THREE-DIMENSIONAL UNSTRUCTURED GRID GENERATION FOR FINITE-VOLUME SOLUTION OF EULER EQUATIONS

Karim Mazaheri, Shahram Bodaghabadi
Sharif University of Technology

Keywords: *Unstructured grid generation, Delaunay triangulation, Euler equations*

Abstract

To solve the Euler equations on a three-dimensional unstructured grid, an efficient procedure for generation of relatively high quality elements is presented. The basic steps of this method, which is essentially based on Delaunay's triangulation, is consisted of generating surface points, initial tessellation, point distribution function interpolation, internal node generation and update triangulation. Special treatments for surmounting degenerate cases are considered within the algorithm. Validation and reliability of algorithm are checked by different examples. An upwind scheme is used to solve Euler equations on these unstructured grids for steady-state problems. Spatial discritization is accomplished by a cell-centered finite-volume formulation using flux-difference splitting method of Roe. Solution is advanced in time by a simple explicit scheme. Local time stepping is used to accelerate convergence to the steady state. Some application cases are treated to test the method.

1 Introduction

Structured grids have been widely used for a long time in CFD applications [1]. Different algorithms of generating structured grids, solving basic equations on them and pre or postprocessing operation on such grids are well developed. They use less memory per unit element compared with unstructured one. Also, methods of generating structured meshes are simpler. However, generating structured grids about complex configurations, especially automation of it, is still a difficult task. Unstructured grids are stronger in covering complex fields, automation of generation process is simple, variation of element size in grid is appliable and adaptation process is simpler. In versus of such advantages, data structure of unstructured grids is more complex than structured one and optimum process of grid generation has a complex logic and need extra work. There are a variety of unstructured grid generation methods. Many of these methods are essentially based on two famous schemes i.e. "Delaunay Triangulation" and "Advancing Front Method" algorithms. Delaunay triangulation methods, with applications in CFD, are presented by Lawson, Green, Sibson [2,3], Bowyer [4] and Watson [5]. In recent years various modifications to Delaunay's method have been presented to optimize and eliminate its practical difficulties. Frey [6] proposed an algorithm for 2D point creation based on circumcircle and inscribed circle centers. This algorithm was linked with Watson algorithm for triangulation. Schroder and Shephard [7] have also used Watson algorithm. To keep the boundary conforming, they used a topological classification for grid entities, and generated a consistent grid for complex and non-manifold given geometries. Muller and his coworkers [8] presented a strong method for two dimensional grid generation by combining Delaunay's triangulation and advancing front method by point creation. Some efforts in automatic grid generation for three-dimensional applications are also made. Researches of Weatherill and Hassan [9], Golias and Tsiboukis [10], Maccrum and Weatherill [11] and Zheng

et al [12] lie in this category. In present work a combined Delaunay/Advancing front method in 3D has been presented. Different methods of outcoming degenerate cases in Delaunay triangulation are considered. Relatively high quality elements are created to be used in CFD applications. Euler equations have been solved in some test problems, which are meshed by current grid generation algorithm. Following sections describe steps of grid generation algorithm and the numerical solution of the Euler equations. Finally results on some test problems are included.

2 Grid generation method

Methods using advancing front type point placement and Delaunay connectivity have been developed by Delaunay [1], Watson, Lawson, Sibson, Muller et al and George et al [13]. This idea is also used in three-dimensional grid generation by Rebey [14], Pirzadeh [15] and Khosravy. Current work is essentially based on work of Khosravy. Special modifications are applied to improve grid quality, surmounting degeneracies and increasing robustness and reliability of algorithm. To better description of the algorithm some basic definitions are presented.

2.1 Delaunay triangulation

Delaunay triangulation is the geometric dual of the Dirichlet tessellation [4]. Given a set P of M points in an n-dimensional space, there exists a region V_i such that

$$V_i = \left\{x; \left|x - P_i\right| \le \left|x - P_j\right| \ for\ all\ i \ne j\right\} \qquad (1)$$

The collection $V = \sum_{i=1}^{M} V_i$ is defined as the Dirichlet tessellation. Given a Dirichlet tessellation V, one may form the Delaunay triangulation by connecting any two points P_i and P_j whose corresponding tiles V_i and V_j are neighbour. In the 3D case, Delaunay triangulation gives a decomposition of the convex hull of P into 3D simplexes (tetrahedra). An important property of Delaunay triangulation is that a circumsphere of tetrahedron contains no points other than its vertices.

