5

Hyperbolic Methods for Surface and Field Grid Generation

William M. Chan

5.1 Introduction

Two of the most widely used classes of methods for structured grid generation are algebraic methods and partial differential equation (PDE) methods. The PDE methods can be classified into three types: elliptic, parabolic, and hyperbolic. This chapter will focus on the use of hyperbolic partial differential equation methods for structured surface grid generation and field grid generation.

In hyperbolic grid generation, a mesh is generated by propagating in the normal direction from a known level of points to a new level of points, starting from a given initial state. For two-dimensional (2D) field grid generation and for surface grid generation, the initial state is a curve. For three-dimensional (3D) field grid generation, the initial state is a surface. The governing equations are typically derived from grid angle and grid cell size constraints. Local linearization of these equations allows a mesh to be generated by marching from a known state to the next. The total marching distance from the initial state and the marching step sizes at each level can be prescribed based on requirements of the specific application.

When generating 2D field grids or surface grids using algebraic interpolation or elliptic methods, grid points on all four boundaries of a nonperiodic mesh have to be specified prior to the generation of the interior points. Thus, exact control of the mesh boundaries is inherent with such methods. When using

0-8493-2687-7/99/$0.00+$.50

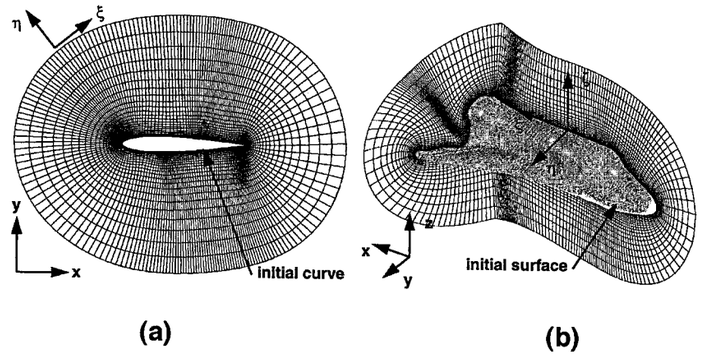

FIGURE 5.1 (a) Example of hyperbolic field grid in two dimensions (airfoil O-grid). The marching direction is given by η. (b) Example of hyperbolic field grid in three dimensions (simplified Space Shuttle Orbiter). The marching direction is given by ζ.

hyperbolic methods, only the initial state can be exactly prescribed as one of the boundaries of the mesh. Exact specification of the side and outer boundaries is not possible with a one sweep marching scheme but limited control is achievable. When exact control of all the mesh boundaries is not needed, less work is required using hyperbolic methods since only one boundary has to be prescribed instead of four. The reduction in effort becomes more significant in 3D field grid generation where only the initial surface needs to be prescribed instead of the six boundary surfaces required for a nonperiodic grid using algebraic interpolation or elliptic methods. Excellent orthogonality and grid clustering characteristics are inherently provided by hyperbolic methods. Since a marching scheme is used, the grid generation time can be one to two orders of magnitude faster than typical elliptic methods.

In a structured grid approach for solving field simulation problems for complex configurations, the complex domain is typically decomposed into a number of simpler subdomains. A grid is generated for each subdomain, and communications between subdomains are managed by a domain connectivity program. The two main methods for domain decomposition are the patched grid approach [Rai, 1986]; [Thompson, 1988] and the Chimera overset grid approach [Steger, Dougherty, and Benek, 1983]. In the patched grid approach, neighboring grids are required to abut each other. Since exact specification of all the grid boundaries is needed, algebraic and elliptic methods are best suited for generating grids for this scheme. In the overset grid approach, neighboring grids are allowed to overlap with each other. This freedom is particularly well suited to hyperbolic grid generation methods. Thus, hyperbolically generated grids are heavily used in most overset grid computations on complex geometries (see Section 5.5 for a sample list of applications; also see Chapter 11).

Field grid generation in 2D using hyperbolic equations was introduced by Starius [1977] and Steger and Chaussee [1980]. A 2D field grid in the Cartesian x-y plane is generated by marching from an initial curve in the plane (see Figure 5.1a). Related work in two dimensions includes that by Kinsey and Barth [1984] for implicitness enhancements, Cordova and Barth [1988] for non-orthogonal grid control, Klopfer [1988] for adaptive grid applications, and Jeng, Shu and Lin [1995] for internal flow problems. Exact prescription of the side boundaries can be achieved by performing elliptic iterations at the end of each step [Cordova, 1991]. Extension of the hyperbolic grid generation scheme to 3D was presented in [Steger and Rizk, 1985]. A 3D volume grid is generated by marching from an initial surface (see Figure 5.1b). Enhancements to the robustness of the basic field grid generation scheme were developed by Chan and Steger [1992]. Hybrid schemes formed by mixing hyperbolic with elliptic and parabolic equations have been used by Nakamura [1987], Steger [1989a], Takanashi and Takemoto [1993].

