Unstructured grid generation over airfoil and other domains

Jayanarayanan Sitaraman
Dept of Aerospace Engg
University of Maryland at College Park

Abstract

Unstructured grid generation is a very powerful tool for building meshes suitable for structural or fluid flow analysis over complex geometries. This report investigates the design and implementation of a 2-D unstructured mesh generator capable of tesselating non-convex multiply connected domains in to near equilateral triangles. Adapting the grids near the body surfaces to form high aspect ratio triangles capable of capturing boundary layer effects is also discussed.

Introduction

Unstructured grid generation techniques have gained popularity over the last fifteen years. Ability to treat complex geometries with almost the same ease as simple ones is one of the chief reasons for their popularity. Few of the widely used unstructured mesh generation techniques are the following:

Quadtree, Octree based grid generations

This is one of the earliest methods used for generating unstructured grids. It involves use of quads and octrees in two and three dimensions respectively. An initial quad is formed which is large enough to cover the entire domain. This is further subdivided recursively until all leaf quads are no larger than the local value of the pre-dictated element size distribution function. At the physical boundaries care is taken to make the grids boundary conforming. This is achieved by moving the closest quadtree vertices to coincide with the boundaries.

Advancing Front Methods

The boundaries of the domain provided are divided into segments using a element size distribution function. These segments comprise the front which advances in space to form the triangles. Once this is done, one edge is selected to form a new triangle by joining the two ends of the current edge to a newly created point, or to an existing point in the front. The current edge is then removed from the front and the new front is constructed adding one or both edges of the newly formed triangle, depending on their visibility. Alternatively, the whole boundary can be advanced along the normal direction to form the new front and triangles can be formed to fill the regions between the old and new front. This strategy is adopted for generating triangles in the boundary layer and

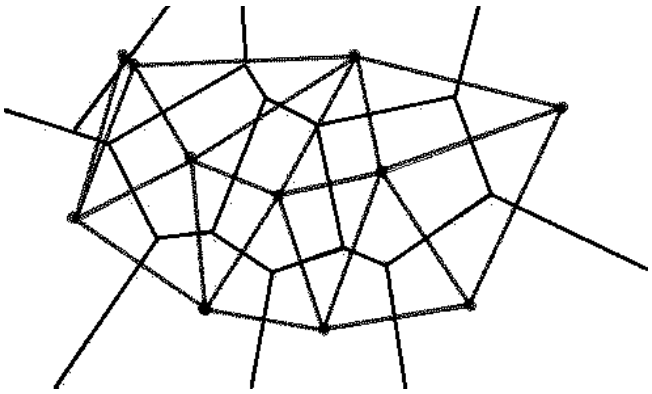

Figure 1: Delaunay triangulation and Voronoi diagram

is discussed in detail later in the paper.

Delaunay Triangulation

The Delaunay triangulation is the geometric dual of the Voronoi tesselation in 2 dimensions. A Voronoi tesselation for a given distribution of points is the collection of geometric objects, each of which represents a region which is closer to a particular point in the domain than any other (fig 1.). Delaunay trianglulation is therefore characterised by certain special properties like empty-circumcircle property and max-min angle property. The empty circumcircle property means that there should not be any points within the cirumcircle of any triangle formed, while the max-min angle property signifies that the Delaunay triangulation for a given set of points is the one which gives the largest minimum angle for all triangular elements formed. These properties make the Delaunay triangulation produce well-shaped elements. The present design incorporates the Delaunay criterion to generate triangles. The algorithm is discussed in detail in the following section.

Algorithm for Constrained Delaunay Triangulation and Refinement

Most computational domains have prescribed edges. A Delaunay triangulation does not guarantee the creation of these edges. So it needs to be forced by the algorithm. There has been several approaches to tackle the problem. The present algorithm closely follows that due to Raveendra[3]. The given domain is initially split into boundary triangles i.e those which have only points of the initially supplied domain as nodes. These triangles are made Delaunay (or almost Delaunay) using Lawsons[2] edge-flip algorithm. The starting triangulation thus obtained is refined using a selective refinement strategy[1]. Points are inserted at the centroids instead of circumcenters to decrease computational cost. A smoothing is performed to improve the quality of the triangulations. These steps are discussed in detail below.

