# Enhancements of a Three-Dimensional Hyperbolic Grid Generation Scheme*

William M. Chan

*MCAT Institute*
*NASA-Ames Research Center*
*Mail Stop T045-2*
*Moffett Field, California 94035*

and

Joseph L. Steger**

*Department of Mechanical, Aeronautical and Materials Engineering*
*University of California at Davis*
*Davis, California 95616*

Transmitted by Joe F. Thompson

## ABSTRACT

A hyperbolic grid generation scheme formulated from grid orthogonality and cell volume specification has been significantly enhanced so that high quality three-dimensional grids can be obtained for a wide variety of geometries. While the speed of the scheme remains one to two orders of magnitude faster than typical elliptic grid generation methods, the robustness of the scheme has been greatly improved over previous applications of the three-dimensional hyperbolic grid generation procedure. Enhancements included the use of spatially-variable smoothing coefficient, metric correction procedures, local treatment of severe convex corners, and new extrapolation treatments of floating and axis boundaries. The versatility of the new hyperbolic grid generation scheme is demonstrated by three-dimensional grids generated for external components of the integrated Space Shuttle vehicle and the SOFIA telescope.

*The authors are grateful to Dr. Pieter Buning and Dr. Yehia Rizk for some very useful discussions. A special thanks is owed to Mr. Chris Atwood for providing the surface grid for the SOFIA telescope.
**Joseph Steger died on May 1, 1992; Computation Fluid Dynamics has lost a giant.

## 1. INTRODUCTION

One of the most popular approaches for generating structured grids is by the solution of a set of partial differential equations. The governing equations can be classified into three types: elliptic, parabolic and hyperbolic. The most widely used grid generation methods require the solution of a set of elliptic equations [4, 14–16]; however, parabolic and hyperbolic equations have also been successfully employed [3, 6, 8, 10, 12, 13] and are advantageous for certain applications.

Since the solution of the elliptic equations satisfies the maximum principle, the grids generated are typically smooth. Moreover, the formulation of the elliptic equations allows exact specifications of all boundary point locations. However, grid orthogonality cannot be maintained with conventional elliptic grid generation methods and boundary surface refinement can be difficult to impose. The user input required to set up boundary distributions can also be time consuming. Hyperbolic grid generation schemes produce nearly orthogonal grids, have excellent clustering control, and can be generated in one to two orders of magnitude less computer time than elliptic methods. However, hyperbolic grid generation methods are less robust, tend to propagate input discontinuities, and the outer boundary location cannot be precisely specified. Hence, they are usually restricted to the generation of grids for external flows or for chimera overset-grid schemes [2], where the exact location of the outer boundary is not constrained. This latter application has become more important, however, and this, coupled with the efficiency and often superior grid quality obtained with hyperbolic grid generation schemes, motivates investigation into improving their robustness.

In the last few years, the hyperbolic grid generation algorithm described in [13] has evolved to include some significant enhancements. A wider range of boundary conditions can now be treated and some feedback features have been added so that dissipative terms (which give the equations a somewhat parabolic nature and smoothness) are now automatically adjusted depending on grid evolution or character. The resulting hyperbolic grid generation scheme is significantly more robust, produces higher quality grids, and can treat a wider variety of topologies.

The governing equations for three-dimensional hyperbolic grid generation are presented in Section 2. The numerical marching scheme employed to solve these equations is described in Section 3. Three factors important in controlling grid quality, the boundary conditions, cell volume specification and added smoothing are discussed in Sections 4, 5 and 6, respectively. Discontinuities due to body shape or initial grid point distribution can present special problems to a hyperbolic solver that dissipation alone cannot satisfactorily cure. Corner points are especially difficult. A metric correction procedure which is essential in providing smooth grids at corners with uneven grid spacings is described in Section 7. Extra robustness at sharp convex corners can be achieved by switching from solving the hyperbolic equations to some other equations at the convex corner points. These schemes are described in Section 8. Two grid quality checks built-in to the grid generation code are described briefly in Section 9. In Section 10, examples from grids produced with the new hyperbolic grid generator for various external components of the Space Shuttle launch vehicle and the SOFIA telescope are presented. Finally, conclusions are given in Section 11.

## 2. GOVERNING EQUATIONS

Generalized coordinates $\xi(x, y, z)$, $\eta(x, y, z)$, $\zeta(x, y, z)$ are sought where the body surface is chosen to coincide with $\zeta(x, y, z) = 0$ and the surface distributions of $\xi = \text{const}$ and $\eta = \text{const}$ are user-specified. With external aerodynamic applications in mind, the location of the outer boundary $\zeta(x, y, z) = \zeta_{\max}$ is not specified. The governing equations are derived from orthogonality relations between $\xi$ and $\zeta$, between $\eta$ and $\zeta$, and a cell volume or finite Jacobian $J$ constraint [13]:

$$x_\xi x_\zeta + y_\xi y_\zeta + z_\xi z_\zeta = 0, \tag{2.1a}$$

$$x_\eta x_\zeta + y_\eta y_\zeta + z_\eta z_\zeta = 0, \tag{2.1b}$$

$$x_\xi y_n z_\zeta + x_\zeta y_\xi z_\eta + x_\eta y_\zeta z_\xi - x_\xi y_\zeta z_\eta - x_\eta y_\xi z_\zeta - x_\zeta y_\eta z_\xi = \Delta V, \tag{2.1c}$$

or, with $\vec{r}$ defined as $(x, y, z)^T$

$$\vec{r}_\xi \cdot \vec{r}_\zeta = 0, \tag{2.2a}$$

$$\vec{r}_\eta \cdot \vec{r}_\zeta = 0, \tag{2.2b}$$

$$\left| \frac{\partial(x, y, z)}{\partial(\xi, \eta, \zeta)} \right| = J^{-1} = \Delta V. \tag{2.2c}$$

Equation (2.1) comprise a system of nonlinear partial differential equations in which $x$, $y$, and $z$ are specified as initial data at $\zeta = 0$. Local linearization of (2.1) about the state 0 results in the system of grid generation equations

$$A_0(\vec{r}-\vec{r}_0)_\xi + B_0(\vec{r}-\vec{r}_0)_\eta + C_0(\vec{r}-\vec{r}_0)_\zeta = \vec{f} \tag{2.3}$$

with

$$A = \begin{pmatrix} x_\zeta & y_\zeta & z_\zeta \\ 0 & 0 & 0 \\ (y_\eta z_\zeta - y_\zeta z_\eta) & (x_\zeta z_\eta - x_\eta z_\zeta) & (x_\eta y_\zeta - x_\zeta y_\eta) \end{pmatrix}, \tag{2.4a}$$

$$B = \begin{pmatrix} 0 & 0 & 0 \\ x_\zeta & y_\zeta & z_\zeta \\ (y_\zeta z_\xi - y_\xi z_\zeta) & (x_\xi z_\zeta - x_\zeta z_\xi) & (x_\zeta y_\xi - x_\xi y_\zeta) \end{pmatrix}, \tag{2.4b}$$

$$C = \begin{pmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ (y_\xi z_\eta - y_\eta z_\xi) & (x_\eta z_\xi - x_\xi z_\eta) & (x_\xi y_\eta - x_\eta y_\xi) \end{pmatrix}, \tag{2.4c}$$

and

$$\vec{f} = \begin{pmatrix} -\left(\dfrac{\partial \vec{r}}{\partial \xi}\cdot\dfrac{\partial \vec{r}}{\partial \zeta}\right)_0 \\ -\left(\dfrac{\partial \vec{r}}{\partial \eta}\cdot\dfrac{\partial \vec{r}}{\partial \zeta}\right)_0 \\ \Delta V - \Delta V_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \Delta V - \Delta V_0 \end{pmatrix}. \tag{2.4d}$$

We can rewrite (2.3) as

$$A_0\vec{r}_\xi + B_0\vec{r}_\eta + C_0\vec{r}_\zeta + \vec{e} \tag{2.5}$$

with $\vec{e} = (0, 0, \Delta V + 2\Delta V_0)^T$. Now $C_0^{-1}$ exists unless $(\Delta V_0) \to 0$, (2.5) can be rewritten as

$$C_0^{-1}A_0\vec{r}_\xi + C_0^{-1}B_0\vec{r}_\eta + \vec{r}_\zeta = C_0^{-1}\vec{e}. \tag{2.6}$$

Although the algebraic verification is not trivial, $C_0^{-1}A_0$ and $C_0^{-1}B_0$ are symmetric matrices [13]. This implies that the linearized system (2.6) is hyperbolic and can be marched with $\zeta$ serving as the "time-like" direction.

