Chapter 8

Mesh Generation

8.1 Mapping Methods

There are two basic *mapping methods* for generating a mesh. One involves transforming a mesh from a simple known geometry onto that of the more complicated system. Usually on starts with a reference mesh, denoted by $\hat{\Omega}$ and then modifies this mesh by a function which maps a point from the reference mesh to one of the real mesh, Ω which assure certain "appropriate" continuity conditions as shown in figure 8.1. This can be stated simply as:

Simple Mesh + Mapping Function → Mesh of actual system

Finding an appropriate mapping function, often called the *transport function*, is the key to producing a robust mesh, i.e. one capable of handling realistic geometries.

Common transport functions often include,

1. Methods based on the domain verticies. Here individual verticies are mapped via the creation of an image I of a vertex of the reference mesh, $\hat{I}$ via $I = F(\hat{I})$. The mapping function, F is described by $F(\hat{x}, \hat{y}) = \sum_i P_i(\hat{x}, \hat{y}) v_i$, where the P_i's are polynomials, and the v_i's correspond to the ith vertex of the real mesh as show in figure 8.2. If we assume that our actual mesh will only vary linearly from the reference mesh, then we can define the set of polynomials such that we have a degree 1 mapping.

$$\begin{aligned}
F(x,y) &= \sum_{i=1}^{4} P_i((\hat{x}),(\hat{y}))v_i \\
P_1(\hat{x},\hat{y}) &= (1-\hat{x})(1-\hat{y}) \\
P_2(\hat{x},\hat{y}) &= \hat{x}(1-\hat{y}) \\
P_3(\hat{x},\hat{y}) &= \hat{x}\hat{y} \\
P_4(\hat{x},\hat{y}) &= (1-\hat{x})\hat{y} \qquad (8.1)
\end{aligned}$$

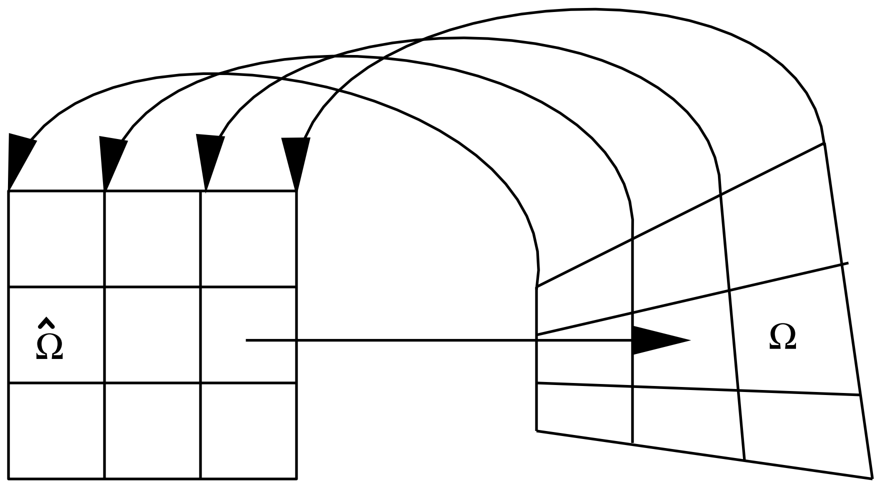

Figure 8.1: Mapping from the reference mesh to the actual mesh.

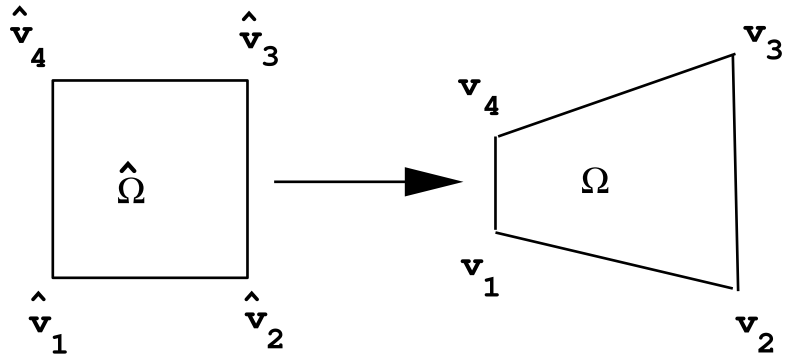

Figure 8.2: Mapping vertices from the reference space to the physical space.

2. Methods which preserve a polygonal contour. In this method, the function F maps a polygonal contour exactly. This approach is applicable in the case of domains (and subdomains) which are topologially similar. In practice, one usually chooses contour domains, Ω in terms of 3 or 4 topologial sides in such a way that the number of points on two logically connected sides is equal.

