THE NATIONAL GRID PROJECT: A SYSTEM OVERVIEW

Adam Gaither, Kelly Gaither, Brian Jean, Michael Remotigue, John Whitmire,
Bharat Soni, Joe Thompson
NSF Engineering Research Center for CFS
Mississippi State University, MS

John Dannenhoffer
United Technologies Research Center
East Hartford, CT

Nigel Weatherill
University of Wales
Swansea, UK

INTRODUCTION

The National Grid Project (NGP) is a comprehensive numerical grid generation software system that is being developed at the National Science Foundation (NSF) Engineering Research Center (ERC) for Computational Field Simulation (CFS) at Mississippi State University (MSU). NGP is supported by a coalition of US industries and federal laboratories. The objective of the NGP is to significantly decrease the amount of time it takes to generate a numerical grid for complex geometries and to increase the quality of these grids to enable computational field simulations for applications in industry. [1]

A geometric configuration can be discretized into grids (or meshes) that have two fundamental forms: structured and unstructured. Structured grids are formed by intersecting curvilinear coordinate lines and are composed of quadrilateral (2D) and hexahedral (3D) logically rectangular cells. The connectivity of a structured grid provides for trivial identification of neighboring points by incrementing coordinate indices. Unstructured grids are composed of cells of any shape (commonly triangles, quadrilaterals, tetrahedra and hexahedra), but do not have trivial identification of neighbors by incrementing an index. For unstructured grids, a set of points and an associated connectivity table is generated to define unstructured cell shapes and neighboring points. Hybrid grids are a combination of structured grids and unstructured grids. Chimera (overset) grids are intersecting or overlapping structured grids.

The NGP system currently provides a user interface that integrates both 2D and 3D structured and unstructured grid generation, a solid modeling topology data mangement system, an internal Computer Aided Design (CAD) system based on Non-Uniform Rational B-Splines (NURBS), a journaling language, and a grid/solution visualization system.

GRAPHICAL USER INTERFACE

Designing and implementing a consistent graphical user interface (GUI) is a key element in ensuring that the grid generation process runs smoothly and effectively. For this reason, a great deal of thought has been put into determining the requirements under which the GUI must perform, the design by which the requirements are maintained, and the implementation through which the technology may be transferred to the user.

Requirements

Creating an environment for users to comfortably interact with technology is the key priority in creating a usable grid generation system. Ensuring that the environment is suitable follows from closely adhering to the following requirements:

- **Consistency** is perhaps the most critical requirement to maintain. This extends to the presentation of the technology as well as the look and feel of the GUI.
- **A logical organization of technology** has been maintained to present the information in a familiar manner to the user. This is accomplished by examining the function or process for which the GUI is designed, and organizing the process into tasks and subtasks.
- **Efficient use of screen real-estate** is an absolute necessity. Because current technology allows only a relatively small amount of two-dimensional screen space, careful consideration is given to the layout of the GUI.
- **Ease of use** is the facet of the GUI that ensures user acceptability or, more often than not, if the GUI is difficult or cumbersome to use, complete loss of information and inability to achieve user satisfaction.
- **Tailoring the system for user interaction** is a way to always keep in mind the technology that is being presented and the platform on which it is implemented.
- **Extendibility** of the system allows growth with both the technology and the intended user audience. It is a requirement from the initial design phase to allow for adaptability in every aspect of the system.

Design

Having established the design requirements for the grid system, it is imperative to adhere to these requirements when completing the design phase. The design of the system can be thought of as consisting of six physical components and a seventh more abstract concept. The six physical components are the menu bar, the global actions, the global settings, the application palette, the message buffer, and the draw screen. A picture illustrating the layout of these physical components is shown in Figure 1.

The **menu bar** is designed to logically present the tasks and subtasks required to perform a high level process and can be found across the top of the GUI.

The **global actions** seen in the top right corner of the GUI are those top level functions that are required to complete the given task.

The **global settings** are designed to be information holders and are found in the space directly beneath the global actions. This global information consists of information that each task or subtask is going to require to complete the higher level process.

Expanded beneath the global settings is the **application palette**. Each task in the menu bar is considered an application, and when the user wishes to open an application, this is the space designated for its expansion. Expanding or opening a task allows it to become the active application. Because of the design of the system, only one application may be the active application at any one time. The application consists of actions and settings. The GUI is designed to be event driven, and therefore waits to be notified of an event that has occurred in the system. Each event in the system travels through one pipeline, and an event monitor polls the event at the end of this pipeline to determine which type it is. If the event is an application

event, the event monitor notifies the active application that an event has just occurred. The notification takes place through the use of a designated "handshake" routine. When the GUI is created, each application is forced to register a routine that will serve as its "handshake" routine. Once the GUI has given the event to the "handshake" routine, it no longer has control. It is then up to the application to decide how to proceed.

A **message buffer** is provided in the lower right corner of the GUI to provide feedback to the user. It is designated to be non-obtrusive, just giving the user a textual indication of the actions that are being performed and those that have been completed.

The **draw screen** makes up the remaining portion of the GUI. This is the space reserved for all graphical input and output. The draw screen is designed to be highly interactive allowing the user to interface with geometries and grids via the mouse and keyboard. All interaction between the user and the GUI must occur using either or both the global and applications actions and settings. The actions are designed to respond to a mouse press on the button, and to a keypress of the registered mnemonic for that action. The mnemonic is registered when the GUI is created and may not be changed. All invocations of action mnemonics must take place by typing the letter with the keyboard while the mouse is inside the draw screen. The size of the draw screen that is shown is what the user sees when the system is initially invoked. A full screen facility is designed to allow the user to get a larger view of the items that are being displayed. Interaction with the global and local actions is maintained while in full screen mode, but any user interaction with the global and local settings must be carried out when the screen is displayed in its regular size. This occurs because the global actions, global settings, application actions, application settings, and the message buffer are not seen when the draw screen is full size. The page up and page down keys on the keyboard are used to toggle between the regular screen size and full screen size.

As a more abstract concept, **user interaction** must also be addressed during the design phase. User interaction includes mouse interaction and graphical display. Careful consideration is made to allow for consistent behavior both in the mouse and in the interaction with the draw screen. The mouse is for the exclusive purpose of carrying out transformations and of acting as a pointer to a piece of information in the draw screen. Any textual interaction is completed through the combined use of the keyboard and settings. The graphical display is laid out in such a manner that user interaction is the primary focus. Several levels of display quality are provided to facilitate the need for user interaction and for the production of high quality pictures. These levels are maintained through the combined use of the resource file, and the view increase and decrease actions in the global actions space. The resource file is a user editable file that contains X Windows and Motif based descriptions of several components of the system. The user may either edit the file outside the system, or inside the system with the use of the resource editor. Several views may be set up and are each given a unique integer id. The views may consist of properties such as color, line thickness, display quality type, etc. These views may be displayed by typing the view number in the draw screen space and then pressing either the view increment or view decrement action. If there is no number in the numeric argument setting, the current view number is either incremented or decremented depending on the action that was invoked.

Implementation

The NGP system employs the above design requirements, and is implemented in C using X Windows and Motif as the interface builder, and mixed-mode GL on the Silicon Graphics workstation as the graphics language. As OpenGL[2] becomes readily available, the system can be easily ported and then will become available for use on a variety of workstations.

SOLID MODELING TOPOLOGY DATA STRUCTURE

The fundamental NGP data structure is called the Grid Topology Model (GTM)[3]. The GTM is based on a Boundary Representation (B-Rep) radial edge non-manifold solid modeling topology data structure.[4] B-Rep data structures are extensively used in current commercial CAD systems where geometric connectivities are an important aspect of geometric manipulations. The data structure provides explicit connectivity information between all geometric and grid entities, and abstracts the user from underlying geometric orientations. This abstraction provides a set of algorithms that simplify the grid generation process by allowing the user to ignore orientation when building multi-volume unstructured grids, multi-block structured grids, or when changing grid control parameters such as point distributions.[3]

The fundamental form of surface that is useful for grid generation is a non-manifold surface that can be accessed from N adjacent volumes and K adjacent surfaces. For structured grid generation, four-sided surfaces must be used. For geometry modeling and unstructured grid generation, multi-sided surfaces with holes (trimmed surfaces) can be used. The GTM provides the capability to read in trimmed surfaces and provide the same topological operations as four-sided surfaces. Figures 2 and 3 show the original geometry and the trimmed result.

