Engineering with Computers (2001) 17: 28–38

Engineering with Computers

Topology Abstraction of Surface Models for Three-Dimensional Grid Generation

Y. Zheng, N. P. Weatherill and O. Hassan

Department of Civil Engineering, University of Wales Swansea, Swansea, SA2 8PP, UK

Abstract. *Surface grid generation and the subsequent volume grid generation is the key to unstructured grid-based computational simulation. The baseline entities of the surface models under consideration for use with the proposed surface grid generator are curves and surfaces. There is a necessity to establish a topology relation between the curves and surfaces, prior to a surface gridding process. The present paper addresses issues related to this topology abstraction. Effort has also been made to generally discuss how to bridge the gap between CAD modelling and surface gridding. The proposed procedures have been incorporated into an Interactive Geometry Utility Environment (IGUE). The IGUE is a sub-environment of a Parallel Simulation User Environment (PSUE), which has been developed for unstructured grid-based computational simulation. Arbitrary computer application software can be integrated into the environment to provide a multi-disciplinary engineering analysis capability within one unified computational framework. Examples of computational applications have been included in the present paper, to demonstrate the use of the PSUE and geometry preparation procedure with an emphasis of topology abstraction.*

Keywords. Computational simulation; Geometry modelling; Grid generation; Surface model; Topology abstraction; User environment

1. Introduction

Over the past decade, significant progress has been made in the development of algorithms for the construction of unstructured grids of triangles and tetrahedra. Such advances have progressed in parallel with algorithms for problem-solving in computational science and engineering, such as computational fluid dynamics. Traditionally, the generation of grids has been an intricate process, in which data files are manually constructed and manipulated, and control files edited. In practice, the time consuming aspects of the generation process are exacerbated, since several grids are usually generated prior to the analysis cycle. Often, geometries are defined with a combination of formats, including analytically and numerically defined components and, in some cases, the individual components may not fit together smoothly, resulting in either overlaps or discontinuities. The user is often faced with the requirement to become intimately familiar with the geometry, being required to have detailed knowledge of coordinate values at key parts of the configuration. Although the algorithms for the generation of grids have advanced and matured, the process of building grids is still time consuming and far from automatic.

Existing geometrical modelling systems, which deal with real-life CAD models, usually do not provide information specially required by grid generators. Many grid generators have their own requirement of additional information other than pure geometry definition, and there has not been a standard for such requirement yet. Major efforts of development and applications in the geometry modelling community and the computational simulation community have been made in their own ways.

Several commercial and semi-commercial packages, such as ANSYS, have been developed in the computational simulation community to deal with grid generation, and incorporate more and more geometry modelling capabilities. These systems are able to perform geometry operations and grid generation from their native geometry description in a seamless manner.

On the other hand, in the computational simulation community, efforts have continuously been made to develop systems in order to bridge the gap between the CAD and grid generation systems. The

Correspondence and offprint requests to: Dr Yao Zheng, NASA Glenn Research Center, MS 5–11, 21000 Brookpark Road, Cleveland, OH 44135, USA. (present address) E-mail: yao.zheng@computer.org, yao.zheng@grc.nasa.gov

capabilities of interest include: (1) bringing CAD geometries to grid generators as accurate as possible; (2) creating the necessary information for grid generators such as grid spacing requirement; (3) providing a mechanism for checking the integrity and quality of the CAD models, with respect to the requirement of grid generators; (4) creating the missing information of geometry models in the grid generation stage; and (5) providing an interactive environment to visually steer the grid generation process for complex geometry configurations.

The present work is to develop an environment for controlling unstructured grid-based multidisciplinary computational simulation. The geometry capabilities equipped are oriented to the grid generation needs. In the present paper, an attempt is to be made to address issues related to connecting CAD modelling and grid generation. Emphasis is to be given to the topology abstraction, which is of key importance to the proposed surface gridding process. Finally, examples are to be investigated against the effectiveness of such geometry preparation.

2. Parallel Simulation User Environment

Advances in computer hardware, particularly in the area of computer graphics workstations, have introduced the possibility of reducing or eliminating some of the more tedious aspects of the user intensive tasks in grid generation. The Parallel Simulation User Environment (PSUE) [1–3] has been developed for pre- and post-processing for unstructured grid-based computational simulation. The attractive aspect of the unstructured grid technology is the ease and speed with which complex geometrical configurations can be treated [4–6]. Arbitrary computer application software can be integrated into the environment to provide a multidisciplinary engineering analysis capability within one unified computational framework. The PSUE harnesses state-of-the-art high performance computing and networking to provide a software environment for a diverse range of application in computational field simulation.

The PSUE consists of the following generic modules: geometry builder, geometry repair, grid generation/analyser, data analysis, database, numerical libraries, computing platforms, and system tools. Applications can be coupled into the PSUE as ‘user’ applications. A general schematic of these modules is shown in Fig. 1. The PSUE consists of a main process and sub-processes of the modules integrated. Figure 2 is a snapshot of the main session of the PSUE during a run time, in which the main PSUE manages the memory and controls individual subprocesses. Five geometry builders are opened with three iconisied, while two grid generators and one grid analyser are invoked, all iconisied.

The PSUE provides an enhanced capability for complex and multiple problem definition. Developments aim at integrating CAD and grid generating capabilities, addressing end-user requirements and data formats. This will include the provision of tools to correct invalid geometries, a graphics environment for guidance through the grid generation process with visual validation of each step, and robust and computationally efficient unstructured grid generation modules. The central functionality of the PSUE is grid generation, and the geometry manipulation capability is primary to the PSUE. Moreover, the associated geometry handling modules form a subenvironment, which is referred to as an Interactive Geometry Utility Environment (IGUE) [7]. This subenvironment essentially consists of a geometry builder and a grid generation interface. The underpinning data structure of this geometry builder is based on Non-Manifold topology.

