N92-24414

RECENT ENHANCEMENTS TO THE GRIDGEN STRUCTURED GRID GENERATION SYSTEM [†]

John P. Steinbrenner and John R. Chawner
MDA Engineering, Inc.
Arlington, TX

ABSTRACT

Significant enhancements are being implemented into the GRIDGEN 3D, multiple block, structured grid generation software. Automatic, point-to-point, interblock connectivity will be possible through the addition of the domain entity to GRIDBLOCK's block construction process. Also, the unification of GRIDGEN2D and GRIDBLOCK has begun with the addition of edge grid point distribution capability to GRIDBLOCK. The geometric accuracy of surface grids and the ease with which databases may be obtained is being improved by adding support for standard CAD file formats (e.g., PATRAN Neutral and IGES files). Finally, volume grid quality has been improved through addition of new SOR algorithm features and the new hybrid control function type to GRIDGEN3D.

LIST OF SYMBOLS

$\vec{r} = [\ x\ y\ z\ ]^T$	Cartesian coordinate vector
(ξ, η, ζ)	computational coordinates
(i, j, k)	computational indices
(Φ, Ψ, Ω)	control functions

INTRODUCTION

Grid generation is perhaps the most manhour intensive task in the process of applying computational fluid dynamics (CFD) methods to complex configurations. This conclusion is supported in the findings of two separate CFD committees: [1] [2]. Not surprisingly, CFD research in the past few years has produced several suites of grid generation software, each aimed at reducing the so-called grid generation bottleneck. This software follows a marked trend towards development of interactive, graphical tools [3] [4] [5]. The trend is quite logical if one considers the speed at which workstation technology is advancing and the fact that grid generation is a highly visual-oriented, geometry-based technology.

[†]Work done on subcontract to Computer Sciences Corp. (CSC-ATD-ER-92-B-117) on contract to NASA Langley Research Center (NAS1-19038)

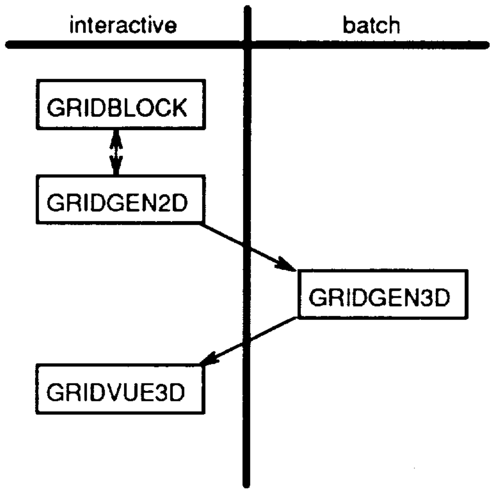

Figure 1: Schematic of the GRIDGEN process.

One particular example of a graphical, interactive software system developed for the generation of multiple block, structured grids is GRIDGEN [9]. The GRIDGEN system consists of four separate codes which accomplish distinct tasks of the grid generation process in the order indicated in Figure 1. The process begins with a user supplied database, which consists of any number of discrete $M \times N$ networks of points. The collection of networks is used by GRIDGEN only to define the 3D surface shape of the configuration. The first code, GRIDBLOCK (Figure 2), is used to decompose the domain around the database into a multiple block structure and to assign a computational coordinate system to each block. GRIDGEN2D (Figure 3), the second code, is then used to generate, using algebraic and elliptic partial differential equation (PDE) methods, grid points on the twelve edges and six faces of each block. The third code, GRIDGEN3D, uses algebraic and elliptic PDE methods to generate the grid points within each block. Finally, GRIDVUE3D is used for visual examination of the completed multiple block volume grid. GRIDBLOCK, GRIDGEN2D, and GRIDVUE3D are interactive graphics codes written specifically for the Silicon Graphics, Inc. IRIS workstations (they also run on IBM RS/6000 workstations with the GL Graphics software), while GRIDGEN3D is a batch code written for a Cray X/MP supercomputer.

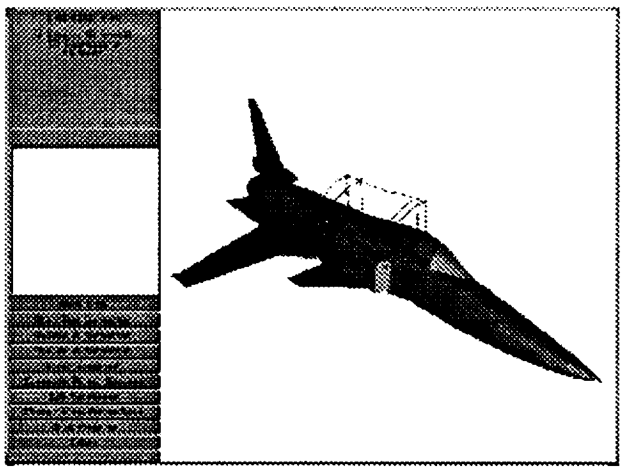

Figure 2: A typical GRIDBLOCK screen.

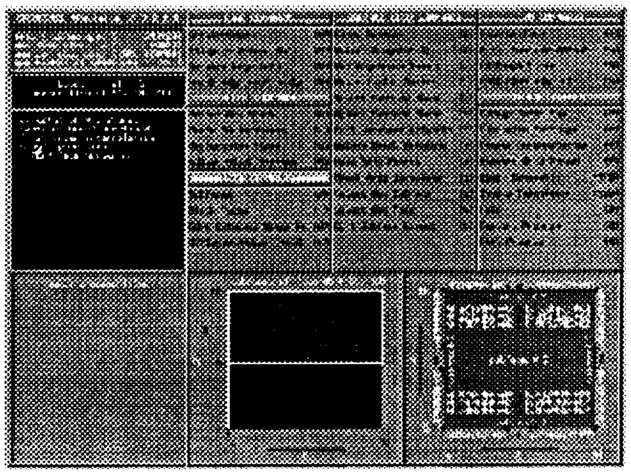

Figure 3: A typical GRIDGEN2D screen.