2.2 The grid generation process

The algorithm in step-by-step manner can be expressed as follows:

1. Define a set of points, which form a convex hull containing all other points.
2. Add the boundary nodes to the initial grid, using Watson algorithm. Obtain a valid triangulation of them and recover all boundary surfaces (if they are lost).
3. Assign a point distribution function to each initial boundary point.
4. Calculate a criterion for quality of elements. If quality of an element is under a threshold value or element doesn't satisfy the point distribution function, it is marked as a bad element.
5. Create a new point for each bad element.
6. For each new point, search for the element containing it.
7. Interpolate the point distribution function for the new point from the containing element.
8. Merge two new points that are too close to each other.
9. Reject new points that are too close to an existing point.
10. Insert each accepted new point in the triangulation using the Watson algorithm.
11. Continue from step 4; repeat until no acceptable point is created.
12. Smooth the grid distribution.

Watson algorithm begins by finding tetrahedron, which contains the new point. This tetrahedron and all other tetrahedrons whose circumsphere contain that point are then deleted. This form a polyhedron bounded by triangular faces. Each face of this polyhedron and the new point form a new tetrahedron. With constructing all such tetrahedra, a new triangulation obtained.

2.3 Boundary point creation

Generally there are four basic topological entities in three dimensions. These are vertex, edge, surface and region. Each entity is surrounded by one lower dimension entity. Regions are surrounded by surfaces, boundary of each surface is an edge and each edge has two end vertices. Boundary nodes creation is a procedure to create proper points in boundary edges and surfaces. Edges division is done due to local spacing requirements. Initially points may be distributed uniformly and then relocated using proper methods [9]. Point creation on surfaces is a difficult task in grid generation. In the case of planar surfaces, algebraic methods can be used. For general curved surfaces, using two-dimensional grid generation in surface parametric space is a common method [1].

2.4 Convex hull

Convex hull is a simple large enough region, which contains all nodes in the mesh. A simple tetrahedron can be used, but experience showed that if the number of nodes on the convex hull were low, probability of occurring degenerate cases would be increased. For all examples presented here, a polyhedron with one node in its center is used.

2.5 Nodes insertion

Node insertion algorithm is based on Watson algorithm. This algorithm deletes all neighbors of the containing element whose circumspheres contain the new node. New elements are created using the new node and each face of the hollowed region. The difficult step of this algorithm is the insphere check. This involves making a decision based upon floating point arithmetic. An incorrect deletion of a tetrahedron can result in a disjoint, non-convex empty polyhedron which, when new tetrahedra are created, intersect each other. It has been found that such incorrect decisions can also disconnect and isolate a previously inserted point such that it no longer appears in the data structure of the forming points. Such problems may even arise with simple geometries, such as a set of points, which define a grid on the surface of a unit cube. Founding degeneracies and surmounting them utilizes a lot of efforts in the three-dimensional grid generation development [7,9,10].

2.6 Resolving degenerate cases

When more than four points of the point set lie on the circumsphere of an element (within the resolution of the numerical calculation), the degenerate case occurs. Resolving degeneracies require a consistent decision for a particular point for being inside or outside. In practice, the judgement whether a point P_i is inside a particular sphere C_i of radius r and circumcenter x_c (Figure 1) can be made based on a tolerance factor ε:

$$P_i \in C_i \quad \text{when} \quad \frac{\|P_i - x_c\|}{r} \le \varepsilon \qquad (2)$$

