Chapter 2

An Overview of Mesh Generation

2.1 Introduction

Mesh generation is one of the key components in device simulation. A quality mesh not only is necessary for obtaining good simulation results, but also has a significant impact on the computation time and efficient usage of computer resources. In this chapter, we will give an overview of the various issues on mesh generation, including mesh requirements, computational algorithms and data structures. First we present a brief review on the numerical discretization process and the mathematical model for semiconductor device. Next, mesh requirements are discussed in detail, including the requirements imposed by the numerical methods, those specific to device simulation and the issues related to software engineering and computer architecture. We then introduce some mesh generation concepts in terms of computational geometry, which provide the basis for the analysis of different mesh schemes. Several classes of mesh schemes are also presented and their computational algorithms are discussed.

2.2 Numerical Simulation and Discretization

Using numerical methods to solve differential equations can be traced back to the early days of calculus in the seventeenth century. It is not until the invention of modern computers that numerical methods for differential equations came to play a significant role in engineering and science. With the rapid and revolutionary development of computers, numerical methods have become the dominant approach for the solution of PDEs in almost every engineering discipline.

The process of solving a PDE system numerically can be graphically depicted as shown in Figure 2.1 and consists of three computational phases: discretization, solution of an algebraic system and interpolation. During the discretization phase, the domain region of interest is partitioned geometrically into many small sub-regions, called mesh elements, followed by an approximation of differential operators with algebraic operators. This results in a set of linear or non-linear algebraic equations. In the second phase, numerical methods are applied to solve the algebraic system. Mesh quality plays a vital role in ensuring the accuracy and reducing computational time of these methods. The result of this phase is the solution for the system at the discrete points. Finally, an interpolation scheme is used to complete the solution for the entire problem domain.

There are two basic discretization methods for the solution of a PDE system, namely the finite difference method (FDM) and the finite element method (FEM). The finite difference method is based on the classical Taylor expansion. By replacing the differential operators with difference operators, a system of algebraic equations is obtained. The finite element method applies on the original problem in a different form, called a weak form or variational form. Instead of finding the solution in a functional space with infinite dimension, a finite functional space is used. The search for a solution in a finite functional space leads to the problem of solving a system of algebraic equations provided that the

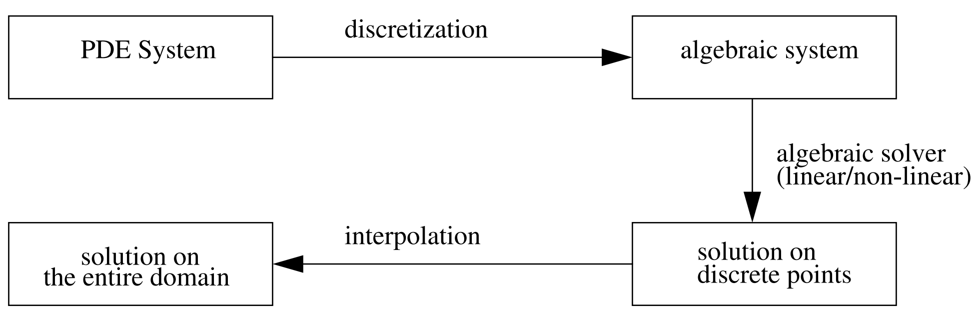


Figure 2.1 Numerical simulation flow.

basis of the functional space is known. Many numerical analysis books provide in depth discussions on these two methods, their convergence and stability properties.

For the semiconductor device problems, it has been suggested that direct application of either FDM or FEM does not always produce good results [PINT90]. The main reason is due to the non-linearity of the problem, which causes the algebraic system to be unstable. In the late sixties, Scharfetter and Gummel [SCHA69] proposed a numerical scheme, which has become the standard discretization form. For the rest of this section, we first present the drift diffusion model which is the most commonly used mathematical model for device simulation, followed by a detailed description of the discretization scheme based on the Scharfetter and Gummel approach.

2.2.1 The Drift-Diffusion Model for Semiconductor Device

Different mathematical models have been used to describe the behavior of device physics. The most commonly used model is the drift-diffusion (DD) model which consists of a system of five partial differential equations. The Poisson equation gives the relation of the potential and the concentration of electrons and holes.

$$\nabla \bullet (\varepsilon \nabla \psi) = -\rho = -q\,(p - n + N^{+} - N^{-}) \tag{2.1}$$

where ψ is the potential, p and n are holes and electrons, ε is the isotropic dielectric permittivity which is material-dependent, ρ is the total electric charge density which is equal to the sum of all electrons, holes, ionized donors (N^{+}) and acceptors (N^{-}). Two current continuity equations can be written based on the conservation law.

$$\frac{\partial n}{\partial t} = \frac{1}{q}\nabla \bullet J_n - U_n \tag{2.2}$$

$$\frac{\partial p}{\partial t} = -\frac{1}{q}\nabla \bullet J_p - U_p \tag{2.3}$$

where J_n and J_p are the current densities and U_n and U_p represent the net recombination rates. The transport relations can be stated as:

$$J_n = q\mu_n n \nabla \psi + qD_n \nabla n \tag{2.4}$$

$$J_p = q\mu_p p \nabla \psi - qD_p \nabla p \tag{2.5}$$

where μ_n and μ_p are the mobilities for electrons and holes and D_n and D_p the diffusivities.

To complete the mathematical model, we also need to impose boundary conditions on a device. Three types of boundary conditions have been used for semiconductor device simulation; they are the metal-silicon contact, insulator contact and non-contacted boundary conditions. The details on the methods and models for device simulation have been given by Pinto [PINT90].

2.2.2 Discretization for the Drift-Diffusion Model

To simplify the discussion, we assume that the problem domain is a 2D geometry region and the mesh element is of triangle type. Similar results can be said for 3D tetrahedron mesh.

To discretize the first three equations given by (2.1) - (2.3), i.e. the potential equation and the current continuity equations, we use the divergence theorem coupled with the classical finite difference method. For the transport equations, a special up-winding scheme, first introduced by Schaeffer and Gummel [SCHA69], is required to solve the system in a stable manner. The following are the five sets of algebraic systems, corresponding to (2.1) - (2.5).

