Automatic Block-Structured Grid Generation — Progress and Challenges

John F. Dannenhoffer, III
Computational & Design Methods Group
United Technologies Research Center
East Hartford, CT 06108
dannenho@percheron.res.utc.com

Abstract

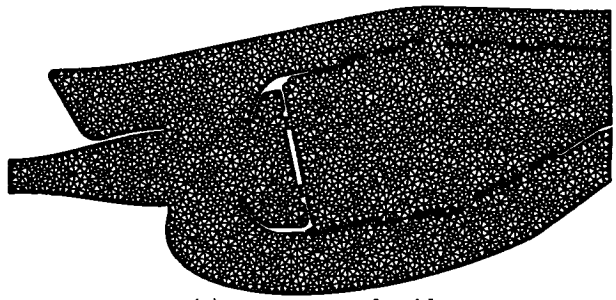
One of the most labor-intensive aspects of performing computational simulations of the flow over aerospace configurations is the design and generation of appropriate computational grids, especially of the block-structured variety. A major difficulty with current systems is the design of a suitable blocking plan (or flowfield decomposition) for an arbitrary configuration. This paper presents an integrated approach to the blocking design problem which is comprised of three key technologies:

- a set of procedures which automatically convert an abstract topological specification into a real block-structured grid;
- a rule-based expert system which controls the blocking process, based upon expertise garnered from a variety of block-structuring experts; and
- a nonlinear, integer optimization technique which is used to "fine-tune" the blocking plan and the resulting computational grid.

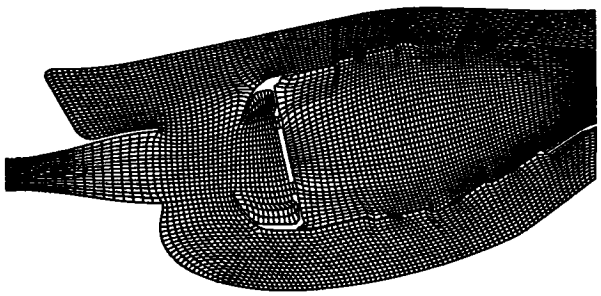
It is shown that the combination of these three technologies makes it possible to efficiently generate near-optimal block-structured grids for previously-studied classes of multi-body configurations. The issue of extending the knowledge base to *arbitrary* multi-body configurations is explored, with special emphasis placed on the challenges posed by three-dimensional geometries.

Introduction

The pacing item in the timely performance of a computational simulation of the flow over an aerospace configuration is frequently the design and generation of an appropriate computational grid. In fact, grid generation typically accounts for more than half of the labor hours spent on a typical configuration. There are currently two major approaches to circumventing this bottleneck, namely the use of unstructured-tetrahedral grids (Fig. 1(a)) and the use of block-structured grids (Fig. 1(b)). The former have the advantage of being simpler to generate; however, flow solution schemes which use these grids do not currently produce solutions of comparable quality (and with comparable computational efficiency) to the solutions produced by block-structured flow solvers. On the other hand, quite accurate flowfield solutions have been produced with block-structured grids; the difficulty with them is the amount of labor which is required to produce the grids, especially for complex configurations. As a result, a long term goal of many CFD researchers has been the creation of techniques which automatically produce high quality block-structured grids over arbitrary three-dimensional configurations. The work presented here represents one of the building blocks which are necessary to reach that long term goal.



(a) unstructured grid



(b) block-structured grid

Figure 1: Gas turbine combustor configuration.

One of the first attempts to automatically design and implement blocking plans, or flowfield decompositions, was the pioneering efforts of A. Vogel (Vogel 1989), in which she used an expert system to capture grid blocking expertise for a wide variety of two-dimensional aerospace configurations. For the configurations which she tested, the grids which were generated were of a very high quality. However, one of the major limitations of her first-generation system was the amount of (and types of) information which the user had to supply in order to generate a grid; the information was of the type that the system had difficulty generalizing its blocking expertise to problems which were somewhat-similar to those for which it had previously generated grids.

In 1988, S. Alwright reported on an ingenious method through which block-structured grids could be specified (Alwright 1988). The essence of his idea was to treat block-structuring abstractly, through the use of "wire-frame schematics"; in this way the blocking topology could be specified somewhat independently of the actual geometry. An additional benefit of this new technique was that the block-structuring could be described with significantly fewer pieces of information than was traditionally required.

