Chapter 4

Algorithms Implemented

Mesh generation using any algorithm involves a large amount of geometric computation, for example, differentiating points inside the domain or outside the domain, finding if two line segments intersect each other, orienting line segments with respect to a point, finding the distance of a line segment from a point, etc. Development of the finite element mesh using these techniques requires performing these geometric computations at each step, for node and element generation. The WoodFrameMesh program implements the advancing front technique for triangular element generation, as discussed by Lee and Hobbs (1999), and a very simple quadrilateral mesh generation method, developed by the author for quadrilateral domains. This chapter presents these algorithms implemented in the WoodFrameMesh program. Detailed graphical figures are presented at several places for better understanding of the algorithms.

4.1 Triangular Mesh Generation - Advancing Front Technique

The advancing front algorithm constructs the mesh mainly from an initial generation front and has been widely implemented using triangles and quadrilaterals in 2D and using tetrahedrons in 3D (Owen, 1998; George, 1991). The main feature of the advancing front technique is the concurrent generation of element nodes and the elements. The validity of each new generated element is checked locally as soon as it is created. The data
required is just the boundary, or more accurately, the polygonal discretization of it. Input is always the set of oriented line segments (Lee and Hobbs, 1999). This technique also demands generation of the initial front from the boundary segments. The dimensions of initial front segments need to concur with the type of grading desired in the mesh. The grading is defined by a node spacing function as discussed by Lo (1992). The initial front is a kind of departure zone for the new elements to be generated. Each time an element is created the front needs to be updated and the element creation process is pursued as long as the front is not empty. The elements in the advancing front method are generated one at a time, and hence the user has full control over the shape and size of the element generated. The complete algorithm for mesh generation on a 2D surface using this technique is discussed in this section. Several set theory notations are used in the discussion of this algorithm, the meanings of which are as following:

\( \epsilon \) - belongs to
\( \{\} \) - represents the set of objects
\( \cap \) - union
\( \forall \) - for all

### 4.1.1 Initial Generation Front

For 2D plane surface mesh generation, the initial generation front is usually a union of simple closed loops of straight line segments, also known as planar straight line graph (PSLG). To start meshing from the boundary, a systematic way for the identification of the interior domain is required. One method of determining the relevant interior domain is the orientation of outer and inner boundary segments in anticlockwise and clockwise directions, respectively, or vice versa. Figures 4.1 and 4.2 show two different orientations of the outer and inner boundaries of a region to be meshed. One should note that defining the boundary segments in such a fashion always puts the domain of interest either on the left, Figure 4.1, or right, Figure 4.2, of the oriented boundary segments. The WoodFrameMesh program assumes that the corner nodes are entered in a continuous fashion, but all the boundary segments are required to be oriented, either as shown in Figure 4.1 or Figure 4.2, before any element generation process is pursued.
4.1.2 Definitions

The following terminologies are used in developing the advancing front technique algorithm discussed by Lee and Hobbs (1999):

1.) Node, N: A node N is defined as a point in a 3D vector space. Figure 4.3 shows a node N in a 3D vector space.
2.) Line Segment, AB: AB denotes a line segment connecting nodes A and B. Figure 4.4 shows a line segment AB.

3.) Shortest Distance, $\delta(N,AB)$: $\delta(N,AB)$, defines the shortest distance between a node N and a line segment AB, i.e., the perpendicular distance between N and AB. Figure 4.5 illustrates the shortest distance between node N and line segment AB.
4.) Generation Front, $\Gamma$: A collection of all active line segments and the nodal points on these segments which give the perimeter of the unmeshed region. Figure 4.6 shows an example of a generation front enclosing an unmeshed region.

![Figure 4.6: Generation front](image)

5.) Area Operator, $A(N,AB)$: $A(N,AB)$ is the operator measuring the area of triangle formed by the node N and the line segment AB. *Note:* If the outer boundary is oriented as shown in Figure 4.7, then if N lies inside the domain, the area operator will give a value greater than zero; otherwise it will give a value less than zero.