Surface grid generation using hyperbolic equations was introduced by Steger [1989b]. A surface grid is generated by marching from an initial curve that lies on a reference surface (see Figure 5.2). The basic

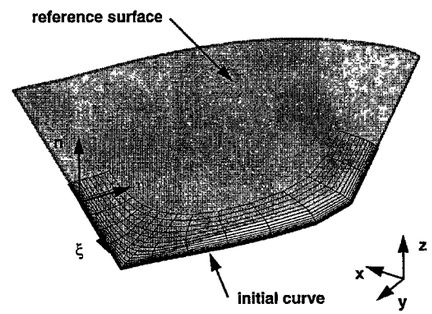

FIGURE 5.2 Example of hyperbolic surface grid (cap grid over wing tip region). The local initial curve direction, local marching direction and local surface normal are indicated by ξ and η, and $\mathbf{n}$, respectively.

scheme allowed only a single rectangular array of quadrilateral cells (single panel network) to be the reference surface. In practical situations, a complex surface geometry is typically described by numerous surface patches where each patch may be a nonuniform rational B-spline (NURBS) surface or some other geometric entity (cf. Part III). For hyperbolic surface grid generation to be applicable in such cases, the scheme has to be able to grow surface grids that can span over multiple patches. A first step toward this goal was made by extending the basic scheme to generate surface grids that can span over multiple panel networks [Chan and Buning, 1995]. Each panel network is used to model a surface patch where the point distribution on the panel network can be made as fine as necessary from the original patch definition.

5.2 Hyperbolic Field Grid Generation

In hyperbolic field grid generation, a field grid is generated in two or three dimensions by marching from a specified initial state. A new grid level is produced by linearizing about the current known level and solving the governing equations. In two dimensions (2D), the initial state is a curve on the Cartesian x-y plane. In three dimensions (3D), the initial state is a surface in three-dimensional space. For practical applications, the initial state is typically chosen to coincide with the configuration body surface to produce body-fitted grids.

5.2.1 Governing Equations for Hyperbolic Field Grid Generation

The governing equations presented below are derived from grid orthogonality and cell size constraints. By demanding that the marching direction be orthogonal to the current known state, an orthogonality relation can be derived for 2D, and two orthogonality relations can be derived for 3D. The system of equations can be closed by a cell area/volume constraint where the local cell areas/volumes are user-specified. A convenient method for specifying cell areas/volumes is described in Section 5.2.3. Other formulations of the governing equations have been used. For example, locally nonorthogonal grids in 2D can be generated by the introduction of an angle source term [Cordova and Barth, 1988]; an arc length constraint can be used instead of the cell area constraint in 2D [Steger and Chaussee, 1980].

In 2D, consider generalized coordinates $\xi(x,y)$ and $\eta(x,y)$. The 2D field grid generation equations can be written as

$$x_\xi x_\eta + y_\xi y_\eta = 0 \tag{5.1a}$$

$$x_\xi y_\eta - y_\xi x_\eta = \Delta A \tag{5.1b}$$

where ΔA is the user-specified local cell area. The initial state is chosen to be at the first η = const. curve.

In 3D, consider generalized coordinates $\xi(x, y, z)$, $\eta(x, y, z)$, and $\zeta(x, y, z)$ corresponding to grid indices j, k, and l, respectively. The 3D field grid generation equations can be written as

$$\vec{r}_\xi \cdot \vec{r}_\zeta = x_\xi x_\zeta + y_\xi y_\zeta + z_\xi z_\zeta = 0 \tag{5.2a}$$

$$\vec{r}_\eta \cdot \vec{r}_\zeta = x_\eta x_\zeta + y_\eta y_\zeta + z_\eta z_\zeta = 0 \tag{5.2b}$$

$$\vec{r}_\zeta \cdot \left(\vec{r}_\xi \times \vec{r}_\eta\right) = x_\xi y_\eta z_\zeta + x_\zeta y_\xi z_\eta + x_\eta y_\zeta z_\xi - x_\xi y_\zeta z_\eta - x_\eta y_\xi z_\zeta - x_\zeta y_\eta z_\xi = \Delta V \tag{5.2c}$$

where $\vec{r} = (x, y, z)^T$ and ΔV is the user-specified local cell volume. The initial state is chosen to be at the first ζ = const. surface.

5.2.2 Numerical Solution of Hyperbolic Field Grid Generation Equations

Local linearization of Eq. 5.1a,b and Eq. 5.2a,b,c results in systems of grid generation equations in 2D and 3D, respectively. Such systems have been shown to be hyperbolic for marching in η in 2D and for marching in ζ in 3D (see [Steger, 1991]). The system of grid generation equations is solved with a noniterative implicit finite difference scheme.

Second-order central differencing is used in the ξ direction in 2D and in the ξ and η directions in 3D. In these directions, appropriate boundary conditions have to be employed (see Section 5.2.4), and smoothing terms have to be added to provide numerical stability (see Section 5.2.5). A first-order implicit scheme is used in the marching direction. An unconditionally stable implicit scheme has the advantage that the marching step size can be selected based only on considerations of grid accuracy. At each marching step, linearization is performed about the previous marching step. More details of the numerical scheme are now presented.

Local linearization of Eq. 5.1 about a known state results in the system of grid generation equations

$$A_0 \vec{r}_\xi + B_0 \vec{r}_\eta = \vec{f} \tag{5.3}$$

where the subscript 0 denotes evaluation at the known state 0, and

$$A = \begin{bmatrix} x_\eta & y_\eta \\ y_\eta & -x_\eta \end{bmatrix} \quad B = \begin{bmatrix} x_\xi & y_\xi \\ -y_\xi & x_\xi \end{bmatrix} \quad \vec{f} = \begin{bmatrix} 0 \\ \Delta A + \Delta A_0 \end{bmatrix} \tag{5.4}$$

The matrix B_0^{-1} exists if $x_\xi^2 + y_\xi^2 \neq 0$. Moreover, $B_0^{-1} A_0$ is a symmetric matrix. Hence, the system in Eq. 5.3 is hyperbolic for marching in η. The numerical solution of the 2D equations follows closely that of the 3D equations. Only the details for the 3D equations are given below.