## 3. NUMERICAL MARCHING SCHEME

The system of grid generation equations given by (2.5) are solved with a non-iterative implicit finite difference scheme. An unconditionally stable implicit scheme has the advantage that the marching step size in $\zeta$ can be arbitrarily selected based only on considerations of accurately generating the grid. Linearization is performed about the previous marching step in $\zeta$.

Let $\Delta\xi = \Delta\eta = \Delta\zeta = 1$ such that $\xi = j - 1$, $\eta = k - 1$, and $\zeta = l - 1$. Central spatial differencing of (2.5) in $\xi$ and $\eta$ with two-point backward implicit differencing in $\zeta$ leads to

$$A_l\delta_\xi\left(\vec{r}_{l+1} - \vec{r}_l\right) + B_l\delta_\eta\left(\vec{r}_{l+1} - \vec{r}_l\right) + C_l\nabla_\zeta\vec{r}_{l+1} = \vec{g}_{l+1}, \tag{3.1}$$

where

$$\vec{g}_{l+1} = \begin{pmatrix} 0 \\ 0 \\ \Delta V_{l+1} \end{pmatrix}$$

and

$$\delta_\xi\vec{r}_j = \frac{\vec{r}_{j+1} - \vec{r}_{j-1}}{2}, \quad \delta_\eta\vec{r}_k = \frac{\vec{r}_{k+1} - \vec{r}_{k-1}}{2},$$

$$\nabla_\zeta\vec{r}_{l+1} = \vec{r}_{l+1} - \vec{r}_l.$$

Throughout, only those indices that change are indicated; thus, $r_{l+1} \Rightarrow r_{j,k,l+1}$ and $r_{j+1} \Rightarrow r_{j+1,k,l}$, etc.

Multiplying through by $C_l^{-1}$ and approximately factoring gives

$$\left(I + C_l^{-1}B_l\delta_\eta\right)\left(I + C_l^{-1}A_l\delta_\xi\right)\left(\vec{r}_{l+1} - \vec{r}_l\right) = C_l^{-1}\vec{g}_{l+1}, \tag{3.2}$$

where $I$ is the identity matrix. The problem is now reduced to solving a sequence of block tridiagonal systems.

Since all $\xi$- and $\eta$-derivatives are approximated by central differencing, numerical dissipation terms are added in these directions. For simplicity, only second differences are used which are explicitly and implicitly included in the basic algorithm as

$$\left[I + C_l^{-1}B_l\delta_\eta - \varepsilon_{i\eta}(\Delta\nabla)_\eta\right]\left[I + C_l^{-1}A_l\delta_\xi - \varepsilon_{i\xi}(\Delta\nabla)_\xi\right]$$

$$\times\left(\vec{r}_{l+1} - \vec{r}_l\right) = C_l^{-1}\vec{g}_{l+1} - \left[\varepsilon_{e\xi}(\Delta\nabla)_\xi + \varepsilon_{e\eta}(\Delta\nabla)_\eta\right]\vec{r}_l, \quad (3.3)$$

where, for example,

$$(\Delta\nabla)_\eta\vec{r} = \vec{r}_{k+1} - 2\vec{r}_k + \vec{r}_{k-1},$$

and with $\varepsilon_{i\xi} \approx 2\varepsilon_{e\xi}$ and $\varepsilon_{i\eta} \approx 2\varepsilon_{e\eta}$. Additional smoothing and implicitness [5] are achieved by differencing $\nabla_\zeta\vec{r} = \vec{F}$ as $\vec{r}_{l+1} - \vec{r}_l = (1 + \theta)\vec{F}_{l+1} - \theta\vec{F}_l$, with $\theta \geqslant 0$. This differencing in $\zeta$ is incorporated into (3.3) as

$$\left[I + (1 + \theta_\eta)C_l^{-1}B_l\delta_\eta - \varepsilon_{i\eta}(\Delta\nabla)_\eta\right]$$

$$\times\left[I + (1 + \theta_\xi)C_l^{-1}A_l\delta_\xi - \varepsilon_{i\xi}(\Delta\nabla)_\xi\right]$$

$$\times\left(\vec{r}_{l+1} - \vec{r}_l\right) = C_l^{-1}\vec{g}_{l+1} - \left[\varepsilon_{e\xi}(\Delta\nabla)_\xi + \varepsilon_{e\eta}(\Delta\nabla)_\eta\right]\vec{r}_l. \quad (3.4)$$

The values of $\theta_\xi$ and $\theta_\eta$ are kept at zero unless the body contains concave profiles in $\xi$ or $\eta$, in which case values of the appropriate $\theta$ of 1–4 are effective in preventing grid lines from crossing.

The coefficient matrices $A_l$, $B_l$ and $C_l$ contain derivatives in $\xi$, $\eta$ and $\zeta$. The derivatives in $\xi$ and $\eta$ are obtained by central differencing while the derivatives in $\zeta$ are obtained from (2.1) as a linear combination of $\xi$- and $\eta$-derivatives as follows

$$\begin{pmatrix} x_\zeta \\ y_\zeta \\ z_\zeta \end{pmatrix} = \frac{\Delta V}{\mathrm{Det}(C)}\begin{pmatrix} y_\xi z_\eta - y_\eta z_\xi \\ x_\eta z_\xi - x_\xi z_\eta \\ x_\xi y_\eta - x_\eta y_\xi \end{pmatrix} = C^{-1}\vec{g} \quad (3.5)$$

with

$$\text{Det}(C) = (y_\xi z_\eta - y_\eta z_\xi)^2 + (x_\eta z_\xi - x_\xi z_\eta)^2 + (x_\xi y_\eta - x_\eta y_\xi)^2.$$

A discussion on a more appropriate way to compute these $\zeta$-derivatives is given in Section 7.

## 4. BOUNDARY CONDITIONS

Five types of implicit boundary conditions have been implemented in the grid generation code at the $\xi$ and $\eta$ boundaries (except for the axis condition, which is only implemented in $\xi$). In the following, coordinate increments $\Delta\vec{r} = \vec{r}_{l+1} - \vec{r}_l$ are represented by $(\Delta x, \Delta y, \Delta z)^T$.

1. *Periodicity*: All derivatives at the end points in the periodic direction are evaluated by 'wrapping around.' A periodic block-tridiagonal solver is used for the inversion of the appropriate factor in the left-hand side of (3.4).

2. *Constant Cartesian plane*: If a $\xi$ or $\eta$ boundary is restricted to an $x$ = const, $y$ = const or $z$ = const plane, then that value is enforced and the other variables are 'floated.' For example, for an $x$ = const plane at the $j = 1$ boundary, $x$ is held constant and $y$ and $z$ are floated using

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}_{j=1} = \begin{pmatrix} 0 \\ \Delta y \\ \Delta z \end{pmatrix}_{j=2}. \tag{4.1}$$

3. *Symmetry plane*: Conventional reflection planes are used to impose symmetry about any $x = 0$, $y = 0$ or $z = 0$ plane and values are updated implicitly. For example, to update a reflected plane at $j = 1$ for a symmetry condition about $x = 0$ corresponding to $j = 2$, $x$ reflects odd and $y$ and $z$ reflect even as:

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}_{j=1} = \begin{pmatrix} -\Delta x \\ \Delta y \\ \Delta z \end{pmatrix}_{j=3}. \tag{4.2}$$

Since the $j = 2$ plane in this example may deviate very slightly from $x = 0$ away from the body surface due to round-off error, post-processing is done to set the $x$ coordinate exactly to zero at $j = 2$ at each incremental level in $l$.