$$\sum_t \left[\varepsilon^t_{k'} \frac{h^t_{k'}}{d^t_{k'}} (\psi_j - \psi_i) + \varepsilon^t_{j'} \frac{h^t_{j'}}{d^t_{j'}} (\psi_k - \psi_i) + \rho^t_i A^t_t \right] = 0 \tag{2.6}$$

$$\sum_t \left[\frac{h^t_{k'}}{d^t_{k'}} I^t_{nk'} - \frac{h^t_{j'}}{d^t_{j'}} I^t_{nj'} - qU^t_{ni} A^t_i - q\frac{\partial n_i}{\partial t} A^t_i \right] = 0 \tag{2.7}$$

$$\sum_t \left[\frac{h^t_{k'}}{d^t_{k'}} I^t_{pk'} - \frac{h^t_{j'}}{d^t_{j'}} I^t_{pj'} + qU^t_{pi} A^t_i + q\frac{\partial p_i}{\partial t} A^t_i \right] = 0 \tag{2.8}$$

$$I^t_{nk'} = q\bar{\mu}^t_{nk'} \bar{\beta}^t_{nk'} (n_j B(\Delta_{k'}) - n_i B(-\Delta_{k'})) \tag{2.9}$$

$$I^t_{pk'} = q\bar{\mu}^t_{pk'} \bar{\beta}^t_{pk'} (p_j B(\Delta_{k'}) - p_i B(-\Delta_{k'})) \tag{2.10}$$

where t is a triangle element with discrete nodes i, j and k. A^t is the area and d_i^t, d_j^t and d_k^t are the edge lengths for the triangle t (see Figure 2.2). The function $B(x)$ is the Bernoulli's function [PINT90]. The summation Σ is taken for all triangular elements in the mesh. The variables ε_i^t and ρ_i^t are the dielectric permittivity and the electric charge density at mesh point i, where

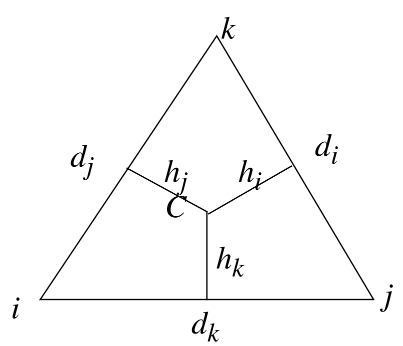
$$\rho^t_i = q\,(p^t_i - n^t_i + N^{+t}_i - N^{-t}_i) \tag{2.11}$$

The five algebraic systems of equations can be solved using non-linear algebraic methods [RAFF85].

2.3 Requirements for Mesh Generation

Mesh generation was originated in engineering and numerical analysis due to the need for solving PDE systems. In the last 20 years, the subject has attracted much attention as a domain partitioning problem in computational geometry. Many algorithms have been proposed and their computational complexity analyzed. In order for a geometry partitioning algorithm to be usable as a meshing scheme for semiconductor device simulation, many issues and requirements related to numerical methods, device physics and software engineering need to be addressed. With these requirements, we can define parameters to quantify the quality of a mesh.

For the rest of this section, we examine in detail the mesh requirements for device modeling. Although no single mesh scheme can satisfy all the requirements, an optimal



A = the area of t *d_i = the length of edge (j, k)*
d_j = the length of edge (i, k)
d_k = the length of edge (i, j)

C = origin of t's circumcircle
h_i = the distance of C to edge (j, k)
h_j = the distance of C to edge (i, k)
h_k = the distance of C to edge (i, j)

Figure 2.2 Triangular element.

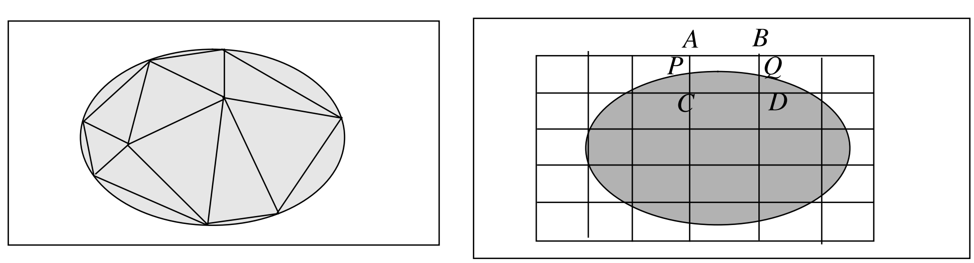
scheme that fulfills as many of these requirements can greatly increase the usability of the mesh generator. In practice, the finite difference method and the finite element method use primarily triangles and quadrilaterals in 2D and tetrahedrons and hexahedrons in 3D. In this study, only triangular elements for 2D and tetrahedral elements for 3D are considered.

2.3.1 Boundary and Mesh Conformity

Let Ω denote the geometry of a region with Γ as its boundary. A mesh is said to be *boundary conformal* (*B-conformal*) if it satisfies the condition that *no mesh edges cross the boundary*. A mesh which does not abide to this rule is called boundary non-conformal. The concept of boundary conformity is illustrated in Figure 2.3.

For a boundary conformal mesh, since no mesh edges cross the boundary, the discretization along the boundary region is quite simple and no special treatment needs to be applied. This is not the case for a non-conformal mesh. For the example in Figure 2.3 (b), the rectangular mesh element *ABCD* contains the boundary *PQ* of the geometry region. Since discretization is based on the vertices of the mesh elements, special treatment must be taken to assure that the boundary conditions are reflected in the resulting algebraic system. Most approaches that have been proposed to handle the boundary conditions of boundary non-conformal mesh are based on interpolation (explicit or implicit).

Several problems arise for non-conformal meshes. First, the original geometric region and the region after the discretization can be quite different, introducing errors into the system of the algebraic equations. Second, the errors due to interpolation can have a



(a) (b)

Figure 2.3 Boundary conformity: (a) Boundary conformal mesh.
(b) Boundary non-conformal mesh.

significant impact on the accuracy of the solution and dominate the local numerical truncation error. Lastly, special treatment may be needed in the boundary elements and complicates software design and programming. Boundary non-conformal meshes are no longer commonly used in practice.

The second type of conformity is related to the mesh elements. For 2D problem, a mesh is not conformal if one of the mesh points is an internal point of an edge. In Figure 2.4 (a), the mesh point P is the internal point of edge (P_1, P_2). The mesh is thus not a conformal mesh.

For a 3D region, a mesh is said to be non-conformal if (i) one of the mesh points is an internal point of an edge or a face or (ii) part of a mesh edge is located on a face. For example, the mesh shown in Figure 2.4(b) is non-conformal because the edge (P_1, P_3) cuts across two faces (P_1, P_2, P_4) and (P_2, P_3, P_4).

Many discretization schemes can only be applied if a mesh is conformal. For example, the discretization scheme discussed in the previous section cannot be used for a non-conformal mesh. For the rest of this thesis, we always assume that a mesh is both boundary conformal and mesh conformal.

2.3.2 Mesh Adaptation

Errors in the solution of a PDE system due to discretization, solution of the algebraic systems and interpolation are not uniformly distributed over a geometry region. An efficient way to reduce the error without introducing excessive mesh elements is to insert points selectively. In other words, more mesh points are inserted locally in the subregions with large error. An adaptive mesh refinement scheme was first formulated

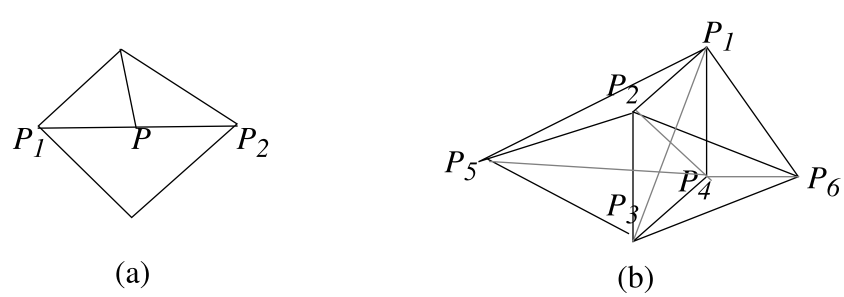


Figure 2.4 Non-conformal mesh: (a) Non-conformal mesh in 2D.
(b) Non-conformal mesh in 3D.

mathematically by Babuska and Rheinboldt [BAB78] and it has been employed often in practical numerical PDE solvers. In Figure 2.5, three new points are added for mesh element (p_1, p_2, p_3). With these new points, smaller elements are introduced to refine the original element. In addition, all three neighboring elements are sub-divided to make the final mesh conformal.