The IGUE provides elementary CAD capabilities including creation of points, curves and surfaces, and clipboard operations such as copy, paste, mirror and transformation. The IGUE enables the user to build simple geometries with ease by using geometry templates. The corresponding data format is based on CAD input and utilises IGES geometry files, although specialised data formats are also supported. The essential use of this capability is to build simple shapes, such as outer boundaries to close domains,

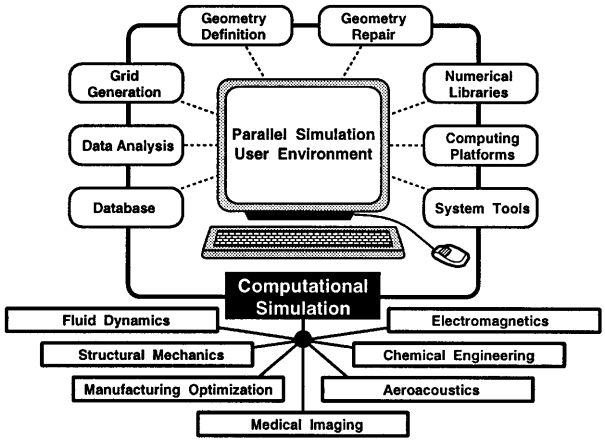

Fig. 1. A general schematic of the Parallel Simulation User Environment (PSUE).

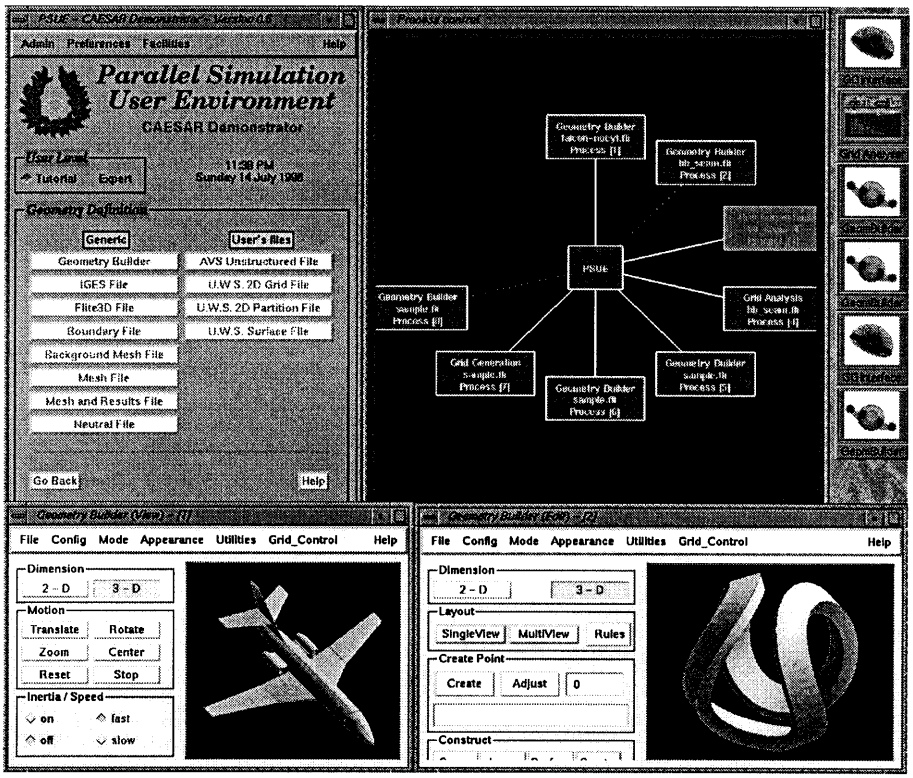

Fig. 2. A selection of windows from the PSUE driving a typical user session.

and to provide the functionality to construct background grids and grid source data which are used to control the grid generation process.

The IGUE has been equipped with geometry validation procedures, which provide checking of validity and completeness of a surface model, and guidance of correcting the geometry. Of importance is the topology abstraction functionality, which is embedded in a GUI panel to provide automatic or manual specification of topology relation between support surfaces and trimming curves in terms of loops attaching to the surfaces. The topology abstraction functionality is the main theme of the present paper.

The grid generation interface, under the IGUE, is a common framework for all the grid generators integrated into this environment. The grid generation interface is initialised from the main PSUE or from the geometry builder. Under the grid generation interface, the data preparation facility can invoke the geometry builder for geometry importing or building, and background grid and source definition setup. It also allows for the geometry repair module to be initialised.

The grid generation interface allows the user to set up grid types, parameters, and file manipulation. The parameters used in grid generation differ according to the different grid types to be created and they control the grid generation process. Within the interface, there are two computer platform options for the user to choose: local machine and remote machine. For the remote machine option, a remote access tool is invoked. Additionally, a parallel machine can be setup by initiating parallel tools from the computing platform module. In this case, parallel grid generation can be executed with monitoring on a local host.

3. Surface Models for Grid Generation

The grid generators connected within this environment are of general purpose. They are able to generate surface and volume grids as well as planar isotropic, anisotropic and hybrid grids. The geometries for two-dimensional grid generation consist of curves, for surface gridding they are made up of surfaces and loops, and volume grids are based on their corresponding surface grids. In the present paper, only three-dimensional geometries are considered. As a volumetric grid is based on a corresponding surface grid, the volume grid is almost always possible as soon as the surface grid is closed

surrounding a volumetric domain. Therefore, the main activity associated with the geometry preparation is to define a valid surface model. Hence, attention is only given to model description for surface gridding in the present paper.

Nowadays, real-life CAD models are represented in NURBS format [8], thanks to the excellent mathematical and algorithmic properties, and the successful industrial applications of NURBS. However, many grid generators usually deal with geometries in forms of lower order, such as bi-cubic surfaces, due to historical reasons of the computational simulation community. The current version of grid generators integrated in the PSUE treat geometries represented by bi-cubic surfaces and cubic splines, although they could be extended straightforwardly to deal with NURBS entities. Currently, within the IGUE, all the surfaces are treated as bi-cubic surfaces, and the curves are defined as cubic splines. Though the conversion from NURBS to cubic splines is non-trivial, and not always possible due to the facts such as tolerancing issues, we describe surfaces models and address the associated topology abstraction in a form of cubic spline without loss of generality.

