

## Chapter 1

# Basic Concepts

Mesh enhancement is best defined as the set of operations designed to improve a mesh. Improvement of a mesh means different things to different people and it involves a variety of numerical algorithms and depends on the target application that the mesh will support. Improving a mesh may mean making the mesh cells or elements as nearly uniform in volume as possible. It might also mean packing mesh cells near a particular boundary according to a specified rule in order to capture a given physical phenomenon. It might mean that variations in cell volume are smooth over the domain; that the mesh is free of isolated large or small cells surrounded by cells of approximately uniform volume.

Historically, mesh improvement has been referred to as *grid generation*. In this monograph, the phrase *mesh enhancement* is used instead, leaving grid generation to mean the operations performed on the given geometry to produce an initial mesh. It is the initial mesh that is to be enhanced through the elliptic methods described herein. The mesh improvement process is also, at times, called *mesh smoothing*. In addition, as the reader has already seen, the terms *mesh* and *grid* are used interchangeably to mean the set of cells, faces, and/or nodes that represent the discretization of physical space contained and defined by geometrical surfaces, *etc.* for use by an approximate numerical algorithm for the purpose of simulating a physical process.

Ultimately, the intent of mesh enhancement is to accomplish the following goals: improve the computability and numerical accuracy of the numerical simulation for which the mesh was initially created by modifying particular mesh characteristics. Computability refers to the ability of the physics simulation to actually compute a numerical solution for a given time-step (or other end-state). There are cases in which extremely large

aspect ratio cells or degenerate cells could cause numerical and/or algorithmic problems. Computability might be thought of as a *robustness* issue, where undesirable mesh characteristics often affect the ability of a particular simulation to proceed to completion. The goal of numerical accuracy is more obvious to the analyst; if the solution does not accurately represent the physical phenomena being modeled, the value of the simulation is uncertain. For example, in Chapter 3 it is shown that the size of cell may have a direct effect on the accuracy or the convergence rate of a computed solution.

There are many ways to achieve these two goals. One approach is to construct a set of mesh geometric criteria, such as orthogonality of cell edges and uniformity of volume of cells, and to construct a functional which is then minimized in a global sense. This approach has some value in maintaining computability of the mesh as the simulation proceeds, addressing the robustness of the mesh in an indirect fashion. A second approach employs mesh refinement or coarsening according to local volume differences or evolving solution criteria. This approach favors the goal of high numerical accuracy of the simulation on the mesh and often requires information from the solution as it proceeds. Clearly, this approach also indirectly influences numerical accuracy, as there is seldom an obvious causal relationship between mesh refinement and solution accuracy. The literature details a large number of other approaches that have some success in improving the suitability of a particular mesh for a given application. Depending on the discipline and application, this “quality improvement” is also often termed *mesh smoothing* or *quality optimization*.

It is important to formally note, however, that mesh quality improvement must be measured with respect to the actual simulation application the mesh is intended for [Molino et al., 2003a]. It is generally meaningless to discuss the improvement of a mesh in a vacuum. In fact, given two distinct applications, improving the suitability of a mesh for one might very well decrease its suitability for the second. For complex simulations, the characteristics of a mesh designed for a particular code and model are generally quite specific to the application. Alternatively, one might say that the best mesh is usually fine-tuned toward the application that will employ it. As such, the process of mesh enhancement is clearly in “the eye of the beholder.”

This book is written to detail a set of tools and technologies that are useful for particular simulation applications, specifically physical simulations involving high speed fluid flow. It is hoped that these approaches and meth-

ods have value outside this application area. This book is targeted toward summarizing a technology base that might be used for further development of the most appropriate approaches for the reader's application. To accomplish this goal, the first part of the book summarizes useful theory, methods, and approaches needed to develop elliptical mesh enhancement methods. Next, the central section of the book summarizes several historical and promising mesh smoothing and enhancement methods detailed in the literature. This presentation, in addition to describing the approaches, is used to build a basis of comparison of the methods on problems that include highly curved boundary geometry and to motivate the final sections of the book. Lastly, the remainder of the book proposes an elliptic mesh enhancement methodology based on the solution of a Laplace-Beltrami equation system on both structured and unstructured meshes.

## 1.1 The Physics Simulation Process

Mesh enhancement is but one part of the mesh generation process, which, in turn, is a part of the Numerical Physics Simulation Process.

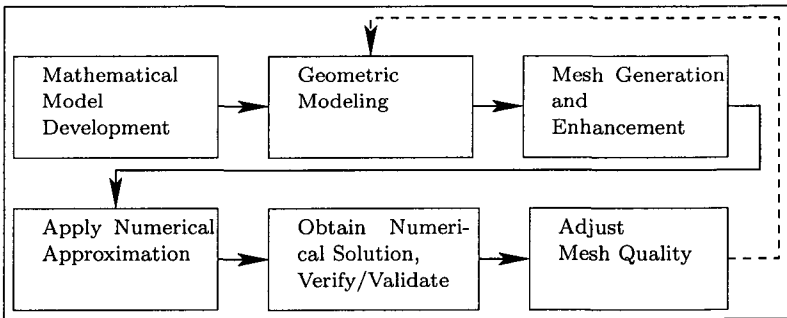


Fig. 1.1 An outline of the Numerical Physics Simulation Process.

The simulation of physical phenomena requires several individual ingredients to fit together. Figure 1.1 shows an outline of the process. The first step in the process is called Mathematical Model Development. In this step, the analyst considers the goals of the simulation and formulates a simulation process or strategy to solve the desired problem. The requirements of each stage of the process, considering the limitations, assumptions, and approximations of this approach, must be addressed. If the simulation is

somewhat routine, implementation of the mathematical model may involve the combination of analysis and engineering skills with existing software tools to perform the simulation. Depending on the particular requirements of the simulation, the process might also involve the analyst performing research activities to develop new models or approaches. In any case, for simulation applications in many fields such as fluid dynamics, heat transfer, or others that involve modeling phenomena described by systems of partial differential equations, mathematical modeling usually includes the formulation of the parent equation set, together with initial conditions and/or conditions at the boundaries of the domain being modeled.