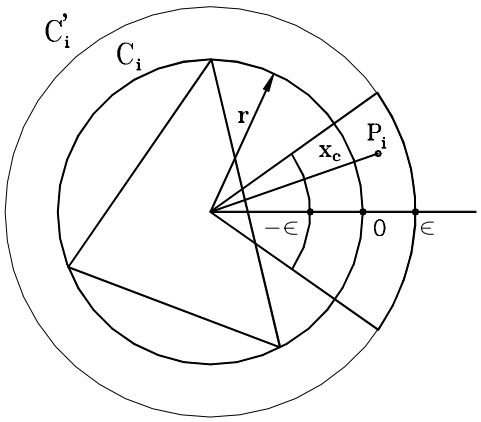

Figure 1 Approximate incircle test

Choosing ε>0 is equivalent to classifying P_i outside of all degenerate C_i. Generally ε is taken positive for convenience. The parameter ε in fact defines an approximate “in/out” sphere C_i' of different radius from the real circumsphere C_i. By using C_i' instead of C_i, the result is that the triangulation may not satisfy the circumsphere condition. But even if ε=0, the effect of numerical calculation errors is to consider C_i' rather than the real C_i. So, to have a valid mesh, the method should account for the

perturbed circumsphere. Three problems can arise when computing with C_i'. First, the resulting triangulation may not be a Delaunay's one within tolerance ε. In terms of generating a computational mesh, as long as the triangulation is a valid one and the elements quality is acceptable, this situation is not of concern. Secondly a point P_i very close to point P_j, where P_j is incorporated into the triangulation prior to P_i, may not be inside any C_i', and therefore will not be participated in the triangulation (Figure 2). In this case, it may be necessary to reduce the size of ε. Thirdly and of most concern, is that inconsistencies are created because the mesh doesn't globally satisfy the Delaunay property of the circumsphere condition. Watson algorithm uses this property to generate an insertion polyhedron to get a valid mesh. The insertion polyhedron must satisfy the condition of point convexity [16], as described below:

A polyhedron with faces F_i, for i=1,…,n_F is strictly point convex to the point P when the inward pointing normal to each polyhedron face, n_i, satisfies the condition of point convexity (Figure 3):

$$n_i.(P - x_i) > 0 \qquad (3)$$

where x_i is an arbitrary point on face F_i.

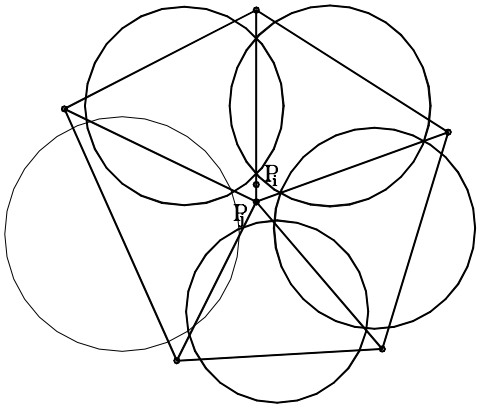

Figure 2 A point not inside any sphere

When the insertion polyhedron is strictly point convex, creating tetrahedra by connecting P with each face F_i produces a structurally consistent mesh. Point convexity checking is readily added to the Watson algorithm to prevent structural inconsistencies. As each point is inserted into the triangulation to create the insertion polyhedron, the condition for point convexity is enforced. In particular when $n_i.(P - x_i) < \delta$ where δ>0, the point convexity condition is not satisfied. When a face F_i does not satisfy the point convexity, the tetrahedron it is connected to, is deleted, forming a new insertion polyhedron. The added faces are again evaluated for point convexity and the procedure is repeated.

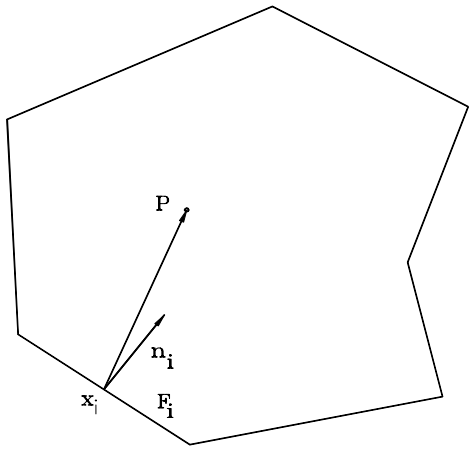

Figure 3 Point convexity rule

2.7 Node creation

Different criteria for checking bad elements are presented [12]. Two of them are the ratio of the smallest edge length to the largest one and the ratio of the inscribed sphere radius to the circumsphere radius. After tagging bad elements, those that lie adjacent to a good one or the boundary, generates the front and new node is created adjacent to each front face. New node is located such that local spacing (based on spacing function) is satisfied and a regular tetrahedron is expected to be created.