4. *Floating edge*: Much as in the case of a constant Cartesian plane, an entire $\xi$ or $\eta$ boundary can be floated using the simple hyperbolic equation $\vec{r}_{\xi\zeta} = 0$ or $\vec{r}_{\eta\zeta} = 0$ to update a boundary plane. This is essentially a zeroth-order extrapolation of $\Delta\vec{r}$ from the adjacent interior value, and it often works well. Occasionally, however, the floating boundary plane itself may tend to 'roll in' or 'kink' while its neighboring planes in the interior remain smooth. Since the interior points are fine, this problem has been remedied by the addition of a fictitious line of points on the body surface next to the floating edge by linear extrapolation from the interior. The dimension of the surface grid in a particular direction is temporarily increased by one or two depending on whether one or two floating edges are present in that direction. The 3D grid is then generated over the extended surface grid, after which the fictitious plane or planes of points is removed. The addition and removal of these fictitious points are carried out internal to the code and is not a concern for the user.

When using the chimera overset-grid scheme, it is particularly desirable to have the floating edges splay outwards, i.e., in the direction away from the interior of the grid, thus providing better overlap between neighboring grids. Although a free floating edge using zeroth-order extrapolation may bend inwards or outwards, it is found that using a mixed zeroth- and first-order extrapolation scheme tends to bend the edge outwards. For example, at the $j = 1$ boundary, we have

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}_{j=1} = \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}_{j=2} + \varepsilon_{ex}\left[\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}_{j=2} - \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}_{j=3}\right], \tag{4.3}$$

where $0 \leqslant \varepsilon_{ex} \leqslant 1$. Zeroth-order and first-order extrapolation schemes are recovered at the two limits of $\varepsilon_{ex}$ respectively. Typically, a value of $\varepsilon_{ex} \approx 0.05-0.2$ is used. An example using the floating edge condition with $\varepsilon_{ex} = 0.2$ at the boundaries of a flat plate is shown in Figure 1.

5. *Axis*: When the axis logic is used in the $j$-direction, it is assumed that the boundary condition in the $k$-direction is either periodic, or that of symmetry, or constant planes at both ends. For example, one may generate a grid for a complete ellipsoid (periodic in $k$) or just one half of it (symmetry or constant planes at end points in $k$).

The treatment of the axis requires special attention in order to produce smooth results. Two methods have been developed. The first method requires the user to adjust certain input parameters and is able to produce smooth results for all cases encountered so far. The second method solves the

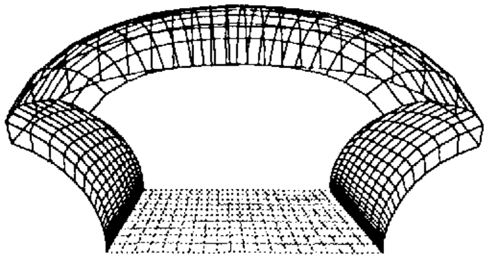

FIG. 1. Outward-splaying edges of a flat plate.

governing equations on a local coordinate system spanning points to either side of the axis. It has the advantage of having fewer parameters to adjust, but it is not appropriate for a conical-like axis and will not be described here.

The first method involves using a mixed zeroth- and first-order extrapolation and volume scaling. The axis point is updated implicitly by imposing that $\Delta\vec{r}$ at the axis is extrapolated by a mixed zeroth- and first-order scheme similar to that given by (4.3). The resulting $k_{\max}$ predicted values, where $k_{\max}$ is the number of points around the axis, are averaged to produce a unique value at the axis. The methods of volume specification described in Section 5 below usually produce volumes that are too large near the axis. Hence, the volumes in the ring of points around the axis are scaled by a reduction factor in the range 0.1–1.0. Typical values of the extrapolation factor and volume scaling factor are 0.4 and 0.3, respectively. Figure 2 shows the symmetry plane of an external tank grid with an axis at the pointed nose and an axis at the flat back. Smoothness is maintained at both the front and the back regions by application of the above scheme.

## 5. CELL VOLUME SPECIFICATION

With the hyperbolic grid generation method, one of the means of controlling the grid is by specification of the cell volumes, $\Delta V_{j,k,l}$. Through the cell volumes, the extent and clustering of the grid can be modified. Since the cell volume at each point must be given, it is clear that the user must devise a simple global method for specifying volumes.

One method is that the specified volume at each point is set equal to the computed surface area element times a user-specified arc length. Specifically,

$$\Delta V_{j,k,l} = \Delta s_{j,k,l}\,\Delta A_{j,k,l}, \tag{5.1}$$

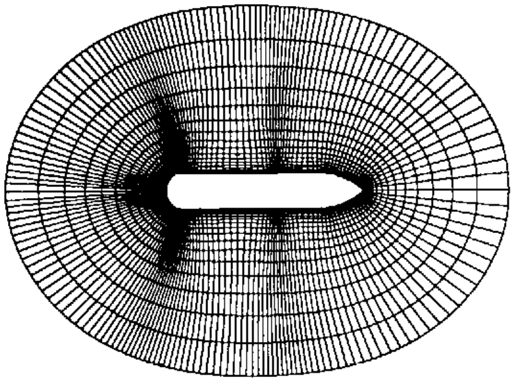

FIG. 2. Symmetry plane of external tank grid with a pointed axis in front and a flat axis at the back.

where $\Delta s_{j,k,l} = s_{j,k,l+1} - s_{j,k,l}$ is the user-specified arc length for marching and $\Delta A_{j,k,l}$ is the surface area element. The radial point distribution function $s_{j,k,l}$ prescribes the arc length between points in the direction normal to the body surface. In the most general case, $s$ is a function of $j$, $k$ and $l$ since each point on the body surface may be stretched to a different outer boundary location. (For example, in the case of a body at a positive angle of attack in hypersonic flow, one may wish to have the outer boundary of the grid farther away on the top surface than on the lower surface). Typically, the points are stretched away from the body exponentially. If grid point spacing control is required at the outer boundary as well as at the body surface, or if more uniform grid point spacing is required away from the inner body surface, hyperbolic tangent stretching can be used. The ability to control the grid spacing at each end of the domain is useful when multiple zones consisting of stretched points or uniformly-spaced points are desired. Also, the presence of more points in the far field of a component grid helps to improve grid-overlap regions for the chimera overset-grid scheme [2].

In this kind of volume control specification, if an initial distribution of points is highly clustered in $\xi$ or $\eta$, then these points tend to remain highly clustered even far away from the body. In order to obtain a more uniform far-field distribution, the volumes specified from (5.1) are averaged in $\xi$ and $\eta$ with each step taken in $\zeta$. For example, the averaged volume $\Delta \overline{V}_{j,k,l}$ can be computed as

$$\Delta \overline{V}_{j,k,l} = (1 - \nu_a)\, \Delta V_{j,k,l} + \frac{\nu_a}{4} \times \left(\Delta V_{j+1,k,l} + \Delta V_{j-1,k,l} + \Delta V_{j,k+1,l} + \Delta V_{j,k-1,l}\right), \quad (5.2)$$

where this is applied one or more times with each step in $\zeta$. A typical value of $\nu_a$ that has been employed is 0.16.

An alternative method for specifying mesh cell volumes is to form a grid about a 'similar' but simple reference body for which the grid can be generated analytically, and to use appropriately scaled cell volumes from this reference grid for the more complex problem. This method is described in more detail in [13].

## 6. SMOOTHING

Since central differencing is used in the numerical scheme, artificial dissipation terms are added to control oscillations arising from the odd-even uncoupling of grid points. An equally important function of the added dissipation is to control the smoothness of the resulting grid. The form and magnitude of the added dissipation are very important in shaping the grid quality.