Boundary triangulation

The minimum necessary information that is required of a domain is the (x,y) location of the boundary nodes and their connectivity. It is also necessary that the edges be directed edges so that their either sides are clearly distinguishable. It is assumed that the domain under consideration is always on the left of a given edge. This necessitates that the outer and inner boundaries be specified in the anti-clockwise direction and the clockwise direction (fig 2.) respectively. The domain is defined by a collection of such directed edges. So the connectivity of the domain is invisible and hence the algorithm treats both singly and multiply connected domains with equal ease. The steps followed are given below

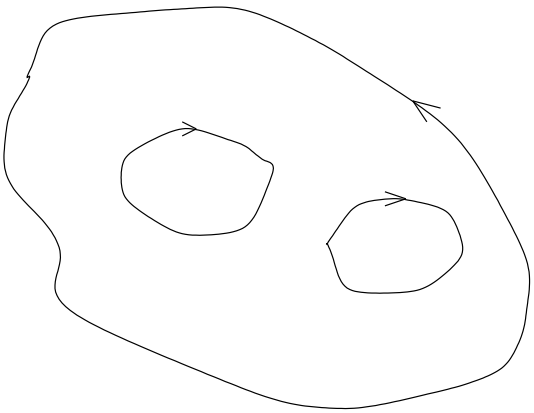

Figure 2: Boundary specification

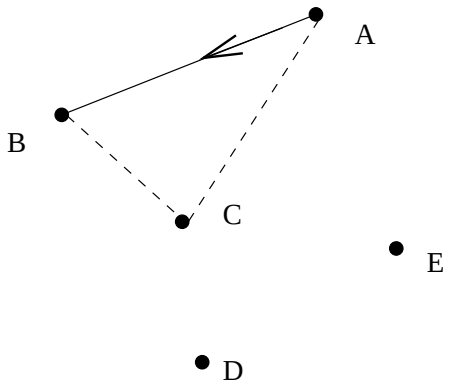

Figure 3: Left points - sign of $\overrightarrow{CA} \times \overrightarrow{CB}$ determines whether the points are to the left elliptic distance = AC+BC

The algorithm

1. *For the edge at the top of the stack find the closest point to its left and construct a triangle.*
2. *Find whether the formed triangle encroaches any of the existing edges of the domain, if yes go back to step 1 and use the next closest point to this edge.*
3. *The formation of the present triangle is feasible. So form the triangle and add it to the formed triangle list.*
4. *Construct a new domain by cutting off the new triangle formed from the present domain.*
5. *follow steps 1,2,3 and 4 recursively until all boundary nodes are nodes of formed triangles.*

The closest left point is found using the following method. First a list of points to the left of a given edge is made. Whether or not a point lies on the left of a edge or not is determined by the sign of the cross product of the vectors connecting the present node and end points of the directed edge taken in order (fig 3.). A positive sign of the cross product indicates the node is on the left. The parameter which defines the closeness is taken as the elliptic distance or in other words the sum of the distances of the node from the end points of the edge. Steps 1 and 2 together determine the closest node which when formed triangle does not encroach any of the edges of the domain. There is a little bit of variation from Raveendra's algorithm in step 4. The new edges formed are included at the top of the stack while constructing the new domain whereas Raveendra's algorithm introduces these edges at the bottom of the stack. This variation makes step 2 optional for most domains (of-course for arbitarily convoluted domains it is necessary) whereas Raveendra's algorithm requires it almost compulsorily. It is to be mentioned that Step 2 is the one involving the highest computational cost as it requires quite a lot of floating point arithmetic to determine if two edges intersect and about 2n such intersection checks to determine whether a triangle is feasible to form (n=number of edges).