While the initial release of GRIDGEN (Version 6) has been very successful, there remains plenty of room for improvement. Therefore, several changes are being made to GRIDGEN that will eventually result in Version 8. These tasks may be organized in three categories: those designed to increase user efficiency, those designed to improve geometric accuracy, and those designed to improve grid quality. First, user efficiency has been improved through addition of edge grid point generation capability to GRIDBLOCK, creation of the domain entity in GRIDBLOCK, and addition of low-level automation to GRIDBLOCK and GRIDGEN2D. Second, surface grid geometric accuracy and the ease with which database files can be obtained is being improved through supporting the popular CAD file formats PATRAN Neutral File and IGES. Finally, grid quality has been improved through upgrades to the control function types and numerical algorithm used with the PDE solver in both GRIDGEN2D and GRIDGEN3D. The remaining sections of this paper describe the rationale behind and benefits of each of these tasks.

IMPROVED USER EFFICIENCY

One of the overall goals of the GRIDGEN software is to make the user a more efficient grid generator. This is attempted through the use of interactive computer graphics such that the user has immediate graphical feedback on each step of the generation process. One path toward improved efficiency is to consolidate GRIDBLOCK and GRIDGEN2D into a single code, thereby eliminating much user confusion. Part of this long term goal is accomplished within Version 8 by adding GRIDGEN2D's edge point distribution capability to GRIDBLOCK. Further improvements are made in the blocking process by adding the domain entity to GRIDBLOCK such that the software can determine all point-to-point interblock connections automatically. Finally, the users workload is reduced through the addition of tools to perform several grid generation functions automatically.

Edge Point Distribution in GRIDBLOCK

Once the database defining the shape of the configuration has been loaded into GRIDBLOCK and visually inspected, the user begins decomposing the domain by drawing the lines that will eventually make up the block edges. These 3D lines are known as connectors. Connectors may be made up of any number of segments. Segments are the primitive line types available and include (see Figure 4): straight lines, elliptical arcs, splined curves, and lines constrained to the database. The connectors may be drawn as though the user were sketching them with pencil and paper; no regard need be given to the order or direction that they are drawn, nor what block they'll be in. The example in Figure 4 shows the connectors that define one block of a system.

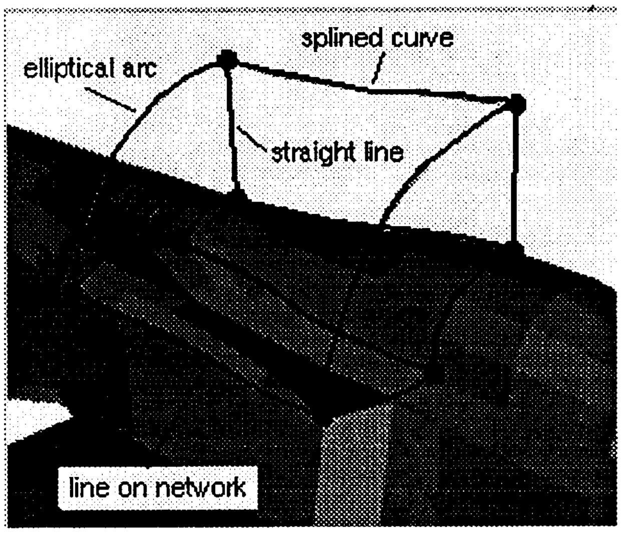

Figure 4: Segment types in GRIDBLOCK.

Block construction in GRIDBLOCK Version 6 proceeded by grouping a number of connectors into a block, and then by specifying orientation of the computational coordinate system within the block. Computational dimensions (i.e., number of grid points in each computational direction) would then be assigned, and finally interblock connections and flow boundary conditions (BC's) would be explicitly set by the user. Generation of actual grid points began within GRIDGEN2D, first by distribution of points in 1D along block edges and then in 2D on block faces.

The separation of the tasks of choosing the number of points in a block (in GRIDBLOCK) and distributing the grid points on edges (in GRIDGEN2D) could sometimes cause problems for the user. Only upon distributing points along edges in GRIDGEN2D did it become apparent whether the number of points chosen in GRIDBLOCK was appropriate. In cases where the user decided that too few or too many points had been specified he or she would exit GRIDGEN2D, re-run GRIDBLOCK, and change the block size. Unfortunately, changing the block size in GRIDBLOCK also required the user to manually re-set all interblock connections and BC's. This would affect most every other block in the system resulting in a lot of re-work.

The solution to this problem was to add the capability to distribute grid points on connectors to GRIDBLOCK Version 8. This allows the user to immediately determine whether or not sufficient points have been chosen for the distribution function, thereby eliminating the need for tedious rework. Part of this new capability consists of adding new segment types to GRIDBLOCK: a surface cubic segment, a piecewise cubic polynomial spline with Bessel interpolants constrained to the database; and a user defined segment read from a formatted ASCII file.

Connectors are assigned a computational dimension via type-in or by copying the dimension of another connector. Grid point distribution is then controlled interactively using tools that combine the functionality of the edge subdivision and grid point distribution menus of GRIDGEN2D Version 6. This combined interface provides a more intuitive and considerably streamlined way of distributing grid points on the connector. GRIDGEN2D's distribution functions (2-sided Vinokur, 1-sided hyperbolic sine and hyperbolic tangent, 1-sided geometric, and equal spacing) have been added to GRIDBLOCK. One new distribution function is based on Monotonic Rational Quadratic Splines (MRQS) [6], whereby a grid point may be placed at a specific location with the stipulation that the grid point distribution vary smoothly across it.

In Version 8, connectors may be redimensioned at any time with minimal effort, and all features of the previous grid point distribution will automatically be applied to the new grid points (Figure 5). Relative grid point distributions are also preserved after the shape of the connector is modified (Figure 6). This is a significant improvement over the current process in GRIDGEN2D Version 6, where the grid points must be redistributed manually if the edge shape is changed and where the edge shape must be redefined if the distribution of points is to be changed.

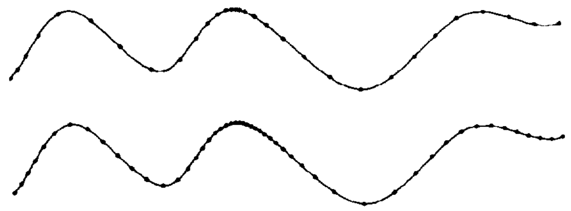

Figure 5: In GRIDBLOCK Version 8 the relative grid point clustering is maintained after the number of points is changed.

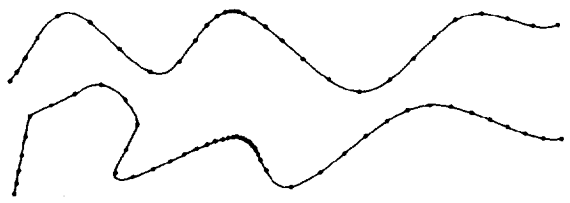

Figure 6: In GRIDBLOCK Version 8 the relative grid point clustering is maintained after the connector shape is modified.