2.8 Mesh smoothing

Mesh smoothing (node relaxation) is a technique of increasing the overall quality by suitable repositioning of nodes. Node relaxation is applied to internal and surface nodes. Surface nodes can be moved only on their geometric surfaces. One of the most usual methods is Laplacian smoothing [6], which is used in the

current work, too. Different quality criteria can be used [12]. In present work the ratio of volume to cubic power of the largest edge length is used. In order to have a picture of the mesh overall quality, the mean (Q_m) and joint (Q_j) qualities are introduced [11]:

$$Q_m = \frac{1}{N}\sum_{i=1}^{N} Q_i$$
$$Q_j = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{Q_i} \qquad (4)$$

where N is the number of tetrahedra and Q_i is its quality.

The joint quality factor has a value that is very sensitive to elements with low qualities. In smoothing procedure, for each node repositioning, the mean and joint qualities values are calculated in new position. If these values increase respect to their old values, the node will be moved.

2.9 Data structure

In three-dimensional grid generation the following information are stored: cell vertices, cell neighbors, cell goodness/badness flag, node coordinates and node spacing function. So, the required storage space is 9N integer and 4M floating words, where M is the number of nodes.

3 Solution scheme of the Euler equations

The Euler equations can be written as:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = 0 \qquad (5)$$

The state vector U ($= [\rho \;\; \rho u \;\; \rho v \;\; \rho w \;\; \rho E]^T$) and the flux vectors F, G, H are expressed in terms of the conservative variables. ρ is density, (u, v, w) are velocity components and E is the total energy. E can be related to the other variables by the perfect gas state equation.

A cell-centered method, in which grid elements are used as domain control volumes, is used. Each cell has four faces and fluxes are computed on these faces. Flux quantities are computed using Roe's flux-difference-splitting [17]. The flux across each cell face is computed using

$$F(U_L, U_R) = \frac{1}{2}\left[F(U_L) + F(U_R) - \left|\bar{A}\right|(U_R - U_L)\right] \qquad (6)$$

Here subscripts L and R denote to left and right of the cell face. The Jacobian matrix $\bar{A}$ is evaluated with Roe-averaged quantities [18].

Solution procedure starts by definition of the initial conditions, and then advances in time by using a simple explicit scheme, i.e.

$$U^{n+1} = U^n - \frac{\Delta t}{V}\sum_{i=1}^{4} F_n^i S_i \qquad (7)$$

where F_n^i is the normal flux across face i, S_i is the face area, V is the cell volume and Δt is the solution time step. Superscripts n and n+1 denote the current and the next time step. Steady state solution is achieved when residual term of this equation is lower than a threshold value. Further discussion of this method can be found in many references like [18] and [19].

Initial condition, boundary conditions and stability

In external flows, it is common that free stream condition is applied as initial condition. In internal flows, like flow in supersonic ducts and cascade flows, uniform flow based on inlet condition is usually used.

For solid boundaries (including symmetry planes), the flow tangency condition (slip condition) is imposed by setting the velocity on the boundary ghost cells as mirror image of the boundary cell velocity. Density and pressure in the ghost cells are simply set equal to the boundary cell values.

For far-field boundary, characteristic boundary conditions are applied using the fixed and extrapolated Riemann invariants corresponding to the incoming and outgoing waves. An approximate method is used by proper selection of the conservative variables to be extrapolated from outside the domain or to be

interpolated from the inside due to the local Mach number.

Stability condition is applied by Courant-Friedricks-Lewy (CFL) condition. Local time stepping accelerates convergence to the steady state by advancing the solution at each cell in time at a CFL number near the local stability limit. The expression for the local time step can be represented as [3]:

$$\Delta t = \beta \frac{V}{\sum_{i=1}^{4} \left(\left| \bar{u}_{\perp} \right| + \bar{c} \right)_i S_i} \qquad (8)$$

Here β is a safety factor, $\bar{u}_{\perp}, \bar{c}$ and S_i are the normal velocity, sound speed and area of the face *i*. Bar quantities show Roe-averaged values.