Local linearization of Eq. 5.2 about a known state 0 results in the system of grid generation equations

$$A_0 \vec{r}_\xi + B_0 \vec{r}_\eta + C_0 \vec{r}_\zeta = \vec{e} \tag{5.5}$$

where

$$A = \begin{bmatrix} x_\zeta & y_\zeta & z_\zeta \\ 0 & 0 & 0 \\ \left(y_\eta z_\zeta - y_\zeta z_\eta\right) & \left(x_\zeta z_\eta - x_\eta z_\zeta\right) & \left(x_\eta y_\zeta - x_\zeta y_\eta\right) \end{bmatrix} \tag{5.6a}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ x_\zeta & y_\zeta & z_\zeta \\ \left(y_\zeta z_\xi - y_\xi z_\zeta\right) & \left(x_\xi z_\zeta - x_\zeta z_\xi\right) & \left(x_\zeta y_\xi - x_\xi y_\zeta\right) \end{bmatrix} \tag{5.6b}$$

$$C = \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ \left(y_\xi z_\eta - y_\eta z_\xi\right) & \left(x_\eta z_\xi - x_\xi z_\eta\right) & \left(x_\xi y_\eta - x_\eta y_\xi\right) \end{bmatrix} \tag{5.6c}$$

and $\vec{e} = (0, 0, \Delta V + 2\Delta V_0)^T$. The matrix C_0^{-1} exists unless $(\Delta V_0) \to 0$. Moreover, $C_0^{-1} A_0$ and $C_0^{-1} B_0$ are symmetric matrices and hence the system of equations is hyperbolic for marching in ζ.

Eq. 5.5 is solved numerically by a noniterative implicit marching scheme in ζ. Additional smoothing and implicitness are attained by differencing $\nabla_\zeta \vec{r} = \vec{F}$ as $\vec{r}_{l+1} - \vec{r}_l = (1 + \theta)\vec{F}_{l+1} - \theta\vec{F}_l$, where values of the implicitness factor θ range between 0 and 4 [Kinsey and Barth, 1984]. After approximate factorization and addition of numerical smoothing terms, the equations to be solved can be written as

$$\begin{aligned} &\left[I + \left(1+\theta_\eta\right) C_l^{-1} B_l \delta_\eta - \varepsilon_{i\eta} (\Delta\nabla)_\eta\right]\left[I + \left(1+\theta_\xi\right) C_l^{-1} A_l \delta_\xi - \varepsilon_{i\xi} (\Delta\nabla)_\xi\right]\left(\vec{r}_{l+1} - \vec{r}_l\right) \\ &= C_l^{-1} \vec{g}_{l+1} - \left[\varepsilon_{e\xi} (\Delta\nabla)_\xi + \varepsilon_{e\eta} (\Delta\nabla)_\eta\right] \vec{r}_l \end{aligned} \tag{5.7}$$

with

$$\delta_\xi \vec{r}_j = \frac{\vec{r}_{j+1} - \vec{r}_{j-1}}{2} \qquad \delta_\eta \vec{r}_k = \frac{\vec{r}_{k+1} - \vec{r}_{k-1}}{2} \tag{5.8a}$$

$$(\Delta\nabla)_\xi \vec{r}_j = \vec{r}_{j+1} - 2\vec{r}_j + \vec{r}_{j-1} \quad (\Delta\nabla)_\eta \vec{r}_k = \vec{r}_{k+1} - 2\vec{r}_k + \vec{r}_{k-1} \tag{5.8b}$$

where $\vec{g}_{l+1} = (0, 0, \Delta V_{l+1})^T$. I is the identity matrix; θ_ξ, θ_η are the implicitness factors in ξ and η, respectively; $\varepsilon_{e\xi}$, $\varepsilon_{e\eta}$ are second-order explicit smoothing coefficients in ξ and η, respectively; $\varepsilon_i \approx 2\varepsilon_e$ in ξ and η; and the subscript l indicates the grid level in the marching direction. The smoothing coefficients can be chosen to be constants or made to vary spatially depending on local geometric demands. Proper choice of spatially varying smoothing coefficients can significantly enhance the robustness of the scheme in cases involving complex geometry (see Section 5.2.5 for more details). Only the indices that change are shown in Eq. 5.8, i.e., $\vec{r}_j \equiv \vec{r}_{j,k,l}$.

The coefficient matrices A_l, B_l and C_l contain derivatives in ξ, η, and ζ. The ξ- and η-derivatives are computed by central differencing, while the ζ-derivatives are obtained from Eq. 5.2 as a linear combination of ξ- and η-derivatives as follows:

$$\begin{pmatrix} x_\zeta \\ y_\zeta \\ z_\zeta \end{pmatrix} = \frac{\Delta V}{Det(C)} \begin{pmatrix} y_\xi z_\eta - y_\eta z_\xi \\ x_\eta z_\xi - x_\xi z_\eta \\ x_\xi y_\eta - x_\eta y_\xi \end{pmatrix} = C^{-1} \vec{g} \tag{5.9}$$

with $Det(C) = (y_\xi z_\eta - y_\eta z_\xi)^2 + (x_\eta z_\xi - x_\xi z_\eta)^2 + (x_\xi y_\eta - x_\eta y_\xi)^2$. In regions with sudden grid spacing jumps in the ξ or η direction, a more robust method for computing the ζ derivatives in described in [Chan and Steger, 1992].