The parameters $\theta_\xi$ and $\theta_\eta$ on the left-hand side of the grid generation Equation (3.4) can be thought of as a type of smoothing in the $\zeta$ marching direction. In the $\xi$ and $\eta$ directions, it is adequate for the purpose of controlling the smoothness of the grid to use just second-order smoothing. The form of this smoothing is described below.

For simple surface topologies, such as an ellipsoid that has a convex profile in all directions, a constant dissipation coefficient is sufficient to provide good grid quality. However, body surfaces encountered in aerodynamic applications are frequently much more complex with combinations of sharp convex and concave corners. Hence, the dissipation, which tends to reduce grid curvature and orthogonality, must be applied more selectively. The reduction of grid curvature can be advantageous in concave regions and detrimental in convex regions. In order to prevent grid lines from converging and crossing, the dissipation has to be relatively high in concave regions (e.g., in the region between the Orbiter fuselage and the wing root in Figure 3). Conversely, the dissipation must be kept small near the body surface and sharp convex corners (such as the convex corners of the IEA box in Figure 4), otherwise the resulting grid spacing in the $\zeta$ direction would become too reduced or even negative. A spatially uniform dissipation coefficient is unable to satisfy all of the above requirements. Spatially-varying dissipation coefficients have been used previously, but they have only accounted for variations of mesh size, specifically, $C^{-1}A$ and $C^{-1}B$ variations. A spatially-varying form of the dissipation coefficient that has worked very well for many cases is described below.

The explicit second-order dissipation $D_e$ added to the right hand side of

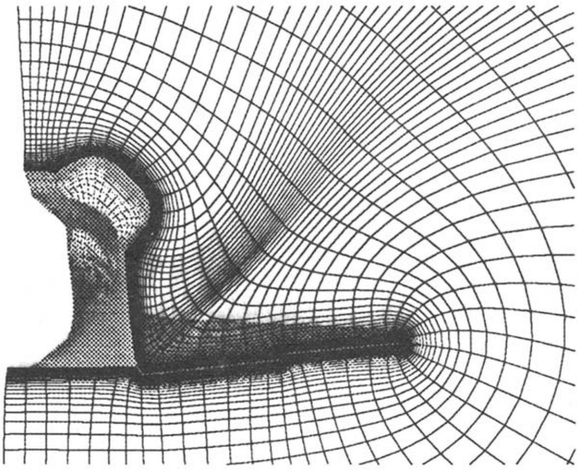

FIG. 3. Plane on the back section of the Orbiter.

the equations is given by

$$D_e = -\left[\varepsilon_{e\xi}(\Delta\nabla)_\xi + \varepsilon_{e\eta}(\Delta\nabla)_\eta\right]\vec{r}_l, \tag{6.1}$$

with

$$\varepsilon_{e\xi} = \varepsilon_e R_\xi N_\xi, \quad \varepsilon_{e\eta} = \varepsilon_e R_\eta N_\eta, \tag{6.2}$$

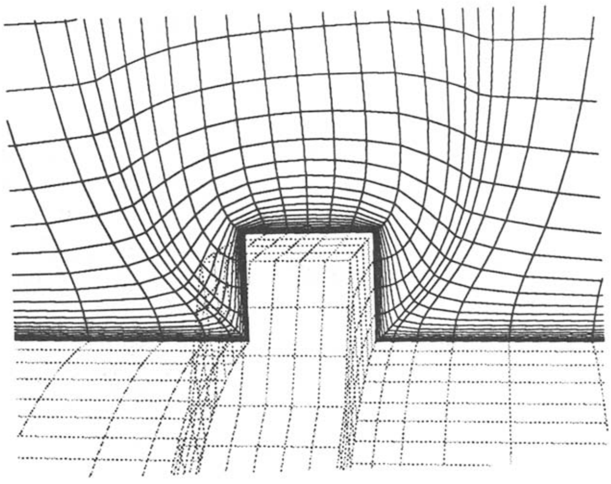

FIG. 4. Plane through IEA box on the ring of the solid rocket booster.

where $\varepsilon_c$ is a user-specified constant of $O(1)$, $N_\xi$ and $N_\eta$ are approximations to the matrix norms $\|C^{-1}A\|$ and $\|C^{-1}B\|$, respectively, given by

$$N_\xi = \sqrt{\frac{x_\zeta^2 + y_\zeta^2 + z_\zeta^2}{x_\xi^2 + y_\xi^2 + z_\xi^2}}, \quad N_\eta = \sqrt{\frac{x_\zeta^2 + y_\zeta^2 + z_\zeta^2}{x_\eta^2 + y_\eta^2 + z_\eta^2}}, \tag{6.3}$$

and $R_\xi$, $R_\eta$ are the dissipation coefficients given by

$$R_\xi = S_l \tilde{d}^\xi_{j,k,l} a^\xi_{j,k,l}, \; R_\eta = S_l \tilde{d}^\eta_{j,k,l} a^\eta_{j,k,l}. \tag{6.4}$$

The dissipation coefficients consist of three functions that provide different ways to automatically adjust the local dissipation appropriately, depending on the local grid topology:

(1) A scaling function $S_l$, which varies with normal distance from the body surface.

(2) A grid point distribution sensor function, $\tilde{d}^\xi_{j,k,l}$ or $\tilde{d}^\eta_{j,k,l}$, depending on the direction, which senses mesh convergence based on the distances between neighboring grid points.

(3) A grid angle function, $a^\xi_{j,k,l}$ or $a^\eta_{j,k,l}$, depending on the direction, which senses mesh convergence based on the angles between neighboring grid points.

The form of the scaling function $S_l$ is given by

$$S_l = \begin{cases} \sqrt{\dfrac{(l-1)}{(l_{\max}-1)}} & 2 \leqslant l \leqslant l_{\text{trans}}, \\[2ex] \sqrt{\dfrac{(l_{\text{trans}}-1)}{(l_{\max}-1)}} & l_{\text{trans}}+1 \leqslant l \leqslant l_{\max}, \end{cases} \tag{6.5}$$

where $l_{\max}$ is the number of points in the $l$ direction and $l_{\text{trans}}$ is restricted to the range $[\frac{3}{4}, 1] \times l_{\max}$. With $d^\xi_{j,k,l}$ and $d^\eta_{j,k,l}$ defined by (6.8a, b), $l_{\text{trans}}$ is set to $l$ when one or both of the following is true:

$$\max_{j,k} d^\xi_{j,k,l} - \max_{j,k} d^\xi_{j,k,l-1} < 0, \tag{6.6a}$$

$$\max_{j,k} d^\eta_{j,k,l} - \max_{j,k} d^\eta_{j,k,l-1} < 0. \tag{6.6b}$$

Once $l_{\text{trans}}$ is located, the above tests are not performed for $l > l_{\text{trans}}$.

The purpose of the scaling function $S_l$ is to guarantee small dissipation, and hence, grid orthogonality near the body surface. As one moves away from the body surface, dissipation is increased since grid lines may begin to converge in concave regions and some loss of grid curvature in convex regions is no longer a problem. Away from the body surface, the grid point distribution sensor function and the grid angle function alone are sufficient to provide the appropriate amount of dissipation. Hence, the influence of $S_l$ is removed by setting it to a constant at some location $l_{\text{trans}}$ away from the body. It is found that a good criterion for locating $l_{\text{trans}}$ is when the convergence of local grid lines is slowing down in some sense given by (6.6a, b). For all the cases encountered so far, it is sufficient to test for $l_{\text{trans}}$ for $l \geqslant \frac{3}{4} l_{\max}$. For some cases, $l_{\text{trans}}$ may be less than $l_{\max}$, which reduces loss of orthogonality near the outer boundary; while for other cases, $l_{\text{trans}}$ may have to be equal to $l_{\max}$, when increasing values of dissipation are needed all the way to the outer boundary to prevent grid lines from converging.