3. Methods which create the mesh by solving a set of partial differential equations. This method is similar to the previous two in which the domains must be geometrically similar. But instead of defining the function in advance, the mapping function is computed by solving a predetermined system of partial differential equations. The pdes are chosen in an "appropritate" way much as in the standard mapping methods.

Let's take a closer look at the mesh generation methods which involve solving systems of partial differential equations.

We let Ω be the real domain which we wish to mesh and is assumed to be topologially similar to a generalized quadrilateral (2D) or generalized hexahedron (3D). We define a reference domain, $\hat{\Omega}$ to be a unit square (or hexahedron or parallelpiped). This domain is meshed canonically by a structered grid.

We then let, x, y, z be the variables attached tot he physical domain, Ω and u, v, w be the variables related to the reference domain, $\hat{\Omega}$.

We then need to specify,

1. A system of appropriate partial differential equations, (S) and

2. A set of appropriate boundary conditions, (BC).

For each pair of (S) and corresponding (BC) for which a solution exists we can obtain a mesh generation method. Obviously, the difficulty comes in choosing the appropriate set of (S) and (BC) which best satifies certain physicl and geometrical attributes of the problem.

There are two general approaches for selecting (S) and (BC):

1. (S) is posed in terms of variables x, y, z with, as boundary conditions, a subset of isovalues specified on the four (or six) faces of the domain. (S) is then inverted (i.e. instead of computing $u(x, y, z)$ we instead compute $x(u, v, w)$) to yield a coupled system which is usually solved in some iterative fashion.

2. (S) is posed in terms of variables u, v, w with, as boundary conditions, (BC), the exact values of the contour of Ω, i.e. the coordinates on this particular contour. (S) is subsequently solved in the reference mesh, $\hat{\Omega}$ and its solution $x(u, v, w), y(u, v, w), z(u, v, w)$

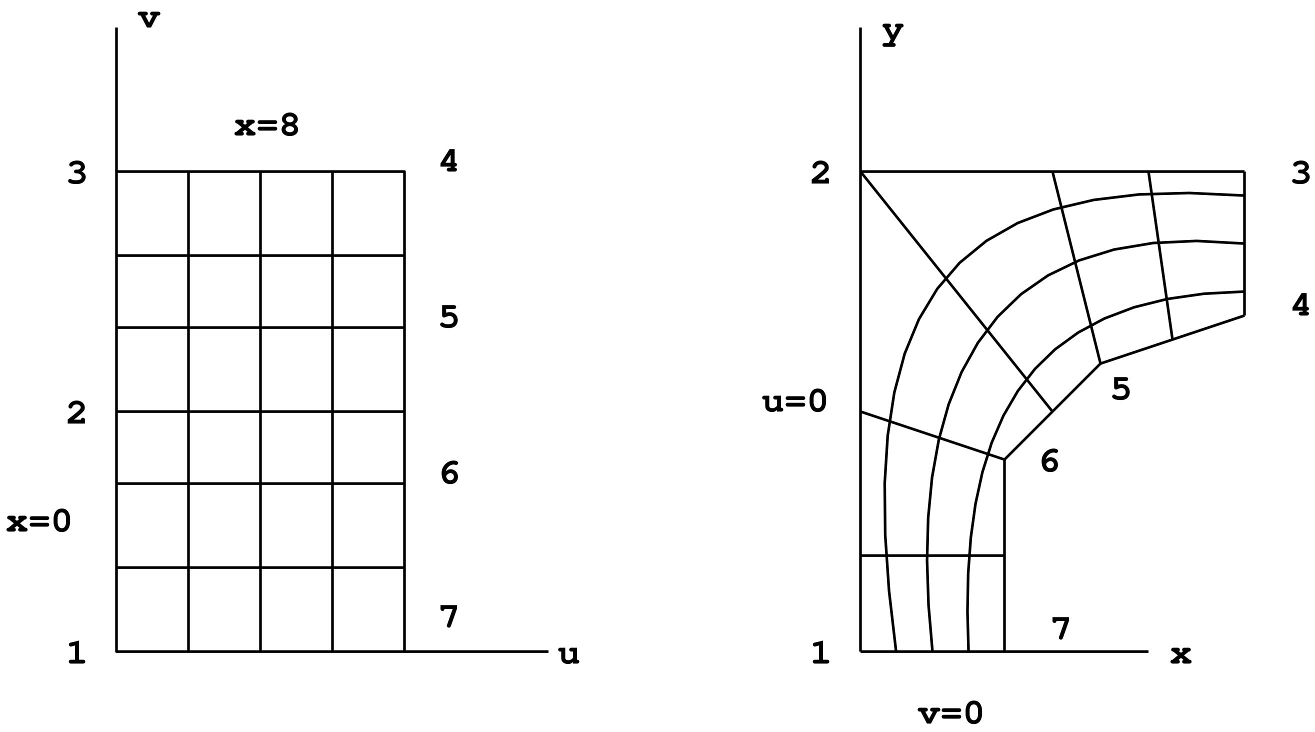

Figure 8.3: The $x-y$ solutions are the $u-v$ isocontours.

is used for the contruction of the the mesh, Ω. (In practice, this turns out to be a simplification of the first method. It usually produces methods which are difficult to provide good theoretical footing and thus, ends up being primarily based upon hueristics).

