AIAA 2002-3191

BEST PRACTICES IN OVERSET GRID GENERATION

William M. Chan *
NASA Ames Research Center
Moffett Field, California 94035

Reynaldo J. Gomez III †
NASA Johnson Space Center
Houston, Texas 77058

Stuart E. Rogers ‡
NASA Ames Research Center
Moffett Field, California 94035

Pieter G. Buning §
NASA Langley Research Center
Hampton, Virginia 23681

Abstract

Grid generation for overset grids on complex geometry can be divided into four main steps: geometry processing, surface grid generation, volume grid generation and domain connectivity. For each of these steps, the procedures currently practiced by experienced users are described. Typical problems encountered are also highlighted and discussed. Most of the guidelines are derived from experience on a variety of problems including space launch and return vehicles, subsonic transports with propulsion and high lift devices, supersonic vehicles, rotorcraft vehicles, and turbomachinery.

1. Introduction

One of the first applications of overset grid techniques[1] to complex geometries took place in 1988 on the Integrated Space Shuttle Launch Vehicle (SSLV). The first computations were performed on a highly simplified model with less than a million grid points.[2] Even for such a simple model, grid generation consumed almost twelve man months. With no tools available to build computational grids from CAD models, special programs had to be written to construct some of the grids from CAD drawings. It was clear that general purpose grid generation tools were urgently needed to improve the process.

The second generation of SSLV grids[3,4] was built on a CAD model with significantly higher geometric fidelity and contained about 16 million points. It took approximately two man years to create using general tools - ICEMCFD[5] for geometry processing and surface grid creation, HYPGEN[6,7] for volume grid generation and version 4 of PEGASUS[8] for domain connectivity. Concepts on the methods and tools needed to speed up grid generation were beginning to emerge, including the initial hyperbolic surface grid generation development[9] and collar grid applications.[10] Geometry clean-up in the CAD package alone devoured one of the two total man years. Surface grids were created using algebraic and elliptic methods on the CAD and required heavy user's effort. The use of hyperbolic methods brought significant savings in volume grid generation time. It was recognized that such methods are best suited to take advantage of the flexibility of the overset grid approach. The domain connectivity input file required thousands of lines of manual input and was another place where improvements were desperately needed.

The Advanced Subsonic Technology Program played a key role in driving the next phase in overset analysis process development. The NASA IGES format[11] was introduced and employed for some of the geometry files. CAD tools such as ICEMCFD and Pro/ENGINEER were used for geometry processing. Hyperbolic surface grids were beginning to be used for some of the components. It was also important

*Computer Scientist, Senior Member, AIAA
†Aerospace Engineer, Member, AIAA
‡Aerospace Engineer, Senior Member, AIAA
§Aerospace Engineer, Associate Fellow, AIAA

AIAA 2002-3191

BEST PRACTICES IN OVERSET GRID GENERATION

William M. Chan
NASA Ames Research Center
Moffett Field, California 94035

Reynaldo J. Gomez III
NASA Johnson Space Center
Houston, Texas 77058

Stuart E. Rogers
NASA Ames Research Center
Moffett Field, California 94035

Pieter G. Buning
NASA Langley Research Center
Hampton, Virginia 23681

32nd AIAA Fluid Dynamics Conference
24-27 June, 2002 / St. Louis, Missouri

to be able to duplicate the entire process quickly for slight design changes. This brought about the development of grid-generation scripts where important grid attributes were parameterized. Advances in various parts of the process resulted in a grid-generation time of 48 man days for a 22 million point grid system on a complete subsonic transport airplane with propulsion and high-lift devices.[12] More than two-thirds of the grid generation time was spent on domain connectivity. Valuable experience was gained on the grid quality required to attain accurate lift and drag predictions.

Key drivers in the further development of overset tools in the late 1990's came from the DOD High Performance Computing and Modernization Program and NASA's High Performance Computing and Communications Program. These programs supported the development of a comprehensive graphical user interface OVERGRID[13] which serves as a unified environment for performing most of the grid generation tasks. Major upgrades also occurred for two domain connectivity programs: version 5 of PEGASUS[14] and DCF with object X-rays.[15] Applications involving complex geometries were routinely performed starting from a CAD model. Highly complex grid systems such as the V-22 Tiltrotor, Comanche, subsonic commercial airplanes could be produced in a few weeks.

Although the effort needed to perform overset grid generation on complex geometries has been significantly reduced, state-of-the-art tools still require the user to make key decisions and to enter some inputs. This paper describes the procedures practiced by experienced users in overset grid generation today. Since the software tools and algorithms are still evolving towards more automation and robustness, more and more of the guidelines given in this paper are expected to be automated in the future. In any case, a new overset-grid user should be able to follow the steps described in this paper to generate high quality overset grids. Advanced users may find this paper to be a useful collection of the current best practices. The principles discussed here are independent of the software tools used, but examples will be given on tools that try to follow these principles.

The grid-generation process can be decomposed into four main steps: geometry processing, surface-grid generation, volume-grid generation and domain connectivity. Details are presented in Sections 2 to 5. The overall strategy is to first create a clean surface geometry definition on which overlapping surface grids can be generated. With sufficient overlap between surface grids, the volume grids can be created easily with hyperbolic marching out to a fixed distance from the surface. The distance is chosen such that the outer boundaries of the near-body volume grids are well clear of the boundary layer. The near-body volume grids are then embedded inside off-body Cartesian grids that extend to the far field. Since grid generation strategies may differ depending on the ultimate goal of the simulation, issues related to the different goals will be discussed. Special attention will be given to the treatment of airfoil shapes since they occur frequently in aerospace and marine applications. Best practices on the overall process are given in Section 6, and concluding remarks are given in Section 7.

2. Geometry Processing

Most grid generation techniques require an underlying geometry definition that represents the boundaries of the objects about which computations are to be performed. Input geometries can range from simple analytic surfaces to computer-aided-design (CAD) generated solid models. These geometry definitions are usually given to the computational fluid dynamics (CFD) user in a variety of formats (Section 2.1). Unfortunately, most input geometries are usually inappropriate in some way and require some work before they can be used in surface grid generation (Section 2.2). This step can be a very significant bottleneck in the overall CFD process. Some experts estimate that geometry repair can consume up to 80% of the total time required for grid generation.[16] Some examples of geometry repair methods are given in Section 2.3.

2.1. Geometry Formats

Typical geometry database formats fall into two categories (Fig. 1):

Analytic databases

These usually contain CAD entities such as Non-Uniform Rational B-Spline (NURBS) curves and surfaces. Typical file formats include Initial Graphics Exchange Specification (IGES), Standard for the Exchange of Product Model Data (STEP), and other native CAD formats from individual vendors such as Pro/ENGINEER.

Discrete databases

These usually consist of a set of vertices with prescribed connectivity to form a collection of quadrilaterals or triangles. A bilinear representation of the surface based on these shapes is usually adopted. Common examples include:

(1) Structured panel networks (multiple patches where each patch contains a rectangular array of quadrilaterals defined by a rectangular array of vertices), e.g., PLOT3D[17] format.

(2) Unstructured triangulated surfaces (unstructured collection of triangles defined by an unstructured collection of vertices), e.g., Stereolithography (STL), Drawing Exchange Format (DXF), Virtual Reality Markup Language (VRML), CART3D (*http://www.nas.nasa.gov/~aftosmis/cart3d/cart3dTriangulations.html*), FAST unstructured.[18]

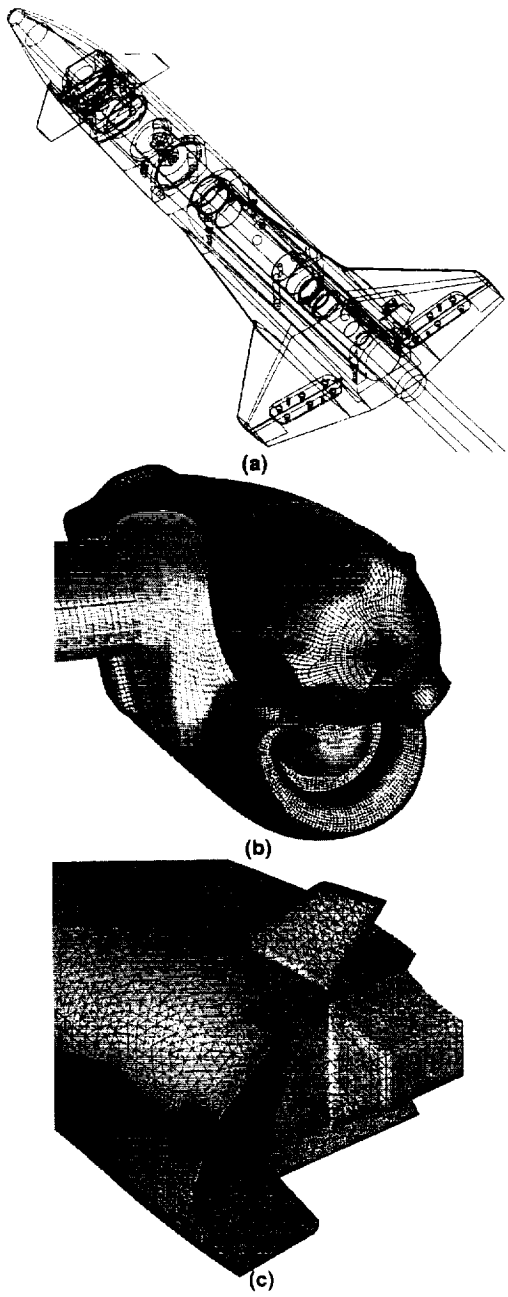

Fig. 1 Analytic and discrete geometry databases. (a) CAD surfaces (analytic). (b) Multiple panel networks (discrete). (c) Surface triangulation (discrete).

Gridgen,[19,20] OVERTURE[21,22] and SURGRD[9] all have hyperbolic surface marching capabilities. SURGRD has the most mature hyperbolic marching capability but is currently limited to marching on discrete geometry databases. Gridgen can march on CAD surfaces and a future version will add the ability to use unstructured STL and VRML formats. A future version of OVERTURE will also introduce capabilities to repair and to march on CAD surfaces.[23]

Some common overset grid generation software and supported geometry formats are given in Table 1.

Table 1.

Software	Supported Geometry Formats
Gridgen	structured and unstructured databases, IGES, native CAD import
ICEMCFD	structured and unstructured databases, IGES, direct CAD interface
OVERTURE	unstructured triangular formats, IGES
SURGRD/ OVERGRID	structured and unstructured databases: PLOT3D, CART3D, FAST

Ideally, surface grids should be generated on the most accurate representation of the geometry available which is typically a CAD geometry in IGES or STEP format. The next best alternative is to use a discrete database. These databases are typically produced using standard manual grid generation tools or through in-house codes which can generate high quality panel networks relatively automatically. Often the manual generation of a discrete-structured database, including any geometry repair, can be less time consuming than the time required to repair a CAD file.

2.2. Defects and Ideal Attributes

The input geometry should accurately represent the objects being simulated. If the geometry represents an existing vehicle or wind-tunnel model, it should be compared with pictures of the as-built configuration. Ideally a geometric verification should be performed using a 3D digitizer to insure that the computed geometry matches the actual geometry. Missing gaps and protuberances can make the difference between a poor comparison and a good one. Minor geometric flaws may cause local pressure variations but may not affect integrated forces and moments. Major flaws, such as missing geometric features, can yield incorrect solutions and can have serious consequences if they are not detected.[24]

A model containing only relevant geometry should be used as the input for grid generators. Production CAD models often include internal geometric components that do not come in contact with the flowfield. Models can include wiring harnesses and fastener details that will have a minimal influence on the flowfield but would require a large number of grid points to model. Ideally these details should be removed in the original CAD system rather than in a CFD grid generator.

A single valued, watertight geometry with consistent normals can simplify grid generation. Many grid-generation techniques project points onto the underlying geometry database. Untrimmed, self-intersecting or overlapping surface geometry should be avoided

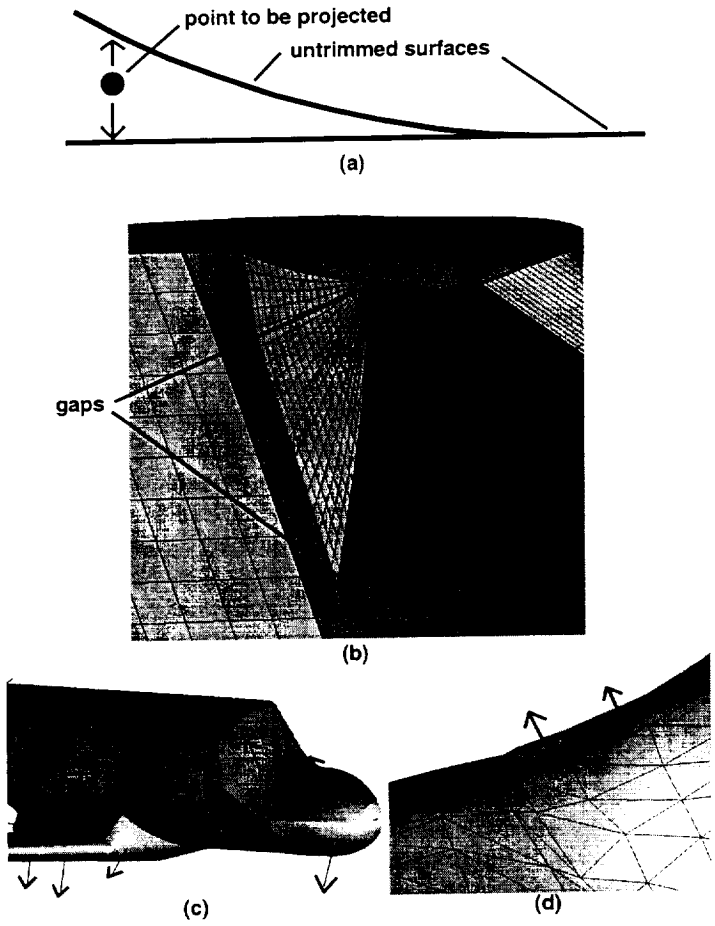

Fig. 2 Potential problems in geometry definitions. (a) Untrimmed surfaces. (b) Gaps between surface patches. (c) Surface normals on multiple panel networks. (d) Surface normals on triangulation.

since it can cause ambiguity in the underlying geometry, i.e., the projected points can land on multiple valid locations (Fig. 2a). Most grid generators can tolerate small gaps (often based on an input tolerance). But larger gaps or "leaks" in the geometry can cause grid points to project to the opposite side of the geometry (Fig. 2b). Some overset-grid-generation codes use surface normals to determine a marching direction. Surface normals on all surface patches should point consistently inwards or outwards (Fig. 2c,d), otherwise the marching direction becomes ambiguous. The geometry surfaces should be "smooth." CAD files can contain facetted surfaces, surface oscillations and poor parameterizations that may not accurately represent the actual geometry. Due to the facetted screen representations used by may CAD and grid generation programs these problems can be difficult to detect.