Making boundary triangulation Delaunay

Actually it is not always possible to make the existing boundary triangulation completely

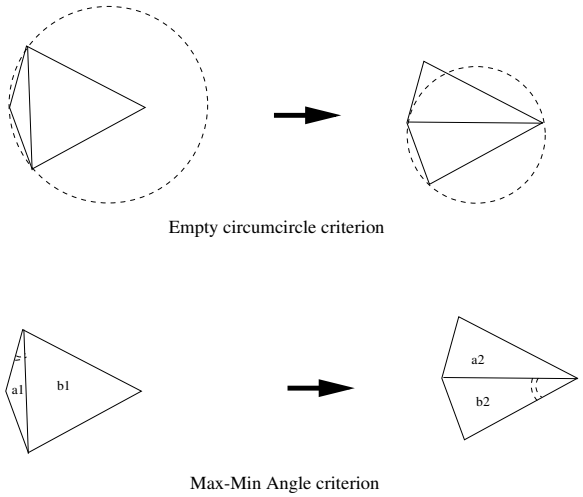

Figure 4: Lawson edge-flip algorithm

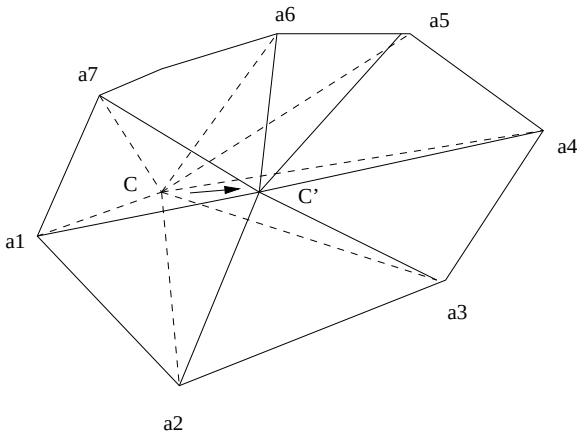

Figure 5: Laplacian smoothing

Delaunay without introducing new points on the boundary edges. Shewchuck[5] discusses this problem. But the domain can be made approximately Delaunay by following Lawsons[2] edge flip algorithm (fig 4.). It was stated earlier that Delaunay triangulation produces triangles whose circumcircles does not contain any other nodes. So for each set of neighbouring triangles if the cirumcircle of either contains the other triangle the Delaunay criterion is violated. If such condition arises the common edge of the triangles is flipped around connecting the previously non-connected vertices. This process is recursively continued until no more flips can be made (ideally this is true, but in actual case it is performed a finite number of times as the domain can't be made completely Delaunay).

Refinement and smoothing

Once the starting triangulation is known, internal nodes can be introduced in various ways. New nodes are inserted at the centroids of the existing triangles. These are connected to the vertices of the parent triangle to form three new triangles. The triangles which are refined are removed from the list. So there is a net gain of two triangles for each node inserted. The triangulation is made Delaunay by sweeping through the triangles and applying Lawson's edge flip algorithm. A Laplacian smoothing is performed to improve the quality of the triangles[6]. Each node is moved to the centroid of the polygon formed by the nodes connected to it (fig 5.). Lawson's edge-flip is again performed on any triangle which is found not Delaunay after the smoothing. The smoothing and edge-flipping is continued until the whole triangulation is Delaunay. This completes the first stage of refinement. The next stage uses this as the starting triangulation.

