Unstructured Mesh Generation Including Directional Refinement For Aerodynamic Flow Simulation*

J. Peraire
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology, Cambridge, MA 02139, USA
and
K. Morgan
Department of Civil Engineering
University of Wales, Swansea SA2 8PP, UK

Abstract

A method for generating directionally refined unstructured tetrahedral meshes is presented. The motivation is the need to efficiently mesh the complex computational domains which are frequently encountered in aerospace applications. In this context, the method is applicable to the construction of meshes suitable for the simulation of inviscid and viscous aerodynamic flows. Minimum user intervention is required and the user specified stretching distribution is achieved by locally modifying an existing mesh. The interfacing of the mesh generation procedure with available geometry modellers is discussed and certain implementation issues are also addressed. Examples are presented which illustrate the performance of the proposed methodology.

Introduction

In the area of computational aerodynamics, unstructured mesh technology has developed considerably over the past ten years. In particular, significant progress has been made in the area of automatic tetrahedral mesh generation, so that the approach is now beginning to be used in a production mode within the aerospace industry. However, the ease with which complex configurations can be meshed is, frequently, offset by the difficulties associated with the generation of highly stretched tetrahedral meshes. Stretched meshes are necessary for the efficient computation of certain inviscid flows, such as flows over wings, and for all flows containing thin viscous layers. Within the unstructured mesh community, different approaches have been adopted in an attempt to address this problem. Thin viscous layers have been meshed using either structured mesh concepts, by essentially "inflating" the surface triangulation, or hybrid approaches, merging structured and unstructured meshes [1]–[6]. Although some degree of success has been achieved, these approaches suffer from well–known disadvantages, such as the lack of geometric flexibility, or the requirement for excessive user intervention, or the generation of low quality meshes. Recognising these disadvantages, Pirzadeh [7] has recently proposed an approach in which more appropriate meshes can be constructed by incorporating a multidimensional stretching capability directly into an unstructured mesh generator.

In this paper, we propose an alternative approach, which allows for the automatic generation of stretched tetrahedral meshes by directionally refining prescribed regions of a previously generated isotropic mesh. The stretched tetrahedra which are generated are such that the largest angle in any tetrahedron is close to ninety degrees. The approach requires the specification of little extra information, above that required by a standard unstructured mesh generator. The directional refinement is driven by a scalar function which measures the distance to a prescribed curve or surface. Features such as leading and trailing edges can be efficiently discretised in this manner, so that appropriate meshes for both inviscid and viscous flows can be generated.

*Based on a Paper Presented at the 5th International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Mississippi, April, 1996

Full details of the basic unstructured mesh generating method are beyonfd the scope of this paper. Instead, we concentrate upon those aspects of the process which are considered to be either novel or relevant to the development of efficient unstructured mesh generators. Efficiency of implementation is a key ingredient when assessing the applicability of most of the algorithms to be described here. With the rapid advances which are occurring in computer hardware, there is a clear demand for fully automatic meshing tools which can produce very large meshes with minimal user intervention. Therefore, an important aspect of the generator is that it can be coupled to existing solid modelling and CAD systems. In addition, it includes capabilities which allow for the manipulation and limited repair of geometry, as well as the definition of additional geometry required in certain simulations, e.g. wakes behind lifting surfaces. The example applications which are included will demonstrate some of these features.

Geometry Modelling

The mesh generation system distinguishes between topological and geometrical entities. The topology of the domain to be meshed generally consists of regions, faces, edges and vertices. Regions are closed volumes in three dimensional space, which may be either simply or multiply connected. The boundary of a region is made up of a collection of faces. Each face is bounded by edges, which form one or more closed loops. Edges connect with each other at vertices. As an additional complexity, "floating" topological entities, such as structural shell elements, modelled as having zero thickness, or thin wires immersed in a three dimensional space, are allowed to be present.

Faces, edges and vertices must have appropriate geometrical representations. These are referred to as surfaces, curves and points respectively. In addition, a geometry can be either native or foreign to the mesh generation system. Native curves are cubic Ferguson splines and native surfaces are bicubic Ferguson patches. Foreign geometry, on the other hand, can be arbitrary if an interface library is provided which enables such geometric entities to be queried. Libraries of this type are provided within most commercial CAD or solid modelling systems.

This conceptual distinction between topology and geometry facilitates the interface between the mesh generator and existing geometry modelling systems. Within a given model, foreign and native geometry can coexist. This feature is particularly useful when repairing existing geometry or when defining edges or surfaces which are not part of a physical model, and therefore not in the solid model, but which are convenient for mesh generation/simulation purposes, e.g. wakes. In addition, the ability to mix different classes of geometry means that it is unnecessary to convert every geometric definition to the native representation. In industrial implementations, this feature has been found to increase the robustness of the overall mesh generation process considerably. For the applications reported in this paper, the Parasolids [9] solid modeller has been employed.

Conventional Mesh Generation Process

In the standard phase of the mesh generation process, edges are discretized into line segments, faces are discretized into triangular planar facets and regions are discretized into tetrahedral elements. The desired spatial distribution of mesh size is specified by means of a background mesh [10, 11] supplemented by a combination of point, line and surface sources [8].

Curve Mesh Generation

The edges are discretized first, by placing points on the associated curves. Contiguous nodes are then joined by straight line segments. These segments and points will form edges and vertices respectively of triangles and tetrahedra in the final mesh. The length of the segments must therefore be consistent with the desired local distribution of mesh size and this can be accomplished in a number of different ways. The method described by Peiró et al [8] is adopted here.

Surface Mesh Generation

In the next step, the discretization of each boundary face is considered in turn. The algorithm employed involves the simultaneous creation of elements and points and is a modification of the basic advancing front method. Peraire et al [10] give a detailed description of this algorithm for planar domains.

The surface parametric coordinates of the points which have been generated on the edges forming the boundary of the face are calculated. An initial front, made up of all orientated straight line segments formed on the boundary edges, is then created. Boundary edges are sorted into closed loops. The orientation of each loop is determined automatically according to the sign of the enclosed area. This area is computed by performing a line integral on the surface parametric plane. When the boundary edges form more than one closed loop, the loop enclosing the largest area is assumed to be the outer loop and all the other loops are assumed to be interior.

