A Three-Dimensional, Block-Structured Finite Element Grid Generation Scheme

A. Ecer,* J. Spyropoulos,† and J. D. Maul†
Purdue University at Indianapolis, Indiana

A finite element grid generation scheme is presented. This scheme was developed particularly for generating computational grids for complex three-dimensional configurations. It provides simple numerical tools for automatically generating grids for a three-dimensional geometry. It also allows the user to control the description of appropriate grids for particular flow regions. It is based on a block structure for both the definition of the geometry and the generation of the grid. The user constructs the grid gradually with increasing details and can interactively make modifications on an existing grid with minimum restrictions. In this paper, the applications of the developed grid generation scheme are provided for a wing-body combination and for a more complex aircraft configuration.

I. Introduction

THE generation of a computational grid around a three-dimensional aircraft geometry requires many important considerations:

1) The geometry of the solid body has to be defined in great detail, since at some critical regions many grid points have to be placed for appropriate modeling of the flow. On the other hand, for some other regions, only a coarse definition of the geometry may be sufficient. Grid points should be located exactly on the surface boundaries without much effort and with the desired accuracy.

2) The user should be able to define appropriate computational grids for different flow regions around the body. There should not be any undue restrictions in joining these different regions of the grid. Also, the operation of combining different types of grids should not produce highly distorted elements.

3) It is almost impossible to design a good grid with a single attempt. The user must be able to modify a particular flow region without excessive effort in terms of remodeling that region and renumbering the remaining portions of the grid.

4) The user should be able to generate the grid as automatically as possible but should not sacrifice control for the sake of automation. For any given automatically generated grid, the user will want to make changes in small groups of grid points and should be able to perform such tasks in an interactive environment.

5) The grid generation scheme should provide a minimum number of grid points for a particular level of accuracy. When three-dimensional flows are modeled, the number of grid points increases dramatically with every attempt at grid refinement. One should be able to design grids in which the grid points are distributed in an optimum fashion, again without undue restrictions.

Finite element grid generation techniques have been developed over the last fifteen years. These techniques have been widely used for solving three-dimensional problems in a design environment, mostly for solid mechanics problems. A review of such schemes can be found in Ref. 1. In the present paper, a finite element grid generation scheme is presented which was developed particularly to analyze transonic potential flows around three-dimensional bodies. An attempt was made to provide a grid generation scheme which offers solutions to some of the problems discussed above. One of the main advantages of the finite element method is its generality in designing a computational grid for an irregular geometry with minimal restrictions. This advantage was utilized in the present grid generation scheme to provide more flexibility in generating new grids or modifying existing ones. Also, it is shown that by using existing geometric modeling capabilities, which have been developed during the past few years, the finite element grid generation scheme becomes very effective for modeling complex three-dimensional flows.

II. Finite Element Grid Generation Scheme

The finite element grid generation scheme presented herein utilizes a multiple block structure. The main objective of the present development was to provide the user with a convenient tool for designing three-dimensional computational grids in three dimensions in a structured fashion. The grid generation is performed in two levels. First, each of these blocks is defined as a superelement, which is initially described by a single isoparametric finite element. Then, appropriate grids are defined for each of these blocks, which are connected automatically. In this section, the basic characteristics of the grid generation scheme are described using simpler two-dimensional examples for clarity. It is assumed that one can easily visualize these simplified examples in three dimensions.

Isoparametric finite elements are commonly used in finite element analysis in describing irregular geometries.[2] One uses a local transformation for each element that transforms a parent element, e.g., a cube to a curved isoparametric element, as shown in Fig. 1.

For an isoparametric element, the position of a point inside an element can be defined in terms of the nodal coordinates as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = N_i(\zeta,\eta,\xi) \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$$

where N_i is called a shape function. These shape functions are written using simple polynomials, where the degree of the

Presented as Paper 84-0004 at the AIAA 22nd Aerospace Sciences Meeting, Reno, NV, Jan. 9-12, 1984; received Nov. 9, 1984. Copyright © American Institute of Aeronautics and Astronautics, Inc., 1985. All rights reserved.

*Professor. Member AIAA.