Discrete databases should have higher resolution in high curvature regions. Typical grid spacing on a wing leading edge is on the order of 0.1% of the local chord. This is considerably higher resolution than that found in most panel networks used by potential solvers. High curvature regions in discrete databases should have resolutions higher than the grid being generated on them. Planar regions can be represented by the coarsest resolution required to close gaps in adjacent panel networks.

2.3. Geometry Repair

A number of authors have attempted to document the wide range of problems that are encountered with poor quality geometry files.[25,26] Some typical repairs include: closing geometric gaps, eliminating unnecessary geometry, removal of duplicate surfaces, and building missing geometry. The prevalence of these geometry problems has been the motivation for the development of dedicated geometry repair applications and the primary subject of conferences addressing the issue of CAD modeling and their use in downstream applications.

Given the attributes listed in the previous section it is important to use diagnostic tools to evaluate the geometry quality prior to grid generation. Inconsistent normals, large gaps, missing geometry, untrimmed surfaces, self-intersecting surfaces and many other problems can often be detected visually. Many commercial CAD packages have built-in capabilities for displaying geometry problems.

The OVERGRID[13] interface can be used to visualize and diagnose problem in discrete geometry databases and offers a number of tools for repairing these databases: changing surface normals, joining adjacent panels to fill gaps, rediscretizing surfaces to remove singularities and/or poor parametric distributions.

Problem CAD geometries are so prevalent that they have spawned their own vocabulary. CAD repair and grid generation applications refer to "CAD healing" and "IGES flavoring" as important features in their respective codes. CAD healing refers to the process of closing gaps between adjacent surfaces, fixing trimmed surfaces that have not been correctly trimmed, converting circular arcs and planar surfaces to NURBS curves and surfaces, and any other repairs necessary to produce an appropriate model. Theoretically, any code with IGES input capability should be able to read any IGES file. In reality, some CAD systems have implemented slightly different interpretations of the IGES standard or have implemented entities that are not compliant with the IGES standard. These variations are known as flavors. Many CAD and grid-generation vendors have implemented specific options to read and write different CAD vendor's IGES flavors.

Some of the ways of dealing with imperfect CAD geometry are:

- Work with local CAD organization to establish specific requirements.
- Use a third party application to repair and flavor CAD geometry.

- Use a grid-generation code that has a robust CAD import capability.
- Import native CAD format to avoid IGES issues.
- Interface directly to the native CAD system.

CADfix from International TechneGroup Inc., Milford, Ohio, and PlanetCAD's online service at *http://www.planetcad.com/index.html* are two commercial options for geometry repair. CADfix has a particularly good range of geometry diagnostic and repair tools.

Gridgen has developed a robust capability to import and repair geometry by creating topology information and "gluing" adjacent surfaces together based on input node and curve tolerances. The OVERTURE developers have chosen to automate as much of the geometry repair process as possible and are planning to provide a set of geometry repair tools.[22,23]

Another solution to the CAD import problem is to add a direct CAD interface to the code. Thereby allowing the grid generation software to work with a CAD part in its original format and potentially enabling grid regeneration on parametrically varying surfaces. ICEMCFD, Inc. has developed versions of their grid generation software that directly interfaces to a number of commercial CAD systems.[27] The CAPRI[28] library provides an application programmer's interface to several different commercial CAD systems, simplifying the process of interfacing grid generation codes to native CAD.

3. Surface Grid Generation

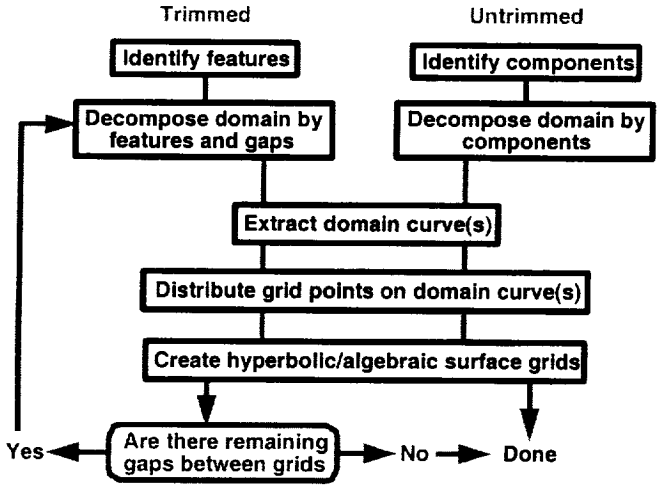

Fig. 3 Surface grid generation procedure for trimmed and untrimmed approach.

The goal of surface-grid generation is to create surface grids that capture both geometric and flow features with sufficient resolution. A strategy for creating overset surface grids begins with this goal in mind and can be carried out via the trimmed or untrimmed approach (Fig. 3). In complex applications, surface grids are usually generated using a mixture of the two approaches. The trimmed approach follows the surface features, and places grid points only on the exposed surfaces. The untrimmed approach follows the geometric components where some points may be placed on unexposed surfaces which will later be removed (see Section 5).

One of the advantages of using overset grids is that various components of a complex body can be added, removed or substituted without regridding the entire configuration. For example, in a fuselage/wing configuration, the untrimmed approach will identify the fuselage and the wing as separate components. After building a fuselage grid, a wing grid can be added by punching a hole in the fuselage grid using a domain connectivity program (Figs. 4a,b). This hole cutting is needed because the fuselage is untrimmed in the vicinity of the wing. A collar grid is then used to fill the resulting hole (Fig. 4c). Further analysis may require the introduction of a tail over the fuselage and a pylon and nacelle under the wing. A new design may require substitution of the old wing with a new wing. In all such cases, existing grids do not have to be disturbed but holes will need to be created or modified. In some cases, if the components introduced are significantly smaller than the parent component, points on the parent grid should be redistributed to match that of the smaller component locally (see Section 3.4).

For the trimmed approach, the wing/fuselage intersection curve, the wing leading and trailing edge curves, the symmetry plane curve and the fuselage nose and tail apexes are identified as surface features. In Fig. 4d, surface grids are built around these surface features. Effectively, the fuselage is trimmed where the wing is to be attached. A wing grid can then be introduced without the need to cut a hole on the fuselage. As will be discussed in Section 5, domain connectivity is robust and highly automated on grid systems that require no hole cutting other than that on the off-body Cartesian grids. Unfortunately, addition of new attachments do require a repartitioning of the parent grid with the trimmed approach which then results in more work.

In practice, it is difficult to anticipate all possible components that will be attached to a body. Care should be taken at the start of an analysis to include as many permanent attachments as possible to maximize usage of the trimmed approach. Optional attachments are more conveniently added utilizing the untrimmed approach. For example, the vertical tail is an important attachment to a space launch vehicle which should be included with the trimmed approach. More detailed subsequent analysis may require the modelling of a small feedline between the fuselage and a booster. The feedline could be attached using the untrimmed

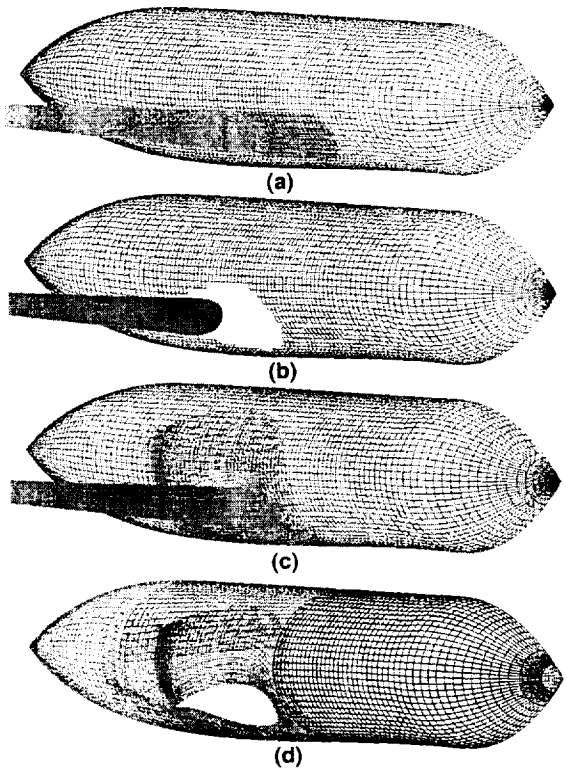

Fig. 4 Untrimmed versus trimmed approach for fuselage/wing configurations. (a) Untrimmed fuselage and wing. (b) Hole cut in fuselage due to wing. (c) Collar grid used to fill fuselage/wing junction. (d) Grid partitioning on trimmed fuselage.

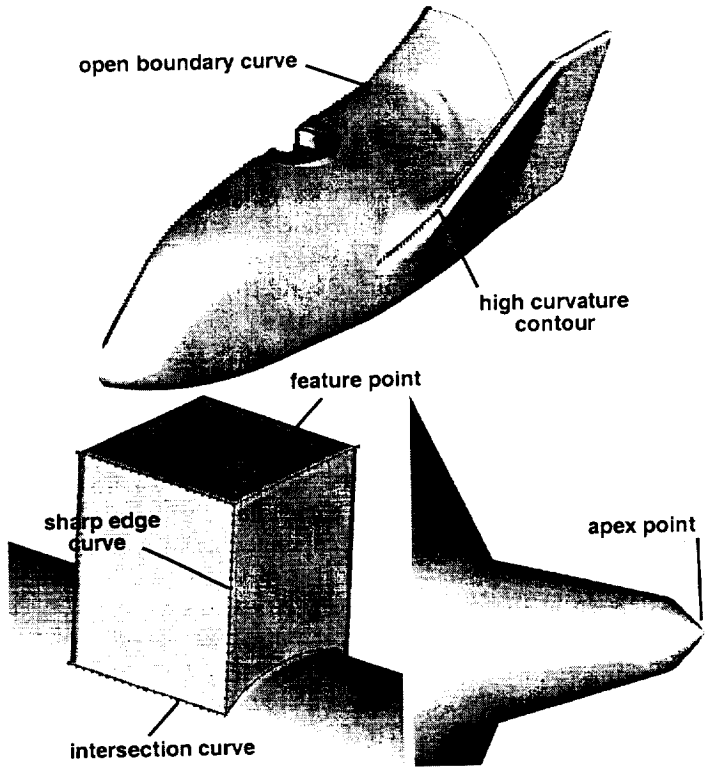

Fig. 5 Examples of surface feature curves and feature points.

approach.

For both trimmed and untrimmed approaches, the surface grid generation procedure can be subdivided into the following steps: feature/component identification, surface domain decomposition, domain curve(s) extraction, grid point distribution and surface grid creation. These steps are presented in more detail in the subsections below.

3.1. Feature/Component Identification

For the trimmed approach, surface features of the geometry are first identified. These features are described by either curves or points.[29] Examples of feature curves include sharp edges on the surface, intersection curve between components, high curvature contours such as those along the leading edges of airfoil shapes, and open boundary curves (Fig. 5). Examples of feature points include the intersection of multiple feature curves and an apex point such as that at the apex of a cone (Fig. 5). In order to accurately capture a feature curve, a grid line should be made to follow the curve. Similarly, a grid point should be placed at a feature point.

The importance of a clean starting point in the geometry description is immediately felt in trying to identify and extract surface features. Since these features frequently lie along the boundary between adjacent geometric patches, gaps between the patches would result in ambiguity in determining such features.

For the untrimmed approach, components of the geometry are first identified. In most cases, the boundaries of a component are clearly defined, e.g. wing, canard, nacelle, pylon, fin, feedline, etc. In some cases, it is not at all clear where one component ends and where another one begins, e.g., the outboard face of the tail fins of the X-38 blends directly into the main body (Fig. 5).

3.2. Surface Domain Decomposition

Surface domain decomposition is the task of decomposing the surface of a solid body into overlapping four-sided domains suitable for surface grid generation. Three-sided domains are sometimes used near surface feature points. An advantage of the overset approach is that the boundaries of such domains can be placed arbitrarily provided they allow sufficient overlap between neighboring grids (see Sections 3.4 and 5 for more details on the overlap required). A domain with complicated topologies tends to result in difficult surface and volume grid generation and domain connectivity, and complex inputs for the flow solver. Also, a large number of small domains tends to reduce flow solver efficiency. Therefore, the ideal surface decomposition should contain as few domains as possible to

resolve the geometry, and that each domain should be topologically as simple as possible.

For the trimmed approach, relevant surface features should be used as starting points for the design of a surface decomposition. For example, a feature curve could form one of the boundaries of a four-sided domain while the remaining boundaries are free floating. The exact locations of free-floating boundaries are usually not known until the surface grid has been created by hyperbolic marching from the fixed boundary. Frequently, a domain is built on each side of a feature curve with the final surface grid built by combining the grids from the two resulting domains. For example, a curve that lies along the wing/fuselage intersection in Fig. 4 is used to construct a domain on the fuselage side and a domain on the wing side. Surface grids created in the two domains are combined to form a single collar grid (see Section 3.2.1). Work on automatic domain decomposition near surface features has been presented in Ref. 29. However, the distances by which such domains should extend from the surface features to maximize surface coverage are currently not automatically determined.

