CHAPTER

TEN

GRID GENERATION

10.1 INTRODUCTION

One of the first steps in computing a numerical solution to the equations that describe a physical process is the construction of a grid. The physical domain must be covered with a mesh, so that discrete volumes or elements are identified where the conservation laws can be applied. A well-constructed grid greatly improves the quality of the solution, and conversely, a poorly constructed grid is a major contributor to a poor result. In many applications, difficulties with numerical simulations can be traced to poor grid quality. For example, the lack of convergence to a desired level is often a result of poor grid quality. In this chapter, techniques for generating grids using both structured and unstructured approaches will be discussed. Since grid generation is a very large field, only the basic ideas of a limited number of methods will be presented.

Structured grid generation can be thought of as being composed of three categories:

1. Complex variable methods
2. Algebraic methods
3. Differential equation techniques

Complex variable techniques have the advantage that the transformations used are analytic or partially analytic, as opposed to those methods that are entirely numerical. Unfortunately, they are restricted to two dimensions. For this reason,

the technique has limited applicability and will not be covered here. For details of the application of complex variable methods, the works by Churchill (1948), Moretti (1979), Davis (1979), and Ives (1982) should be consulted. Algebraic and differential equation techniques can be used on complicated three-dimensional (3-D) problems. Of the structured methods, these have received the most use and will be discussed in this chapter. Unstructured methods are primarily based on using triangular or prismatic elements, although recently, randomly shaped cells with arbitrary connectivity have been increasingly used in flow simulations. Unstructured grid generation schemes may be thought of as being divided into three groups:

1. Point insertion schemes
2. Advancing front methods
3. Domain decomposition techniques

Details of point insertion methods will be discussed. However, only the procedure for constructing grids satisfying the Delaunay (1934) criterion using point insertion will be given. The detailed description of advancing front schemes or those using arbitrarily shaped cells is beyond the scope of this text. Domain decomposition techniques rely on recursively subdividing domains to provide a cell structure that may be used to complete a field calculation. These methods will not be discussed in detail in this chapter, but some information on the basic idea will be included.

Early work using finite-difference methods was restricted to problems where suitable coordinate systems could be selected in order to solve the governing equations in that base system. As experience in computing solutions for complex flow fields was gained, general mappings were employed to transform the physical plane into a computational domain. Numerous advantages accrue when this procedure is followed. For example, the body surface can be selected as a boundary in the computational plane, permitting easy application of surface boundary conditions. In general, transformations are used that lead to a uniformly spaced grid in the computational plane, while points in physical space may be unequally spaced. This situation is shown in Fig. 10.1. When this procedure is used, it is necessary to include the metrics of the mapping in the differential equations.

In applying finite-volume methods to the solution of physical problems, the direct application of the conservation statement to elements in the physical plane can be made without transforming the original differential equations. As we have seen in previous chapters, the discrete equations may be generated with this procedure and the metrics that appear with the generalized mapping now appear through the direct use of the volumes and the surface areas of the cell faces. With either approach, the physical domain is divided into volumes or cells. Techniques for creating the cell or element structure forms the basis of grid generation.

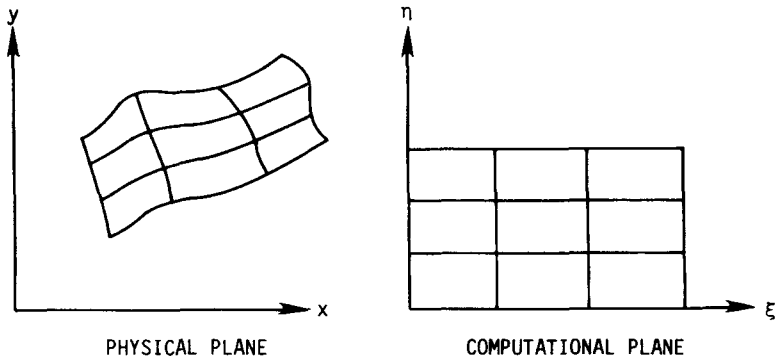

Figure 10.1 Mapping to computational space.

10.2 ALGEBRAIC METHODS

In Section 5.6 we used algebraic expressions to cluster grid points near solid boundaries to provide adequate resolution of the viscous boundary layer. In another example, a domain-normalizing transformation was used in order to align the grid lines with the body and shock wave in physical space. These are examples of simple algebraic mappings. To generate computational grids using this technique, known functions are used in one, two, or three dimensions to take arbitrarily shaped physical regions into a rectangular computational domain. Although the computational domain is not required to be rectangular, the usual procedure uses a rectangular region for simplicity.

The simplest procedure available that may be used to produce a boundary fitted computational mesh is the normalizing transformation discussed in Section 5.6.1. Suppose a body fitted mesh is desired in order to solve for the flow in a diverging nozzle. The geometry of the nozzle is shown in Fig. 10.2, and the describing function for the nozzle is given as

$$y_{\max} = x^2 \qquad 1.0 \leqslant x \leqslant 2.0 \tag{10.1}$$

In this example, a computational grid can easily be generated by choosing equally spaced increments in the x direction and using uniform division in the y direction. This may be described as

$$\begin{aligned} \xi &= x \\ \eta &= \frac{y}{y_{\max}} \end{aligned} \tag{10.2}$$

where $y_{\max}$ denotes the y coordinate of the nozzle wall. In this case the values of x and y for a given ξ and η are easily recovered. The mesh generated in the physical domain is shown in Fig. 10.3.

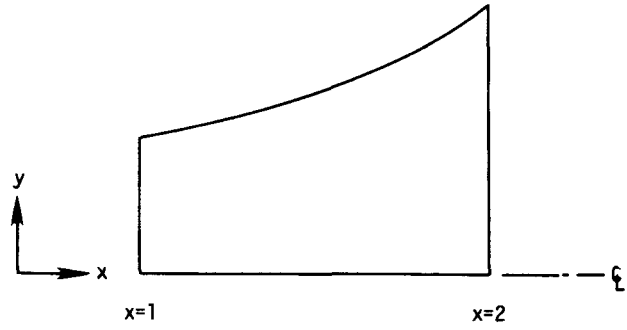

Figure 10.2 Nozzle geometry.

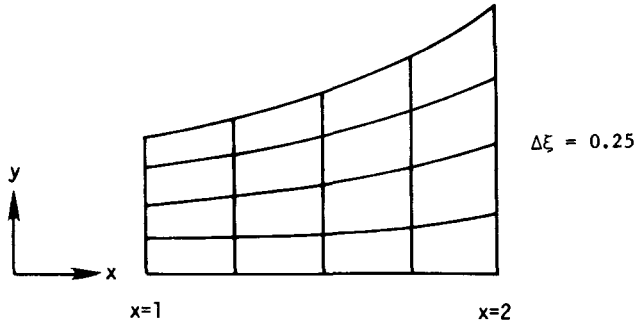

Figure 10.3 Computational mesh in physical space.

Care must be exercised when the metrics of the transformation are derived. In particular, the η_x derivative of Eq. (10.2) is written

$$\eta_x = -\frac{y}{y_{\max}^2}\frac{dy_{\max}}{dx} = -\frac{2\eta}{\xi} \tag{10.3}$$

and

$$\eta_y = \frac{1}{y_{\max}} = \frac{1}{\xi^2} \tag{10.4}$$

In the example just completed, the transformation was analytic and the point distribution was obtained through the given mapping. The same normalizing transformation could have been constructed by assigning points in the physical plane along constant ξ and constant η lines and numerically computing the metrics by using second-order central differences. This has the advantage of permitting assignment of grid points in the physical plane where desired. The disadvantage is that all metrics must be determined using numerical techniques. In this case the transformation would be numerical and not algebraic.

If numerical methods are used to generate the required transformation, the terms x_ξ, x_η, y_ξ, and y_η are determined using finite differences. The quantities ξ_x, ξ_y, η_x, and η_y appear in the differential equation, which must be solved. These quantities are obtained from the expressions

$$\begin{aligned} \xi_x &= \frac{y_\eta}{\mathrm{I}} \\ \xi_y &= -\frac{x_\eta}{\mathrm{I}} \\ \eta_x &= -\frac{y_\xi}{\mathrm{I}} \\ \eta_y &= \frac{x_\xi}{\mathrm{I}} \end{aligned} \tag{10.5}$$

where the inverse Jacobian (I) is given by

$$\mathrm{I} = (J)^{-1} = x_\xi y_\eta - y_\xi x_\eta$$

More details will be presented in the section treating mappings governed by differential equations.

Example 10.1 Compare the metrics for the simple normalizing transformation just discussed by computing them analytically and also by using a finite-difference approximation.

Solution We select the point (1.75, 2.2969) in the nozzle of Fig. 10.3 to compare the metrics. From Eq. (10.3), the analytic evaluation is

$$\eta_x = -\frac{2(0.75)}{1.75} = -0.85714$$

The numerical calculation is performed by using Eq. (10.5). The inverse Jacobian (I) is evaluated first as

$$\mathrm{I} = \cancel{x_\xi}^{1} y_\eta - \cancel{y_\xi}^{0} x_\eta = \frac{3.0625 - 1.53125}{2(0.25)} = 3.06250$$

Next, the y_ξ term is computed as

$$y_\xi = \frac{3 - 1.6875}{0.5} = 2.6250$$

Thus

$$\eta_x = -\frac{2.6250}{3.0625} = -0.85714$$

In this example, the metrics computed by analytical and numerical methods give equally good results. Of course, this is not true for many problems.

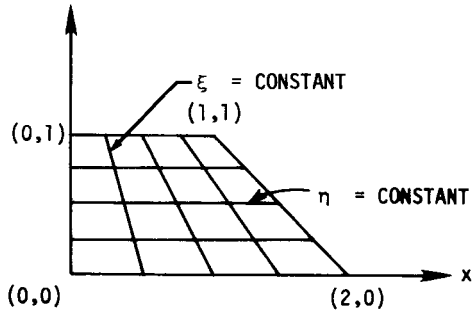

Figure 10.4 Trapezoid to rectangle mapping.

