Proceedings of MARTEC 2010
The International Conference on Marine Technology
11-12 December 2010, BUET, Dhaka, Bangladesh

A PRACTICAL APPROACH TO TRIANGULATION OF PLANAR SURFACE

Md. Shahidul Islam, Naem-Ibne-Rahman and Md. Mezbah Uddin

Department of Naval Architecture and Marine Engineering,
Bangladesh University of Engineering and Technology, Dhaka.
E-mail: shahid113@gmail.com, naem_nir@hotmail.com

ABSTRACT

A triangular mesh generation algorithm is presented in this paper. Based on this procedure a fully automatic object oriented program in C++ language is developed for any arbitrary 2D geometry. The program generates unstructured triangular surface mesh which can be used for finite element analysis. The program gives mesh output in a script file format for viewing in AutoCAD. Importance is given to the quality of the triangles generated so that better results can be achieved by the finite element analysis. A very practical approach has been taken in this regard. A number of example meshes are presented to show the effectiveness of the algorithm and program.

Key words: Triangular mesh generation, C++, Object oriented programming, Finite Element Method, Unstructured surface mesh.

1. INTRODUCTION

Finite element method (FEM) is a powerful tool for the numerical solution for a wide range of engineering problems. Applications range from deformation and stress analysis of automotive, aircraft, building and bridge structures to field analysis of heat flux, fluid flow, magnetic flux, seepage and other flow problems. FEM allows detailed visualization of where structures bend or twist, and indicates the distribution of stresses and displacements. FEM software provides a wide range of simulation options for controlling the complexity of both modeling and analysis of a system. FEM allows entire designs to be constructed, refined, and optimized before the design is manufactured.

In this method of analysis a complex region defining a continuum is discretized into simple geometric shapes called finite elements. The material properties and the governing relationships are considered over these elements. An assembly process, duly considering the loading and constraints, results in a set of equations. Solution of these equations gives the approximate behavior of the continuum.

The current research basically develops a program to discretize a problem domain into triangular shaped finite elements. Element generation is commonly known as mesh generation. Triangles have been chosen as the mode of discretization simply because it is the most elementary geometrical shape when it comes to mesh generation. Again, if the number of total triangles is even, the mesh can be converted in to quadrilateral mesh.

Automatic unstructured mesh generation is a relatively new field. Within its short life span it has tremendous advancement in many diverse fields. There are a number of triangulation techniques. Of these the Delaunay method [1] is one of the most popular techniques of triangular mesh generation. The Delaunay criterion in itself is not an algorithm for generating a mesh. It merely provides the criteria for which to connect a set of existing points in space. As such it is necessary to provide a method for generating node locations within the geometry. A typical approach is to first mesh the boundary of the geometry to provide an initial set of nodes/points. The domain bounded by boundary nodes is then triangulated according to the Delaunay criterion which involves Voroni diagram [2]. In two dimensions the Voroni diagram of a group of vertices divides two-dimensional space into regions bound by straight line segments. Each region or Voroni cell belongs to one vertex. Although the Delaunay criterion has been known for many years, it was not until the work of Charles Lawson [3] and Dave Watson [4] that the criterion was utilized for developing algorithms to triangulate a set of vertices. The criterion was later used in developing meshing algorithms by Timothy Baker [5] at Princeton, Nigel Weatherill [6] at Swansea, Paul-Louis George [7] at INRIA among others.

Another very popular triangular mesh generation algorithms is the advancing front, or moving front

method. It was first presented by Lo in 1985 [8]. He presented a new mesh generation algorithm that initially defines the boundary of the domain by a set of line segments, the initial advancing front. All nodes are generated within the domain bounded by the initial advancing front and valid triangular elements are then formed from the line segments and interior nodes. Each element is generated by joining the end nodes of a line segment to a third node with the condition that it does not intersect the advancing front. The front is then updated and the element creation process goes on as long as the front is not empty. Peraire et al [9] presented a modified version of advancing front method in 1987. Most of the subsequent advancing front method research has been based upon Peraire's algorithm.

As the triangulation algorithms discussed above are very complicated, a very simple approach is presented in this paper which can generate good quality triangles. The developed program based on this method requires the boundary nodes of the problem domain in counter clockwise direction as input. The main operation then includes selection of the protruding point, generation of triangles from the boundary nodes, insertion of internal points in primary triangles satisfying the rules of triangulation, generation of triangles from the inserted points, edge swapping and finally the resultant triangles are smoothed using Laplacian smoothing [10] technique. After completing these operations the final resulting mesh is presented in a script file format for viewing in AUTOCAD software. Nodes and points represent same entity in this study.