Usually, a surface model does not consist of complete surfaces; there are intersections between surfaces, and ruled surfaces. Therefore, the geometry definition for gridding includes support surfaces, and loops (that is, sets of curves) with the appropriate specification of regions to be gridded. Figure 3 illustrates a support surface, loops, and the corresponding grid. It can be seen that each region is defined by one loop of curves, and is embedded by a support surface to represent its interior.

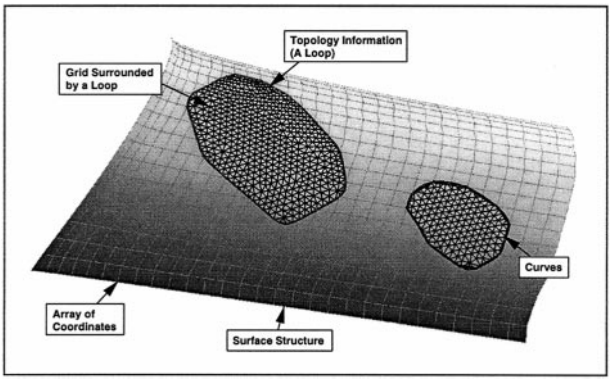

Fig. 3. A support surface, loops and the corresponding grid.

3.1. Cubic Splines

As a constitutive entry of a loop, a curve is defined in the form of cubic spline. The property of cubic order ensures the compatibility between this kind of splines and the surface patches describing boundary surfaces to be mentioned in the following sections.

In three dimensions, the three coordinate components along a curve are considered to be independent of each other, and cubic spline interpolation is used, respectively. Without loss of generality, let us consider an interpolation in a two-dimensional plane. Given a set of n points (x_i, y_i) $(i = 1, \ldots, n)$ with assumption of $x_1 < x_2 < \ldots < x_n$. If function $S(x)$ satisfies the following three conditions, then it is referred to as a cubic spline function passing the given n points:

(1) $S(x_i) = y_i$ $(i = 1, \ldots, n)$;
(2) $S(x)$ is a 3-degree polynomial on an interval $[x_i, x_{i+1}]$ $(i = 1, \ldots, n–1)$;
(3) There exist continuous first and second derivatives of $S(x)$ on interval $[x_1, x_n]$.

It is well-known that function $S(x)$ exists and is unique if one of the following boundary conditions holds:

(a) The first derivative of $S(x)$ is given at both end points of interval $[x_1, x_n]$, that is, $S'(x_1) = a_1$ and $S'(x_n) = a_n$, where a_1 and a_n are knowns.
(b) The second derivative of $S(x)$ is zero at both end points of interval $[x_1, x_n]$, that is, $S''(x_1) = S''(x_n) = 0$.
(c) The function $S(x)$ is periodical, and $S(x_1) = S(x_n)$, $S'(x_1) = S'(x_n)$ and $S''(x_1) = S''(x_n)$ hold.

In the current application, condition (b) is assumed. For use of this interpolation formulation, it is reasonable to assume $x_i = i$ $(i = 1, \ldots, n)$. During the point generation for each curve, arc length and spacing control source have been taken into account. The separation (spacing) between two generated points is measured in terms of arc length along the aforementioned spline. The desired spacing is described by the background grid and control sources, such as point, line and triangular sources.

3.2. Bi-Cubic Surfaces

An arbitrary surface is usually defined using a finite number of points. Within this geometry utility environment, the underlying surface definition is given by means of Coons patches, for geometry operations such as surface reconstruction, surface-

surface intersection, and even the associated surface grid generation. The Coons patch technique uses the given data to construct a numerical model of the surface so that any point on the surface may be obtained in terms of two parameters. The Coons patch uses continuity of slopes of the surface as well as the point data. Hence, this formulation results in a smooth representation of a three-dimensional surface.

The Coons patch technique is a method of surface representation which requires a surface mesh of quadrilaterals and for each patch the coordinates of the surrounding points are used to produce an equivalent parametric element in a bi-cubic form. Using the parametric patch, the problem is reduced from three dimensions to two dimensions, and the Cartesian coordinates anywhere on the surface may be calculated using the parametric patches.

The bi-cubic patch of parametric coordinates (ξ, η) is given by

$$\mathbf{X}(\xi,\eta) = \mathbf{F}(\eta)\mathbf{A}\mathbf{F}(\xi)^T \quad (0 < \xi < 1 \text{ and } 0 < \eta < 1) \tag{1}$$

where $\mathbf{X}(\xi, \eta) = [x(\xi, \eta), y(\xi, \eta), z(\xi, \eta)]^T$ and $\mathbf{F}(u) = (F_1(u), F_2(u), F_3(u), F_4(u))$. Each component of $\mathbf{F}(u)$ has cubic poperties, which ensures continuity and smoothness throughout the surface. Actually, $\mathbf{F}(u)$ takes the form of Hermite blending functions

$$\begin{cases} F_1(u) = (1 + 2u)(u - 1)^2 \\ F_2(u) = u^2(3 - 2u) \\ F_3(u) = u(1 - u)^2 \\ F_4(u) = u^2(u - 1) \end{cases} \tag{2}$$

The 4×4 matrix $\mathbf{A}$ containing coordinates, tangents and twists on a patch can be expressed as

$$\mathbf{A} = \begin{bmatrix} \mathbf{X}(0,0) & \mathbf{X}(0,1) & \mathbf{X}_\eta(0,0) & \mathbf{X}_\eta(0,1) \\ \mathbf{X}(1,0) & \mathbf{X}(1,1) & \mathbf{X}_\eta(1,0) & \mathbf{X}_\eta(1,1) \\ \mathbf{X}_\xi(0,0) & \mathbf{X}_\xi(0,1) & \mathbf{X}_{\xi\eta}(0,0) & \mathbf{X}_{\xi\eta}(0,1) \\ \mathbf{X}_\xi(1,0) & \mathbf{X}_\xi(1,1) & \mathbf{X}_{\xi\eta}(1,0) & \mathbf{X}_{\xi\eta}(1,1) \end{bmatrix} \tag{3}$$

where $\mathbf{X} = (x, y, z)^T$, $\mathbf{X}_\xi = \partial X/\partial\xi$, $\mathbf{X}_\eta = \partial X/\partial\eta$ and $\mathbf{X}_{\xi\eta} = \partial^2 X/\partial\xi\partial\eta$, and the coordinates ($\xi$, η) take values of 1 or 0, depending upon the parametric position. Such a patch is often referred to as a tensor-product surface.