Termination condition

A user specified parameter called Node Spacing Function is provided for all the boundary nodes. Physically the NSF for any node is the shortest distance that can be possible between that node and the node closest to it. The NSF of the internal nodes are interpolated from that of the boundary values. A new node is inserted at the centroid of a triangle only if the distances of this node from the vertices of the parent triangle and the centroids of the neighbouring triangles are greater than their respective NSF values. The triangulation is termed complete

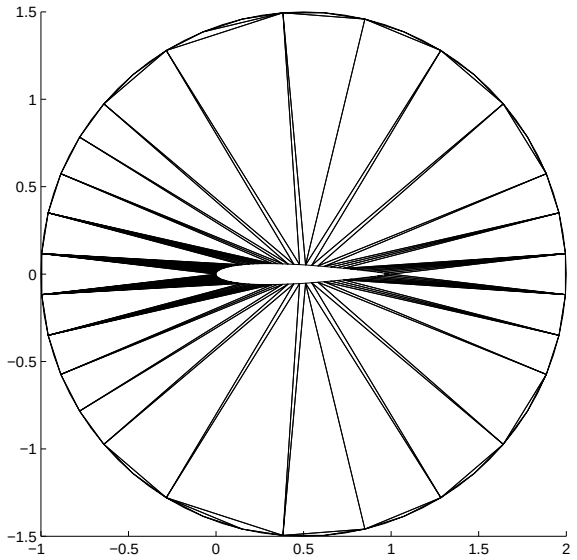

Figure 6: Boundary triangulation over NACA 0012 airfoil with circular outer boundary

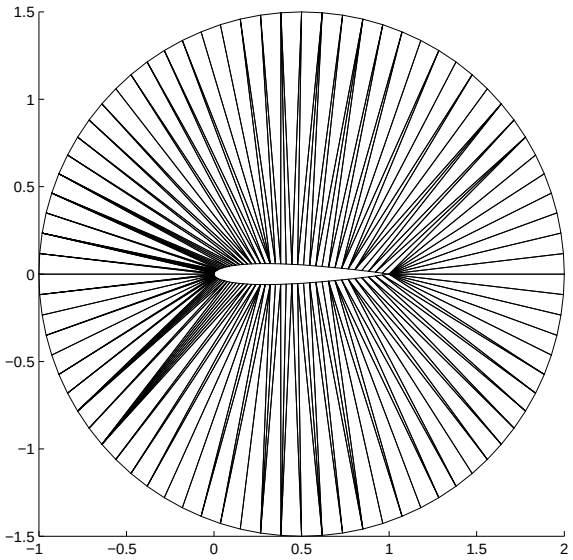

Figure 7: After performing recursive lawsons edge-flip to make Delaunay

if no more new nodes can be inserted and no more edge-flips could be made. The use of user-specified NSF facilitates easier grid clustering and control. The sequence of formation of triangles in a domain consisting of a NACA 0012 airfoil and a circle of diameter 3 units around it is illustrated in figures 6 to 11.

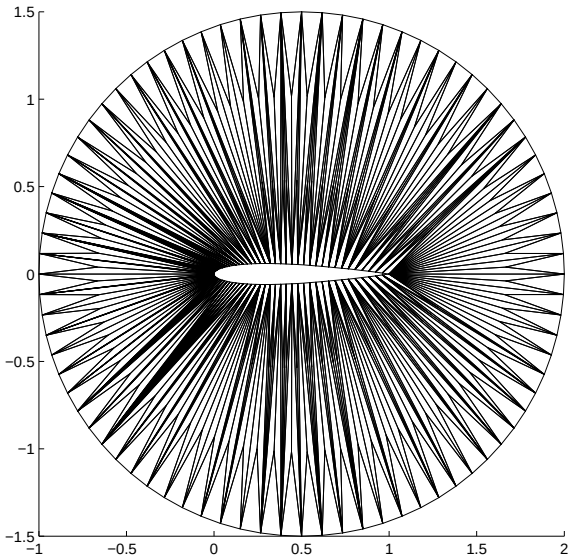

Figure 8: 1st refinement - after insertion of points at the centroids

Generating high aspect ratio triangles in the boundary layer

Resolving boundary layers in Navier-Stokes calculation using thin-layer approximations require cell aspect ratios of around 1:10000. The algorithm discussed earlier produces equiangular triangulations. So it is not possible to form cells of high aspect ratios. Cells in the boundary layer could be made small enough by controlling the Node Spacing Function, but this would lead to a enormous increase in number of triangles formed due to the gradual gradient in the areas of adjacent cells. Hence a advancing front method is used to create triangles in the boundary layer (fig .12). The number of layers required in boundary layer, the thickness of the layer closest to the surface and the stretching factor are user specified parame-