The GTM is designed using a set of requirements formulated through interaction with the grid generation and CFS practitioners in the NGP consortium. The following is a list of the basic goals met by the GTM:

- **A foundation for general grid generation:** Includes 2D and 3D structured, unstructured, hybrid and chimera (overset).
- **No grid point duplication:** Grid points on shared geometries are not duplicated. Duplicating grid points causes wasted memory and holes or gaps between domains due to numerical inaccuracies.
- **Explicit adjacencies between volumes and surfaces:** Adjacencies between geometries and grids are explicitly defined. This enables multi-block point-point matching, point-point mismatching, full-face interfaces and partial-face interfaces. Access methods are available to provide adjacency, grid and orientation information to grid generators, elliptic smoothers, and CFS solvers.
- **Abstraction from geometric orientations:** The user does not have to keep track of block or surface orientations when building a grid.
- **Propagation of grid control information:** The user has the ability to propagate grid control information (number of points and point distributions) in a computational direction within a multi-block topology. The system also verifies that the propagated information is consistent throughout the grid.
- **Geometry verification:** The user is graphically prompted if the current geometry configuration is ready to be used for grid generation. Gaps, holes or overlaps where surfaces do not meet (within a user-specified tolerance) are indicated.
- **Semi-Automatic Boundary Detection (Blocking):** A set of algorithms detect bounded regions for both structured and unstructured grid generation.
- **Composite Edges and Faces:** Composite edges and faces allow for partial-face matching in multi-block structured grid generation.

A hierarchy of topology and geometry elements defines the classes of information contained in the GTM data structure. Each level of the hierarchy represents a level of abstraction between geometry and adjacency elements (Figure 4).[3]

All adjacent elements in the GTM are directly accessible via pointers (Figures 5 and 6). Searching is limited to traversing a doubly-linked circular list with length equal to the number of adjacent elements. Grid generation applications utilize the adjacency elements to store and access grid points on shared geometries. CAD utilities utilize the adjacency and geometry abstraction elements to determine degeneracies, orientations and adjacency. [3]

GEOMETRY ENGINE

All geometry in the NGP system is represented by Non-Uniform Rational B-Splines (NURBS) [5-9]. Some of the reasons the NURBS representation is chosen for the geometrical database are as follows:

- **Unified Mathematical Model:** NURBS allow one mathematical form to represent both analytic and free form shapes. Thus the system can represent such shapes as conic sections, quadrics (cones, cylinders, spheres, etc.), and surfaces of revolution as well as free form sculpted surfaces with a single homogeneous database. The ability of NURBS to represent both analytic and free form surfaces with a single mathematical model is of particular importance because it simplifies coding of algorithms and reduces maintenance requirements on the geometry engine. A single data type can be used to represent all possible geometric entities within the system, and a single suite of evaluation and manipulation routines can be used to interact with them.
- **Generality:** NURBS offer the user flexibility to design a large number of shapes.
- **Efficiency:** NURBS evaluation algorithms are reasonably fast and numerically stable.
- **Geometric Interpretation:** NURBS can be interpreted geometrically. This allows designers with limited knowledge of the mathematics of curves and surfaces, but with a good knowledge of descriptive geometry, to create desired shapes with ease.
- **Local Control:** The property of local control allows relatively small areas of a curve or surface to be modified without affecting the entire spline.
- **Affine Invariance:** NURBS are invariant under affine, perspective, and parallel transformations (e.g. scaling, translation, rotation, shear, etc.).

Geometry File I/O Capabilities

Exchange of geometry data with third party CAD systems is accomplished in NGP via IGES files which comply with IGES version 5.1 [10], but also includes the NASA IGES standard [11]. In addition to reading and writing IGES files in ASCII form, files compressed with the UNIX /bin/compress utility or the gzip utility can also be read. Other file formats which can be used for exchanging geometry information include Gridgen database files [12] and formatted, unformatted, and binary Plot3D [13] files. Discrete curve and surface data such as those in Plot3D files, Gridgen .dba files, and IGES entity 5001 are fitted with a linear or cubic NURBS spline [8].

The Internal CAD System

Unlike other contemporary NURBS-based grid generation systems such as ICEM [14,15], NGP generates surface grids directly on the NURBS patches [16]. This feature allows surface grids to retain a high degree

of fidelity to the original geometry data; however, it also requires a high quality CAD model in order to produce good grids. The model should not contain any unwanted gaps, overlaps, or intersections between surfaces. The primary function of the internal CAD system is to allow the user to construct, modify and/or repair imported geometry in preparation for mesh generation. Models imported from traditional CAD systems generally require some degree of modification in order to make them suitable for use in the grid generator. The modifications may be as simple as deleting unwanted details or as complex as reconstructing large portions of the model. In either case, the CAD system provides the user with a rich set of tools to accomplish the task. In addition, many construction tools are provided which enable complex objects to be created from scratch.

The internal CAD system consists of nine individual applications grouped according to the functionality they contain. The buttons, toggles, and numeric fields within each application are arranged in a way which gives the user visual cues to functionality by grouping functions together with their input and/or option fields. The applications are as follows: **Points**, **Vectors**, **Curves 1**, **Curves 2**, **Surfaces**, **Blocks**, **Utilities**, **Edit**, and **GM Util**.

Point Operations—The point is used in construction of vectors, curves, and surfaces, to designate end points or control information. The point can be created in any application, but other useful operations are grouped in the **Points** application. These operations include averaging of points, total distance between points, angle between three points, and the closest point on a curve or surface.

Vector Operations—The vector is also used in construction of curves and surfaces and to designate slope, direction, or axis information. The vector can be created in any application by two points or a point and the components. The **Vectors** application allows the functions to calculate the normal and tangent vectors on a surface or the tangent vector for a curve. A vector direction can also be reversed.

Curve Operations—The CAD system provides both curve construction and modification/repair functions. The most used functions are located in the **Curves 1** application while lesser used creation functions are located in **Curves 2**. These curve functions consist of: 1) conic sections (circular arcs, circles, ellipses, parabolas, and hyperbolas). 2) spline curves (interpolation through points), 3) basic curves (lines, quadratic curves, and cubic curves), 4) averaging curves, 5) offset curves 6) splitting curves, 7) unioning curves, and 8) intersecting curves.

Surface Operations—It is generally necessary to modify an existing CAD model to correct for defects before it can be used for mesh generation. The functionality outlined below attempts to minimize the time and effort required to render a given set of surfaces usable for grid generation. Unlike the curve functions, all the surface construction functions are contained in a single **Surfaces** application. The following surface operations are available: 1) blending/ruled surface (two curves), 2) Transfinite Interpolation (TFI) surface (four curves), 3) degenerate TFI surface (three curves), 4) sweeping (one curve along another), 5) extruding (curve along a vector direction), 6) revolving a curve, 7) carpeting (projected TFI surface over a surface network)[17], 8) averaging, 9) extending or extrapolating, 10) intersecting surface with surface, 11) intersecting curve with surface, 12) splitting, 13) unioning, 14) smoothing (Smooths a surface according to curvature [5]), 15) reparameterizing, and 16) surface from cross sections.

Most of the above operations are exact in that the result is computed directly from the NURBS form of the given information. Exceptions to this are the carpet surface and surface from cross sections.

Utility Functions—The Utility functions are generic functions that can be performed on any NURBS curve or surface. These can be grouped into three categories: 1) transformations (translations, rotations,

scaling, and mirroring), 2) projection, and 3) extraction.