The surface generation is a recursive process which terminates when the number of available edges in the front is zero. When a edge is selected from the front, the creation of a triangle involves two main steps. The first step consists of the selection of the connecting point and the creation of a new point, if necessary. In the second step, the connectivity information is verified and updated. The operations involved in the first step, such as the determination of the triangle quality, are best performed in the three dimensional space while the second step requires only logical operations and intersection checks, which are readily performed in the two dimensional parametric space.

Determining the optimum location for new points is an expensive operation, but it is found to have a significant influence on the quality of the generated meshes and on the robustness of the overall algorithm. Essentially, a point on the surface is sought which is at a distance, determined by the local specification of mesh size, from the two points that form the front segment under consideration. This problem is easily cast in the form of computing the intersection between a circle and a surface. For efficiency, the normal direction to the surface can be used to bound the portion of the circular arc that needs to be checked for the intersection.

Operations such as the evaluation of the surface parametric coordinates of a point, or the calculation of the intersection between a circular arc and a face, are carried out by procedures which depend on the type of geometry associated to the face. If the surface is native, these operations are performed in the mesh generation code. They are performed by the solid modeller, Parasolids [9] in this case, if the surface is foreign. It is often found that the procedures provided by the solid modeller take advantage of the specific surface types, e.g. analytical or NURBS, and this makes the operations more robust and efficient.

Volume Mesh Generation

The volume mesh generation uses a Delaunay incremental Bowyer–Watson point insertion algorithm [12, 13]. The main stages in the volume generation process are:

Initialization

As a first step, a tetrahedron is created which is sufficiently large to completely enclose the three dimensional domain which is to be meshed.

Insertion of Boundary Points

The points generated on the edges and faces of the domain are inserted into the triangulation. Points generated on any floating topological feature are also inserted at this stage.

Insertion of Interior Points

Additional interior points will be placed, as necessary, at the centroids of existing tetrahedra, following the algorithm proposed by Weatherill et al [14]. The specified desired element size distribution is employed to determine the elements which need to be subdivided. Points are inserted in groups, ensuring that, within each group, points do not lie too close to each other. The process is repeated until all the elements satisfy the prescribed element size distribution. The approach is highly efficient as, unlike the basic Bowyer-Watson algorithm, it generally avoids the requirement of searching for the element containing the inserted point. It is possible that the element which originally contained a given point might have been destroyed by the previous insertion of another point in the same group. In this case, a local search is easily accomplished. An additional check, which requires that a new point should not be too close to an existing face or edge, has also been implemented. The threshold distance employed in this check is determined by the local distribution of

spacing. It has been found that, following the implementation of this check, the process of recovering the boundary triangulations is simplified.

Boundary Recovery

It is required, at this stage, to ensure that all the triangles on the faces are facets of tetrahedra in the volume mesh. To achieve this, the volume mesh may be changed, by performing edge or face swaps, or the volume mesh and the face triangulations may be changed, by adding points on the faces. The latter option is only adopted when the swapping algorithms which are employed are incapable of recovering the boundary triangles. When points need to be added on the surface, it is often possible to remove them, in a postprocessing operation, once the boundary triangulation has been recovered. The procedures used to recover the boundary faces follow closely those presented by Weatherill et al [14]. An alternative approach, which allows for the boundary to be recovered without the need for modifying the boundary triangulation (provided no interior points lie on the boundary triangles), is presented by George [15]. In this approach, new points are added, if necessary, in the domain interior only. When all the boundary triangles exist in the three dimensional mesh, tetrahedra that are exterior to the region to be meshed are flagged and removed. Each region is considered in turn and the set of faces forming the region boundary is determined from the topological data. Automatic face orientation is performed by a procedure analogous to that described for edge orientation. In addition, if the boundary faces form more than one shell, the shell enclosing the largest volume is assumed to be the outer shell and all the others are assumed to be interior, representing holes or cavities in the domain. It is found that the ability to automatically orientate the edges relative to faces, and faces relative to regions, greatly simplifies the input data and the process of interfacing with existing modelling systems. It should be noted that the edges and faces need to be discretised before they are orientated, as the discretization is required to perform the boundary integrals efficiently. When the boundary of each region, with the corresponding orientation is known, it becomes an easy matter to identify the elements which lie inside and outside the domain.

Directional Refinement

Directional refinement can be applied to an existing isotropic mesh. In the regions which are directionally refined, the mesh consists of stretched elements which are such that the largest angle in any triangle or tetrahedron is close to ninety degrees. The directional refinement approach is, in many respects, an extension of the procedure proposed by Barth [16] for the two dimensional case. The amount of directional refinement, or stretching, which is required is based upon the distance to a specified curve or surface. Every point in space can be assigned a scalar value δ, which represents the distance between the point and the nearest point on a specified curve or surface. A one dimensional point distribution needs to be specified by the user for every curve or surface which is to be used for the refinement, see figure 1. This represents the target for the distribution of mesh points in the vicinity of the curve or surface in the directionally refined mesh. A geometric point spacing distribution is usually employed, with the additional possibility of having a number of equally spaced points immediately adjacent to the curve or surface.

Each element in the mesh is considered in turn and, if the element size measured with the distance metric (i.e. $\delta_{max} - \delta_{min}$) is larger than the required point spacing distribution, new points are inserted. A list of candidate points is formed which lie on the lines connecting the element vertices and the corresponding nearest point on the curve/surface. The new points are located at sites determined by the specified one dimensional point distribution. The candidates are ordered as shown in figure 1, with the first and second candidate points corresponding to the vertices with minimum and maximum distance respectively. All other vertices are assigned two additional points. The list of candidates is considered in turn, and the first point in the list that is not too close to an existing point or to a boundary face is inserted. The distance used as a threshold in this check is a fraction of the local one dimensional point spacing.

The directional refinement process is implemented in three stages.

Edge Refinement

Points are inserted on the edges to match the one dimensional spacing distribution until all the segments satisfy the required directional refinement requirements.