†Research Assistant. Student Member AIAA.

polynomial depends on the number of grid points for an element. For the elements shown in Figs. 2a-c, linear, quadratic, and cubic polynomials are employed, respectively. Figure 2d shows an element where different numbers of grid points and different order polynomials are defined along each edge of an element. The term isoparametric originated when the same polynomial employed for the geometric transformation was employed for approximating the variation of the flow variables over the element.[2]

Definition of a Block Structure for Grid Generation

The user employs several blocks of elements (superelements), initially to obtain a coarse description of the geometry and to control the design of the mesh. For a simple example shown in Fig. 3, the two-dimensional flowfield is divided into six superelements to roughly describe the geometry and the grid generation procedure. In describing the geometry, the following capabilities are provided:

1) Linear, parabolic, or cubic shape functions can be employed to describe each of the edges and to provide a close approximation of the block boundaries. For example, in Fig. 3, blocks II and V have linear, parabolic, and cubic edges, depending on the complexity of the geometry required to describe each edge of the block. The user works with this approximate geometry to design the grid. The accurate geometry of the boundaries is later added to the model.

2) Cubic shape functions are basically employed to align the grids. For example, by using a cubic edge between blocks I and II, one can generate a curved edge between these two blocks and force the grid lines to be generated normal to the surface of the body.

3) Besides the standard isoparametric block with six surfaces, one can use wedge-type blocks as shown in Fig. 4a. Along the edges, these blocks can again have linear, parabolic, or cubic representations. These blocks can again be employed to model irregular geometries and also to generate grids that do not correspond to a regular rectangular grid with $m \times n$ grid points. These wedge-type blocks are generated by collapsing the edges of the regular blocks. A common application of wedge-type blocks is shown in Fig. 4b, where two regular blocks are collapsed into two wedge-type blocks.

4) The blocks can be coupled together after they are defined as separate blocks. One can still generate the grid independently for each block, and the coupling is performed automatically. For the case shown in Fig. 4b, the grid generation is defined for two blocks separately by the user and coupled automatically.

5) Finally, another capability is the placement of cavities between two neighboring blocks as shown in Fig. 4c. Blocks II and V in Fig. 3 are separated by a cavity. A user might generate a series of blocks and then introduce cavities at a later time. This provides an efficient tool for implementing modifications on existing grids. The user may start with a basic geometry and later add details by using this capability, which is also employed to introduce cuts in the grid for modeling the sharp trailing edges of the wings.

By using all of the above capabilities, the user will be working with a simple rectangular block structure of 2×3 blocks to generate the system shown in Fig. 3. The advantages of such a capability for treating complex aircraft geometries are described in this paper.

Generation of Grids for Individual Blocks

Once the geometry is roughly defined by the blocks, the objective is to generate grids for each of the individual blocks. For the grid generation scheme, the same shape functions as those in Eq. (1) are employed. A regular grid is transformed to the grid shown in Fig. 5 by using the polynomials defined in Eq. (1). Again, these shape functions are used for each block to define the location of the grid points.

Although the above transformation is straightforward, the finite element method provides certain flexibilities to this procedure:

1) Both regular blocks or wedge-type blocks can be used in the same grid. This provides different types of grid generation patterns that may be required for special flow regions; for example, the flow around the leading or trailing edges of an airfoil. Wedge-type elements can be employed to design irregular grids. A typical quadrilateral block in Fig. 5 enables specification of $m \times n$ elements only. This restriction propagates to the neighboring blocks along the adjoining edges; i.e., neighboring blocks should also have m or n elements along a particular edge. Such a restriction propagates throughout the entire grid. For example, in the case

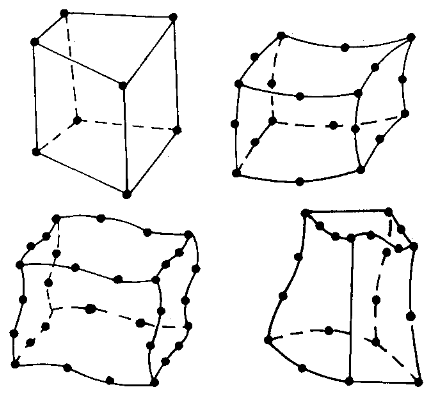
Fig. 2 Different-order isoparametric finite elements.