The first method utilizes boundary information in the u, v, w space which when mapped (via solving a partial differential equation), yields the transformed mesh in the x, y, z system. The x, y, z "solutions" can be considered as u, v, w *isocontours*. An example of this method is shown in figure 8.3 If the coordinate points are specified on the entire closed boundary of the physical region, the pdes are elliptic in nature. If, however, only a portion of the boundary is specified, then the resulting pdes would be either parabolic or hyperbolic. The latter would occur, for instance, when an inner boundary is specified, but an outer boundary is left arbitrary.

The simplest elliptic system which exhibits a one-to-one mapping, takes advantage of the smoothing properties of the Laplcian operator and involves solving Laplace's equation.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad \text{in } \Omega \tag{8.2}$$

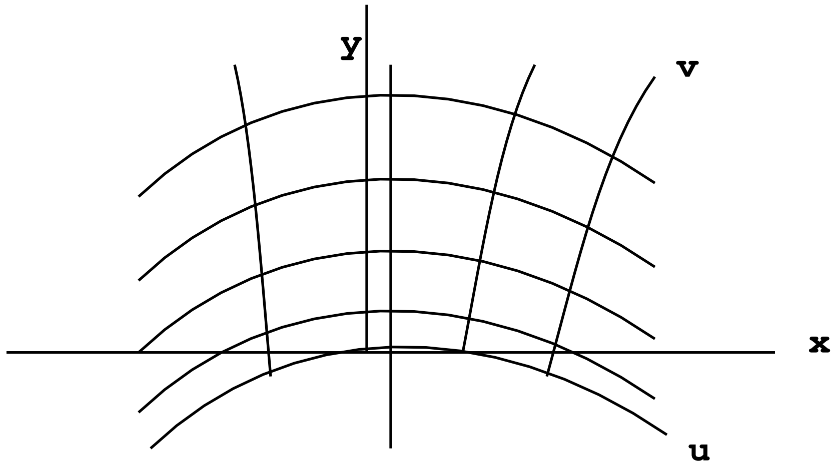

Figure 8.4: $v_{xx} > 0$ because of the convex curvature of the lines.

with a set of boundary conditions on Γ and

$$\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad \text{in } \Omega \tag{8.3}$$

again with a set of boundary conditions on Γ.

As we noted in our discussion of finite elements, we can write an equivalent minimization formulation of the elliptic pde, which states that we must minimize the following functional,

$$F = \int |\nabla u_i|^2 d\Omega \tag{8.4}$$

with respect to the set of coordinates u, v, w.

Since the coordinate lines are located at equal increments, the quantity, $|\nabla u_i|^2$ can be considered a measure of the grid point density along the coordinate line on which u_i varies. This means that u_i must change rapidly in physical space where grid points are clustered. Minimization of the integral leads to the smoothest corrdinate line distribution over the field.

Thus in the abscence of boundary curvature the transformation will yield equally spaced coordinate lines. The lines will become more closely spaced over convex boundaries as shown in figure 8.4 and less closely spaced over concave regions as shown in figure 8.5

Let's go back to the system of Laplace's equations and solve them. Assuming that the solution to these equations can be found, we want to invert the relationships $u(x, y)$ and $v(x, y)$ to obtain $x(u, v)$ and $y(u, v)$. We can do this by considering the relationship between the space defined by (u, v) and (x, y) to be a coordinate transformation. This could be, for example, the relationship between curvilinear and rectangular coordinates. We can do this in two ways. The most general is to first describe covariant and contravariant base vectors and then to construct a metric tensor. We can then define generalized transformations and

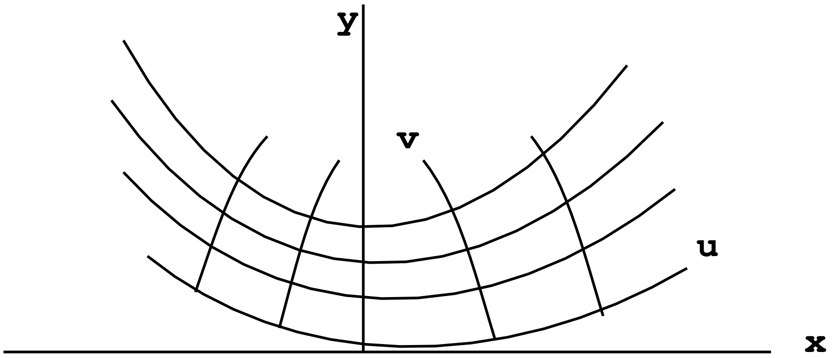

Figure 8.5: $u_{xx} < 0$ because of the concave curvature of the lines.