An additional feature of GRIDBLOCK Version 8 is that database constrained segment types are maintained in parametric coordinates. This means that when the segment is edited in GRIDBLOCK the points will remain on the database, whereas in Version 6 they could be moved off the surface. This feature also implies that edge shapes will not have to be re-drawn in GRIDGEN2D in order to use the database parametric elliptic PDE solver.

Development of the Domain Entity

The second major addition to GRIDBLOCK Version 8 is the development of a new methodology for block construction. The impetus for this methodology comes largely from the fact that

GRIDGEN allows non-full face interblock connections and flow boundary conditions, a general boundary condition type supported by a number of flow solvers [7],[8]. Earlier it was mentioned that GRIDBLOCK Version 6 defined blocks by the connectors that formed the twelve physical edges of the block. No geometrical information pertaining to regions interior to a block's face (such as the horizontal connector between blocks B and C that also lies in the face of Block A in Figure 7) was associated with that block. Hence, it was impossible to determine inter-block connections automatically. The user was responsible for setting all connections manually by typing in block numbers, face numbers, and index ranges.[‡] This error-prone procedure was the most confusing step in GRIDBLOCK Version 6. Further, the fact that dimensions were assigned independently on the block level allowed inconsistent block dimensions to go undetected until the user attempted to set an impossible connection such as the one in Figure 8. Problems of this sort are being eliminated in GRIDBLOCK Version 8 with the introduction of a new entity, called a domain, which fits between connectors and blocks.

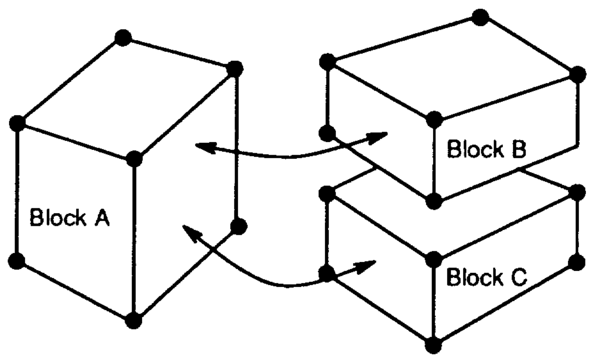

Figure 7: In GRIDBLOCK Version 6 blocks were represented by connectors and nodes only.

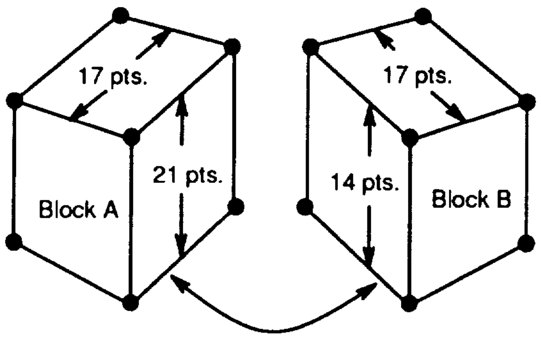

Figure 8: In GRIDBLOCK Version 6 inconsistencies in dimensioning could lead to impossible connections.

A domain is a 2D region on the face of a block. It is analogous to a subface in GRIDGEN2D terminology. A domain is created by defining a closed loop of connectors that outlines a region of a face of a block. This may either be a region of a particular BC, a surface on which two blocks connect, or a subset or superset of either. The sole requirement is that the domain represent a

[‡]Note that if GRIDGEN were limited to full face to face connections, the necessary connections could be determined automatically.

computationally rectangular region of the computational domain. A single surface represented by numerous domains is shown in Figure 9.

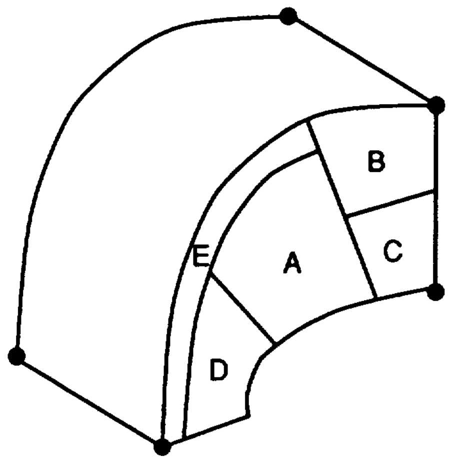

Figure 9: In GRIDBLOCK Version 8 a single block face may be represented by several domains.

Dimensional inconsistencies are now caught by GRIDBLOCK Version 8 during domain construction rather than in connection specification. This is possible since connectors have a number of grid points associated with them. When a domain is completed, it is compared to all other domains to insure uniqueness. A check is also made to see if two adjacent edges in the domain overlap with any other domain. If so, the user may have inadvertently defined the same physical surface by two differing domains, a mistake likely to cause problems for the remainder of the process (see Figure 10). Alternately, two domains with coincident adjacent edges may indeed represent different surfaces, as illustrated in Figure 11. The wing in this figure with a cusped trailing edge has domain B defined on the upper wing and domain A defined on the lower wing. The trailing edge connectors are coincident, as are the short connectors adjacent to the trailing edge at the wing root and tip. A blind description of the two domains might suggest that they lie on the same surface when in fact they do not. GRIDBLOCK will warn the user to exercise care under such circumstances. User generation of unique domains is expected to be a source of confusion for the novice user, and so additional tools are being added to GRIDBLOCK to ease confusion. For example, domain splitting and concatenation routines are included.

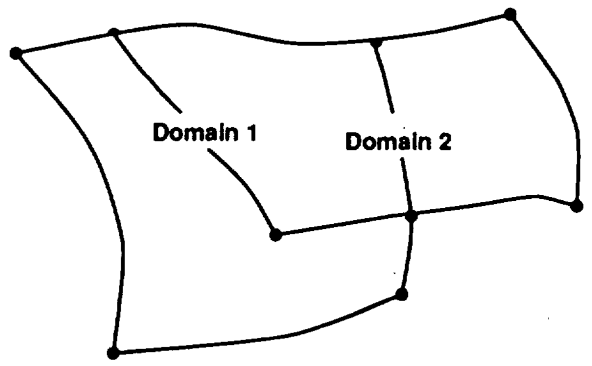

Figure 10: An example of overlapping domains.