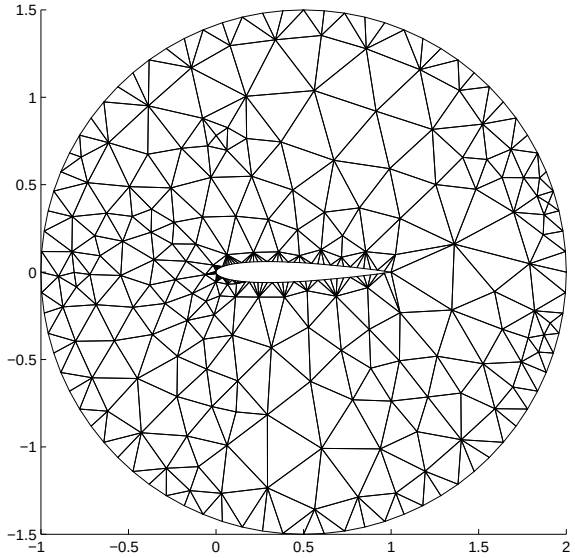

Figure 9: 1st refinement - after performing Laplacian smoothing and edge-flip to make Delaunay

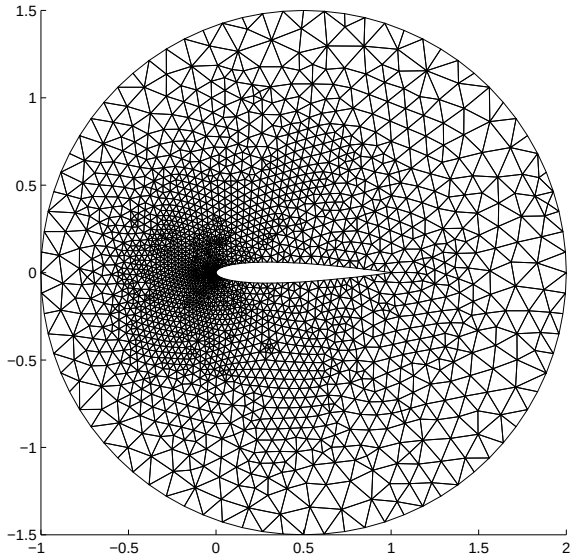

Figure 10: Final triangulation - 4324 triangles, 2242 nodes

ters. The outermost layer produced by the advancing front method is used as the input to the Delaunay grid generator (fig 13.). The triangulations produced are merged matching the boundaries where they meet (fig 14.).

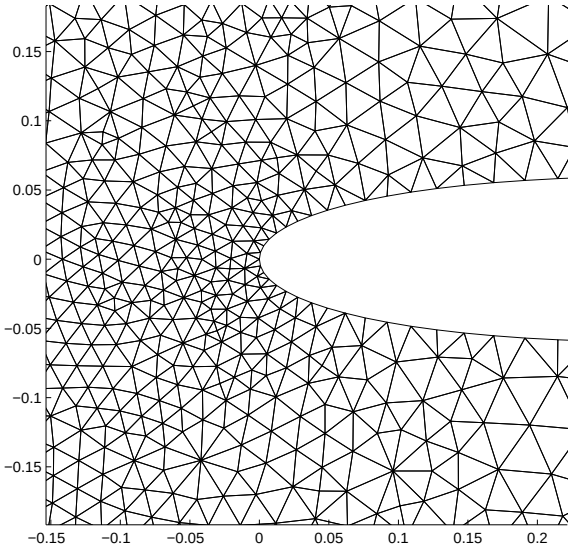

Figure 11: Final triangulation-zoomed near the leading edge

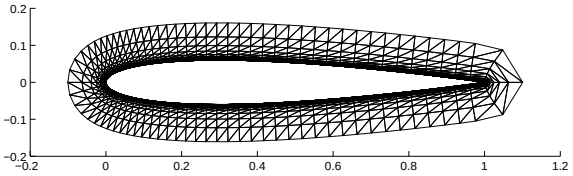

Figure 12: Advancing front triangulation alone - 3278 triangles, thickness of first layer=5×10^{-6}, number of layers=20, stretching factor=1.6