2. THE PROPOSED TRIANGULATION PROCEDURE

2.1 Properties of Triangulation:

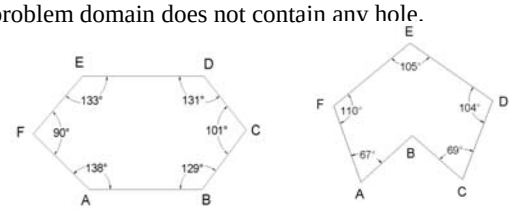
Irrespective of the technique followed for the triangular mesh generation, the resulting triangulation must follow some rules for it to be a correct one. These are rules derived from simple geometry which gives the relation between number of vertices, edges and triangles generated. If the number of triangles, edges, internal edges, boundary points and internal points are represented by $|T|$, $|E|$, $|E_i|$, $|P_b|$ and $|P_i|$ respectively, then these properties are:

$$|T| = 2\,|P_i| + |P_b| - 2$$
$$|E| = 3\,|P_i| + 2\,|P_b| - 3$$
$$|E_i| = 3\,|P_i| + |P_b| - 3$$

2.2 The Problem Domain

The triangulation process starts with the definition of the problem domain. The boundary nodes of the domain are given in the anti clockwise direction. Thus by joining these boundary nodes the geometry of the problem domain will be established.

Since the geometry is defined by boundary points and lines joining those points, the resultant domain can be of two types: (1) convex region and (2) concave region (Figure 1). A convex region is one in which all the interior angles are not greater than 180°. On the other hand a region is said to be concave if any of its interior angles are greater than 180°. This program is capable of triangulating both the convex and concave regions. For our current analysis the problem domain does not contain any hole.



(a) Convex region (b) concave region

Figure 1. Problem domain

2.3 Steps of Triangulation

Our presented triangulation has five steps and these are as follows:

Initial triangle generation
Internal point insertion
Further triangulation
Edge-swapping
Laplacian smoothing

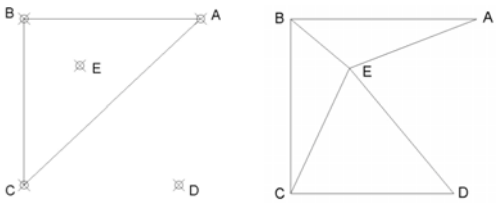
All these steps are described below.

2.3.1. Initial Triangle Generation

In this step, initial triangles are formed from the given boundary nodes. **If there is 'n' number of boundary nodes then the number of initial triangles will be 'n-2'.** In order to understand the initial triangle generation process first it is important to be familiar with the concept of 'the Proper triangle' and 'the Protruding Point'.

Proper Triangle:

The vertices of any initial triangle will be three of the input boundary nodes. A proper triangle is defined as one that does not contain any other boundary nodes except for its three vertices. The concept is illustrated in Figure 2.



(a) Improper triangle (b) Proper triangles

Figure 2. Proper and improper triangles

It is worth mentioning that improper triangles can only be produced in concave domains.

Protruding Point:

Protruding points are defined as the points that produce proper triangles within the limits of a defined domain. If P_i is a protruding point and P_{i-1} and P_{i+1} are points before and after P_i respectively, then the triangle produced by P_i is $T_i = \{ P_i, P_{i-1}, P_{i+1} \}$. The condition for a point P_i to be a protruding point is:

The interior angle of the problem domain at P_i must be less than 180°.

T_i must not contain any other boundary node of the problem domain other than P_i, P_{i-1} and P_{i+1}.

The above mentioned conditions determine whether a point is a protruding point or not. Using the proper triangle and protruding point concept the initial triangulation is done. Figure 4 shows the resultant mesh of the region bounded by points A, B, C, D and E.

2.3.2 Internal Point Insertion

The next step in the triangulation process is the internal point insertion. Internal points are important because the initial triangles may be too large or may have too small an angle to work with. Points are generated at the centre of each initial triangle (Figure 3).

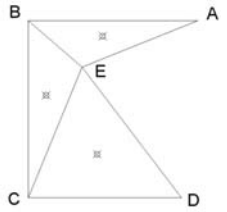


Figure 3. Internal point insertion for the domain shown in Figure 2(b).

2.3.3 Further Triangulation

In this step, the initial triangles are further triangulated. The internal point in each of the initial triangles will divide the parent triangle into three new separate triangles by joining the three vertices to the internal point (Figure 4).