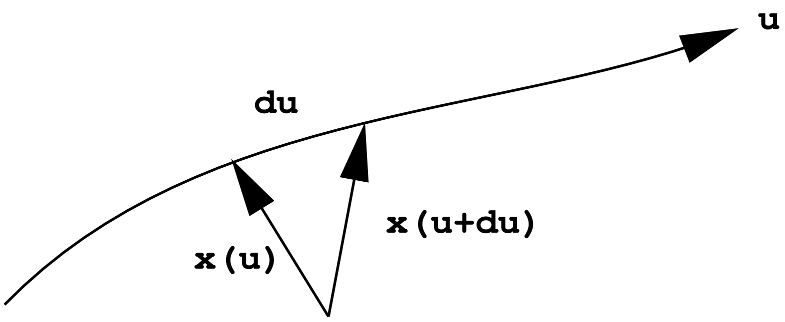

Figure 8.6: Covariant vectors (tangent vector).

geometrical quantities such as arc length, surface area, etc. One can define covariant vectors (tangent vectors), see figure 8.6, as

$$a_i = x_{u^i} \quad i = 1, 2, 3 \tag{8.5}$$

and contravariant vectors, see figure 8.7, as

$$a^i = \nabla u^i \tag{8.6}$$

We can then define the metric tensor as

$$(ds)^2 = \sum_{i=1}^{3} \sum_{j=1}^{3} a_i a_j du^i du^j = \sum_{i=1}^{3} \sum_{j=1}^{3} g_{ij} du^i du^u \tag{8.7}$$

where

$$g_{ij} = a_i \cdot a_j \tag{8.8}$$

The second way is to specifically define the coordinate transformations between $u(x, y)$ and $x(u, v)$. We start by using the chain rule:

$$dx = \frac{\partial x}{\partial u} du + \frac{\partial x}{\partial v} dv$$

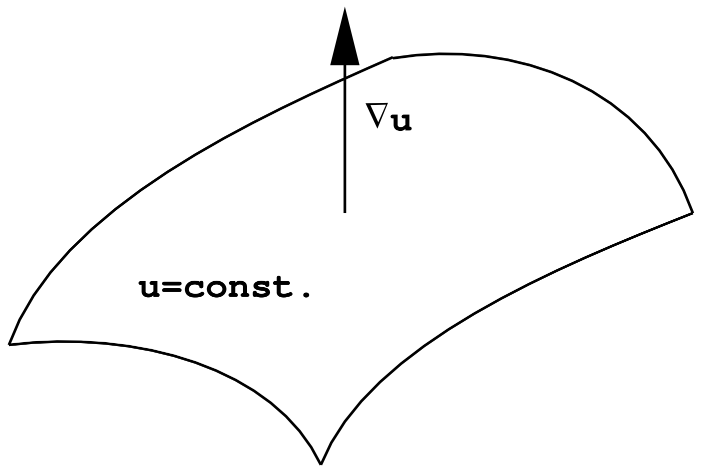

Figure 8.7: Contravariant vectors.

$$dy = \frac{\partial y}{\partial u}du + \frac{\partial y}{\partial v}dv$$
$$du = \frac{\partial u}{\partial x}dx + \frac{\partial u}{\partial y}dy$$
$$dv = \frac{\partial v}{\partial x}dx + \frac{\partial v}{\partial y}dy \tag{8.9}$$

Using the above equations, we can define the soon to be useful relationships:

$$dx = \frac{\partial x}{\partial u}[\frac{\partial u}{\partial x}dx + \frac{\partial u}{\partial y}dy] + \frac{\partial x}{\partial v}[\frac{\partial v}{\partial x}dx + \frac{\partial v}{\partial y}dy] \tag{8.10}$$

and

$$dy = \frac{\partial y}{\partial u}[\frac{\partial u}{\partial x}dx + \frac{\partial u}{\partial y}dy] + \frac{\partial y}{\partial v}[\frac{\partial v}{\partial x}dx + \frac{\partial v}{\partial y}dy] \tag{8.11}$$