Extra robustness at sharp convex corners can be achieved by demanding that the marching increment $\Delta\vec{r}_l = \vec{r}_{l+1} - \vec{r}_l$ at the corner point be the average of the marching increments of its neighbors. This can be achieved by solving the following averaging equation at the corner point:

$$\Delta\vec{r}_{j,k} = \frac{1}{2}\left(\mu_\xi + \mu_\eta\right)\Delta\vec{r}_{j,k} \tag{5.10}$$

where

$$\mu_\xi \Delta\vec{r}_{j,k} = \frac{1}{2}\left(\Delta\vec{r}_{j+1,k} + \Delta\vec{r}_{j-1,k}\right) \qquad \mu_\eta \Delta\vec{r}_{j,k} = \frac{1}{2}\left(\Delta\vec{r}_{j,k+1} + \Delta\vec{r}_{j,k-1}\right) \tag{5.11}$$

Approximate factorization of Eq. 5.10 gives

$$\left(I - \frac{1}{2}\mu_\xi\right)\left(I - \frac{1}{2}\mu_\eta\right)\Delta\vec{r} = 0 \tag{5.12}$$

which has the same form as the block tridiagonal matrix factors of the hyperbolic equations. A switch is made to solve Eq. 5.12 if a sharp convex corner exists in either the ξ or η direction. For example, the switch can be made if the external angle of the corner is greater than 240°. The averaging equation works particularly well if the surface grid spacings on the two sides of the corner are equal.

5.2.3 Specification of Cell Sizes

The local cell sizes (ΔA in 2D and ΔV in 3D) have to be specified at each point on each grid level as the mesh is marched from the initial state. There is no unique method for prescribing the cell size but a convenient method is described below. Other schemes for cell size specification are described in [Steger and Chaussee, 1980] and [Steger and Rizk, 1985].

For both 2D and 3D, a one-dimensional (1D) stretching function in the marching direction is specified by the user. The stretching function provides the step sizes to be used at each grid level in the marching direction. In 2D, ΔA is computed by the product of the local arc length and the marching step size at the current level. In 3D, ΔV is computed by the product of the local cell area and the marching step size at the current level. The step size specification via the stretching function provides good grid clustering control near the initial state which is usually chosen to be at a solid surface for fluid flow computations. Such clustering control is particularly important in viscous calculations.

The stretching function used is typically geometric or hyperbolic tangent, although any arbitrary stretching function can be employed (cf. Chapter 32). For both geometric and hyperbolic tangent stretching, the total marching distance and the number of points to be used in the marching direction have to be specified. The geometric stretching allows the prescription of grid spacing at one end of the domain only (usually at the initial state). The hyperbolic tangent stretching allows grid spacing at one or both ends of the domain to be specified. This is convenient when it is necessary to control the outer boundary grid spacing. Such a situation frequently arises in overlapping grid systems where it is desirable to have comparable grid spacings at the boundaries of neighboring grids.

For typical applications, the same initial/final spacings and marching distance are applied at every grid point on the initial state. However, certain applications require the use of different initial/final spacings and marching distances at different points on the initial state (see three-element airfoil example in Section 5.5). A convenient method for specifying the variable grid spacings and marching distances is to prescribe these parameters at key control points on the initial state, and then use interpolation to provide their values at the remaining points.

Grid smoothness can be enhanced to a certain degree by performing smoothing steps on the prescribed cell sizes. This has the effect of making the cell sizes more uniform which is typically a desirable

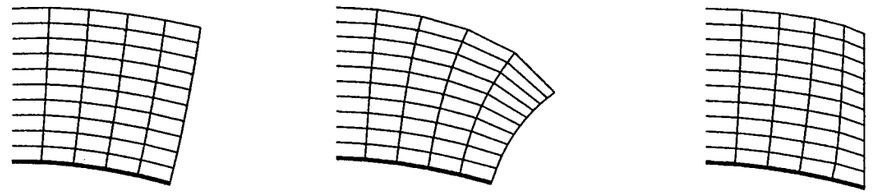

FIGURE 5.3 A field grid slice with (a) a free floating boundary, (b) a free floating boundary with splay, and (c) a constant plane boundary (—— initial curve).

characteristic in the far field regions of a grid. For example, a smoothed cell volume $\Delta\overline{V}_{j,k,l}$ in 3D can be computed as

$$\Delta\overline{V}_{j,k,l} = \left(1 - \nu_a\right)\Delta V_{j,k,l} + \frac{\nu_a}{4}\left(\Delta V_{j+1,k,l} + \Delta V_{j-1,k,l} + \Delta V_{j,k+1,l} + \Delta V_{j,k-1,l}\right) \tag{5.13}$$

where this is applied one or more times under each marching step. A typical value of ν_a that has been employed is 0.16.