These two advances, along with some recent work in design optimization (Tong & Gregory 1990), formed the impetus for the current work. The objective of the research described herein was to develop a second-generation system for the automatic design and generation of block-structured grids about arbitrary configurations.

Progress

A system for the automatic blocking of complex two-dimensional aerodynamic configurations has been developed and fielded at a variety of sites. The system is built upon an *interactive* blocking system which is described fully in (Dannenhoffer 1991a); the *automatic* system is described fully in (Dannenhoffer 1991b).

Interactive system

The basis of the interactive blocking system is the specification of the multi-block topology on a *topology plane*, which is actually an abstraction of the physical plane, on which one can specify the exact topology (not shape) of the grid lines in the final multi-block grid. It can be thought of as either the "sketch" which a user usually creates when designing a blocking scheme or as a "squared-up" representation of the given configuration and of the computational grid which is to be produced. The topology plane and a blocking specification on it are composed of a number of components which are described in the following paragraphs. Figure 2 gives an example of the topology plane for a CH-grid around an isolated airfoil.

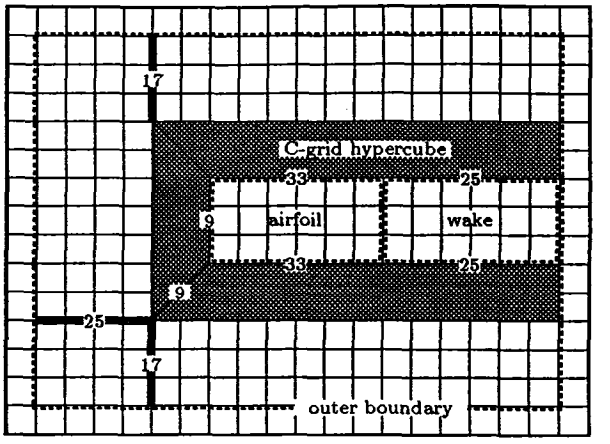


Figure 2: Example of the topology plane for a CH-grid around an isolated airfoil.

Background Grid — The entire topology plane is covered by an integer Cartesian grid called the background grid, which can be viewed as a piece of graph paper on which the other components are drawn. Specifically, the placement of the other components are constrained such that their corners and edges lie along background grid lines. This requirement makes it possible to quickly and definitively determine the relative locations of the various components. Another significant feature of the background grid is that it serves to describe the default grid line directions; that is, unless otherwise directed, grid lines (or more correctly, their abstractions) run horizontally and vertically throughout the domain.

Bounds — Bounds, which are drawn as dashed rectangles in the current implementation, correspond to "squared-up" representations of each of the fixed geometric entities of the given configuration in the physical plane. Examples of these include the outer boundary and the airfoil (or other body over which a grid is required); also, a grid line can be constrained to lie along a prescribed path (for example, along a wake line) by representing the wake as a bound. As shown in Figure 2, appropriate placement of bounds on the background grid can convey the fact that the airfoil and its wake both fall completely within the outer boundary and that the wake connects the downstream edge of the airfoil with the downstream edge of the outer boundary.

Hypercubes — Hypercubes are the topology plane components which allow one to "wrap" grid lines around other components, that is, they are the topology plane component which alters grid line directions from the default horizontal and vertical directions. Hy-

percubes are represented on the topology plane as two nested rectangles, with diagonal lines connecting the inner and outer rectangles (where needed). The diagonal corners of the hypercube serve to "bend" one family of grid lines around the center of the hypercube, and "deflects" the other family. In Figure 2, the only hypercube contained in the figure represents a C-grid which wraps grid lines around both the airfoil and its wake.

Sizes — In order to specify the number of grid lines in each region, sizes are added to the topology plane (as shown by the numbers in Figure 2). These sizes specify, for example, that the wrapping C-grid contains nine circumferential lines and that there are 25 lines normal to both the upper and lower sides of the wake.

Rulers — Sometimes it is impossible to establish the number of grid lines in a region simply by sizing bounds and hypercubes; in these cases another topology plane component, called a ruler, is needed. A ruler is simply a device which specifies the number of grid lines between its two endpoints. Rulers are shown in the current implementation as a set of double lines. Figure 2 contains three rulers which specify that there are 17 "horizontal" lines passing above and below the wrapping C-grid and that 25 "vertical" lines pass in front of the wrapping C-grid.