Frequently, domains built around the surface features do not fill the entire surface geometry. Additional domains need to be constructed in the gaps. Such gap domains are usually filled by (1) creating surface curves that form the four boundaries of the domain, and then the interior grid is generated by transfinite interpolation, or (2) creating one surface curve through the middle of the domain and surface grids on each side of the curve are generated by hyperbolic marching. An algorithm to automatically identify and fill such gaps is given in Ref. 30. Unfortunately, this algorithm tends to generate a large number of small domains which are inefficient for flow solver processing.

For the untrimmed approach, components of the geometry are the starting points for a surface decomposition. Each component may be divided into one or more domains depending on its geometric complexity and that of other neighboring components. A separate domain is usually needed at the junction between two intersecting components (see Section 3.2.1). Since domains designed in this manner will cover each component completely, there is no gap-filling step needed as found in the trimmed approach. For example, the External Tank surface is divided into 15 domains (some are shown in Fig. 6). Domains on the upper side are required to be more dense to communicate with the attached hardware and the Orbiter. Collar grid domains are used at the junction between attached feedlines and the surface of the tank.

For both the trimmed and untrimmed approaches, the following guidelines are found to be useful in performing surface domain decomposition:

(1) High flow-gradient regions tend to result in stiffness for the flow equations. Hence, it is best to maintain an implicit scheme through such regions by avoiding domain boundaries parallel to the high flow gradients. A similar condition holds true for geometric discontinuities. This is automatically taken care of by following the guidelines for the trimmed approach.

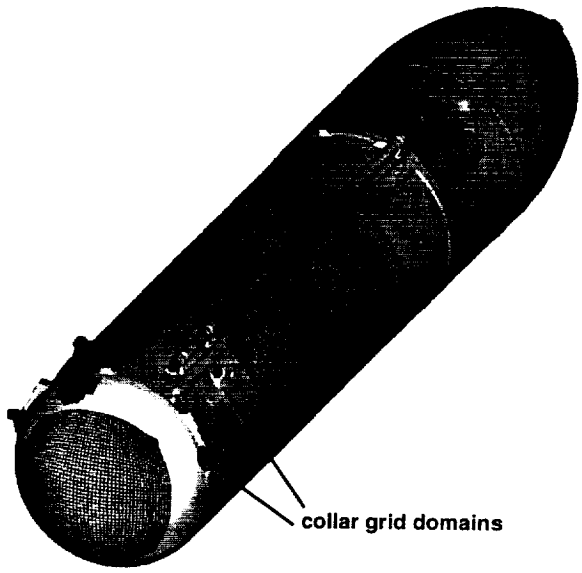

Fig. 6 Surface domains on the External Tank.

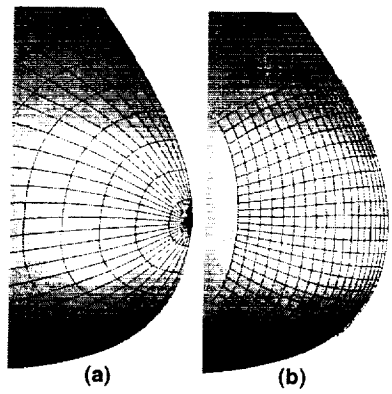

Fig. 7 (a) Singular axis point at fuselage nose. (b) Nose cap grid.

(2) Highly skewed domains require highly skewed grid topologies. Since highly skewed grid cells usually result in poor accuracy, their use should be avoided if alternatives exist.

(3) Domains that result in singular grid topologies should be avoided. Singular surface grid topologies include axis points and slits. Since the grid cells tend to be small and skewed around such regions, they are usually detrimental to flow solver stability, especially for time accurate computations. For example, instead of collapsing the surface grid to a singular point at the nose or tail of a fuselage, a cap grid can be used as shown in Fig. 7. Similarly, another kind of cap grid can be utilized to cover the tip of a wing instead of collapsing the surface grid to a slit (see Section 3.2.2).

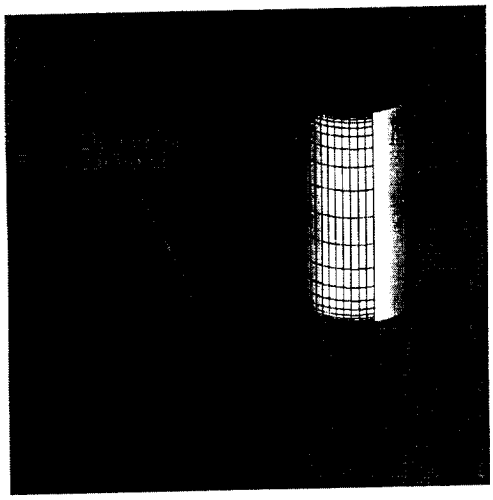

Fig. 8 Patched grids that would result in poor overset grids when extrapolated.

(4) Contrary to naive thinking, extrapolating abutting patched grid domains do *not* usually result in high quality overset domains. This is because patched grid domains frequently violate conditions (1) and (2) above. For example, a common practice of the patched grid approach is to place a domain boundary along a geometric discontinuity. As shown in Fig. 8, extrapolating the indicated patch by hanging it over or wrapping it around the sharp edge on the left would result in bad overset, i.e., the overset boundary runs parallel and is close to a geometric discontinuity. From the patched grid viewpoint, it may seem like a burden to make grids overset. However, from the overset viewpoint, the ability to freely place the grid boundaries (given the guidelines above) is a flexibility that is most valuable.

Surface domain decomposition remains one of the most difficult tasks for which to design robust automation and relies heavily on the experience of the user for an effective strategy. The guidelines provided here should provide a start for further automation research.

3.2.1. *Collar Grids*

The concept of collar grids was originally introduced in the context of intersecting components using the untrimmed approach.[10] The term is used today to describe a grid around the junction between two components for both the untrimmed and trimmed approaches. With the trimmed approach, the collar grid is the result of combining the grids on each side of the intersection curve between two neighboring components. For the untrimmed approach, a collar grid is needed to fill the junction between two components. In order to understand why this is necessary, consider the wing/body example in Fig. 9. The untrimmed body volume grid contains grid points that lie inside the wing and the untrimmed wing volume grid contains grid points that lie inside the body. After removing

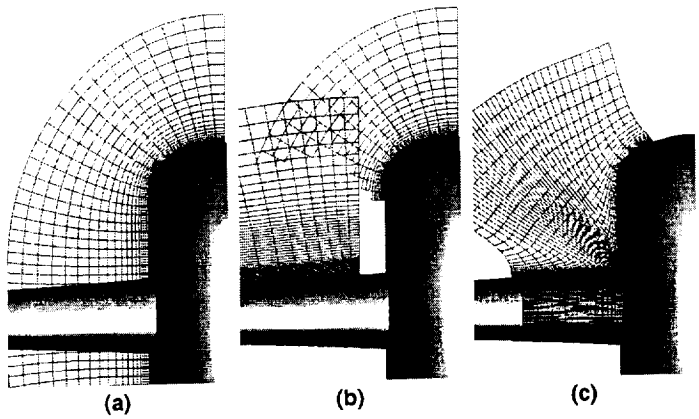

Fig. 9 (a) Volume slice from untrimmed body. (b) Volume slices from wing and body after hole cutting. (c) Volume slice and surface grid for collar.

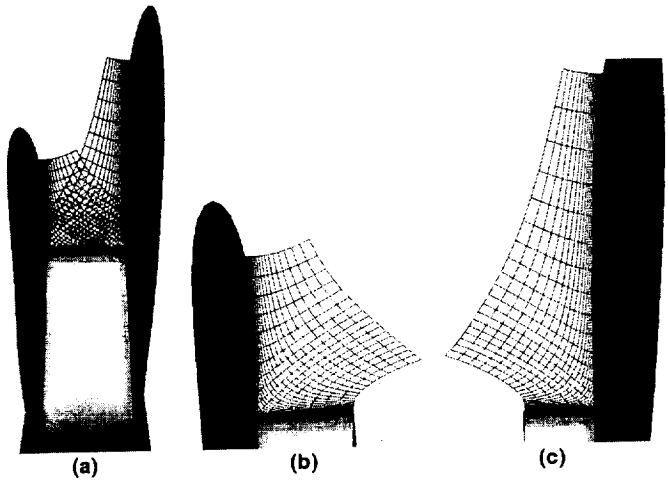

Fig. 10 (a) Two collars in concave region. (b) Close-up view of left collar. (c) Close-up view of right collar.

these non-physical grid points by hole cutting (see Section 5), the junction between the components is left empty. The collar grid is introduced to fill the gap in this region.

Concave corners that arise from collar grid topologies sometimes pose a difficult problem for any volume grid generator. In particular, generating a single volume grid that completely fills such a region can be difficult or even impossible (Fig. 10). Opposite faces of the corner will restrict the total distance that the grid can be marched without overlapping itself. Breaking the concave region into two or more overlapping grids can significantly simplify the volume gridding process and improve the local grid quality by reducing the number of constraints on the grid generator.

3.2.2. *Airfoil Shapes*

In aerospace applications, airfoil shapes may appear in many different geometric components. These include wings, pylons, nacelles and tails in space vehicles

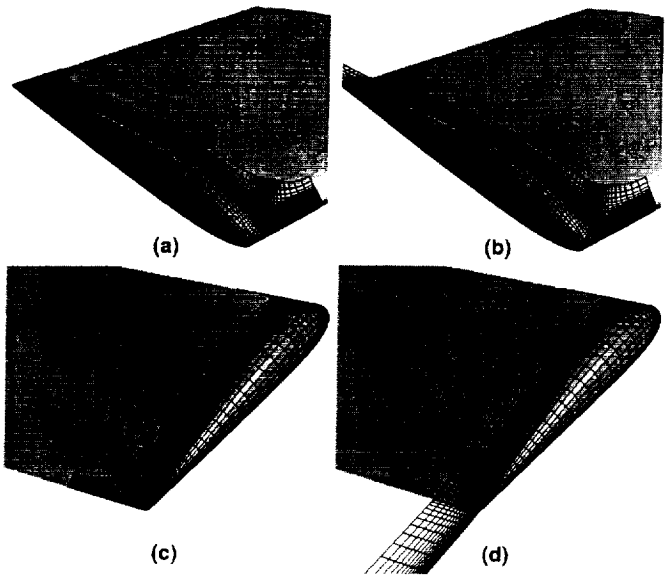

Fig. 11 (a) O-type tip cap (front view). (b) C-type tip cap (front view). (c) O-type tip cap (back view). (d) C-type tip cap (back view).

and airplanes, turbine blades in turbomachinery, slats and flaps in high-lift devices, rotor blades in rotorcrafts, fins and canards in missiles, and sails in submarines. Choosing a grid topology is the first order-of-business for any structured grid approach. For overset gridding of wings or other extruded shapes, there are two special regions, the root and the tip.

As mentioned above, the wing root can be conveniently covered with a collar grid, with the streamwise topology and grid spacing usually chosen to match the main wing grid. In the spanwise direction, the grid wraps onto the fuselage as shown in Fig. 9c, conveniently producing a collar grid with only one viscous direction.

The wing tip region is most effectively covered by a tip cap grid. As opposed to closing the tip into a slit or trying to wrap the wing grid around the tip, the tip cap can resolve high gradients from the tip vortex while maintaining relatively smooth grid spacing. Again depending on the wing grid topology, the cap can either wrap from the tip onto the wing trailing edge (O-type cap) or extend into the wake along the wake cut (C-type cap) (Fig. 11).

With care, tip caps can also cover squared-off tips while preserving the sharp edge. This topology is useful for covering an engine pylon shelf region (Fig. 12), or the ends of flaps or control surfaces (Fig. 13). Tip cap grids can be generated using the "wingcap" tool in Chimera Grid Tools (CGT, *http://www.nas.nasa.gov/~rogers/cgt/doc/man.html*). While automated, "wingcap" has a large number of parameters to control and fine-tune its operation.

One final issue on gridding wings is how to handle blunt (or finite thickness) trailing edges. While ideal-

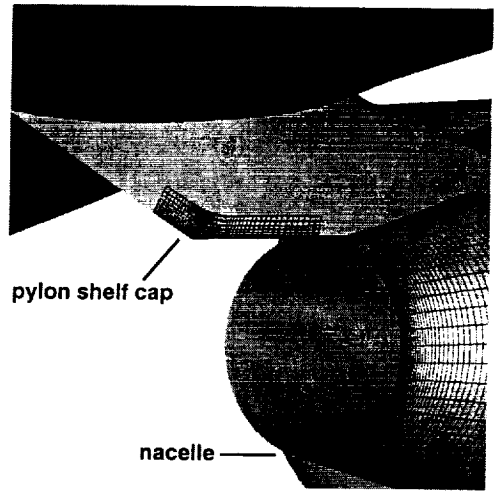

Fig. 12 Pylon shelf cap grid.

Fig. 13 Flap end cap grid.

ized wings have a sharp trailing edge, most real wings have have a blunt trailing edge, requiring some grid approach to handling this geometric feature. For cases where wake resolution is not critical, an O-grid can used to wrap around the trailing edge. Otherwise, a C-grid topology can be utilized in one of several ways, depending on the trailing edge thickness compared to the streamwise grid spacing at the trailing edge. If the thickness is less than 0.2% of the local chord, it is recommended that the grid be continued aft of the trailing edge and the gap closed in the first point in the wake (Fig. 14a). If the trailing edge thickness is greater than 0.2% chord, this approach can result in too abrupt a change in grid spacing and direction. An alternative is to wrap the surface grid onto the base and then out the wake (Fig. 14b).[31,32] The drawback to this second approach is that it tends to result in unsmooth, skewed grids.