The forms of the grid point distribution sensor functions $\tilde{d}^{\xi}_{j,k,l}$, $\tilde{d}^{\eta}_{j,k,l}$ are given by

$$\begin{aligned} \tilde{d}^{\xi}_{j,k,l} &= \max\left[\left(d^{\xi}_{j,k,l}\right)^{2/S_l}, 0.1\right], \\ \tilde{d}^{\eta}_{j,k,l} &= \max\left[\left(d^{\eta}_{j,k,l}\right)^{2/S_l}, 0.1\right], \end{aligned} \tag{6.7}$$

where

$$d^{\xi}_{j,k,l} = \frac{|\vec{r}_{j+1,k,l-1} - \vec{r}_{j,k,l-1}| + |\vec{r}_{j-1,k,l-1} - \vec{r}_{j,k,l-1}|}{|\vec{r}_{j+1,k,l} - \vec{r}_{j,k,l}| + |\vec{r}_{j-1,k,l} - \vec{r}_{j,k,l}|}, \tag{6.8a}$$

$$d^{\eta}_{j,k,l} = \frac{|\vec{r}_{j,k+1,l-1} - \vec{r}_{j,k,l-1}| + |\vec{r}_{j,k-1,l-1} - \vec{r}_{j,k,l-1}|}{|\vec{r}_{j,k+1,l} - \vec{r}_{j,k,l}| + |\vec{r}_{j,k-1,l} - \vec{r}_{j,k,l}|}. \tag{6.8b}$$

The distribution of grid points in the $\xi$ and $\eta$ directions are monitored by the functions $d^{\xi}_{j,k,l}$ and $d^{\eta}_{j,k,l}$, respectively. The quantity $d^{\xi}_{j,k,l}$ is the ratio of the distances between a grid point and its neighbors in the $\xi$ direction at level $(l-1)$ to that at level $l$. This ratio is high in concave regions and hence, more dissipation is provided here. It is of order one or smaller in flat or convex regions, where less dissipation is needed. Similarly, $d^{\eta}_{j,k,l}$ represents the corresponding quantity in the $\eta$ direction. The quantities $\tilde{d}^{\xi}_{j,k,l}$ and $\tilde{d}^{\eta}_{j,k,l}$ are constructed from $d^{\xi}_{j,k,l}$ and $d^{\eta}_{j,k,l}$, respectively, which are raised to the power $2/S_l$ in order to counteract the small value of $S_l$ close to the body surface. Also, the values of $\tilde{d}^{\xi}_{j,k,l}$ and $\tilde{d}^{\eta}_{j,k,l}$ are limited from becoming too low by a limiter of 0.1. Note that a grid point distribution sensor function based on cell area ratios is not as effective since the grid lines could be converging in one direction but not the other and the cell areas would not decrease very much.

The grid angle functions $a^{\xi}_{j,k,l}$, $a^{\eta}_{j,k,l}$ are more conveniently defined in terms of the following unit vectors. Let the vectors pointing in the plus and minus $\xi$ directions at grid point $(j, k, l)$ be represented by $\vec{r}_j^{\,+}$ and $\vec{r}_j^{\,-}$, respectively, where

$$\vec{r}_j^{\,+} = \vec{r}_{j+1,k,l} - \vec{r}_{j,k,l}, \quad \vec{r}_j^{\,-} = \vec{r}_{j-1,k,l} - \vec{r}_{j,k,l}, \tag{6.9}$$

and let $\hat{r}_j^+$ and $\hat{r}_j^-$ be the respective unit vectors for $\vec{r}_j^{\,+}$ and $\vec{r}_j^{\,-}$. Similar expressions are defined for $\vec{r}_k^{\,+}$, $\vec{r}_k^{\,-}$ and $\hat{r}_k^+$, $\hat{r}_k^-$ for vectors and unit vectors pointing in the plus and minus $\eta$ directions, respectively at grid point $(j, k, l)$. The local unit normal $\hat{n}_{j,k,l}$ based on the cross product of the above unit vectors is given by

$$\hat{n}_{j,k,l} = \frac{\left(\hat{r}_j^+ - \hat{r}_j^-\right) \times \left(\hat{r}_k^+ - \hat{r}_k^-\right)}{\left|\left(\hat{r}_j^+ - \hat{r}_j^-\right) \times \left(\hat{r}_k^+ - \hat{r}_k^-\right)\right|}. \tag{6.10}$$

The cosine of the local half angle $\alpha_{j,k,l}$ in the $\xi$ direction is then given by

$$\cos \alpha_{j,k,l} = \hat{n}_{j,k,l} \cdot \hat{r}_j^+ = \hat{n}_{j,k,l} \cdot \hat{r}_j^- . \tag{6.11}$$

The grid angle function $a^{\xi}_{j,k,l}$ is then defined as

$$a^{\xi}_{j,k,l} = \begin{cases} 1\big/\left(1 - \cos^2 \alpha_{j,k,l}\right) & \text{if } 0 \leqslant \alpha_{j,k,l} \leqslant \dfrac{\pi}{2}, \\ 1 & \text{if } \dfrac{\pi}{2} < \alpha_{j,k,l} \leqslant \pi. \end{cases} \tag{6.12}$$

A similar expression is defined for the grid angle function $a^{\eta}_{j,k,l}$ in terms of the cosine of the local half angle $\beta_{j,k,l}$ in the $\eta$ direction.

The smoothing provided by the grid point distribution functions in (6.7) has to be modified locally at grid points located at very sharp concave corners. These corners are detected by computing the half angles $\alpha$ and $\beta$ subtended by neighboring grid points in the $\xi$ and $\eta$ directions respectively (see (6.11)). At a severe concave corner point, extra dissipation is required to prevent crossing of grid lines. The functions $a^{\xi}_{j,k,l}$ and $a^{\eta}_{j,k,l}$ serve to provide the appropriate local modifications to the dissipation required at these corner points. For a typical mesh, the values of $a^{\xi}_{j,k,l}$ and $a^{\eta}_{j,k,l}$ are close to unity at most grid points. Figure 5 shows the grid marching out of a concave angle of 20 degrees. Note that orthogonality is still maintained for the first point off the surface. Grids for concave angles down to 5 degrees have been obtained with the above algorithm.

A minor local refinement at $l = 2$ of the above dissipation scheme for sharp corners is given in the Appendix.

## 7. METRIC CORRECTION

Often a body surface will have a corner or edge region that is a discontinuity to the hyperbolic grid equations. Nevertheless, the spatially-varying dissipation described in Section 6 will typically generate a sufficiently smooth grid at either convex or concave corners if the grid spacing to each side of the corner is approximately equal. If the surface grid possesses corners with uneven grid spacings, dissipation alone is usually not sufficient to provide a smooth volume grid and additional remedies are needed. The metric correction procedure described below generally provides a satisfactory treatment of corner discontinuities in all but extremely convex cases.

With $\zeta$-derivatives as defined in (3.5), the direction in which the grid will emanate from the corner is such that it is perpendicular to the line joining the two neighbor points of the corner. For a corner with unequally spaced points, this is clearly not a desirable feature (see Figure 6a). In order to guide the grid out in a direction that bisects the angles at a point subtended by its neighbors in both the $\xi$ and $\eta$ directions, the derivatives $x_\zeta$, $y_\zeta$, $z_\zeta$ at the point are modified to $x'_\zeta$, $y'_\zeta$, $z'_\zeta$ as follows:

$$\begin{pmatrix} x'_\zeta \\ y'_\zeta \\ z'_\zeta \end{pmatrix} = \frac{\Delta V}{Det(C')} \begin{pmatrix} y'_\xi z'_\eta - y'_\eta z'_\xi \\ x'_\eta z'_\xi - x'_\xi z'_\eta \\ x'_\xi y'_\eta - x'_\eta y'_\xi \end{pmatrix}, \tag{7.1}$$

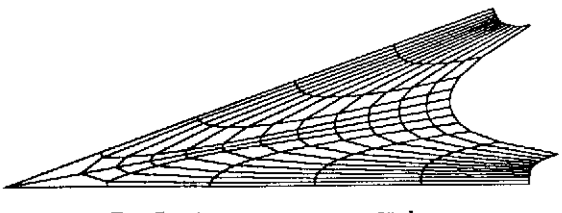

FIG. 5. A concave corner at 20 degrees.