Example 10.2 The trapezoidal region shown in Fig. 10.4 is mapped into a corresponding rectangular region by the equations given by

$$x = \left(\frac{1+\xi}{2}\right)\left(\frac{3-\eta}{2}\right)$$
$$y = \frac{\eta+1}{2} \tag{10.6}$$

In this example the physical domain is mapped into a rectangular region centered at the origin. This demonstrates the use of a normalizing transformation in one direction along with a simple translation.

This choice of parameterization will give a different grid even for the case where Lagrange interpolation is used. While the preceding examples show acceptable results for computational grids, it is not always possible to construct a satisfactory grid without a more systematic approach. In particular, the application of general interpolation techniques provides a more formal approach toward the generation of grids using algebraic methods.

Smith and Weigel (1980) developed a flexible method of directly providing grids using interpolation between surfaces. In this method, the domain of interest is defined by an upper and a lower boundary in the physical plane. As shown in Fig. 10.5(a), these boundaries are denoted by

$$\mathbf{r}_1 = \mathbf{r}_{B_1}(\xi) \tag{10.7}$$

$$\mathbf{r}_2 = \mathbf{r}_{B_2}(\xi) \tag{10.8}$$

The upper and lower boundaries have been parameterized using the computational coordinate as the parameter. These curves may also be written in terms of the scalar coordinates

$$\begin{aligned} x_{B_1} &= x_1(\xi) \\ y_{B_1} &= y_1(\xi) \\ x_{B_2} &= x_2(\xi) \\ y_{B_2} &= y_2(\xi) \end{aligned} \tag{10.9}$$

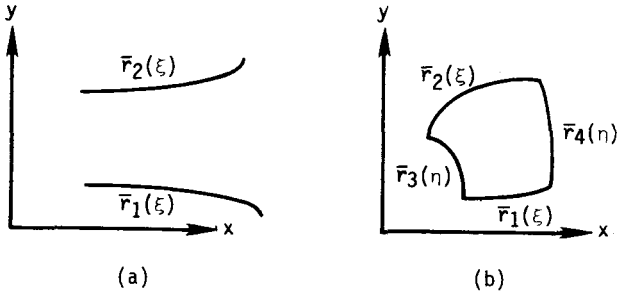

Figure 10.5 Algebraic interpolation domains. (a) Open domain for two-surface methods. (b) Closed domain for transfinite interpolation.

The range on ξ in the computational plane is

$$0 \leqslant \xi \leqslant 1$$

and the transformation is defined so that at $\eta = 0$,

$$\begin{aligned} x_{B_1} &= x_1(\xi) = x(\xi, 0) \\ y_{B_1} &= y_1(\xi) = y(\xi, 0) \end{aligned} \tag{10.10}$$

and at $\eta = 1$,

$$\begin{aligned} x_{B_2} &= x_2(\xi) = x(\xi, 1) \\ y_{B_2} &= y_2(\xi) = y(\xi, 1) \end{aligned} \tag{10.11}$$

A function defined on $0 \leqslant \eta \leqslant 1$ with parameters on the two boundaries completes the algebraic relation. This is chosen to be of the form

$$\begin{aligned} x = x(\xi, \eta) &= F\left(x_1, \frac{dx_1}{d\eta}, \ldots, x_2, \frac{dx_2}{d\eta}, \ldots\right) \\ y = y(\xi, \eta) &= G\left(y_1, \frac{dy_1}{d\eta}, \ldots, y_2, \frac{dy_2}{d\eta}, \ldots\right) \end{aligned} \tag{10.12}$$

If linear variations across the domain are selected, the grid is determined on the basis of Lagrange interpolation, and the form of the expression for (x, y) is

$$\mathbf{r}(\xi, \eta) = (1 - \eta)\mathbf{r}_1(\xi) + \eta\mathbf{r}_2(\xi) \tag{10.13}$$

or in scalar form,

$$\begin{aligned} x &= (1 - \eta)x_1(\xi) + \eta x_2(\xi) \\ y &= (1 - \eta)y_1(\xi) + \eta y_2(\xi) \end{aligned}$$

Example 10.3 To demonstrate the use of this approach, suppose we wish to map the trapezoid defined by the equations

$$\begin{aligned} x &= 0 \\ x &= 1 \\ y &= 0 \\ y &= 1 + x \end{aligned}$$

into the computational plane. In this case, the upper and lower boundaries may be written

$$x_{B_1} = x_1(\xi) = \xi$$
$$y_{B_1} = y_1(\xi) = 0$$
$$x_{B_2} = x_2(\xi) = \xi$$
$$y_{B_2} = y_2(\xi) = 1 + \xi$$

This produces the mapping required in Eq. (10.13) and is of the form

$$x = \xi$$
$$y = (1 + \xi)\eta$$

This parameterization produces the simple normalizing transformation discussed earlier in this section. In this example, both the right and left boundaries are also correctly mapped. This is coincidental and will not occur in more general problems. A different point distribution can be obtained by choosing a nonlinear function for the boundary parameterization. For example, if

$$x_1 = \xi^2$$
$$x_2 = \xi^2$$

then

$$x = \xi^2$$
$$y = \eta(1 + \xi^2)$$

In this case, the nonlinear boundary parameterization produces some clustering of the grid points. However, the ability to cluster points is limited to the influence of the boundary point distribution on the interior through interpolation.

Additional control of the grid point distribution can be attained if higher-order interpolation polynomials are used. Hermite interpolation is often employed, since the derivatives specifying the initial slope of the constant coordinate curves leaving the boundaries are also included in the interpolation. The expression for the interpolated grid coordinates is of the form

$$\mathbf{r}_{\text{inter}}(\xi, \eta) = \mathbf{r}_1(\xi)f_1(\eta) + \mathbf{r}_2(\xi)f_2(\eta) + \frac{d\mathbf{r}_1(\xi)}{d\eta}f_3(\eta) + \frac{d\mathbf{r}_2(\xi)}{d\eta}f_4(\eta) \tag{10.14}$$

where the functions are given by

$$\begin{aligned} f_1(\eta) &= 2\eta^3 - 3\eta^2 + 1 \\ f_2(\eta) &= -2\eta^3 + 3\eta^2 \\ f_3(\eta) &= \eta^3 - 2\eta^2 + \eta \\ f_4(\eta) &= \eta^3 - \eta^2 \end{aligned} \tag{10.15}$$

This additional flexibility can be used to produce orthogonality at the upper and lower boundaries [see Kowalski (1980) for details].

In most problems, the boundaries are not analytic functions but are simply prescribed as a set of data points. In this case, the boundary must be approximated by a curve fitting procedure to employ algebraic mappings. Eiseman and Smith (1980) discuss possible methods of accomplishing this and particularly recommend tension splines. Tension splines are suggested because higher-order approximations including cubic splines tend to produce wiggles in the boundary. The tension parameter in the tension spline allows control of this phenomenon.

With the simple interpolation schemes investigated thus far, only two boundaries are matched. Unfortunately, many problems include the necessity of matching four boundaries enclosing the physical domain. Gordon and Hall (1973) describe transfinite interpolation, and Rizzi and Eriksson (1981) provide application details for generating boundary conforming grids with algebraic interpolation. To understand transfinite interpolation, consider the geometry of the physical domain given in Fig. 10.5(b). The simple interpolation scheme of Eq. (10.13) does not produce a grid that matches the boundaries denoted by 3 and 4 in Fig. 10.5(b). In fact, this simple interpolation produces an error at these boundaries that may be specifically identified. The error at boundaries denoted by 3 and 4 may be written

$$\begin{aligned} \mathbf{e}_3 &= \mathbf{r}_3(\eta) - \mathbf{r}_{\text{inter}}(0, \eta) \\ \mathbf{e}_4 &= \mathbf{r}_4(\eta) - \mathbf{r}_{\text{inter}}(1, \eta) \end{aligned} \tag{10.16}$$

where the maximum and minimum values of ξ are taken to be 0 and 1, respectively, to define the left and right boundaries of the domain and the subscript, inter, indicates the interpolated values of (x, y). If Lagrange interpolation is used, these errors are given by

$$\begin{aligned} \mathbf{e}_3 &= \mathbf{r}_3(\eta) - (1 - \eta)\mathbf{r}_1(0) - \eta\mathbf{r}_2(0) \\ \mathbf{e}_4 &= \mathbf{r}_4(\eta) - (1 - \eta)\mathbf{r}_1(1) - \eta\mathbf{r}_2(1) \end{aligned} \tag{10.17}$$

To eliminate these errors, we interpolate them onto the domain, so the errors at both the left and right boundaries are eliminated. The expression for the interpolated error may be written

$$\mathbf{e}(\xi, \eta) = (1 - \xi)\mathbf{e}_3(\eta) + \xi\mathbf{e}_4(\eta) \tag{10.18}$$

The expression for the interpolated computational coordinates is written

$$\mathbf{r}(\xi, \eta) = \mathbf{r}_{\text{inter}} + \mathbf{e}(\xi, \eta) \tag{10.19}$$

and the final result becomes

$$\begin{aligned} \mathbf{r}(\xi, \eta) = (1 - \eta)\mathbf{r}_1(\xi) + \eta\mathbf{r}_2(\xi) + (1 - \xi)[\mathbf{r}_3(\eta) - (1 - \eta)\mathbf{r}_1(0) - \eta\dot{\mathbf{r}}_2(0)] \\ + \xi[\mathbf{r}_4(\eta) - (1 - \eta)\mathbf{r}_1(1) - \eta\mathbf{r}_2(1)] \end{aligned} \tag{10.20}$$

This result shows that the transfinite interpolation (TFI) is composed of interpolations between the corresponding edges and an interpolation from each of the corner points. This interpolation will match all of the edge data required by a fixed domain. If Hermite interpolation is used, a different final result is obtained that matches the domain boundaries and also the initial slope requirements. In this case, it is possible to control the orthogonality of the mesh at the boundaries, and details can be found in the work of Chawner and Anderson (1991). The TFI method can also be used when only three boundaries are matched. This can be accomplished by noting that the error need not vanish on the fourth boundary, but may create a fourth boundary that assumes any shape. This is similar to using a TFI scheme to create a grid on an open domain. When transfinite interpolation is used, it must be remembered that no control over the Jacobian of the transformation is maintained. Grid crossing can and will occur when different parameterizations of the boundaries are used or when the derivatives at the boundaries (Hermite) are given certain values. These behavior traits can be seen through the applications provided by the problems at the end of this chapter.