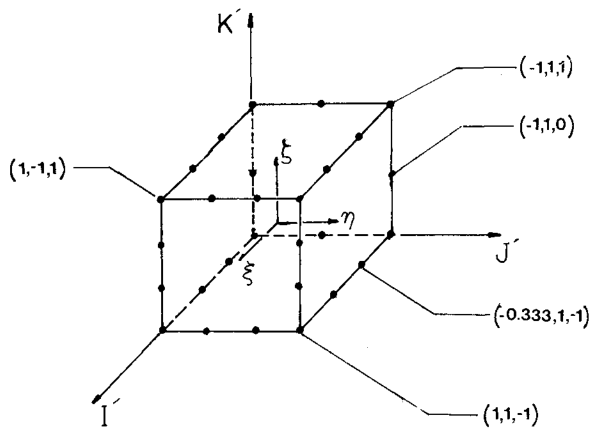

Fig. 1 Parent element for a 32-node isoparametric finite element.

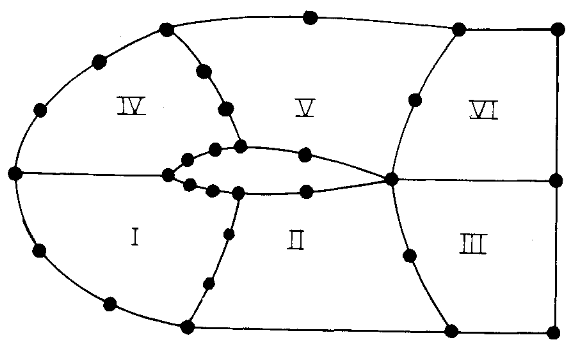

Fig. 3 Block structure for a simple problem.

of an O-type grid, the number of grid points in the circumferential direction has to remain constant. By using the wedge-type elements, one can design grids where the number of elements may change from *n1* to *n2* in a given direction. Similar combinations are possible in all three directions. When *n1* and *n2* are specified, the wedge-type elements are automatically defined by the grid generation scheme for this particular block.

2) Linear, parabolic, and cubic elements can be used at any portion of each block. Once a grid is generated, any element in the block can be specified as a higher order element. Suppose only three elements are declared as cubic elements in a single block after the grid pattern is defined. All the transition elements that connect these three cubic elements to the surrounding linear elements are then automatically generated. The user is not concerned with numbering the nodes, which is done automatically. This is a very important capability since one can design a rather regular mesh and then provide a dense region of grid points by simply increasing the order of the elements in this region. For example, around the leading edge of an airfoil, one would achieve a local grid refinement by using this capability, without placing unnecessary grid points away from the critical area for the sake of generating regular grids.

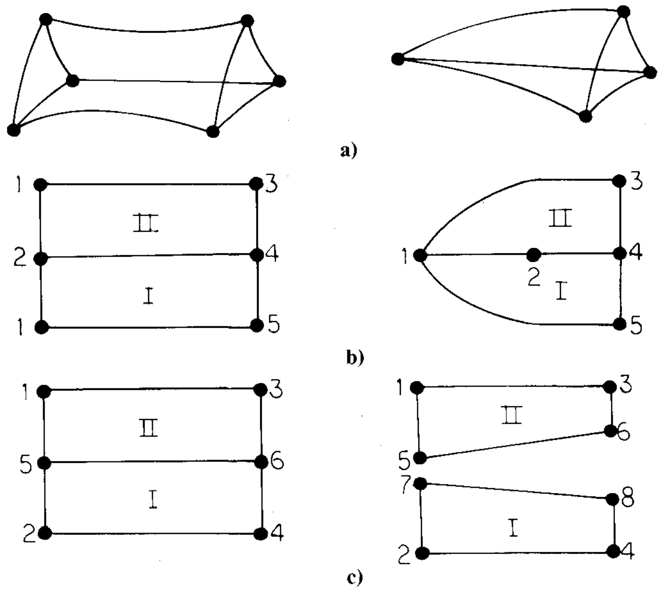

Fig. 4 Generation of irregular grid structures: a) wedge-type isoparametric blocks; b) generation of wedge-type blocks by collapsing regular blocks; c) placement of cavities between neighboring blocks.

In defining the finite element grids, two types of controls are defined for specifying the biasing in the directions of the elements as shown in Fig. 5.

First, the relative size of the elements along each edge is specified by an edge density (D_1-D_4).