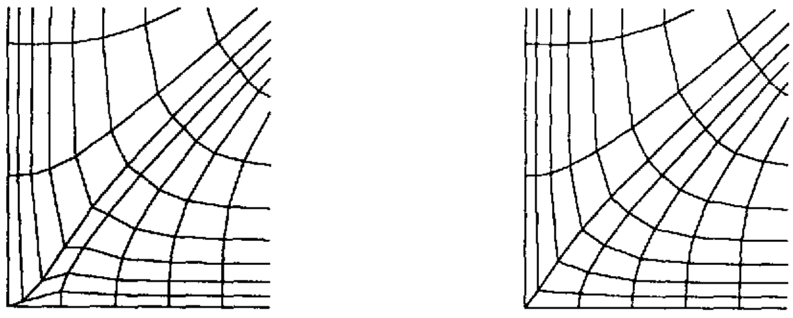

FIG. 6. Comparison of treatment of concave corner with unequal grid spacings. (a) Without metric correction, (b) with metric correction.

with

$$Det(C') = (y'_\xi z'_\eta - y'_\eta z'_\xi)^2 + (x'_\eta z'_\xi - x'_\xi z'_\eta)^2 + (x'_\xi y'_\eta - x'_\eta y'_\xi)^2,$$

where

$$(x'_\xi, y'_\xi, z'_\xi)^T = \frac{1}{4}\left(|\vec{r}_j^{\,+}| + |\vec{r}_j^{\,-}|\right)\left(\frac{\vec{r}_j^{\,+}}{|\vec{r}_j^{\,+}|} - \frac{\vec{r}_j^{\,-}}{|\vec{r}_j^{\,-}|}\right), \tag{7.2a}$$

and

$$(x'_\eta, y'_\eta, z'_\eta)^T = \frac{1}{4}\left(|\vec{r}_k^{\,+}| + |\vec{r}_k^{\,-}|\right)\left(\frac{\vec{r}_k^{\,+}}{|\vec{r}_k^{\,+}|} - \frac{\vec{r}_k^{\,-}}{|\vec{r}_k^{\,-}|}\right). \tag{7.2b}$$

While the $\zeta$-derivatives should be computed by (7.1) near the body surface, the original method of computing these quantities should be restored away from the body surface. This can be achieved smoothly by

$$(x_\zeta, y_\zeta, z_\zeta)^T = (1 - \nu_l)(x_\zeta^o, y_\zeta^o, z_\zeta^o)^T + \nu_l(x'_\zeta, y'_\zeta, z'_\zeta)^T. \tag{7.3}$$

where $\nu_l = 2^{2-l}$ and $x_\zeta^o$, $y_\zeta^o$, $z_\zeta^o$ are computed by (3.5). The form of (7.1) is the same as (3.5) except that the $\xi$- and $\eta$-derivatives are replaced by the corresponding primed quantities. These modified $\xi$- and $\eta$-derivatives are constructed in such a way that the neighboring points of the corner appear to be of equal distance from the corner. The result of applying this procedure to a concave corner is shown in Figure 6b.

## 8. TREATMENT OF CONVEX CORNERS

Two methods are presented here that provide extra robustness at convex corners. They both involve switching from solving the hyperbolic grid generation Equation (3.4) to some other equations at the convex corner point. The first method is an implicit averaging scheme. Instead of solving the hyperbolic equations at a convex corner, the following average equation is solved.

$$\Delta\vec{r}_{j,k} = \tfrac{1}{2}(\mu_\xi + \mu_\eta)\Delta\vec{r}_{j,k}, \tag{8.1}$$

where

$$\mu_\xi \Delta\vec{r}_{j,k} = \tfrac{1}{2}\left(\Delta\vec{r}_{j+1,k} + \Delta\vec{r}_{j-1,k}\right), \tag{8.2a}$$

$$\mu_\eta \Delta\vec{r}_{j,k} = \tfrac{1}{2}\left(\Delta\vec{r}_{j,k+1} + \Delta\vec{r}_{j,k-1}\right). \tag{8.2b}$$

In other words, the marching increment at the corner is the average of the marching increment of its four neighboring points. The form of the above scheme can be made compatible with the hyperbolic grid generation scheme given by (3.4) by approximate factorization. The equation to be solved at convex corner points is then

$$\left(I - \tfrac{1}{2}\mu_\xi\right)\left(I - \tfrac{1}{2}\mu_\eta\right)\Delta\vec{r} = 0. \tag{8.3}$$

At $l = 2$, the switch to solve (8.3) is performed if the external angle of a convex corner in *either* the $\xi$ or $\eta$ direction exceeds 240 degrees. As the grid marches out in $l$, the switch to return to the normal hyperbolic scheme is performed when the minimum of $\cos\alpha_{j,k}$ and $\cos\beta_{j,k}$ becomes larger than about $-0.2$. Although some factorization error is present, this implicit averaging scheme has worked well for a variety of convex corners, including some cases where the normal scheme has failed. One such case is illustrated by the NACA0012 airfoil shown in Figures 7a and 7b. The point of this example is *not* to show that an O-grid should be used for the airfoil but simply to demonstrate the ability of the scheme to produce high quality grids around sharp convex corners.

The implicit averaging scheme described above possesses two desirable properties. It can be shown geometrically that if the grid marches out

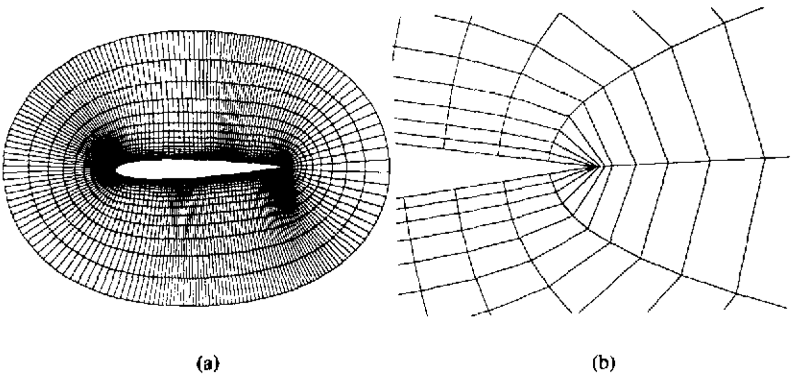

FIG. 7. Airfoil with sharp trailing edge and O-grid topology. (a) Far view, (b) close up view of sharp trailing edge region.

orthogonally at the neighbors of the convex corner, then the distance marched out at the corner point is always smaller than the distance marched out at the neighboring points. Moreover, the distance marched out at the corner point becomes smaller as the convex corner becomes sharper. This is very useful in helping to bend the neighboring grid lines towards the sharp corner as the grid is marched out away from the body surface (see Figure 7b). The second property is that the grid marching out from the corner point will bisect the angle at the corner provided the neighboring points march out the same distance and in symmetrical directions to each other.

An alternative but potentially more robust method than the implicit averaging scheme above is described below. The exact location of the grid point in the next marching step out from a convex corner is predicted in advance. The predicted point is located by marching the grid a distance of $\Delta\tilde{s}_{j,k,l}$ in the direction given by the angle-bisecting unit normal defined by $(x'_\zeta, y'_\zeta, z'_\zeta)^T$ in (7.1). The marching distance is scaled as

$$\Delta\tilde{s}_{j,k,l} = \Delta s_{j,k,l} \times \min(\sin \alpha_{j,k,l}, \sin \beta_{j,k,l}), \tag{8.4}$$

where $\Delta s_{j,k,l}$ is the user-specified arc length in the normal direction (see Section 5). The scaling with the sine of the half angle causes the corner point to march out a smaller distance than its neighbors, thus helping to bend the neighboring grid lines towards the corner. Thus, $\Delta\vec{r}$ at the convex corner point can be computed in advance and combined with the grid generation Equation (3.4). From numerical trials with different geometries, it was found that this procedure also works well for many types of convex corners.