Advancing front algorithm

Similar to the Delaunay algorithm the domain is defined by a collection of edges. The inner boundary alone is used for the advancing front method. For each edge the end nodes are moved in the direction of the surface normal. Quads can be formed between the newly formed edges and their parent edges. Each of these quads are split in to triangles. The domain formed of the all the newly formed edges is taken as the new front and the process is repeated. 2n (n=number of edges)triangles are formed at each step. These steps are repeated until the user-specified number of layers are produced.

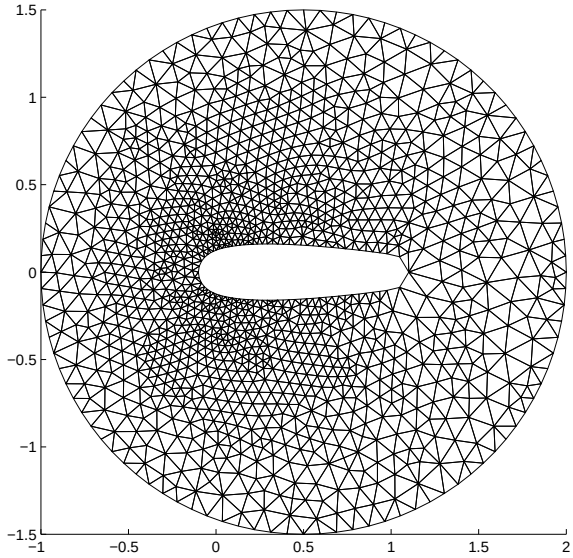

Figure 13: Delaunay triangulation alone - 2538 triangles

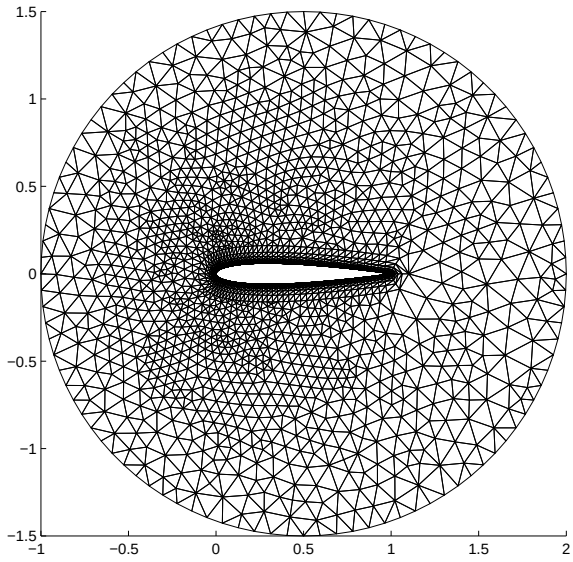

Figure 14: Both triangulations after merging and boundary matching -5816 triangles, 2998 nodes

Treatment of trailing edge

Edges which are at high angles to each other causes highly oblique triangles to be formed. This is especially true near the trailing edge. Such triangles would lead to problems in the flow solver. So such triangles are split into two lesser oblique triangles. This approach was found to produce acceptable results (fig 15. and fig 16.).