The basic idea behind the most common algebraic methods used to construct grids was presented in this section. With this basic understanding of the TFI scheme, the method can be developed with considerably more rigor. However, the intent here is to introduce the most common methods of grid generation, and a more sophisticated presentation will not be included.

10.3 DIFFERENTIAL EQUATION METHODS

In the previous section, algebraic methods were presented that can be used to produce usable grids. Any procedure that results in an acceptable grid is a valid one. One of the most frequently used and most highly developed procedures is the differential equation method. If a partial differential equation is used to generate a grid, we can exploit the properties of the solution of the grid-generating equation in producing the mesh. All three classes of partial differential equations have been used to produce grid construction methods, and a short discussion of each is presented in this section.

10.3.1 Elliptic Schemes

Elliptic partial differential equations (PDEs) have the property that the solutions are generally very smooth. This smoothness can be used to advantage, and for this reason, Laplace's equation is a good choice. To better understand the choice of Laplace's equation, consider the solution of a steady heat conduction problem in two dimensions with Dirichlet boundary conditions. The solution of this problem produces isotherms that are smooth (class C^{II} properties) and are nonintersecting. The number of isotherms in a given region can be increased by adding a source term. If the isotherms are used as grid lines, they will be

smooth, nonintersecting, and can be densely packed in any region by controlling the source term.

One of the attractive features of using Laplace's equation is that the Jacobian is guaranteed to be positive as a result of the maximum principle for harmonic functions. Unfortunately, this theorem only applies to the analytic equations and solution (Thompson et al., 1985). When the differential equation is discretized, the truncation errors may lead to grid crossing even though the maximum principle holds for the solution of the analytic equation. This point must be clearly understood. If the numerical formulation of the differential equation satisfies the consistency condition, then the maximum principle will be satisfied in the limit of vanishing mesh size. However, no promises can be made for finite mesh sizes. In some cases, an estimate can be made to determine when mesh crossing will occur (see Prob. 10.14).

This idea of using elliptic differential equations is based on the work of Crowley (1962) and Winslow (1966) and transforms the physical domain into the computational plane, where the mapping is controlled by a Poisson equation. Thompson et al. (1974) have worked extensively on using elliptic PDEs to generate grids. When the Poisson grid generators are used, the mapping is constructed by specifying the desired grid points (x, y) on the boundary of the physical domain with the interior point distribution determined through the solution of the equations

$$\begin{aligned} \xi_{xx} + \xi_{yy} &= P(\xi, \eta) \\ \eta_{xx} + \eta_{yy} &= Q(\xi, \eta) \end{aligned} \tag{10.21}$$

where (ξ, η) represent the coordinates in the computational domain and P and Q are terms that control the point spacing on the interior of D. Equations (10.21) are then transformed to computational space by interchanging the roles of the independent and dependent variables. This yields a system of two elliptic equations of the form

$$\begin{aligned} \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} &= -\mathrm{I}^2(Px_\xi + Qx_\eta) \\ \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} &= -\mathrm{I}^2(Py_\xi + Qy_\eta) \end{aligned} \tag{10.22}$$

where

$$\begin{aligned} \alpha &= x_\eta^2 + y_\eta^2 \\ \beta &= x_\xi x_\eta + y_\xi y_\eta \\ \gamma &= x_\xi^2 + y_\xi^2 \\ \mathrm{I} &= \frac{\partial(x, y)}{\partial(\xi, \eta)} = x_\xi y_\eta - x_\eta y_\xi \end{aligned}$$

This system of equations is solved on a uniformly spaced grid in the computational plane. This provides the (x, y) coordinates of each point in physical space. For simply connected regions, Dirichlet boundary conditions can be used

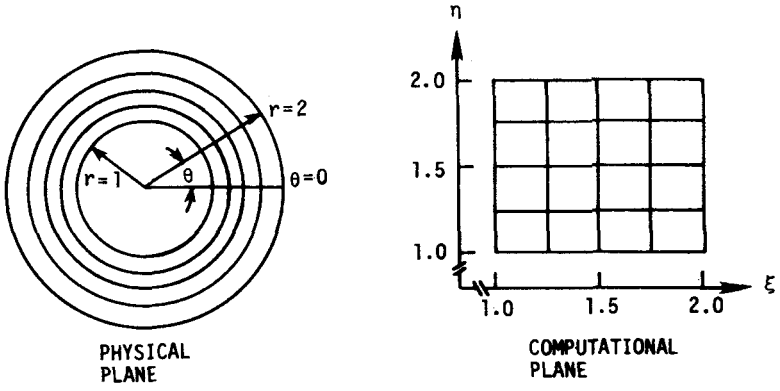

Figure 10.6 Application of Thompson scheme.

at all boundary points. The advantages of using this technique to generate a computational mesh are many. The resulting grid is smooth, the transformation is one to one, and complex boundaries are easily treated. Of course, there are some disadvantages. Specification of P and Q is not an easy task, grid point control on the interior is difficult to achieve, and boundaries may be changing with time. In the latter case, the grid must be computed after each time step. This can consume large amounts of computer time.

A simple example demonstrating the application of the Thompson scheme is shown in Fig. 10.6. The region between two concentric circles is mapped into the computational domain, and the resulting constant ξ and η surfaces in physical space are shown. The inner circle is of radius r_0, and the outer circle is of radius r_1. For this problem, the circle is cut at $\theta = 0$ and mapped into the region between 1 and $\xi_{\max}$ and 1 and $\eta_{\max}$ in computational space. In this problem the mapping is determined by a solution of two Laplace's equations,

$$\nabla^2 \xi = 0$$
$$\nabla^2 \eta = 0$$

subject to boundary conditions

$$\begin{aligned} r &= r_0 & \eta &= 1 \\ r &= r_1 & \eta &= \eta_{\max} \\ \theta &= 0 & \xi &= 1 \\ \theta &= 2\pi & \xi &= \xi_{\max} \end{aligned}$$

The solution is of the form

$$x = R \cos \phi$$
$$y = R \sin \phi$$

where

$$R = r_0\left(\frac{r_1}{r_0}\right)^{(\eta-1)/(\eta_{\max}-1)}$$

$$\phi = \left(\frac{\xi - 1}{\xi_{\max} - 1}\right)2\pi$$

This solution is interesting, in that a uniform grid in the physical domain is not achieved in this case. The distribution in the radial direction is a series of concentric circles. To obtain the mapping with a series of uniformly spaced concentric circles, $P = 0$ and $Q = 1/\eta$ (see Prob. 10.9).

As previously noted, one of the difficulties with this scheme is point control on the interior of the domain. This requires that methods for developing P and Q be devised in order to obtain the desired point distribution. Middlecoff and Thomas (1979) have developed a method that provides approximate control of point spacing by evaluating P and Q according to the desired point distribution on the boundary.

In order to demonstrate this idea, we suppose that a solution of Eq. (10.21) is required subject to Dirichlet boundary conditions. We elect to write P and Q in the form

$$\begin{aligned} P &= \phi(\xi, \eta)\left(\xi_x^2 + \xi_y^2\right) \\ Q &= \psi(\xi, \eta)\left(\eta_x^2 + \eta_y^2\right) \end{aligned} \tag{10.23}$$

where ϕ and ψ will be specified through the boundary conditions. With this convention, our original system [Eq. (10.22)] may be written

$$\begin{aligned} \alpha(x_{\xi\xi} + \phi x_\xi) - 2\beta x_{\xi\eta} + \gamma(x_{\eta\eta} + \psi x_\eta) &= 0 \\ \alpha(y_{\xi\xi} + \phi y_\xi) - 2\beta y_{\xi\eta} + \gamma(y_{\eta\eta} + \psi y_\eta) &= 0 \end{aligned} \tag{10.24}$$

Middlecoff and Thomas (1979) proposed writing these equations along either constant ξ or η surfaces corresponding to the boundaries of the domain, assuming that the grid was orthogonal at the boundaries and that the opposite family of lines had zero curvature at the intersection. If we are interested in finding the values of ϕ along a constant η boundary, it is assumed that the constant ξ curves intersecting this boundary have no curvature at the intersection point and that the two are orthogonal. If S represents arc length along the constant η boundary, then the expression relating this arc length to the grid control function ϕ is

$$S_{\xi\xi} + \phi S_\xi = 0 \tag{10.25}$$

In a similar fashion, if arc length along the constant ξ boundaries is denoted by N, the equation for the relationship between N and the grid control parameter ψ is given by

$$N_{\eta\eta} + \psi N_\eta = 0 \tag{10.26}$$

Since S and N represent arc length along the boundaries, the values of (x, y) specified on the domain boundaries permit S and N to be determined. Finite-difference forms of the above two equations may be used to find the values of ϕ and ψ that are needed to determine the interior grid from the Thomas and Middlecoff (TM) form of the Thompson scheme. The interior values for ϕ and ψ are found by interpolating the boundary values onto the interior. A simple Lagrange interpolation is usually adequate.

The interior point distribution or clustering is determined by either P and Q in the Thompson formulation or by ϕ and ψ in the TM formulation of the Poisson grid generation equations. In order to control grid point distribution on the interior of the domain, it is important to understand how the construction of these grid control functions influences grid point location. In the original TM formulation, the values of ϕ and ψ were determined from the boundary and interpolated to determine the interior distribution. Next we discuss why the values of ϕ and ψ found from the approximations at the boundaries result in control of the grid points.