For mesh adaptation, the new mesh points to be added should be localized to certain desired subregions. Unnecessary mesh points not only waste computing resources but also may introduce interpolation error. For the structured mesh shown in Figure 2.6, a refinement in the region (p_1, p_2, p_3, p_4) causes many points to be added which may not have real contribution in reducing the local error in that region.

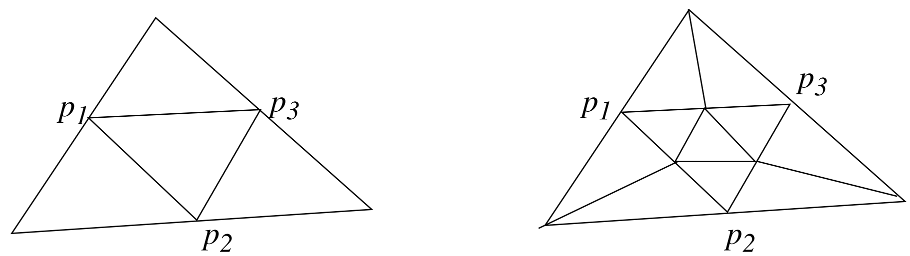


Figure 2.5 Adaptive mesh.

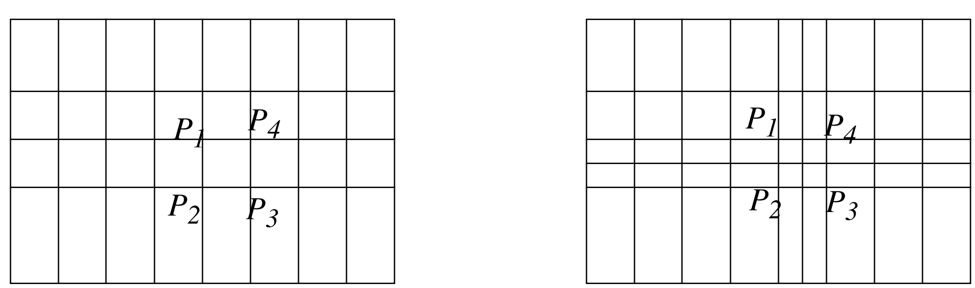


Figure 2.6 Structured mesh refinement.

Another problem in mesh adaptation is that ill-shaped mesh elements may be introduced due to the additional inserted points. This problem can be illustrated in Figure 2.7, where the mesh element (p_1, p_2, p_3) is to be refined. When the original triangle (p_1, p_2, p_3) is well-shaped, the 4 refined sub-triangles inside (p_1, p_2, p_3) will remain to be well shaped. However, the refinement also affects the three neighboring triangles. It is necessary to sub-divide the triangle (p, p_2, p_3) due to the new point p_4. The only way to achieve this without affecting the neighboring mesh elements is to connect p and p_4, which leads to an ill-shaped element (p_2, p, p_4). One scheme to overcome this shape deterioration is through edge swapping, a simple and effective technique based on Delaunay triangulation. However these types of schemes are very difficult to implement for 3D regions.

2.3.3 Element Shape

As discussed previously, mesh element shape has a significant effect on numerical simulation. Local truncation error for discretization is directly related to the minimum angle of a mesh element. Ill-shaped mesh elements affect not only the convergence of the algebraic solver but also the numerical stability. Various measures have been used to

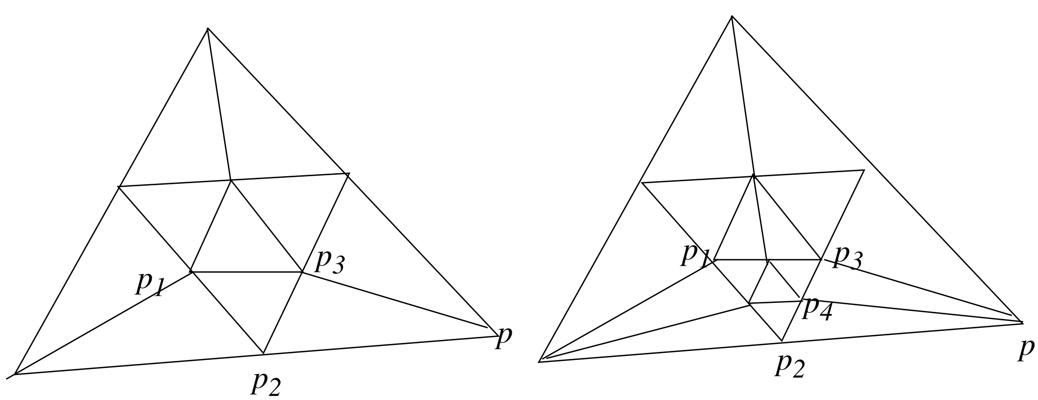
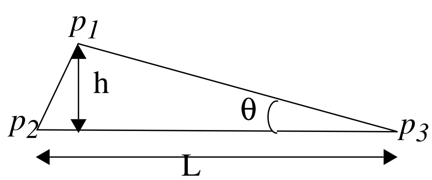


Figure 2.7 Ill-shaped elements due to mesh refinement.

quantify the shape of a mesh element. One is the smallest angle of a triangle and an alternative measure is the aspect ratio, which is defined as the ratio of the height of a triangle to the length of the longest side, as illustrated in Figure 2.8. A very important task for a mesh generation scheme is to minimize the number of ill-shaped elements.

Three approaches have been proposed to eliminate ill-shaped elements. The first approach is to create a good distribution of mesh points in the first place so that ill-shaped elements can be prevented. The second approach is to improve the shape of the mesh elements through better connection among mesh points, as illustrated by examples in Figure 2.9. The third approach is to "smooth" the mesh points after mesh elements are generated, as illustrated in Figure 2.10.



$\text{smallest\_angle}(p_1, p_2, p_3) = \theta$

$\text{aspect\_ratio}(p_1, p_2, p_3) = h/L$

Figure 2.8 Measures for the mesh element shape.

ill-shaped elements well-shaped elements

Figure 2.9 Mesh improvement by better connection of mesh points.