Second, the points along each edge are employed as control points for defining the grid density (C_1-C_{11}). The location of these control points along each edge specifies the variation of density along the edge. By shifting the control points along a particular edge, one can modify the grid distribution along that edge.

It is expected that one will design an irregular block structure to provide clustering of grid points around initial areas. The finite element method is general in treating such grid structures. The stretching of individual blocks provides additional flexibility in defining such clustered grids.

Implementation of the Exact Geometry

It should be remembered that up to this point, the grid generation is considered for a block geometry which only approximates the correct geometry. By using this simplified geometry, the user can design blocks, generate grids and, after viewing, interactively make several modifications. As a last step, after the block design is finalized, the generated nodes are repositioned to be located on the exact geometry of the flow boundaries. For each block, the free surfaces are automatically detected. These are the surfaces that are not attached to a neighboring block. For each of these free surfaces, a detailed description of the boundary is provided by the user. This can be in the form of an algebraic equation. If a set of discrete data points is provided, a curve fitting is performed. Then the nodes located on the approximate surfaces are moved in the normal direction of the isoparametric curve to be located on the exact surfaces. Figure 6 describes the procedure for locating the exact surfaces. Points P_T are the generated nodal points on the block surface which do not necessarily lie on the physical surface. The normal vector, N_T, is defined for the boundary surface Y with a functional relationship. Point P_T is then relocated as point P_{TR} in the local coordinate system, which is determined by intersecting the normal vector with the boundary surface. Evaluation of the intersection point involves an iterative process. This scheme can be improved to move several layers of grid points near the boundary surfaces in a parametric fashion.

It is important to note again that for the purpose of providing a simplified system to work with, the details of the exact geometry are kept separate from the grid generation procedure until the end. By using the above scheme, the user works with rather simple geometric descriptions while designing the grid. After the exact geometry is included and the grid is generated by using the exact geometry, one may decide to make modifications to improve certain regions of

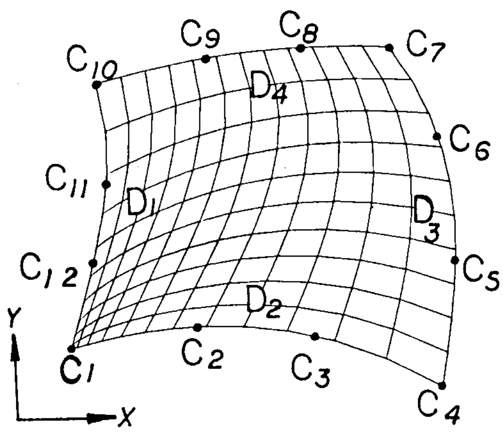

Fig. 5 Generation of grids for individual blocks.

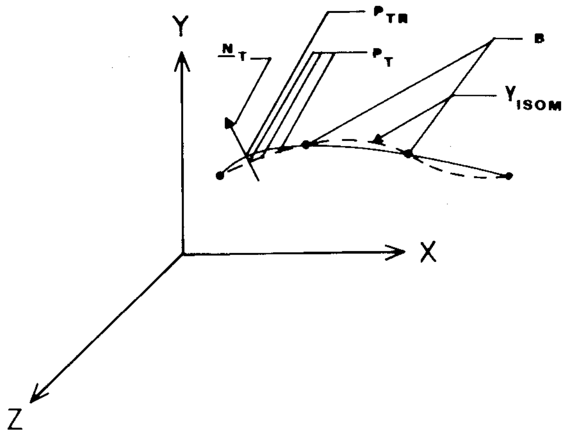

Fig. 6 Relocation of the surface nodes to fit the exact geometry.

the grid. At that point, the user will change only one or two blocks which describe a particular region. One may decide to add new blocks or construct several blocks out of a single block. This is again done locally. During the modification process, the nodes are always renumbered internally and automatically. In fact, by using an element-based solution scheme,[3] the data transfer is always performed on an element base. If one decides to specify higher order elements or to use wedge-type elements, as shown in Fig. 4, to modify the element densities as discussed above, these operations are again performed on a block basis.

As evident from the preceding discussion, one of the important objectives of the above scheme was to provide a step-by-step procedure for grid generation. Since we expect a computational grid to be properly designed only after several iterations, a modification is implemented for a particular region of the grid interactively without changing other portions of the data.