5.2.4 Boundary Conditions

Numerical boundary conditions have to be supplied in ξ for 2D cases, and in ξ and η for 3D cases. The boundary conditions used are dictated by the topology of the specified initial state or by the desired boundary behavior of the grid being generated. For example, a periodic initial curve in 2D demands the use of a periodic boundary condition in ξ. For a nonperiodic initial curve in 2D, the user has several choices in influencing the behavior of the grid side boundaries emanating from the two endpoints of the initial curve. The boundaries can be allowed to float freely, splay outward, or made to be at a constant Cartesian plane station (see Figure 5.3).

An implicit boundary scheme is used to implement the above boundary conditions. A periodic solver is used to invert the left-hand-side factor in Eq. 5.7 that corresponds to a periodic direction. A mixed zeroth- and first-order extrapolation scheme is used for the free-floating and splay conditions. For example, the dependent variable $\Delta\vec{r} = (\Delta x, \Delta y, \Delta z)^T = \vec{r}_{l+1} - \vec{r}_l$ at the $j = 1$ boundary can be made to satisfy

$$\left(\Delta\vec{r}\right)_{j=1} = \left(\Delta\vec{r}\right)_{j=2} + \varepsilon_x\left[\left(\Delta\vec{r}\right)_{j=2} - \left(\Delta\vec{r}\right)_{j=3}\right] \tag{5.14}$$

where $0 \le \varepsilon_x \le 1$ is the extrapolation factor. The appropriate elements at the end points of the block tridiagonal matrix on the left-hand side of Eq. 5.7 are modified by Eq. 5.14. A free-floating condition is achieved by setting $\varepsilon_x = 0$. Increasing ε_x from zero has the effect of splaying the boundary of the field grid away from the grid interior. A constant plane condition in x, y, or z can be imposed by simply setting the appropriate component of $\Delta\vec{r}$ to zero. For example, a constant ξ plane condition at the $j = 1$ boundary is set by imposing $(\Delta x, \Delta y, \Delta z)^T_{j=1} = (0, \Delta y, \Delta z)^T_{j=2}$.

In 3D, more complicated topologies are possible with a surface as the initial state. The surface may be

1. Nonperiodic in both ξ and η directions,
2. Periodic in one direction and nonperiodic in the other direction (cylinder topology),
3. Periodic in both directions (torus topology).

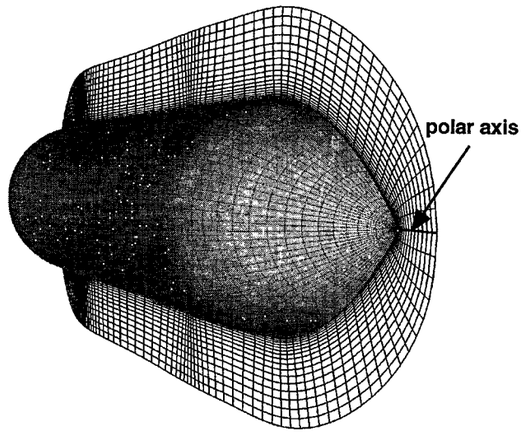

FIGURE 5.4 Surface grid with a singular axis point and slices of volume grid with a polar axis emanating from the singular axis point.

At a nonperiodic boundary, the same nonperiodic boundary schemes may be applied as in the 2D case (see Figure 5.3). Furthermore, singularities may be present at a surface grid boundary such as a singular axis point or a collapsed edge. Special numerical boundary treatment is needed at these boundaries. A singular axis point is a surface grid boundary where all the points are coincident. The volume grid contains a polar axis emanating from the axis point on the surface grid (see Figure 5.4). A collapsed edge condition is sometimes applied at a wing tip under a C-mesh or O-mesh topology. The C-type or O-type grid lines on the wing surface grid collapse to zero thickness at the wing tip to form a collapsed edge. Figure 5.5 shows a collapsed edge case for a C-mesh of a wing. The slice of the volume grid emanating from the collapsed edge forms a singular sheet ($k = kmax$ slice in Figure 5.5). Further illustration of different boundary conditions in 3D are shown in [Chan, Chiu, and Buning, 1993].

5.2.5 Grid Smoothing Mechanisms

There are three mechanisms through which smoothing is supplied to a grid generated with the scheme described above. All three mechanisms can be controlled by the user. The first is through the implicitness factors θ_ξ and θ_η in Eq. 5.7. Values of these parameters in the range 1–4 are mildly effective in preventing crossing of grid lines in concave corner regions in the ξ and η directions, respectively. The second smoothing mechanism is introduced by the number of times the specified cell areas/volumes are smoothed as described in Section 5.2.3. This has the effect of spreading clustered grid lines apart so that the cells sizes tend towards a uniform distribution as the number of smoothing steps is increased. The strongest and the most important is the third smoothing mechanism governed by the second-order smoothing coefficients in Eq. 5.7. These are discussed in more detail below.