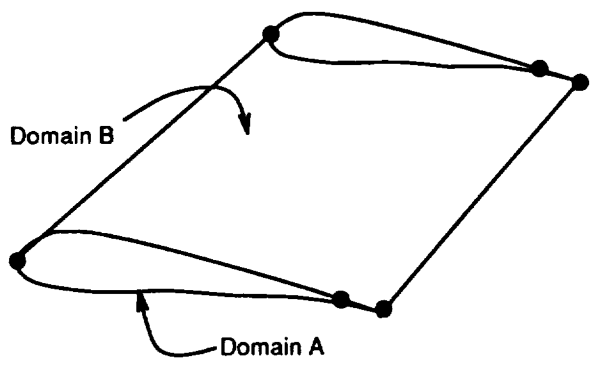

Figure 11: An example of non-overlapping domains with coincident edges.

In GRIDBLOCK Version 8, a block is constructed by interactively grouping domains into the six faces of the block. A face may consist of a single domain. However, the more interesting case is when the face is comprised of several domains (see Figure 9). Here, the user builds the face by systematically selecting the component domains that make up the face. The algorithm developed for face construction is order dependent, and as such allows faces to be constructed from an ambiguous set of domains in an unambiguous way. For example, the C-type face in Figure 12 representing a wing and wake region by four domains may be defined unambiguously by connecting the domains in the order A, B, C and A again. Without any order, it would be impossible to ascertain whether the domains connect at the leading edge, the wing trailing edge, the downstream connector, or at the inboard or outboard connectors on domain A.

The face construction algorithm developed for GRIDBLOCK will detect errors in face dimensioning as well as connections which result in a non-rectangular face. When the final face is assigned to a given block, dimensional and geometrical consistency is automatically checked. Block construction is then completed by assigning computational directions ξ, η, ζ at a corner of the block. This too is done graphically, in a manner similar to GRIDBLOCK Version 6. Since a block's computational orientation is the final step in block construction, later reorientation of a block is trivial. Hence, what formerly required a complete respecification of the block shape, dimensions, and interblock connections and BC's may now be achieved by a single, simple command.

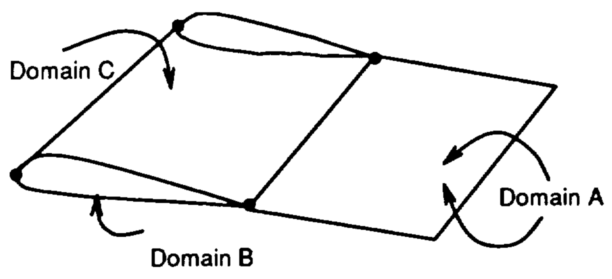

Figure 12: A face represented by the four domains A,B,C,A.

If the blocks have been defined without domain overlap, each domain will be assigned to a block exactly zero, one or two times.[§] If a domain is not assigned to any block, it is a superfluous domain

[§]A domain defined more than twice signifies an impossible blocking structure, usually detectable by the user.

and is not part of the developing blocking system. A domain assigned once in the block structure represents an external boundary of the composite block structure. Domains of this type normally correspond to block surface regions on which BC's would be set, such as solid surfaces, planes of symmetry, farfield conditions, etc. The user may set these BC's by picking the domain from the screen and the BC type from a flow solver-specific menu similar to GRIDBLOCK Version 6. Finally, domains assigned twice in the structure represent interblock connections. Since such domains exist in two places, the GRIDBLOCK Version 8 will automatically be able to determine all regions of point to point connections in the multiple block structure. This is a significant improvement over GRIDBLOCK Version 6, which as described above forced manual establishment of all connections. Hence, this new capability will eliminate both the time needed to set connections and the error associated with it. The instances where doubly defined domains in a blocking arrangement are not intended to define a flow-through condition (such as a flowfield obstruction with zero thickness) are easily handled by setting a BC on each occurrence of the domain, thereby circumventing the need for GRIDBLOCK to find a connection at that domain.

Automation Tools

Introduction of the domain entity and edge point distribution in GRIDBLOCK Version 8 also has a beneficial side effect that falls into the category of automation. Since domains (essentially subfaces) have been defined in terms of four edges, and since grid points have been distributed on the connectors that make up the edges it will be possible for GRIDGEN2D Version 8 to automatically initialize surface grid points within each domain using algebraic techniques. This is true even in cases where the connectors have been defined in terms of the database and the domain is intended to maintain the database shape since the parametric coordinates are now stored with the connector. Hence, the role of GRIDGEN2D Version 8 will be one of a surface grid refinement tool rather than a surface grid generation tool.

The changes to GRIDBLOCK Version 8 will also allow a user's change in the number of points on a connector to be propagated throughout the entire multiple block structure semi-automatically. In this utility, the user would select a connector, enter the new dimension, and all affected domains, faces and blocks would be updated semi-automatically to maintain dimensional consistency. New grid point distributions would be calculated automatically, and a new surface grid system reflecting the new blocking dimensions could be initialized in C RIDGEN2D. In this way, a low-level change to the blocking system would be enforced on the entire grid in a nearly automatic fashion.

The term "semi-automatically" used above to describe dimensional updating indicates what might happen in a complicated blocking structure. If a face is defined by a single domain with single connectors on each of the four edges, it is easy to see how changes in the number of points on one connector (edge) can be propagated to the opposite face edge. However, Figure 13 depicts a block face defined by domains numbered 1-6. In this example, if the dimension of connector A is changed, the connectors on the opposite side of domain 1 (connectors B, C and D) must be redimensioned such that the sum of their dimensions equals the new dimension of connector A. This in turn requires a redimensioning of connectors E, J, K, F, G and H. Clearly then, the redimensioning of a single connector can throw the dimensional consistency of an entire multiple block system out of balance. Notice that there exists no pre-definable manner in which the connectors in Figure 13 should be redimensioned to maintain consistency. In GRIDBLOCK

Version 8 the user will be led through the areas of ambiguity, being asked to make decisions along the way, until the entire block system is resized correctly.