3.3. Domain Curves Extraction

The surface grid for a domain may be created from one or more bounding curves depending on how many boundaries are fixed or are free to float. Curves that lie on fixed boundaries have to be extracted from the surface geometry. Whether the trimmed or untrimmed approach is taken, these are typically just the feature curves of the geometry. For the trimmed approach, some domains are used to fill the gaps between grids

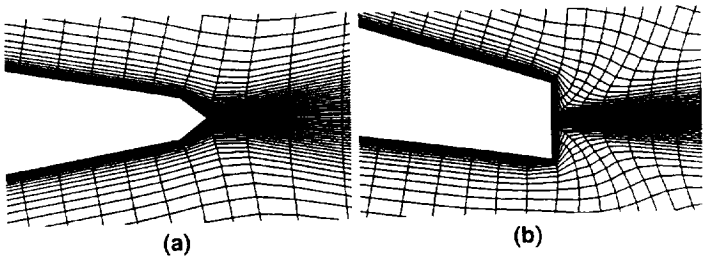

Fig. 14 Finite thickness trailing edge grids. (a) Collapse in first wake point. (b) Wrap around the base then to wake.

created from the surface features. All boundaries of such domains are usually free floated. The number of bounding curves to be extracted then depends on whether a hyperbolic or algebraic method is employed for grid generation (see Section 3.5 for further discussion).

The domain curves extraction method is dependent on the format of the geometry definition. For a CAD geometry description, curve extraction should be performed inside the same CAD tool from which the CAD model is built. Sometimes feature curves, such as intersection curves, are already present in the CAD definition. At other times, they have to be extracted manually. If the geometry description is in the form of panel networks or triangles, surface features can be extracted using the algorithms and tools described in Ref. 29. Automatic extraction is possible for sharp edges, intersection curves and open boundaries. In such cases, small manual clean up using a graphical interface is still necessary as extraneous curves are frequently produced as a result of poor surface resolution on high curvature regions and gaps between surface patches. For high curvature contours, manual extraction using a graphical interface such as OVERGRID is still the most robust method.

3.4. Grid Point Distribution

After extraction of the domain bounding curve(s), grid points should be appropriately distributed on these curves. The grid point distribution should provide sufficient resolution of the surface geometry and sufficient overlap between neighboring domains (see Section 5). A grid induced truncation error analysis[33,34] could be used to diagnose grid point distribution quality.

Decisions on grid point distribution usually begin with the choice of a grid spacing for near-field resolution Δs_g. This is the typical grid spacing on the smooth and flat regions of the surface geometry and serves as the upper limit on the grid spacings of all surface and volume grids in the near-body flow field. In other words, finer grid spacings will be used in high surface curvature regions (e.g., leading edges), sharp convex corners, and high flow gradient regions. The choice of Δs_g directly affects the total number of grid

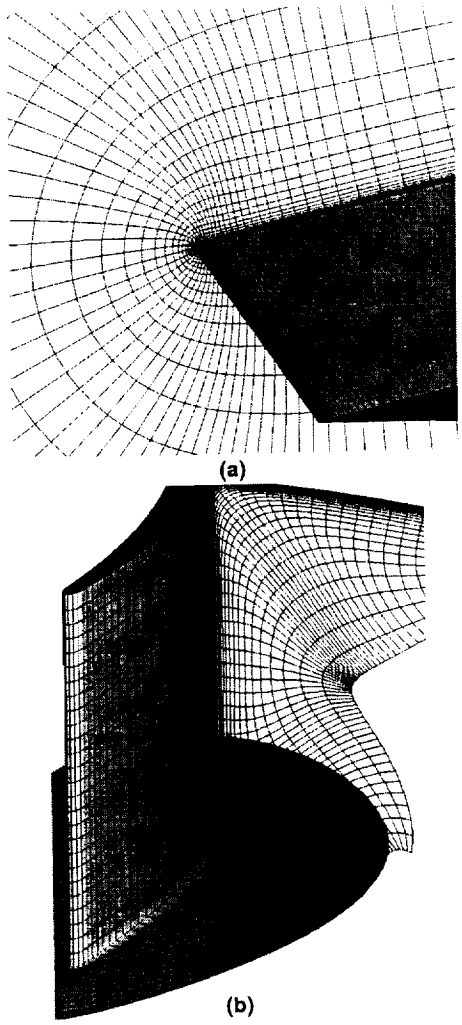

Fig. 15 Surface and volume grid point distributions at (a) convex, and (b) concave corners.

points required in the configuration and hence could be influenced by the affordability of the simulation. For example, a one inch near-field resolution may be desirable for a particular scenario, but limited time and computer resources may dictate a larger value.

It is clear that grid clustering should be used where the geometry is changing rapidly in the convex sense, i.e., at convex high curvature regions and convex sharp corners. The faster the change, the smaller the grid spacing needs to be to counteract the diverging effect of grid spacing in the volume grid around such a region (Fig. 15a). By the same token, grid spacings should not be clustered into regions where the geometry is changing in the concave sense since the volume grid lines tend to converge as they grow away from the surface. Moreover, equal grid spacings and stretching ratios should be used on both sides of a corner to maintain a uniform resolution of the corner (Fig. 15b).

The truncation error induced by the grid is related to the grid stretching ratio.[33] In order to minimize truncation errors, a small stretching ratio should be used. Again, this is counter-balanced by the afford-

ability of using a large number of grid points with a small stretching ratio. In practice, a stretching ratio of 1.2 or below for surface grids and 1.3 or below for volume grids is found to work well.

Section 2 already discussed the removal of irrelevant geometric features. Relevant geometric features should therefore be represented adequately, no matter how small. Most flow solvers utilize a 3 to 5 point differencing stencil which implies that at least 5 points should be used on any geometric feature. Obviously, similar sized geometric features at different locations should be resolved with similar grid point distributions. Also, for flow solvers with multigrid options, a multigridable number of points $(2^n + 1)$ is preferred for each grid dimension.

One of the most important aspects of grid point distribution is to ensure that the grid resolutions are comparable in regions where inter-grid information is exchanged. For overset grids, grid spacings from neighboring grids in the overlapped zones should be comparable. A common mistake is to dump the wake of an airfoil C-grid into a coarse background Cartesian mesh a short distance downstream from the airfoil. The coarse Cartesian mesh clearly cannot resolve the flow features passed on from the viscous spacing in the C-grid with the same fidelity. Moreover, the C-grid also receives poor resolution interpolation data from the coarse Cartesian mesh thus resulting in possible upstream contamination (see Section 4.1.3 for further discussion). The grid spacing compatibility issue also arises when a small component is added to a large component. For example, an analysis may first be performed on a wing with no pylon/nacelle attached. The wing may have uniform spacing in the span direction. With the introduction of a pylon/nacelle attachment, the spanwise grid spacing around the pylon intersection on the wing should be reduced to match the pylon grid spacing (Fig. 16). Higher spanwise resolution is also needed to capture the high angle-of-attack flow physics caused by the nacelle.

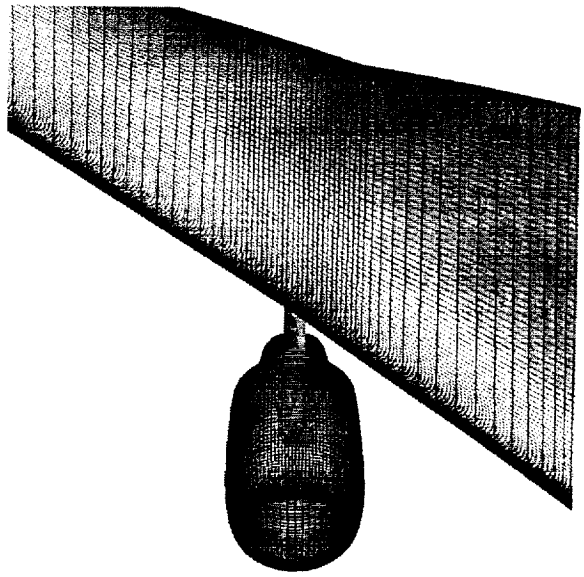

Fig. 16 Wing surface grid is refined near pylon attachment region to provide proper communication with pylon grids.

3.4.1. Airfoil Shapes

Clustering for overset wing grids must satisfy geometric and flow gradient requirements, as well as provide adequate resolution in overlap regions for good communication between grids. In that the majority of the wing surface can be covered with a single grid, many of these requirements are the same as for patched/structured or unstructured grid approaches.

Required clustering at wing leading and trailing edges is determined by expected flow gradients. For the leading edge, this depends on the leading edge radius and the range of stagnation point locations, but a rule of thumb is to use a grid spacing of 0.1% of the local chord. For the trailing edge, a spacing of 0.2% chord is recommended in order to capture the high flow gradient associated with the equalization of upper and lower surface pressure. The use of 101 grid points on the upper surface and 101 points on the lower surface will avoid high grid stretching and maintain adequate resolution when using these leading and trailing edge clusterings. More points may be needed for good communication between high lift elements or for precise capturing of shocks, while fewer points can be used for lightly loaded components like pylons.

In the spanwise direction, grid clustering for the wing is determined by having sufficient resolution to overlap the wing root collar and tip cap grids. Typical spacing in the root and tip overlap regions is 5% of the local chord. There may be additional requirements in the middle of the wing due to geometry or expected flow gradients. For example, Fig. 16 shows a refined wing grid in the region of an underwing pylon/nacelle.

3.5. Surface Grid Creation

The goal of surface-grid creation is to generate a surface grid from one or more supplied bounding curves of the domain. Grid points created during the gridding process should lie on the geometry definition, and grid cells should follow the guidelines of good mesh quality such as orthogonality and stretching constraints. It is good practice to do a quick check on surface grid quality prior to volume grid generation, e.g., by looking at a graphical plot of grid quality functions as those found in the OVERGRID interface.

At the surface domain decomposition step, the number of prescribed bounding curves for each domain is already determined. By following the overset gridding strategy described so far, domains with just one specified bounding curve frequently arise, e.g., the collar

and cap grids discussed in Section 3.2.2. Such cases are perfectly suited for hyperbolic surface grid generation methods.[9] From the prescribed initial curve, the surface grid is created by a marching scheme under orthogonality and cell area constraints. Side boundaries are usually free floated or restricted to float along a given plane or curve. The boundary opposite the initial curve is always free floated. Enforcing orthogonality with the hyperbolic marching is not always appropriate as in the case of growing a surface grid (part of a collar) onto a swept wing from the wing root. The underlying geometry dictates that the grid lines be swept to follow isoparametric lines on the wing. In this situation, an algebraic marching scheme is preferred.[9]

Domains with two, three or four prescribed bounding curves could also arise in certain situations. Such grids are best treated with algebraic or elliptic methods. Ref. 9 presents an implementation that uses transfinite interpolation for the grid interior. Missing bounding curves are first automatically created for cases with two or three prescribed boundaries. The class of grids associated with airfoil shape components are good examples requiring such treatment. For example, the leading and trailing edge curves of a wing are two opposite bounding curves for a surface grid on the upper or lower surface of the wing. One could, of course, break the upper surface into two domains, each with only one prescribed bounding curve, and use hyperbolic marching from the leading edge curve and the trailing edge curve respectively. However, in order to preserve flow solver implicitness over such important shapes, it is preferable to model the airfoil shape in one domain in the streamwise direction.

4. Volume Grid Generation

The goal of volume-grid generation is to fill the flow field with discrete points fine enough to resolve the fluid flow around an object. Body-conforming volume grids should clearly be used near the surface. In the normal direction, the grid spacing should be small near the body surface and should increase with distance from the surface. For a simple object that can be modelled with just one surface grid, it is appropriate to grow a single volume grid from the surface to the far field. For more complex objects with multiple surface grids, the near-body volume grids usually cannot uniformly fill the three-dimensional space in the vicinity of the body due to grid spacing stretching. In order to provide a uniform resolution of the near-body flow field, and to fill any gaps that might exist between the near-body volume grids, it is advantageous to grow these grids to just a short distance from the body, and then embed the near-body grids in stretched Cartesian grids that extend to the far field.

The question then remains on how far the near-body volume grids should extend from the surface. With grid spacing stretching in the normal direction, the normal grid spacing will increase with distance from the surface until it reaches a value close to the near-field resolution spacing Δs_g discussed in Section 3.4. The distance D_m from the surface at which this value is reached is approximately how far the near-body volume grids should extend for the following reasons.

(1) For viscous flows, this distance should be well clear of any wall-bounded viscous effects, i.e., the boundary layer flow field is completely captured by the fine near-body grids.

(2) A typical length scale of grid cells on the surface grids is approximately equal to Δs_g. This cell size is propagated into the near-body volume grids in the tangential direction. At distance D_m from the surface, the volume cell size is then approximately Δs_g in all three directions. This means enclosing the near-body grids inside a uniform Cartesian grid with this spacing would provide optimal inter-grid communication, thus easing the task of domain connectivity. Moreover, the Cartesian grid could uniformly fill any gaps that may occur between the near-body grids as the inter-grid overlap starts to deteriorate with distance away from the surface under certain situations.

4.1. Near-Body Grid Generation

The near-body volume grids should conform to the body to provide proper modelling of the body geometry. Tight normal-direction clustering should be maintained near the wall to provide good boundary layer resolution for viscous cases. Mesh orthogonality should be maximized and grid stretching should be minimized for better solution accuracy. All of the above desirable attributes could be accomplished by using hyperbolic methods.[6] Moreover, hyperbolic volume grid generation only requires the specification of a surface grid while the remaining boundaries do not have to be prescribed. The hyperbolic-marching scheme is computationally much cheaper than elliptic methods. Together, the simple inputs and efficient marching scheme result in extremely fast grid generation time.

With the surface grids already designed to provide sufficient overlap, this property is inherited by the near-body volume grids in the vicinity of the body. The overlap can be maintained away from the body by applying a splay boundary condition on the floating boundaries of the volume grid during hyperbolic marching. Several visualization tools, such as Visual3[35] and Fieldview,[36] have the ability to create and display cutting planes through a collection of volume grids for checking grid overlap quality. This can be a useful diagnostic tool prior to running a domain connectivity code.