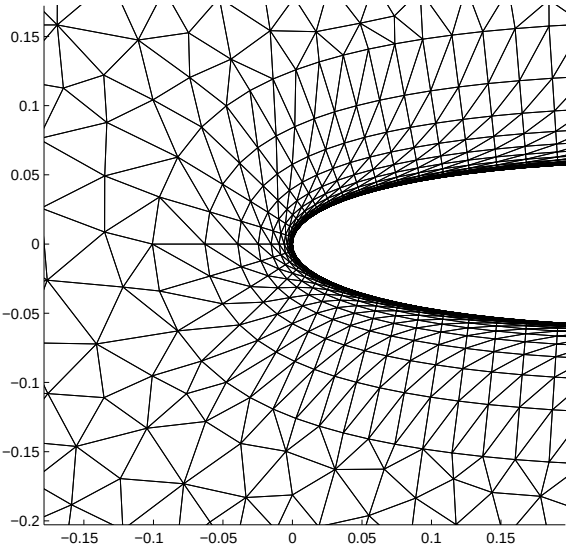

Figure 15: Leading edge

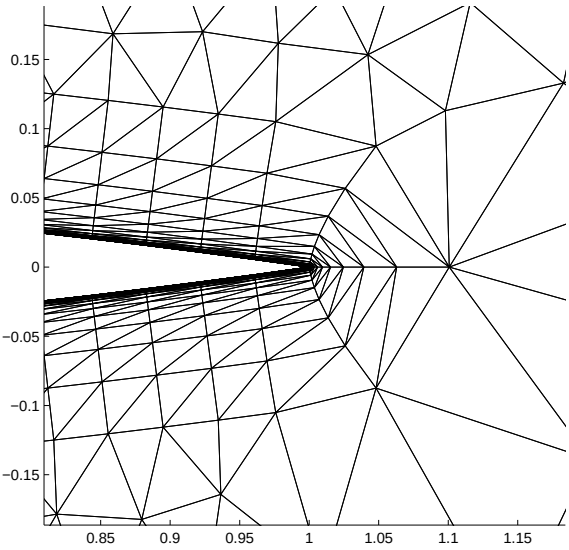

Figure 16: Trailing edge

Implementation

The program was coded in C and tested in Ultra-sparc and Dec work stations. A triangle based data structure was used instead

of a commonly used edge based data structure. Each triangle stores information of its nodes and immediete neighbours in order. This is similar to Shewchuck's[5] approach. For the boundary triangulation a stack of edge data structures was used to define the domain. Also a connectivity stack was maintained for each node. The stack of a particular node contained information of all the nodes connected to the that node. This stack was used to speed up the Laplacian smoothing progress. Advancing front algorithm made use of the same data structures as those used by the Delaunay triangulation algorithm. All nodes and triangles were indexed starting from zero. While merging the triangulation, a index matching and location matching was performed to preserve boundary integrity. A numerical tolerance limit of $\epsilon = 10^{-5}$ was used for the left check and the intersection check.

Results and discussion

The program was found to give well shaped triangles for a range of tested domains. As the problems of interest were mostly fluid flow problems multiply connected domains were tested. It was found that the use of intersection check was not necessary in almost all cases. The results of the grid generator with and without the boundary layer grids is given in figures 10 and 14. It was found that smoothing improves the quality of triangles considerably. Also splitting triangles with high oblique angles in the advancing front method was found to give better triangulations. The triangulation of a domain around NACA 0012 airfoil is illustrated in figures 6 to 16. Grids for more complicated domains are illustrated in figures 17 and 18.

Extension to 3-Dimensions

Direct extension to 3-D is possible, but does not guarantee robustness as it is not always possible to form a Boundary Tetrahedralisation for a prescribed set of boundary faces. Also it is a NP-hard problem to determine whether or not a polyhedron can be split into boundary tetrahedrons. The boundary information need to be specified as triangular facets on the surfaces. This necessitates the need of a surface grid generation algorithm which could be achieved by mapping triangles created in 2-D to patches on the surface[6].

Limitations

The most vulnerable limitation is memory. The highest number of triangles the grid generator could produce was around 50,000. Maintaining connectivity stacks for each node to facilitate faster Laplacian smoothing is main reason of the memory crunch. This could be improved by implementing Ruppert's[4] incremental insertion algorithm. Also a time based efficiency study was not conducted. Approximately 8,000 triangles were produced in 10 seconds on a ultra-sparc machine. Most of the time was spent in file I/O, boundary triangulation and smoothing. There was no robustness problem which could be identified yet. But without intersection check there is a fair chance of failiure for highly convoluted domains.