AUTOMATIC BLOCKING

One of the most labor-intensive tasks associated with generating a block-structured grid is the creation of a framework of edges, faces, and blocks which together define what is known as the block structure. It is not uncommon for the generation of this block structure to consume well over half of the labor associated with any grid generation problem. Clearly, techniques for efficiently generating such block structures have the potential to increase the utility of block-structured grids into arenas which today are treated with overset or unstructured grids.

Basic Approach

The basic idea behind the automatic generation of block structures is to shift the user paradigm from one which is *prescriptive* to one which is *descriptive*.

In a prescriptive process, the user has a set of low level tools which prescribe exactly *how* the problem is to be solved. Examples of these low level tools include drawing a line in space to serve as an edge of some block, generating an edge which is constrained to lie along a certain input surface, and connecting four edges into a face. While control at this level is certainly very powerful (and sometimes indispensable), it is clear to see that the labor associated with such an approach grows very rapidly, especially in configurations with multiple interacting components.

Alternatively, a descriptive process describes *what* the grid should look like. Example descriptions include the fact that the grid should wrap around the wing leading edge, that the inboard portion of the wing grid should lie on the fuselage, and that the store grid should lie within the wing grid. With this approach, the user could employ a divide-and-conquer approach, where the grid in the vicinity of each component is described separately, and the automatic blocker worries about interacting them to achieve an overall block-structured grid.

The key to being able to *describe* a block-structured grid is the development of a block-structuring *language* of some sort. In the current work, a graphical language composed of objects called cubes, wraps, attaches, holds, rulers, and spacers is used. (More details on the specifics of this language can be found in another paper in these proceedings [18].)

Integration with NGP

As described above, the NGP system is composed of a large number of *applications* which execute independently, but which share the data which describes the configuration and grid. A key to integrating an automatic blocker into such a system is determining how and when the blocking procedure should be executed and on which data the blocker should operate.

An overall view of the grid generation process suffices to answer the first question:

- define the configuration, including any repair (to fill gaps, etc.) and augmentation (such as generating a far-field boundary) which is required to define a water-tight domain;
- develop a block structure, including specification of required number of grid points in each region, any user-specified spacings, etc.;

- generate the grid on the block structure, enforcing any grid controls specified above; and
- visualize and inspect the grid and either loop back to one the steps above to repair it or output it for use by a flow solver.

Clearly, an automated blocker should be executed after the configuration is defined but before the grid is generated.

The answer to the second question is not as straightforward. It is useful to think about two independent topologies within the grid generation process:

- the first is the topology of the configuration, which was essentially prescribed by the interrelationships of the surfaces and curves which define the configuration. The topological information contained here is generally of the form "the west edge of surface A is coincident with the north edge of surface B", or "curve C is formed by the intersections of surfaces D and E". This topological information is crucial for determining the structure and connectivity of the inputs.
- the second is the topology of the block structure, which prescribes the interrelationships of the edges, faces, and blocks which together define the block structure of the grid.

While there are certainly some correspondences between these two topologies, one in general does not want the configuration topology to constrain the block structure topology to any great degree. So in essence, the automatic blocker's job is start with an original (configuration) topology and to create a new (block structure) topology.

This is accomplished with the following procedure:

1. if the current topology is not a block-structure topology (that is, one generated in step 3 below by a previous execution of the automatic blocker),
   - then, save (in a file) a copy of the current (configuration) topology
   - otherwise, restore the (configuration) topology which was previously saved (in the file);
2. provide graphically-driven commands for the user to create the blocking objects (cubes, wraps, attaches, holds, rulers, and spacers) which describe the block structure;
3. create a new (block structure) topology within NGP which corresponds to the edges, faces, and blocks of the automatically-generated block structure; and
4. remove the original (configuration) topology from the system. Note that this step does not delete the curves and surfaces which define the configuration, but rather the topological information on which the block-structured grid generator will execute. (That is, we only want the block-structured grid generator to execute on the new (block structure) topology.)

Note that the first step, which involves the saving and restoring of the original (configuration) topology, is needed so that the automatic blocker always executes on a configuration topology.

One final note: it was stated above that control at the very detailed (low) level is sometimes indispensable for the generation of a suitable grid. This control can still be exercised within NGP by using the low-level

tools on the new (block structure) topology before the grid is actually generated; the only potential problem which one runs into when doing this is that these low level "tweaks" will be lost if the automatic blocker is re-executed (since is will again start with the "saved" configuration topology).

STRUCTURED GRID GENERATION

The structured grid generation capabilities in NGP are based on the GTM data structure. This expedites the generation of both 2D and 3D systems. The use of the topological entities of the system (i.e. edges, faces, and blocks) to manage the grid information for each domain, allows the grid points to be stored once and be used on as many adjacent entities that share these common points. These shared points are determined by the topological creation of the faces and blocks in the domain.

Topological Definition

All NURBS surfaces are automatically identified as topological faces. Other topological faces are created in a Right Handed System (RHS) from the wireframe description automatically, through the connectivity provided by the GTM data structure or the bounding curves can be manually selected. A recursive Depth First Search (DFS) is used to find all possible non-degenerate three and four curve cycles in a set of curves. These faces do not have any geometry associated with them other than which curves compose the boundary.

The topology-based blocking algorithm also uses the connectivity information provided by the GTM data structure to detect O-grid and H-grid blocks. A DFS algorithm is also used to find all possible three and four surface cycles that have valid end caps or the faces can be selected manually. Once a topological block has been detected, the surface orientations are flipped until a valid RHS is detected.

With the topological mapping of the edges, faces, and blocks, partial face matching is allowed by the construction of composite edges and faces. *Composite* edges are obtained by the concatenation of a series of edges into a single string of grid points, and *composite* faces are generated by the topological "linking" of the sub faces into a single consistent set of faces.

Point Distributions

With the domain decomposition done in conjunction with the GTM, setting the number of points and setting spacing requirements is simplified by the propagation abilities implied by the data structure. The number of points applied to an edge of a face is automatically propagated to the opposing edge of the face. Through the connectivity the number of points are propagated to all other edges that lie in the same computational direction. Point distributions can be set on a single edge or propagated to a series of selected edges at the appropriate endpoints. The system also allows the user to specify interior distributions on an edge at selected points specified by the user, or by the detection of discontinuities. The topological mapping also allows spacings to be matched at vertices of connecting faces in a 2D system or for connecting blocks in a 3D system.

Surface Grid Generation

Three types of surfaces are present in the grid generation system, NURBS surfaces, Four Curve Surfaces (FCS), and *Composite* Faces.

NURBS Surfaces—All meshes generated on NURBS surfaces are calculated in the parametric space of the underlying geometry to ensure the accuracy of the grid generation process. [16] Transfinite Interpolation (TFI) with arclength based interpolants is used to calculate the grid points in parametric space.

Four Curve Surface—TFI is used to generate the free form mesh for a FCS. This mesh can be projected onto a network of NURBS surfaces, thus creating a grid *carpet*.[17] Figure 7 depects a typical carpet grid spanning over surfaces patches of a shuttle canopy.

Composite Face—A *composite* face surface grid is created by the logical mapping of the grid points of the surface grids that compose the face into a single set of points.

Any modification of the edges automatically marks the adjacent faces as "dirty" and are regenerated. If any of the sub domains of the *composite* face are changed, the *composite* face surface grid is automatically updated to reflect the alteration.

Volume Grid Generation

Volume grid generation is done on the blocks throughout the field using the 3D TFI routine. Each block is gridded individually, sharing common face, edge, and vertex points between blocks. Volume grids can also be generated through the use of Hypgen.

Hyperbolic Grid Generation—The hyperbolic grid generation algorithm is used to generate a 3D volume grid by marching away from an initial surface definition with a given initial step size and a stretching function in the normal direction. This is accomplished by solving the 3D hyperbolic grid generation system of equations (two orthogonality relations and one cell volume constraint).[19] Solving this system of equations allows for nearly orthogonal grid generation with excellent clustering control, and can generally be generated in orders of magnitude less computer time than elliptic methods. The method does not, however, allow outer boundary location to be precisely specified, and any discontinuities of the original surface tend to be propagated into the volume field.[20]

Hyperbolic grid generation is available in NGP through the use of the NASA Ames developed Hypgen hyperbolic grid generator. In cooperation with NASA Ames, Hypgen has been integrated into NGP as a subroutine to the overall structured grid generation. The system now allows for several hyperbolic grids to be generated at once, or several grids in series, if different control settings are required for each block. Once the hyperbolic grid has been generated, the information from the block is stored in the same data structure as the rest of the system, including the five faces, eight edges, and four vertices created by the hyperbolic grid.