![Figure 4.7: Area operator](image)
6.) Intersection Operator, $\cap(AN,NB,\Gamma)$: This operator tests whether or not the two line segments AN and NB intersect with the generation front $\Gamma$. It will return a value 0 if and only if $AN \cap \Gamma \notin \{AN,\{A,N\}\}$ and $NB \cap \Gamma \notin \{NB,\{N,B\}\}$, i.e., neither segment intersects any other segment on $\Gamma$. In all other cases $\cap(AN,NB,\Gamma)$ will return a value of 1. Figure 4.8 shows how intersection operator works.

![Intersection operator diagram](image)

Figure 4.8: Intersection operator

7.) Valid Node Set, $\wedge(AB,\Gamma)$: Any node N belonging to $\Gamma$ is a valid node with respect to the segment AB if the following conditions are satisfied:

(i) $A(N,AB) > 0$,

(ii) $\cap(AN,NB,\Gamma) = 0$,

(iii) The triangle ABN does not contain any existing node in the mesh.

The valid node set $\wedge(AB,\Gamma)$ with respect to segment AB is defined as the collection of all the valid nodes in $\Gamma$ with respect to AB i.e. $\wedge(AB,\Gamma) = \{N \in \Gamma, \text{ such that above three conditions are satisfied}\}$. Figure 4.9 shows a collection of valid node set where each valid node is represented by a star (*) symbol.
8.) $\alpha$ - Quality of Triangle: To generate the best triangle each time an element is formed, it is necessary to have a measure of quality of triangle which is defined as $\alpha$. For a given triangle $\Delta(ABN)$, the $\alpha$ value is defined as shown in equation (1).

$$\alpha(ABN) = \frac{2\sqrt{3} \left\| \mathbf{AB} \times \mathbf{AN} \right\|}{(\|\mathbf{AB}\|^2 + \|\mathbf{BN}\|^2 + \|\mathbf{AN}\|^2)} \tag{1}$$

In this expression $2\sqrt{3}$ is a normalizing factor so that the ideal (equilateral) triangle will attain a maximum value of 1. The higher the value of $\alpha$, the better is the quality of the triangle. Figure 4.10 shows the $\alpha$ quality of some example triangles.
9.) \( \mu \) - Quality of Triangle: In the generation of the graded mesh in which the element size changes progressively, both the shape and the deviation of element size from the desired value should be taken into account. Hence in view of the element size effect, the \( \mu \) coefficient given in equation (2) is defined to judge the quality of triangular elements:

\[
\mu = \frac{s}{\rho} (2 - \frac{s}{\rho}) \alpha
\]

(2)

\( s \) and \( \rho \) are the actual and desired size\(^1\) of the elements, respectively, and \( \alpha \) is as defined above.

10.) Optimal Offset Height, \( h_o \): \( h_o \) is defined as an optimal offset height for an offset node \( N \) from the line segment \( AB \) of the generation front \( \Gamma \). If it is required that a node \( N \) needs to be introduced such that \( \Delta(ABN) \) attains a maximum value of \( \mu \), then equation (3) can be used.

\[
\mu = \frac{h_o}{H} \left( 2 - \frac{h_o}{H} \right) \frac{2\sqrt{3}h_o d}{(2h_o^2 + 1.5d^2)}
\]

(3)

\(^1\) The desired size of any element is obtained from the node spacing function.
where $H = \frac{\sqrt{3}}{2} \rho$, and $\rho$ is the desired element size at $M$, the mid point of $AB$ with $d = ||AB||$. Putting $f = \frac{h_o}{d}$, $k = \frac{H}{d}$ and rearranging the above equation (3) we get:

$$\mu = \frac{\sqrt{3}}{k} \left(2f - \frac{f}{k}\right) \frac{f^2}{(f^2 + 0.75)}$$

(4)

Now by setting $\frac{d\mu}{df} = 0$, we get:

$$f^3 + 2.25f - 3k = 0$$

(5)

This equation (5) can easily be solved by a few steps of Newton Raphson iteration with

$$f_0 = \frac{3k}{2.25} \text{ and } f_{i+1} = \frac{(2f^3 + 3k)}{(3f^2 + 2.25)} \text{ for } i = 0, 1, 2$$

(6)

Once the value of $f$ is found, then $h_o$ can be set equal to $f \cdot d$
4.1.3 Triangular Element Generation Process - Pseudo code

Figure 4.11 presents the pseudo code for the triangular element generation process as discussed by Lee and Hobbs (1999). A slight modification is done in the implementation by using the node spacing function discussed by Lo (1992) as compared to the background mesh interpolation used by Lee and Hobbs (1999). Figure 4.12 presents the formula given by Lo (1992) which derives the node spacing values from the initial generation front sizes.