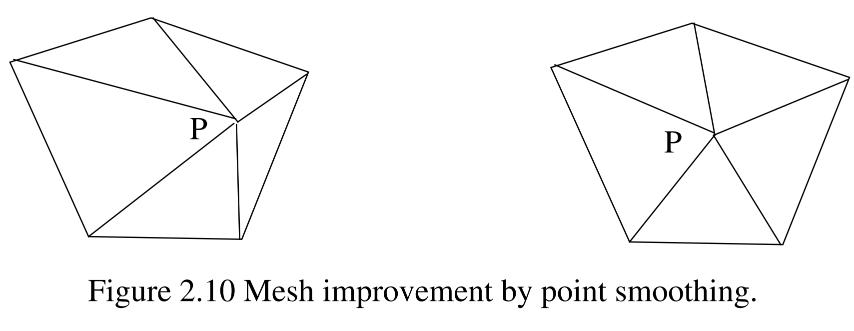


Figure 2.10 Mesh improvement by point smoothing.

Different mesh smoothing schemes which are primarily heuristic have been proposed. Two methods that are commonly employed are the Laplacian smoothing operator and the spring smoothing techniques [LOH85, BER91].

2.3.4 Mesh Density Distribution

For a CMOS device, the electric current flows near the surface of the silicon when the device is in transient state. To obtain good simulation results, more mesh points should be placed along the surface. On the other hand, excessive number of mesh points and elements could lead to expensive use of computing resources. The mesh generated should accurately reflect this phenomenon with an appropriate mesh density distribution with an appropriate mesh density distribution, as illustrated in Figure 2.11.

2.3.5 Non-Isotropic Mesh

For MOS device structures, which are very common in modern VLSI, current flows in certain direction. In order to reveal this accurately, more mesh points should be inserted along the current flow direction. Even though long narrow elements should not be generated in general, they are actually desired in this special case as illustrated Figure 2.12.

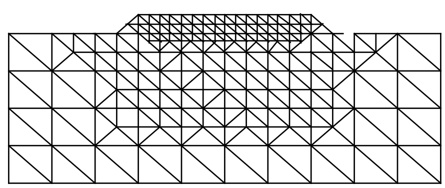


Figure 2.11 An appropriate mesh density distribution.

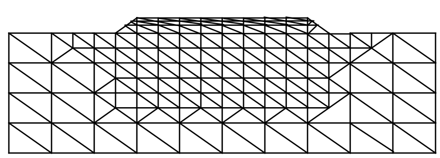


Figure 2.12 A denser mesh required along the channel.

2.3.6 Moving Boundary

Before we conclude the requirements specific for semiconductor device simulation, another requirement should be mentioned here that does not have a direct effect on device simulation. This is the problem of moving boundary, which is one of the difficult challenges in mesh generation for process simulation (see Figure 2.13).

2.3.7 Interface with Information Model and Solid Modeler

A mesh generator needs to access the information model to acquire the geometry and other properties about the device. In addition, the mesh generator should be linked or integrated with a solid modeler to take full advantage of the services provided by a solid modeler. This is especially important for 3D device structure. Many services provided by a solid modeler such as geometry operations, topological query and location query are very useful for mesh generation.

2.3.8 Integration with Device Simulators

As part of a simulation system, a mesh generator should be well integrated into the device simulation environment. To automate the simulation process, numerical results obtained from the algebraic solver should be analyzed using some error estimation schemes. The error distribution is used to refine the mesh accordingly. This feedback process should be continued until a satisfactory result is reached. To implement such a

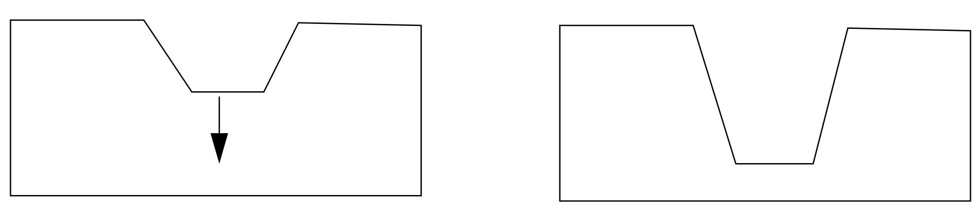


Figure 2.13 Moving boundary.

feedback loop, as shown in Figure 2.14, well defined interfaces between mesh generator, device simulator and error analyzer are indispensable.

2.4 Classification of Mesh and Mesh Generation Schemes

A mesh is a partition of geometric region into a set of non-overlapping sub-regions. Each sub-region is called an element, which is characterized by its points (also called vertices), edges and faces. Mesh elements are simple connected, convex polygons/ polyhedrons. In practice, only triangles and quadrilaterals in 2D and tetrahedrons and hexahedrons in 3D are used. Different discretization schemes may require different types of meshes. One way to classify mesh type is based on the data structure used to store the mesh information. A mesh in 1D can be described by a list of real numbers, which can be stored using an array structure. An arbitrary 3D tetrahedron mesh on the other hand would require a much more complex data structure to represent such a mesh.

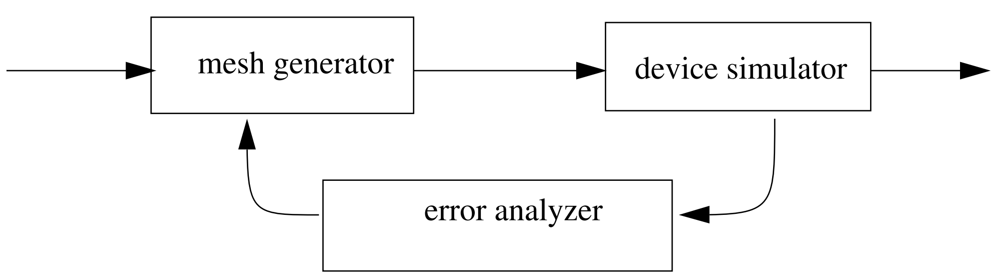


Figure 2.14 Mesh refinement with error estimation.

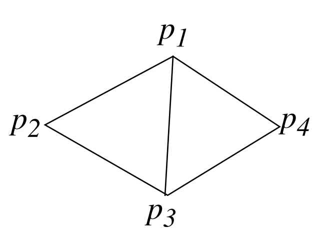


Figure 2.15 The mesh points alone are not sufficient to describe a mesh.

In this section, first we categorize different types of meshes according to the complexity of their data structure. Then a classification of mesh generation schemes is presented, followed by a brief overview of the different meshing techniques.

2.4.1 Mesh Classification

A mesh is determined by its points (sometimes also called vertices), edges and faces. Note that mesh points alone are not sufficient to describe a mesh. For example in Figure 2.15, the same mesh points p_1, p_2, p_3 and p_4 result in two different meshes because the edges are connected differently.

To fully characterize a mesh, we need to know the mesh points, edges and faces as well as relationships among these objects. This implies that a complex data structure may be required to store the information about a mesh. Much efforts have been taken in computational geometry to develop an optimal data structure for representing a mesh [GUI92]. We often find that a general complex data structure is an overkill for many mesh types, especially when a simple geometry region is involved. As shown in Figure 2.16, the region is a simple rectangle and the mesh consists of small rectangles which are organized in a very regular manner. To describe such a mesh, we only need to have a two dimensional array, one for the lines along the x-axis and the other for lines along the y-axis. If we use a general data structure to include all the mesh points and the edges as well as the relationships among the points and the edges, much storage space is wasted. Furthermore, an algorithm based on a general data structure is very likely to be complex.