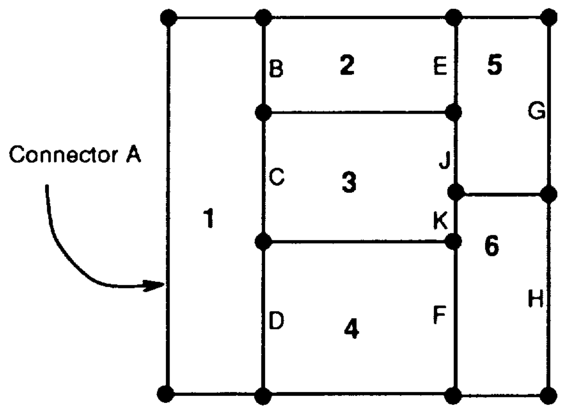

Figure 13: An example of an ambiguity in redimensioning a connector.

User Customization of GRIDGEN2D

Two tools were added to GRIDGEN2D to decrease the amount of user input required to generate surface grids: a verbosity setting and preferencing.

Verbosity setting refers to the ability to decrease the number of text prompts and menu options that the user of GRIDGEN2D Version 6 currently has to deal with. By reducing this, novice users will find GRIDGEN2D Version 8 easier to use because they won't be inundated with queries for lesser used options that may obscure the grid generation tasks. Also, experienced users will be able to quickly move through certain commands with fewer keystrokes. The verbosity setting may be either normal or terse. GRIDGEN2D with normal verbosity is the default setting, and it appears to the user as it always has. In terse mode, GRIDGEN2D hides certain menu buttons from the user (e.g., the button for Change Spline Fit from the edge shape definition menu) and eliminates seldom used options from text prompts. An example of the latter is the Algebraic Solver main menu command which is shown here in normal verbosity:

```
Interpolate      Method
-------------------------------------------------------
x,y,z(i,j)       1. arclength based (Soni) TFI (default)
                 2. linear (original) TFI
                 3. polar coordinate TFI
                 4. TFI with boundary orthogonality
                 5. TFI  stretching along i = const. lines
                 6. TFI  stretching along j = const. lines
                 7. TANH stretching along i = const. lines
                 8. TANH stretching along j = const. lines
x,y,z(u,v)       9. fit grid to parametric surface shape
u,v(i,j)        10. interp. u,v and fit to par. surface
x,y,z(u,v)
```

```
---------------------------------------------------------
Select interpolation scheme 1-9. (default = 1)
```

and here in terse verbosity:

```
Interpolate      Method
---------------------------------------------------------
x,y,z(i,j)     1. arclength based (Soni) TFI (default)
               3. polar coordinate TFI
---------------------------------------------------------
Select interpolation scheme 1,3. (default = 1)
```

Preferencing refers to the ability of an experienced GRIDGEN user to specify ahead of time the types of certain grid generation options to be used such that the related text prompts or menu buttons don't appear when the code is run. For example, it is possible to set arclength based TFI as a preference such that when the Algebraic Solver main menu command is invoked, no prompts for the specific method appear (as shown above) and arclength based TFI is run automatically. Preferences may be set for various edge generation options (edge shape, distribution function, etc.), algebraic solver (method type), and elliptic PDE solver (control function, relaxation factor, etc.).

Using both terse verbosity and preferencing, it is now possible to start the elliptic PDE solver running simply by invoking the main menu command, rather than having to answer the dozen text prompts displayed by Version 6. It is also possible to have grid points distributed on an edge in the same automated fashion.

The verbosity setting, preferencing and several other customizable features can be set in any of three ways: interactively, through the new Customization main menu command; through environment variables at runtime; or through a run commands file called .ggrc that contains a namelist of the various options.

IMPROVED GEOMETRIC INTERFACING

It is well understood in the CFD community that the transition from geometric definition of a configuration (on a CAD system) to a computational grid is one of the major hurdles in the overall CFD process [2]. Two approaches for smoothing this transition immediately come to mind. On one hand, grid generation methods could be added as modules to existing CAD packages [4]. On the other hand, CAD-like functionality could be added to existing grid generation software. An advantage to the latter approach is that grid generation users wouldn't have to learn to operate the CAD system, which, as technology stands today, is much more complex than the grid generator. This latter approach is the one adopted for the GRIDGEN system.

GRIDGEN Version 6 required geometry models ordered in a discrete point array form, so that a connected rendering of the model (database, in GRIDGEN terminology) would appear as a wireframe model. A GRIDGEN database for an F-15 SMTD aircraft is displayed in Figure 14.

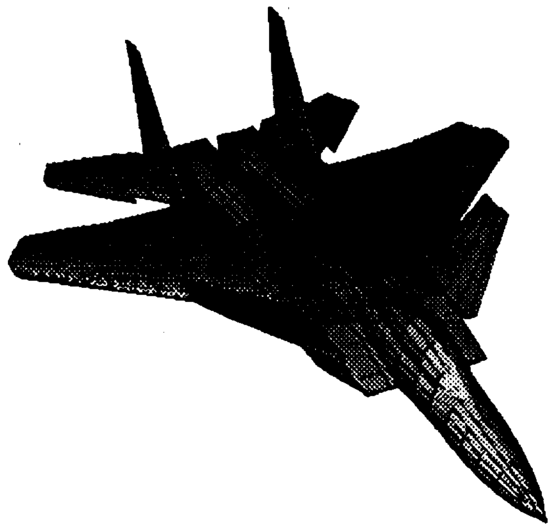

Figure 14: A discrete database for the F-15 fighter.

Unfortunately, a number of limitations are inherent in a discrete database. First, a discrete approximation to a mathematically continuous model is often a poor representation of the geometry, particularly when the database is either sparsely discretized or if the model has regions of large surface curvature. GRIDGEN places grid points on the faceted model of a continuous geometry. Secondly, a discrete database is not readily output by standard CAD systems, since most use a mathematical surface form more sophisticated than simple point data. This often forced the CAD operator to generate a database by piecing together a collection of discretized cross sectional cuts, a tedious and error-prone process.

GRIDGEN Version 8 will support entities written in the PATRAN Neutral File format. PATRAN [10] is a general engineering software system originally designed for solid mechanics applications, and the neutral file is a format intended for access by other computer hardware and software. The entities to be supported in this format are cubic curves for edge shape definition, and bi-cubic patches and trimmed surface models for surface shape definition.

GRIDGEN Version 8 will also support entities written in the Initial Graphics Exchange Specification [11] (IGES) format supported by the National Institute of Standards and Technology (formerly the National Bureau of Standards). This format, supported by most CAD packages, will provide a direct link between the CAD software and the GRIDGEN codes. Seven IGES entity types are immediately useful in GRIDGEN, including the parametric curves and surfaces, the rational B-spline curves and surfaces, and trimmed surface entities. Implementation of the rational B-spline entities will be done through extensive use of the DT_NURBS library [12], a library of Fortran subroutine B-spline utilities developed for the Navy's David Taylor Research Center.