III. Applications of the Grid Generation Scheme

The developed grid generation scheme was first applied for generating a three-dimensional finite element grid around a wing with a cylindrical body. The basic features of the grid generation scheme are illustrated here for this sample problem.[4] The applicability of the scheme for describing a more complex aircraft configuration is also presented in this paper.

Wing-Body Problem

To study the applicability of the developed grid generation scheme, the transonic flow around a wing-body combination was considered. This particular configuration was originally tested by Hinson and Burdges.[5] With an aspect ratio of 3.8, it was originally designed for transonic cruise and tested in the presence of a body in a high Reynolds number wind tunnel. The geometry of the wing is defined at the root and tip sections by discrete data points. The wing coordinates at other spanwise stations are then evaluated by linear interpolation between the root and the tip sections. The details of the geometry can be found in Ref. 5. The fuselage used in the test is a simple shape: an elliptical forebody and afterbody with a constant section in the wing region. In the present study, the fuselage was taken as an infinite cylinder with a constant cross section.

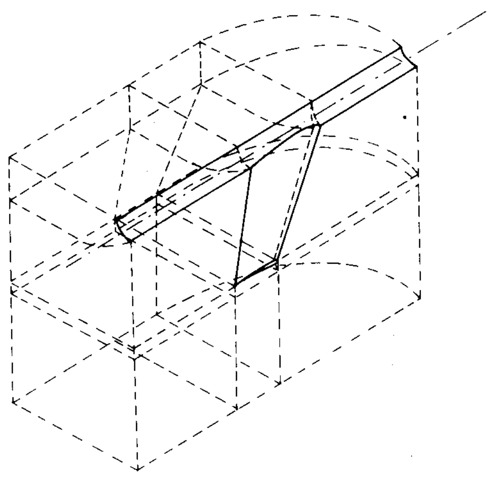
Fig. 7 Block structure for the wing-body problem.

Description of the Computational Grid

The computational space to be analyzed is bounded by an infinite fuselage having a circular cross section of constant radius and the wing attached to this body. The computational space is truncated at finite distances from the wing surface. It is assumed that the flow is symmetric about the vertical plane containing the fuselage centerline, so that symmetry conditions can be applied and the flow has to be analyzed only in the half-space.[6] The far-field boundaries are placed approximately three chord lengths from the wing surface in the streamwise and surface normal directions; in the spanwise direction, the far field is located one span length from the wing tip. The C-type grid chosen wraps around the wing leading edge and becomes a rectangular grid past the wing trailing edge.

Definition of the Blocks

In the numerical solution of a particular fluid mechanics problem, the accuracy of the solution and the computation time depend strongly on the computational grid employed in the analysis. In general, for subregions of flow domain where high gradients of flow parameters are expected, the grid has to be finer than in other regions to maintain the same level of accuracy. It is also desirable to maintain other properties, such as orthogonality of the grid and streamlining in the flow direction, to reduce artificial smearing of the flow parameters.

In the case of transonic flow analysis around this particular wing-body combination, the wing has a sharp leading edge. Around this region, high gradients of velocity are expected to occur. In addition, the physical boundary around the leading edge shows a rapid change in the curvature which requires considerable concentration of grid points in order to represent the exact boundaries without losing the details of the leading edge.

Considering the above physical characteristics of the flow problem, the computational grid was designed based on the

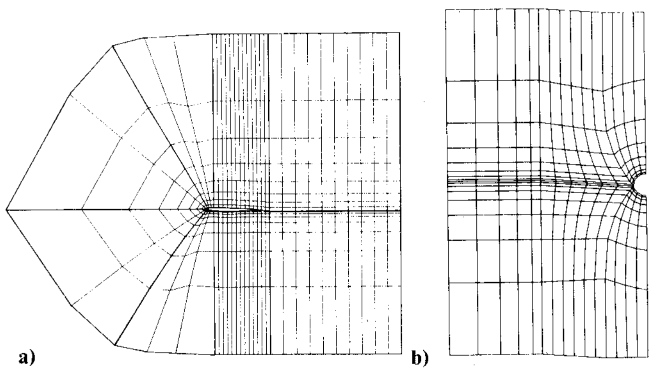

Fig. 8 Grid distribution around the wing-body: a) in the flow direction; b) along the cross section of the wing.