The second-order smoothing applied in Eq. 5.7 serves to provide numerical dissipation needed for the central differencing scheme. A direct effect of this smoothing term is the enhancement of grid smoothness, but at the same time, a reduction in grid orthogonality also occurs. For a complex geometry, it is clear that different regions of the field grid require different amounts of added numerical smoothing. A low amount of smoothing is desired in regions where grid orthogonality should dominate. This is typically needed in regions near the body surface and in low curvature regions of the geometry. In concave regions of the surface, a high amount of smoothing is needed to prevent grid lines from crossing. A spatially variable dissipation coefficient based on the above attributes was designed and shown to work well for

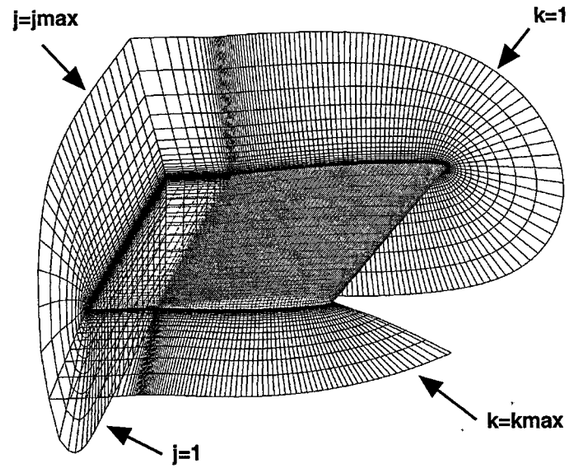

FIGURE 5.5 Surface grid wand slices of volume grid near a collapsed edge for a C-mesh topology. The surface grid has *jmax* by *kmax* points in the j and k directions with the collapsed edge at $k=kmax$.

a wide variety of cases [Chan and Steger, 1992]. Essential highlights of the dissipation model are discussed below. The original reference can be consulted for further details.

Let D_e be the explicit second-order dissipation added to the right-hand side of Eq. 5.7 given by

$$D_e = -\left[\varepsilon_{e\xi}(\Delta\nabla)_\xi + \varepsilon_{e\eta}(\Delta\nabla)_\eta\right]\vec{r}_l \tag{5.15}$$

The coefficients $\varepsilon_{e\xi}$ and $\varepsilon_{e\eta}$ are designed to depend on five quantities as follows:

$$\varepsilon_{e\xi} = \varepsilon_c N_\xi S_l d_\xi a_\xi \qquad \varepsilon_{e\eta} = \varepsilon_c N_\eta S_l d_\eta a_\eta \tag{5.16}$$

The only user-adjustable parameter is ε_c. All other quantities in Eq. 5.16 are automatically computed by the scheme.

1. ε_c is a user-supplied constant of $O(1)$. A default of 0.5 can be used but the level of smoothing in difficult cases can be raised by changing ε_c.
2. Scaling with the local mesh spacing is provided through N_ξ and N_η, which are approximations to the matrix norms $\|C^{-1}A\|$ and $\|C^{-1}B\|$, respectively, given by

$$N_\xi = \sqrt{\frac{x_\zeta^2 + y_\zeta^2 + z_\zeta^2}{x_\xi^2 + y_\xi^2 + z_\xi^2}} \qquad N_\eta = \sqrt{\frac{x_\zeta^2 + y_\zeta^2 + z_\zeta^2}{x_\eta^2 + y_\eta^2 + z_\eta^2}} \tag{5.17}$$

3. The scaling function S_l is used to control the level of smoothing as a function of normal distance from the body surface. It is designed to have a value close to zero near the body surface where grid orthogonality is desired, and to gradually increase to a value of one at the outer boundary.
4. The grid convergence sensor functions d_ξ and d_η are used to locally increase the smoothing where grid line convergence is detected in the ξ and η directions, respectively. The d_ξ function is made to depend on the ratio of the average distances between grid points in the ξ direction at

level $(l-1)$ to that at level l. This ratio is high in concave regions where grid lines are converging and hence more dissipation is provided here. It is of order one or smaller in flat or convex regions where less dissipation is needed. A limiter is used to prevent the value of the d_ξ function from becoming too low in convex regions. The d_η function behaves similarly in the η direction.

5. The grid angle functions a_ξ and a_η are used to locally increase the smoothing at severe concave corner points in the ξ and η directions, respectively. Both a_ξ and a_η are designed to have the value of one except at a severe concave corner point. Extra smoothing is added only at the concave corner point as opposed to the entire concave region as supplied by d_ξ or d_η. Grids for concave angles down to 5° have been obtained with this scheme.

5.3 Hyperbolic Surface Grid Generation

In hyperbolic surface grid generation, a surface grid is generated by marching from a specified initial curve on a given surface geometry (reference surface). As in hyperbolic field grid generation, a new grid level is produced by linearizing about the current known level and solving the governing equations. After each marching step, the new set of points are projected on to the reference surface prior to the next marching step. The scheme described below is independent of the form of the reference surface (see Section 5.3.3).

5.3.1 Governing Equations for Hyperbolic Surface Grid Generation

Consider generalized coordinates $\xi(x, y, z)$ and $\eta(x, y, z)$ and let $\hat{n} = (\hat{n}_1, \hat{n}_2, \hat{n}_3)^T$ be the local unit surface normal. An orthogonality relation is derived by demanding that the local marching direction η be orthogonal to the local curve direction ξ of the current known state. A cell area constraint and the surface tangency of the marching direction are used to close the system by providing the remaining two equations. The governing equations can be written as

$$\vec{r}_\xi \cdot \vec{r}_\eta = x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta = 0 \tag{5.18a}$$

$$\hat{n} \cdot \left(\vec{r}_\xi \times \vec{r}_\eta\right) = \hat{n}_1\left(y_\xi z_\eta - z_\xi y_\eta\right) + \hat{n}_2\left(z_\xi x_\eta - x_\xi z_\eta\right) + \hat{n}_3\left(x_\xi y_\eta - y_\xi x_\eta\right) = \Delta S, \tag{5.18b}$$

$$\hat{n} \cdot \vec{r}_\eta = \hat{n}_1 x_\eta + \hat{n}_2 y_\eta + \hat{n}_3 z_\eta = 0, \tag{5.18c}$$

where $\vec{r} = (x, y, z)^T$ and ΔS is a user-specified surface mesh cell area. This can be prescribed using a similar method as that for ΔA described in Section 5.2.3.