Face Refinement

Points are inserted on the faces until the size of all the face triangles satisfy the appropriate refinement criterion. Candidate points need to be on the face, and need to be at a prescribed distance from the curve/surface used for refinement. This is achieved by working on the tangent plane to the surface and then projecting the points onto the surface. After projection, the distance to the face/edge used for refinement may be changed. This problem is solved with a simple iterative algorithm. When the new point is inserted, the triangle containing that point is subdivided into three triangles and then local swapping operations are performed using the minimization of the maximum angle as the criterion for swapping. Both edge and face refinement are carried out before the three dimensional Delaunay triangulation process.

Delaunay Point Insertion

The process of creation of interior points, meeting the isotropic mesh size distribution specified by the background mesh, has already been described. Additional points are now inserted to meet the directional refinement criterion, i.e. for every tetrahedral element whose size, measured by the distance metric, is larger than specified, one point chosen from the list of candidates is inserted, provided that it is not too close to another point or to the boundary. This process is repeated until no more points can be inserted. Reconnection at this stage is still performed using the Delaunay criterion.

Minmax Point Insertion

After the boundary triangulation has been recovered, additional points are inserted followed by face/edge swaps based on the minmax criterion to improve the mesh quality. It has been found that this step is necessary to ensure that the projected size of all elements is adequate. Using the Delaunay reconnection criterion, it is possible that some elements will not satisfy the required distance projection criterion, even though none of the candidate points associated with it can be accepted for insertion. This will occur when the candidate points all lie too close to an existing point. At this stage, if an element is found to be too large, and can not be broken using the above procedure, a point is inserted at the centre of the largest edge of the element. The largest edge is determined using the distance metric.

Finally, points are added to the edges and faces to improve the element quality. A typical situation where a point would be added to an edge is illustrated in figure 2.

Face/Edge Swapping

It has been noted above that face and edge swapping is applied during the processes of recovery of the boundary triangles, mesh enhancement by post–processing and point insertion in the directional refinement. The criteria which drive the swapping procedure might be different in each case but the operations involved are analogous. In addition to the straightforward face–edge swapping, in which two tetrahedra sharing a triangular face are converted into three tetrahedra sharing an edge, and vice–versa for the edge–face swapping, more sophisticated operations are also performed. Face and edge swapping is considered simultaneously for all the elements that have a common edge. The space occupied by these tetrahedra is re–tessellated in all possible ways and the combination that best meets the driving criterion is selected. Different combinations are possible, depending upon the number of elements sharing a given edge. The following table gives the number of possible configurations and total number of elements created as a function of the number of tetrahedra originally sharing an edge:

Number of Original Tetrahedra	3	4	5	6	7	8	9	10	11	12	etc.
Possible Combinations	1	2	5	14	42	132	429	1430	4862	16796	etc.
Number of Generated Tetrahedra	2	8	20	40	70	112	168	240	330	440	etc.

Although some optimization is possible, because a given tetrahedron will appear in several combinations, the procedure becomes impractical when the number of original tetrahedra is large. For this reason, edges having more then seven neighbours are rarely swapped.

Implementation

It can be proved that the algorithms described above are certain to be successful when the computations involved are performed using exact arithmetic. However, special care needs to be taken in a practical

implementation to ensure a robust method.

A Delaunay triangulation algorithm requires the implementation of operations such as the calculation of the volume of a tetrahedron, the determination of whether a point lies inside or outside the sphere defined by the four vertices of a tetrahedron, etc. Barth [16] noted that all such operations can be cast in such a form that only addition, subtraction and multiplication are necessary. Since division is absent, these operations can be carried out exactly using integer arithmetic. For instance, determining unambiguously whether the triangle *123* of figure 3 intersects with the segment *45* can be readily accomplished once the volumes of the tetrahedra *1234*, *1235*, *1254*, *2354* and *3154* have been determined. The volume V of the general tetrahedron *1234* can be readily determined as

$$6V = \begin{vmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ x_4 - x_1 & y_4 - y_1 & z_4 - z_1 \end{vmatrix}$$

Exact arithmetic is implemented as a redundant operation for those cases where the test carried out in standard double precision becomes ambiguous. This means that only a very small percentage (less than 0.01%) of the operations required for mesh generation need to be carried out in integer arithmetic mode. The procedure is implemented by normalizing all the point coordinates so that they lie inside the interval $[-10^9, 10^9]$. After normalization, coordinates are allowed to have only integer values. Note that, within this range, coordinates can be represented using 4–byte integers, but the operations can not. It has been found to be convenient to transform these integer coordinates to 256–radix, thus representing integer numbers as character strings. Summation, subtraction and multiplication are performed using 256–radix arithmetic. Comparison procedures are similarly coded and are used to determine whether the result of a given operation is greater than, smaller than or equal to zero. As an illustration, if two numbers `U = 1111111111` and `V = 1234567890` are considered, their 256–radix representation will be

```
U =  000  000  000  000  000  000  000  066  058  053
V =  000  000  000  000  000  000  000  073  150  002
```

In this representation, the operations of subtraction and multiplication result in

```
U-V =  255  255  255  255  255  255  255  248  164  050
UV =   000  000  000  019  009  104  044  009  197  171
```

where `U-V = -123456779` and `UV = 1371742099862825790`. The longest integers that need to be handled in practice require strings which are 20 characters long and these are generated when the Delaunay circumsphere test is performed. These integers are potentially as large as $(10^9)^5 = 10^{45}$.

Another important implementation issue is the selection of suitable data structures which are flexible enough to allow for the insertion/deletion, and efficient searching, of elements. The Alternate Digital Tree (ADT) data structure [17] is extensively used in the current implementation. For example, all triangles generated on the faces are stored in an ADT so that when performing directional refinement based on a given face, the triangles on that face, which are nearest to a given point, can be efficiently identified.

Examples

Three examples are included which illustrate the described directional refinement capabilities.