This (ξ, η) coordinate system is for one patch only with ($0 < \xi < 1$ and $0 < \eta < 1$). An overall parameter coordinate system can be introduced for a surface as composite patches, and is denoted as system (u, v). Without loss of generality, it is convenient to assume that u and v take integral values at nodal points for the overall surface. Referring to Fig. 4, the components of $\mathbf{A}$ can be rewritten in terms of the parametric coordinates as

$$\mathbf{X}_\xi(u_{k,l},v_{k,l}) = f_{k,l}\frac{\partial \mathbf{X}}{\partial \xi_{k,l}} = f_{k,l}\mathbf{X}_u(u_{k,l},v_{k,l}) \tag{4}$$

$$\mathbf{X}_\eta(u_{k,l},v_{k,l}) = g_{k,l}\mathbf{X}_v(u_{k,l},v_{k,l}) \tag{5}$$

$$\mathbf{X}_{\xi\eta}(u_{k,l},v_{k,l}) = f_{k,l}g_{k,l}\frac{\partial^2 \mathbf{X}}{\partial \xi_{k,l}\partial \eta_{k,l}} = f_{k,l}g_{k,l}\mathbf{X}_{uv}(u_{k,l},v_{k,l}) \tag{6}$$

where $u_{k,l}$ and $v_{k,l}$ are parametric coordinates, and $f_{k,l}$ and $g_{k,l}$ are the curvilinear arc lengths along the u and v directions, respectively.

Using notations given in Fig. 4, Eq. (1) can be written as

$$\mathbf{X}(u,v) = \mathbf{F}(\eta)\mathbf{B}\mathbf{F}(\xi)^T \tag{7}$$

with the $\mathbf{B}$ matrix consisting of

$$\mathbf{B} = \begin{bmatrix} \mathbf{X}^{k,l} & \mathbf{X}^{k+1,l} & f_{k,l}\mathbf{X}_u^{k,l} & f_{k+1,l}\mathbf{X}_u^{k+1,l} \\ \mathbf{X}^{k,l+1} & \mathbf{X}^{k+1,l+1} & f_{k,l+1}\mathbf{X}_u^{k,l+1} & f_{k+1,l+1}\mathbf{X}_u^{k+1,l+1} \\ g_{k,l}\mathbf{X}_v^{k,l} & g_{k+1,l}\mathbf{X}_v^{k+1,l} & (fg)_{k,l}\mathbf{X}_{uv}^{k,l} & (fg)_{k+1,l}\mathbf{X}_{uv}^{k+1,l} \\ g_{k,l+1}\mathbf{X}_v^{k,l+1} & g_{k+1,l+1}\mathbf{X}_v^{k+1,l+1} & (fg)_{k,l+1}\mathbf{X}_{uv}^{k,l+1} & (fg)_{k+1,l+1}\mathbf{X}_{uv}^{k+1,l+1}) \end{bmatrix} \tag{8}$$

where $\mathbf{X}_u = \partial\mathbf{X}/\partial u$, $\mathbf{X}_v = \partial\mathbf{X}/\partial v$, $\mathbf{X}_{uv} = \partial^2\mathbf{X}/\partial u\partial v$, and the k and l subscripts indicate the position on the Cartesian grid.

3.3. Surface Grid Generation

Basically, the surface grid generation consists of two steps. It first generates points on curves, that is, on loops. Then each individual region surrounded by a loop is dealt with in a sequence for the creation of triangular elements. As the region is supported

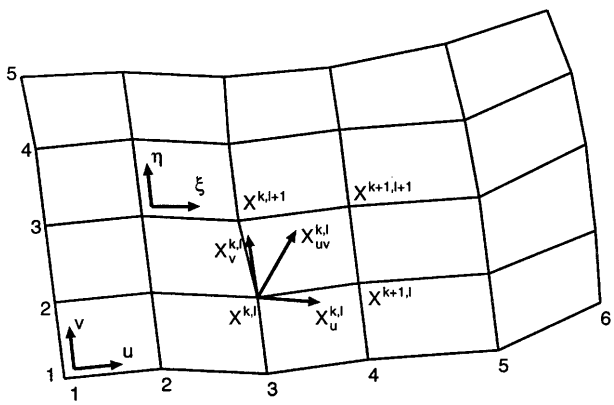

Fig. 4. Notations of parametric coordinate system (ξ, η) and (u, v) for a surface consisting of patches.

by a bi-cubic surface, the surface grid generation is performed within the corresponding parametric space. The front advancing method [9,10] has been employed in the surface gridding. Although the coordinates in parametric space are used to describe a spatial point, its physical coordinates in three dimensions are also used as a dual representation. Moreover, the controls such as point spacing are performed in the physical space.

A question should be raised regarding how to calculate the parameter coordinates of a point on the surface given by its physical coordinates, when the initial front is to be generated for surface gridding in a parametric plane. This question is to be answered later when the proximity of a point and a surface is considered.