Anderson (1987) examined the TM form of the Poisson grid generation equations written along the constant coordinate lines without the assumption of orthogonality and zero curvature of the intersecting family. The resulting equations show that

$$S_{\xi\xi} + S_\xi \left[\phi - (\mu_\xi - 2\nu_\xi) \cot\theta - \frac{S_\xi \nu_\xi}{N\eta \sin\theta} \right] = 0 \tag{10.27}$$

and

$$N_{\eta\eta} + N_\eta \left[\psi + (\nu_\eta - 2\mu_\eta) \cot\theta + \frac{\mu_\xi N_\eta}{S_\xi \sin\theta} \right] = 0 \tag{10.28}$$

In these expressions, the first terms inside the square brackets are the same as the TM terms that are associated with the orthogonality and local curvature of the grid. The values of ν and μ represent the local inclination of constant ξ and η lines, respectively, and θ is the angle of intersection between the two families of curves. If the grid control parameters are sufficiently large in comparison with the other terms, the grid will be determined primarily by the values of ϕ and ψ. The governing equations for the arc lengths are then consistent with the TM formulation. The expressions given by Eqs. (10.27) and (10.28) are equidistribution laws, and the values of the grid control parameters are related to weight functions for this equidistribution. Consider the equidistribution of a weight function w in the discrete form

$$(\Delta S)w = \text{const} = C \tag{10.29}$$

where ΔS is the distance between any two mesh points along a constant η curve. If ΔS is large, w is small, and vice versa. This shows that control of the mesh spacing can be attained by correctly formulating the weight function. The continuous equivalent of the discrete equidistribution law may be written

$$S_\xi w = C \tag{10.30}$$

where the arc length derivative is now controlled by the weight function. If this equation is differentiated, we obtain

$$S_{\xi\xi} + S_\xi w_\xi / w = 0 \tag{10.31}$$

This is similar to the form of the original TM equation and shows that the TM method of finding the correct values of ϕ and ψ is an approximate equidistribution law with

$$\phi = w_\xi / w \tag{10.32}$$

The grid spacing control described above shows why control can be exercised by proper construction of the weight functions, or equivalently, the values of ϕ and ψ. Geometric functions that provide clustering near points or lines have been developed and are generally written in the form of an exponential (Thompson, 1975, 1980). A function that clusters near the line $\eta = \eta_j$ is of the form

$$\psi(\xi, \eta) = -A \operatorname{sgn}(\eta - \eta_j) e^{(-B|\eta - \eta_j|)} \tag{10.33}$$

where A and B are positive constants. To cluster near a point (ξ_j, η_j), the function has a correction to the distance and is of the form

$$\psi(\xi, \eta) = -A \operatorname{sgn}(\eta - \eta_j) e^{[-B\sqrt{(\xi - \xi_j)^2 + (\eta - \eta_j)^2}]} \tag{10.34}$$

where the constants A and B are taken to be positive. A corresponding expression may be written for ϕ.

Other techniques for the control of interior grid point locations with control of the orthogonality at the boundaries have been developed. Sorenson and Steger (1983) and Hilgenstock (1988) have presented methods for the control of the orthogonality at boundaries and the spacing of the first mesh interval on the interior. These procedures use an iteration scheme to attain orthogonality at the boundaries and satisfy the specified spacing. The orthogonality constraint is typically allowed to attenuate into the interior to prevent overspecification of the problem. If orthogonality at the boundary is a critical issue for a given application, these methods are very effective.

Other variations on the use of elliptic differential equations may be found in the literature. One of the interesting variations has been presented by Winslow (1981). In the original Poisson grid generation equations, the control of the arc lengths requires that two grid control parameters (in two dimensions) be specified. A simpler approach might only require the specification of the grid cell area or volume. This would necessitate prescription of only one parameter, regardless of the number of dimensions in the problem. Winslow (1981) called this parameter the diffusion and wrote the governing equations in the form

$$\nabla \cdot (D \nabla \xi) = 0 \tag{10.35a}$$

$$\nabla \cdot (D \nabla \eta) = 0 \tag{10.35b}$$

The parameter D may be specified to control the spacing of the computational coordinates, as can be seen if these equations are integrated over an arbitrary

control volume. Anderson (1990) has shown analytically that the diffusion parameter is approximately proportional to the Jacobian of the transformation. Consequently, to specify the cell area or volume, the diffusion is set equal to the desired volume multiplied by a scaling factor. Of course, the simplicity of this approach must be traded off against the loss of ability to control anything except the cell volume.

General construction of orthogonal grids using elliptic methods is also of great interest, especially if the mesh spacing is also controlled. This may be accomplished in 2-D problems, and the works of Eiseman (1982), Arina (1986), and Sharp and Anderson (1991) are recommended reading.

Many other researchers have contributed to the state of the art in elliptic grid generation, and the interested reader is encouraged to consult the many conference publications on grid generation and the recent review paper by Thompson (1996).

10.3.2 Hyperbolic Schemes

Hyperbolic systems can also be used to generate grids. The advantage in using this type of partial differential equation is that the grid may be generated by solving the governing equations only once. This type of grid generation scheme is usually applied to problems with open domains consistent with the type of PDE describing the physical problem. The initial point distribution is specified along an initial data line with appropriate boundary conditions, and the solution is marched outward. The outer boundary at the end of the computation must be accepted wherever it occurs, with the shape that has resulted from the calculation. Steger and Sorenson (1980) described a method using a system of hyperbolic equations to generate a mesh. They have proposed an arc length orthogonality scheme and a volume orthogonality method. Only the latter will be presented in detail here.

In a 2-D problem, the Jacobian of the transformation controls the magnification of area elements between the physical and computational planes. If we imagine that mesh spacing in computational space is given by $\Delta\xi = \Delta\eta = 1$, then the area elements are also one unit in size. The inverse of the Jacobian,

$$x_\xi y_\eta - y_\xi x_\eta = \mathrm{I} \tag{10.36}$$

then represents the area in physical space for a given area element in computational space. If I is specified as a function of position, then Eq. (10.36) can be used as a single equation specifying grid control in the physical plane. A second equation is obtained by requiring that the grid lines be orthogonal at the boundary in physical space. Along a boundary where $\xi(x, y) = \text{const}$, we may write

$$d\xi = 0 = \xi_x\, dx + \xi_y\, dy$$

or

$$\left.\frac{dy}{dx}\right)_{\xi=\text{const}} = -\frac{\xi_x}{\xi_y} = \frac{y_\eta}{x_\eta} \tag{10.37}$$

Along an η = const surface

$$\left.\frac{dy}{dx}\right)_{\eta=\text{const}} = -\frac{\eta_x}{\eta_y} = \frac{y_\xi}{x_\xi} \tag{10.38}$$

If we require that ξ and η surfaces be perpendicular, the slopes must be negative reciprocals. This requirement becomes

$$x_\xi x_\eta + y_\xi y_\eta = 0 \tag{10.39}$$

The system given by Eqs. (10.36) and (10.39) is linearized by expanding about a known state denoted by the tilde. Using this convention, we may linearize one of the terms in Eq. (10.39) as

$$\begin{aligned} x_\xi y_\eta &= (\tilde{x} + x - \tilde{x})_\xi (\tilde{y} + y - \tilde{y})_\eta \\ &= \tilde{x}_\xi \tilde{y}_\eta + \tilde{y}_\eta (x_\xi - \tilde{x}_\xi) + \tilde{x}_\xi (y_\eta - \tilde{y}_\eta) + O(\Delta^2) \\ &= \tilde{y}_\eta x_\xi + \tilde{x}_\xi y_\eta - \tilde{x}_\xi \tilde{y}_\eta + O(\Delta^2) \end{aligned} \tag{10.40}$$

If the other terms are linearized in a similar manner, we obtain

$$[A]\mathbf{w}_\xi + [B]\mathbf{w}_\eta = \mathbf{f} \tag{10.41}$$

where

$$\mathbf{w} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$[A] = \begin{bmatrix} \tilde{x}_\eta & \tilde{y}_\eta \\ \tilde{y}_\eta & -\tilde{x}_\eta \end{bmatrix} \qquad [B] = \begin{bmatrix} \tilde{x}_\xi & \tilde{y}_\xi \\ -\tilde{y}_\xi & \tilde{x}_\xi \end{bmatrix} \qquad \mathbf{f} = \begin{bmatrix} 0 \\ \mathrm{I} + \tilde{\mathrm{I}} \end{bmatrix} \tag{10.42}$$

The eigenvalues of $[B]^{-1}[A]$ must be real if the system is hyperbolic in the η direction. These eigenvalues are

$$\lambda_{1,2} = \pm\sqrt{\frac{\tilde{x}_\eta^2 + \tilde{y}_\eta^2}{\tilde{x}_\xi^2 + \tilde{y}_\xi^2}} \tag{10.43}$$

This shows that Eq. (10.41) is hyperbolic in the η direction and can be marched in η so long as $\tilde{x}_\xi^2 + \tilde{y}_\xi^2 \neq 0$.

The procedure to use in generating a grid with this scheme is to assume the body is the $\eta = 0$ surface and specify the distribution of points along the body. Next, the inverse Jacobian I in Eq. (10.36) is computed. Steger and Sorenson suggest that I be determined by laying out a straight line with length equal to that of the body surface (l) and distribute the body points on this line. Next, a line parallel to the first is drawn at an η = const surface as desired. Once this is done, the quantity I is easily determined by estimating the area elements of the

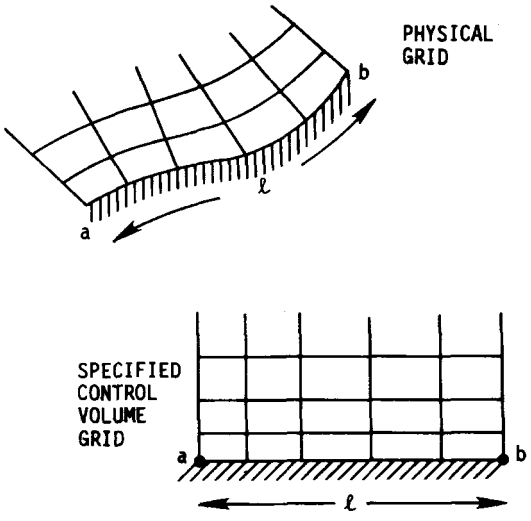

Figure 10.7 Area element computation.

grid. This procedure is illustrated in Fig. 10.7. The system of governing equations given by Eq. (10.41) is now solved using any standard method for solving systems of hyperbolic PDEs.

Since we specify I in this scheme, a smoothly varying grid is obtained if I is well chosen. However, poor selection of the I variation leads to possible "shocks" or discontinuous propagation of this information through the mesh. It is also true that discontinuous boundary data are propagated in the mesh. On the other hand, the mesh is orthogonal and is generated very rapidly. Figure 10.8 shows the grid generated about a typical airfoil shape. In this case, points have been clustered near the body in order to permit resolution of the viscous boundary layer.