This yields the following identities,

$$x_u u_x + x_v v_x = 1$$
$$x_u u_y + x_v v_y = 0$$
$$y_u u_x + y_v v_x = 0$$
$$y_u u_y + y_v v_y = 1 \tag{8.12}$$

or in matrix form,

$$\begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix} = \frac{1}{J}\begin{pmatrix} y_v & -x_v \\ -y_u & x_u \end{pmatrix} \tag{8.13}$$

where $J = x_u y_v - x_v y_u$ is the Jacobian and determinant of the matrix,

$$M = \begin{pmatrix} x_u & x_v \\ y_u & y_v \end{pmatrix} \tag{8.14}$$

Using the fact that $u_x = y_v/J$ we can obtain u_{xx},

$$u_{xx} = \frac{\partial}{\partial u}[\frac{y_v}{J}]u_x + \frac{\partial}{\partial v}[\frac{y_v}{J}]v_x \tag{8.15}$$

or

$$u_{xx} = \frac{1}{J^2}[y_v y_{vu} - y_u y_{vv}] + \frac{1}{J^3}[-J_u y_v^2 + J_v y_v y_u] \tag{8.16}$$

with similar relationships for u_{yy}, v_{xx} and v_{yy}.

Since $\nabla^2 u = 0$ and $\nabla^2 v = 0$ we can now solve the previous equations in terms of (x, y), which yields,

$$g_{11}x_{uu} + g_{22}x_{vv} + 2g_{12}x_{uv} = 0 \tag{8.17}$$

and

$$g_{11}y_{uu} + g_{22}y_{vv} + 2g_{12}y_{uv} = 0 \tag{8.18}$$

Here, g is the metric tensor that we mentioned previously and can be defined as,

$$g_{ij} = \sum_{m=1}^{2} A_{mi}A_{mj} \tag{8.19}$$

where,

$$A_{mi} = (-1)^{i+m}(\text{Cofactor}_{m,i} \text{ of } M) = (-1)^{i+m}\det(M_{ij}) \tag{8.20}$$

To give you an idea of how to evaluate the metric tensor, g_{ij}, let's consider a few examples.

$$g_{11} = \sum A_{m1}A_{mj} = A_{11}A_{11} + A_{21}A_{21} = x_u^2 + y_u^2 \tag{8.21}$$

where the A_{mi} terms are evaluted as

$$A_{11} = (-1)^2\det(M_{11}) = x_u \tag{8.22}$$

Therefore, upon evaluating all the g_{ij} terms, equation (8.17) yields

$$(x_u^2 + y_u^2)x_{uu} + (x_v^2 + y_v^2)x_{vv} + 2(x_u x_v + y_u y_v)x_{uv} = 0 \tag{8.23}$$

with a similar equation for (8.18).

These form a set of quasi-linear elliptic equations whose solution will yield a mesh in $\hat{\Omega}$.

While the solution of the set of Laplace's method has been a popular one, we can obtain more control of our meshes by utilzing an inhomogeneous elliptic equation (most usually Poisson's equation) such that we can explore different characteristics with different source functions.

If P and Q are our right hand sides added to the previous pair of Laplace's equations, the inverse system to be solved is,

$$g_{11}x_{uu} + g_{22}x_{vv} + 2g_{12}x_{uv} + J^2(Px_u + Qx_v) = 0 \tag{8.24}$$

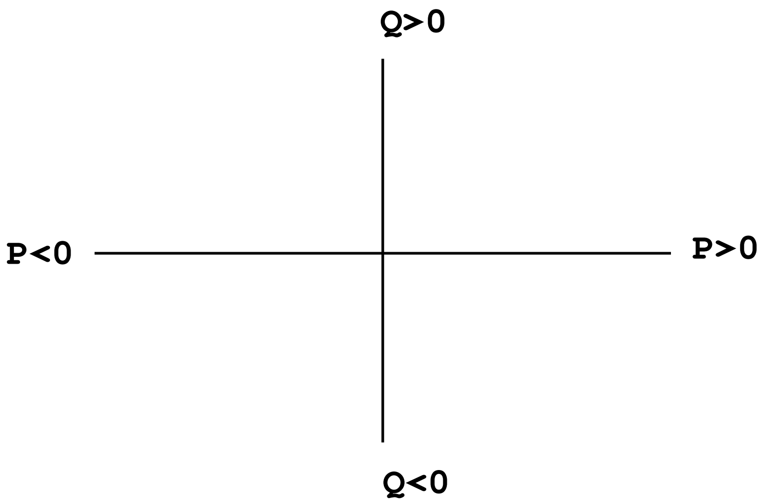

Figure 8.8: Using P and Q to attract points.

and

$$g_{11}y_{uu} + g_{22}y_{vv} + 2g_{12}y_{uv} + J^2(Py_u + Qy_v) = 0 \tag{8.25}$$

The inhomogeneous right hand sides have the following effects on our mesh:

- If $P < 0$ - points are attracted to the right.
- If $P > 0$ - points are attracted to the left.
- If $Q > 0$ - points are attracted to the top.
- If $Q < 0$ - points are attracted the the bottom.

as shown in figure 8.8

When we get close to the boundary, P and Q effect the lines by shifting toward the boundary much in the way for the Laplace's system does. This can be seen in figures 8.9 and 8.10.