Elliptic Grid Smoothing

Elliptic smoothing is available in both the 2D and 3D systems in the form of surface and volume smoothers. FCSs are not smoothed as a single entity in the 3D system, but are smoothed in the volume smoothing. These faces are smoothed in two ways. First, if the face is shared by two blocks, a "sandwich" volume between the two blocks is created and that volume will be smoothed to keep the face updated with the interior. The second form of smoothing is performed if the FCS is a boundary face. The user sets the face to a sliding point condition, which will allow the points to float in accordance with the volume grid.

2D Elliptic Smoothing—To achieve slope continuity between regions, edges shared by two regions are smoothed by extracting the points for the edge and a strip of points from each adjacent face. Only the interior of the "sandwich" is smoothed, thereby adjusting the original edge to provide better continuity between faces. Vertices completely enclosed by regions are smoothed using a similar approach to receive continuity at the abutting of regions in the interior of the field grid.

Surface Elliptic Grid Smoothing—For 3D surface grids associated with NURBS surfaces, two smoothing

techniques are available. The first being an algebraic adaptive method to uniformly smooth the arclength distribution of the grid in the parametric space of the surface. This technique is usefully when a surface with an initially estranged parameterization is encountered (see Figure 8).

The second method of smoothing employs the elliptic solver on the parametric values of the NURBS Surface. Once the parametric grid has been adjusted by the elliptic solver, the parametric values are re-evaluated to give the new X,Y,Z points. Boundary conditions applicable for the NURBS surface smoothing are as follows: Fixed, Neumann Orthogonality, or Grape Orthogonality.

Volume Elliptic Grid Smoothing—To achieve slope continuity between blocks, faces shared by two blocks are smoothed by extracting the points for the face and a plane of points from each adjacent block. Only the interior of the "sandwich" is smoothed, thereby adjusting the original face to provide better continuity between faces. Edges and Vertices completely enclosed by blocks are smoothed using a similar approach to receive continuity at the abutting of blocks in the interior of the field grid.

UNSTRUCTURED GRID GENERATION

Unstructured grid generation follows the same steps as the structured grid generation. First the boundary curves are specified, the surface grids are calculated, then the volume grids are calculated. The only difference between structured and unstructured is that the domain does not have to be decomposed. The user only has to determine a valid boundary of curves for 2D, or surfaces for 3D, prior to starting the grid generation process. In 2D, the user can manually pick boundaries and specify them, or use an automated feature that will detect the inner and outer boundaries of a configuration. In 3D, a set of fully connected surfaces is all that is needed.

Support is available for single and multiple domain 2D and 3D unstructured grid generation, parametric surface grid generation and multi-curve surface creation (where only curves are needed to build an unstructured 3D surface grid). Currently, both the Delaunay and Advancing Front methods are used for both surface and volume generation. These methods differ only in the point insertion criteria since both use the same boundary recovery technique as described in [21]. The points are automatically generated using the boundary point spacings to define the interior distribution and number of points. Boundary integrity is imposed by triangle and tetrahedra transformations. The implementation of the unstructured approaches are well documented in [21,22]. Figure 9 shows a Delaunay unstructured grid on region of the trimmed gasket. Figure 10 is an Advancing Front solution on the same region. Using these methods provide extremely efficient grid generation for very large problems. Most grids for super-computing scale solutions can be generated interactively on mid-range engineering workstations.

2D Grid Generation

2D unstructured grid generation entails that the user specify planar curves that form an outer boundary and any number of inner boundaries. Currently, these curves must be on the XY plane. Once the geometry is built, the user must build loops to identify the outer and inner boundaries (a single button push). Once the boundaries are identified, the user must indicate the number of points on each boundary, point distributions, point sources and/or curve sources in the field.

Surface Grid Generation

To retain the accuracy of grid points on a surface, unstructured surface grids are calculated in the parametric space of the underlying geometry. [21] The generation of an unstructured grid on a surface is

defined by two independent variables U and V. This allows the grid to be computed in a square 2D space then mapped back into arbitrary 3D space by evaluating the UV values into XYZ values. This mapping is dependent on the shape of the surface and its underlying parameterization. For degenerate surfaces (a surface with one or more edges that collapses to a point) it is necessary to modify the shape of the parametric space.

Parametric Problems on Surface—Unfortunately, the parametric distribution on the underlying surface may not be optimal for grid generation. Currently NGP uses a mapping technique developed at McDonnell Douglas as part of their in-kind contribution to NGP. However, these mapping functions cannot solve all of the problems with parametric skewing in the grid. The ultimate answer (yet to be implemented) is to iterate between physical space and parametric space as it is done in the structured grid generation system.

Sources

Point sources and curve sources control the size and density of grid points in a specified area or volume. Sources are available both on surfaces and in the volume. A source specified on a surface effects both the surface and the volume grid. The sources can be specified directly from a solver to enable grid adaptation through the paradigm. Point and curve sources are associated with vertex and edge topology elements respectively. A user-specified number of point sources can be placed on a curve (in parametric or physical space) in the same manner grid points are placed on an edge.

Source effects through boundaries—Currently, a source placed near a boundary has no effect on the boundary. Many problems are caused by this including rapid change in element size and extremely skewed elements. To offset some of the effects of the problem, boundary elements have priority over point source elements. This ensures that the boundary grid is consistent adjacent to geometry (for at least the first element width). However, this is only a temporary fix, the obvious answer to this problem is to let the effects of a source "pass through" the boundary. In 2D, the source would effect both boundary grids (edge grids) and any multi-domain regions on the other side of the boundary. In 3D, the source would effect edge grids, surface grids, and any other multi-domain region within the sphere of influence of the source.

Multi-domain Unstructured Grids

In some problems it is necessary to define multiple regions, surfaces and/or curves within a field that must have grid points associated with it. Tools are available to the user that allow surfaces or curves in the interior of a grid domain to be marked as "transparent". This allows the grid generation algorithm to fix points on the original surface or curve, while building tetrahedra or triangles on both sides.

Surface Defined by Arbitrary Curves in Space

Another feature of the 2D unstructured grid generation is the ability to build a 3D curved surface from a set of curves in space. This is extremely useful for capping open boundaries with an arbitrary surface definition where otherwise the original geometry would have to be cut to enable capping with four sided surfaces. Another useful feature for this type of surface creation is building planes of symmetry where a complicated geometry (such as an airplane) needs to only include half the geometry. Instead of making the outer boundary out of tens (or hundreds) of small surfaces, the user can simply define a few large surfaces for the outer boundary, and an arbitrary surface for the symmetry plane. The only restriction is that the multi-curve surface definition can only be created if the surface is singular (non-overlapping boundaries).

VISUALIZATION

Over the past two decades, significant progress has been made in the levels of physical realism that can be incorporated in the numerical simulations of fluid flow. At the same time, slow but steady progress has been made in increasing the range of geometrical complexity which can be accomidated. This has been achieved through a combination of grid types, including block-structured, unstructured, and hybrid grids.

Currently the vast majority of labor hours in a typical CFD study is consumed in the grid generation phase. This is primarily because the closed-form solutions do not exist for flow fields around complex configurations and numerical techniques have to be used, thus requiring grid generation techniques to discretize the given domain. The process of generating a grid can be thought of as a set of iterations cycling through the following steps:

- **Specify:** The various "inputs" needed to generate a grid are prepared in this step. Examples of typically required inputs are grid topology, stretching factors, and numbers of points in various regions. By its very nature, this step is interactive.
- **Generate:** The actual grid points, their locations, and their interconnections are generated in this step. Because of its compute-intensive nature and its relatively long computation time, this step is usually performed in a batch mode.
- **Evaluate:** The user then needs to determine if the grid meets the requirements set by the expected flow physics and the flow solver which will be used. As with "specify", this step is interactive in nature.