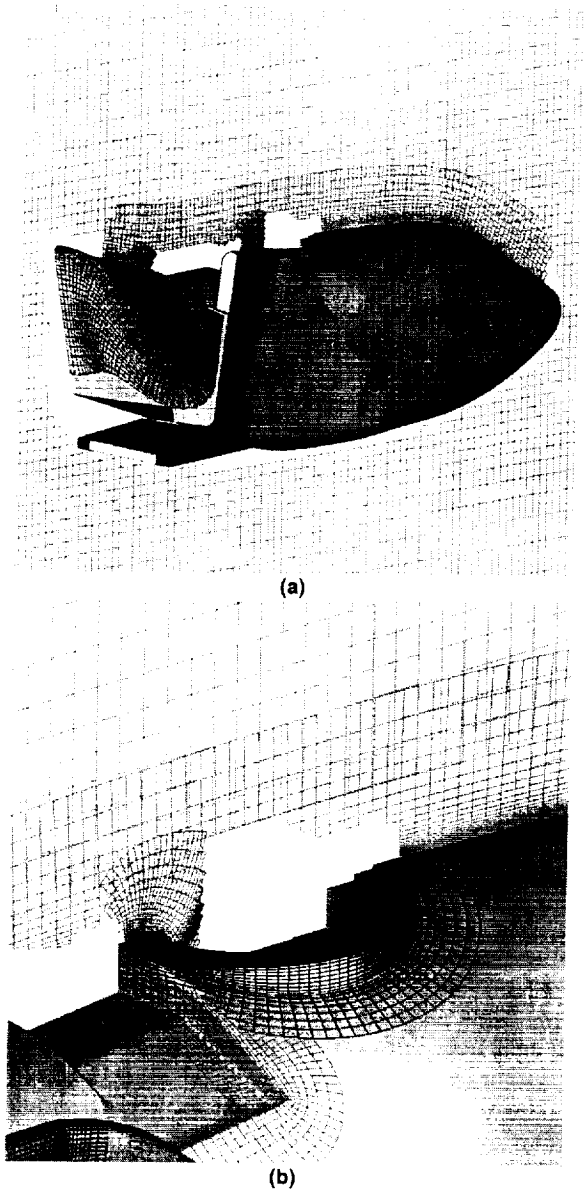

Fig. 17 Near-body volume grids and off-body Cartesian grid for X-38 V-131R vehicle. (a) Far view. (b) Close-up view showing marching distance of near-body grids and grid spacing matching with off-body Cartesian grids.

Typically the same wall spacing, marching distance and total number of points are used for all near-body grids in the marching (normal) direction. Moreover, if the surface grids are designed with relatively simple topologies, side boundary control (periodic, axis, constant plane, free floating) can be automated and the same smoothing parameters could be employed for all grids. For such cases, a multi-million point volume grid system can be created in just a few minutes on a typical workstation or PC with almost no user intervention (Fig. 17).

It is important to ensure the volume grids have no negative Jacobians and that the grids are of high quality prior to proceeding to domain connectivity and the flow solver. Since Jacobian computation is non-unique, the most appropriate check is to use the same algorithm as that in the flow solver. For example, the HYPGEN[7] tool contains a Jacobian calculator that is identical to that used by the OVERFLOW[37] flow solver. A quick visual inspection is also highly recommended since having positive Jacobians for all cells does not guarantee a properly formed grid, as shown in the self-intersecting case in Fig. 18.

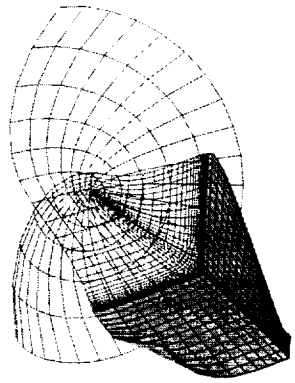

Fig. 18 Self-intersecting volume grid where all cell volumes are positive.

Some considerations that arise for near-body volume grids are discussed in the subsections below.

4.1.1. *Viscous Wall Spacing*

The requirements for the normal-direction spacing at the wall for viscous calculations depend heavily on the Reynolds number and on the type of turbulence modeling used. Although the boundary layer will vary significantly over the surface of a body, it is typical practice to maintain the same wall spacing for an entire configuration. Here the discussion is limited to considerations of Reynolds-averaged Navier-Stokes solvers. Two-equation turbulence models often recommend wall spacing with y^+ values less than one, where y^+ is the non-dimensional turbulent distance. One equation turbulence models require wall spacing given by y^+ approximately equal to one. If wall functions are used, a significantly larger wall spacing can be used, corresponding to y^+ between 35 to 100.

The turbulent y^+ distance can be estimated by using flat plate formulas for either subsonic/transonic flows[38] or supersonic/hypersonic flows.[39] These formulas are summarized as follows. Let

$$y^+ = \frac{\rho_w u^* y}{\mu_w} \tag{1}$$

where ρ_w is the density at the wall, y is the physical coordinate normal to the wall, μ_w is the viscosity at the wall, and u^* is the friction velocity given by $u^* = \sqrt{\tau_w/\rho}$ where τ_w is the wall shear stress. Assuming $\rho_w = \rho_\infty$ and $\mu_w = \mu_\infty$, we can write

$$y^+ = Re\sqrt{\frac{c_f}{2}}y \tag{2a}$$

$$y = \frac{y^+}{Re\sqrt{\frac{c_f}{2}}} \tag{2b}$$

We use a flat-plate correlation from Ref. 38 to estimate the skin friction coefficient c_f:

$$c_f \approx \frac{0.455}{ln^2(0.06Re_x)} \tag{3}$$

where Re_x is the Reynolds number based on some downstream distance x at which we wish to compute y^+. A typical choice would be at 10% of the reference length, or $x = 0.1$ and $Re_x = 0.1 \times Re$.

This method gives reasonable values for subsonic flow, and conservative values for higher Mach numbers. A more sophisticated model from Ref. 39 incorporates compressibility as a function of freestream temperature and Mach number. Defining a compressibility factor f_{comp} and a reference temperature T^*,

$$f_{comp} = 1 + 0.1157M_\infty^2 \tag{4a}$$

$$\frac{T^*}{T_\infty} = f_{comp} \tag{4b}$$

A viscosity ratio is computed using Sutherland's law, and a compressibility correction r_{comp} is defined for the Reynolds number:

$$\frac{\mu^*}{\mu_\infty} = \left(\frac{T^*}{T_\infty}\right)^{\frac{3}{2}} \left(\frac{1+\frac{S}{T_\infty}}{\frac{T^*}{T_\infty}+\frac{S}{T_\infty}}\right) \tag{5a}$$

$$r_{comp} = \frac{1}{\frac{\mu^*}{\mu_\infty} f_{comp}} \tag{5b}$$

where the Sutherland constant $S = 199°R$ for air. The compressible skin friction coefficient is then estimated as

$$c_f \approx \frac{0.455}{ln^2(0.06Re_x r_{comp}) f_{comp}} \tag{6}$$

The difference between free stream and wall conditions are accounted for as

$$\frac{T_w}{T_\infty} = 1 + \frac{\gamma-1}{2} Pr^{\frac{1}{3}} M_\infty^2 \tag{7a}$$

$$\frac{\mu_w}{\mu_\infty} = \left(\frac{T_w}{T_\infty}\right)^{\frac{3}{2}} \left(\frac{1+\frac{S}{T_\infty}}{\frac{T_w}{T_\infty}+\frac{S}{T_\infty}}\right) \tag{7b}$$

$$\frac{\rho_w}{\rho_\infty} = \left(\frac{T_w}{T_\infty}\right)^{-1} \tag{7c}$$

Typically the Prandtl number for air is taken as $Pr = 0.72$.

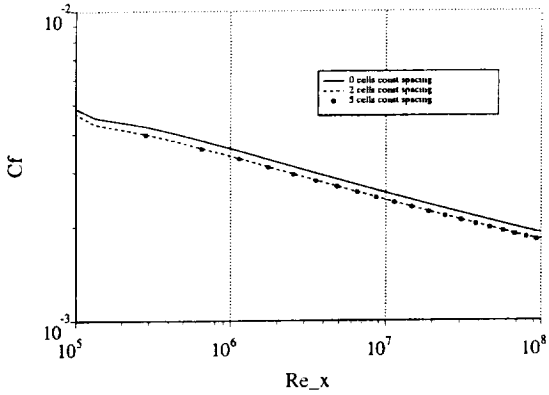

Fig. 19 Skin friction coefficient vs. plate Reynolds number, for grids with 0, 2, and 5 cells of constant spacing next to the plate (solid line, dashed line, and filled circles resp.)

A modified expression for the distance to the wall can now be written as

$$y = \frac{y^+}{Re\sqrt{\frac{c_f}{2}}} \left(\frac{\frac{\mu_w}{\mu_\infty}}{\sqrt{\frac{\rho_w}{\rho_\infty}}}\right) \tag{8}$$

Note that for $M_\infty = 0$ this equation reduces to the simpler version in Eq. 2a, regardless of temperature.

Typically, 20 to 30 points in the boundary layer is considered good resolution. Besides viscous grid spacing, initial stretching also affects the accuracy of calculated skin friction. If the flow solver uses first-order one-sided differencing for grid metrics at the wall, several points of constant spacing should be used.[40] For example, Fig. 19 compares skin friction on a flat plate for grids with 0, 2, and 5 cells of constant spacing at the wall before beginning grid stretching. It can be seen that even 2 cells of constant spacing removes the stretching dependence on the skin friction.

4.1.2. *One Versus Two Viscous Directions*

In Section 3.2.2 a collar grid topology with only one viscous direction was discussed. This is adequate for resolving integrated aerodynamic forces and most corner flow features. It simplifies the use of a thin-layer viscous approach and avoids extremely small grid cells that result from grids which require viscous spacing in two directions. Such small cells can limit the stable time step, slowing convergence. One drawback of single viscous-direction approach is the treatment for a wake cut of a C-grid collar. For the fuselage/pylon/nacelle topology shown in Fig. 20a, the fuselage in the wake of the pylon is represented as a solid surface plus a flow-through wake cut. Because of coordinate direction differencing, the flow near the surface may effectively see a partial grid-cell "tab" at the root of the wake cut. At significant angles-of-attack this can interrupt the smooth flow along the fuselage, degrading the accuracy of the calculation. This can be

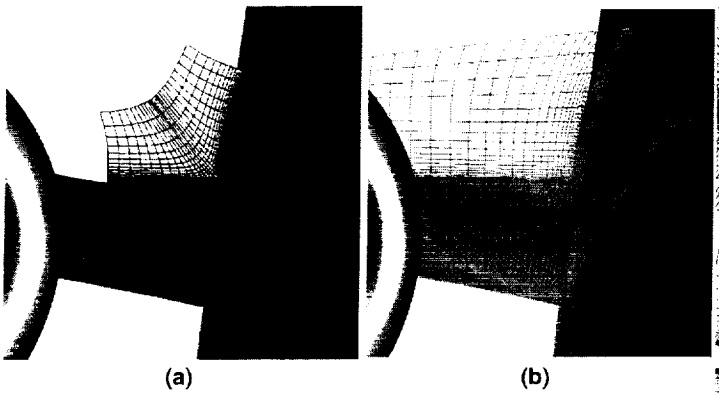

Fig. 20 Slice of collar grid at fuselage/pylon junction. (a) C-type topology (one viscous direction). (b) CH-type topology (two viscous directions).

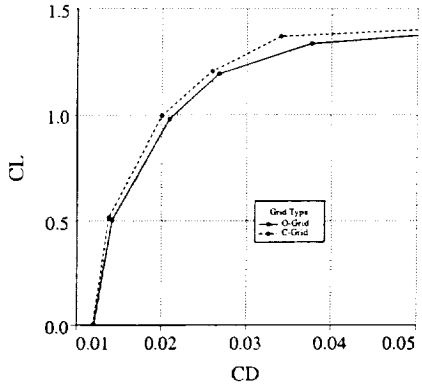

Fig. 21 Drag polars for a NACA 0015 airfoil using an O-grid (solid) and C-grid (dashed).

avoided by using an H-grid topology in the spanwise direction (Fig. 20b), but results in a collar grid with two viscous directions.

4.1.3. *C and O Grids and Wake Smoothing*

While the choice of a C- or O-type grid for a wing has been referred to obliquely as affecting other grid choices such as caps and collars, it is primarily a choice determined by the wing. Typically a C-grid is chosen for wings to provide grid resolution in the wake region and to avoid wrapping a grid around the obtuse angle of a sharp trailing edge. Conversely, configurations where the wing wake does not impinge on other aerodynamic surfaces, or for geometries with thick trailing edges such as space access or reentry vehicles, the wake may be more conveniently resolved with an O-grid. Even for a relatively sharp NACA 0015 airfoil, Fig. 21 shows a difference of only about 5% between O-grid and C-grid drag polars.

Another issue with C-grid wakes is viscous clustering at the wake cut. It has been found that maintaining a $y^+ = 1$ spacing in the wake can result in convergence difficulties for some flow solvers (e.g., OVERFLOW). This cell spacing is unnecessarily small in the wake region, and generally does not coincide with the actual

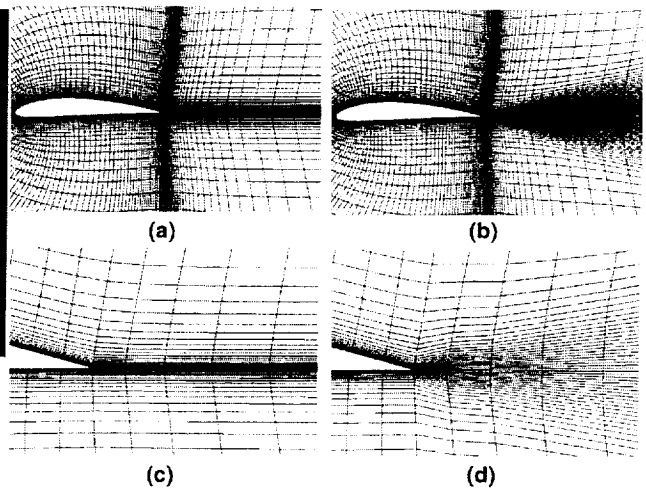

Fig. 22 Airfoil wake region. (a) Before smoothing (far view). (b) After smoothing (far view). (a) Before smoothing (close-up). (b) After smoothing (close-up).

wake. Furthermore, one typically stretches the streamwise spacing aft of the trailing edge, resulting in grid cells with extremely large aspect ratios. One way to resolve these problems is to smooth the grid in the wake region, relaxing the viscous clustering (Fig. 22). The "smogrd" tool in CGT can perform this function for wing, collar and wing cap grids.