Implementation of the improved geometric interface will begin with GRIDBLOCK and GRIDGEN2D extensions to read and interpret surface and curve geometries in either IGES or PATRAN formats. From there several new connector segment shapes will be added to GRIDBLOCK, including a free curve segment based on B-splines, a conic section segment based on rational B-splines (which will complement the current ellipse segment), and surface-constrained curve segments allowing general curves to be drawn directly on the IGES and/or PATRAN surfaces. In addition, an interface will be developed which uses the DT_NURBS library for calculating intersection curves on two NURBS surfaces.

The two main surface grid generation algorithms in GRIDGEN2D will also require modification to accept the IGES and PATRAN formats. In the conventional surface solver, only two of the three Cartesian coordinates of surface grid points are calculated via a surface form of Poisson's equation, and the third is calculated by interpolating from the geometric surface. If the defining surface is in IGES or PATRAN form (rather than the GRIDGEN database format), ray tracing routines from the DT_NURBS library will be accessed to interpolate the third physical coordinate from the surface. In the parametric surface solver, surface grid points are calculated as a solution to Poisson's equation transformed into the surface's intrinsic (parametric) coordinates. The resulting parametric coordinate solution is then transformed back to physical coordinates via the geometry data. Hence, GRIDGEN2D's parametric solvers will be upgraded to read both IGES and PATRAN data as the parametric surface representation.

ALGORITHMIC IMPROVEMENTS

This section describes enhancements to the volume grid generation software, GRIDGEN3D, that are primarily designed to improve the quality of the resulting grid.

Control Functions

GRIDGEN3D is a batch code that takes as input the surface grids generated by GRIDGEN2D (see Figure 1) and creates the grid on the interior of each block in the system using algebraic and PDE methods. Determining the quality of the resulting grid is a nebulous task at best. However, the attributes of smoothness, clustering, and orthogonality are qualitative measures that are often cited as desirable. In GRIDGEN3D, these qualities are obtained by the PDE methods through the use of several control function formulations as shown in Figure 15 [13],[14]. The reader is referred to Reference [9] for a more detailed explanation of this material. It is unfortunate, however, that each of these control functions concentrates on a single quality measure. A new control function type was developed for use in GRIDGEN3D that is a hybrid of the types described above. This hybrid control function is also available in GRIDGEN2D.

The concept of the hybrid control functions involves applying two control functions simultaneously. One, the background control function, tends to affect the grid near the block interior. The other, the foreground control function, tends to affect the grid near the block surfaces. As applied in GRIDGEN3D, the background control function may be either LaPlace, Thomas & Middlecoff, or Fixed Grid (see Figure 15). The foreground control function is always Sorenson.

By way of example, consider the Thomas & Middlecoff background control functions. The goal of these control functions is to cluster grid points on the interior of the block based on the clustering on the six faces. They are derived based on the assumption that grid lines transverse to a face are locally straight and intersect the face orthogonally. The calculation proceeds by computing control function components on each face as follows:

control function	effect on grid
LaPlace	smooth variation of cell volume, tending toward evenly distributed points, throughout the interior
Thomas & Middlecoff	clustering on block interior is based on clustering on faces
Fixed Grid	kinks in the original grid are removed
Sorenson	orthogonality of grid lines with faces, tight clustering at faces, LaPlace effect on interior

Figure 15: Effect of control functions on the volume grid.

$$\Phi_b = \frac{\vec{r}_\xi \cdot \vec{r}_{\xi\xi}}{\vec{r}_\xi \cdot \vec{r}_\xi} \quad \text{on j = 1, j = J, k = 1, k = K} \tag{1}$$

$$\Psi_b = \frac{\vec{r}_\eta \cdot \vec{r}_{\eta\eta}}{\vec{r}_\eta \cdot \vec{r}_\eta} \quad \text{on k = 1, k = K, i = 1, i = I} \tag{2}$$

$$\Omega_b = \frac{\vec{r}_\zeta \cdot \vec{r}_{\zeta\zeta}}{\vec{r}_\zeta \cdot \vec{r}_\zeta} \quad \text{on i = 1, i = I, j = 1, j = J} \tag{3}$$

Two-dimensional TFI is then used to interpolate Φ_b onto each i constant plane, Ψ_b onto each j constant plane, and Ω_b onto each k constant plane. By themselves, the Thomas & Middlecoff control functions are very robust and result in good grids. A plane from a volume grid computed using Thomas & Middlecoff control functions is shown in Figure 17b. Note, however, that the assumption of orthogonality at the faces used in the derivation, does not result in practice.

On the other hand, Sorenson control functions (used in the foreground) are computed to enforce a specific clustering and transverse angle at each face. Given any grid, one can compute the control functions at each point based on the computational derivatives of the Cartesian coordinates. Sorenson's method uses this relationship to compute the control functions on each of the six faces but only after the computational derivatives transverse to the face have been altered to enforce orthogonality and clustering. Consider computation of Φ_f on the $\zeta = \zeta_{min}$ face. At each point on the face one can compute the derivatives $\vec{r}_\xi$, $\vec{r}_{\xi\xi}$, $\vec{r}_\eta$, $\vec{r}_{\eta\eta}$, and $\vec{r}_{\xi\eta}$ using finite differences and the known grid point coordinates on the face. The derivative transverse to the face $\vec{r}_\zeta$ is not known. Once it is, however, it will be possible to compute $\vec{r}_{\zeta\zeta}$ (from $\vec{r}_\zeta$ and the transient volume grid) and $\vec{r}_{\eta\zeta}$ and $\vec{r}_{\zeta\xi}$ (by differencing $\vec{r}_\zeta$). The problem is then reduced to determining appropriate values for $\vec{r}_\zeta$ to enforce orthogonality.