Turbine Blade

The first example shows the effect of performing directional refinement driven by the distance to a single curve. A coarse discretization of a generic turbine blade is generated, and the corresponding surface discretization of the hub and blade is shown in figure 4. A directionally refined mesh is then constructed by specifying a geometric distribution of mesh spacing away from the turbine leading edge. The background spacing distribution is kept unchanged. The surface discretisation of the directionally refined mesh is illustrated in figure 5, which shows the enhanced resolution which has been achieved in the vicinity of the leading edge.

Wing

The second example involves the generation of a mesh for the calculation of a viscous flow over a wing. Figure 6 shows the initial discretization of the wing surface. In this case, the figure also shows the discretisation of a planar surface which has been added to the original geometry to represent the wake. This mesh is directionally refined using the requirement that, at any point, the mesh spacing is governed by the distance to the leading and trailing edges and their corresponding one dimensional spacing distributions. The resulting surface mesh is shown in figure 7. When, as in this case, several refinement edges (and/or faces) are used simultaneously, there exists the flexibility for employing different spacing distributions for each edge or face. The distance associated to a given point is then determined as the minimum distance specified by the relevant one dimensional spacing distributions. For the case shown in figure 7, for instance, the spacing distribution for the leading edge is twice as fine as that specified for the trailing edge. Finally, one additional refinement has been carried out, driven by the distance to the wing and wake surfaces. The final surface mesh is shown in figure 8. It is noted than when the refinement procedure is driven by an edge, the tetrahedral elements which are generated have two small dimensions and a larger one. By contrast, when the refinement process is driven by a surface, the elements generated have only one small dimension and two larger ones.

Twin Engine Executive Jet

The final example is a realistic aerospace application and involves the generation of a mesh suitable for a viscous flow simulation over a twin engine executive aircraft. The first mesh which is generated is shown in figure 9. Although they are not shown in this figure, additional surfaces were added to allow for the accurate modelling of the wakes from the wing, the horizontal stabilizers and the engine pylon. To resolve the engine plume, a cylindrical surface, starting at the nacelle trailing edge and extending to about a distance of three quarters of the fuselage length downstream, was also added. A number of refinement operations are performed with the objective of generating a suitable mesh. It has been found to be beneficial to consider surfaces which meet approximately at right angles, such as wing/body or fin/tail, in separate refinement stages. The resulting mesh in this case is found to be approximately the union of the boundary layer meshes which would be produced by considering each of the surfaces alone. This provides enhanced resolution in corner regions, which are likely to be dominated by viscous effects. The mesh produced after 3 directional refinements is shown in figures 10, 11 and 12. The effect of the directional refinement due to the viscous walls and wake surfaces is readily apparent. Although this mesh is probably still too coarse for a realistic calculation, it illustrates the main features and capabilities of the approach.

Conclusions

A directional refinement procedure for unstructured tetrahedral meshes has been implemented and applied to a number of examples. The amount of user information which has to be provided is minimal and, hence, the mesh generation process is highly automatic. The applicability of the proposed approach will ultimately be determined by the quality of the numerical results which can be produced. Computations, using meshes generated by this method, are already being undertaken [18].

The directional refinement capability is found to be very effective in resolving features such as leading and trailing edges. High mesh resolution is provided in corner regions where viscous effects are likely to dominate the flow.

It can be observed that the refinement produces a highly regular pattern within the mesh near the selected curves/surfaces. This means that tetrahedral cells could be readily grouped into triangular prisms. Although a grouping of this form would not completely eliminate the tetrahedral cells in these regions, it would result in a considerable reduction in the total number of cells in the mesh.

For large scale viscous problems, the ability to generate meshes in some form of distributed mode will be essential. A desirable feature will be that when the mesh generation process is completed the mesh is already partitioned and distributed across the processors. Initial efforts in this direction are already underway [19].

Acknowledgements

We wish to express our thanks to General Electric CR & D, Schenectady, for partially supporting this work and to Electronic Data Systems for making the Parasolids system available for this research. The support

of R. Haimes, O. Hassan and J. Peiró is also gratefully acknowledged.

References

[1] K. Nakahashi, 'FDM–FEM zonal approach for computations of compressible viscous flows', in *Lecture Notes in Physics*, **264**, Springer–Verlag, Berlin, 494–498, 1986.

[2] S.D. Connell and M.E. Braaten, 'Semi–structured mesh generation for 3D Navier-Stokes calculations', *Mechanical Systems Laboratory, GE R& D Center Report*, 94CRD15, 1994.

[3] Y. Kallinderis, A. Khawaja and H. McMorris, 'Hybrid prismatic/tetrahedral grid generation for complex geometries', *AIAA Paper 95–0211*, 1995.

[4] O. Hassan, K. Morgan, E.J. Probert and J. Peraire, 'Unstructured tetrahedral mesh generation for three–dimensional viscous flows', *Int. J. Num. Meth. Engng.*, **39**, 549–567, 1996.

[5] M.J. Marchant and N.P. Weatherill, 'Unstructured grid generation for viscous flow simulations', in *Proc. 4th Int. Conf. on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Pineridge Press, Swansea, 151–162, 1994.

[6] S. Pirzadeh, 'Viscous unstructured three–dimensional grids by the advancing layers method', *AIAA Paper 94–0417*, 1994.

[7] S. Pirzadeh, 'Progress toward a user–oriented unstructured viscous grid generator', *AIAA Paper 96–0031*, 1996.

[8] J. Peiró, J. Peraire and K. Morgan, 'FELISA system reference manual and user's guide. Volume 1', *University of Wales Swansea Report*, CR/821/94, 1994.

[9] Parasolid v7 System Manuals, Electronic Data Systems, 1995.

[10] J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz, 'Adaptive remeshing for compressible flow computations', *J. Comput. Phys.*, **72**, 449–466, 1987.

[11] J. Peraire, J. Peiró and K. Morgan, 'Adaptive Remeshing for Three- Dimensional Compressible Flow Computations', *J. Comput. Phys.*, **103**, 269–285, 1992.

[12] A. Bowyer, 'Computing Dirichlet tessellations', *Comput. J.*, **24**, 162–166, 1981.