One can chose P and Q such that they can be attracted to either lines or points as well. For example, if P is defined as:

$$P(u,v) = -\sum_{i=1}^{n} a_i sign(u-u_i)e^{-c_i|u-u_i|} - \sum_{i=1}^{m} b_i sign(u-u_i)e^{-d_i[(u-u_i)^2+(v-v_i)^2]^{\frac{1}{2}}} \tag{8.26}$$

where n and m are the number of lines in u and v.

- If $a_i > 0, i = 1, n$ the lines of u are attracted to the line u_i.
- If $b_i > 0, i = 1, m$ the lines u are attracted to the point u_i, v_i.

as shown in figures 8.11 and 8.12. The distances to the line/point can be modulated by the coefficients c_i and d_i.

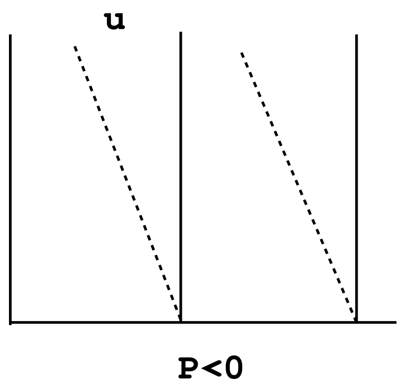

Figure 8.9: Effects of P near the boundary.

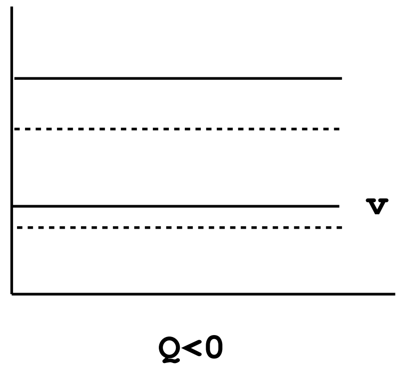

Figure 8.10: Effects of Q near the boundary.

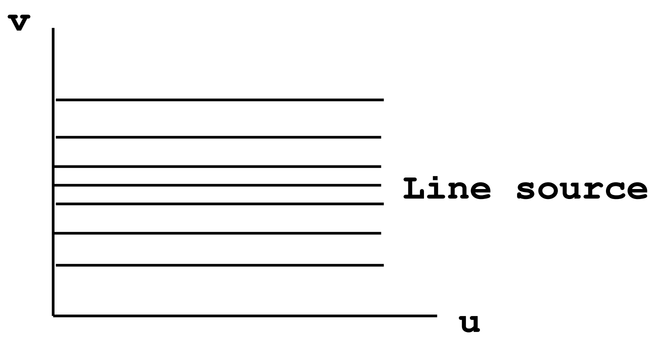

Figure 8.11: Effecting changes near a line source.

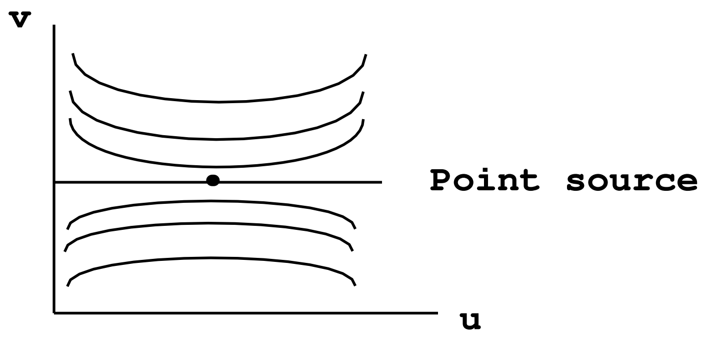

Figure 8.12: Effecting changes near a point source.

8.2 Delaunay Triangulation

Delaunay traingulation and/or Voronoi diagrams and/or Dirichlet tesselation are all related to a kind of mesh generation that utilizes a point insertion process to create the polygonal discretization.

The two general classes of problems which are tackled by the Delaunay approach include,

- problems in which the boundary is given. A mesh is produced which is contrained by the boundary, but allows extra points to be inserted on said boundary.
- problems in which the a dicretized boundary is given. A mesh is produced that is constrained by the discretized points on the boundary and no extra points are allowed to be added.