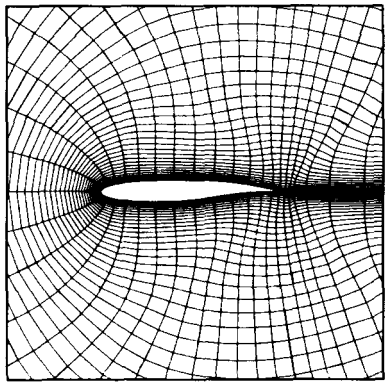

Figure 10.8 Grid for an airfoil configuration.

10.3.3 Parabolic Schemes

Parabolic PDEs are also solved by advancing the solution away from an initial data surface while satisfying boundary conditions at the ends of the domain. As was true for hyperbolic grid generators, parabolic generators should be useful in producing grids using a single-pass strategy. Nakamura (1982) and Edwards (1985) developed the basic ideas used in parabolic grid generation, and these techniques provide another way of producing acceptable grids.

The idea of parabolic grid generation is based on using the Laplace or Poisson grid generator and specially treating the parts of the equation that control the elliptic behavior. In order to understand the basic idea, consider Laplace's equation as the fundamental generating equation. The geometry of the domain is assumed to be consistent with that shown in Fig. 10.7 for the hyperbolic case. The initial data are given as the coordinates of all points along the $\eta = 0$ surface. The idea is to advance the solution for the grid outward from this surface subject to the boundary conditions along the minimum and maximum ξ edges.

If either Laplace's or Poisson's equation is used, the problem is elliptic, and the solution cannot be advanced in the η direction because the central differencing requires that information from the advanced $(j + 1)$ level be used. To illustrate this problem, consider the differencing of the second derivative,

$$\frac{\partial^2 \mathbf{r}}{\partial \eta^2}$$

If a second-order central difference is used, this is represented as

$$\frac{\mathbf{r}_{j+1} - 2\mathbf{r}_j + \mathbf{r}_{j-1}}{(\Delta\eta)^2}$$

If the integration of the equation is started with given data at the location indicated by $j - 1$, the unknown level is then indicated by j. However, the difference equations show that information from the next level at $j + 1$ is needed. We supply this information by assuming that this can be approximated by replacing any value at $j + 1$ by the outer boundary value, as originally suggested by Nakamura. It is also necessary to use this idea in evaluating the cross-derivative terms and first-derivative terms. When this approach is selected, the grid generation equation in discrete form may be solved as a marching problem, with the unknowns at the jth level constituting the values to be determined. At each step, the $j + 1$ level information is supplied by simply continuing to use the values on the outer boundary of the domain. This method creates a technique that allows a solution to the elliptic equation to be computed via a marching scheme. It has the conceptual advantage of producing a grid in a single pass.

The original parabolic methods of Nakamura (1982) and Edwards (1985) used the outer boundary to evaluate the necessary $j + 1$ point data in solving Laplace's equation. Other methods may be used to approximate the information

needed at the advanced levels. One way is to use a reference grid (Noack, 1986), where an initial grid is constructed using any simple method, usually an algebraic scheme, and the reference grid point locations are used to supply the needed advanced point information for the solution of Laplace's equation. If, after the initial solution is computed, the reference grid is taken to be the first iterative pass from a Laplace or Poisson equation solution, an additional pass through the grid solver is made, and this process is repeated until a satisfactory grid is produced. This effectively becomes a solution to the elliptic equation that has not been completely converged. Hodge et al. (1987) also extended the idea of parabolic grid generation by using the Poisson equation in place of Laplace's equation and also provided some latitude in the selection of the direction that the equations could be parabolized.

At this point, no information has been given to suggest a means to control the grid spacing. In the works of Nakamura (1982) and Edwards (1985), grid control was accomplished by using nonuniform spacing in the computational domain. This variation in the cell sizes in the computational domain was used in solving a Laplace grid generation equation, providing control of mesh spacing in the physical domain. Some control of orthogonality was also provided by altering the location of the outer boundary points. This effectively is accomplished by altering the source terms that appear in the difference equations. The reference grid was used by Noack (1985) as a means of controlling the space of the grid points. Hodge et al. (1987) has given some guidance in the selection of the source terms in the Poisson equation for parabolic grid generation.

Parabolic grid generation has the advantage that no grid shocks occur as is possible in the hyperbolic case. In this sense, we expect grids to be relatively smooth. However, the effort required to set up the reference grid, or the outer boundary, as well as select a variable step size to control the grid point locations is time consuming. As with any method, there are advantages and disadvantages. However, if sufficient familiarity with these techniques is gained through experience, parabolic grid generation can be very effective.

10.4 VARIATIONAL METHODS

Variational methods have recently gained in popularity as a grid generation tool. When a function is minimized, several measures of mesh quality can be included. Brackbill and Saltzman (1980) and Brackbill (1982) have developed a technique for constructing an adaptive grid using a variational approach. In their scheme, a function that contains a measure of grid smoothness, orthogonality, and volume variation is minimized using variational principles. The smoothness of the transformation is represented by the integral

$$I_s = \int_D \left[(\nabla \xi)^2 + (\nabla \eta)^2 \right] dV \tag{10.44}$$

A measure of orthogonality is provided by

$$I_0 = \int_D (\nabla\xi \cdot \nabla\eta)^2 \mathrm{I}^3 \, dV \tag{10.45}$$

and the volume measure is given as

$$I_v = \int_D w\mathrm{I} \, dV \tag{10.46}$$

where w is a given weighting function.

The transformation relating the physical and computational domains is determined by minimizing a linear combination of the above three integrals. This linear combination with coefficient multipliers λ_v and λ_0 is written

$$I_t = I_s + \lambda_v I_v + \lambda_0 I_0 \tag{10.47}$$

In order to minimize I_t, the Euler-Lagrange equations must be formed (Weinstock, 1952). As an example, the smoothness measure, Eq. (10.44), may be written

$$I_s = \iint \left(\frac{x_\xi^2 + x_\eta^2 + y_\xi^2 + y_\eta^2}{\mathrm{I}} \right) d\xi \, d\eta \tag{10.48}$$

when the variables are interchanged and the integration is performed in computational space. If we construct the Euler-Lagrange equations corresponding to I_s, they are of the form

$$\begin{aligned} \left(\frac{\partial}{\partial x} - \frac{\partial}{\partial \xi}\frac{\partial}{\partial x_\xi} - \frac{\partial}{\partial \eta}\frac{\partial}{\partial x_\eta} \right) \left(\frac{x_\xi^2 + x_\eta^2 + y_\xi^2 + y_\eta^2}{\mathrm{I}} \right) &= 0 \\ \left(\frac{\partial}{\partial y} - \frac{\partial}{\partial \xi}\frac{\partial}{\partial y_\xi} - \frac{\partial}{\partial \eta}\frac{\partial}{\partial y_\eta} \right) \left(\frac{x_\xi^2 + x_\eta^2 + y_\xi^2 + y_\eta^2}{\mathrm{I}} \right) &= 0 \end{aligned} \tag{10.49}$$

If the differentiation is performed, these expressions may be written

$$\begin{aligned} A(\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta}) - B(\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta}) &= 0 \\ -B(\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta}) + C(\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta}) &= 0 \end{aligned} \tag{10.50}$$

The coefficients A, B, C, α, β, and γ are functions of the metrics, and their evaluation is left as an exercise (see Prob. 10.13). If

$$B^2 - AC \neq 0$$

these equations may be written as

$$\begin{aligned} \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} &= 0 \\ \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} &= 0 \end{aligned} \tag{10.51}$$

This is the form of the original mapping given by Winslow and is also the basic system of equations for Thompson's work. If I_t as defined in Eq. (10.47) is minimized, each of the integrals, I_v and I_0, contribute terms to a significantly

more complicated set of Euler-Lagrange equations than those given in Eq. (10.51).

The use of a variational approach provides a solid mathematical basis for the grid but also entails additional effort in solving more PDEs. The Euler-Lagrange equations must be solved in addition to those governing the fluid motion. In the example shown here, the adaptive grid is constructed by implementing a new mesh after each iteration or time step and computing the grid speed by using a backward difference. The variational approach clearly offers a powerful method for constructing computational grids. The disadvantage is that a considerable effort must be expended in solving the equations that govern the grid generation. If a linear combination of the integrals of Eq. (10.47) is used, the λ's must also be selected. However, some remarkable results have been obtained with the proper choice of these coefficients.

The book by Knupp and Steinberg (1993) is a good source for a comprehensive treatment of the application of variational methods to the grid generation problem. Examples of the application of direct methods may be found in the literature, and typical of this is the work of Kennon and Dulikravich (1985) and Carcaillet (1986). Future applications of the variational approach will likely involve more work on direct minimization of integrals as opposed to the construction of the Euler-Lagrange equations. This simplifies the work by eliminating the laborious construction of the governing differential equations by using additional CPU time. Integrals representing a measure of desired qualities in a grid can be minimized with a number of well-proven methods that are readily available in the literature. The Euler-Lagrange equations can in practice be obtained with symbolic manipulators that also remove much of the difficulty in application if this classical approach is used. Variational techniques are a powerful way to formulate measures of grid quality and provide guidance in the construction of grid generation schemes. With continuing improvements in CPU power and inexpensive storage, more extensive use will be made of these methods.

10.5 UNSTRUCTURED GRID SCHEMES

Unstructured grid generation schemes have gained in popularity in recent years for a number of reasons. The increase in computer power and the reduction in memory costs have been major factors. One of the attractive features of unstructured mesh generation schemes is the promise they seemingly hold of ultimately providing a method that automates the grid generation process. In constructing grids using a structured approach, the grid must be segmented into blocks due to the topology of the domain and the configuration of interest, with the logical structure defined to provide appropriate connectivity. The flow solver must also be written to interpret and use the data format produced by the grid generator. This process of generating a structured mesh is a time-intensive task

for engineers and scientists working in the field. Although good progress has been made in attempting to automate the blocking and subdivision for structured grids (Dannenhoffer, 1991, 1995, 1996), interactive grid generation is still used for the majority of structured mesh problems. When an unstructured approach is employed, defining the configuration of interest forms the most complex portion of the problem for the user, and the unstructured grid generator is employed to create the grid automatically. This is the case at least in concept, although in reality, the ability to generate grids automatically, in general, is still beyond the state of the art. For unstructured grids, the connectivity information stored is cell-to-cell as opposed to block-to-block, so additional storage is necessary when compared to the structured approach. However, the increase in available CPU power and memory makes the trade-off between CPU time and engineering hours favor the unstructured approach. There are other factors that may play an equally important role. One consideration is the solver efficiency. Due to the problem of random cell location and connectivity, unstructured solvers are usually not as computationally efficient as their structured counterparts. One must also try to construct cells where the volumes are as nearly equal or change very smoothly to avoid the problem of introducing errors in the solutions that are grid induced. This problem of the smoothly changing volume size is common to both techniques. Unstructured mesh schemes must also be monitored to reduce the thin or so-called high aspect ratio cells that are created in the generation process, since these cells contribute to increased errors.