4 Results

In this section sample generated meshes are presented to show validity of the proposed algorithm. Also the supersonic flow around a wedge and a cone are simulated using the Eluer equations.

4.1 Sample meshes

A simple rectangular domain around a wedge is the first example. Figure 4 shows a cross section of this grid. The quality distribution for this mesh is shown in Figure 5. The second example is a grid around a cone. Because of the symmetry, only one-half of the cone is considered. The far-field boundary is a cylindrical surface. A cross section of this grid is shown in Figure 6. Third sample mesh is a more complex domain, i.e. grid around ONERA M6 wing. A spherical surface is used as the far-field boundary. This mesh is shown in Figure 7.

It is appear from Figure 5 shows that the quality factor has a distribution similar to the normal distribution. Average quality for this mesh is about 0.4 (this is true for other examples, too). Of course, the distribution is preferred to be as close as possible to unity. Smoothing could move the peak of the distribution curve to one, but in all examples, the number of cells with quality greater than 0.9 is decreased, as it can be observed in Table 1. In other words, smoothing distributes the "badness" of cells among all cells. Since the overall quality of the grid is important (and not quality of a special cell), it can be deduced that smoothing procedure is advantageous. An important factor, which is also observed in Table 1, is the very low value of the minimum quality in the grid. It means that some bad cells always present in grid. These cells may affect the solution accuracy and in the worst case deteriorate the stability.

4.2 Applications

Supersonic flow around a wedge is a good test problem, which is a really two-dimensional problem, so comparison of the three-dimensional solution with the 2D one is possible. Since the exact solution exists for this problem. A supersonic flow with Mach number M**.** =2.5 around a wedge with half-angle of 15 degree is considered. Mach contours in a 2D section of the domain are shown in Figure 8. Mach number and pressure ratio across the shock is extracted from the solution and is compared with the two-dimensional and the exact solutions in Figure 9. 2D solution is obtained from an adaptive solver [20]. Adaptive 2D grid is at least two levels finer than the 3D grid and has a higher quality. Since the wedge flow is essentially a 2D flow, the difference between 2D and 3D solutions is acceptable.

Steady state solution of the supersonic flow with freestream Mach number of 2 with 10 degrees angle of attack around a cone with 10 degrees half-angle is the second test problem. This is a three-dimensional problem. Mach contours on the cone surface are shown in Figure 10. Pressure coefficient on the cone surface is compared with Shankar [21] solution in Figure 11. As expected, the pressure reaches to its minimum on the leeward side of the cone and then increases. In the current solution this minimum is detected almost accurately. Weak shock waves are not well developed on the leeward side, and strong windward shocks dominate the physics of this problem. Indeed the

higher order schemes and adaptation can lead to better results in the solution.

References

[1] Thompson J F, Soni B K, Weatherill N P. *Handbook of grid generation*. 1st edition, CRC Press, 1999.

[2] Barth T J. On unstructured grids and solvers. *Von Karman Series 1990-03 in Computational Fluid Dynamics*, pp 1-65, 1990.

[3] Khosravi R. Solution of compressible flow equations on 3D unstructured grids. *MS Thesis of Mechanical Engineering*, Department of Mechanical Engineering, Sharif University of Technology, 1997.

[4] Bowyer A. Computing Dirichlet tessellations. *The Computer Journal*, Vol. 24, No. 2, pp 162-166, 1981.

[5] Watson D F. Computing n-dimensional Delaunay tessellation with application to Voroni polytopes. *The Computer Journal*, Vol. 24, No. 2, pp 167-172, 1981.

[6] Frey W. Selective refinement: a new strategy for automatic node placement in graded triangular meshes. *International Journal for Numerical Methods in Engineering*, Vol. 24, pp 2183-2200, 1987.

[7] Schroeder W J, Shephard M S. Geometry-based fully automatic mesh generation and Delaunay triangulation. *International Journal for Numerical Methods in Engineering*, Vol. 26, pp 2503-2515, 1988.