Much effort has been expended in reducing the total time needed to generate grids. Some of the efforts has been centered on seamlessly integrating the above steps, resulting in the production of grid generation systems such as EAGLEView [24], GRIDGEN [12], GRAPEVINE [25], and ICEM [14,15].

Current grid evaluation techniques fall into two groups. The first involves the stand-alone computation of "grid-quality" measures, allowing the user to assess the local goodness of the grid. The second set of grid evaluation techniques is aimed at producing visual representations of the grid and possibly its quality measures. Unfortunately, to date insufficient effort has been focused on techniques for the rapid and effective *interactive* evaluation of grids in their various stages of generation. Such interactivity is essential for providing information about the grids in a way which can enable the user to correct or improve the grid with the available grid tools.

Hence, the goal of the present grid visualization system is two-fold: first, to provide a suite of techniques for effectively assessing the quality of block-structured, unstructured, and hybrid grids; and second, to ensure that the techniques can be used during the grid generation process as an aid in assessing the validity of the operations performed thus far.[26]

Design Considerations

The design of the techniques presented here was driven by two considerations: what questions would the user like the system to answer, and which visualization tools convey the answers to those questions most succinctly.

The questions which the user is likely to ask include:

- Has the grid been generated in the correct region in space?
- Are there parts of the grid where the grid points are either unacceptably close to or far from each other?
- Are there local areas of the grid which have problems? If so, where are they and more specifically, what are the problems?
- What are the causes of the problem(s) and how can it (they) be fixed?
- What is the overall quality of the grid and is it valid to progress to the next phase (i.e., surface grid to volume grid to field solution)?

The selection of an appropriate set of visualization tools was guided by examining those used in other systems and identifying weaknesses. The major shortcomings identified included:

- **Information overload:** Many grid systems are not discriminating about what is displayed. Wire frame displays of grids, even those which employ advanced techniques such as depth-cueing and motion, are extremely hard to decipher.
- **Information deficit:** The current systems do not "point out" to the user the problems which exist in a grid. Many CFD projects have been sabotaged by bad grids which did not get discovered until after the flow solver engineer labored intensively over the problem.
- **Lack of context:** There are really three problems here. First, most current techniques do not give the user an effective frame of reference; once the user is alerted to the problem, it is hard to determine exactly where the problem is and how it interacts with the configuration. Second, they do not show the interrelationships between the various grid quality measures which may be locally important. And third, the techniques which are used separate the display from the discrete nature of the grid cell data. Iso-surface plots of grid quality only convey that there is a problem, but not why there is a problem.
- **Difficult to learn:** Because most systems are nonintuitive, they have an extremely long learning curve.

Visualization Hierarchy

Analysis of the above questions shows them to be hierarchical in nature, thus leading naturally to a set of visualization tools which are also hierarchical. The following sections describe a specific hierarchy of grid visualization and analysis tools used to not only demonstrate the quality of a grid, but to allow the user to specifically isolate the area or areas of the grid that are causing problems. Making up the hierarchical tree of evaluation are qualitative tools which allow individuals to determine the basic overall quality of a grid by inspection, quantitative methods to flag areas of numerical interest, and grid browsing techniques to step into and through the grid and view the areas of interest. This visualization hierarchy is intended to act as a three-stage process for determining the quality of a grid, with each level portraying a more localized and detailed set of information.

Qualitative Evaluation—The qualitative evaluation stage is intended to assess the overall placement of grid points, both for block-structured and unstructured grids. Since at this initial stage in the analysis, only grid points (and not their connections) are important, a visualization tool called a *point cloud* was developed. This visual display is created by placing a small point on the screen at each grid node. An example of an unstructured *point cloud* is shown in Figure 11.

The simplicity of this display technique offers two distinct advantages. First, because of the absence of grid lines, the screen is relatively uncluttered, especially as compared with previous visualization techniques; this is very important for unstructured grids which heretofore have proven difficult to visualize. Second, the interactive speed of pans, zooms, and rotates is high enough so as to make "motion" of the display an effective means of extracting the three-dimensional nature of the grid.

Because this technique is fast, and because it has been included "on line" within the NGP system, any deficiencies which are discovered with this tool should be fixed before continuing on with the grid generation process.

Quantitative Evaluation—The next, more in depth, evaluation stage is quantitative evaluation. Here, the objective is to elicit from the system quantitative measures of grid quality. Of course, there is no single measure of quality but rather a set of mutually interacting measures such as skew and stretch. The relative importance of each of these measures can only be assessed in the context of the flow solver which is to be used.

The need to present complex, mutually-interacting data is not unique to grid visualization, or even to scientific visualization as a whole. A particularly clever solution to this problem, the *weathermap*, has been developed by meteorologists. On one screen, this can convey information such as the general weather patterns as well as the location of critical weather areas.

By analogy, a grid *weathermap* can be used to tell the user "at a glance" what the overall quality metrics of a grid are and in particular which areas of the grid need improvement. This is done in the present work by the superposition, on the graphics screen, of:

- **node-based grid quality measures** - Certain grid quality measures are node-based, that is, they are defined to be valid at a given node. Examples include the ratio of maximum-to-minimum volumes of elements attached to a given node or the number of edges incident at a node. Specific examples are described below. Nodes whose quality measure fall outside some user-specified tolerance band are displayed as a color-coded symbol (the color indicates which quality measure is out of bounds). In addition, it has been found to be useful to connect neighboring nodes which exceed the threshold with the appropriate grid lines.
- **cell-based grid quality measures** - Other grid quality measure are valid for a grid cell. Examples include the skew of a cell; again specific examples are described below. The cells whose grid quality measures exceed the user-specified thresholds are drawn directly. Using this technique, not only are "out of tolerance" areas of the grid highlighted, but by showing the shape of the offending cell(s), the user is better able to determine the cause of the problem.
- **the configuration** - This gives the user a sense of context, that is where the problems are in relation to each other and to the critical parts of the given geometry. This too aids the user in isolating the cause of the problem.

It is a specific strength of the design of the *weathermap* technique to allow users to both adjust the tolerance of the given measures as well as add any new measures which are appropriate to a specific field solver.

Grid Browsing and Querying—Once an area has been identified as being critical, methods are needed to view and analyze the behavior of the cell and the nodes in that region. This gives the user a method for understanding the problem and hence its cause. Many techniques have been tried, but most are confusing.

Current techniques generally do not give the user a direct three-dimensional view of the data, but rather require the user to mentally assemble a three-dimensional view from a series of two-dimensional slices.

The visualization tool which was developed here to browse block-structured grids gives the user an actual three-dimensional view of an i, j, or k-plane of grid points by constructing a three-dimensional model of the grid cells adjacent to the selected plane. This is accomplished by superimposing opaque, lighted representations of the bottoms and sides of the appropriate cells, thus creating an image which looks similar to an *egg carton*. By viewing into the cells, the user can get a realistic view of the shape of the cell and its size relative to the neighbor. With the addition of motion parallax (via interactive zooming, panning, and rotating), the three-dimensional nature of the cells is even more apparent.

An additional difference between the current and traditional techniques is that the new technique automatically extends the grid plane in the current block logically into its neighboring blocks. In this way, the user gets a direct sense of the entire grid, that is not a block-by-block snapshot. Of course, sometimes the user wants to examine a block which has been graphically disconnected from its neighbors. In order to accomplish this, the user has the ability of exploding the grid; this is analogous to the exploded (or assembly) view which is often found in service manuals.

A similar visualization technique is applicable to unstructured grids. The difficulties here are the lack of grid planes; so instead, one must substitute a plane in physical space to select the cells to display. Also, determining which face to not show so that a user can look into the cell is not as easy as it was in structured grids. For these reasons, the *egg carton* has not yet been implemented for unstructured grids.