Conclusion

A fully automated simple algorithm to generate Delaunay triangulaion for convex/non-convex multiply connected planar domains is presented. Coupling advancing front method with the Delunay triangulation method to produce high aspect ratio triangles in the

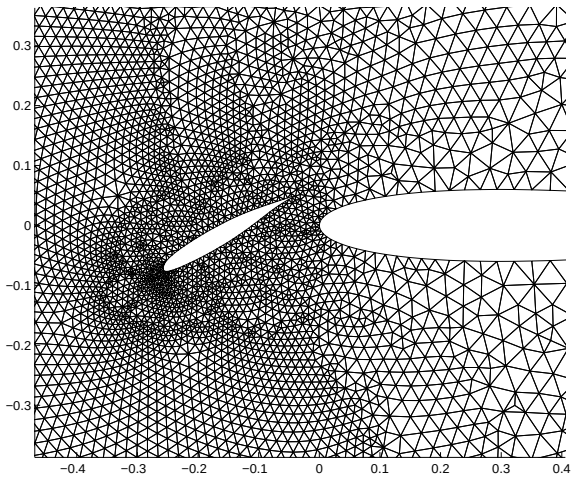

Figure 17: Airfoil with slat

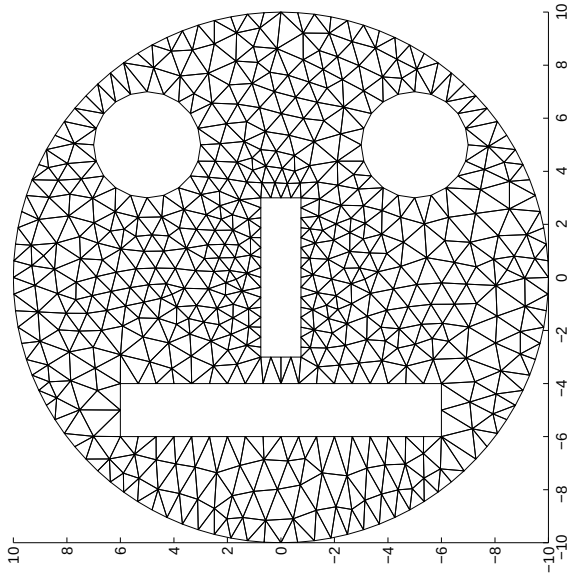

Figure 18: Smile!

boundary layer is also discussed. Effectiveness of the algorithm is illustrated using examples of triangulations over a NACA 0012 airfoil.

References

1. Frey.,W.H. "Selective refinement: a new strategy for automatic node placement in graded triangular meshes", *International Journal for Numerical Methods in Engineering*, Vol.24, 2183-2200 (1987).

2. Lawson.,C.L. Software for C^1 Surface Interpolation. *Mathematical Software III* (John R. Rice, editor), pages 161-194. Academic Press, New York, 1977.

3. Raveendra, V.V.S., Subramanian, G. and Kamath, M.G. "Robust Boundary Triangulation and Delaunay Triangulation of Arbitrary Planar Domains," *International Journal for Numerical Methods in Engineering*, Vol. 37, No. 10, 1994, pp. 1779-1789.

4. Ruppert, Jim, A. Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. *Journal of Algorithms*, 18(3):548-585, May 1995.

5. Shewchuck, Jonathan, Richard. "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator",*First Workshop on Applied Computational Geometry* (Philadelphia, Pennsylvania), pages 124-133, Association for Computing Machinery, May 1996. *http://www.cs.umd.edu/ quake-papers/triangle.ps*

6. Sitaraman, Jayanarayanan. "Modified Panel methods for estimation of viscous Drag",*Undergraduate Thesis* ,IIT Madras, 1998.