3.4. Underlying Geometry Data

Basically, a support surface can be specified by given coordinates of certain points on the surface, and a curve also can be given by specifying associated points. A loop consists of a group of curves, and further, it is attached to a surface. This attachment is referred to as a topology relation, which should be established prior to surface grid generation. This information can be imported together with the basic surface and curve definition from external geometry modellers, in addition, it can be automatically generated within this environment or manually by utilizing the topology relation panel. For the automatic specification, an algorithm of surface topology abstraction is involved.

The present environment has been developed with an orientation towards grid generation for computational science and engineering, rather than as a generic geometric modelling tool. Surface description can be built externally and then imported into this environment. An example of data exchange format employed is IGES. Apart from imported geometry data, geometric models can be built within this environment. A simple example of this type of utilization is building an outer boundary for a domain. However, the equipped functionality enables the user to modify and validate the models, to enhance local features, and even to specify sophisticated controls to grid generation in various ways.

3.5. Utilization of CAD Data

For real industrial applications, CAD systems are generally utilized. To bridge the gap between CAD systems and the IGUE for use of the grid generation technique, CAD data exchange is a must. However, such a data exchange is only part of the solution. Data transfer only provides the exchange of geometry entities as it is. A CAD model usually requires significant modification, even if it is a clean model, to get it into a form suitable for grid generation and then for computational simulation.

For a surface gridding algorithm oriented to bi-cubic surfaces, all other types of surfaces from the CAD model should be converted into bi-cubic class. Curves other than cubic splines should be converted as well. This task sometimes causes inconsistence of accuracy and tolerance between the original and the resulting entities.

To upgrade a drawing based CAD model into a surface model may well be an ill-defined process [11]. For instance, a wireframe model lacks information about support surfaces. In such a case, the included surface reconstruction panel can be used to construct a surface to fill the region bounded by the wires, on the basis of these wires and adjacent surfaces. On the other hand, for regions, of which the bounding curves are not yet defined, the surfaces can be used to figure out the required curves by means of using a surface-surface intersection panel.

While the data exchange can be fully automatic with IGES, the repair and modification process are typically subjective, application specific, and require engineering judgment. Therefore, they are not readily automated. In the manufacturing design, engineering analysts are increasing being pressed to reuse or to upgrade CAD geometry in the cycle of grid generation and computational simulation. A routine procedure to automatize such repair and modification is an ultimate task.

4. Topology Abstraction of Surface Models

As mentioned previously, a region on a support surface to be gridded is described in terms of a loop. Therefore, the loop consisting of curves should be attached to a particular support surface. This process is referred to as topology abstraction. A loop generator is provided to automatize loop construction and topology abstraction (that is, assignment of relations between loops and surfaces). Before discussing further the automatic loop generation, basics on the proximity of a point and a surface is considered.

4.1. Proximity of a Point and a Surface

This distance can be evaluated either by numerical minimization or when possible by computing the implicit functional definition of the surface, that is, $Ax + By + Cz = 0$ for a plane, and $F(x,y,z) = 0$ for a general surface. Considering the trade-off between computing cost and resulting accuracy, we determine this distance by using the following iterative procedure, taking into account the native description of the surfaces in the current systems. First check the distance between the given point P and each of the corners of the surface. If these distances are all big enough, then try to find the distance between point P and each of the four edges of the surface. This can be performed by means of line search minimization. If the distances between point P and all the edges are big enough, then use minimization procedure to find the minimal distance of point P to the interior of the surface. This procedure can be performed by a number of trials of combinations of line search minimization and search direction adjustment in the parametric plane.

In certain circumstances, due to numerical precision issues, alternative approaches may be of interest. One is named as selective splitting, which provides robustness, but is more costly. Assume the surface under consideration is represented by $n_u \times n_v$ support points. Find one of these support points, denoted as P_1, which makes a minimal distance to point P. The interval spacing of these support points in the parametric plane is denoted as l_0, and $l_0 = 1$ obviously. Now consider a square S_1 centered at point P_1 with size $\lambda_1 = n_s l_1 = 3l_0$, where n_s is the number of splitting divisions of square S_1 (n_s should be even, say $n_s = 6$), and l_1 is the corresponding interval spacing. Then find a point P_2 on this grid of $(n_s + 1) \times (n_s + 1)$ points, which makes a minimal distance to point P. Then consider a square S_2 centered at point P_2 with size $\lambda_2 = n_s l_2 = 3l_1$.

Following this way, a sequence of squares S_i centered at point P_i with size $\lambda_i = n_s l_i = 3l_{i-1}$ will be obtained until λ_i or the distance between points P_i and P is small enough. Figure 5 illustrates the method of selective splitting in the parametric plane. As $l_i = (3/n_s)l_{i-1} = \ldots = (3/n_s)^i l_0 = (3/n_s)^i$, the sequence of squares will converge to a point P_∞ ultimately, or the minimal point is one of the grid points on a square in the sequence.

It is obvious that point P can be considered to be on the surface if a surface point found makes its distance to point P small enough. This answers the question raised previously how to calculate the parameter coordinates of a point on the surface given by its physical coordinates.

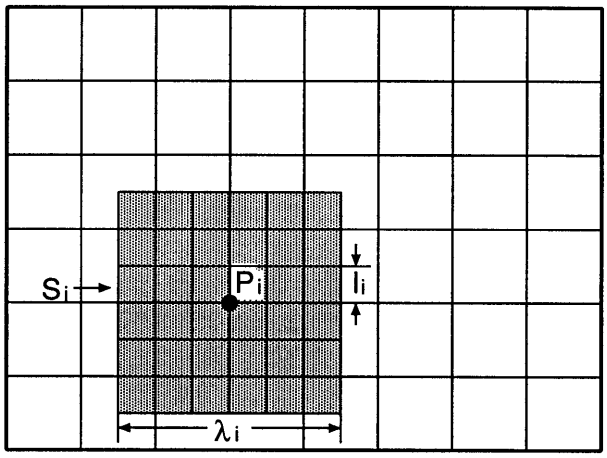

Fig. 5. Diagram for selective splitting method in the parametric plane.