For both block-structured and unstructured grids, the user can query the system for information through a simple graphical pick. The information returned to the user consists of the identity and attributes (location and quality metrics) of the picked entity, as well as the identity and attributes of all entities connected to it. This operation can be applied recursively to form progressively larger shells. The numerical information returned by the querying mechanism has proven to be extremely useful in debugging some of the grids which have been generated to date.

SUMMARY

The various components of the NGP system have been presented and can be summarized as follows:

- The **user interface** is designed to be easily customized, extended and ported to a variety of workstation hardware platforms. The entire interface environment can be modified by the user either interactively through the interface, or through "resource" files. New applications can be added with minimum effort. X-windows/Motif is used to enhance portability due to its availability on several types of hardware platforms. The same global metaphors are used for user interaction throughout NGP to ensure consistency between applications.
- The **data structure** is based on a Boundary Representation (B-Rep) radial edge non-manifold solid modeling data structure for both surface and grid topology. All geometry and grid connectivities are explicitly defined. The user is abstracted from the underlying geometry orientation, and a set of algorithms is provided that simplifies grid generation, surface interrogation and geometry construction.
- The **Computer Aided Design (CAD)** system uses a Non-Uniform Rational B-Spline (NURBS) geometry representation. Geometry can be imported into the NGP via the Initial Graphics Exchange Specification (IGES), discrete XYZ's or can be created by the internal CAD system. Geometry read into the NGP system is converted to a NURBS representation. CAD tools are available within the

system that allow the user to build or modify points, curves and surfaces. An important feature of the CAD system is the ability to lay a "carpet" surface over several surfaces to fix defective geometry definitions having gaps and overlaps between surfaces.

- The **automatic blocking system** provides a suite of tools for quickly generating the set of edges, faces, and blocks which define the framework for a block structured grid. This block structure is automatically generated given just an abstraction of the configuration, simple blocking commands (such as wrap a grid around the wing leading edge), and other descriptive information (such as the number of points in various regions and any reguired clusterings). Block structures generated with this technique generally require about an order of magnitude less labor to generate than is required by the traditional constructive techniques used in other block structured grid generators.

- The **structured grid generation** system enables the user to create both 2D and 3D structured grids. Surface grids are calculated in the parametric space of the underlying NURBS. Both surface and volume grid generation are available for 3D, and planar grid generation is available for 2D. Automated face and block detection algorithms allow the user to concentrate on building the blocking structure around a complex geometry without the burden of having to define face-to-face and block-to-block orientations. Elliptic smoothing across domains is available on the surfaces and in the volume.

- The **unstructured grid generation** system enables the user to create both 2D and 3D unstructured grids using both Delauney and Advancing front methods. All surface grids are created in parametric space on the NURBS. 2D grid generation uses wire frame geometry while 3D grid generation uses surface geometry. Automated loop detection is available for 2D grid generation and surface trimming. Surface and volume grid smoothing and optimization options are also available.

- The **visualization** system provides diagnostic feedback on the quality of the grids and selected CFS solutions calculated on the grids. A number of viewing options are available for both structured and unstructured grids. Plane sweeping through multiple blocks, *egg-carton* and *weathermap* options are available for structured grids. *Point cloud* and *weathermap* options are available for unstructured grids.

REFERENCES

1. Thompson, J.F.: The National Grid Project, Computing Systems in Engineering, Vol. 3, Nos 1-4, pp. 393-399, 1992.

2. OpenGL Architecture Review Board, *OpenGL^{TM} Reference Manual: The Official Reference Document for OpenGL, Release 1*, Addision-Wesley Publishing Company, 1992.

3. Gaither, A.: A Topology Model for Numerical Grid Generation. *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Proceedings of the 4th International Conference, Pineridge Press Ltd, Swansea, U.K. April 1994.

4. Weiler, K.J.: Topological Structures for Geometric Modeling, PHD Dissertation, Rensselaer Polytechnic Institute, Troy, New York, 1986.

5. Farin, G.: Curves and Surfaces for Computer Aided Geometric Design. Third Edition, Academic Press, San Diego, CA, 1992.

6. Mortensen, M.E.: Geometric Modeling. Wiley & Sons, New York, NY, 1985.

7. Piegl, L.: On NURBS: A Survey. IEEE Computer Graphics & Applications, pp. 55. January 1991.

8. Yamaguchi, F.: Curves and Surfaces in Computer Aided Geometric Design. Springer-Verlag, New York, NY, 1988.

9. Hoschek, J.; and Lasser, D.: Fundamentals of Computer Aided Geometric Design. A K Peters, Ltd., Wellesley, MA, 1993.

10. *The Initial Graphics Exchange Specifications (IGES) Version 5.1*, National Computer Graphics Association, IGES/PDES Organization, 1991.

11. Blake, M.W.; Chou, J.J.; Kerr, P.A.; and Thorp, S.A.: The NASA-IGES Geometry Data Exchange Standard. NASA CP-3143, April 1992.

12. Steinbrenner, J.P.; Chawner, J.R.; and Fouts, C.L.: The GRIDGEN 3D Multiple Block Grid Generation System, WRDC-TR-90-3022, Vol. I & II. 1990.

13. Walatka, W.Z.; Buning, P.G.; Pierce, L.; and Elson, P.A.: PLOT3D User's Manual, Version 3.6, NASA TM-101067, 1990.

14. Akdag, V.; and Wulf, A.: Integrated Geometry and Grid Generation System for Complex Configurations. NASA CP-3143, April 1992.

15. Bertin, D.;, et al: A New Automatic Grid Generation Environment for CFD Applications, *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Proceedings of the 4th International Conference, Pineridge Press Ltd, Swansea, U.K. April 1994.

16. Khamayseh, A.: Elliptic Surface Grid Generation on Analytically Defined Geometries. PHD thesis, Mississippi State University, Mississippi State, MS, 1994.

17. Jean, B. A.; and Hamann, B.: Interactive Techniques for Correcting CAD data, *Proceedings of the 4^{th} International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, April 1994.

18. Dannenhoffer, J.: Automatic Blocking for Complex 3D Configurations. NASA CP-3291, May 1995.

19. Chan, W.M.; Chiu, I.; and Buning, P.G.: User's Manual for The HYPGEN Hyperbolic Grid Generator and the HGUI Graphical User Interface, NASA TM-108791, 1993.

20. Chan, W. M.; and Steger, J. L.: Enhancements of a Three-Dimensional Hyperbolic Grid Generation Scheme. Elseveir Science Publishing Co., New York, NY, 1992.

21. Weatherill, N.P.: Adaptive Inviscid Flow Solutions for Aerospace Geometries on Efficiently Generated Unstructured Tetrahedral Meshes, AIAA-93-0341, 1993.

22. Marcum, D.L.; and Weatherill, N.P.: Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection. AIAA-94-1926, June 1994.

23. Marcum, D.L.: Generation of Unstructured Grids for Viscous Flow Applications. AIAA-95-0212, January 1995.

24. Remotigue, M.G.; Hart, E.T.; and Stokes, M.L.: EAGLEView: A Surface and Grid Generation Program

and Its Data Management. NASA CP-3143, April 1992.

25. Sorenson, R. L.; and Mccann K. M.: GRAPEVINE: Grids about Anything Poisson's Equation in a Visually Interactive Networking Environment. NASA CP-3143, April 1992.