Other considerations are of importance in the construction of grids for solution of flow problems. The grid point or cell densities that give adequate resolution for flow problems create difficulties for both structured and unstructured grids. For example, in the boundary layer, the use of structured mesh schemes naturally suggests a cell shape that is elongated in the flow direction. This configuration is consistent with the boundary layer assumptions, in that more cells appear in the normal direction as compared to the flow direction, where only small changes in the flow may occur. On the other hand, the use of unstructured grids, for example, triangles in a 2-D problem and tetrahedra in 3-D, requires a higher cell density in the boundary layer because the cells need to be as nearly equilateral (analogous to orthogonality in structured meshes) as possible in order to avoid grid-induced errors in the solution. The storage requirements are much larger for the unstructured grid. This can be visualized by imagining that a 2-D structured mesh is used as a base and the mesh is then triangulated by simply inserting the diagonal in each cell. In this example, the number of cells produced is larger by a factor of 2 for the unstructured result. In 3-D problems, the number of cells produced using this procedure is at least a factor of 5 larger. In addition, the cells produced may be long and narrow (high aspect ratio), and mesh refinement is then needed to reduce this aspect ratio.

In this section, the procedure for construction of a Delaunay (1931) mesh

will be outlined using the Bowyer (1981) insertion scheme. This is intended to provide an introduction to some of the concepts associated with the logic for constructing unstructured grids.

10.5.1 Connectivity Information

As a starting point, consider the connected triangles shown in Fig. 10.9. We must determine what information is necessary to completely identify the cell and all of the neighbors of that cell in the computational mesh. In generating an unstructured mesh, the point locations are arbitrary, and we may choose to place them at any desired position. As in the structured case, each point must be identified. We consider a point insertion scheme where each point is independently inserted and the cell connectivity resulting from this insertion is determined. This suggests that points be identified sequentially as they are inserted. If 35 points have been inserted into the mesh, the next point that needs to be inserted is identified as number 36. In addition to the identification of the grid point number, the coordinates of this point must be known and stored as $[x(36), y(36)]$.

After a grid point is inserted into an existing mesh, logic for establishing the new connectivity is employed. Data that identify the grid points that form a given cell are needed. As each cell is formed, the cell is numbered, and the forming points for that cell are also stored. For example, the convention can be taken that the forming points for this 2-D example are numbered in a counterclockwise direction around each triangle. We number these as forming point one, fp1(ncell), and continue around the triangular cell including all three points. This is illustrated in Fig. 10.9. In this figure, the three triangles that constitute the cell structure are formed by using five points, which are numbered on the exterior of each triangle vertex. The number assigned to each cell is shown in parentheses on the interior, and the forming point convention shows the local identification of the forming points as 1, 2, 3 on the interior of each cell near the vertices where the forming points are located.

In addition, the neighbor cell information is needed. Cells are considered to be neighbors if they share a common face. As a convenient convention, we may

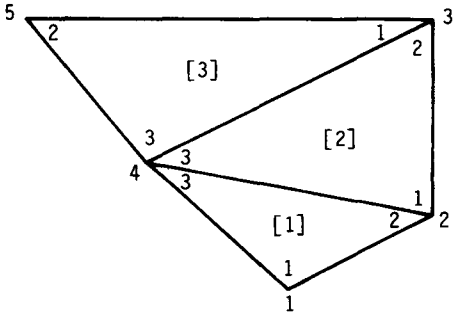

Figure 10.9 Notation for unstructured cells.

identify the first neighbor of a cell as that cell opposite to forming point 1. For example, if cell 2 is given as shown in Fig. 10.9, the second neighbor is identified as cell 1, and the first neighbor is cell 3. The neighbor information for this figure is given as

$$\begin{aligned} &\text{nei1}(1) = 2 \\ &\text{nei1}(2) = 3, \qquad \text{nei2}(2) = 1 \\ &\text{nei2}(3) = 2 \end{aligned}$$

In general, each triangular cell will have three neighbors. In this example, the first and third cells have only one and the second cell has two neighboring cells. The forming point information for these cells would be stored as

$$\begin{aligned} &\text{fp1}(1) = 1 \quad \text{fp2}(1) = 2 \quad \text{fp3}(1) = 4 \\ &\text{fp1}(2) = 2 \quad \text{fp2}(2) = 3 \quad \text{fp3}(2) = 4 \\ &\text{fp1}(3) = 3 \quad \text{fp2}(3) = 5 \quad \text{fp3}(3) = 4 \end{aligned}$$

The data contained in the cell-numbering scheme, the neighbors, and the forming points are sufficient to establish any of the parameters that are needed in the mesh or in a computational fluid dynamics (CFD) code using the cell structure derived from this mesh.

The discussion in this section regarding the information storage for a triangular mesh is also applicable to a 3-D tetrahedral cell structure except that the convenience of some of the numbering conventions identifying the neighbors may not apply. The same data are required when rectangular cells are mixed with triangles in a 2-D case. Of course, when mixed-cell hybrid grids are used, the flow solver must be written to accept any cell structure and any arbitrary connectivity.

10.5.2 Delaunay Triangulation

When an unstructured grid is constructed, the task is simplified if a fixed set of rules is followed, leading to a grid that has certain attractive properties. The Delaunay triangulation provides a grid where a fixed set of rules applies to the construction, and the grid properties include the following:

1. Given a set of points, the triangulation is unique.
2. The triangulation produces the most equilateral mesh for the given point set.
3. The grid point generation and the triangulation are decoupled.

The origins of this approach go back to the work of Dirichlet (1850), where a technique for decomposing a given domain into a set of convex polygons was studied. The geometric dual of this construction is called the Delaunay triangulation. The Delaunay triangulation has a number of implementations and includes the diagonal swapping (Cendes et al., 1985), the Bowyer insertion scheme (Bowyer, 1981), and the sweepline method (Fortune, 1987). While this approach has the advantages enumerated above, there are disadvantages as well.

These include the following:

1. Lack of uniqueness when four points lie on a circle and the counterpart in 3-D
2. The complex logic required to preserve boundaries
3. The lack of uniqueness resulting from the numerical implementation of the analytical theory of the triangulation
4. The solution errors associated with high aspect ratio or elongated cells (slivers)

These issues will become clear as the details of the triangulation emerge.

Given a point set $P = p_i(x_i)$ that is not colinear and does not have four points that lie on a circle, the set of points that is closer to vertex v_i than any other vertex is called the Voronoi polygon (Voronoi, 1908). This is illustrated in Fig. 10.10, where the Voronoi polygons are shown for a finite set of points. The dashed lines are the Voronoi polygons formed by constructing cells with sides corresponding to the perpendicular bisectors of the line segments in the triangulation. The vertices of the polygons are formed from the intersection of the perpendicular bisectors of the lines connecting the points, $P = p_i(x_i)$. As the mesh grows, more cells are added due to the addition of more line segments connecting the points in the triangulation. As the tesselation continues, the boundary polygons are those on the convex hull of the domain. The complete set of polygons including those closed on the interior and those open on the boundary of the domain is referred to as the Voronoi tesselation of the domain.

When the nuclei (point p_i contained in the polygon) of the Voronoi polygons are connected to the two nearest neighbors, the resulting structure is called the Delaunay triangulation or Delaunay tesselation. This is also shown in Fig. 10.10. In CFD, the cell structure used for a finite-volume solution of a flow problem may be applied to either the Voronoi polygons or the Delaunay tesselation. For the 2-D discussion presented here, the triangular cells always have three cell faces, while the Voronoi cells, sometimes called the mesh dual, may have a random number of edges. This suggests that a flow solver that uses the Delaunay triangulation for control volumes may have simpler logic and be easier to construct.

Although the discussion has centered on 2-D space, the ideas are also applicable to 3-D. In that case, the edges of the cell are planes, and the cells are tetrahedra or polyhedra. The increase in complexity of the grid generation problem in going from 2-D to 3-D is dramatic. Consequently, only 2-D cases will be considered here.

The circumcircle test is the simplest method to construct the Delaunay mesh and determine the connectivity of a set of points. For the planar case, three points determine a circle. For a triangular cell, the cell is a valid cell if no other point falls within the circle defined by the forming points of the circle. This is the standard test used in the Bowyer algorithm to complete the connections for the Delaunay tesselation. Figure 10.11 shows four points that

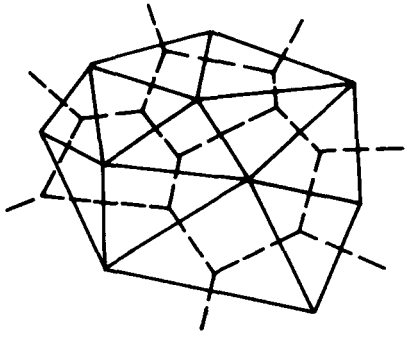

Figure 10.10 Voronoi (dashed lines) and Delaunay (solid lines) tesselations.

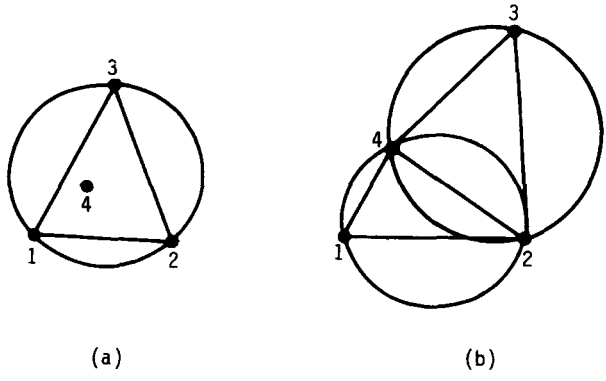

Figure 10.11 Circumcircle test. (a) Incorrect connectivity. (b) Correct connectivity.

are to be connected but the proper connections must be established. The first connection shows that the circle formed by connecting points 1, 2, and 3 encloses point 4. This violates the circle criterion, and other connections must be made. The proper connections and the associated circumcircles are also shown where points 1, 2, and 4 form one cell, and points 2, 3, and 4 form the second cell.