Start:
1.) Find AB, the shortest segment on generation front $\Gamma$;
2.) Find M, the mid point of AB;
3.) Get NSF, the node spacing function\(^2\) at M;
4.) Find $h_0$, equal to $\frac{\sqrt{3}}{2} (NSF)$;
5.) Find N, a point which lies on the perpendicular bisector of AB and forms a positive area with AB.
6.) Form C, a search circle with radius $h_0$ and center at N. This search circle contains all the points lying on or inside this search circle. Denote this list as $\psi$.
7.) If $\psi = \{\phi\}$, the empty set i.e. there are no other points inside the search circle C, then test point N for the following three conditions:
   7.1) $\cap (AN, NB, \Gamma) = 0$,
   7.2) $\delta(N, CD) \geq \frac{\|AB\|}{2} \quad \forall \ CD \in \{\Gamma-AB\}$,
   7.3) $\Delta(ABN)$ does not contain any existing node in the mesh.
8.) If 7.1, 7.2 and 7.3 are satisfied, then go to step 11.
9.) If 7.1, 7.2 and 7.3 are not satisfied, then:

\(^2\) Use the formula discussed by Lo(1992) for finding node spacing function.
9.1) Form the valid node set \( \Lambda(AB, \Gamma) \). Find the node \( V \in \Lambda(AB, \Gamma) \) such that \( \mu(ABV) \geq \mu(ABK) \) \( \forall K \in \Lambda(AB, \Gamma) \) i.e. \( V \) is the node in \( \Lambda(AB, \Gamma) \) which will form the best quality element with \( AB \).

9.2) Reduce the trial offset height \( h \) by setting \( h = 0.9 \ h_o \).

9.3) If \( h \geq 0.25 h_o \) then re-establish the trial offset point \( N \) according to the new value of \( h \). Test \( N \) for conditions in 7.1, 7.2 and 7.3.

9.3.1) If \( N \) satisfies these conditions and \( \mu(ABN) > \mu(ABV) \) then go to step 12 with \( \Delta(ABN) \) as new element.

9.3.2) If \( N \) satisfies these conditions and \( \mu(ABN) < \mu(ABV) \) then go to step 12 with \( \Delta(ABV) \) as new element.

9.3.3) If \( N \) does not satisfy these conditions then go to step 9.2.

9.4) If \( h < 0.25 h_o \) then go to step 11 with \( \Delta(ABV) \) as new element.

10.) If \( \psi \neq \{ \phi \} \) i.e. there are one or more other points inside the search circle \( C \), then form the valid node set \( \Lambda(AB, \psi) \).

10.1) If \( \Lambda(AB, \psi) = \{ \phi \} \) then go to step 7.

10.2) If \( \Lambda(AB, \psi) \neq \{ \phi \} \) then find the node \( V \in \Lambda(AB, \psi) \) such that \( \mu(ABV) \geq \mu(ABK) \) \( \forall K \in \Lambda(AB, \psi) \). Now, go to step 11 with \( \Delta(ABV) \) as new element.

11.) Form a triangle with segment \( AB \) and the new node and then deactivate the segment \( AB \) from the generation front i.e. \( \Gamma = \Gamma - AB \).

12.) Check the remaining active segments in the generation front \( \Gamma \) as obtained from step 11. If the remaining active segments are reduced to zero i.e. the domain is completely meshed, then exit; else go to step 1.