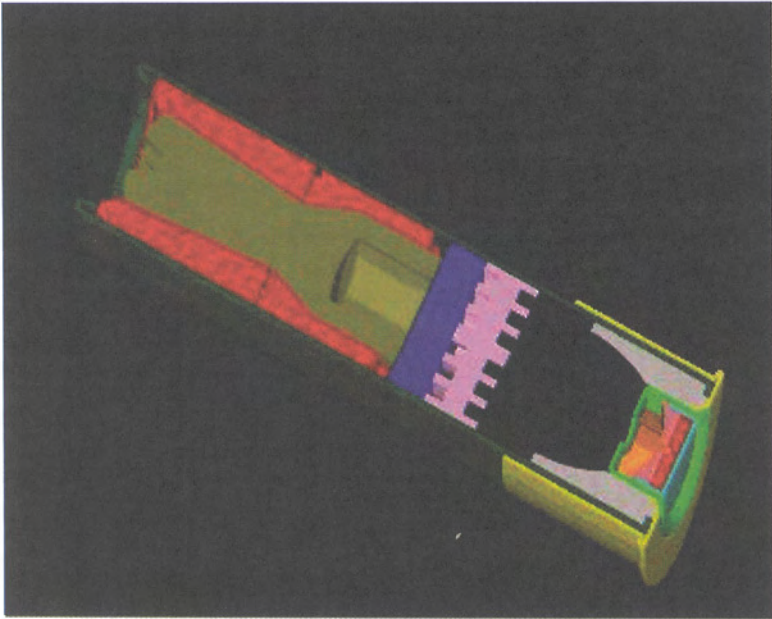


Fig. 1.2 A geometric model of a shotgun shell.

The next step is Geometric Modeling. This step encompasses the operations necessary to prepare an engineering design (or model) for numerical simulation. Generally, a mathematical model contains a definition of the region of space, and components included in this space that interact with the phenomena being analyzed. The geometric model is the description of this spatial domain, the objects that it contains, and other relationships

necessary to describe the configuration of the model. Geometric modeling includes the art of simplifying the configuration while making the assumptions necessary to maintain a tractable computational process given available resources without sacrificing the accuracy of the results. As many simulation processes are based on mathematical models that involve model subdivision, or discretization, the geometric model must also support the subdivision operations encountered later in the process. In other words, the geometric model must meet the requirements of both the discretization operation and the simulation model that uses the discretization to analyze the equations of interest.

The initial engineering representation is often in the form of a computer-aided design (CAD) model, which is a computational abstraction of the engineering components that make up the design. Figure 1.2 shows a CAD visualization of a cross-section of a shotgun shell.

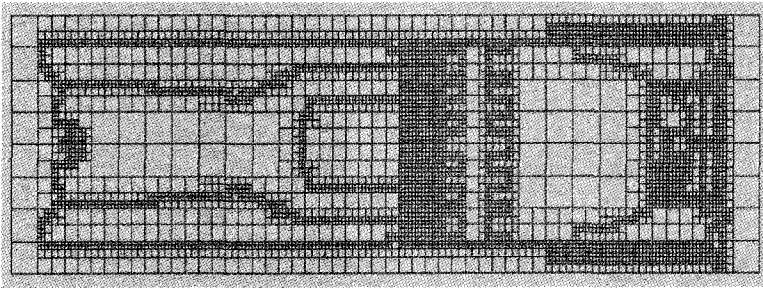


Fig. 1.3 A mesh generated on a two-dimensional simplification of the CAD-based shotgun shell model of Fig. 1.2.

Once the geometric model is complete, the discretization, or Mesh Generation step can occur. Mesh generation typically entails the placement of points that indicate subdivision locations on the curves and surfaces of the geometric model. These points, and information about how these points are connected, define the various boundaries of the computation (*i.e.*, the boundary of the model and boundaries between various components inside the model). Furthermore, this step also includes the placement of points and connectivity information between model boundaries such that the phenomena of interest may be approximated within the internal volumes or areas that form the geometric model. Figure 1.3 illustrates an example of a mesh on a simplified, two-dimensional version of the shotgun shell. If the simulation requirements warrant, three-dimensional mesh generation may

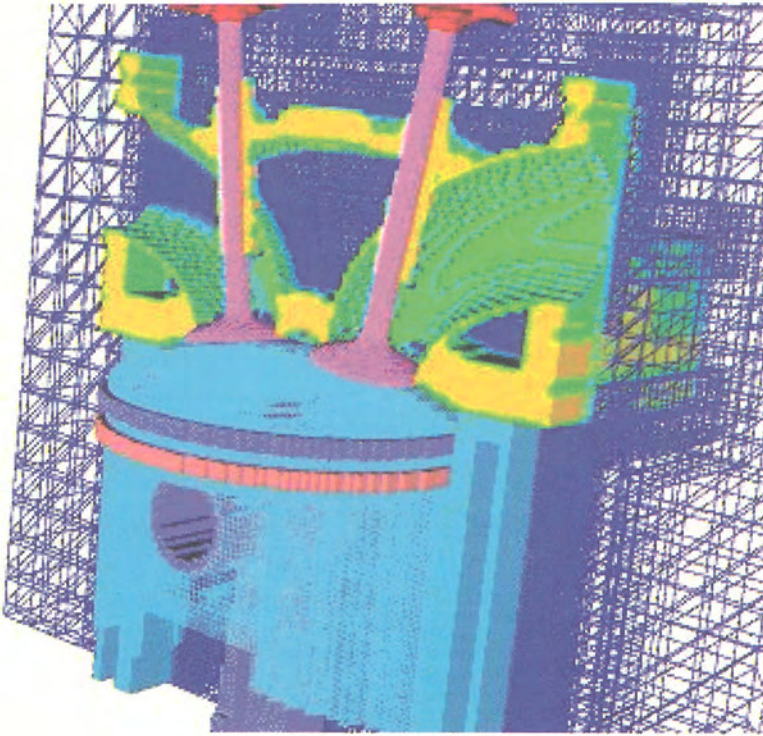


Fig. 1.4 A three-dimensional mesh for an engine showing detail associated with the piston and valves.

be required. For example, Fig. 1.4 shows a mesh for a model of the piston and cylinder area of an internal-combustion engine. As may be visually inferred based upon this discretization, three-dimensional mesh generation is often an involved process, especially if the geometric model is complex.