$\vec{r}_\zeta$ is computed at each point on the face based on user-specified angle and spacing constraints, which in turn are calculated from the known angle and spacing data on the four edges of the face. Referring to Figure 16, the grid point's coordinates on adjacent faces allow us to compute the arclength spacing Δs, and the two intersection angles θ and δ on four edges using

$$\Delta s = \mid \vec{r}_\zeta \mid \tag{4}$$

$$\cos\theta = \frac{\vec{r}_\zeta \cdot \vec{r}_\xi}{|\vec{r}_\zeta||\vec{r}_\xi|} \tag{5}$$

$$\cos\delta = \frac{\vec{r}_\zeta \cdot \vec{r}_\eta}{|\vec{r}_\zeta||\vec{r}_\eta|} \tag{6}$$

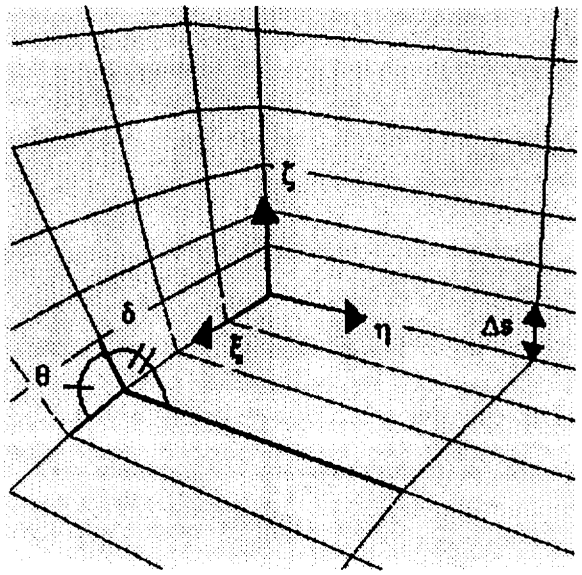

Figure 16: Computation of angle and spacing constraints for the Sorenson control functions.

These constraints are then interpolated from the four edges onto the interior of the face using TFI. The spacing Δs is interpolated using computational based LaGrange blending functions and the angles θ and δ are interpolated using computational based exponential blending functions that blend the edge angles to $\frac{\pi}{2}$ on the face interior. Then, the angle and spacing constraints are used to compute $\vec{r}_\eta$ on the face interior followed by computing Φ_f on the face.

In the standard Sorenson scheme, the control functions are computed on each of the six faces and then interpolated onto the block interior using TFI with computational based exponential blending functions that decay the control functions to zero on the block interior. An example of this method is shown in Figure 17c. One can see that the grid is indeed clustered and orthogonal to the faces yet points in the interior are smoothly distributed, where the governing PDE is locally reduced from Poisson's to LaPlace's equation.

The best features of the Sorenson and Thomas and Middlecoff control functions are maintained with the hybrid control functions calculated as follows. The background control functions Φ_b, Ψ_b, and Ω_b are computed throughout the entire block and the foreground control functions Φ_f, Ψ_f, and Ω_f are computed on only the six block faces. First, the difference of background and foreground control functions are formed on the six faces as

$$\begin{aligned} \Phi_d &= \Phi_f - \Phi_b \\ \Psi_d &= \Psi_f - \Psi_b \\ \Omega_d &= \Omega_f - \Omega_b \end{aligned} \tag{7}$$

Next, the control function differences are distributed from the faces to the block interior using TFI with exponentially decaying blending functions such that the differences vanish far into the block interior. Finally, hybrid control functions are computed by adding the background functions to the difference functions. The resulting functions will then have $\Phi \rightarrow \Phi_f$ near the block faces, and $\Phi \rightarrow \Phi_b$ near the block interior, with equivalent relations for the Ψ and Ω control functions. Thus, the hybrid control functions locally assume the form of the background or foreground control functions in the regions where the particular control functions are best suited.

$$\begin{aligned} \Phi &= \Phi_d + \Phi_b \\ \Psi &= \Psi_d + \Psi_b \\ \Omega &= \Omega_d + \Omega_b \end{aligned} \tag{8}$$

An example of the hybrid combination of Thomas & Middlecoff and Sorenson control functions is shown in Figure 17d. One can see that not only are the angle and spacing constraints enforced at the faces but the grid clustering is carried through the block interior.

Numerical Method

Solution of Poisson's Equation in GRIDGEN3D is performed using an explicit, pointwise SOR algorithm. The robustness and efficiency of this algorithm was improved by the addition of the following features.[¶]

1. Variable sweep direction. Each iteration begins the SOR sweep through the i,j,k indices in a different corner of the block to prevent biasing of the grid in any one computational direction.
2. One sided differencing. Forward or backward differencing of the coordinate derivatives based on the sign of the control function improves numerical stability by making the system of equations more diagonally dominant.
3. Grid sequencing. In grid sequencing, the PDEs are first solved on a coarse grid consisting of a sparse subset of the full grid. The solution on the coarse grid converges more rapidly in the PDE solver. At some point the coarse grid solution is stopped and the grid point coordinates and control functions on the full grid are interpolated from the coarse grid and the PDE solution proceeds on the full grid. The net computer time required to converge the grid is drastically reduced.

CONCLUSION

The improvements discussed in this paper are expected to move the GRIDGEN system to a new level of user efficiency and geometric and computational accuracy. Initial release of the new

[¶]These three features are also available in GRIDGEN2D.