[13] D.F. Watson, 'Computing the n–dimensional Delaunay tessellation with application to Voronoi polytopes', *Comput. J*, **24**, 167–172, 1981.

[14] N.P. Weatherill, O. Hassan, and D.L. Marcum, 'Calculation of steady compressible flow fields with the finite element method', *AIAA Paper 93–0341*, 1993.

[15] P.L. George, F. Hetch and E. Saltel, 'Automatic mesh generator with specified boundary', *Comp. Meth. Appl. Mech Engng.*, **92**, 269–288, 1991.

[16] T. Barth, 'Aspects of unstructured grids and finite–volume solvers for the Euler and Navier–Stokes equations', *VKI Lecture Series on Computational Fluid Dynamics*, Brussels, 1994-05.

[17] J. Bonet and J. Peraire, 'An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems', *Int. J. Num. Meth. Engng.*, **31**, 1-17, 1991.

[18] J. Peraire, O. Hassan and K. Morgan, 'Navier Stokes computations in 3D using unstructured meshes', *AIAA Paper 96–1998*, 1996.

[19] T. Okusanya and J. Peraire, 'Parallel unstructured mesh generation', Presented at *5th Int. Conf. on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Mississippi, 1996.

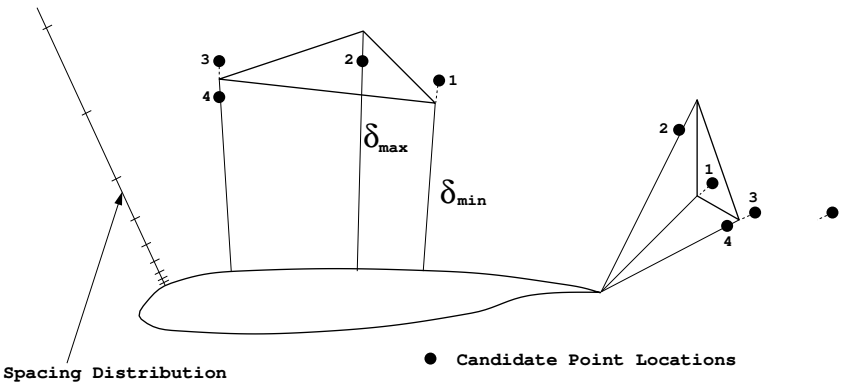

Figure 1: Directional Refinement

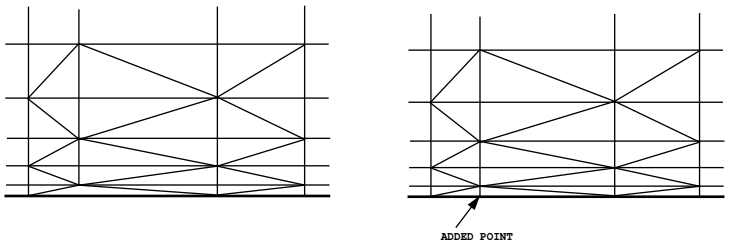

Figure 2: Edge refinement to improve element quality

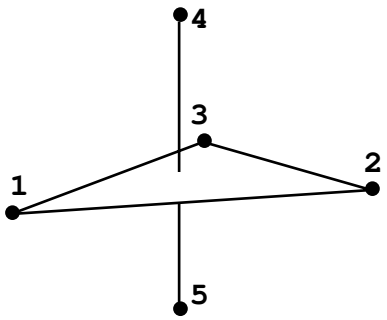

Figure 3: Triangle-segment intersection

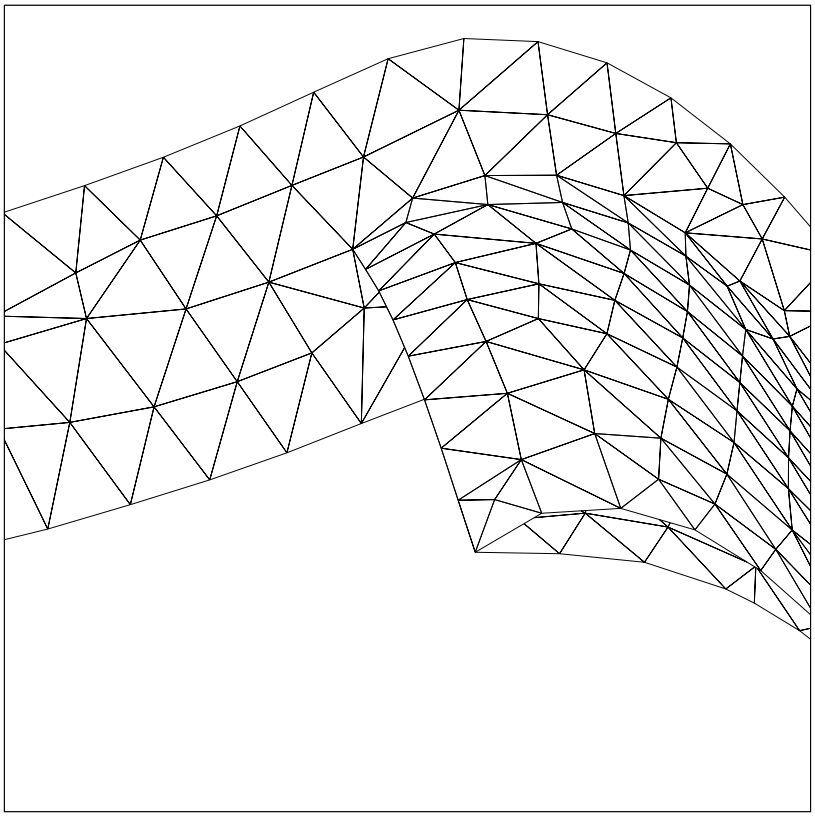

Figure 4: Turbine blade: original unrefined mesh

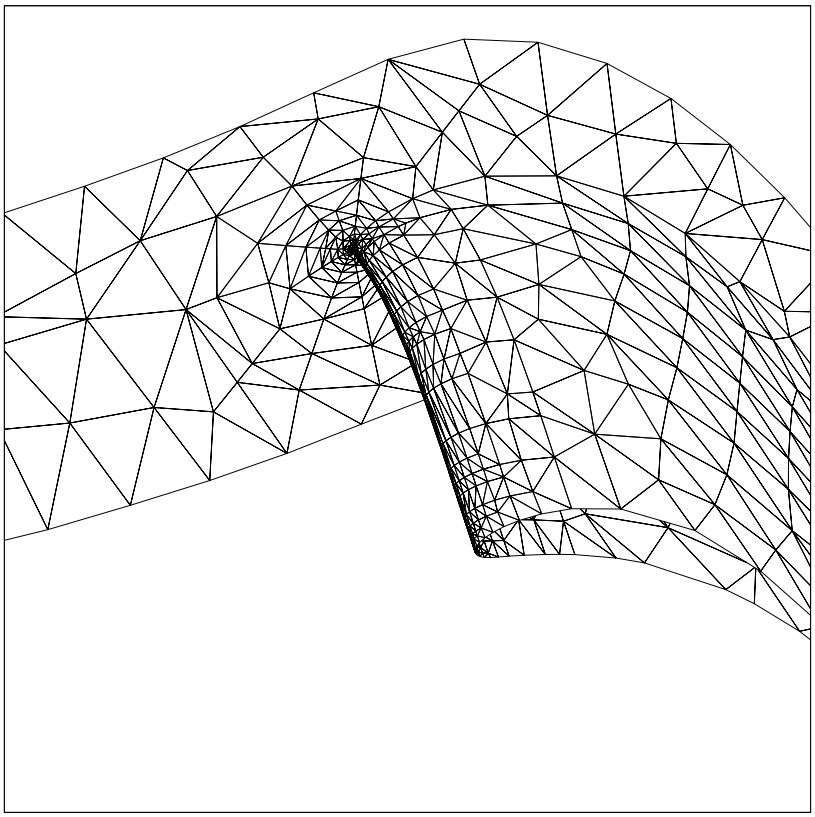

Figure 5: Turbine blade: directional refinement based on distance to blade leading edge

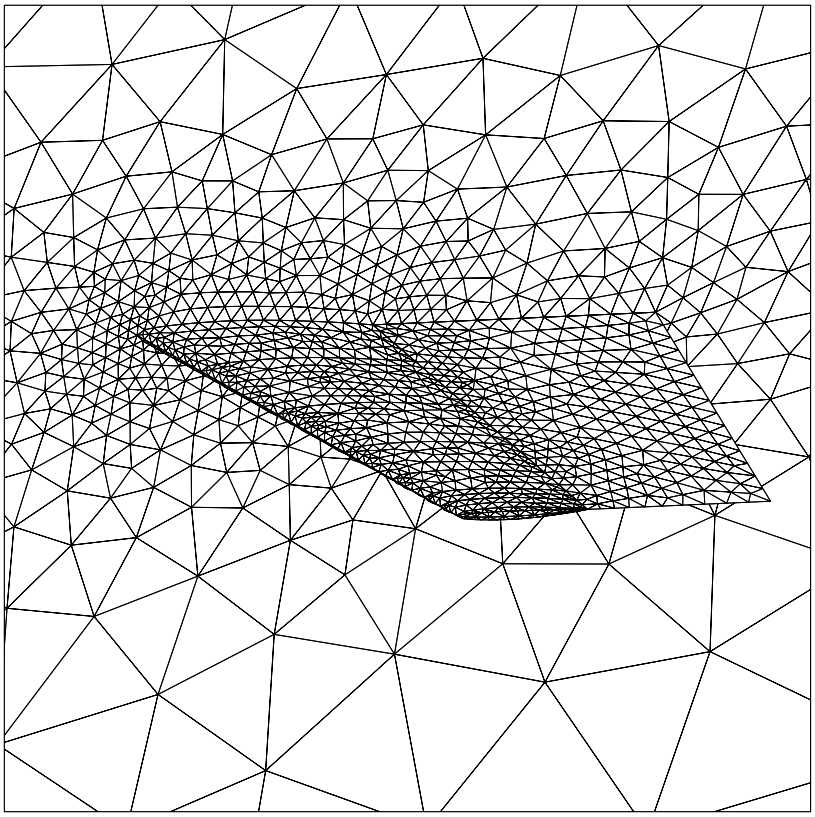

Figure 6: Wing : original unrefined mesh

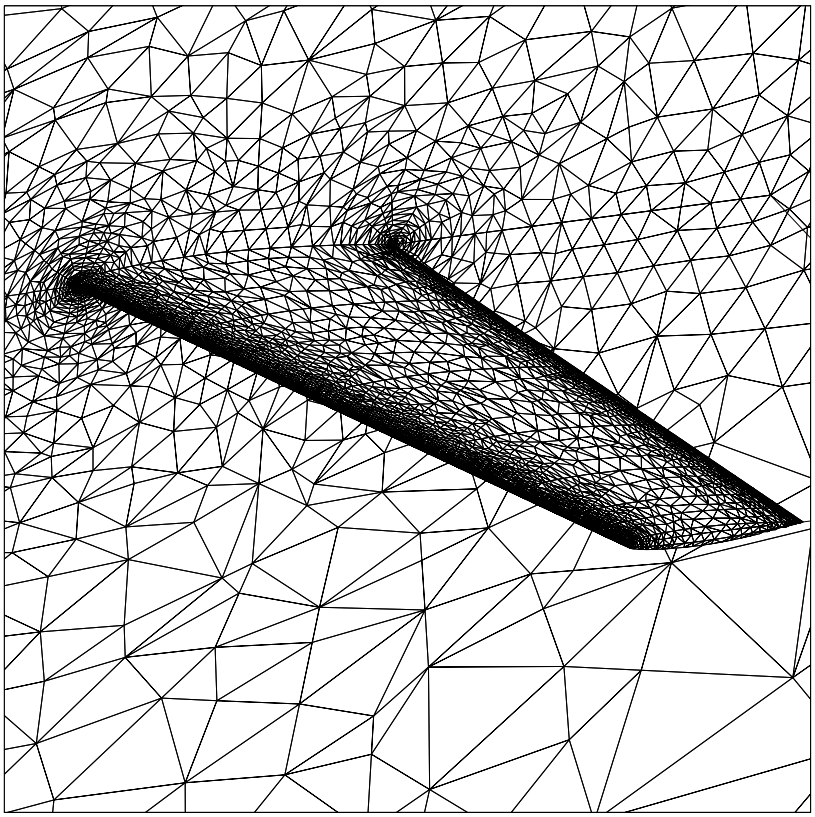

Figure 7: Wing : directional refinement based on distance to leading and trailing edges