4.2. Automatic Specification

If all the support points of a curve are on a surface, then the curve is considered to be proximal to the surface. The loop generation procedure first establish proximity relationship between curves and surfaces in a given model data. Then it attempts to form close loops to describe regions on the surfaces. Figure 6 is a snapshot of the topology relation panel in the geometry builder.

Figure 7 shows a sequence of four pictures. The first one is the original surface model with information on curves and surfaces. The second picture provides loop information by highlighting one loop and the associated surface, of which the correspond-

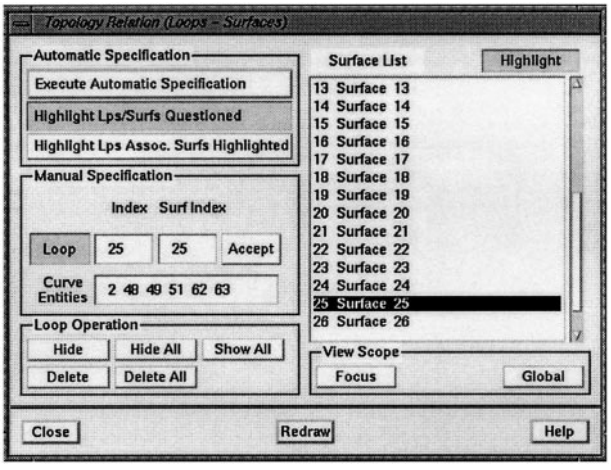

Fig. 6. Topology relation panel in the geometry builder.

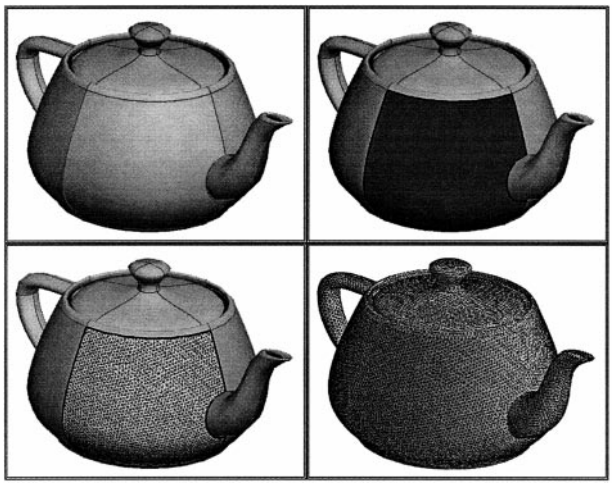

Fig. 7. Topology abstraction of the surface model of a teapot.

ing grid is given in the third picture. The last picture finally shows the whole surface grid on the surface of the teapot.

Often a CAD model comes with ambiguous information of edges. For instance, there are duplicated curves, curves partially overlapped, curves of a common location and different degree of discretization, intersecting curves without common supporting nodes, and curves which could not form a close loop to define a region (see Fig. 8). For such cases, the loop generator highlights them graphically to prompt user involvement.

4.3. Manual Specification

These ill-defined surface models require manual specification of loop information. Manual specification is also provided *via* the topology relation panel. During this specification, loops should be constructed first, then the relationship between loops and surfaces established. The visualization facility including features of hiding and showing a part of the model facilitates interactive discovery, location and fixing of the bad curves and loops.

Additionally, it is a challenging task to implement such a loop generator which works for arbitrary complete surface models, as many factors affect the robustness of the generator. Therefore, manual specification is necessary when difficulty has been encountered in the related cases.

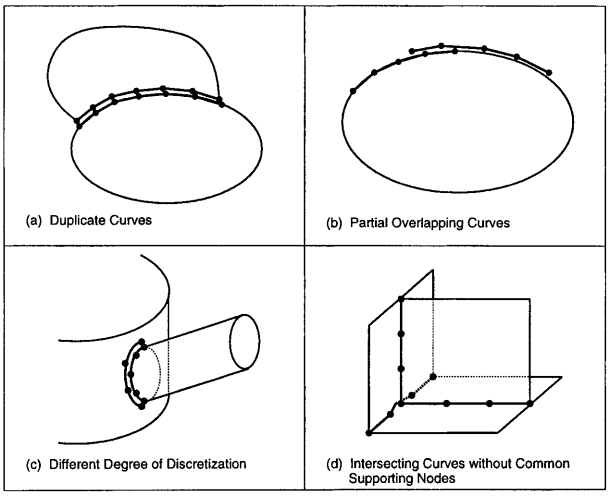

Fig. 8. Examples of invalid surface description.

5. Examples of Computational Applications

5.1. Metal Forming

Ring rolling is a versatile metal forming process for manufacturing seamless annular forgings. Figure 9 depicts schematics of a radial-axial rolling mill with guide rolls [12]. Figure 10 gives a visual representation within the IGUE of the geometry, surface and volume grids of the ring and tools during rolling, where the grids are cut to provide an internal view. The automatic loop specification procedure has been conducted to create the valid surface model in the context of region information.

5.2. Aerodynamics

An example from computational fluid dynamics has been analyzed under the PSUE with its sub-environment IGUE. The example results in the simulation of compressible flow around the Dassault Falcon executive plane which has been performed using software developed for parallel architecture computers.

The geometry definition of the aircraft is imported

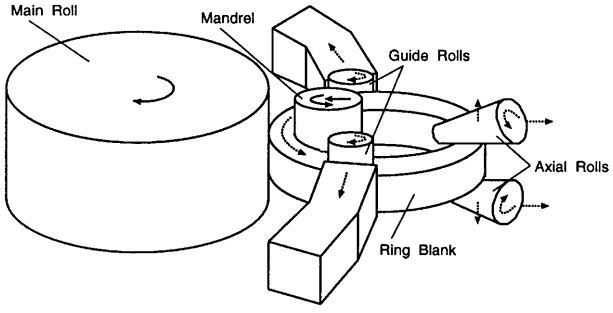

Fig. 9. Schematics of ring rolling.