Paths — Finally, to connect the abstraction in the topology plane to the real geometry, paths are used. Each path which is drawn in the topology plane corresponds to exactly one body (or wake or outer boundary) of the given configuration. For clarity, paths are not shown in Figure 2.

Once the grid topology is specified on the topology plane, a set of procedures, which are fully described in (Dannenhoffer 1991a), are used to generate a suitable assembly of grid blocks. The key steps include the transformation of hypercubes to grid blocks, filling the remaining regions, establishment of one-to-one face matches, determination of the sizes for each block, elimination of degenerate blocks, and conglomeration of grid blocks (based upon some user-specified criterion). Once the grid blocks and their interconnections have been automatically created, multi-block transfinite interpolation and elliptic grid generation schemes are used to generate a smooth computational grid.

The main advantage of using this system for blocking, or domain decomposition, is that the number of *inputs* which are required to generate a multi-block grid is relatively small, at least as compared with other multi-block systems (Thompson et al. 1988, Steinbrenner, Chawner & Fouts 1989, and Sorenson 1989), This is an important consideration because the complexity of any reasoning process scales with at least the number of independent variables. Hence, an automatic reasoning system which uses the current grid generation system can be built more easily since it requires fewer pieces of expertise, that is, fewer "rules-of-thumb."

Automatic system

In the above discussion, phrases such as "which allow one to specify" are used to describe the interactive system. In the automated system, a forward-chaining expert system is actually used to analyze the configuration, "draw" the appropriate topological representation of the configuration, and specify a suitable grid topology.

The first step in generating the domain decomposition knowledge base was to determine the basic problem-solving paradigm which grid experts use in designing block-structured grids. After discussions with many block-structuring experts, it was determined that blocking design is most often done first from the bodies (airfoils) out, then from the outer boundary in, and finally by connecting the two; any discrepancies which arise are handled in a clean-up step at the end. This led to the overall design of the knowledge base used here.

The automatic design system begins by determining the type of grid topology which is required (desired) in the vicinity of each body. In order to do this, the bodies are first classified based upon local geometric metrics. The classified bodies are then matched against *templates* of previously-solved cases, yielding near-body designs which are scheduled to be placed on the topology plane. The far-field grid topology is then determined in a similar manner through matches with a set of far-field templates.

The most difficult step was next, namely that of *placing* the near-body templates onto the topology plane such that reasonable grid line connections are made. The basic strategy used here is to place the near-body templates in approximately the same relative position as the positions of the bodies in physical space. For example, if one body is centered above the trailing edge of another body in physical space, its near-body template is placed in the topology plane so as to be centered above the trailing edge of the other body's near-body template. This results in about 16 different placement arrangements (for two simple bodies in two dimensions) which have to be considered in the knowledge base.

The final step involved the selection of the number of computational points in (or sizes of) each topological region. This process is carried out in two stages. In the first, a nominal number of grid points are assigned to each region based upon experience gained with generating grids for a variety of configurations. Attached to each of these sizes is a range of permissible values and a confidence factor which expresses the relative confidence that the nominal value should be used in the

final grid. In the second stage, all those sizes with confidence levels which are less than some specified threshold value are taken as the independent variables in a nonlinear optimization, whose objective function is to maximize the grid quality while constraining the total number of surface and field points. By systematically adjusting the confidence threshold level, the number of independent variables which the optimizer has to vary can be controlled, yielding a near-optimal grid with a minimum of cost.

The nonlinear integer optimization technique used herein is an adaptation of the flexible polyhedron/tolerance method (Paviani & Himmelblau 1969). The technique is based upon the flexible polyhedron technique (Nelder & Mead 1965), which marches a polyhedron (for example, a triangle for two independent variables) downhill toward the minimum of the objective function. Successive iterations are taken by repeated application of reflection, expansion, and contraction operators. The major advantage of this method, as compared with more traditional conjugate-gradient approaches, is that the flexible tolerance method does not require explicit computations of gradients—which are difficult to determine is cases where the independent variables can only take on integer values (for example, the number of grid points in various grid blocks).