## 9. GRID QUALITY CHECKS

Two grid quality checks are performed in the hyperbolic grid generation code. The first check is a cell volume check by decomposing each cell into six tetrahedra [7]. This test is only passed if the volume of each tetrahedron in the cell is positive. The second check involves computing the Jacobian of each cell using a finite volume scheme employed by typical flow solvers. Negative volumes or Jacobians are reported to the user and smoothing parameters can then be adjusted appropriately to remove the bad cells.

## 10. RESULTS AND APPLICATIONS

The vectorized version of the current hyperbolic grid generator runs at 142 megaflops on the CRAY-YMP and requires about 9.7 microseconds of CPU time per grid point (i.e., about a hundred thousand points per CPU second). As an example, generation of the largest grid in the Space Shuttle launch vehicle grid system, the Orbiter grid for flight Reynolds number ($98 \times 77 \times 57 = 430122$ points), takes 4.17 seconds of CPU time. This is about one to two orders of magnitude faster than typical elliptic grid generators.

The various external components of the integrated Space Shuttle vehicle contain a wide variety of different geometric features that are found in many other applications. Hence, these geometric components provide good tests of robustness for the hyperbolic grid generation scheme described above. Some examples are given below.

The following two examples show how the spatially-variable dissipation coefficient works at different types of corners. The first example shows the grid around a sequence of sharp convex and concave corners that appears at the IEA box on the attach ring of the solid rocket booster as in Figure 4. Low dissipation values are needed at the body surface and above the convex corners to maintain orthogonality while high dissipation values are needed in the concave regions to provide grid smoothness. Figure 3 shows the smoothness of the grid in a large concave region, the wing root region at the back of the Orbiter.

The robustness of the axis logic is demonstrated in the next example where the geometry is noncircular in the circumferential direction around the axis and that the distribution of grid points in this direction is nonuniform. This type of axis condition is present at the tip of the vertical tail section of the Orbiter as in Figure 8. The mixed extrapolation scheme with volume scaling is able to produce a smooth grid for this case.

When the chimera overset-grid method is used on two grids whose body surfaces intersect each other, the grid points in the region around the

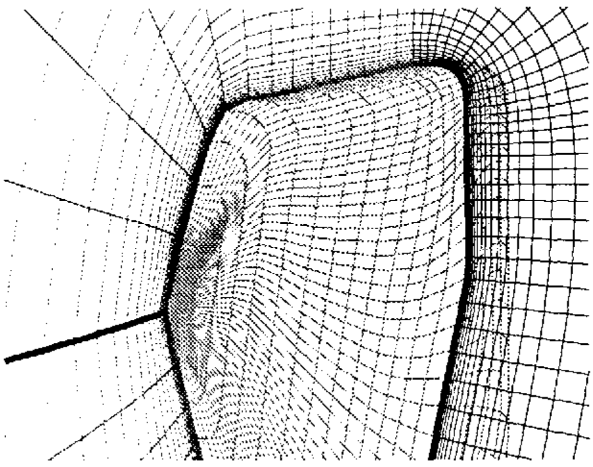

FIG. 8. Vertical tail grid showing part of the surface grid near the tip, the 3D grid on the symmetry plane and a section normal to the symmetry plane. The axis is marked by the thick dark line.

intersection line which are common to both grids are left with no interpolation stencils. In order to remedy this problem, a collar grid can be introduced which covers the region around the intersection line [11]. The most challenging example tested by the grid generator so far is the collar grid, which covers the intersection region between the vertical tail and the Orbiter as in Figures 9a, b. The methods used to generate the collar grid surface are explained in [9]. The surface of the collar grid is made up of two parts. The top part lies on

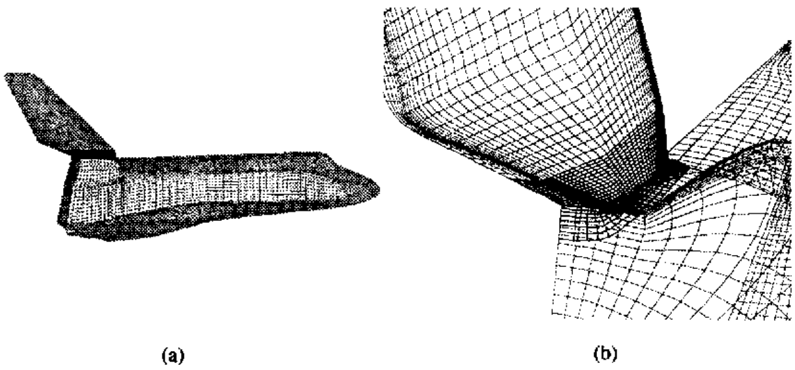

FIG. 9. Views of the collar grid surface. (a) Position relative to vertical tail and Orbiter, (b) close up view relative to vertical tail and Orbiter.

the surface of the tail down to the intersection line with the Orbiter. The lower part has two sections. The first section starts at the intersection line with the tail and then follows the top surface of the Orbiter. The second section folds over the back of the Orbiter and follows the backward-facing aft-bulkhead of the Orbiter. The difficult feature of this geometry is the presence of a region where grid lines are concave in one direction and convex in the other. Slices of the 3D grid viewed from the front and back ends of the collar grid are shown in Figures 10a, b.

The final example is taken from the telescope grid for the SOFIA vehicle [1]. The SOFIA is a modified Boeing 747 with a telescope mounted inside a cavity on the upper surface of the plane. The telescope is topologically similar to a hollow bowl with a truncated cylinder in the middle of the inside of the bowl. Figure 11a shows the surface geometry (shaded) for half the telescope and slices of the 3D grid. The external surface of the telescope consists of both the outside and inside of the bowl together with the middle cylinder. The symmetry plane of the 3D grid is shown in Figure 11b. These figures show that, although the surface grid may not possess the sufficient number of points to resolve the detailed flow structures, the grid generator is able to produce a smooth grid over the complex combination of concave and convex corners using the techniques described in Sections 6, 7 and 8. The outer boundary of the telescope grid need not be placed far away from the body surface since the entire telescope grid is surrounded by a larger cavity grid and communication between the two grids is achieved via the chimera overset-grid scheme.

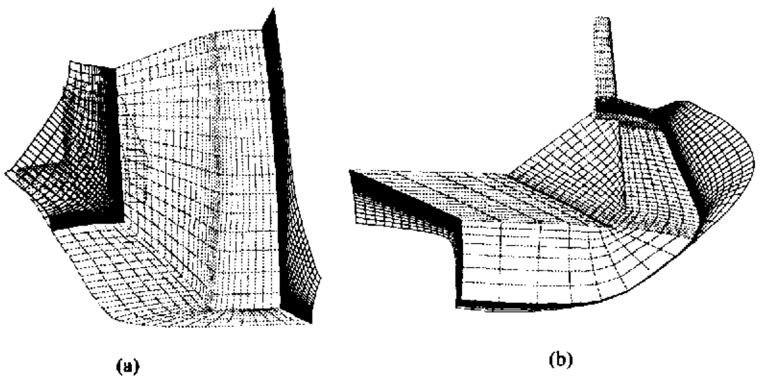

FIG. 10. Views of sections of the 3D collar grid joining the vertical tail and the Orbiter. (a) Front view, (b) back view.