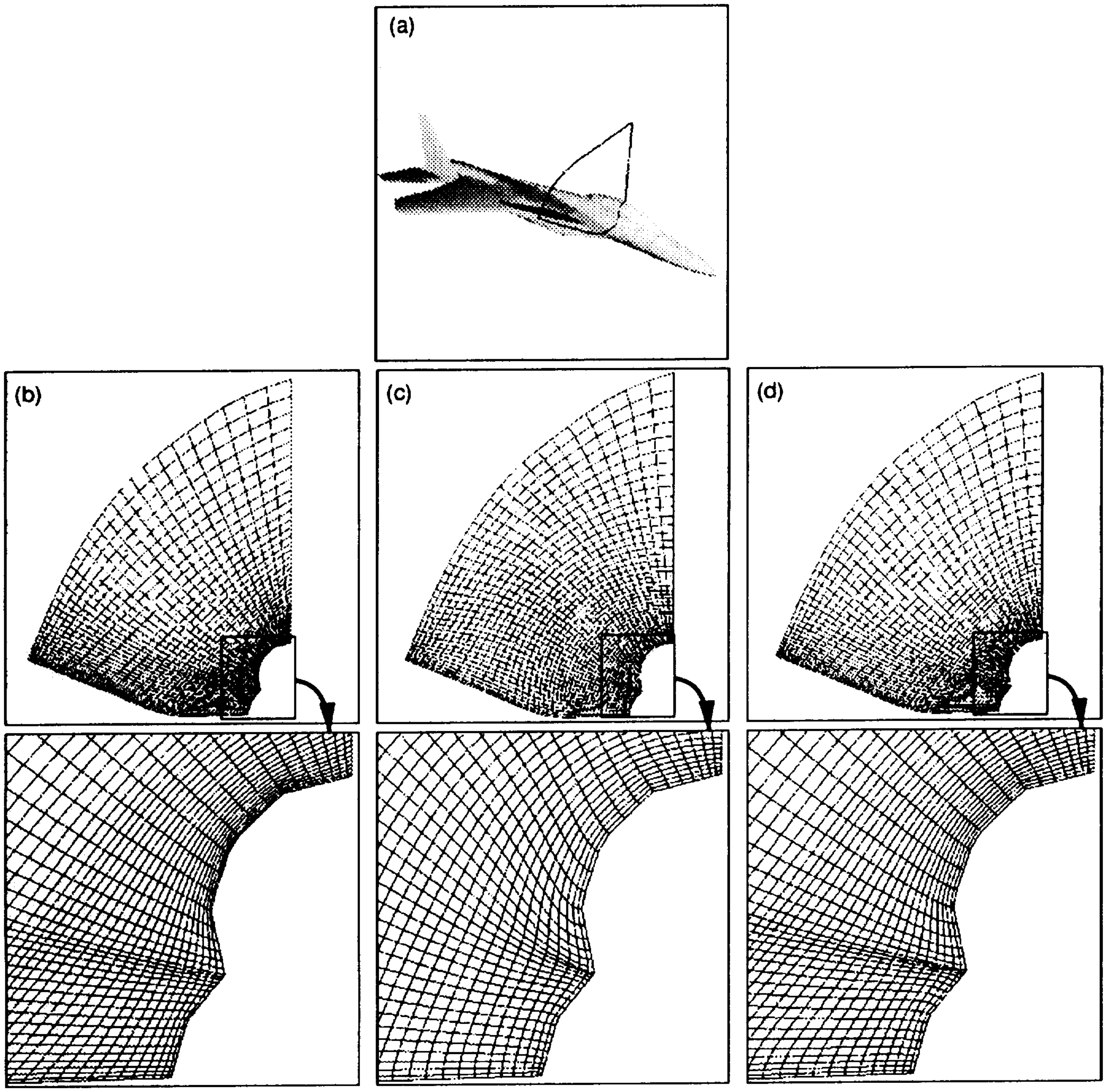

Figure 17: For a plane from a 3D grid (a) three control function formulations are compared: Thomas & Middlecoff (b), Sorenson (c), and hybrid Thomas & Middlecoff and Sorenson (d).

software, which will remain in government contractor domain, is anticipated near mid-year 1992. Long-term plans for GRIDGEN beyond this date include further consolidation of the four component codes, increased grid automation and rework elimination through the new geometry hierarchical structure, further development of CAD system functionality, development and incorporation of improved grid quality procedures such as grid adaption, and expansion of the GRIDGEN umbrella to unstructured grid techniques.

ACKNOWLEDGEMENT

GRIDGEN upgrades are being performed under subcontract to the Computer Sciences Corporation, and under contract to the Air Force Wright Laboratory at Eglin Air Force Base. The authors would like to acknowledge Dr. Robert Smith of NASA Langley Research Center and Mr. Robert Ames of David Taylor Research Center and their respective agencies for initiating and sponsoring these efforts. The authors also acknowledge Dr. Donald Kinsey of the Air Force Wright Laboratories at Wright Patterson AFB for sponsoring GRIDGEN's initial development.

REFERENCES

1. Steger, J.L.: *Technical Evaluation Report on the Fluid Dynamics Panel Specialists' Meeting on Application of Mesh Generation to Complex 3-D Configurations.* ed. by Schmidt, W., AGARD-AR-268, AGARD, France, March 1991.

2. Smith, R.; Choo, Y.; Van Dalsem, W.; and Bircklebaw, L.: *Directions for Surface Modeling/Grid Generation in NASA*. 1991 NASA CFD Conference, NASA Ames Research Center, Mar. 1991.

3. Thompson, J.F; et al.: 'Program EAGLE Numerical Grid Generation System Users Manual." AFATL-TR-87-15, Vols. I-III, Air Force Armament Laboratory, March 1987.

4. Seibert, W.: "A Graphic-Iterative Program-System to Generate Composite Grids for General Configurations." *Numerical Grid Generation in Computational Fluid Mechanics '88*, edited by S. Sengupta et al., Pineridge Press Ltd., Swansea, U.K., 1988.

5. Amdahl, D.J.: "Interactive Multi-Block Grid Generation." *Numerical Grid Generation in Computational Fluid Mechanics '88*, ed. by S. Sengupta et al., Pineridge Press Ltd., Swansea, U.K., 1988.

6. Abolhassani, J.; Sadrehaghighi, I.; Smith, R.E.; and Tiwari, S.N.: "Application of Lagrangian Blending Functions for Grid Generation Around Airplane Geometries." *Journal of Aircraft*, Vol. 27, No. 10, October, 1990, pp. 873-877.

7. Raj, P.; et al.: "Three-Dimensional Euler Aerodynamic Method (TEAM)." AFWAL-TR-87-3074, Vols. I-III, Flight Dynamics Laboratory, Wright Research and Development Center, 1987.

8. Cooper, G.K.; and Sirbaugh, J.R.: "PARC Code: Theory and Usage." AEDC-TR-89-15, Arnold Engineering Development Center, Arnold Air Force Base, 1989.

9. Steinbrenner, J.P.; Chawner J.R.; and Fouts, C.L.: "The GRIDGEN 3D, Multiple Block Grid Generation System." WRDC-TR-90-3022, Vols. I and II, Wright Patterson Air Force Base, 1990.

10. *PATRAN Plus User Manual.* PDA Engineering, July, 1987.

11. Smith, B.; et. al.: *Initial Graphics Exchange Specification (IGES) Version 4.0.* U.S. Department of Commerce NBSIR 88-3813, June 1988.

12. *David Taylor Research Center Spline Geometry Library DT_NURBS Users Manual.* Boeing Computer Services, February, 1990.

13. Thomas, P.D.; and Middlecoff, J.F.: "Direct Control of the Grid Point Distribution in Meshes Generated by Elliptic Equations." *AIAA Journal*, Vol. 18, 1979, pp. 652-656.

14. Sorenson, R.L.: "The 3DGRAPE Book: Theory, Users Manual, Examples." NASA TM-102224, NASA Ames Research Center, July, 1989.