[8] Muller J D, Roe P L, Deconinck H. A frontal approach for internal node generation in Delaunay triangulations. *International Journal for Numerical Methods in Fluids*, 1992.

[9] Weatherill N P, Hassan O. Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. *International Journal for Numerical Methods in Engineering*, Vol. 37, pp 2005-2039, 1994.

[10] Golias N A, Tsiboukis T D. An approach to refining three-dimensional tetrahedral based on Delaunay transformations. *International Journal for Numerical Methods in Engineering*, Vol. 37, pp 793-812, 1994.

[11] Macrum D L, Weatherill N P. Unstructured grid generation using iterative point insertion and local reconnection. *AIAA Journal*, Vol. 33, No. 9, pp 1619-1625, 1995.

[12] Zheng Y, Lewis R W, Gethin D T. Three-dimensional unstructured mesh generation part 1-3. *Computer Methods in Applied Mechanics and Engineering*, Vol. 134, pp 249-268, 1996.

[13] George P L, Hermeline F. Delaunay's mesh of a convex polyhedron in dimension d. Application to arbitrary polyhedra. *International Journal for Numerical Methods in Engineering*, Vol. 33, pp 975-995, 1997.

[14] Rebay S. Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm. *Journal of Computational Physics*, Vol. 106, pp 125-138, 1993.

[15] Pirzadeh Sh. Three-dimensional unstructured viscous grids by the advancing-layers method. *AIAA Journal*, Vol. 34, No. 1, pp 43-49, 1996.

[16] Schroeder W J, Shephard M S. A combined octree/Delaunay method for fully automatic 3-D mesh generation. *International Journal for Numerical Methods in Engineering*, Vol. 29, pp 37-55, 1990.

[17] Roe P L. Approximate Riemann solvers, parameter vectors and difference schemes. *Journal of Computational Physics*, Vol. 83, pp 125-138, 1981.

[18] Hirsch C. *Numerical computation of internal and external flows*. 2nd edition, John Wiley & Sons, 1988.

[19] Frink N T. Upwind scheme for solving the Euler equations on unstructured tetrahedral meshes. *AIAA Journal*, Vol. 30, No. 1, pp 70-77, 1992.

[20] Lessani B. Unsteady solution of two-dimensional compressible flow on unstructured grids. *MS Thesis of Mechanical Engineering*, Department of Mechanical Engineering, Sharif University of Technology, 1997.

[21] Shankar V. Treatment of conical and nonconical supersonic flows by an implicit marching scheme applied to the full potential equation. *Proc ASME/AIAA conference on Computers in Flow Predictions and Fluid Dynamics Experiments*, Washington D.C., pp 163-170, 1981.