4.1.4. *Multi-Element Systems*

High-lift aerodynamics analysis presents one of the most difficult challenges for CFD due to the complex geometry and complex flow physics. This stems from the multiple bodies that make up the high-lift components (wing, flaps, slats, and various attaching hardware) and their large range of length scales. The small gaps between successive elements lead to boundary-layer and wake interactions, a complex viscous fluid-dynamic phenomenon which greatly affects the performance of the high-lift system. This complex geometry and flow puts specific constraints on the volume grid generation. The body-normal spacing must be fine enough to resolve the wakes flowing from upstream elements. Also, the spanwise spacing on the wing and flaps must be fine enough to resolve the vortical structures formed at the spanwise ends of the leading-edge devices. Fig. 23 shows the leading edge of a Boeing 777-200 aircraft[41] in a high-lift configuration, near the engine pylon. The inboard slat and the Krueger slat can be seen, and the small gaps (both spanwise and streamwise) between the elements are apparent.

An example of the effect of the body-normal spacing in resolving a high-lift flow field was shown previously by Rogers for flow computations about a three-element airfoil.[42] Fig. 24a shows some grids with inadequate normal spacing in the mesh around the downstream

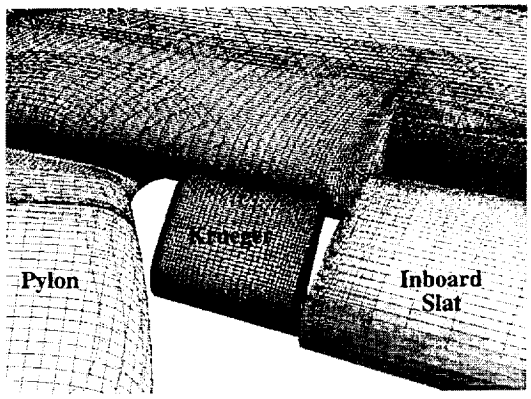

Fig. 23 Leading edge devices in high-lift configuration.

element. The hole in the downstream-element grid caused by the upstream element is in an area where the normal spacing is too coarse. The grid points denoted with circles are the only fringe points which will receive data interpolated from the wake of the upstream element. They surround the wake, but there are no points within the wake. This causes the downstream grid to entirely miss the wake velocity defect, and hence compute too much mass flow in this region. This mass-flow error is passed back into the upstream-element grid when it receives interpolated data at its downstream outer boundary. This causes too much mass to flow over the upper surface of the upstream element resulting in large errors in the computed pressure. An improved grid for this geometry is shown in Fig. 24b, which has significantly better grid resolution in the normal direction. It can be seen that there are more fringe points to resolve the wake of the upstream element. Results from this grid were reported by Rogers[42] to be in good agreement with experimental results.

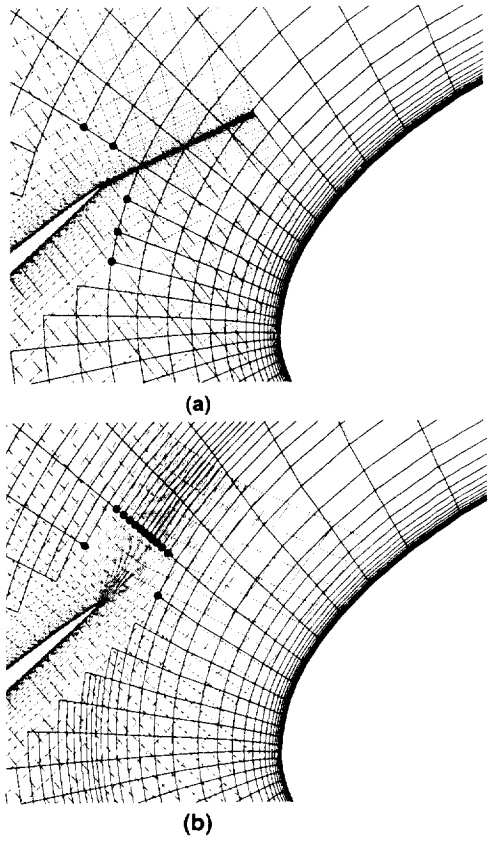

Fig. 24 Close-up view of wake of upstream element and upper surface of downstream element. (a) Grids with resolution mismatch. (b) Grids with more resolution compatibility.

4.2. Off-Body Cartesian Grid Generation

As already mentioned, it is advantageous to employ uniform off-body Cartesian grids to provide uniform resolution of important near-field flow features. Beyond the near-field, stretched grids are used to extend the computational domain to the far field. Two approaches in use today are described below.

The first approach encloses the near-body curvilinear volume grids with a small number of Cartesian grids. Each of these Cartesian grids has a uniform core region that completely encloses some components of the near-body grids, and contains stretched outer layers that extend away from the body (Fig. 17). The far field can be resolved with either a large coarse Cartesian grid or with a curvilinear grid with ellipsoidal topology (Fig. 25).

The second approach utilizes many Cartesian grids with successive levels of refinement.[43] These grids are first generated automatically based on proximity to

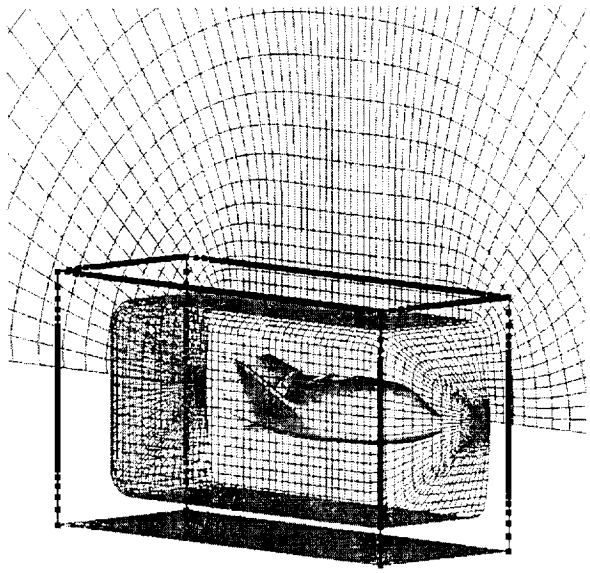

Fig. 25 Off-body Cartesian grid and ellipsoidal far field grid.

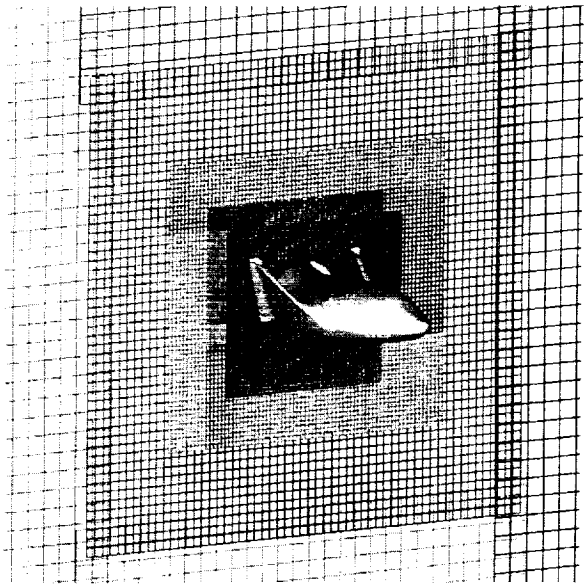

Fig. 26 Off-body Cartesian grids with successive levels of refinement by proximity to body.

the body where the grid spacing of the first and finest level is matched to the near-field grid resolution Δs_g. The grid spacings are coarsened by a factor of two for each level of the outer layers which extends to the far field (Fig. 26). All levels of these Cartesian grids can be modified to adapt to the solution.

Both approaches are highly automated and provide very good results if the main purpose of the simulation is the accurate prediction of forces and moments. If off-body flow features such as shocks need to be captured accurately, the second approach is clearly more appropriate. However, it also tends to be more expensive due to a higher volume of inter-grid communication between a larger number of grids.

5. Domain Connectivity

The task of domain connectivity consists of three processes: projection, hole cutting, and fringe-point interpolation. Projection involves shifting surface points to remove the effect of surface discretization errors. Hole cutting is the removal of grid points from one grid that reside inside the solid wall of another grid. The flexibility of allowing neighboring grids to arbitrarily overlap often results in this situation. Fringe points are boundary points which are to be updated by the flow solver using interpolated solution data from a neighboring grid. Fringe points arise from two sources: at the outer boundary of each grid where no flow-solver boundary conditions are specified, and at the boundary of holes. A valid interpolation stencil needs to be provided for each fringe point. If a valid interpolation cell cannot be obtained for a fringe point, this is called an orphan point. Orphan points can occur where the hole-cutting operation removes too many points, too few points, or where there is inadequate overlap between neighboring overset grids.

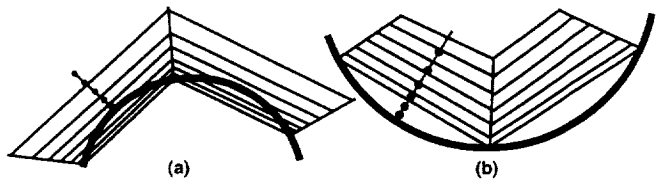

Fig. 27 Discrete surface mismatch on a curved surface. (a) Convex surface. (b) Concave surface.

There are a number of different software programs which were written to perform these overset domain connectivity tasks. A partial list of these codes includes: PEGSUS4,[8] PEGASUS5,[14] DCF,[15] BEGGAR,[44-47] CHALMESH,[48] OGEN,[49] and software from the CFDRC corporation.[50]

5.1 Projection

Surface to surface projection is an issue for overset grids because of discretization errors. When two different randomly-overlapped meshes discretize the same curved analytic surface, they each represent a different approximation to the original surface. Thus at any random point in the flow volume above the solid body, each mesh will have a different measure of the distance from this point to its own discrete surface. Consider the simplified representation of a two-dimensional mesh on a curved surface shown in Figs. 27a and 27b, where the body-circumferential direction has been immensely compressed in scale in relation to the body-normal direction. This way the figure can represent the high-aspect ratio grid cells typically found in high Reynolds-number simulations, where the tangential spacing is typically on the order of 100 to 1000 times the normal spacing at the wall. The circular arcs in the figures represent the exact surfaces; the 3x7 meshes represents a subset of a donor mesh; and the circles drawn along a wall-normal line represent fringe points from a neighboring mesh.

In Fig. 27a, it can be seen that the first fringe point above the surface actually lies in the fifth donor cell above the surface. In actual high-Reynolds number overset meshes, the situation is typically worse, with first-cell fringe points residing in donor cells which are 10-20 cells above the wall. When the curvature of the surface is reversed from convex to concave as in Fig. 27b, the fringe points close to the wall can actually lie inside the solid body of the neighboring mesh. The convex scenario will result in significant errors in the resulting overset solution; recipient fringe points at the first point above the wall can receive donor solution data from cells in the middle of the boundary layer. The concave scenario will result in fringe points for which no donor interpolation cell can be found, resulting in orphan points.

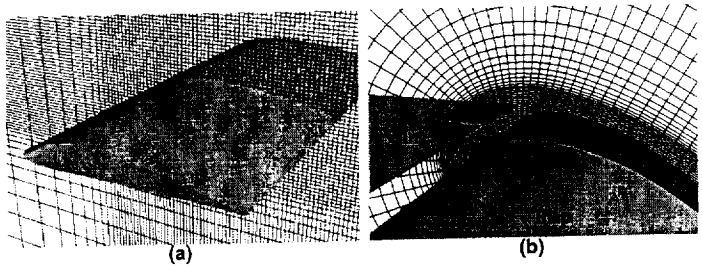

Fig. 28 (a) Cartesian box grid intersecting a wing surface. (b) Wing and flap grid; flap grid require hole cutting.

The solution to this problem is to perform some type of projection operation such that all fringe points measure their distance above their neighbor's walls consistently with the distance from their own walls. The details of implementing such a fix differ significantly from one domain connectivity code to another. The procedure adopted by the PEGSUS4 code involves using the PROGRD software (*http://www.nas.nasa.gov/~rogers/cgt/doc/progrd.html*) to shift the actual coordinates of each surface fringe point and the line of points immediately above. This requires running PEGSUS4 twice, first to identify the fringe points, and again after running PROGRD. The PEGASUS5 code offers a much cheaper and more automatic solution by incorporating parts of the PROGRD software internally. The interpolation process then uses these projected mesh definitions to determine interpolation coefficients. However, it does not modify the grid coordinates that will be used by the flow solver. A similar approach is adopted by the OGEN code in determining the interpolation coefficients. The method developed by Noack[46] for the BEGGAR code uses what he terms a subcell transfinite interpolation (TFI). This approach applies a standard TFI to the computed projection errors between two overlapping bilinear surfaces. This provides an error correction term for the distance to the wall for any subcell location, and this is used to correct the trilinear interpolation mapping for the fringe points.

5.2 Hole Cutting

Typical situations where holes are required to be cut in volume grids are illustrated in Figs. 28a, 28b and 9a. Fig. 28a shows one plane of a stretched Cartesian box grid which intersects the surface of a wing grid; the box-grid points which lie inside the wing must be cut. Fig. 28b shows a wing and a flap surface, with a spanwise plane from the flap grid which has grid points lying inside the wing. Fig. 9a shows one of the most difficult types of holes to be cut: an intersection of two untrimmed components, in this case, of a wing and fuselage. The figure shows a streamwise plane from the fuselage grid in a region where the wing will have to cut a hole that pierces the fuselage surface.