The solution of the approximate physics equation system on this mesh is the fourth step of the simulation process. It is here that the physics code, or application, reads the mesh and problem statement and calculates the results. This step, Apply Numerical Approximation, is closely coupled to the mesh generation process. The input mesh must meet the requirements and expectations of the simulation application, in most applications the mesh is rather closely adjusted to both the physics code and the application being studied. Furthermore, it may happen that subtle changes in the input mesh lead to substantial changes in the results from the simulation code.

Certainly, the generated mesh characteristics are usually a strong function of the application it is intended for. For example, if the finite difference method is used to approximate the parent equations, a structured mesh (*c.f.*, Section 1.3.2) must be produced, requiring a left-right, top-bottom, front-back type of mesh topology.

Finally, with the approximation in place in the physics code and the mesh defined, the algorithm and mesh are employed to obtain the physical solution. This result should be carefully examined to verify that the problem being approximated contains the desired physics and that the results are truly the solution. The simulation process does not always achieve a satisfactory conclusion on the first pass; it is often iterative in nature. In analyzing the results, modifications to the mesh (or even the mathematical model) are often required to improve mesh quality. More correctly, the results provide guidance where changes to the mesh may be employed to achieve a better result for the following iteration. These modifications often include adding or subtracting cells to better capture details within the simulation application.

The Mesh Generation Process includes two components, Mesh Generation and Enhancement. An overview of geometric modeling components necessary to support mesh generation and enhancement is introduced in Chapter 2. Mesh generation prepares an initial discretization of the geometric boundaries and the space between geometries defining the solution domain. This mesh may be sufficient as it is for the application at hand, but will likely require enhancement. Basic information on grid generation is given in Section 1.3.4. There are, of course, numerous sources for further information on mesh generation, including the extensive handbook on the subject by [Thompson et al., 1985].

The second component, mesh enhancement, is the principal subject of this book. Mesh enhancement includes the operations performed upon an existing mesh to make it more suitable for the simulation application. Depending on the nature of the generated mesh and the simulation of interest, the raw mesh from the generator may not meet the needs of the physics code. Given the complexity of the mesh generation process, it is common to have a distinct operation that generates the mesh to meet some generic set of criteria and then employ a smoothing or enhancement procedure to this result to achieve a more suitable mesh for input to the application code.

Frequently, the output of the mesh generator is not directly usable by the application. For example, algebraic mesh generation methods are often used for efficiency. These methods can result in meshes with rapid changes

of mesh characteristics over small distances within the mesh. It is usually desirable to smooth out these changes, or blend abrupt boundary features into a gently-flowing mesh structure. It is also often necessary to impart mesh characteristics that predict physical phenomena, such as refining the mesh where discontinuities or shock waves are expected in the solution or carefully fitting the mesh near boundaries where boundary layer flow results are important. In fact, for some geometric configurations, the mesh generation process may fail to create a mesh that meets the topological requirements of the application as in, for example, a mesh containing no gaps or overlaps of mesh cells in the interior of the domain. The generation of a mesh with sufficient resolution to capture boundary information, for example, may or may not be a part of the initial mesh generation algorithm. The enhancement process is expected to modify any deficiencies, incorporating or addressing expected solution requirements and removing any undesirable artifacts produced in the mesh by the mesh generation process.

There are many popular approaches detailed in the literature that could be considered effective mesh smoothing and enhancement technologies. Currently, there is no general method fully effective for every mesh type and application. In fact, as the requirements of simulation applications are so diverse, one would expect that a careful selection of the approach used to enhance the mesh for a particular application would be required. As a rule, the structure of the mesh determines the most effective enhancement approach. Meshes composed primarily of triangular and tetrahedral cells are often best served by classical shape optimization approaches [Carey, 1997; Frey and George, 2000]. A review of the literature, however, immediately reveals that smoothing, enhancement, and optimization of these meshes still remains a research topic; recent work includes several contributions in the Grid Generation Handbook [Thompson et al., 1999], mesh refinement methods [Shewchuk, 2002], aerospace and heat transfer applications [Douglass et al., 2002], to name just a few.

Viscous and high-speed flow applications often require the mesh to be aligned with or fitted to the boundary geometry, and that the mesh connectivity structure support expected directions of the simulation flow behavior. These requirements often lead to the use of structured or semi-structured meshes, or the use of unstructured quadrilateral or hexahedral elements. These application requirements, along with the idiosyncrasies of mesh generation methods producing meshes of these types, typically mandate entirely different approaches for mesh smoothing and enhancement. In this case, smoothing and enhancement methods based on the solution of

partial differential equations (PDE) systems are commonly used. These methods include variational approaches ([Knupp and Steinberg, 1994; Brackbill and Saltzman, 1982; Steinberg and Roache, 1986]), elliptic and Poisson methods ([Thompson et al., 1974; Thompson et al., 1977; Thompson et al., 1985; Mastin and Thompson, 1984; Winslow, 1963; Spekrijse, 1995]), along with a host of other approaches too numerous to list. In this book, elliptic PDE systems will be developed and used to enhance meshes. An overview of the use of elliptic partial differential equation systems for mesh smoothing and enhancement, for both structured and unstructured meshes, is covered in detail in Chapter 7.

## 1.2 Mesh Enhancement using Elliptic Methods

Elliptic methods are widely regarded as effective approaches for smoothing (or generating) structured and semi-structured meshes for computational physics applications. Qualitatively, elliptic methods have several advantages:

- They are often derived from particular mathematical criteria, such as satisfying an extremum principle, which may result in a smoothing or generating method that discourages or prevents the mesh from spilling outside concave boundary areas. Furthermore, elliptic approaches may be formulated to discourage “bow-tie” cells (non-convex computational elements).
- Elliptic methods often result in a global, equidistribution principle. They tend to behave well across the entire domain, and usually result in the sharing of suboptimal geometric artifacts with neighbor cells within the domain. Alternatively, elliptic methods tend to dissipate local non-uniformities across the domain.
- It is possible to formulate elliptic methods that allow some localized control of the mesh spacing in areas of interest, *i.e.*, along particular boundaries.