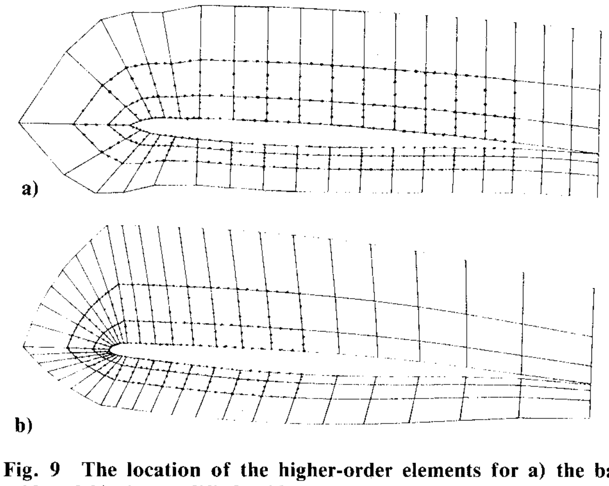

Fig. 9 The location of the higher-order elements for a) the basic grid and b) the modified grid.

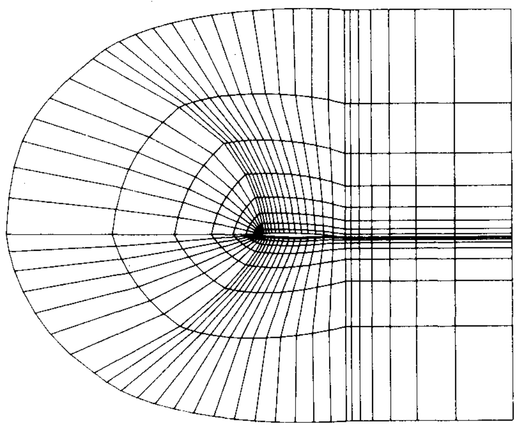
Fig. 10 Grid distribution in the streamwise direction for the modified grid.

block structure shown in Fig. 7. The wing and the trailing vortex sheet are placed between two layers of blocks similar to the simpler case described in Fig. 4c. In the streamwise direction, three layers of blocks are employed where the first row of blocks were coupled to each other in the flow direction. With this combination, a C-type radial grid was obtained, for which high gradients of elements around the leading edge can be introduced. In the spanwise direction, physical constraints required four layers of blocks. In this case, the block layer past the wing performs the transition between the far field and the wing tip in the spanwise direction.

Grid Distribution

The element distribution inside the blocks is determined by the number of elements along three principal directions and the gradients defined along the block edges. Although these are specified separately for each block, the grid generation scheme assumes the continuation of the same number of elements inside the blocks, which are connected to each other along the principal directions. Also, it is required that the stretching parameter definition for the same edge shared by up to four blocks should be the same unless irregular blocks are employed.

The element distribution around the leading edge is designed to be finer along the streamwise direction, while over the wing a smooth change toward a coarser grid is used for efficiency. Figure 8 shows the grid distribution along the streamwise and spanwise directions at typical sections. This basic grid has 28 elements in the streamwise direction, 18 of them located on the wing. There are 15 elements in the spanwise direction, 4 of them on the body and 6 of them on the wing. In the third direction, normal to the wing upper and lower surfaces, 16 elements are utilized, with the wing located at the middle. As can be seen in Fig. 8a, in each block, element gradients along the streamwise direction are assumed to be uniform in this first attempt; i.e., each element inside the block has the same length in the streamwise direction. In the direction normal to the wing surface, gradients are specified in such a manner that the ratio of the element size on the surface to the one at the far field is 1:25.