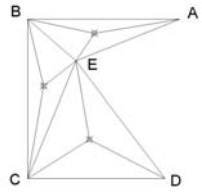


Figure 4. Further triangulation of the domain shown in Figure 2(b)

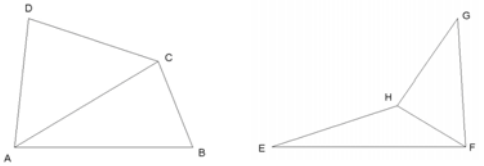
2.3.4 Edge Swapping

Once triangulation for the second time is done, there will be no more division of triangles. But the triangles generated at this point may not look to be of good quality. A quality triangle is one that has edges of almost equal dimension and has internal angles of as close to 60° as possible. The triangles generated so far thus have some room for improvement. For some of the initial triangles it is possible to increase the minimum internal angles by the edge-swapping operation. So far the prominent geometrical shape in this paper has been triangles. But to be able to apply this technique the quadrilateral shape needs to be considered. The technique is explained below.

Quadrilateral Generation:

For edge swapping it is necessary to generate quadrilaterals from the triangles that have been generated so far. This is not a very tough job since every two adjacent triangles form a quadrilateral where the common edge is a diagonal of that quadrilateral.

Even though every two triangle will form a quadrilateral, not all the quadrilaterals present in the domain will need edge swapping. In fact edge swapping will not be possible for quadrilaterals having an internal angle greater than or equal to 180°. Thus only the quadrilaterals having all four internal angles less than 180° will be suited for edge swapping. The unsuitable ones will have to be left unchanged.



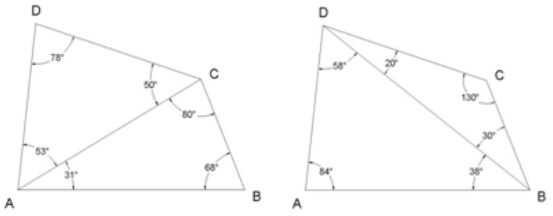
(a) A suitable quadrilateral. (b) Not suitable for edge swapping

Figure 5. Suitability of quadrilaterals for edge swapping

In Figure 5(a), the quadrilateral ABCD is constructed by the triangles ABC and ADC with AC as the common edge. Here all the angles A, B, C and D are less than 180°. Thus ABCD is suitable for the edge swapping operation. In case of Figure 5(b) triangles EFH and FGH makes quadrilateral EFGH. But here the angle H is greater than 180°. So EFGH will be skipped from the edge swapping operation. The procedure of edge swapping is shown by using the quadrilateral in Figure 5(a).

As discussed above, when a quadrilateral is constructed from two adjacent triangles, the common edge acts as a diagonal of that quadrilateral. The Figure 6 (a) shows the six internal angles of the triangles ADC and ABC. It is seen that the smallest angle in this case (X1) is equal to 31°. If the diagonal of the quadrilateral is swapped as in Figure 6(b), the smallest angle X2 becomes 20°. The condition for edge swap is X2 must be greater than X1. The whole idea behind the edge swapping is to increase the

quality of the triangles by increasing the minimum internal angles in the quadrilaterals. Thus edge swapping is not suitable for quadrilateral ABCD.

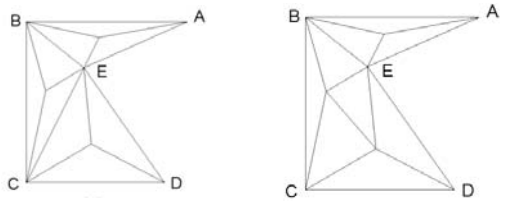


(a) Option 1 (b) Option2

Figure 6. Angle checking for edge swapping

Another important thing that should be kept in mind during programming is that edge-swapping is an iterative process. Each time a diagonal is swapped, the process should be reset for the whole domain again. Because swapping of a single edge changes a considerable number of internal angles in the domain. By iteration process one point will come when none of the edges in the domain will need swapping. That will be the end point of the edge swapping process.

For the domain in Figure 2(b), the result after edge-swapping will be as Figure 7.



(a) Before swapping. (b) After swapping.