5.3.2 Numerical Solution of Hyperbolic Surface Grid Generation Equations

Local linearization of Eq. 5.18 about a known state 0 results in a system of grid generation equations

$$A_0 \vec{r}_\xi + B_0 \vec{r}_\eta = \vec{f}, \tag{5.19}$$

with

$$A = \begin{bmatrix} x_\eta & y_\eta & z_\eta \\ \hat{n}_3 y_\eta - \hat{n}_2 z_\eta & \hat{n}_1 z_\eta - \hat{n}_3 x_\eta & \hat{n}_2 x_\eta - \hat{n}_1 y_\eta \\ 0 & 0 & 0 \end{bmatrix} \tag{5.20a}$$

$$B = \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ -\hat{n}_3 y_\xi + \hat{n}_2 z_\xi & -\hat{n}_1 z_\xi + \hat{n}_3 x_\xi & -\hat{n}_2 x_\xi + \hat{n}_1 y_\xi \\ \hat{n}_1 & \hat{n}_2 & \hat{n}_3 \end{bmatrix} \tag{5.20b}$$

$$\vec{f} = \begin{bmatrix} 0 \\ \Delta S + \Delta S_0 \\ 0 \end{bmatrix} \tag{5.20c}$$

The matrix B_0^{-1} exists unless the arc length in ξ is zero. Moreover, $B_0^{-1} A_0$ is symmetric and the system of equations is hyperbolic for marching in η (see [Steger 1989b, 1991] for more details). A local unit vector in the marching direction η can be obtained by the cross product of the local unit surface normal $\hat{n}$ with a local unit vector in the ξ direction.

Eq. 5.19 is solved numerically by a non-iterative implicit marching scheme in η, similar to the scheme employed for solving the field grid generation equations described in Section 5.2.2. The nearby known state 0 is taken from the previous marching step. Central differencing with explicit and implicit second-order smoothing is employed in ξ while a two-point backward implicit differencing is employed in η. The numerical scheme can be written as

$$\left[I + (1+\theta) B_k^{-1} A_k \delta_\xi - \varepsilon_i (\Delta\nabla)_\xi\right]\left(\vec{r}_{k+1} - \vec{r}_k\right) = B_k^{-1} \vec{g}_{k+1} - \left[\varepsilon_e (\Delta\nabla)_\xi\right] \vec{r}_k \tag{5.21}$$

where

$$\delta_\xi \vec{r}_j = \frac{\vec{r}_{j+1} - \vec{r}_{j-1}}{2} \qquad (\Delta\nabla)_\xi \vec{r}_j = \vec{r}_{j+1} - 2\vec{r}_j + \vec{r}_{j-1} \tag{5.22}$$

and $\vec{g}_{k+1} = (0, \Delta S_{k+1}, 0)^T$. I is the identity matrix, j, k are the grid indices in ξ and η, respectively, θ is the implicitness factor as introduced for Eq. 5.7, ε_e and ε_i are the explicit and implicit smoothing coefficients, respectively, with $\varepsilon_i \approx 2\varepsilon_e$; These can be chosen to vary spatially as described in Section 5.2.5. Only the indices that change are shown in Eq. 5.21 and Eq. 5.22, i.e., $\vec{r}_{j+1} \equiv \vec{r}_{j+1,k}$, etc.

The elements of A contain derivatives in η. These derivatives can be expressed in terms of derivatives in ξ using Eq. 5.18 and are computed as

$$\begin{pmatrix} x_\eta \\ y_\eta \\ z_\eta \end{pmatrix} = B^{-1} \vec{g} = \frac{1}{\beta} \begin{bmatrix} x_\xi - \hat{n}_1 w & \hat{n}_2 z_\xi - \hat{n}_3 y_\xi & \hat{n}_1 s_\xi^2 - x_\xi w \\ y_\xi - \hat{n}_2 w & \hat{n}_3 x_\xi - \hat{n}_1 z_\xi & \hat{n}_2 s_\xi^2 - y_\xi w \\ z_\xi - \hat{n}_3 w & \hat{n}_1 y_\xi - \hat{n}_2 x_\xi & \hat{n}_3 s_\xi^2 - z_\xi w \end{bmatrix} \vec{g} \tag{5.23}$$

where

$$w = \hat{n} \cdot \vec{r}_\xi = \hat{n}_1 x_\xi + \hat{n}_2 y_\xi + \hat{n}_3 z_\xi \tag{5.24a}$$

$$s_\xi^2 = \vec{r}_\xi \cdot \vec{r}_\xi = x_\xi^2 + y_\xi^2 + z_\xi^2 \tag{5.24b}$$

$$\beta = Det(B) = s_\xi^2 - w^2 \tag{5.24c}$$

5.3.3 Communications with the Reference Surface

At the beginning of each marching step, the local unit surface normal at each point on the current known state have to be computed. These normals are needed in the matrices for marching the grid generation equations. After each marching step, the new points have to be projected back onto the reference surface. For a high-quality grid, the local step size should be small relative to the local curvature of the surface. This ensures that the distances moved by the grid points due to projection are small, which would in turn guarantee that the final grid spacings in the marching direction are close to the step sizes originally specified.