System Demonstration

As a proof-of-concept demonstration, consider a biplane of `NACA0012` airfoils which are staggered such that the leading edge of the upper airfoil is one-half chord above and behind the leading edge of the lower airfoil. Both airfoils are well classified as `blunt-sharp`, meaning that their leading edges are blunt and their trailing edges are sharp. According to one grid expert (Sorenson 1990), the "best" computational grids for bodies of this type are generally `C-type`; this expertise is built into the knowledge base (for subsequent automatic execution). In addition, a `bullet-shaped` outer boundary was selected as being typical of the type used to solve such problems.

Based upon the above considerations, the computer-aided blocking system generated the topological specification shown in Figure 3, which is a four block solution to this problem. The resulting computational grid is shown in a closeup view in Figure 4. Notice that the arbitrary decision to make the wakes of equal length (and thus connect the airfoil trailing edges with a grid line) results in significant skew. The nonlinear optimizer is then used to adjust the number of grid points which are associated with each wake, and as a result the number of grid lines which lie in the overlap region between the two airfoils, yielding the "optimal" grid shown in two views in Figure 5. The total time required to produce this result was less than two minutes on an engineering workstation.

A wide variety of configuration changes were

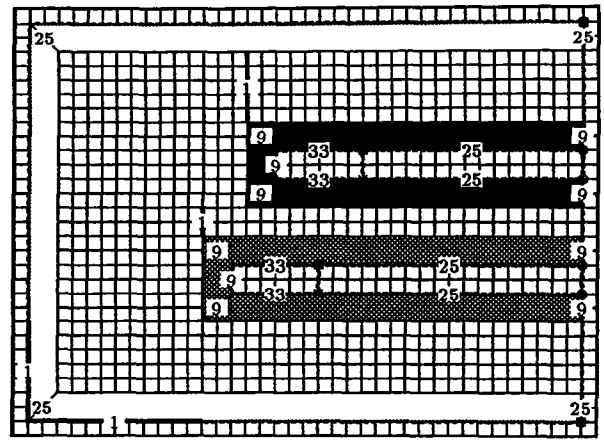


Figure 3: Topology plane for grid around a staggered biplane.

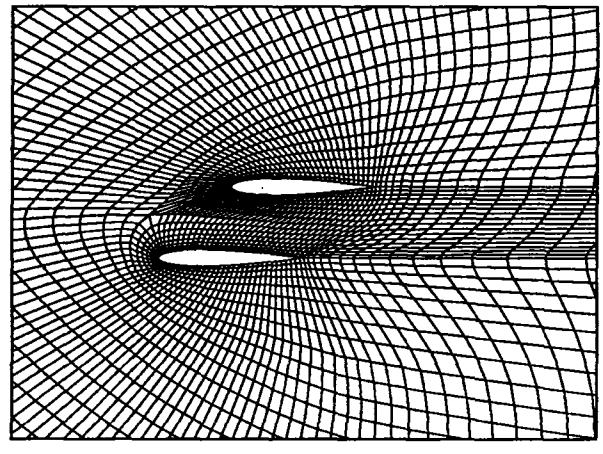
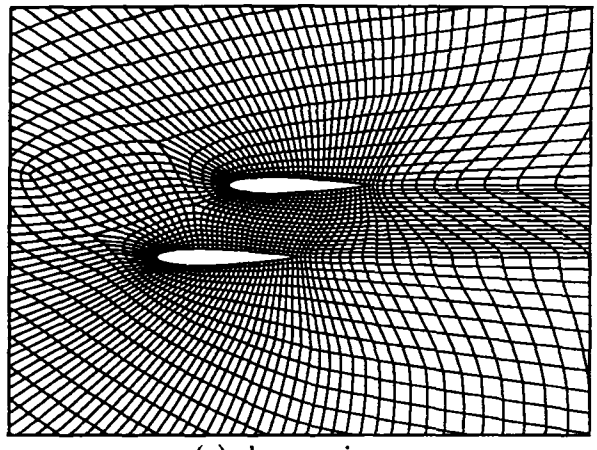


Figure 4: Close-up of nominal (non-optimized) grid around a staggered biplane.