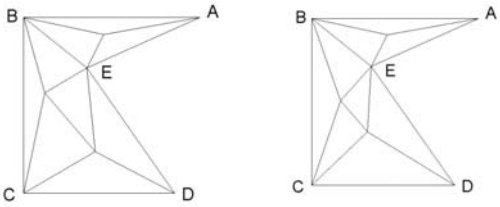
Figure 7. Example of edge swapping

2.3.5 Laplacian Smoothing

After finishing the edge swapping process, the problem domain is now discretized into a finite number of triangles. Thus the domain is ready for finite element analysis. But the quality of the resulting triangles may not still be satisfactory enough and it has been discussed before that, poor quality elements can give poor accuracy in results. Hence one final step is required to complete the triangulation. Mesh smoothing and refinement is a well known method to improve the quality of a mesh. These methods adjust the positions of the points in the mesh while preserving its topology. Laplacian smoothing [10] is the most popular method for node-based mesh smoothing. In an iterative manner, it repositions the points of the mesh by moving each interior node to the geometric centre of its neighbors. It is often used because it is computationally inexpensive and easy to implement.

In this smoothing process the position of a point and its neighboring points are taken into consideration. The neighboring points of a certain point will be the points that share the same edge with that certain point. Each internal node (not the boundary ones) is moved to a new position given by the average of the neighboring nodes.

Figure 8 (a) shows a domain just after triangulation and Figure 8 (b) shows the same domain after the application of smoothing process.



(a) Before smoothing (b) After smoothing

Figure 8. Laplacian smoothing

3. THE PROGRAM

As stated before, the triangulation is done in this study using object oriented programming. Object oriented programming approaches the problem in a very organized manner. As a result it is very easy to construct, maintain and modify for future developments. It is also easier to understand compared to other programming approaches. The programming language used here is C++ as it is a very well known object oriented programming language for its relative simplicity and compactness in nature.

Some attributes of the program are given below:

Source files:

i. `Triangulation.cpp` which contains the function '`main()`'
ii. `Point.cpp`
iii. `Line.cpp`
iv. `Triangle.cpp`
v. `Function.cpp`

Header files:

i. `Point.h`: This contains the class `Point`. The functions of this class mainly define a point by the values of x and y coordinates.
ii. `Line.h`: This header file contains the class `Line`. The functions define lines by the points it contain. There are also functions that can determine the middle point of a certain line and the length of the line etc. There is also a `bool` function that can compare between two lines to say if the two lines are infact the same line or not.
iii. `Triangle.h`: This contains the class `Triangle`. It has functions that define a triangle by its three vertices and the three edges. It also contain function to determine the centre of the triangle and `bool` function that can

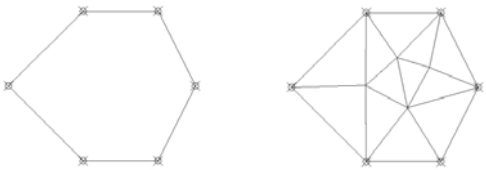
determine the common edge shared by two adjacent triangles.

iv. function.h: This header file includes functions that are needed for other general purposes of the program. The most important of these is a function that determines whether an input domain is a convex or a concave one. Another important function determines whether a certain point lies within a triangle or outside it.

First the input data are typed in the text file input.txt. The program stores these data in a vector named point_vector. Then initial triangles are generated and the protruding points are stored in a_point_vector. Upon completion of the initial triangle generation process all the initial triangles are stored in triangle_vector. Then interior points are inserted in each member of the triangle_vector and the internal points are inserted in updated_point_vector. Next further triangulation is done and the resulting triangles are kept in updated_triangle_vector. Then edge-swapping and smoothing operation are conducted in updated_triangle_vector consequently updating the triangles in it. Finally the results are scripted in the file output.scr in a format that enables the resulting triangulation to be viewed in AutoCAD.

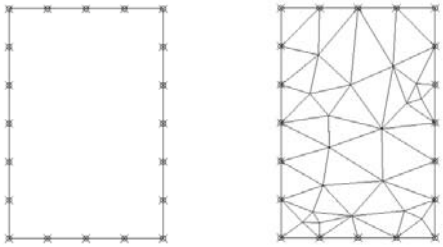
4. RESULTS

In this section some example mesh generated from the given boundary nodes are presented. Model 1, 2 and 3 shows meshing of concave regions. The variations of domain also have variation in point distances, especially in model 3. The resulting mesh shows the effectiveness of the program for these types of domains,