An important issue with hole-cutting is the optimization of the size of the hole. Ideally, one strives to create overlaps where the sizes of the donor and recipient grid cells are similar. If this is not the case, detailed solution features can be lost when passing interpolated information from a fine grid onto a coarse grid. For example, consider the grids shown in Fig. 28a. If the hole cut in the outer box grid only removed box-grid points which are inside the wing, this would leave box-grid fringe points very near the body surface. These near-surface fringes points would receive interpolated data from the boundary-layer region of the wing. The box grid certainly does not have the resolution required to resolve a boundary layer, and it would thus create an inaccurate solution consisting of a highly dissipated shear layer. If this solution in the box grid is used to update any other interpolation fringe points, such as the outer-boundary points of the wing grid, it will contaminate the entire solution, leading to very poor accuracy. Thus, not only does the hole cutting procedure have to remove grid points which are embedded inside solid surfaces, it must attempt to optimize the location where the overlap between neighboring grids is to occur.

The PEGSUS4 code requires the user to provide a description of all the surfaces that are used to cut each hole. This method is very tedious and error prone, and demands a significant amount of user expertise. The DCF code provides a big improvement over PEGSUS4 with its object X-ray method. Hole cutters for components in the geometry are first identified and object X-rays are built for each hole cutter. The user does have to supply the list of meshes that each object X-ray is allowed to cut, and an offset distance with which to grow each hole away from the body. Efficiency is sometimes gained by breaking a single component object X-ray into multiple X-rays, as is shown in Figs. 29a and 29b. The PEGASUS5 code also offers a significant improvement in the reduction of input required by the user. It contains an automatic hole-cutting algorithm whose input only requires knowledge of the solid-wall boundaries, which must form a closed surface. It also contains an additional hole optimization step, which will effectively grow the holes away from the solid bodies, and optimize the overlap between two or more neighboring grids.[14]

5.3 Fringe Point Interpolation

During this process, fringe points must be identified, and interpolation stencils must be computed. The stencils will be used by the flow solver to update the dependent variables defined at the fringe point using trilinear interpolation. The process of identifying all of the fringe points is straight-forward, and come from two sources. All grid points which are immediately adjacent to a hole point must be classified as fringe

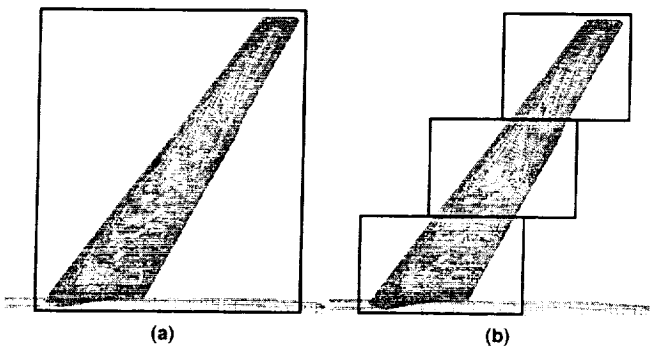

Fig. 29 Object X-rays for swept wing. (a) One X-ray. (b) Multiple X-rays.

points. All mesh outer boundaries which are not part of a solid wall, symmetry plane, periodic boundary, inflow, outflow, or similar boundary condition must also be a fringe point. Identification of such outer boundary points is therefore automatable and has been implemented in recently developed software such as PEGASUS5 and DCF. The code merely requires that the user supplies all of the boundary conditions which are to be given to the flow solver.

Some of the domain connectivity software also allows the user to require that a double layer of fringe points be present at the hole boundaries and the outer boundaries, rather than just a single layer. Using a double layer of fringe points is the recommended practice. This allows the flow solver to retain the same differencing stencil or flux construction for cells for all interior points, including the ones adjacent to the fringe points. With a single layer of fringe points, 4th-order smoothing at points adjacent to a fringe point would have to drop to 2nd-order (for central differencing), and 2nd- or 3rd-order upwind differencing would have to drop to 1st-order. The accuracy of the final solution would then be degraded in regions with non-zero flow gradients. It has also been observed that values of stagnation enthalpy contours in a multi-element airfoil are much closer to one (hence less error) for double fringe versus single fringe. The downside to requiring double fringe points is that it does require more overlap between neighboring grids, and thus requiring more grid points. However, the benefits are typically worth the extra cost.

All of the more recently developed domain connectivity programs utilize some form of tree data structure as part of the procedure for finding fringe-point interpolation stencils. These methods are typically used to find a close starting location for a Newton stencil-walk inversion. The use of these tree structures significantly increases the robustness and efficiency as compared to the Newton stencil-walk method by itself. The PEGASUS5 and the CFDRC codes use an alternating digital tree.[51] The BEGGAR code uses a polygonal mapping tree, which is a combination of the octree and binary space partition (BSP) tree data structures.

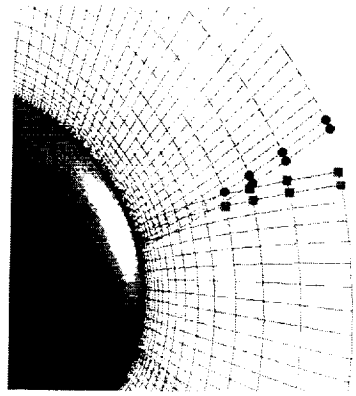

Fig. 30 Orphan points due to inadequate overlap between two volume grids (circle symbols - orphan points from grid 1, square symbols - orphan points from grid 2).

5.4 General Guidelines

No matter which software program is used to perform domain connectivity, the key quality check is to inspect the orphan points before proceeding to the flow solver. Orphan points are present wherever the hole-cutting has failed, and wherever the volume grids do not have adequate overlap. The domain connectivity software should report the number and location of any orphan points. Typically the best method for examining orphan points is to graphically display their location together with a plot of nearby grids planes, or with a representation of the outline of the hole cuts. Both of these plotting functions are implemented in the PLOT3D[17] and OVERGRID[13] codes. An example of this type of plot is given in Fig. 30, which shows some orphan points occurring at the outer boundaries of two meshes due to inadequate overlap. The volume mesh generation of these two meshes did not splay outward enough to maintain their overlap. To fix this problem, the volume grids need to be regenerated such that the overlap is maintained.

Besides inadequate overlap in the original volume grids, the other primary source of orphan points is a failure in the hole-cutting operation. This failure could cause either too small of a hole, resulting in orphan points left inside a solid body, or too large of a hole resulting in not enough overlap left between neighboring meshes. Again, plotting the orphan points and/or the outline of the hole cuts is a good method for determining the cause of the orphan points.

6. Overall Process

As described in the previous sections, overset structured grid generation is a multi-step process involving a variety of software tools where the user needs to provide inputs at each step. It is clearly convenient to have all the necessary tools accessible from one central environment such as a graphical interface where the results from each step are displayed and inspected

prior to moving to the next step. This is especially useful for novice users who are not necessarily familiar with the details of each tool. Examples of such graphical interfaces for overset grid generation include OVERGRID[13] and the OVERTURE suite of tools.[21]

In a design environment, repetitive analyses may be performed with just slight changes in the geometry of the vehicle. It is clearly tedious to repeat the multi-step grid generation process manually for each small design change. The preferred treatment is to record the different steps into a script during the first analysis. Repeated subsequent analyses can then be performed in batch mode with little user's effort in just a small fraction of the first analysis time. Although building the script requires some work and expertise, the advantages usually far outweigh the effort spent. It has been observed that script building is worthwhile even for a one-time analysis of a vehicle. Not only does the script provide a documentation of the entire grid generation process, it also allows for rapid parameter studies such as grid refinement.

Ideally, the typical process script would perform most or all of the following in batch mode: modifying the geometry definition into a form suitable for surface grid generation, generation of surface and volume grids, creation of input files for domain connectivity, flow solver and solution post-processing tools, and running the domain connectivity program. An appropriate scripting language for this task should support modular function calls and floating point arithmetic. For example, Tcl/Tk,[52] Perl,[53] Python[54] are good candidates while Bourne and C shell scripts are not.

Important variables in the process should be parameterized in the script. For geometry processing, key parameters could include geometric attributes such as deflection angles of control surfaces, locations of fins and canards, number of blades in a turbopump component, etc. For surface grid generation, key parameters could include grid spacings, stretching ratios, number of grid points, and marching distances. One could assign a different parameter for each key grid spacing applied to each grid. This would make it quite cumbersome to control grid refinement. A better approach is to use just a small number of independent grid spacing parameters such as the global near-field grid spacing Δs_g (Section 3.4) and a leading edge spacing, etc. Rules could then be established to express grid spacings at other specific locations as a function of the few independent parameters. A single change in the value of these independent parameters would then result in wide spread grid refinement/coarsening.

Another tempting choice for a parameter is the number of grid points applied to a given grid. Unfortunately, a new number of grid points would have to be determined if the physical size of the grid changes, e.g., a longer fuselage would require more points to preserve similar grid quality as a shorter fuselage. A better parameter choice would be the grid stretching ratio which could remain constant over the entire geometry and over different designs. Again a single change in this parameter would propagate grid refinement globally.

For volume grid generation, the typical parameters needed are the normal spacing at the wall, the marching distance, and the stretching ratio in the normal direction. These parameters are usually the same for all near-body volume grids. Also, only a few parameters are usually employed for domain connectivity. These may include the orphan point stencil quality and the number of fringe points (single or double). Sometimes, a few difficult grids may require special treatment such as higher smoothing values for volume grid generation and different hole cutting parameters for domain connectivity.

Many parts of the overset analysis process require the same information from each grid such as grid topology (periodicity, axis, symmetry-plane, wake cut, etc.), location of solid wall boundaries, and flow characteristics (in-flow and out-flow boundaries). It is convenient to define this information just once, then store it in a hash table or array that could be created by a script, or read by the script from a file. Different modules called by the script could then share the same information. This procedure is followed by the configuration automation script system in the Chimera Grid Tools package (*http://www.nas.nasa.gov/~rogers/cgt/doc/man.html*).

The scripting methods discussed here have been applied to a number of complex configurations. These include a subsonic transport with propulsion and high-lift devices[12] and a turbopump with inlet guide vanes, impeller and diffuser. Both scripts were created in under 4 weeks. The procedures handled by the script for the inlet guide vanes of the turbopump are illustrated in Fig. 31. The subsonic transport grid system contains about 23 million grid points while the turbopump contains about 34 million grid points. Surface and volume grids could be regenerated in a workstation in under 20 minutes for the subsonic transport and under 8 minutes for the turbopump. Domain connectivity information could also be regenerated with 32 processors of an SGI Origin in about 15 minutes for both grid systems.

7. Concluding Remarks

The procedures and guidelines described in this paper represent the current practices of experienced users in overset grid generation. With tools evolving further towards automation, it is expected that these procedures will also evolve. Some anticipated future development are discussed below.

Geometry processing continues to consume a large portion of user's time. The effort required can vary significantly from case to case depending on the com-

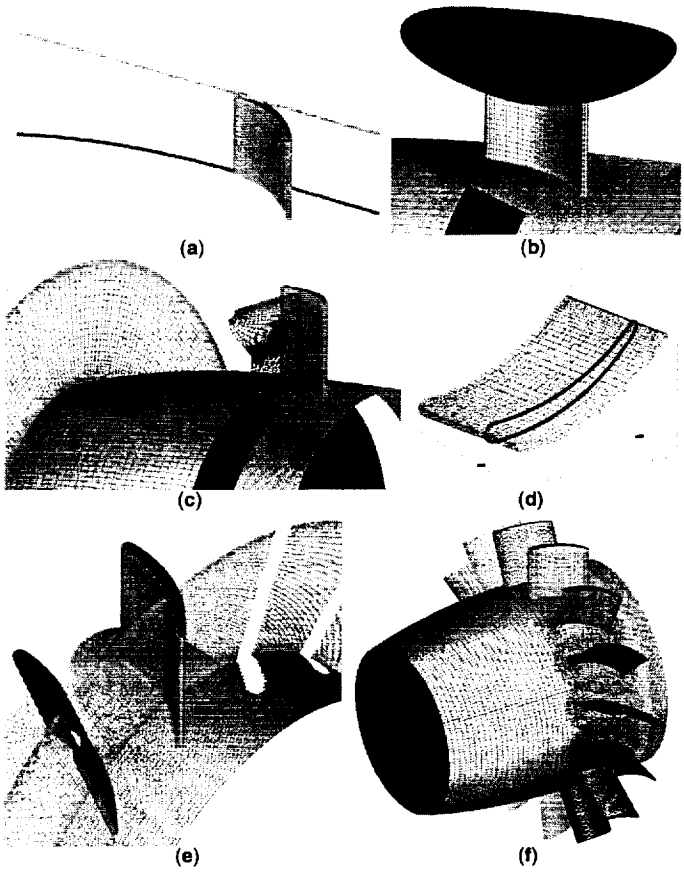

Fig. 31 Automated procedures handled by script for inlet guide vanes of turbopump. (a) Geometry definition. (b) Surface grid for one blade with collar topology. (c) Volume grid slices. (d) Object X-ray for blade. (e) Result of domain connectivity program with input generated by script. (f) Complete 360 degree grid system.

plexity and readiness of the model for grid generation. The best solution here is perhaps to educate the geometry suppliers on the geometry definition requirements for computational analysis purposes. A difficult geometry issue that needs to be automated is the treatment of sliding surfaces. As two surfaces slide over each other, unexposed geometry may become visible and exposed geometry may become retracted. For example, this occurs during the deployment of an aerospace vehicle's control surfaces such as elevons and flaps.

In order for overset technology to be more widely utilized by novice users, the grid generation procedure needs to be more automated. In the time spent after geometry clean-up, surface grid generation and domain connectivity usually dominate the effort required. A typical percentage breakdown in time spent is as follows: surface grid generation (40 - 80%), volume grid generation (1 - 10%), domain connectivity (20 - 50%). Complete automation of surface domain decomposition remains an elusive goal. With more automated surface feature detection, and with the assistance of special tools in a graphical interface, the process may become very fast. New concepts will be explored for automatic geometry coverage using hyperbolically and algebraically grown surface grids. Faster and more robust domain connectivity algorithms will be researched. Tools for rapid creation of scripts for the novice will also be investigated. In the applications area, development of more advanced tools will allow fast input creation and simulations involving multiple rigid or deformable bodies in relative motion.