Elliptic approaches also have drawbacks, such as:

- Their application usually results in the need to solve a large, possibly nonlinear, algebraic system.
- Elliptic methods do not generally provide detailed control over geometric mesh criteria. Control or forcing functions may be employed

to adjust the mesh; however, these functions are not usually intuitive and may be problem and domain specific.

- There are only limited applications of elliptic methods to unstructured meshes.

Elliptic methods are the focus of this book. This emphasis is based upon experience in applying such methods to two- and three-dimensional problems, usually involving high-speed fluid flows. These applications are categorized as “multi-physics,” in that they are based on several governing equation systems, including but not limited to high-speed fluid dynamics, field energy, equations of state, and fluid-structure interactions. The elliptic approaches have been quite successful for multi-dimensional, multi-physics problems. The use of elliptic methods for such applications mandate the modification of existing approaches, primarily to address unstructured and semi-structured meshes on domains defined by complex boundaries. These modifications and the results thereof are detailed in this book.

From a user-interaction standpoint, there are three general categories of elliptic smoothing and mesh enhancement approaches,

- (1) Fully-automatic,
- (2) Methods that employ domain and problem specific parameters to adjust mesh characteristics, and
- (3) Methods that are locally-prescriptive.

The first class, fully-automatic methods, are developed and applied generally; without the use of any domain or application specific information. Examples of these approaches include Laplacian and Winslow-Crowley smoothing. Methods in this category are typically also called smoothers or relaxers, their primary function being to equidistribute geometric criteria (area, volume, included angle, or edge length variation) across the domain. In many cases, these approaches are fully satisfactory, depending on the requirements of the physics code and application. Their strongest advantages lie in their generality and computational efficiency.

The second class of elliptic approaches generally enhance the mesh, addressing domain requirements that extend beyond simple smoothing. Examples include methods developed by Khamayseh, Thompson, Thames, Mastin, and Warsi, who employ a set of mesh control functions which may be extracted from either geometrical or physical requirements, and/or specified directly by the user. Spekrijse also develops a similar approach, wherein a parameter space is used to develop control functions that enhance

the suitability of the final mesh.

Lastly, locally-prescriptive methods, developed by [Thompson et al., 1985] and by [Hansen et al., 2004b], are based on a comparison between a representative *ideal* element with other elements within the computational domain. This approach is further advanced here; the geometric space metric tensor is used as an aid toward formulating an elliptic system where a target metric for each element is *prescribed*. As the system is solved, the geometry of this element approaches that of the target metric through the solution of an elliptic differential equation system.

None of these methods are necessarily better than another, *per se*. Indeed, the most effective method for the specific application may not include any of the above approaches. The last two classes will generally provide better results when solution criteria must be included in the final mesh, or when boundaries are concave or complex. The second class is often superior if it is convenient to specify or develop the necessary control functions from the details of the problem. However, if this is difficult or intractable and a general method is needed, the first class may be most appropriate for some applications.

The methods developed and presented in this book approach the problem from a given, albeit fairly general, direction. It is assumed that the mesh has been constructed in some manner; *i.e.*, the space between interior or exterior boundaries has been divided into computational cells. It is additionally assumed that the number of cells is fixed, that only the nodes associated with the cells are moved in the enhancement process.

### 1.3 A Brief Primer on Meshes

In order to place the discussions in this book on a common footing, this section provides background on several topics basic to all meshes. Terminology and methodology are covered at an introductory level with sources for further reading identified. Topics in mesh quality and its meaning are also covered.

#### 1.3.1 Mesh Topology

A mesh can be thought of as being constructed from cells, which are themselves composed of faces (equivalent to edges if two-dimensional), which are defined by nodes. This hierarchy of mesh entities (*e.g.*, cells to faces

to nodes) along with the relationship between a cell and its neighboring entities defines the mesh topology. Using topological attributes allows the separation of qualitative adjacency information from the quantitative geometric data of the mesh (*c.f.*, [Carey, 1997, pp. 31–32]). Adjacency information provides answers to questions such as which cells have a given face in common, or for a given node, what is the graph of all nodes adjacent to it (on a radial-edge basis)? Geometric information pertains to coordinate and related data. The distinction between structured or unstructured meshes, then, becomes a mesh topology distinction. For example, if the neighboring nodes of a node may be inferred through a simple indexing scheme, the mesh is structured. If, however, the neighboring node information is not directly inferred, the mesh is unstructured. The next two sections discuss these classes of meshes in some detail.

### 1.3.2 Structured Meshes

Geometrically, or topologically, structured meshes are used in a broad range of physics simulations. They arise naturally in finite difference approximations (*c.f.*, Chapter 3). Structured meshes have the following traits in common:

- All cells are of similar shape (*e.g.*, rectangles in two dimensions and hexagons in three dimensions)
- Cell faces or edges in physical space may be mapped to lie parallel to coordinate axes in logical space (*e.g.*,  $(x, y) \rightarrow (\xi, \eta)$ )
- A simple data structure is used to describe the mesh
- Mesh quality deteriorates with increasing complexity of the domain and its internal geometry

To amplify upon the implications of these traits, consider structured connectivity. Structured meshes have the property that, given a node in the domain, the location of its neighboring nodes is known by a simple relationship. That is, if the node has an index value for each of the coordinate directions defining the domain such as an  $(i, j, k)$  notation, where, for example,  $i$  ranges from 1 to  $N_x$ , the number of nodes in the  $x$ -direction, then the next node in the positive  $x$ -direction is known as the  $i + 1^{\text{st}}$  node. Likewise for the  $j$ - and  $k$ -directions. Typically, the domain is discretized into cells (line segments in one dimension, areas in two dimensions, volumes in three) by connecting mid-points (or other points on lines connecting ad-

jacent nodes) and building cell faces (or edges) from those connections.