*End;*

---

*Figure 4.11: Pseudo code for triangular element generation process*
Figure 4.12: Formula for NSF at any point P(x, y)

\[ d_i = \| PA_i \| \]
\[ \rho_i = \| A_i B_i \| \]
\[ NSF(x, y) = \frac{\sum_{i=1}^{i=N} \rho_i}{\sum_{i=1}^{i=N} d_i} \]

The pseudo code presented is implemented in the WoodFrameMesh program and is capable of meshing complex domains with multiple openings and multiple constraint lines. Figure 4.13 shows an example domain with multiple openings and multiple constraint lines. The WoodFrameMesh program handles the domain with openings by defining their boundary segments in the opposite orientation as compared to the outer boundary segments i.e. if outer boundary segments are oriented clockwise with respect to a normal then opening boundary segments will be oriented anti-clockwise with respect to that normal or vice versa. For handling a constraint line, the program generates a very tiny virtual opening, and once the domain is completely meshed, these virtual opening nodes are merged back upon the constraint line nodes.
4.1.4 Example Domains Meshed Using AFT Algorithm

This section presents a few example figures of domains meshed using advancing front technique implemented in the WoodFrameMesh program. These figures are viewed in the SAP 2000 version 7.4 program. We have observed that if the geometry being triangulated is irregular then after the generation of the mesh, some poor aspect ratio elements may be generated in some regions. In such a case the generated mesh may require some corrections to enhance its quality, i.e., rearrangement of the topological structure of the mesh. As of now, no such correction technique is implemented in the WoodFrameMesh program. If the geometry is simple, then the mesh generated using the advancing front technique is almost uniform and no corrections are needed.
Figure 4.14: Square shaped domain in xy-plane

Figure 4.15: L-shaped domain in xy-plane
Figure 4.16: Square-shaped domain with an opening in xy-plane

Figure 4.17: Square-shaped domain with a constraint line in xy-plane
Figure 4.18: Square shaped domain with two openings and two constraint lines in xy-plane

Figure 4.19: Virginia Tech logo in xy-plane
4.2 Algorithms Inside AFT

Implementing the advancing front technique involves several small geometric algorithms. A few important geometric algorithms implemented in the WoodFrameMesh program are discussed in this section.

4.2.1 Finding Point Inside the Domain

For a simple situation of a convex domain, as shown in Figure 4.20, bounded by straight lines, one can loop over the boundary segments either clockwise or anticlockwise ensuring that the point lies either to the right of the traversed path or to left of the traversed path. Such a method does not work for non-convex domains, as shown in Figure 4.21, as the point in question might not lie at the desired position (either left or right) for one or more line segments.

![Figure 4.20: A convex domain with an inside point lying to the left of all oriented boundary segments](image)

Figure 4.20: A convex domain with an inside point lying to the left of all boundary segments
Figure 4.21: A non-convex domain with an inside point lying to the left of all the boundary segments except for the one shown with dash style.

Such a case of a non-convex polygon is solved by using a popular method known as ray tracing. In this method another point is defined strictly outside the domain and a ray is formed joining the point in question with a new point. The number of intersections of the ray with the boundary line segments is recorded. The point in question is inside the domain if the number of intersections is odd and outside if the number of intersections is even. Figure 4.22, shows two examples presenting ray tracing method.

Figure 4.22: Showing ray tracing method
However, two exceptions need to be handled while solving the problem using the ray tracing method. For example, what if a ray intersects the boundary line segments at their vertices as shown in Figure 4.23? What if the ray is collinear with one of the edges as shown in Figure 4.24?

Figure 4.23: Rays intersecting vertices of the domain

Figure 4.24: Rays overlapping edges of the domain

Both of these cases are handled by ensuring that boundary edges are on either side of the ray while recording the number of intersections, or otherwise zero intersections are recorded. Figures 4.25 and 4.26 present the solution to the above mentioned problem.
Figure 4.25: Showing solution to the problem of ray intersecting the vertices of boundary edges

- Record 1 intersection, if both the boundary line segments lie on either side of the ray, else record 0 intersection
- Boundary segments lying on the either side of the ray are shown in dash style

Figure 4.26: Showing solution to the problem of ray being collinear with the boundary edges
4.2.2 Checking Intersection of two Line Segments

To generate a valid element using the advancing front technique, one needs to check that the new element does not intersect the domain boundaries. The element intersection in effect is the check for intersection of line segments. To develop a method to solve this problem, we consider two line segments AB and CD, as shown in Figure 4.27, for which we need to check if they intersect. The procedure for checking for intersection is as follows:

1.) Find vector AB, AC, CD
2.) Find cross-product of AC, CD i.e. AC×CD
3.) Find cross-product of AB, CD i.e. AB×CD
4.) Compute scalar s; s = \frac{(AC×CD)\bullet(AB×CD)}{\|AB×CD\|}
5.) Compute scalar t; t = \frac{(-AC\bullet CD + s(AB\bullet CD))}{\|CD\|^2}
6.) If (s ≥ 0 and s ≤ 1 and t ≥ 0 and t ≤ 1) then AB and CD intersect; otherwise they do not.

