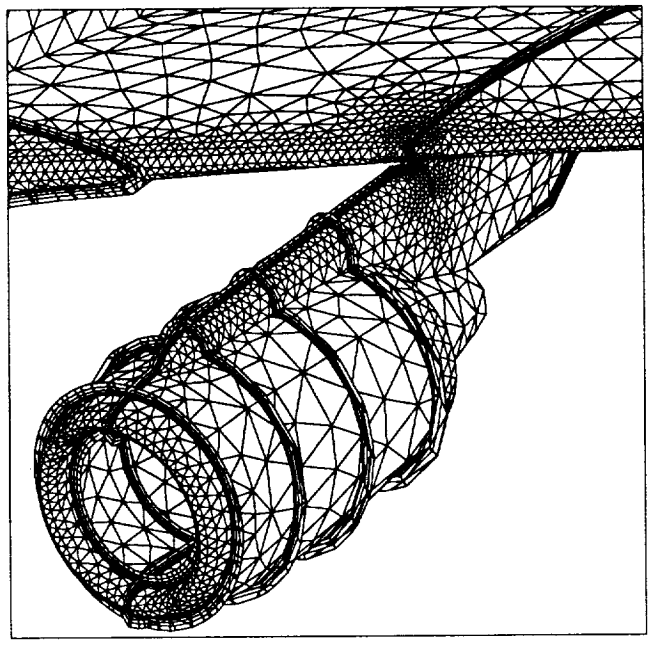


Figure 9 Prismatic mesh for installed nacelle

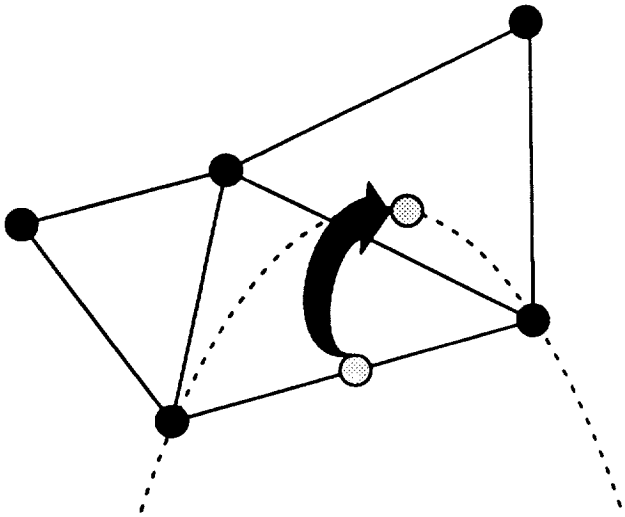


Figure 10 crossover problem for refined mesh

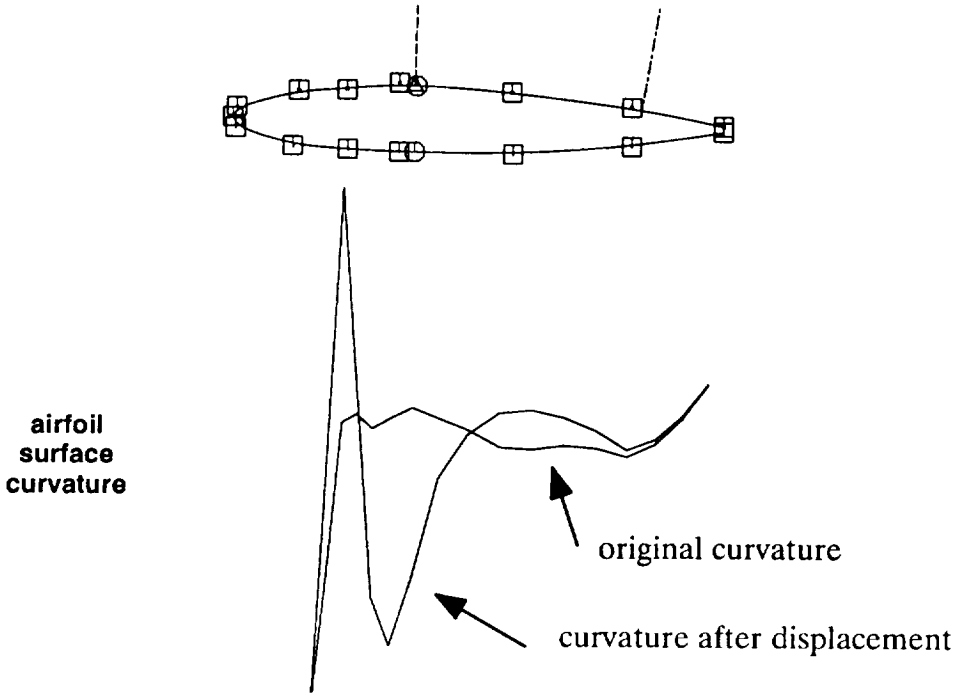


Figure 11 Response of surface curvature to displacement of a single cubic spline control point on the upper airfoil surface.

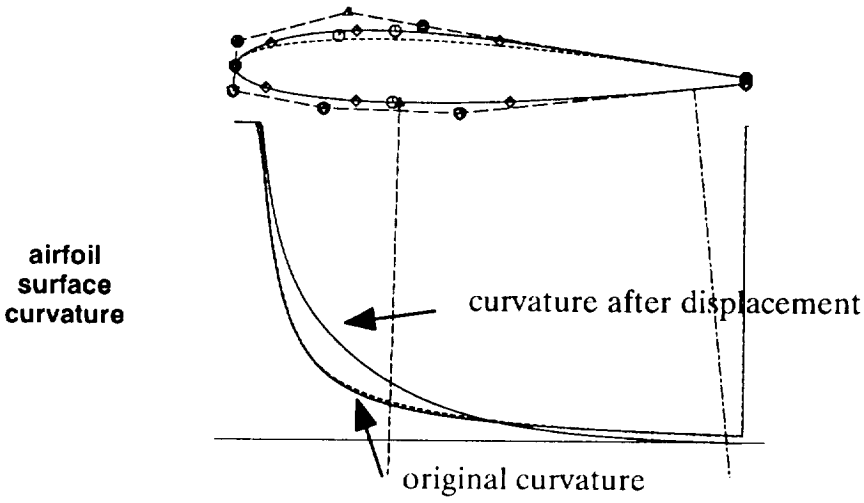


Figure 12 Response of surface curvature to displacement of a single Bezier control point (Bezier curve of degree 4).