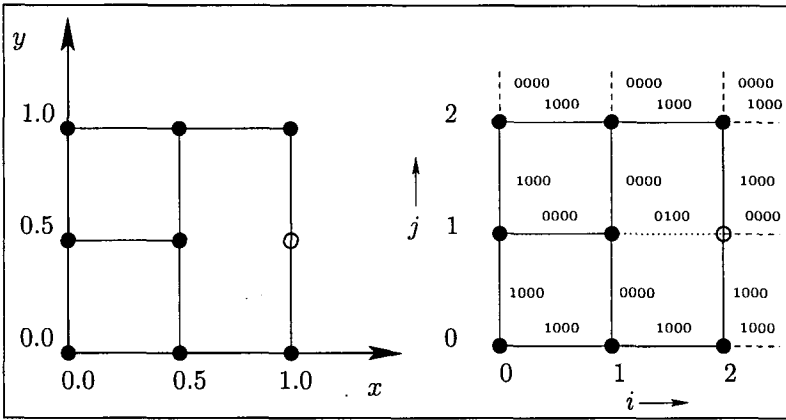


Fig. 1.5 A simple 3-cell,  $3 \times 3$ -node semi-structured mesh with terminating lines.

In addition, one characteristic used to distinguish between structured and unstructured meshes is the existence of an obvious mapping between logical space and physical space for the mesh. Physical space, of course, refers to the Cartesian coordinate space used to house the actual mesh (and geometry). Logical space refers to an equivalent domain in which the logical cell-face-node connectivity is represented. Both spaces are shown in Fig. 1.5 for an example structured mesh. If a mapping is available between these two representations, the mesh is structured. If a logical mesh cannot be created for a mesh, then the mesh is unstructured. As one might expect, however, there is not always a clear distinction between these two cases. For example, consider the two meshes in Figs. 8.2 and 8.3, in which a semi-structured mesh is shown in the converging-diverging portion of a rocket nozzle. The mesh in Fig. 8.3 is termed *semi-structured* because in large parts of the mesh, it is clearly structured. Yet, at (four) discrete locations in the mesh, there is a transition between blocks of structured sub-meshes through the use of so called *dendrites* or *terminating lines*. Each block is structured, but the whole is not. Dendritic structures commonly provide meshes whose cell volume is approximately uniform across the entire mesh. Further discussion on structured and other mesh topologies is given in Section 8.1.

The data structure for structured meshes in its simplest form assigns a unique  $(i, j, k)$ -index to each node with the coordinate values also uniquely defined for that index-set, with the added requirement that neighboring

nodes are found by an increase or decrease of unity of an index.

An example of a two-dimensional semi-structured mesh is shown in Fig. 1.5. In this figure, three cells are shown along with the  $x, y$  and  $i, j$  coordinate systems. The  $x, y$  system can be thought of as the *physical* coordinate system in which the correct physical shape of the domain is shown along with the actual coordinate values. The  $i, j$  system can be thought of as a *logical* coordinate system, showing how nodes and cells are logically connected and referenced.

The data file shown in Table 1.1 reflects but one possible data structure for storing structured meshes. The basic idea behind this data structure is that each cell is defined by attributes assigned to the lower-left hand node of the cell in the logical system. Attributes include an  $i$  and  $j$  index,  $x$  and  $y$  coordinate values and a set of flags denoting internal, boundary, or terminating edges. Flags are used to provide a mechanism for handling semi-structured, multi-block structured meshes. Any property associated with either that cell (*e.g.*, mass, density, or material description) or its edges (*e.g.*, boundary conditions) may be appended to that record of the file.

Consider the entries in Table 1.1. The first row in the data structure shows the number of nodes in the  $i$ -direction (3 in this case). Each of the next 3 rows by 3 columns of data are composed of the following attributes. First, the node's  $i$  and  $j$  indices in logical space are specified. These are followed by the node's  $x$  and  $y$  coordinate values. Finally two 4 digit integers are given, representing an edge flag. Edge flags are typically not used in structured meshes. However if an edge is missing, as in the example shown in Fig. 1.5, a distinction must exist to convey which type of edge is present. The first integer is the  $i$  edge's flag (0000 for an *internal* edge, 0100 for a *terminated* edge, and 1000 for an *external*, boundary edge). The second 4 digits are the  $j$  edge's flag. From this information, a complex mesh may be described.

A terminated edge causes a physical mesh to be created which has a cell with more than four faces. The intent with the flags given for this simple example (shown in Fig. 1.5) is that the cell containing the six nodes:  $\{(1, 0), (2, 0), (1, 1), (2, 1), (1, 2), (2, 2)\}$  should contain only 5 faces (edges). If the edge flags associated with these six nodes are examined, the edge that connects nodes (1, 1) and (2, 1) is marked as a *terminated* edge. Node (2, 1) (the hollow circle) does not exist and the edges (2, 0)-(2, 1) and (2, 1)-(2, 2) are joined into a single edge spanning (2, 0)-(2, 2). Thus, through edge flags, the transition from a finer to coarser mesh can be handled.

Table 1.1 A two-dimensional data file for a simple structured mesh.

3	3				
0	0	0.000000000000e + 00	0.000000000000e + 00	1000	1000
0	1	0.000000000000e + 00	5.000000000000e - 01	0000	1000
0	2	0.000000000000e + 00	1.000000000000e + 00	1000	0000
1	0	5.000000000000e - 01	0.000000000000e + 00	1000	0000
1	1	5.000000000000e - 01	5.000000000000e - 01	0100	0000
1	2	5.000000000000e - 01	1.000000000000e + 00	1000	0000
2	0	1.000000000000e + 00	0.000000000000e + 00	1000	1000
2	1	1.000000000000e + 00	5.000000000000e - 01	0000	1000
2	2	1.000000000000e + 00	1.000000000000e + 00	1000	0000

### 1.3.3 Unstructured Meshes

Unstructured meshes are common to finite element solution methods and to those methods based upon a Lagrangian frame of reference (in which a given, discrete mass of material is followed). There is no presumed order to the mesh so that, given a cell, there is no inherent presumption as to which cells are adjacent to it and no obvious logical relationship to the physical mesh. Simply stated, a cell can be viewed as being composed of bounding faces (or edges in two-dimensions, or points in one dimension) which are, in turn composed of nodes. To complete the topology of the mesh, it is then necessary to have additional information about the cells to which a face is common. There are many ways to define a data structure depending upon specific needs of the problem being simulated or if the problem is to be run in series or parallel. Some of these are discussed in the article of [Beall and Shephard, 1997].