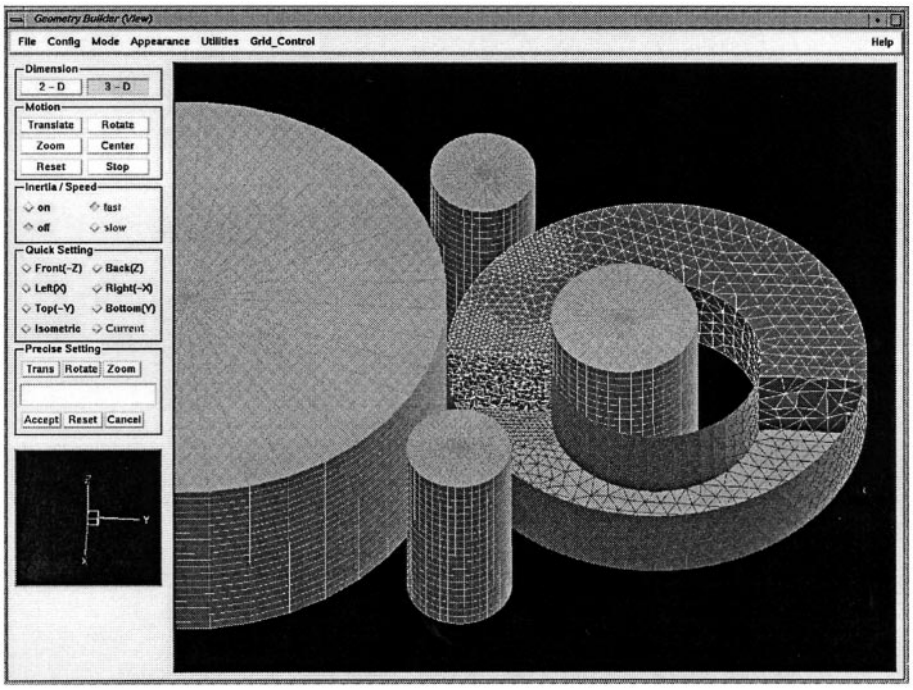

Fig. 10. Visual representation of the geometry, surface and volume grid of ring and tools during rolling.

into the IGUE, it describes half of the aircraft. Figure 11 shows the configuration of the whole aircraft with a half mirrored from the original. It has been seen from Fig. 11, that the imported data consists of support surfaces and curves. Subsequently topology abstraction has been performed to create loops attaching to surfaces, which describe regions to be gridded on the surfaces. The initial geometry definition does not include the symmetry plane and farfield boundaries necessary for generating a grid suitable for flow simulation. However, the geometry templates available in the IGUE are capable of creating the appropriate farfield boundary, as either a hemisphere or half cylinder, which is adjacent to the symmetry plane created for the geometry.

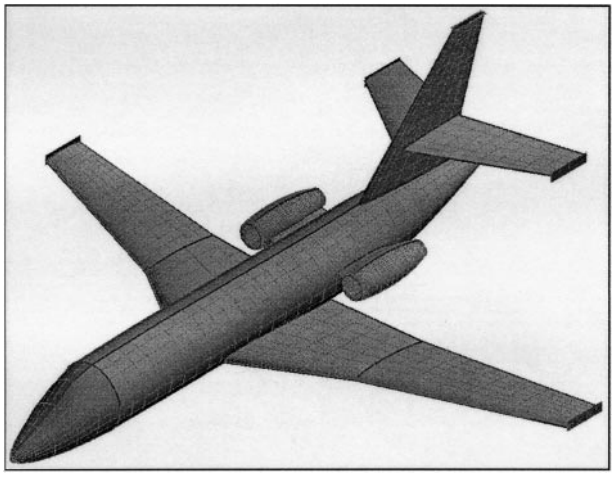

Fig. 11. Surface model of the Falcon aircraft.

Then the completed geometry is passed through a surface topology extraction algorithm, which uses the information of curves and surfaces to establish loops and to check the geometry definition for completeness. In case for a dirty model geometry repair and sometimes even manual topology specification is desired.

Based on the surface model, the grid generation module creates a surface grid and then a volume grid. Figure 12 illustrates a part of the surface grid and a part of volume grid for the aircraft. The visualization of the volume grid is shown in the form of the triangles on the surface of the volume elements cut by a user specified cutting cylinder. The flow simulation has been carried out using a CFD code. A CFD solution has been obtained, as shown in Fig. 13, which shows the contours of pressure on the surface of the aeroplane.

6. Conclusions

The PSUE provides an enhanced capability for complex and multiple problem definition for unstructured grid-based computational simulation. It features vis-

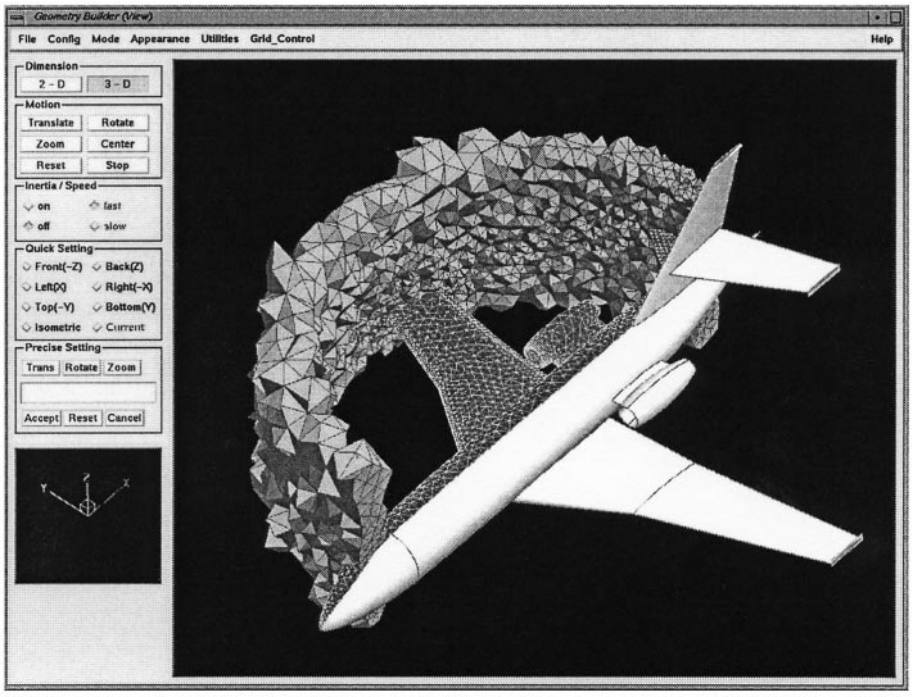

Fig. 12. Visual representation of the geometry, surface and volume grid (with cutting plane) for the Falcon aircraft.