The method that we will outline will be appliciable to both sets of problems. We now outline (following [43] a general Delaunay based point insertion process for creating two- and three-dimensional uniform and non-uniform meshes.

1. Let $\{x_k\}$ be the set of points in space.
2. Let T_i be a mesh whose element verticies are represented by the first i points of $\{x_k\}$.
3. The mesh T_{i+1} is derived from the mesh T_i such that the point x_{i+1} of the initial set of points is one of its vertices.
4. We can define the location of the point x_{i+1} exactly given the following two sets:
   (a) $\mathbf{C}_i = \bigcup_j C_j$. C_j is definied as the circumsphere (circle) of the element, k_j of T_i.
   (b) $T_i = \bigcup_j k_j$ is the triangulation (tetrahedralization in 3d) which includes the first i points of the mesh.

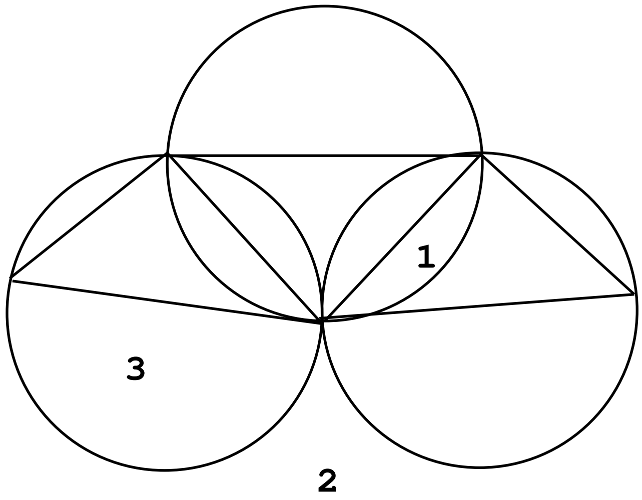

Figure 8.13: The three ways of inserting a point into a Delaunay triangulation.

For Delaunay triangulation there are only three distinct possible ways of inserting a point, shown in figure 8.13. The points are part of the following sets.

1. $x_{i+1} \in T_i \quad (\forall k_j \in T_i s.t. x_{i+1} \in k_j$. Here $T_i = \bigcup_j k_j \equiv$ the triangulation of the first i points.

2. $x_{i+1} \notin C_i \quad (x_{i+1}$ is not in any circle (sphere) C_j.

3. $x_{i+1} \notin T_i$ but $x_{i+1} \in C_i \quad (x_{i+1}$ is not an element k_j of T_i but is in a circle.

Now let's looks at each of the cases one by one. In case number 1 we are adding a point to a region that is already triangulated. Therefore we need to break the existing triangulation and then reform a new triangulation in such a way as to preserve the Delaunay criteria. Let $S = \{T_i\}$ whose circumcircle contains the new point x_{i+1} and let $F_1, \ldots, F_p$ be the external edges of this set. Then our new mesh is given by

$$T_{i+1} = (T_i - S) \cup \{F_j, x\} \quad 1 \leq j \leq p \tag{8.27}$$

This can be expressed more easily in words. The mesh T_{i+1} contains the $i+1$ points as vertices. The new elements, $\{F_j, x\}$ are created by joining point x_{i+1} to the external edges F_j such that their surface is positive (i.e. counterclockwise numbering). Once the triangulation is broken, then the external edges make up what is know as the *insertion polygon*. Simply put, one breaks the triangulation and then connects the vertices corresponding to the external edges (the insertion polygon) to the new point x_{i+1} to create the new triangulation T_{i+1}. This procedure can be seen in 8.14.