Unstructured meshes have the following properties in common

- Cells have variable local topology and size
- Cell faces or edges do not have an implied orientation
- Generally are more difficult to generate
- The data structure is complex
- Mesh quality remains high as the domain complexity increases

In Section 3.4 (introducing the finite element method), the element has nodal locations at which the element interpolation may be performed exactly. The spatial coordinates of these points must be stored. Moving to the global algebraic problem from the local element requires knowledge of which elements are sharing a given element's nodes. Each element may have its own local nodal numbering scheme, say  $\{1, 2, 3\}$  for a triangular

cell, while the global problem can have a large number of nodes if there are a large number of elements contained within the mesh. The net effect is that a mapping must exist relating local node numbers to global node numbers. As a result, it is not uncommon to renumber the nodes in an initial global nodal assignment, minimizing bandwidth in the resulting algebra problem. Several node renumbering algorithms have been developed, including methods described in [Rosen, 1968] or by [Cuthill and McKee, 1969]. Others are mentioned in [Beall and Shephard, 1997].

### 1.3.4 *Basic Grid Generation Algorithms*

Given the geometry representing a computational domain, one may construct a mesh on that geometry. This mesh may be suitable as is or it may be intended as an initial condition for further enhancement. There are many effective grid generation approaches, depending on the requirements of the simulation. A few of these are discussed in [Carey, 1997], [Thompson et al., 1999], and [Frey and George, 2000] and are summarized below.

Methods for generating grids developed incrementally, beginning with the manual prescription of nodal data. Today, the state of mesh generation employs rather complex geometric and analytic methods. Though the classification of grid generation methods is necessarily arbitrary, the overview of different techniques in the monograph [Frey and George, 2000] provides a convenient point of discussion. Based on the intrinsic properties of mesh generation methods, Frey and George list the following five categories

- Manual or semi-automatic methods. These are often straightforward techniques applicable to simple domains; the element and nodal features are input sequentially as array data.
- Mapping methods. The mesh is obtained as a mapping from a parameter space to the appropriate physical space, resulting in a structured grid. Two main approaches belonging to this class are:
  - Algebraic interpolation methods, such as transfinite interpolation.
  - Solution-based methods based on numerical solution to partial differential equations for the grid coordinates.
- Domain decomposition methods where the domain to be discretized is split into smaller subdomains. Two main approaches are:
  - Block decomposition. The domain is subdivided into several subregions. Then, the mesh for each subregion is generated indepen-

dently by a (semi-)structured method such as transfinite interpolation.

- Spatial decomposition. The domain is partitioned into a collection of cells that are further decomposed into mesh elements. Quadtree and octree techniques belong to this class.
- Element creation methods. Given a discretized domain boundary, new elements or nodes are created by advancing toward the domain interior. Advancing front and Delaunay methods are representative of this class.
- Constructive methods. This method relies on merging the meshes obtained by any of the above listed methods.

A detailed description of the methods listed in the classification scheme of Frey and George may be found in the literature. The remainder of this section provides an outline of various methods that have been used to generate the grids shown later in this book. It also summarizes approaches such as automatic mesh generation from solid geometry [Field, 1995], the theory and practice of unstructured mesh generation [Teng and Wong, 2000], and a reflection on the state of grid generation in the 1990's [Thompson, 1996].

#### 1.3.4.1 Transfinite Interpolation Method (TFI)

In two dimensions, the transfinite interpolation (TFI) method is a mapping of a unit square  $\Omega_\xi = \{0 \leq \xi \leq 1, 0 \leq \eta \leq 1\}$  in parameter (logical) space onto a domain  $\Omega_{\mathbf{x}}$  in physical space  $\mathbf{x} = (x, y)$ . Consequently, four distinct, connected geometries (*i.e.*, curves) must exist to define  $\Omega_{\mathbf{x}}$ . If the bottom, top, left, and right boundaries of  $\Omega_{\mathbf{x}}$  are described, respectively, by the parametric equations

$$\begin{aligned} \mathbf{x}_b(\xi), \quad \mathbf{x}_t(\xi), \quad 0 \leq \xi \leq 1, \\ \mathbf{x}_l(\eta), \quad \mathbf{x}_r(\eta), \quad 0 \leq \eta \leq 1, \end{aligned}$$

then the boundaries of the square  $\Omega_\xi$  are mapped onto the boundaries of  $\Omega_{\mathbf{x}}$ . Explicit transformation formulae, satisfying the consistency conditions [Knupp and Steinberg, 1994], are

$$\begin{aligned} \mathbf{x}(\xi, \eta) = & (1 - \eta)\mathbf{x}_b(\xi) + \eta\mathbf{x}_t(\xi) + (1 - \xi)\mathbf{x}_l(\eta) + \xi\mathbf{x}_r(\eta) \\ & - [\xi\eta \mathbf{x}_t(1) + \xi(1 - \eta) \mathbf{x}_b(1) \\ & + \eta(1 - \xi) \mathbf{x}_l(0) + (1 - \xi)(1 - \eta) \mathbf{x}_b(0)]. \end{aligned}$$

The grid is then generated by defining the discrete coordinates

$$\xi_i = \frac{(i-1)}{(N_\xi-1)}, \quad i = 1, \dots, N_\xi$$

$$\eta_j = \frac{(j-1)}{(N_\eta-1)}, \quad j = 1, \dots, N_\eta$$

which creates a  $N_\xi \times N_\eta$  node structured mesh, including nodes on the boundaries. The physical coordinates of the nodes are

$$\mathbf{x}_{ij} = \mathbf{x}(\xi_i, \eta_j)$$

Transfinite interpolation, described in more detail in Sections 8.1 and 8.1.1, is used to generate most of the initial meshes considered in this book.

#### 1.3.4.2 Solution-Based Methods

Grid generation methods based on solving a system of partial differential equations can be traced back to the generation of a natural coordinate system, in which the coordinate lines coincide with all the boundaries [Thompson et al., 1974]. A general method of generating such a coordinate system is to let the coordinates be solutions of an elliptic partial differential equation system in the physical domain, with Dirichlet (fixed nodal coordinates) boundary conditions at the boundaries. Non-elliptic grid generators are reviewed by [Knupp and Steinberg, 1994] and [Frey and George, 2000].