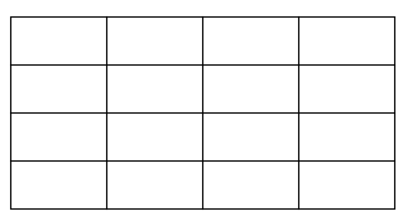
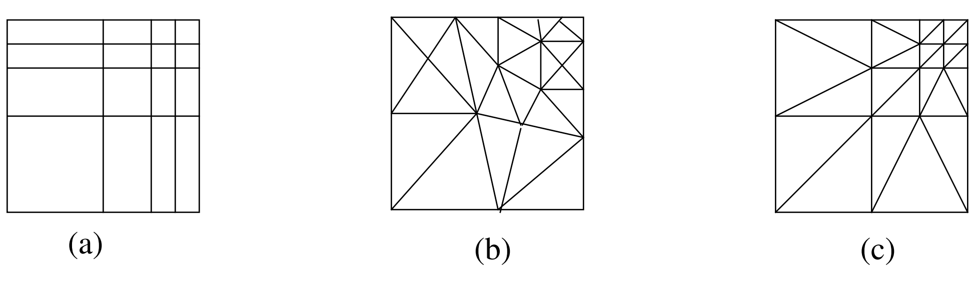


Figure 2.16 Mesh organized in a structured manner.

Before we discuss the different meshing schemes, we would like to classify the mesh types according to the complexity of the data structure required. They are the structured mesh, the unstructured mesh and the semi-structured mesh, which are depicted in Figure 2.17. Generally speaking, a structured mesh only requires an array or array-like data structure. A semi-structured mesh needs a tree or tree-like data structure. An unstructured mesh requires a most complex and general data structure.

A mesh is structured if there is a one-to-one and onto map T from $\Omega_{\xi\eta}$ (logical space) to the geometry region Ω (physical space), where $\Omega_{\xi\eta}$ is a rectangle and $T(\xi, \eta)$ are non-overlapping lines in the physical space Ω if either ξ or η is fixed. A point $T(\xi, \eta)$ is a mesh point if and only if both ξ and η are integers. A mesh point $T(\xi, \eta)$ is the neighbor of another point $T(\xi_1, \eta_1)$ if and only if $|\xi_1 - \xi| + |\eta_1 - \eta| = 1$. A mesh edge is a pair of neighboring mesh points. A quadrilateral element consists of 4 mesh points $T(\xi, \eta)$, $T(\xi+1, \eta)$, $T(\xi, \eta+1)$ and $T(\xi+1, \eta+1)$. Similar arguments can be applied to 3D structured meshes. A special case of structured mesh is a tensor product mesh where T is an affine map, i.e. $T(i, j) = (a*i+b, c*j+d)$ and a, b, c, d are real numbers. It should be pointed out that in practice, primarily quadrilaterals for 2D and hexahedrons for 3D are considered for structured mesh.



(a) (b) (c)

Figure 2.17 Mesh organized in a non-structured manner.

A mesh that requires full knowledge of points, edges and faces to describe is an unstructured mesh. The main advantage of an unstructured mesh is its flexibility to conform with the boundary. However, the data structure to accommodate an unstructured mesh is much more complex because it needs to store not only the mesh point information but also how these points are connected to form the triangles or tetrahedrons. Another disadvantage is that many geometric operations such as point location query that are useful for mesh generation process become much more difficult to implement.

A third type of mesh is the tree structure based mesh, also called semi-structured mesh. In general, the data structure for semi-structured mesh is much simpler than that needed for the unstructured mesh but retain the flexibility in boundary conformity. A major part of this thesis will be devoted to describe our approach to develop semi-structured meshes for device simulation.

2.4.2 Classification of Mesh Generation Schemes

Mesh generation is a process in which the entire geometry region is partitioned into many small elements. Each element is represented by its components, i.e. mesh points, edges and faces. Our discussion is based on grouping mesh generation schemes into three categories according to how and when these components are generated.

For the first group of meshing schemes, only mesh points are generated. Edges and faces are implied by the mesh points with some predefined computational rules. An example is a structured mesh. Later in this section, we will review two such schemes, a conformal mapping scheme and an algebraic scheme.

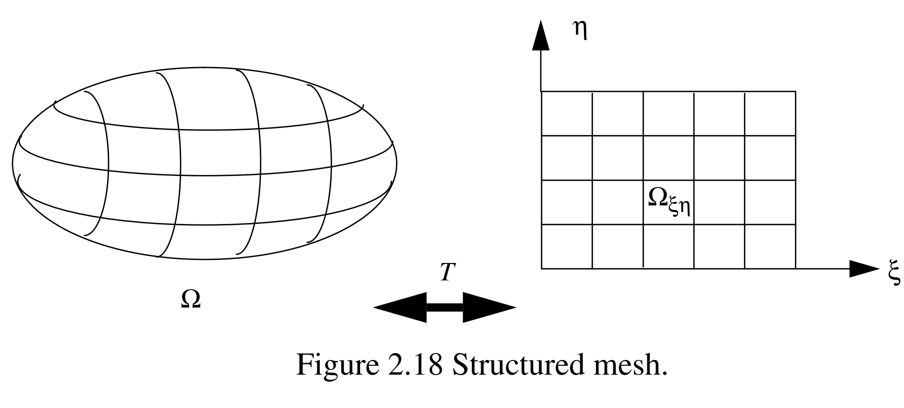


Figure 2.18 Structured mesh.

In the second group of meshing schemes, the mesh points are first generated, followed by connecting them to form edges and faces. The mesh points can be created uniformly throughout the entire geometry region or created based on some predefined density distribution. This process is illustrated in Figure 2.19. An example is the Delaunay triangulation scheme, which will be discussed later in this section.

For the third group of meshing schemes, mesh points, edges and faces are created simultaneously. This type of meshing schemes is mostly used for generating unstructured meshes [PINT90, TRI, JIN90, PER90].