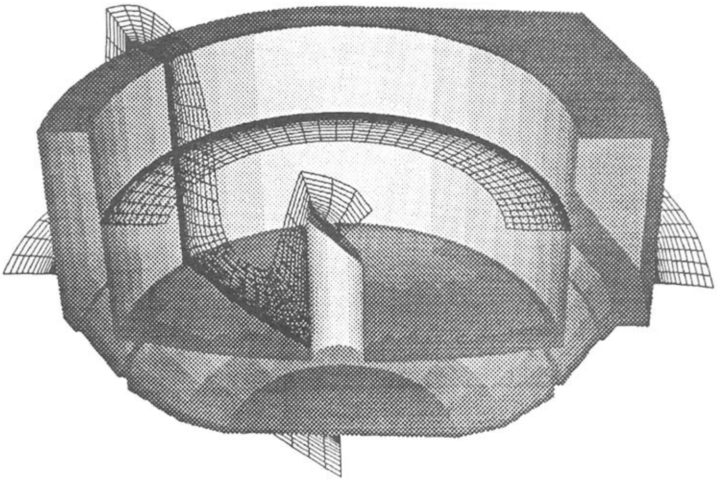

FIG. 11a. Surface geometry and slices of the 3D grid for the SOFIA telescope.

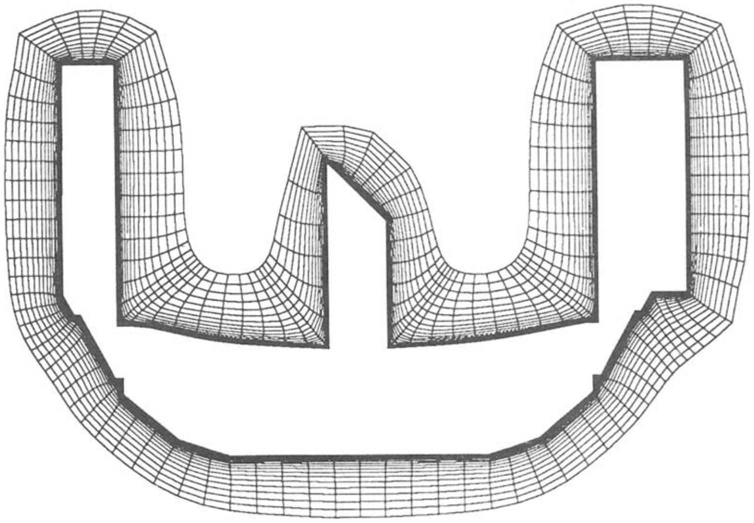

FIG. 11b. Symmetry plane of the 3D grid for the SOFIA telescope.

## 11. CONCLUSIONS

A robust three-dimensional hyperbolic grid generation scheme that is able to produce high quality grids for a wide variety of geometries has been presented. The improved robustness and the speed advantage of the scheme have made it extremely attractive for users of chimera flow-solvers. Since the scheme is fast, the user can readily adjust the input parameters to fine-tune the grid quality.

The use of a spatially-varying dissipation coefficient based on distances and angles between neighboring grid points gives the grid generator the ability to cope with geometries that are more complex than before. The grid angle bisecting property is provided by the metric correction procedure near the body surface. Sharp convex corners are automatically detected and the grid generation equations are altered at these corners to further enhance smoothness and robustness.

## APPENDIX

In order to guarantee orthogonality near the body surface, the dissipation coefficients $R_\xi$ and $R_\eta$ described in Section 6 are made to be zero everywhere at $l = 2$ through the scaling function $S_l$. However, near concave and convex corners, some dissipation has to be restored to maintain smoothness. This can be accomplished by introducing the blanking function $b_{j,k}$, which multiplies $R_\xi$ and $R_\eta$ at $l = 2$. The blanking function is zero everywhere except near concave ($\alpha_{j,k}$ or $\beta_{j,k} < \pi/3$) and convex ($\alpha_{j,k}$ or $\beta_{j,k} > 2\pi/3$) corners. For a concave or convex corner in the $\xi$-direction, we set

$$b_{j,k} = b_c, \quad b_{j\pm1,k} = 0.5, \quad b_{j\pm2,k} = 0.25, \tag{A.1}$$

while for a concave or convex corner in the $\eta$-direction, we set

$$b_{j,k} = b_c, \quad b_{j,k\pm1} = 0.5, \quad b_{j,k\pm2} = 0.25, \tag{A.2}$$

where $b_c = 1$ for a concave corner and $b_c = 0$ for a convex corner. At convex corners, dissipation at the neighboring points to the corner is still needed, but the dissipation at the corner point itself should be set to zero in order to produce the desired effects for the special schemes described in Section 8. In constructing $b_{j,k}$, it is assumed that successive corners in a coordinate direction are separated by at least four points. This is a reasonable assumption if one wishes to provide sufficient resolution for the flow around such corners.

## REFERENCES

1. C. A. Atwood and W. R. Van Dalsem, Flowfield Simulation about the SOFIA Airborne Observatory. AIAA Paper 92-0656, 1992.

2. P. G. Buning, I. T. Chiu, S. Obayashi, Y. M. Rizk, and J. L. Steger, Numerical Simulation of the Integrated Space Shuttle Vehicle in Ascent. AIAA-88-4359-CP, *Proceedings of the AIAA Atmospheric and Flight Mechanics Conference*, Minneapolis, Minnesota, 1988.
3. J. Q. Cordova and T. J. Barth, Grid Generation for General 2-D Regions using Hyperbolic Equations, American Institute of Aeronautics and Astronautics Paper 88-0520, 1988.
4. S. K. Godonov and G. P. Prokopov, The use of moving meshes in gasdynamical computations, *U.S.S.R. Comput. Math. and Math. Phys.* 12:182–195 (1972).
5. D. W. Kinsey and T. J. Barth, Description of a Hyperbolic Grid Generation Procedure for Arbitrary Two-Dimensional Bodies, AFWAL TM 84-191-FIMM, Wright Aeronautics Laboratory, Wright-Patterson AFB, Ohio, 1984.
6. G. H. Klopfer, Solution adaptive meshes with a hyperbolic grid generator, in *Proceedings of the Second International Conference on Numerical Grid Generation in Computational Fluid Dynamics*, Miami, Florida, 1988, pp. 443–453.
7. W. Kordulla and M. Vinokur, Efficient computation of volume in flow predictions, *AIAA J.* 21: 917–918 (1983).
8. S. Nakamura, Marching grid generation using parabolic partial differential equations, in *Numerical Grid Generation*, (J. F. Thompson, Ed.), North-Holland, New York, 1982, pp. 775–783.
9. S. J. Parks, P. G. Buning, J. L. Steger and W. M. Chan, Collar Grids for Intersecting Geometric Components Within the Chimera Overlapped Grid Scheme, AIAA Paper 91-1587, in *Proceedings of the AIAA 10th Computational Fluid Dynamics Conference*, Honolulu, Hawaii, 1991.
10. C. Starius, Constructing orthogonal curvilinear meshes by solving initial value problems, *Numer. Math.* 28:25–48 (1977).
11. J. L. Steger, Notes on Surface Grid Generation using Hyperbolic Partial Differential Equations, Internal Report TM CFD/UCD 89-101, Department of Mechanical, Aeronautical and Materials Engineering, University of California, Davis, 1989.
12. J. L. Steger and D. S. Chaussee, Generation of body-fitted coordinates using hyperbolic partial differential equations, *SIAM J. Sci.Statist. Comput.* 1:431–437 (1980).
13. J. L. Steger and Y. M. Rizk, Generation of Three-Dimensional Body-Fitted Coordinates Using Hyperbolic Partial Differential Equations, NASA TM 86753, NASA-Ames Research Center, Moffett Field, California, 1985.
14. J. F. Thompson, ed., *Numerical Grid Generation*, North-Holland, New York, 1982.
15. J. F. Thompson, F. C. Thames, and C. M. Mastin, Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies, *J. Comput. Phys.* 15:299–319 (1974).
16. A. M. Winslow, Numerical solution of the quasilinear poisson equation in a nonuniform triangle mesh, *J. Comput. Phys.* 2:149–172 (1967).