In an article devoted to the solution of a quasi-linear Poisson equation in a non-uniform mesh composed of triangle elements, [Winslow, 1966] mapped the mesh from physical space  $(x, y)$  into parameter space  $(\xi, \eta)$ . The *Winslow mapping*:  $\xi = \xi(x, y), \eta = \eta(x, y)$  satisfies the Laplace equations

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

$$\eta_{xx} + \eta_{yy} = 0,$$

on a convex parameter domain. Here the subscript notation designates appropriate partial derivatives. When these equations are inverted and expressed in a logical space, the system becomes

$$\begin{aligned}
g_{22} x_{\xi\xi} - 2g_{12} x_{\xi\eta} + g_{11} x_{\eta\eta} &= 0 \\
g_{22} y_{\xi\xi} - 2g_{12} y_{\xi\eta} + g_{11} y_{\eta\eta} &= 0.
\end{aligned}$$

In this system, the components of the metric tensor are defined as

$$\begin{aligned}
g_{11} &= x_{\xi}^2 + y_{\xi}^2, \\
g_{12} &= x_{\xi}x_{\eta} + y_{\xi}y_{\eta}, \\
g_{22} &= x_{\eta}^2 + y_{\eta}^2.
\end{aligned}$$

The Winslow system can be generalized further to provide more control over characteristics of the final mesh. The popular Thompson-Thames-Mastin (TTM) mesh generator is but one example (*c.f.*, Section 8.2.1). When applied to unstructured meshes, the TTM grid generator and its extensions will form the basis of mesh smoothing methods advanced in this book. These methods will be re-derived in Section 8.2.1 within the context of harmonic coordinates.

#### 1.3.4.3 *Multiblock Method*

In fluid dynamics applications, the region of interest is often decomposed into blocks fitting together in an unstructured manner. Within each block the mesh is logically rectangular, described as a mapping from a global parameter coordinate space. Depending upon the application at hand, the mesh lines adjacent to the block boundary may be joined in a continuous or discontinuous fashion. If the lines are joined in a continuous manner, the mesh is called a composite multiblock grid.

#### 1.3.4.4 *Advancing Front Method*

This method, reviewed in [Peraire et al., 1999], always leads to unstructured grids. The advancing front algorithm first discretizes the boundary geometry. In two dimensions, the boundary is replaced by a collection of contiguous line segments. In three dimensions, the boundary surface is typically triangulated to approximate the surface contour. This discretized boundary is called the initial front. The method then proceeds in steps, building new triangles or tetrahedra at each step, using the existing edges or faces as a base from which to propagate. In two dimensions, at each step, the algorithm branches by either constructing a new vertex followed

by a new triangle or building a new triangle using an existing vertex. The branch selection is based on a “figure of merit.” Usually, construction using existing nodes is favored, unless the figure of merit of the triangle formed by the nodes of the base and existing nodes (within a sphere of influence from the base) is below a predefined threshold. In the latter case, a new node is constructed. Figure 1.6 shows the advancing front construction at the stage where 35 triangles have been added to the base triangles of two circles. The complete mesh and its smoothed counterpart are shown in Section 9.2.6. An adaptive scheme for the advancing front method in three dimensions was studied by [Seveno, 1997].

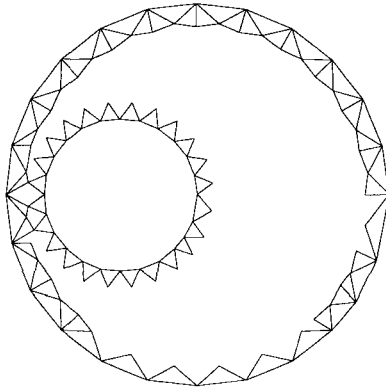


Fig. 1.6 Advancing front with 35 triangles added to the base triangles.

#### 1.3.4.5 *Delaunay Triangulation Method*

The Delaunay triangulation method [Carey, 1997] is closely related to the concept of a Voronoi diagram. Geometrically speaking, the set of all points closer to a given point than to all other points in the set is called the Voronoi polygon for the point. The union of all Voronoi polygons for a point set is called its Voronoi diagram [O’Rourke, 1998; Sedgewick, 1992]. The dual of the Voronoi diagram has lines drawn between each point and all the points closest to it. This is called the Delaunay triangulation. A simple example illustrating these definitions is shown in Fig. 1.7, where the Delaunay triangulation is shown together with the Voronoi diagram of a set of 9 points. In this example, the Delaunay triangulation is built within a large triangle whose

vertices are infinitely far away from the selected points. This technique (<http://www.cs.cornell.edu/Info/People/chew/Delaunay.html>) makes the algorithm simpler as additional treatment is needed to handle new points that are outside the convex hull of polygons composed of previous points.

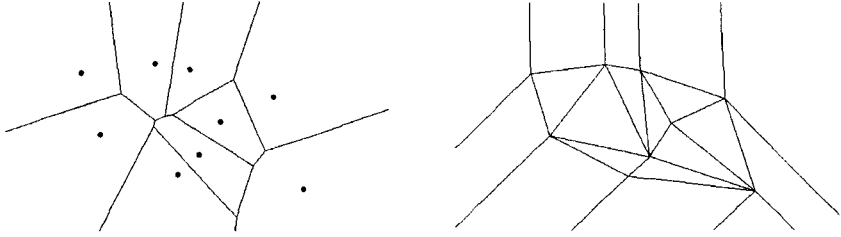


Fig. 1.7 Voronoi diagram of a set of 9 points and the corresponding Delaunay triangulation.

A Delaunay mesh generally starts with a boundary discretization, followed by the creation of a set of vertices on which the Delaunay algorithm is applied. This method, applicable both to two and three dimensions, has several variations. Two examples are constrained and conforming Delaunay triangulations. The constrained triangulation method accommodates the boundary description by introducing predefined edges that must be present in the final triangulation. This requirement may lead to a grid that is not truly Delaunay. A conforming triangulation is a true Delaunay result in which each additional segment may have been subdivided into several edges by the insertion of additional vertices. These additional vertices are called Steiner points.

The application of Voronoi diagram techniques to the generation of quadrilateral meshes is discussed in [Bern and Eppstein, 1997]. Chen and Bishop investigate the Delaunay triangulation method on curved surfaces [Chen and Bishop, 1997].