Figure 4.27: Sowing line segments AB and CD
4.2.3 Orienting Line Segment

Orienting boundary line segments is another operation which may be required if the input data is not oriented as desired in generating meshes using the advancing front technique. For example, what if both the outer and the interior domains are oriented in the same direction? Here one should recall that outer and interior domains are required to be oriented anti-clockwise and clockwise or vice-versa to generate the mesh. So, a positive normal to the plane containing the boundary line segments is defined and then the boundary segments are oriented either clockwise or anti-clockwise with respect to the defined normal. If AB, as shown in Figure 4.28, is the boundary segment and C is a point lying inside the domain, then to find the orientation of AB with respect to the normal, one needs to solve for $$\| (AB \times AC) - N \|$$.

If $$\| (AB \times AC) - N \| \approx 0$$ then AB is anti-clockwise with respect to the normal; otherwise it is clockwise.

![Figure 4.28: Showing orientation of AB with respect to normal N](image)

4.2.4 Finding Distance of Line Segment from a Point

In the advancing front technique, to check for proximity of newly generated nodes from the generation front one needs to find the distance between the new point and the line segments on the generation front. If C and AB are the point and line segment, respectively, in space, as shown in Figure 4.29, then the distance between them could be derived as follows:

1.) Find $$\theta = \cos^{-1} \left( \frac{AB \cdot AC}{\|AB\| \|AC\|} \right)$$
2.) Find \( d = \|AC\| \sin \theta \)

d gives the minimum distance i.e. perpendicular distance of line segment from point C.

![Figure 4.29: Showing distance of point from a line segment](image)

4.3 Quadrilateral Mesh Generation

Quadrilateral mesh generation for quadrilateral domains inside WoodFrameMesh program is accomplished in the following four steps. The method discussed always generates a structured quadrilateral mesh. An example is presented to mesh a quadrilateral domain shown in Figure 4.30. The only requirement is that nodes for the domain must be entered in a continuous fashion.

![Figure 4.30: A simple quadrilateral domain](image)
1.) Name the boundary nodes i.e. A, B, C and D in the continuous order, as shown in Figure 4.31.

Figure 4.31: Boundary nodes named in continuous order

2.) Find the shorter of the segments from opposite edges i.e. AB, CD and AD, BC. Now divide these short segment lengths by the mesh size and find the number of subdivisions. Divide these shorter segments into the number of subdivisions, and record the boundary nodes. Figure 4.32 shows the subdivision of shorter segments.

Figure 4.32: Showing subdivision of shorter segments from opposite edges
3.) Divide the opposite longer edges into the same number of subdivisions, as shown in Figure 4.33.

![Figure 4.33: Dividing opposite edges into same number of subdivisions as the opposite shorter edges](image)

4.) The elements are generated one by one, along any one of the edges of the domain. Figure 4.34 shows the complete element generation process. The elements are generated in the order of the numbers printed on them.
As the quadrilateral mesh is desired only for walls which in most of the houses have rectangular shapes, a good quality quadrilateral mesh is obtained using this method. Also, this method proves to be very fast and efficient as it does not require performing any global intersection or searching. The drawbacks of this method are: (1) it is applicable only to quadrilateral domains, and (2) in addition it will not generate a good quality mesh for highly skewed domains as the one shown in Figure 4.36.
Apart from the above presented algorithms, there are several small but very frequently occurring run time operations involving points and line segments in space which are implemented in the program. These operations get enhanced by performing these operations in an object oriented style. Exact arithmetic is computationally expensive and can slow down these computations considerably. So, special care is taken while implementing these numerical algorithms which get affected by floating point round off errors. A tolerance is set wherever required.