10.5.3 Bowyer Algorithm

Bowyer (1981) developed a scheme that can be used to triangulate a set of points. This approach is usually termed the "Bowyer insertion algorithm" because the scheme is based on inserting points into a valid Delaunay mesh and retriangulating the mesh. The basic technique relies on the circle test and a series of data tree searches to determine the new connectivity. The search can be efficiently carried out and the method can be used to refine the grid by simply inserting additional points, and finding the new connectivity as each point is inserted. The Bowyer algorithm consists of a number of steps as described below.

Step 1. Generate a set of grid point locations that are desired for the domain of interest. This set should include the boundary points and all of the interior points. The points can be generated a number of ways including the following:
   a. A random number generator
   b. Structured grid generator such as TFI or elliptic generation
   c. A self-adjusting method that determines the largest cell in the mesh, the highest aspect ratio, or some other characteristic of the generated grid and inserts a point at the circumcenter of the circle or some other predetermined location to refine the mesh to the desired level.
   d. Methods based on domain decomposition (discussed in more detail below)

Step 2. Create an initial supertriangle that completely encloses the entire domain. This may be any valid triangulation and the simplest geometry is a supertriangle or a rectangle that is triangulated.

Step 3. Insert a mesh point from the list established in Step 1 in the existing triangulation, and delete the first triangle that fails the circumcircle test. This will be the cell where the point is inserted.

Step 4. Initiate a search of the neighbors of the first deleted cell to determine if any other neighbor cells have violated circumcircles. If a neighbor cell is deleted, the common face between that neighbor and the first deleted cell must be removed, and the search proceeds through the neighbors of this cell. The tree search continues until the complete list of deleted cells is compiled.

Step 5. Establish the new connectivity by connecting the newly inserted point with the boundary points of the cavity created by the deleted cells. Add each of the new cells to the list of valid triangles.

Step 6. Repeat this procedure, starting with Step 3, until all the grid points generated in Step 1 have been inserted.

The Bowyer insertion technique described above provides a correctly triangulated mesh for convex domains. Unfortunately, most of the domains that surround practical shapes are not convex. However, the unstructured grid can be constructed by beginning with the superstructure and filling the entire superstructure as well as the body interior and including the boundary of the physical domain. This valid triangulation must undergo a postprocessing phase to remove all triangles interior to the body and those triangles between the outer boundary of the superstructure and the outer boundary of the domain.

One of the problems that must be addressed is that of preserving the integrity of the body surface when a given set of points is triangulated. When the set of points for a domain is compiled, the outer boundary of the domain and the boundary of the body are usually the first points selected to insert into the initial superstructure. The interior points between these two boundaries may be determined by any of the methods noted in step 1. However, the integrity of the body surface must be preserved, and without some special checks, this cannot be

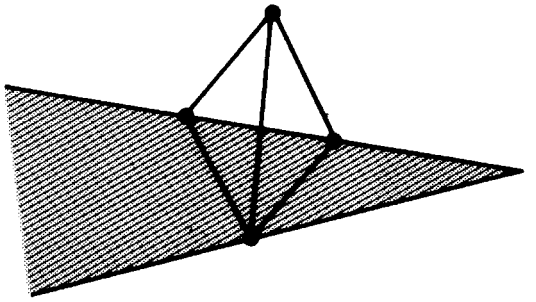

Figure 10.12 Surface fidelity violation near an airfoil trailing edge.

guaranteed. Figure 10.12 shows a connection across a body surface near an airfoil trailing edge where an improper connection has been made. This must be corrected as part of the postprocessing phase of the grid generation. Two popular ways of dealing with this problem are diagonal swapping routines or point insertion schemes that delete and reform the triangles so that the surface segment forms a cell edge.

An example of a grid constructed for an NACA 0012 airfoil is given in Fig. 6.14. This grid shows the mesh density increasing near the body in order to provide the resolution desired for accurate pressure calculations using an Euler code. The process of constructing this grid follows the steps given above, and a good reference source for additional details on constructing unstructured grids using the Bowyer scheme is that of Holmes and Snyder (1988).

Baker (1987) studied the Bowyer insertion scheme and has shown that the method is based upon two theorems.

Theorem 1 Given a Delaunay triangulation T of a planar set of points S, introduce a new point $p \in S$ and remove all triangles that fail the Delaunay circle test. All the edges of the Delaunay cavity are visible from point p.

Theorem 2 The retriangulation of a Delaunay cavity, by joining the point p to each of the boundary points of the cavity is Delaunay.

One additional issue addressed by Baker deals with the problem of precision in applying the circle test. Since the circle test is performed with a computer with finite accuracy, the precision of the test will determine whether or not the circle test is satisfied. As a consequence, if care is not exercised, the test may actually hinge on the round-off error of the machine. Baker has stated the following theorem, regarding the precision of this test.

Theorem 3 Let p_i be a finite point set, and let $d(p_i, p_j)$ represent the Euclidian metric. If L represents $\max[d(p_i, p_j)]$ and ϵ represents $\min[d(p_i, p_j)]$,

the precision of the floating point accuracy used in the circle test with Delaunay triangulation must be greater than ϵ^2/L^2.

This places a substantial restriction on the precision of the test procedure. When standard engineering workstations are used, it is imperative that double-precision arithmetic be employed.

In addition to the point insertion scheme provided by the Bowyer approach, a Delaunay mesh may be constructed by using the sweepline algorithm first suggested by Fortune (1987). This is an advancing front method that builds Delaunay cells as the front proceeds over the domain including the configuration that is the object of the study. For details on the application of this scheme, the work of Fang et al. (1993) for the 2-D case and Fang (1995) for the 3-D case is recommended.

10.6 OTHER APPROACHES

In Section 10.5.3, the Bowyer insertion scheme was outlined as a technique for constructing a Delaunay mesh. Other important methods of constructing unstructured grids have been developed using advancing fronts. While these schemes forego the Delaunay criterion, they have been used with good success in a variety of applications (Löhner and Baum, 1990; Löhner and Parikh, 1988). With this approach, the grid is advanced by adding cells at the front as it advances into the domain. These fronts are usually started from known structures such as a body or other boundary and may be composed of either structured or unstructured cells. When advancing fronts collide, rules are needed for treating the collisions and constructing cells under such circumstances. Unfortunately, these rules are constructed to treat individual exceptions, and general theorems providing construction rules are difficult to identify. However, advancing front

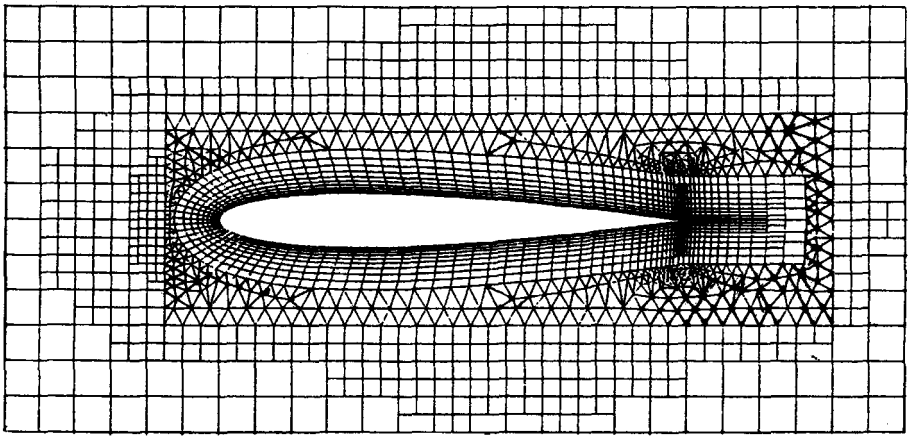

Figure 10.13 Hybrid grid for NACA 0012 airfoil.

schemes have been used to grid very complex configurations, and these grids have been used with success in solving very difficult fluid mechanics problems.

The concept of hybrid grids is also of great interest in applications in CFD. The works of Kallinderis and Ward (1993), Kallinderis (1996), and Noack et al. (1996) are representative of the state of the art. These methods are mixed, in the sense that combinations of structured and unstructured grids are used to completely cover the domain. Regions around bodies are usually grided with a structured body-conforming scheme, and the zones away from walls are covered with unstructured sections. The interface between these different zones requires logic to provide the connectivity to close the problem. This approach shows great promise as a technique to simplify the automatic grid generation problem. As an example, Fig. 10.13 shows a hybrid grid around an NACA 0012 airfoil.

The use of rectangular grids has also been of interest for some time in the CFD field. These schemes are based on using quadtree or octree data structures (Yerry and Shephard, 1983, 1984). Rectangular grid schemes have the promise of completely automating the grid generation process. The idea involves recursive subdivision of a domain until the body surface is identified at the highest refinement level in the mesh. After the refinement level is satisfied, the body-surface cells are then specially treated by considering the way the body slices these cells (Karman, 1995a, 1995b; Coirer and Powell, 1995). This is a very natural scheme to consider and forms an automated way to grid a domain once the logic for the sliced cells is complete. However, the problem of storage and data management must be carefully considered. The use of domain subdivision methods requires large storage, and usually, long computation times are necessary. The problem becomes clear when considering the resolution required to solve for flow in the turbulent boundary layer of a typical vehicle. In many cases, the refinement in the boundary layer may be extreme to achieve the desire level of refinement. A sketch of the idea used with rectangular cell

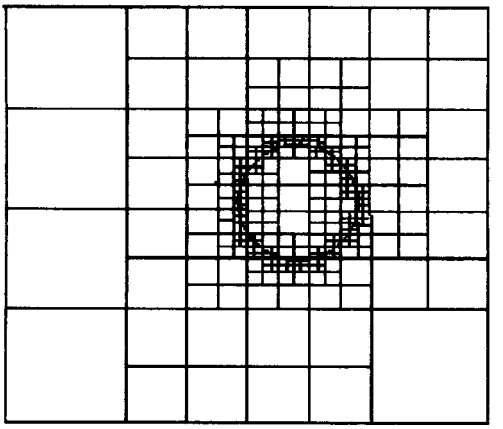

Figure 10.14 Domain subdivision using rectangular cells.

subdivision is shown in Fig. 10.14. The body has a nearly circular geometry, and note that the subdivision is completed so that only one level of refinement is accomplished at any cell boundary. This is desirable from both a logic and a solver accuracy point of view. The effectiveness of this approach is apparent when one considers the simplicity of the concept applied to very complex objects. Again, the major difficulty is in defining logic to produce correct body-surface cells. This idea of subdivision is not restricted to rectangular cells but can also be applied using other geometric structures as a base.