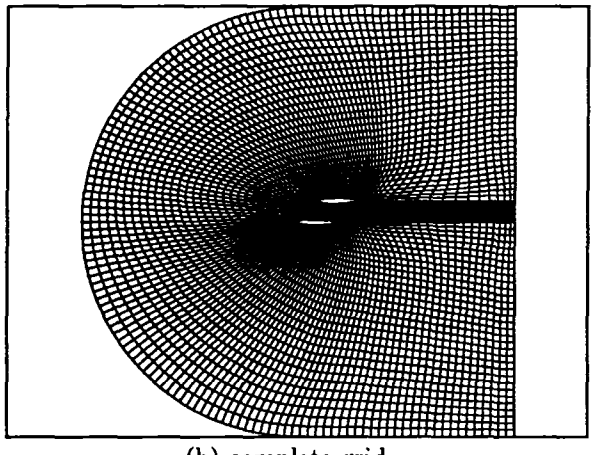
made in (Dannenhoffer 1991b) to demonstrate the configuration-sensitivity of the current scheme. These changes included both changes in body shapes, outer boundary shapes, and body placement; in all cases, satisfactory grid resulted. Other unpublished demonstrations have been made for other configurations types, including forced-mixer geometries and various nozzle geometries, with similar good grids resulting.

Challenges

The extension of the present technique to a wider variety of geometric configurations requires that the "blocking expertise" in the knowledge base be augmented by rules which are appropriate to the new configuration type. This is a particularly difficult task, especially in three dimensions. The major challenges



(a) closeup view



(b) complete grid

Figure 5: Optimized computational grid around a staggered biplane.

which one encounters include:

- Recall from above discussion that the basic design approach is a "divide-and-conquer" method which requires that the system be able to automatically decompose a configuration into its components. For example, the system must be able to divide a complete aircraft (which is defined in terms of a collection of surface patches) into a fuselage, wing, tail, engine inlet, etc. This task is made particularly difficult by the fact that appropriate break points are generally not well defined by geometric (or slope) discontinuities; often they can only be found by simultaneously considering the configuration from a variety of length scales. In other words, the system needs to understand the intent of the configuration in order to generate a good decomposition.
- Another challenge is posed by the difficulty of describing the "shape" of each component. For example, a reasonable description for a simple wing is that it has a rounded leading edge, a sharp trailing edge, that the wing tip is flat, and that it intersect the fuselage at its root. It is clear that for more complex wings (or any other component for that matter), a suitable description language needs to be developed to adequately describe all of a configurations geometric features. Also, the description language has to capable of treating components which intersect in any way, as well as components which self-intersect and contain degeneracies.
- Also, as is the case for most automatic reasoning systems, the grid system needs to degrade gracefully, that is it needs to do reasonable things (or alert a user) when it encounters situations which outside its domain of expertise. Since it is expected that a completely automatic system for completely arbitrary configurations is not achievable in the next five years, the grid system will need to work cooperatively with a user, providing as must assistance as possible.

References

Alwright, S. E. 1988, Techniques in Multiblock Domain Decomposition and Surface Grid Generation, in *Numerical Grid Generation in Computational Fluid Mechanics '88*, Pineridge Press Limited.

Dannenhoffer, J. F. 1991a, A Block-Structuring Technique for General Geometries, AIAA-91-0145.

Dannenhoffer, J. F. 1991b, Computer-Aided Block-Structuring Through the Use of Optimization and Expert-System Techniques, AIAA-91-1585.

Nelder, J. A. and Mead, R. 1964, A Simplex Method for Function Minimization, *Computer Journal*, Vol 7, pp 308.

Paviani, D. A., and Himmelblau, D. M., 1969, Constrained Nonlinear Optimization by Heuristic Programming, *Operations Research*, Vol 17, No 5, pp 872–882.

Sorenson, R. L. 1989, The 3DGRAPE Book: Theory, Users' Manual, Examples, NASA-TM-102224.

Sorenson, R. L. 1990, personal communication.

Steinbrenner, J. P., Chawner, J. R., and Fouts C. L. 1989, A Structured Approach to Interactive Multiple Block Grid Generation, AGARD FDP Specialist Meeting in Loen, Norway.

Thompson, J. F., Lijewski, L. E., and Gatlin, B. 1988, Program EAGLE User's Manual, AFATL-TR-88-117 (Volumes I, II, and III).

Tong, S. S. and Gregory, B. A. 1990, Turbine Preliminary Design Using Artificial Intelligence and Numerical Optimization Techniques, ASME 90-GT-148.

Vogel, A. A. 1989, A Knowledge-Based Approach to Automatic Flow-Field Zoning for Computational Fluid Dynamics, NASA TM-101072.