26. Parmley, K.: Techniques for the Visual Evaluation of Computational Grids. AIAA-93-3353, July 1993.

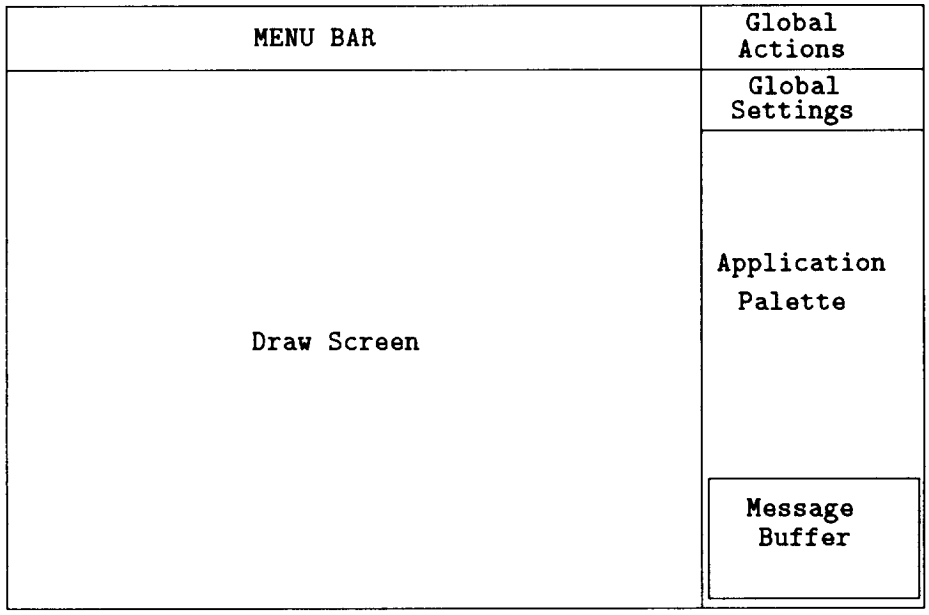

Figure 1: GUI layout of NGP

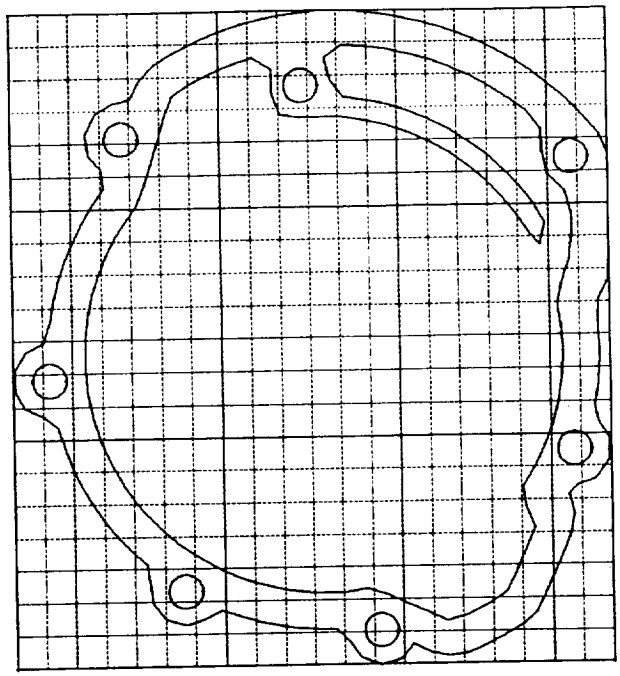

Figure 2: Untrimmed geometry of gasket.

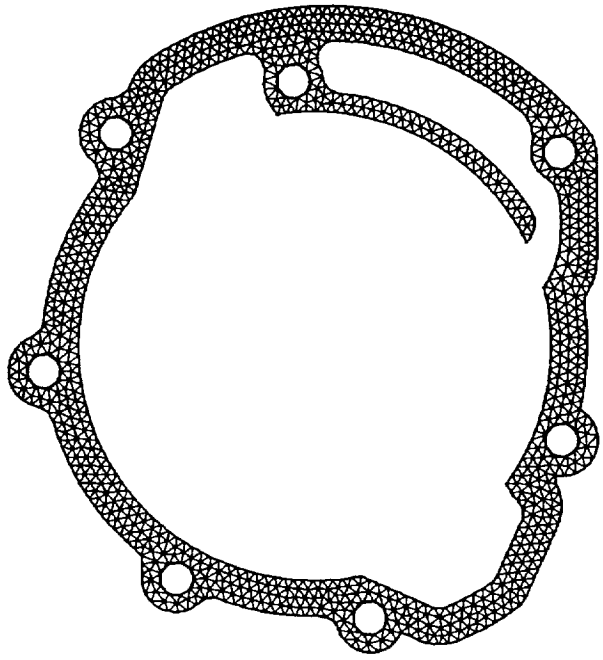

Figure 3: Trimmed geometry of gasket.

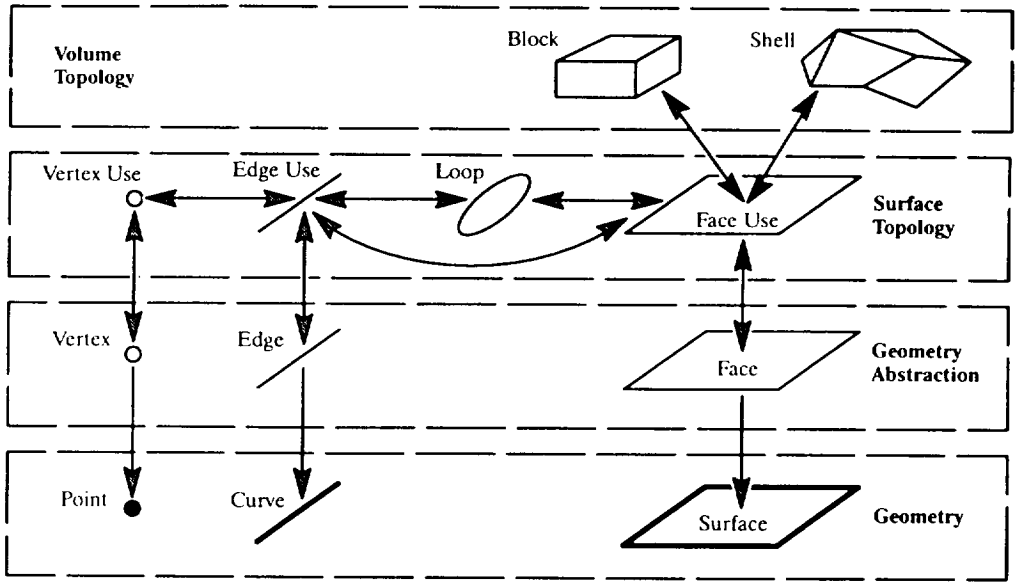

Figure 4: Topology hierarchy.

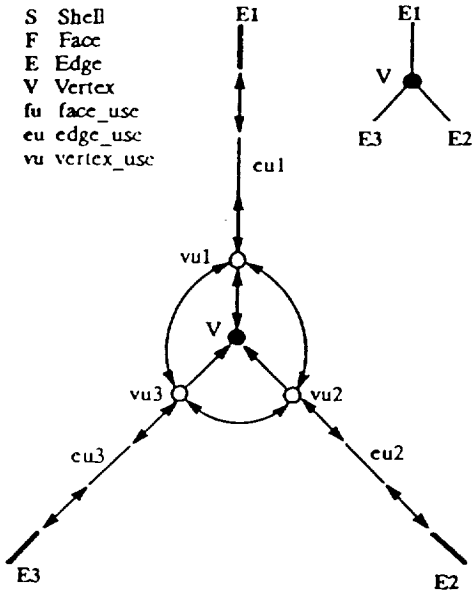

Figure 5: Topology of three edges sharing a vertex.

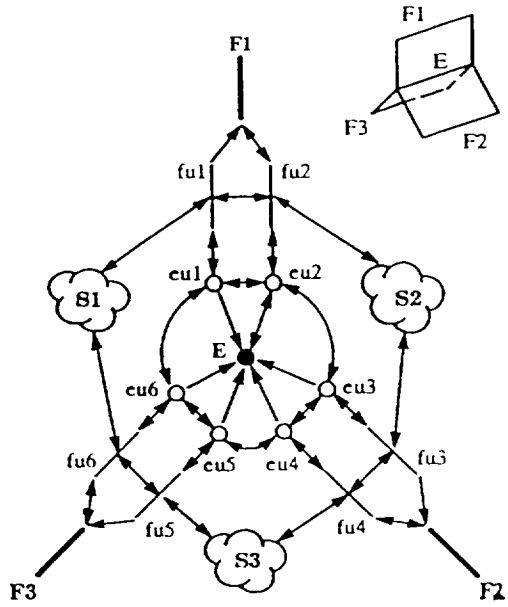

Figure 6: Topology of three faces sharing an edge.