2.4.3 Conformal Mapping

Conformal mapping based schemes are often used to generate structured mesh. Unlike tensor product mesh, conformal mapping schemes can handle much more complex geometry. As discussed earlier, to generate a structured mesh is equivalent to find a map T from a rectangle $\Omega_{\xi\eta}$ to the geometry region Ω . There are different ways to construct such mappings. The two that we discuss here are based on solutions of a PDE system. Let us consider a map T: (ξ, η) -> (x, y) and the map is smooth. The coordinates x and y can be considered functions of ξ and η which satisfy the following system of equations:

$$F\left(\frac{\partial^2 x}{\partial \xi^2}, \frac{\partial^2 x}{\partial \zeta \partial \eta}, \frac{\partial^2 x}{\partial \eta^2}, \frac{\partial^2 y}{\partial \xi^2}, \frac{\partial^2 y}{\partial \xi \partial \eta}, \frac{\partial^2 y}{\partial \xi \partial \eta}, \ldots\right) = 0 \tag{2.12}$$

$$G\left(\frac{\partial^2 x}{\partial \xi^2}, \frac{\partial^2 x}{\partial \zeta \partial \eta}, \frac{\partial^2 x}{\partial \eta^2}, \frac{\partial^2 y}{\partial \xi^2}, \frac{\partial^2 y}{\partial \xi \partial \eta}, \frac{\partial^2 y}{\partial \xi \partial \eta}, \ldots\right) = 0 \tag{2.13}$$

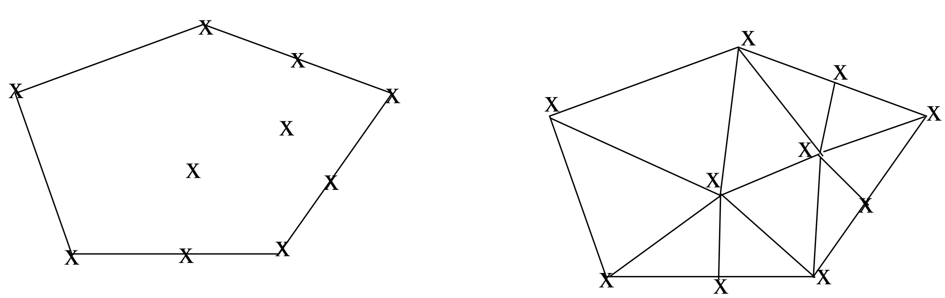


Figure 2.19 Two-step mesh scheme.

for some functions F and G. If we can find such functions F and G, then the map T can be constructed by solving the system of PDEs. There are two issues to be addressed for using this approach.

1. The system of PDEs should be well posed, i.e. the solution of the system exists and is unique;
2. The mesh defined by the solution should satisfy the desired requirements as discussed in Section 2.3.

It is well known that there exists a harmonic mapping from any simple connected region to a rectangle. In other words, there exist ξ and η as functions of x and y that satisfy the following Laplacian equations:

$$\xi_{xx} + \xi_{yy} = 0 \tag{2.14}$$

$$\eta_{xx} + \eta_{yy} = 0 \tag{2.15}$$

After applying some mathematical transformations, we can also write x and y as functions of ξ and η satisfying the following equations:

$$a x_{\xi\xi} - 2b x_{\xi\eta} + g x_{\eta\eta} = 0 \tag{2.16}$$

$$a y_{\xi\xi} - 2b y_{\xi\eta} + g y_{\eta\eta} = 0 \tag{2.17}$$

where

$$a = x_\xi^2 + y_\xi^2 \tag{2.18}$$

$$b = x_\xi x_\eta + y_\xi y_\eta \tag{2.19}$$

$$g = x_\eta^2 + y_\eta^2 \tag{2.20}$$

with the boundary conditions that maps Γ onto $\Gamma_{\xi\eta}$. Many numerical methods can be applied to solve this system. One example is the stationary and linear successive point over-relaxation scheme, as suggested by Weatherill [WEAT90].

The second method to construct the mapping functions F and G is the classical conformal mapping technique. It is well known that there exists a (unique) conformal map T which maps a simple connected region to a rectangle. After scaling, we can show that a region can be mapped to a square, where the coordinates x and y satisfy the following equations:

$$M^2 x_{\xi\xi} + x_{\eta\eta} = 0 \tag{2.21}$$

$$M^2 y_{\xi\xi} + y_{\eta\eta} = 0 \tag{2.22}$$

where M is some scaling factor. This PDE system is linear and is much simpler to solve. Different approaches have been proposed to construct M (see the discussion by Mastin in [CAS91]).

2.4.4 Algebraic Method

Another scheme that can be used to generate structured mesh is the algebraic method, in which the map T is constructed using algebraic interpolation. One advantage for such a method is efficiency. Unlike the conformal mapping scheme, algebraic methods generate a mesh without solving a PDE system. The disadvantage is its inflexibility to handle complicated geometry.

First let us explore the basic idea behind the algebraic methods. Suppose two opposite boundaries of the region Ω are given as $R_1 = (x_1(\xi), y_1(\xi))$ and $R_2 = (x_2(\xi), y_2(\xi))$ where ξ is between [0, 1]. Then we can construct a map $T(\xi, \eta)$:

$$T(\xi, \eta) = \begin{bmatrix} x1(\xi) + \eta(x2(\xi) - x1(\xi)) \\ y1(\xi) + \eta(y2(\xi) - y1(\xi)) \end{bmatrix} \tag{2.23}$$

where η is between [0, 1]. If in addition to the boundary points, derivative information is also available, then we can construct higher order interpolation to provide better smoothness. The above scheme works well if the data for the two opposite sides of the boundary are given and the other two sides are assumed to be linear. If the latter condition

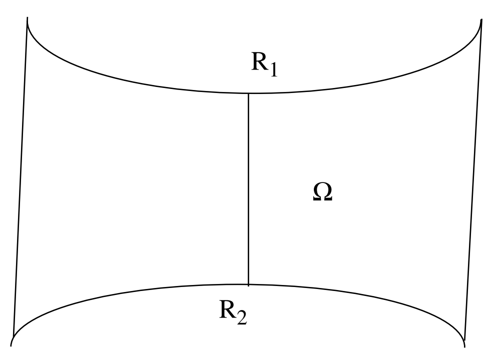


Figure 2.20 Algebraic method for generating structured mesh.

is not satisfied, i.e. the data for all four sides of the region are given, then the following more complex mapping T is used. Let's denote $R(\xi, \eta)$ be a map from R^2 to R^2. $R(0, \eta)$, $R(1, \eta)$, $R(\xi,0)$ and $R(\xi,1)$ are four boundary curves where ξ and η are between 0 and 1. The mapping function T can be constructed as following:

$$T(\xi,\eta) = \begin{array}{c} (1-\xi)R(0,\eta) + \xi R(1,\eta) + \\ (1-\eta)R(\xi,0) + \eta R(\xi,1) - \\ (1-\xi)(1-\eta)R(0,0) - \xi(1-\eta)R(1,0) - \\ (1-\xi)\eta R(1,0) - \xi\eta R(1,1) \end{array} \tag{2.24}$$

We can check that the map T is boundary conforming. This type of interpolation is called transfinite interpolation and the functions ξ, $(1–\xi)$, η and $(1–\eta)$ are called blending functions. The idea can also be extended to include derivative information on the boundary. There are many variations based on the above ideas. Further discussion on this subject can be found in [EISE90].

2.4.5 Delaunay Triangulation

Delaunay triangulation is probably one of the most interesting problems in combinatorial/computational geometry. It is also among the earliest mesh schemes to be used for generating unstructured mesh. The meshing scheme based on Delaunay triangulation is a 2-step scheme as discussed in Section 2.4.2, i.e. first generating the mesh points and then triangulation. There are many advantages in using Delaunay triangulation. First, it is suitable for generating unstructured mesh, which is very desirable for device simulation since the geometry for a semiconductor device is in general irregular. Second, an optimal mesh quality is achieved with respect to a given set of mesh points due to the maximum-minimum angle property to be discussed later. The disadvantage for such a scheme is that a very general data structure is required. In this section, we focus on the triangulation process for 2D problems. For a more detailed discussion, interested readers are referred to the textbook on computational geometry by Preparata and Shamos [PREP85].