As is true in most of the hybrid and unstructured schemes, the issue is one of deciding what trade-off of labor hours versus CPU time is a good one. If an automated approach using any of these successful schemes can be utilized to completely solve for the flow around a vehicle in a matter of days, this becomes an attractive option. This is especially true when the calculation can be completed on an engineering workstation. Of course, the flow solver must be written to take advantage of the special features of these grid generation schemes. Future research will define the optimum use of these techniques.

10.7 ADAPTIVE GRIDS

Techniques for generating grids as a prelude to numerically solving a PDE were presented in the previous section. One problem in solving a PDE with this approach is that the grid is constructed and points are distributed in the physical domain before details of the solution are known. As a consequence, the grid may not be the best one for the particular problem.

Adaptive methods for solving PDEs have been developed to aide in increasing the accuracy of computed solutions. These methods have been classified into three categories. These categories naturally appear if one views the adaptivity as a means of reducing some measure of the global error in the solution.

In finite-element theory (Oden, 1988), the adaptive method is referred to as an h method if mesh refinement is used, an r method if the number of grid points is fixed but is redistributed, and a p method if the order of the solution scheme is increased. In finite-difference and finite-volume applications, the h and r ideas are the most popular due to the way these methods are constructed.

The adaptive grid strategies that are followed when a fixed number of points are redistributed to improve the solution are usually based on an application of the equidistribution scheme outlined previously in this chapter. Early applications of this idea include the works of White (1982), Dwyer et al. (1979, 1980), and Gnoffo (1980). These authors applied the equidistribution idea in one dimension in solving a variety of problems in fluid mechanics and heat transfer. The application of the equidistribution idea to multidimensional problems has been accomplished in several ways. The simplest to understand are the Poisson grid generators with control functions based on equidistribution of a weight function as given in Eq. (10.32), or using Eqs. (10.35*a*) and (10.35*b*) with the diffusion set equal to a constant times the desired cell volume. Other approaches

that have been applied with success are the spring analogy of Nakhashi and Diewert (1986) and the application of the strict equidistribution law to multidimensional problems by Anderson (1983) and Eiseman (1983). Variational methods as outlined in Section 10.4 based on the original work of Brackbill and Saltzman (1982) are also useful in constructing adaptive grids. These methods have been applied to structured meshes in most cases. Mesh redistribution schemes have also been applied to unstructured meshes. The difficulty is that the connectivity must be altered for these cases if the mesh point movement is very large. While the mechanics of changing this connectivity are automatically accounted for in the grid generation algorithm, the associated redistribution of the flux terms for the fluid dynamic variables may not be as easily accomplished when compared with the structured grid approaches. Other ways of r adaption of unstructured grids based on measures of solution quality can be cited. Hagmeijer and Kok (1996), Catherall (1996), Carpenter and McRae (1996), and Riemslagh and Vierendeels (1996) give representative results using these methods.

Adaptive grid construction is applied to both steady flow problems and to time-accurate flow calculations. For adapting grids in a steady flow problem, the grid is adapted or refined after a predetermined number of iterations or time steps have been taken. When the solution converges, the grid will stop adjusting to the changes that occur and will reflect the properties that appear in the solution that have been used to calculate the grid motion and refinement. In the time-accurate case, the grid point motion and refinement are performed in conjunction with the time-accurate solution of a physical problem. This requires the time-accurate coupling of the PDEs of the physical problem and those describing the grid movement or the mesh refinement.

Grid movement schemes can produce substantial improvements in solution quality. However, mesh refinement methods promise significantly better results because no limitations exist that define the limit on grid resolution that can be attained. The Bowyer scheme for generating an unstructured mesh was presented in Section 10.5.3 and can be used as a simple technique to refine a grid to the desired level. The idea of mesh refinement can be applied without limitation to any grid. The idea works if one starts with structured, unstructured, or hybrid grids that are formed of arbitrarily shaped cells. Of course, the use of grid refinement necessitates storage of information as if the grid was unstructured even though the original grid, before refinement, was structured. The rectangular grid schemes that use subdomain division are based on refinement of the mesh until a desired cell size is achieved. The division of cells using either triangular or rectangular shapes (2-D case) relies on splitting an edge or edges of existing cells. When triangular grids are used, the splitting results in construction of new cells where connecting nodes produce either four new cells when each edge is divided or two new cells when only one edge is split. In the case of rectangular cells, the subdivision of cells based on edge splitting leads to extra nodes that appear in the center of edge segments when a cell is divided on one side. This does not create problems, since the numerical method is assumed to be cell

based and the flux terms can be associated with the parent cells and redistributed. The major issue for refinement is correctly managing the database associated with the changing mesh and selecting an appropriate criterion to use to determine the need for additional subdivision. Numerous recent papers show results that illustrate the use of these methods. Examples of recent work include Schneiders (1996), Kallinderis et al. (1993, 1995, 1996), Noack et al. (1996), and Smith and Johnson (1996).

PROBLEMS

10.1 Verify the equations for the transformation metrics given in Eq. (10.5).

10.2 Suppose that a physical domain is defined on the interval $0 \leqslant x \leqslant 1$ with an upper boundary given by

$$y_{\text{upper}} = 1 + 0.2 \sin(\pi x)$$

and a lower boundary given by

$$y_{\text{lower}} = 0.1 \cos(\pi x)$$

Devise a transformation that provides a uniform distribution of mesh points between the upper and lower boundaries. Use a simple normalizing transformation.

10.3 In Prob. 10.2 the interval was defined by two x = const lines. If the left boundary is defined as

$$y_{\text{L}} = 10x$$

and the right boundary is defined by

$$y_{\text{R}} = 4(x - 1)$$

with the same upper and lower boundaries, determine a normalizing transformation to provide equal grid spacing in the physical plane. Why does this become so much more complicated than the transformation of Prob. 10.2?

10.4 Work Prob. 10.2 using the algebraic method demonstrated in Example 10.3. Use linear functions to verify your results and then use cubic functions.

10.5 Work Prob. 10.3 using linear functions with the method given in Example 10.3.

10.6 Suppose that you are required to solve a system of PDEs in (t, x, y) on the rectangular domain

$$0 \leqslant x \leqslant 1$$
$$0 \leqslant y \leqslant 1$$

A surface $F(t, x, y) = 0$ is to be tracked similar to a shock and computed as part of the solution. Devise a transformation that converts the physical plane into two rectangular computational domains joined at the boundary $F(t, x, y) = 0$. Assume that the surface is smooth and always intersects the left and right boundaries in physical space.

10.7 Verify the transformation given in Eq. (10.14) and the associated f_i functions.

10.8 The Thompson scheme for generating grids is based upon Eq. (10.21). Derive the computational domain equations given in Eq. (10.22).

10.9 Show that the mapping governed by the differential equations

$$\nabla^2 \xi = 0$$
$$\nabla^2 \eta = \frac{1}{\eta}$$

maps uniformly spaced circles in physical space into a uniform rectangular grid in the computational plane.

10.10 Show that a solution of the Cauchy-Riemann equations is a solution of Laplace's equation but the reverse is not necessarily true.

10.11 Repeat Prob. 10.3 and use the Thompson technique to obtain the mapping using the method of Middlecoff and Thomas [Eq. (10.25) and Eq. (10.26)] to effectively determine P and Q. Discuss your result, and point out any difficulties encountered in establishing your choice in selecting ϕ and ψ.

10.12 Construct the Euler-Lagrange equations that result when a mesh is obtained using a minimization of the orthogonality measure given by Eq. (10.45).

10.13 Complete the differentiation indicated in Eq. (10.49), and determine the coefficients identified in Eq. (10.50).

10.14 Consider the 1-D form for the Poisson equation. Use a central difference for the second derivative and estimate the maximum value of the control function that may be used before grid crossing occurs.

10.15 The equations given in Prob. 10.2 define the upper and lower boundaries of a physical domain. If the right and left boundaries are straight lines connecting the end points of these defining equations, use transfinite interpolation with Lagrange polynomials to construct a grid covering this domain.

10.16 Work Prob. 10.15 using Hermite polynomials. Show that the proper choice of the coordinate line slopes at the boundaries must be made to prevent grid crossing.

10.17 Develop the TFI expression that may be used to grid an open domain where the outer boundary is not prescribed. Construct a numerical example illustrating this application.

10.18 You have been assigned the task of constructing a grid for a NACA 0012 airfoil. Use parabolic grid generation with Laplace's equation to construct this grid. Select the outer boundary to be uniformly two chord lengths from the body, and use the outer boundary as the forward point in the difference approximation.

10.19 Construct an algorithm using the Bowyer insertion scheme to correctly triangulate a given set of points. Assume the initial Delaunay triangulation is given by a single triangle and insert a total of 10 points to verify your work.

10.20 Using the computer code from Prob. 10.19, insert points in the supertriangle defining a rectangular outer boundary enclosing a NACA 0012 airfoil.

10.21 With the code developed in Prob. 10.19, insert points between the airfoil boundary and the outer boundary to provide adequate resolution to complete a flow field computation. Base the refinement on selecting the largest triangle, and insert a point at the circumcenter of this triangle. Perform this exercise for several grid point densities. Discuss your results and include any problems you identify with this technique.

10.22 Devise a method to eliminate cells interior to the airfoil and exterior to the rectangular outer boundary using the code from the previous problem. Be sure to reorder the cell structure as a continuous list for ease of use with a flow solver.