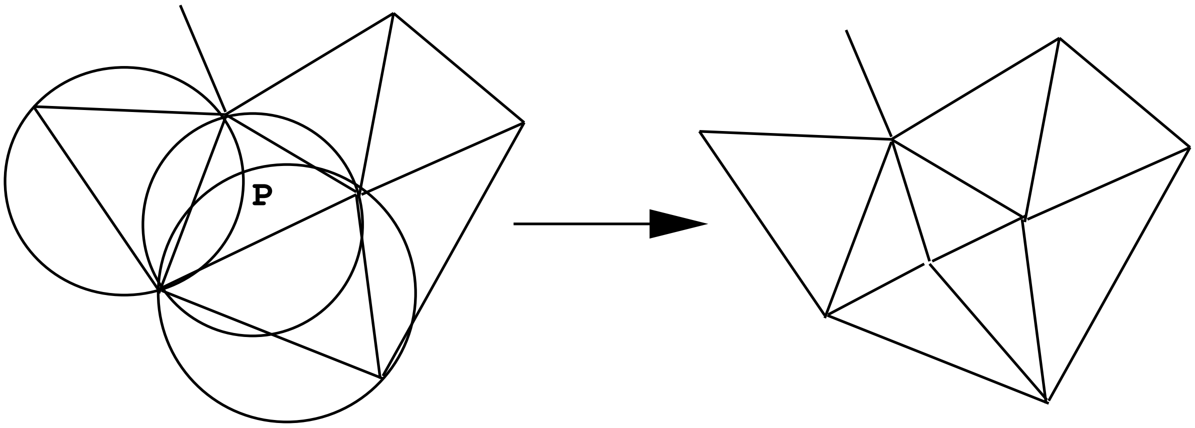

Figure 8.14: Case number 1 for inserting a point.

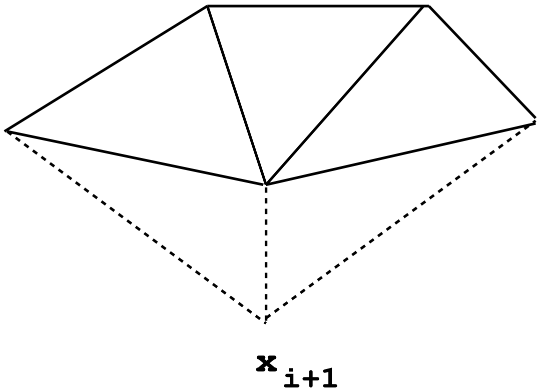

Figure 8.15: Case number 2 for inserting a point.

The next case arises when we wise to insert a point that lies outside the current triangulation. Since $x_{i+1} \notin C_i$ and $x_{i+1} \notin T_i$ we look for $F_1, \ldots, F_p$, the external edges of T_i, which define the half-plane seperating x_{i+1} and T_i. Then we create a new triangulation by

$$T_{i+1} = T_i \cup \{F_j, x\} \quad 1 \leq j \leq p \tag{8.28}$$

This means that we attach the vertices corresponding to the external edges to our new point x_{i+1} to form our new mesh T_{i+1}. This is shown in 8.15.

The third situation arises when we want to add a point that is outside the triangulation, but inside one or more of the circumcircles. Again we'll let $S = \{T_i\}$ whose circumcircle contains x_{i+1} (just as in the first case). We'll let $F_{i_1}, \ldots, F_{i_q}$ = edges not common to two elements of S defining a half-plane not separating x_{i+1} and T_i. If we let $F_{i_{q+1}}, \ldots, F_{i_p}$ be the other edges, then our new mesh is formed by

$$T_{i+1} = T_i - S \cup \{F_{i_j}, x\}_j \quad 1 \leq j \leq p \tag{8.29}$$

this says that the cicumcircle of a Delaunay triangle cannot contain another vertex in its

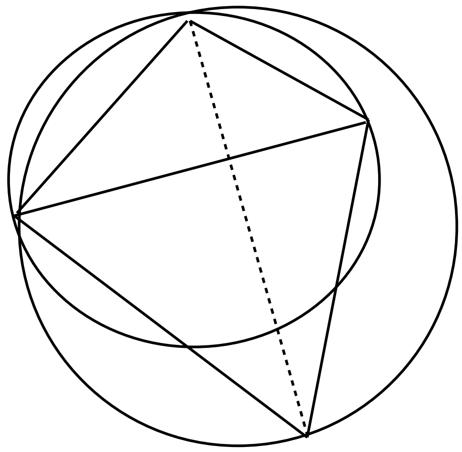

Figure 8.16: Case number 3 for inserting a point.

interior. One wold then break the exisiting triangulation and swap the diagonals as shown in 8.16.

Algorithmic concerns include:

- Searching for element(s) containing the point to be inserted.
- Searching for element(s) to be removed.
- Order of insertion.
- Respecting boundary data.
- Contruction of internal data points - if necessary.

Here is are the steps to create a Delaunay triangulation:

1. Establish a bounding box around all the points $\{x_k\}$.
2. Calculate circumcenter values.
3. Add point one by one.
4. A master list of completed Delaunay triangles is maintained which contains essential topology (node numbers, circumcircles, etc.).
5. For each new point inserted, the list is searched to identify all triangles whose cicumcircle contains the new point (note that the center of each circumcircle is the vertex of the voronoi diagram).
6. For the triangles whose circumcircle contains x_{i+1} is deleted from the list.

7. The empty polygon (the insertion polygon) vertices are connected to x_{i+1} as described before. Thenew elements are added to the list.

8. Remove all points and elments connected to the bounding box.