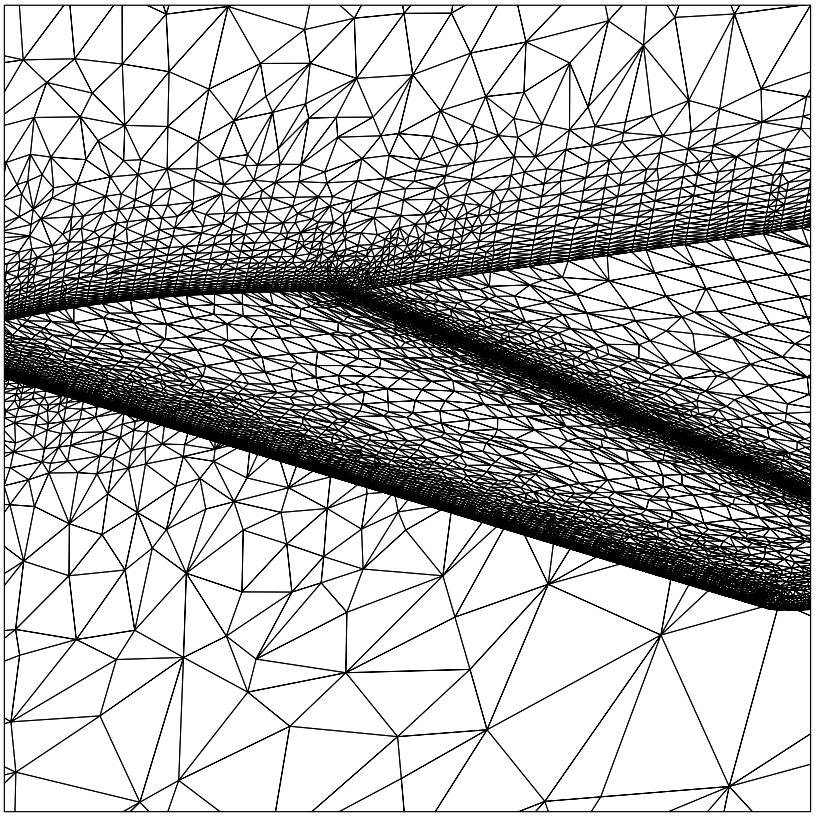

Figure 8: Wing : directional refinement based on distance to wing and wake surfaces