Let us first define some terminologies used in the triangulation process. Let S be a set of N points $\{p_0,...p_{N-1}\}$ and let C be the convex hull of S.

Definition 2.1 (Voronoi Diagram): The Voronoi polygon associated with each point p_i in S is the locus of points that are closer to p_i than to any other point of S. For any point set S, the Voronoi polytopes partition the plan into a convex net, called the Voronoi diagram of S and denoted as $V(S)$. The vertices and the edges of the net are called Voronoi points and edges.

The example in Figure 2.21 shows the Voronoi polygon associated with the point set $S = \{p_i, i=0, 1,..., 7\}$. Every vertex of the Voronoi diagram is the common intersection of exactly three edges of the diagram [PREP85].

We now consider the *straight-line dual* of the Voronoi diagram, i.e., the graph embedded in the plane obtained by adding a straight-line segment between each pair of points of S whose Voronoi polygons share an edge. The result is a graph on the original N points. Figure 2.22 shows the straight-line dual of the previous Voronoi diagram.

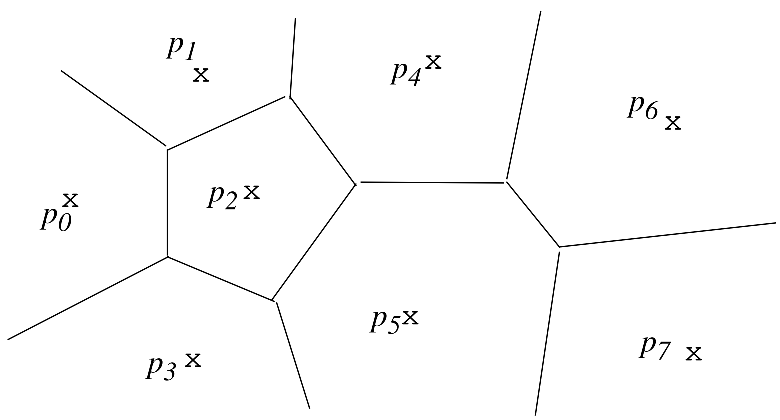


Figure 2.21 Voronoi polygons associated with each point of S.

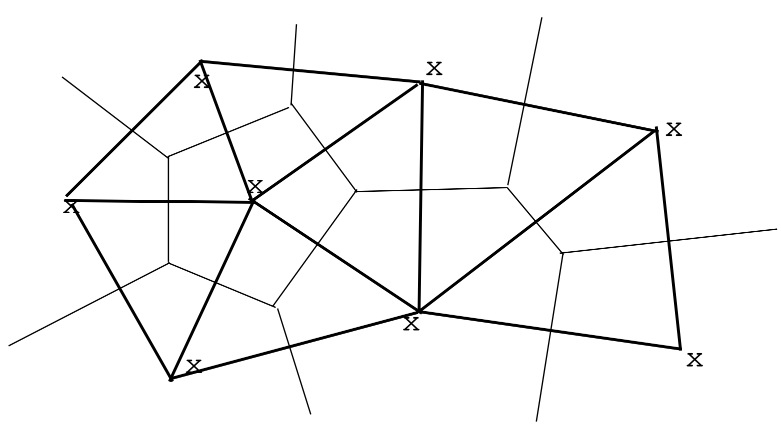


Figure 2.22 The straight-line dual.

Theorem 3.1 (Delaunay): The straight-line dual of the Voronoi diagram is a triangulation of *S*.

Delaunay triangulation has two most important properties, the in-circle property and the maximum-minimum angle property, which are stated in the following.

Definition 2.2 (in-circle): A triangulation T satisfies the in-circle criterion if and only if there are no point in the set S that are interior to the circumcircle of any triangle in T.

To illustrate the in-circle property, consider the four points p_1, p_2, p_3 and p_4, as shown in Figure 2.23. If the points p_1, p_2, p_3 form one triangle and the points p_2, p_3, p_4 the other, it can be seen that the circumscribing circle through the points p_1, p_2, p_3 encloses the fourth point p_4. Hence, this connection of points does not satisfy the in-circle criterion. However, if the triangles are formed by the points p_1, p_2, p_4 and p_1, p_3, p_4 respectively, then it is obvious that the two circumscribing circles satisfy the in-circle criterion. This property enables us to detect a non-Delaunay triangulation very easily.

Definition 2.3 (The maximum-minimum angle property): A triangulation T satisfies the maximum-minimum angle property if the diagonal of any strictly convex quadrilateral is replaced by the opposite one, the minimum of the six internal angles will not decrease.

A triangulation satisfying the maximum-minimum angle property is "optimally equivalent". This property is very important because this shows that if a mesh is

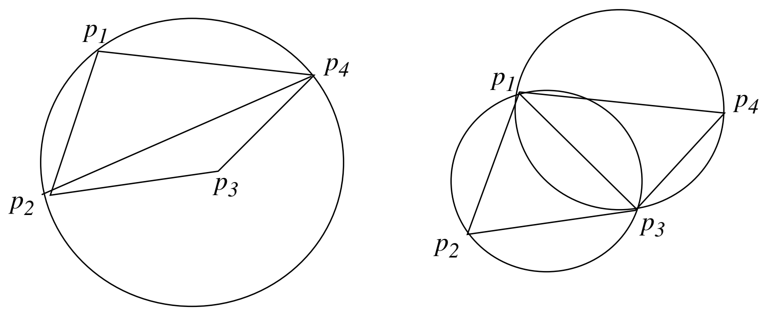


Figure 2.23 In-cycle property.

triangulated by the Delaunay procedure, then the shape of the mesh elements is optimal (see Figure 2.24).

.

We conclude this section by describing a procedure for Delaunay triangulation [PREP85]. The procedure is based on a divide-and-conquer idea. The input for the procedure is a set of points S and the output is a list of Voronoi edges *VEList.*

PROCEDURE DelaunayTriangulation(*S: EdgeSet, VEList: PolygonVertexList*)

Partition S into two subsets S_1 and S_2, of approximately equal sizes;
DelaunayTriangulation(S_1, *List1*);
DelaunayTriangulation(S_2, *List2*);
(* now need to combine *List1* and *List2* *)
Construct the polygonal chain A, separating S_1 and S_2;
Discard all edges of *List1* that lie to the right of A and call this *List1'*;
Discard all edges of *List2* that lie to the left of A and call this *List2'*;
VEList = UnionOf (List1', List2');

END DelaunayTriangulation

A polygonal chain A for points S_1 and S_2 is defined to be the set of Voronoi edges that are shared by pairs of polygons V_i and V_j of *VEList* where V_i is in S_1 and V_j is in S_2. Details on the procedure to find A for a given partition S_1 and S_2 can be found in [PREP85].