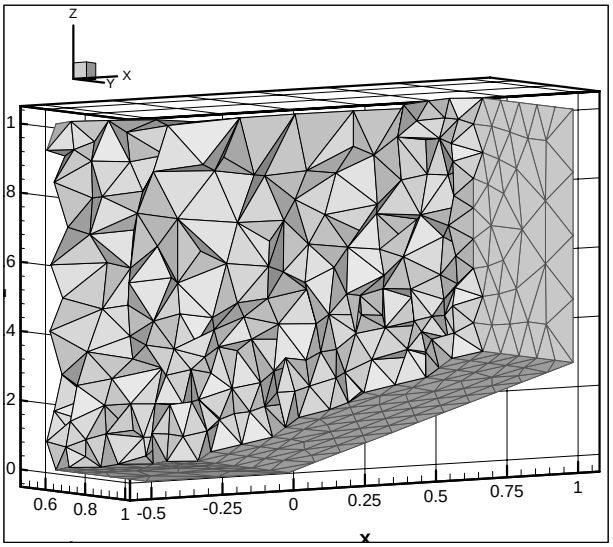

Figure 4 Wedge mesh cross section

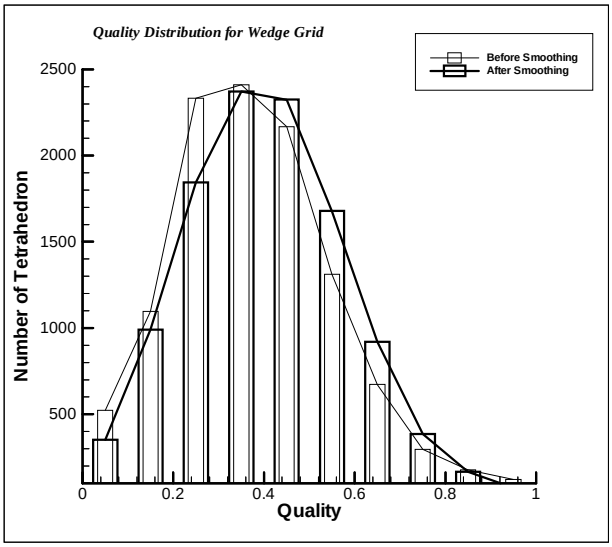

Figure 5 Quality distributions for wedge grid

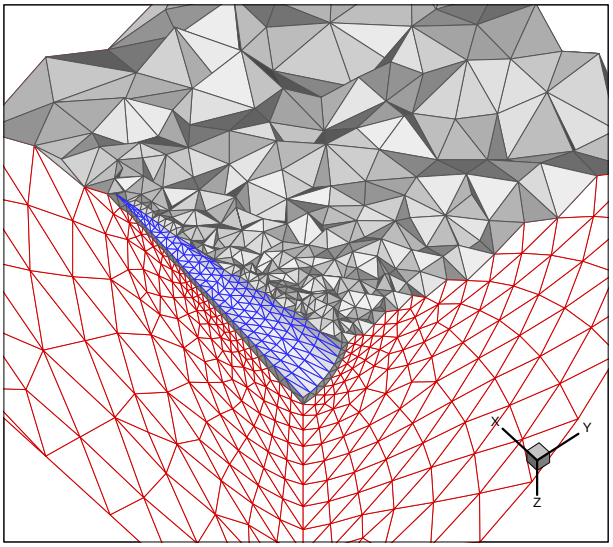

Figure 6 Cross section of mesh over cone

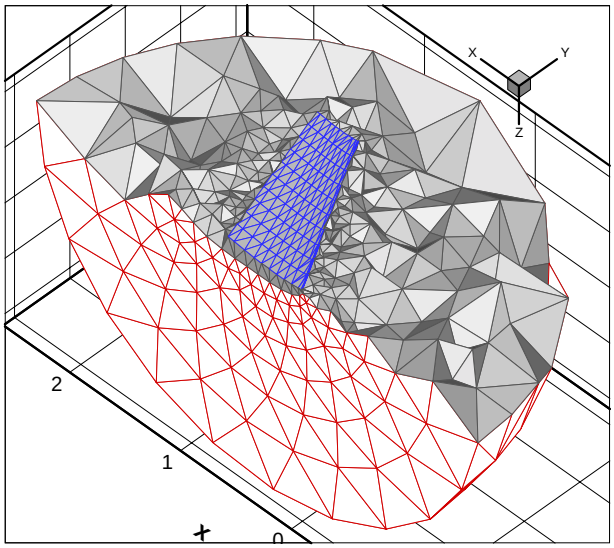

Figure 7 Mesh around ONERA M6 wing

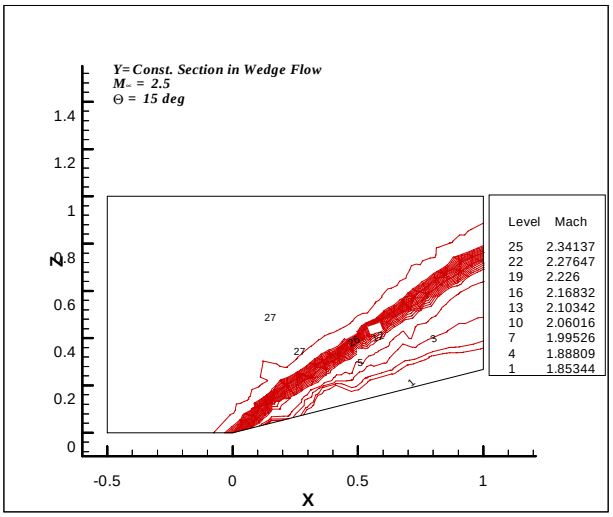

Figure 8 Mach contours in a section of wedge

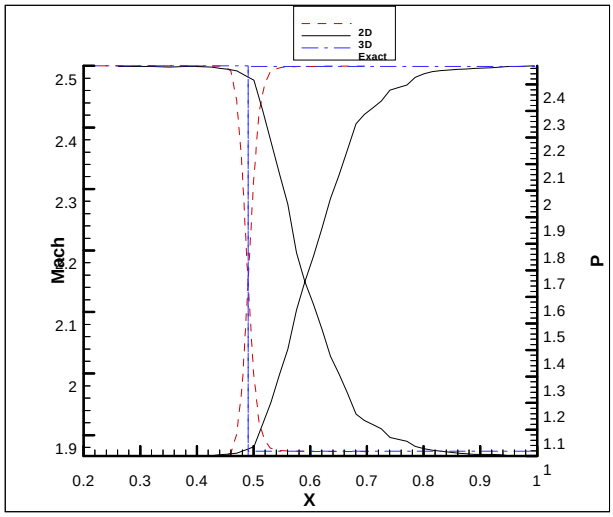

Figure 9 Comparison of 3D, 2D and exact solutions