Acknowledgements

Many of the concepts discussed in this paper are seeded from ideas envisioned by the late Prof. Joseph Steger. After years of development, these ideas have finally taken shape in the form of software tools and have found their way into complex production applications. In addition to drawing from experiences of the authors, the best practices discussed in this paper have also been heavily influenced by other active developers and practitioners of overset technology, in particular, Dr. Robert Meakin of the Army Aeroflightdynamics Directorate and Jeffrey Slotnick of the Boeing Company. The CAD model for the Liquid Glide Back Booster (Fig. 1a) was kindly provided by Bandu Pamadi of NASA Langley Research Center.

References

[1] Steger, J. L., Dougherty, F. C. and Benek, J. A., "A Chimera Grid Scheme," *Advances in Grid Generation*, K. N. Ghia and U. Ghia, eds., ASME FED-Vol. 5, June, 1983.

[2] Buning, P. G., Chiu, I. T., Obayashi, S., Rizk, Y. M., and Steger, J. L., "Numerical Simulation of the Integrated Space Shuttle Vehicle in Ascent," AIAA Paper 88-4359, AIAA Atmospheric Flight Mechanics Conference, Minneapolis, Minnesota, August, 1988.

[3] Gomez, R. J. and Ma, E. C., "Validation of a Large Scale Chimera Grid System for the Space Shuttle Launch Vehicle," AIAA Paper 94-1859, *Proceedings of the 12th AIAA Applied Aerodynamics Conference,* Colorado Springs, Colorado, June, 1994.

[4] Slotnick, J. P., Kandula, M. and Buning, P. G., "Navier-Stokes Simulation of the Space Shuttle Launch Vehicle Flight Transonic Flowfield Using a Large Scale Chimera Grid System," AIAA Paper 94-1860, *Proceedings of the 12th AIAA Applied Aerodynamics Conference,* Colorado Springs, Colorado, June, 1994.

[5] Wulf, A. and Akdag, V., "Tuned Grid Generation with ICEMCFD," *Surface Modeling, Grid Generation, and Related Issues in Computational Fluid Dynamic (CFD) Solutions*, NASA Lewis Research Center, NASA CP-3291, 1995, pp. 477–488.

[6] Chan, W. M. and Steger, J. L., "Enhancements of a Three-Dimensional Hyperbolic Grid Generation Scheme," *Appl. Math. & Comput.*, Vol. 51, 1992, pp. 181–205.

[7] Chan, W. M., Chiu, I. T. and Buning, P. G., "User's Manual for the HYPGEN Hyperbolic Grid Generator and the HGUI Graphical User Interface," NASA TM 108791, October, 1993.

[8] Suhs, N. E. and Tramel, R. W., "PEGSUS 4.0 User's Manual," AEDC-TR-91-8, AEDC/PA, Arnold Air Force Base, Tennessee, 1991.

[9] Chan, W. M. and Buning, P. G., "Surface Grid Generation Methods for Overset Grids," *Computers and Fluids*, Vol. 24, No. 5, 1995, pp. 509–522.

[10] Parks, S. J., Buning, P. G., Steger, J. L. and Chan, W. M., "Collar Grids for Intersecting Geometric Components Within the Chimera Overlapped Grid Scheme," AIAA Paper 91-1587, *Proceedings of the 10th AIAA Computational Fluid Dynamics Conference*, Honolulu, Hawaii, 1991.

[11] Blake, M. W., Kerr, P. A., Thorp, S. A. and Chou, J. J., "NASA Geometry Data Exchange Specification for Computational Fluid Dynamics (NASA IGES)," NASA Reference Publication 1338, April, 1994.

[12] Rogers, S. E., Roth, K., Nash, S. M., Baker, M. D., Slotnick, J. P., Whitlock, M. and Cao, H. V., "Advances in Overset CFD Processes Applied to Subsonic High-Lift Aircraft," AIAA Paper 2000-4216, 18th AIAA Applied Aerodynamics Conference, Denver, Colorado, August, 2000.

[13] Chan, W. M., "The OVERGRID Interface for Computational Simulations on Overset Grids," AIAA Paper 2002-3188, 32nd AIAA Fluid Dynamics Conference, St. Louis, Missouri, June, 2002.

[14] Suhs, N. E., Rogers, S. E., and Dietz, W. E., "PEGASUS 5: An Automated Pre-processor for Overset-Grid CFD," AIAA Paper 2002-3186, 32nd AIAA Fluid Dynamics Conference, St. Louis, Missouri, June, 2002.

[15] Meakin, R. L., "Object X-rays for Cutting Holes in Composite Overset Structured Grids," AIAA Paper 2001-2537, 15th AIAA Computational Fluid Dynamics Conference, Anaheim, California, June, 2001.

[16] Farouki, R. T., "Closing the Gap Between CAD Model and Downstream Application," SIAM News, Volume 32, Number 5, June, 1999 (*http://www.siam.org/siamnews/06-99/cadmodel.htm*).

[17] Walatka, P. P., Buning, P. G., Pierce, L., and Elson, P. A., "PLOT3D User's Manual, Version 3.6," NASA TM 101067, Jan. 1990.

[18] Walatka, P. P., Clucas, J., McCabe, R. K., Plessel, T. and Potter, R, "FAST User Guide," RND-93-010, NASA Ames Research Center, 1994.

[19] Gridgen User Manual, Version 13.3, Pointwise, Inc., Fort Worth, TX, 2000.

[20] Steinbrenner, J., Wyman, N. and Chawner, J., "Development and Implementation of Gridgen's Hyperbolic PDE and Extrusion Methods," AIAA paper 2000-0679, Reno, NV, January, 2000.

[21] Brown D. L., Henshaw, W. D. and Quinlan D. J., "Overture: An Object Oriented Framework for Solving Partial Differential Equations on Overlapping Grids," Object Oriented Methods for Interoperable Scientific and Engineering Computing, SIAM, 1999, pp. 245-255 (*http://www.llnl.gov/CASC/Overture/*).

[22] Henshaw, W. D., "An Algorithm for Projecting Points Onto a Patched CAD Model," *Proceedings of the 10th International Meshing Roundtable,* Newport Beach, California, October, 2001 (*http://www.andrew.cmu.edu/user/sowen/imr10.html*)

[23] Petersson, N. A. and Chand, K. K., "Detecting Errors in CAD Surfaces and Repairing Geometries for Mesh Generation," *Proceedings of the 10th International Meshing Roundtable,* Newport Beach, California, October, 2001 (*http://www.andrew.cmu.edu/user/sowen/imr10.html*).

[24] Cosner, R. R., "Future Requirements in Surface Modeling and Grid Generation," *Proceedings of the Workshop on Surface Modeling , Grid Generation and Related Issues in Computation Fluid Dynamic (CFD) Solutions,* NASA CP3291, NASA Lewis Research Center, May 1995.

[25] Mezentsev, A. A., "Methods and Algorithms of Automated CAD Repair For Incremental Surface Meshing," *Proceedings of the 8th International Meshing Roundtable,* South Lake Tahoe, California, U.S.A., October, 1999, pp. 299–309.

[26] Jones, M. R., Price, M. A. and Butlin G., "Geometry Management Support for Auto-Meshing Proceedings," *Proceedings of the 4th International Meshing Roundtable,* Sandia National Laboratories, October, 1995, pp. 153–164.

[27] Akdag, V., Magnuson, A. and Wulf, A. , "Efficient Integration of CFD into Product Design," The 9th Annual Thermal Fluids Analysis Workshop, NASA Glenn Research Center, August 1998 .

[28] Haimes, R., and Follen, G., "Computational Analysis PRogramming Interface," *Proceedings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations,* Eds. Cross, Eiseman, Hauser, Soni and Thompson, July 1998.

[29] Chan, W. M. and Gomez, R. J., "Advances in Automatic Overset Grid Generation Around Surface Discontinuities," AIAA Paper 99-3303, in *Proceedings of the AIAA 14th Computational Fluid Dynamics Conference,* Norfolk, Virginia, June, 1999.

[30] Chan, W. M. and Meakin, R. L., "Advances Towards Automatic Surface Domain Decomposition and Grid Generation for Overset Grids," AIAA Paper 97-1979, *Proceedings of the AIAA 13th Computational Fluid Dynamics Conference,* Snowmass, Colorado, June, 1997.

[31] Jiang, F., An, M. Y., and Mysko, S. J., "Computational Analysis of Aileron Effectiveness Characteristics on an Advanced Wing," AIAA Paper 2000-4324, AIAA 18th Applied Aerodynamics Conference, Denver, Colorado, August, 2000.

[32] Vassberg, J. C., Buning, P. G., and Rumsey, C. L., "Drag Prediction for the DLR-F4 Wing/Body Using OVERFLOW and CFL3D on an Overset Mesh," AIAA Paper 2002-0840, AIAA 40th Aerospace Sciences Meeting, Reno, Nevada, January, 2002.

[33] Thompson, J. F., Warsi, U. A. and Mastin, C. W., "Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations – A Review," *J. Comp. Phys.*, Vol. 47, 1982, pp. 1–108.

[34] Okamoto, K. K., Klopfer, G. H. and Chattot, J. J., "Assessing Grid Quality of Structured Meshes by Truncation Error Analysis," in *Proceedings of the 15th International Conference on Numerical Methods in Fluid Dynamics*, Monterey, California, 1996, pp. 340–345,

[35] Haimes, R., Giles, M. and Darmofal, D., "Visual3 - A Software Environment for Flow Visualization," Computer Graphics and Flow Visualization in Computational Fluid Dynamics, VKI Lecture Series No. 10, Brussels, September, 1991.

[36] Legensky, S. M., "Recent Advances in Unsteady Flow Visualization," AIAA Paper 97-2087, *Proceedings of the 13th AIAA Computational Fluid Dynamics Conference,* Snowmass Village, Colorado, June, 1997.

[37] Buning, P. G., Jespersen, D. C., Pulliam, T. H., Klopfer, G. H., Chan, W. M., Slotnick, J. P., Krist, S. E. and Renze, K. J., "OVERFLOW User's Manual," Version 1.8q, August, 2000.

[38] White, Frank M., *Viscous Fluid Flow*, McGraw Hill, Inc., New York, 1974.

[39] Sommer, S. C., and Short, B. J., "Free-Flight Measurements of Turbulent Boundary Layer Skin Friction in the Presence of Severe Aerodynamic Heating at Mach Numbers from 2.8 to 7.0," *Journal of the Aeronautical Sciences*, Vol. 23, No. 6, June, 1956, pp. 536–542.

[40] Lawrence, S., "OVERFLOW: Facts on Friction," 1999 NASA High-Speed Research Program Aerodynamic Performance Workshop, Volume I–Configuration Aerodynamics, NASA/CP-1999-209704/VOL1/PT1, D.E. Hahne, Ed., Dec. 1999, pp. 401–416.

[41] Rogers, S. E., Roth, K., Cao, H. V., Slotnick, J. P., Whitlock, M., Nash, S. M. and Baker, M. D., "Computation of Viscous Flow for a Boeing 777 Aircraft in Landing Configuration," AIAA Paper 2000-4221, 18th AIAA Applied Aerodynamics Conference, Denver, Colorado, August, 2000. Published in *Journal of Aircraft*, Vol. 38, No. 6, Dec. 2001, pp. 1060–1068.

[42] Rogers, S. E., " Progress In High-Lift Aerodynamic Calculations," AIAA Paper 93-0194, Jan. 1993. Published in *Journal of Aircraft*, Vol 31, No. 6, Nov. 1994, pp. 1244–1251.

[43] Meakin, R. L., "Automatic Off-Body Grid Generation for Domains of Arbitrary Size," AIAA Paper 2001-2536, 15th AIAA Computational Fluid Dynamics Conference, Anaheim, California, June, 2001.

[44] Maple, R. C. and Belk, D. M., "Automated Set Up of Blocked, Patched, and Embedded Grids in the Beggar Flow Solver," in *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, N.P. Weatherill, ed, 1994, Pine Ridge Press, pp. 305-314.

[45] Belk, D. M. and Maple, R. C., "Automated Assembly of Structured Grids for Moving Body Problems," AIAA Paper 95-1680, 12th AIAA Computational Fluid Dynamics Conference, June 1995.

[46] Noack, R. W., "Improved Interpolation for Viscous Overset Grids," AIAA Paper 98-0199, Jan. 1997.

[47] Prewitt, N. C., Belk, D. M., and Shyy, W., "Parallel Computing of Overset Grids for Aerodynamic Problems with Moving Objects," Progress in Aerospace Sciences, Vol. 36, No. 2, 2000, pp. 117-172.

[48] Petersson, N. A., "Hole-Cutting For Three-dimensional Overlapping Grids," SIAM J. Sci Comput., Vol. 21, No. 2, 1999, pp. 646–665.

[49] Henshaw, W. D., "Ogen: An Overlapping Grid Generator for Overture", Lawrence Livermore National Laboratory Research Report, UCRL-MA-132237, 1998.

[50] Wang, Z. J., Parthasarathy, V., and Hariharan, N., "A Fully Automated Chimera Methodology for Multiple-Moving Body Problems," AIAA Paper 98-0217, Jan. 1998.

[51] Bonet, J. and Peraire, J., "An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems," International J. Numerical Methods in Engineering, Vol. 31, 1991, pp. 1–17.

[52] Welch, B., *Practical Programming in Tcl and Tk*, 3rd ed., Prentice Hall, 1999.

[53] Wall, L., Christiansen, T. and Orwant, J., *Programming Perl*, 3rd ed., O'Reilly, 2000.

[54] Lutz, M., *Programming Python*, 2nd ed., O'Reilly, 2001.