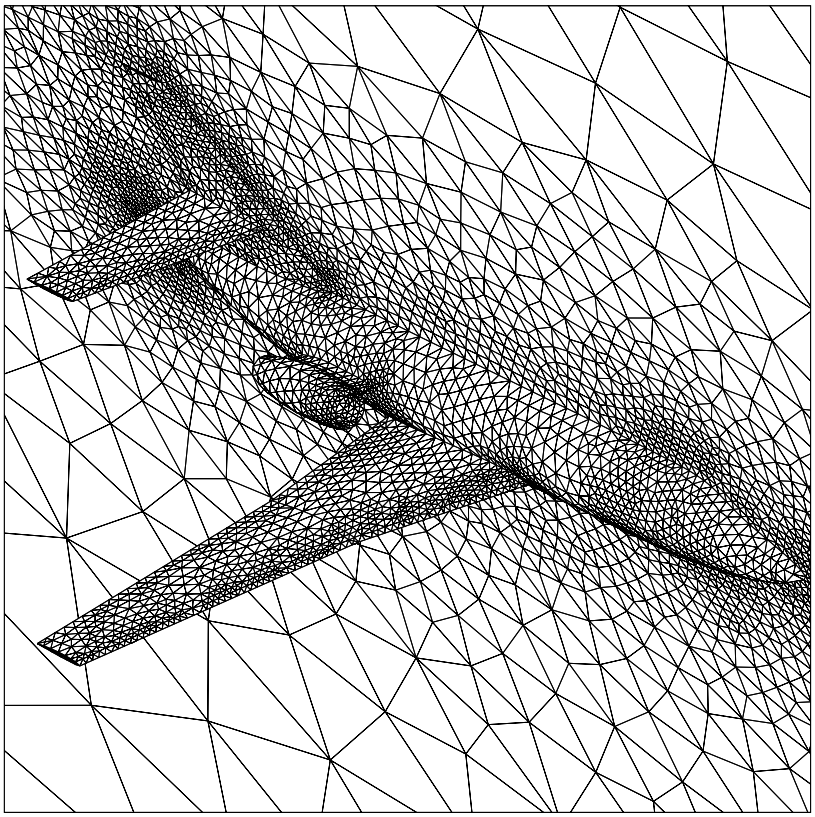

Figure 9: Twin engine jet : original unrefined mesh

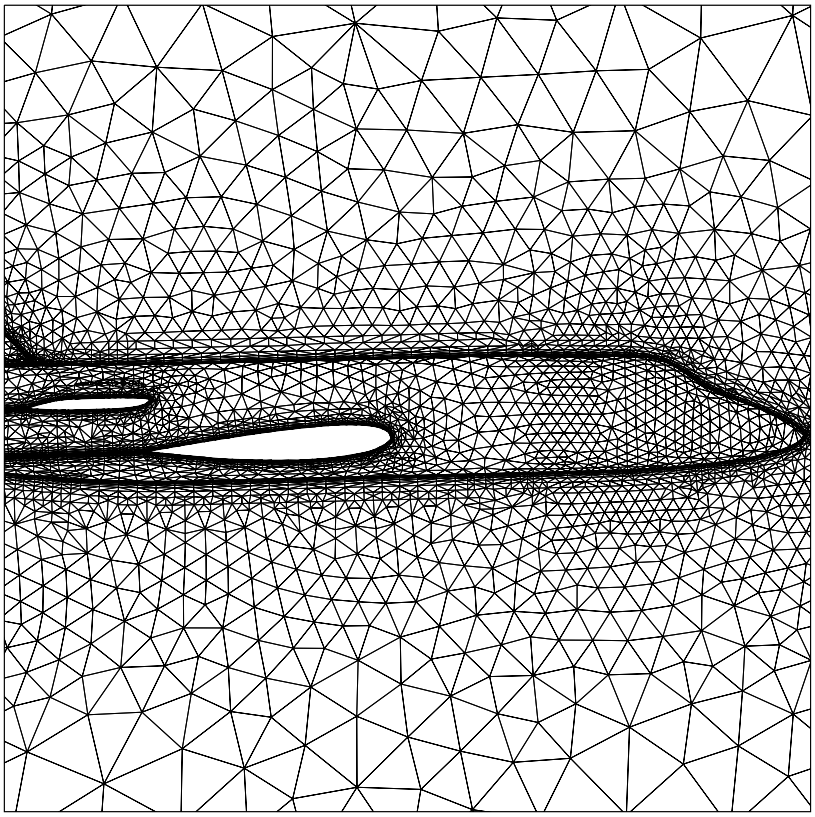

Figure 10: Twin engine jet : directionally refined mesh

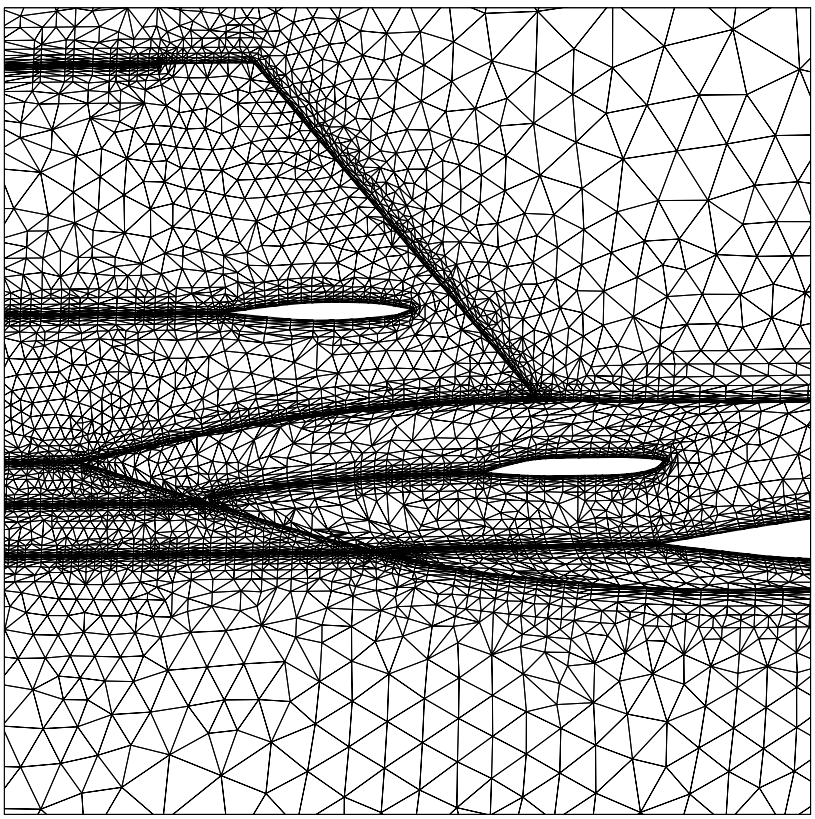

Figure 11: Twin engine jet : directionally refined mesh

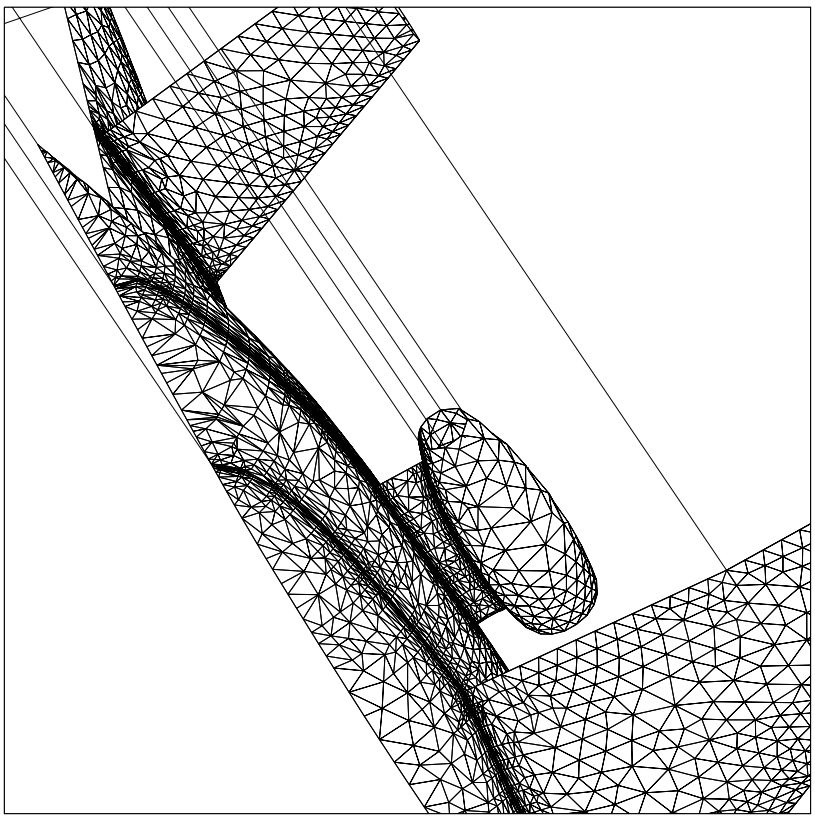

Figure 12: Twin engine jet : directionally refined mesh