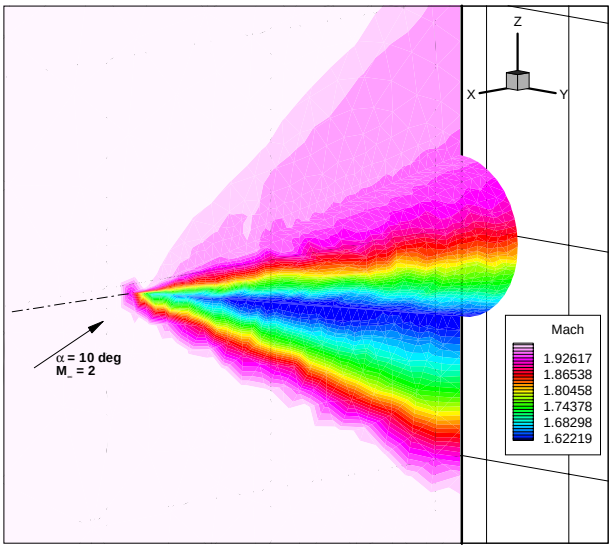

Figure 10 Mach contours on cone surface

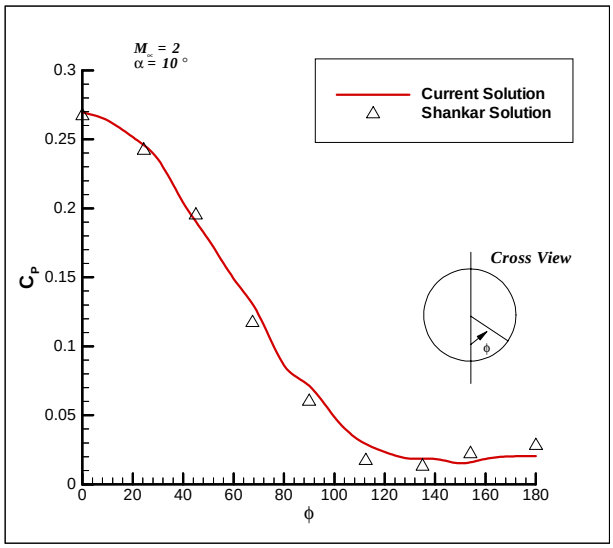

Figure 11 Circumferential pressure coefficient around cone

Table 1 Quality comparison of meshes before and after smoothing

Mesh	Q_m		Q_j		Q_{min}		Q_{max}	
	Before	After	Before	After	Before	After	Before	After
Wedge	0.3815	0.4071	0.0583	0.0636	0.0000	0.0001	0.9978	0.9958
Cone	0.3731	0.4010	0.2032	0.2725	0.0010	0.0044	0.9924	0.9916
M6 wing	0.3484	0.3770	0.1606	0.2085	0.0011	0.0012	0.9913	0.9781