#### 1.3.4.6 *Quadtree-Octree Methods*

When applied to mesh generation, tree methods partition the region to be meshed into a union of cells having, in general, unequal sizes. These cells are obtained from a recursive refinement of a root cell, which forms a bounding box. As a result, a tree-based grid, like the quadtree grid in two dimensions and octree grid in three dimensions, becomes a hybrid semi-structured grid (Figs. 1.8 and 1.9). Tree-based grids have repeating geometric structure, but there is no rigid connectivity.

Since the tree structure is built from axis-oriented elements, the benefits of accurate spatial derivatives and high-order cancellation can be employed to advantage in fluid dynamics computations. On the other hand, the large number of terminating lines at the partitions may prevent one from conveniently describing the mesh using a global mapping. In most cases, a tree mesh cannot easily be described as a structured mesh.

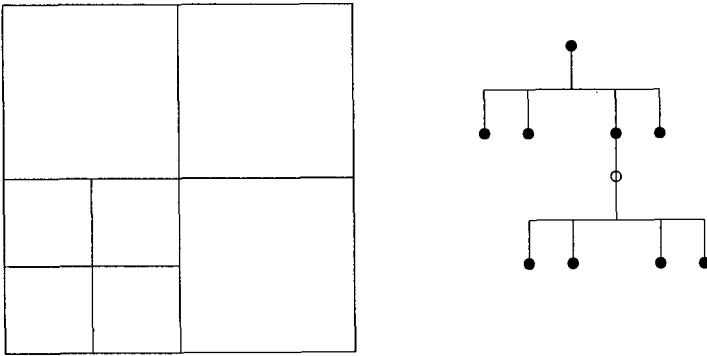


Fig. 1.8 A two-dimensional quadtree data structure. Two levels of refinement are shown.

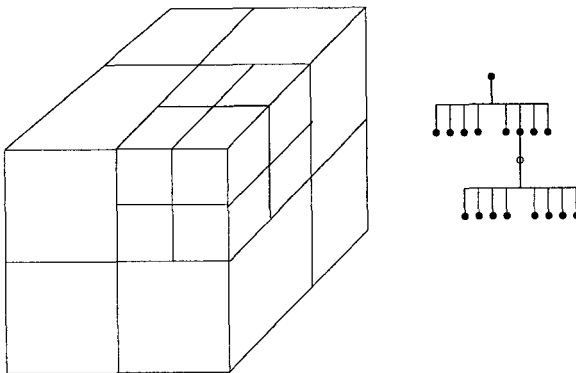


Fig. 1.9 A three-dimensional octree data structure. Two levels of refinement are shown.

### 1.3.5 Mesh Quality Issues

There is a growing body of literature that is devoted to defining and assessing the quality of a particular mesh (*c.f.*, Sections 8.1 and 8.1.2). If one is interested in selecting the best smoothing or enhancement method for a particular mesh or the application of interest, it would be most convenient to have one or more quality metrics that could be used to compare various approaches with an eye on selecting the superior method. Unfortunately, it is usually difficult to select an appropriate set of metrics to allow one to decide if one mesh is better than another for a particular application.

Mesh quality often begins by measuring and comparing convenient geometric criteria, such as norms that provide a mean cell included angle, major/minor edge length variation, variation of cell areas (or volumes), aspect ratio variation, *etc.* Using these criteria, it is straightforward to generate, for example, a triangular mesh around an airfoil having certain geometric quality metrics, such as aspect ratio, interior angle, *etc.* This mesh, however, is probably not very useful if the application is to calculate fluid-structure interactions between the airflow and airfoil. This application would require the accurate computation of the thin boundary layer flow around the wing; uniform triangulated meshes are generally not fully effective in this case. Clearly, mesh quality is an attribute shared between the mesh and the simulation code using it. Furthermore, it is usually also relative to the particular simulation being performed.

As such, mesh quality is not a mesh property; given two distinct meshes, one is likely to perform better with a particular physics code on a particular application than the other. The better mesh is usually better aligned with the assumptions codified within the physics code, which best captures the phenomena of interest within the problem domain. Said another way, a particular mesh is of high quality when its use by a particular physics code in the problem of interest results in an answer of acceptable accuracy and validity. Indeed, this is a rather subjective topic, one not lending itself to a quantitative discussion.

Experience with physical applications shows that the set of geometric metrics mentioned above do indirectly measure the quality of the results from the physics code on a given application. However, many other issues are equally as important, such as:

- The average cell size, and/or the number of cells within a domain or subdomain.

- The cell size distribution; it is generally more effective to place smaller cells in areas of high gradient within the domain.
- Structure of the mesh and how well the mesh fits the boundaries of the domain.
- Alignment of the mesh relative to the flow direction of the problem. Approximation errors may be high when the fluid flux is diagonal, across cell corners.
- Boundary interface impedance on fluid-structure interaction problems; best results may be obtained if the mass contained within cells is comparable on both sides of a solid interface.
- Mesh smoothness; minimal variation in mesh geometric properties across a subdomain.
- Connectivity of the mesh; particular applications may prefer particular cell arrangements in particular areas of the domain.
- Cell type and structure; many fluid dynamics applications prefer quadrilateral or hexahedral elements over triangular or tetrahedral cells.

The discussions within this book are targeted at methods and approaches of enhancing meshes in a general sense, using elliptic methods. The discussion of particular applications and physics codes are significantly beyond our scope; applications will be discussed only in passing. As such, it will not be possible to present credible discussions or results on the topic of mesh quality or optimization. However, this book does seek to compare various methods of enhancement. The approach used will be limited to studying metrics related to the survivability of particular meshes when used in a generic physics code. For example, it is often the case that smoother meshes provide more uniform results. Toward this goal, a smoothness functional is employed to compare different mesh enhancement approaches. In the case of two-dimensional meshes, it is often sufficient to visually compare two meshes, especially if the results are significantly different. If the bounding geometry is carefully selected to be a representation of that expected within the eventual simulation and if one enhancement approach results in a mesh that conforms to the boundary better than another approach, the first method is likely superior to the second for that application. It is acknowledged that these types of comparison metrics may not be particularly useful for the reader's particular interests or needs. For these cases, this book attempts to provide sufficient information so that the reader may examine the approach to assess its performance on his selected metric set.