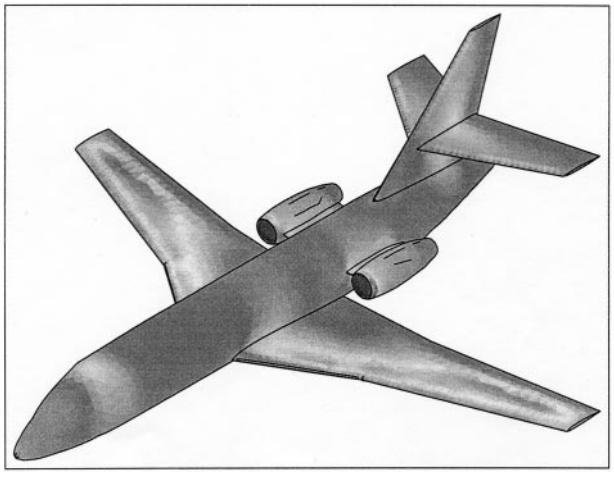

Fig. 13. Pressure contours for Falcon aircraft flow simulation.

ual steering of grid generation, which includes the provision of tools to correct invalid geometries, a graphics environment for guidance through the grid generation process with visual validation of each step, and robust and computationally efficient unstructured grid generation modules. The essential geometric capability of the PSUE is provided through its sub-environment – IGUE.

For real computational applications, geometry data preparation is time consuming and far from automatic. To bridge the gap between CAD modelling and surface gridding, related issues have been identified, and efforts have been made to develop algorithms for automatizing this process. For arbitrary geometries, fully automatic procedures such as topology abstraction are challenging to implement, all the efforts are steps towards automation.

Acknowledgements

The authors are grateful to the European Union for the financial support through the ESPRIT CAESAR project. The authors would like to thank the other members of the PSUE group for their contributions. The development of the IGUE has benefited from collaboration with the other key partners involved in the CAESAR project, in particular IPK (Berlin, Germany), Sowerby Research Centre, British Aerospace (UK), DASA (Germany), Odense (Denmark) and BASF (Germany). Their contribution is also gratefully acknowledged. The authors would like to thank the reviewers for their constructive comments.

References

1. Marchant, M. J., Weatherill, N. P., Turner-Smith, E., Zheng, Y., Sotirakos, M. (1996) A parallel simulation user environment for computational engineering. In: Soni, B. K., Thompson, J. F., Haeuser, J., Eiseman, P. (Editors), Numerical Grid Generation in Computational Field Simulations (Proceedings of the 5th

International Conference), Vol 1, Mississippi, USA. pp. 741–751
2. Weatherill, N. P., Marchant, M. J., Turner-Smith, E., Zheng, Y., Sotirakos, M. (1996) The design of a graphical user environment for multi-disciplinary computational engineering. ECCOMAS'96 (2nd Conference on Numerical Methods, and 3rd Conference on Computational Fluid Dynamics), Paris, France
3. Zheng, Y., Weatherill, N. P., Turner-Smith, E. A., Sotirakos, M. I., Marchant, M. J., and Hassan, O. (2000) Visual steering of grid generation in a parallel simulation user environment. In: Houstis, E. N., Rice, J. R., Gallopoulos, E., Bramley, R. (Editors), Enabling Technologies for Computational Science: Frameworks, Middleware and Environments, Kluwer Academic, Boston, pp. 339–349
4. Weatherill, N. P. (1988) A method for generating irregular computational grids in multiply connected planar domains. Int J Numerical Methods in Fluids, 8, 181–197
5. Weatherill, N. P., Hassan, O. (1994) Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. Int J Numerical Methods in Engineering, 37, 2005–2039
6. Weatherill, N. P., Marchant, M. J., Hassan, O., Marcum, D.L. (1994) Grid adaptation using a distribution of sources applied to inviscid compressible flow simulations. Int J Numerical Methods in Fluids, 19, 739–764
7. Zheng, Y., Weatherill, N. P., Turner-Smith, E. A. (2000) An interactive geometry utility environment for multi-disciplinary computational engineering. Int J Numerical Methods in Engineering (submitted)
8. Piegl, L., Tiller, W. (1995) The NURBS Book, Berlin, Springer-Verlag
9. Peraire, J., Perio, J., Formaggia, L., Morgan, K., Zienkiewicz, O. C. (1988) Finite element Euler computations in three dimensions. Int J Numerical Methods in Engineering, 26; 2135–2159
10. George, P. L., Seveno, E. (1994) The advancing-front mesh generation method revisited. Int J Numerical Methods in Engineering, 37(21), 3605–3619
11. Butlin, G., Stops, C. (1996) CAD data repair. Proceedings 5th International Meshing Roundtable, Pennsylvania, USA, pp. 7–12
12. Hu, Z. M., Pillinger, I., Hartley, P., McKenzie, S., Spence, P. J. (1995) Thermo-plastic finite element modelling of the rolling of a hot titanium ring. In: Shen, S. -F., Dawson, P. R. (Editors), Simulation of Materials Processing: Theory, Methods and Applications, Proc 5th Int Conf on Numerical Methods in Industrial Forming Processes (NUMIFORM 95), Balkema, The Netherlands, pp. 941–946