Each hyperbolic marching step is performed independently from the surface normal evaluation before the step and the point projection after the step. This implies that different representations of the reference surface can be easily substituted if routines are provided to

1. Compute the surface normal at a given point on the reference surface,
2. Project a given point onto the reference surface.

A scheme is outlined below for the above two steps for a reference surface consisting of a collection of multiple panel networks. Each panel network contains a rectangular array of points. The surface patch represented by these points is typically approximated by a set of bilinear quadrilaterals with vertices located at the points.

Surface normals on a panel network can be computed as follows. The surface normal of a quadrilateral is given by the cross-product of its diagonals. The surface normal at a vertex point on the panel network is then computed as the average of the surface normals of the quadrilaterals that share the vertex. For a given point on the panel network, bilinear interpolation of the normals at the vertices of the quadrilateral on which the point lies is used to obtain the normal at the point.

A stencil walk method can be used to project a given point onto the multiple panel networks. First, Cartesian bounding boxes of each panel network are employed to determine the set of panel networks that may contain the point. Next, a quadrilateral from the set of candidate panel networks that is closest to the given point is taken to be the starting location of the stencil walk. On seeking bilinear interpolation coefficients of the given point on the chosen quadrilateral, the results either indicate the point is inside the quadrilateral, or the next quadrilateral in the appropriate direction should be tried. When the stencil walk hits a boundary of the panel network, the walk continues on to the neighboring network if there is one. In practical situations, small gaps may exist between neighboring panel networks. A tolerance parameter may be used to extrapolate the boundaries of each panel network to cover the gaps for projection purposes.

The stencil walk continues until the point converges inside a quadrilateral. For points close to the reference surface, the stencil walk typically converges very quickly. However, if the point is far away from the reference surface (e.g., as a result of taking too large a marching step relative to the curvature of the surface), convergence may not occur or the point may converge to an erroneous location. For further information, see Chapter 29.

5.4 General Guidelines for High-Quality Grid Generation

Hyperbolic grid generation requires the specification of an initial state from which a field or surface grid is generated. The grid point distribution on the initial state directly affects the quality of the hyperbolic grid that can be produced. Two important areas of concern are described below.

5.4.1 Grid Stretching

In a given direction, let the grid spacings on each side of an interior point be Δs_1 and Δs_2. The grid stretching ratio R at an interior point in the given direction is defined to be

$$R = \max(\Delta s_1, \Delta s_2)/\min(\Delta s_1, \Delta s_2) \tag{5.25}$$

TABLE 5.1 Approximate Speeds of Hyperbolic Field Generator on a Variety of Platforms

Platform	Approximate Speed (number of points generated per CPU second)
CRAY C-90	220,000
SGI R10000 175 MHz	28,000
SGI R4400 250 MHz	20,000
HP 9000/755 99 MHz	16,000
Pentium PC 90 MHz	1,400

In order to limit truncation error, a stretching ratio of about 1.3 should not be exceeded in any direction on the initial state and in the marching direction, i.e., large and sudden jumps in the grid spacings in any direction should be avoided.

5.4.2 Point Distribution Near Corners

Proper grid point placement near convex and concave corners on the initial state can significantly enhance the quality of the resulting hyperbolic grid. The grid spacings on each side of a convex or concave corner should be equal. Moreover, grid points should be clustered toward a convex corner with sharper corners requiring more clustering. These grid properties are desirable for producing smooth grids and are also essential for satisfactorily resolving the flow around the corner. On the other hand, grid points should *not* be clustered into a concave corner on the initial state. A uniform or declustered grid spacing at the concave corner can significantly reduce the tendency for grid lines to converge as the grid is marched out from the corner.

5.5 Applications

In field grid generation, hyperbolic methods are usually used to produce body-fitted grids, i.e., the initial states are chosen to lie on the body surface of the configuration. Such methods have been frequently employed in single grid computations where the outer boundary of the grid lies in the far field. These methods have been equally successful in producing multiple body-fitted grids in complex configurations using the overset grid approach. In such applications, individual grids are typically generated independently of each other and the outer boundaries are not too far from the body surface. The freedom of allowing neighboring grids to overlap makes hyperbolic grids particularly well suited for this gridding approach.

Typical speeds of a hyperbolic field grid generator for a 3D problem on a number of computing platforms are given in Table 5.1. The speed is given by the number of grid points generated per CPU second, e.g., a 3D field with 220,000 points requires about 1 CPU second to generate on the Cray C-90. Typical speed of a hyperbolic surface grid generator is about 20,000 points per CPU second on a SGI R10000 machine.

Sample grids from several overset grid configurations are presented in the subsections below. Other interesting applications not shown here include the F-18 Aircraft [Rizk and Gee, 1992], a joined-wing configuration [Wai, Herling, and Muilenburg, 1994], the RAH-66 Comanche helicopter [Duque and Dimanlig, 1994, 1995], and various marine applications [Dinavahi and Korpus, 1996].

5.5.1 Applications Using 2D Hyperbolic Field Grids

5.5.1.1 Three-Element Airfoil

The first example on the use of 2D hyperbolic field grid generation is a three-element airfoil configuration consisting of five grids shown in Figures 5.6a–5.6d ([Rogers, 1994]). Hyperbolic grids are generated