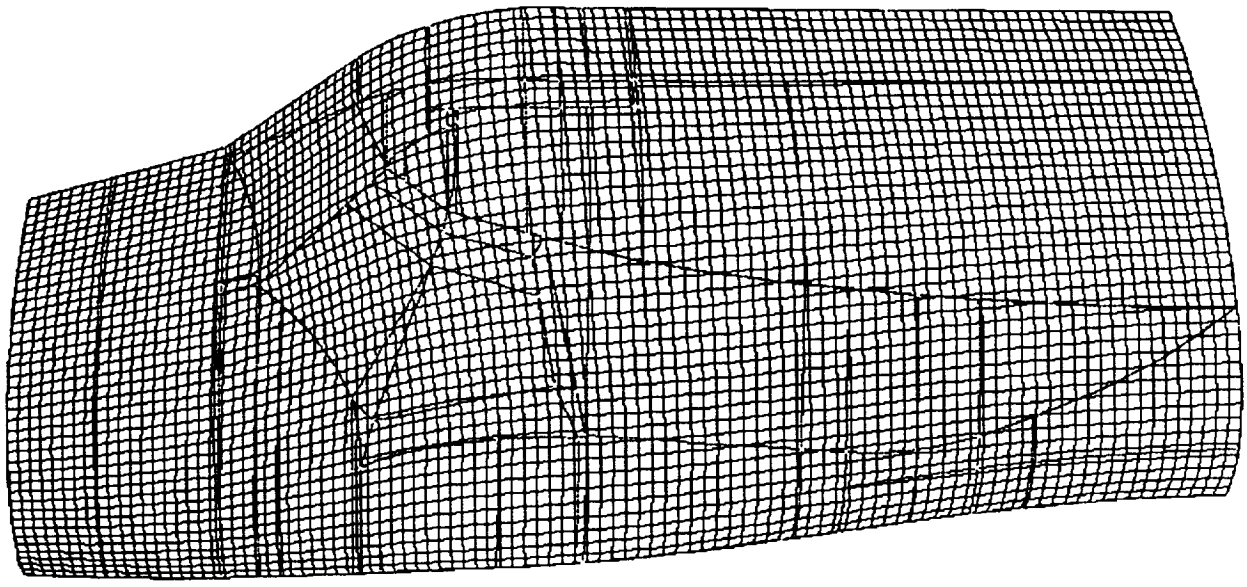

Figure 7: 101x41 carpet grid over network of shuttle surfaces.

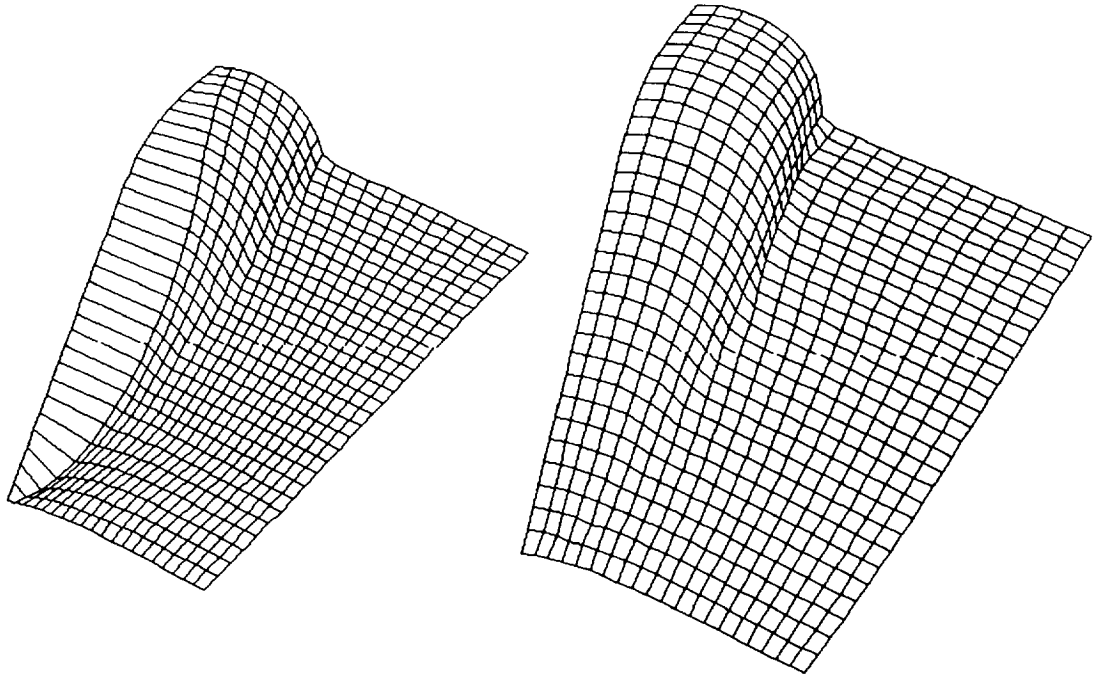

Figure 8: Original grid and algebraic adaption on canopy/fuselage configuration

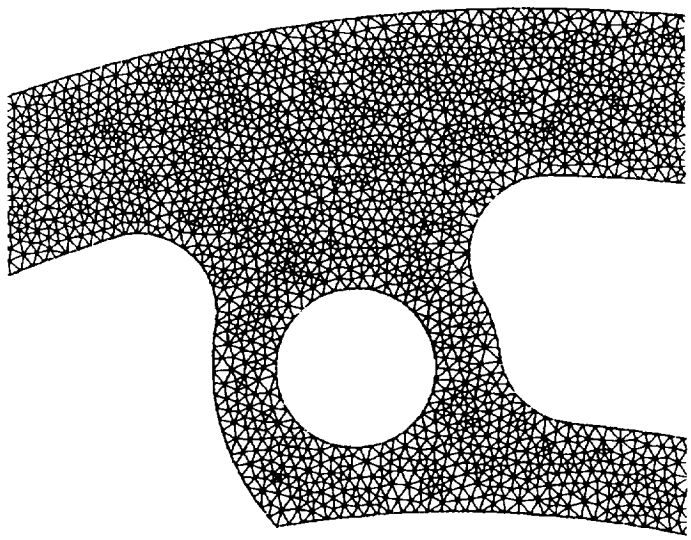

Figure 9: Delaunay unstructured grid on gasket.

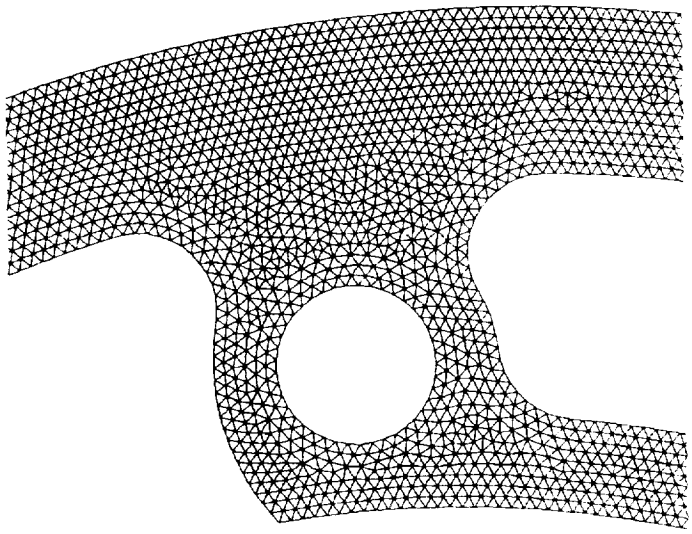

Figure 10: Advancing front unstructured grid on gasket.

Figure 11: Unstructured pointcloud on inlet configuration.