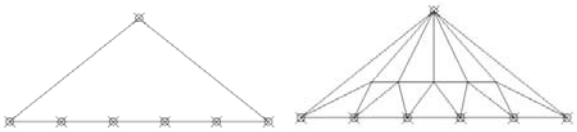
(a) Problem domain (b) Meshed domain

Figure 9. Model 1



(a) Problem domain (b) Meshed domain

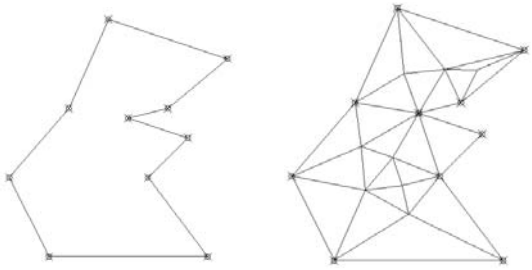
Figure 10. Model 2



(a) Problem domain (b) Meshed domain

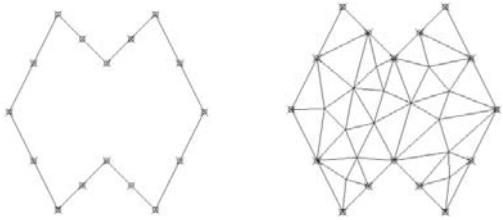
Figure 11. Model 3

Model 4, 5, 6 and 7 represent convex regions with variations so that the capability of the program can be tested for different cases.



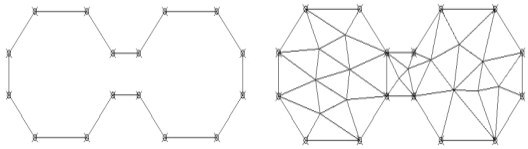
(a) Problem domain (b) Meshed domain

Figure 12. Model 4



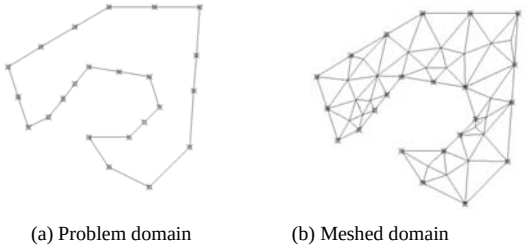
(a) Problem domain (b) Meshed domain

Figure 13. Model 5



(a) Problem domain (b) Meshed domain

Figure 14. Model 6



(a) Problem domain (b) Meshed domain

Figure 15. Model 7

5. CONCLUSION

At present, mesh generation is a very widely studied topic. Lot of researches is taking place on this subject. There are numerous methods and modes of mesh generation being developed recently. This research is just simply one of those efforts. The main aim of this research was to develop a simple mesh generation algorithm in a way that would be easy to understand to everyone who has simply the knowledge of elementary geometry.

This program is a work-in-progress. The aim from the beginning has been to develop our very own FEM software. Even in the triangulation, there is still a lot of room for improvement. For example, after the initial triangulation, if the triangles are first divided into quadrilaterals and then again the quadrilaterals are split into triangles and at last using smoothing may be an alternate procedure. Anyway, a lot of examples are presented in the result section and the quality of the mesh is reasonably good. The future plan is to develop a program that can surface mesh a three dimensional body which later can be discretized into hexahedrons [11]. The solid analysis will then be possible.

REFERENCES

[1**]** Delaunay, Boris, N., "Sur la Sphere" Vide. Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Matematicheskii iEstestvennyka Nauk, vol 7, pp.793-800 (1934) .

[2] Voroni, G., " Nouvelles applications des parameres continues a la theorie des formes quadratiques. Recherches sur les parallelloedres primitives", *Journal Reine angew.Math*, 134, (1908).

[3] Lawson, C. L. ,"Software for C1 Surface Interpolation", *Mathematical Software III*, pp.161-194 (1977).

[4] Watson, David F., "Computing the Delaunay Tesselation with Application to Voronoi Polytopes", *The Computer Journal*, vol 24(2) pp.167-172 (1981)

[5] Baker, Timothy J., "Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation", *Engineering with Computers*, vol 5, pp.161-175 (1989).

[6] Weatherill, N. P. and 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).

[7] George, P. L., Hecht, F. and Saltel, E., "Automatic Mesh Generator with Specified Boundary", *Computer Methods in Applied Mechanics and Engineering*, North-Holland, vol 92, pp.269-288 (1991).

[8] Lo, S.H.,"A new mesh generation scheme for arbitrary planar domains", *International Journal for Numerical Methods in Engineering*, 21(8): 1403-1426, 1985.

[9] Peraire, J., Peiro, J, Formaggia, L., Morgan, K., Zienkiewicz, O.C., "Adaptive remeshing for compressive flow computations", *Journal of Computational Physics*, 72(2): 449-466,1987.

[10] Hansbo, P., "Generalized Laplacian smoothing of unstructured grids", *Communications in numerical methods in engineering*, vol. 11, 455-464 (1995).

[11**]** Islam, Md. Shahidul, "Whisker weaving based plastering algorithm-A new approach to hexahedral mesh generation", *Doctor of Engineering Dissertation*, Yokohama National University (2005).