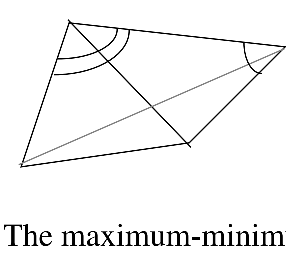


Figure 2.24 The maximum-minimum angle property.

2.4.6 Advancing Front Method

The Delaunay triangulation is a 2-step meshing scheme, which requires that the mesh points are created first. One problem in the Delaunay procedure is that we cannot always perform Delaunay triangulation and keep the mesh to be boundary conformal. For the example shown in Figure 2.25, we are given a set *S* of ten points p_i, $i=1,...,10$. It is clear that the Delaunay triangulation contains triangle (p_1, p_2, p_6) which cuts through the edge p_2 and p_3. The resulting mesh is not boundary conformal.

In this section, we review another scheme called advancing front method, which always generates a mesh which satisfies the Delaunay criterion. Advancing front method is a one step scheme, generating mesh points and edges simultaneously. This method was first introduced by Peraire et al [PERA87] and has since been extended to 3D [PERA90]. The procedure described here follows the work by Jin and Wiberg [JIN90].

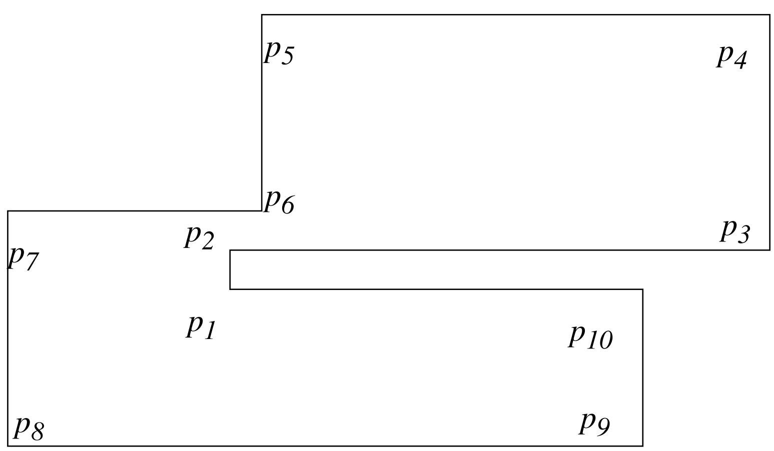


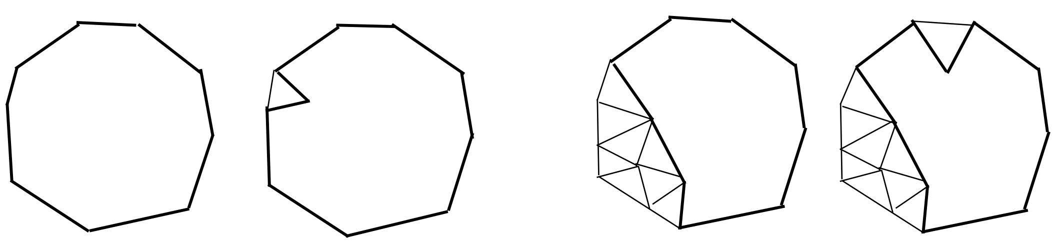
Figure 2.25 Non-boundary conformal Delaunay triangulation.

The basic idea for the advancing front method is to reduce the mesh region by subtracting triangles along the boundary. In other words, the boundary of the geometry region is treated as a wave front moving inwards, as illustrated in Figure 2.26. The initial front consists of the boundary of the geometry region (Figure 2.26 a). The shortest edge is picked among all the edges from the front to construct a new triangle. The newly constructed triangle is then deleted from the original region and a new smaller active region is formed. Finally the front is updated. This process can be summarized as follows:

PROCEDURE AdvancedFrontMethod()

(* create initial front *)
front = the boundary of the region;
WHILE (*front* is not empty) DO
Pick the shortest edge *E* from *front*;
(* based on *E* to create a new triangle *)
tri = CreateANewTriangle(E, front);
(* delete *tri* from *Front* *)
front = subtract(front, tri);
END;

END AdvancedFrontMethod



(a) (b)

Figure 2.26 Advancing front: (a) at the beginning of mesh generation. (b) during generation.

Let us now discuss the construction of a triangle based on an edge $E = (p_1, p_2)$. To construct the triangle, we need to find the third point in the triangulation region. The third point p is selected such that the isosceles triangle does not cause any conflict with the boundary and p is not too close to the boundary, as shown in Figure 2.27. The triangle size is based on the mesh density requirement near p_1 and p_2. If no such point can be found, it implies that either there is a front point which is too close to p (Figure 2.28 (a)) or p is too close to one of the front edges (Figure 2.28 (b)). In the first case, the front point can be used as the third vertex of the triangle. In the second case, the point from the front which is closest to p can be chosen as the third vertex of the triangle.

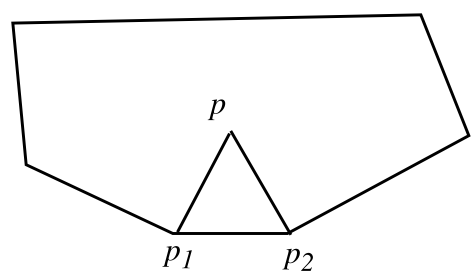
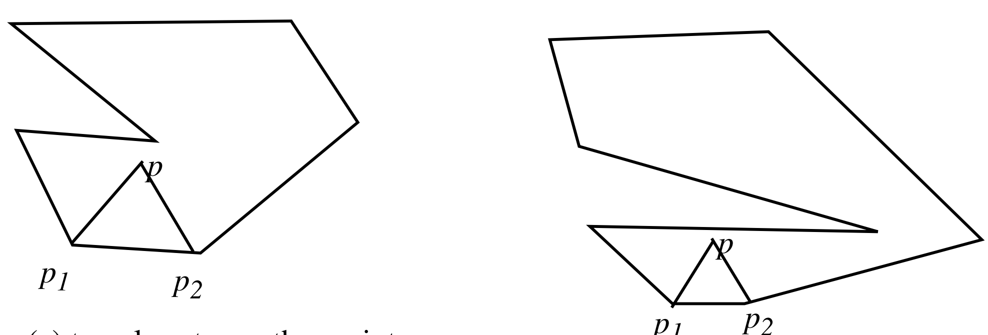


Figure 2.27 *P* is good for the third point of the triangle



(a) too close to another point

(b) too close to an edge

Figure 2.28 *P* is not good for the third point of the triangle.

2.5 Summary

In this chapter, we have reviewed the mathematical model and its discretization for semiconductor device simulation. We discussed different requirements which should be observed by a mesh generation software. Finally we discussed mesh types and different mesh generation schemes. In the next chapter, we will present a new mesh scheme, which is based on spatial decomposition with enhancements for a better control of mesh distribution and an improvement on mesh quality along the geometry boundary.