As mentioned earlier, in order to resolve rapid changes in the flow variable, higher-order elements were employed in the critical flow regions. For this grid, two cubic element layers were placed over each of the wing upper and lower surfaces as shown in Fig. 9a. Each layer had 15 cubic elements along the streamwise direction and 6 in the spanwise direction. The total number of cubic elements in the grid was 360. The total number of elements added up to 6720, while there were 10,105 grid points. A similar study presented in Ref. 6 required 21,119 grid points to obtain the same level of grid refinement around the leading edge when

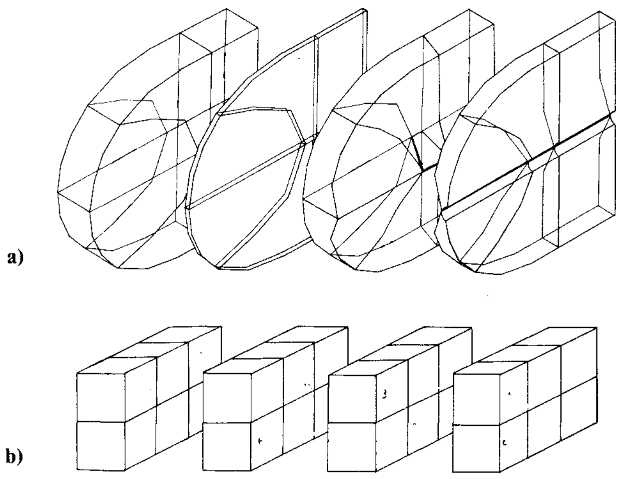
Fig. 11 Block structure of the modified grid: a) real geometry; b) transformed geometry.

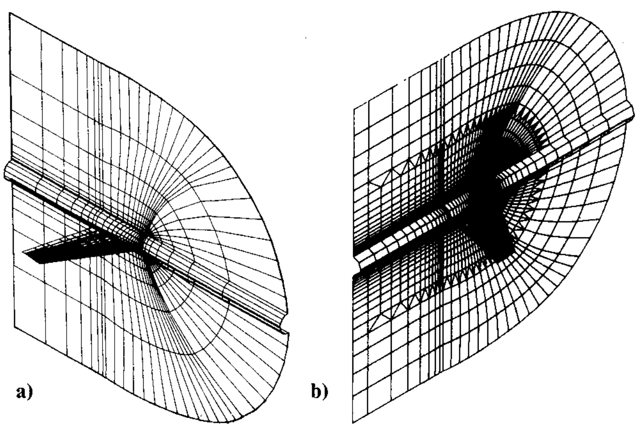
Fig. 12 Grid distribution over the wing-body and the side wall for a) the modified grid; b) the modified grid with wedge-type elements.

only linear elements were used. The use of cubic elements is, in itself, very important for efficient grid design.

Grid Studies

Case 1. Basic Grid

The computational grid designed in the previous section was employed to analyze the transonic flow around the particular wing-body combination as an initial test case. The finite element procedure developed previously for the analysis of transonic potential flows was employed.[6] In this paper, details of the analysis procedure are omitted since the main objective is to present the developed grid generation scheme. The results of the analysis are reported in detail in Refs. 4, 6, and 7.

When the results computed by using the basic grid were compared with the experimental data, discrepancies were observed around the leading and trailing edges of the wing profile. In order to improve the accuracy of the solution in these regions, it became necessary to reduce element sizes and design a finer grid in these regions. In addition to these critical areas, it was observed that the accuracy of the solution was satisfactory over the remaining flow domain. This suggested a local refinement of the grid rather than an increase in the number of grid points everywhere.

Case 2. Modified Grid

Based on the above considerations, a second grid was designed to show the effect of the distribution of grid point and element densities. The total number of elements and grid points was kept the same. For this new grid distribution, several changes were made along the streamwise direction as shown in Fig. 10. The edges of the blocks surrounding the leading edge of the wing were shortened, and the number of elements was increased in this region as the orthogonality of the leading-edge elements was preserved. The new block structure after the above modification is shown in Fig. 11.

As is evident from Fig. 10, the upstream boundary can be specified more accurately with the present grid. The overall distribution of the grid points is more suitable to represent the flowfield. The curved radial edges of the blocks ensure the orthogonality of the element edges generated around the leading edge. All these changes are made locally, while most of the original data for the first grid were not modified. Inside the blocks, the element gradients were modified to provide a finer element distribution toward the leading edge. Again, by using different gradients, smaller elements were placed around the trailing edge. Although the number of higher-order elements was kept the same, their locations and sizes were modified as shown in Fig. 9b. Figure 12a shows the grid distribution over the wing-body and side boundaries. It is important to note that all of the above changes were made individually for each block by modifying the data for one block at a time.

The comparison of the two different grids with the same number of elements, illustrates some of the important aspects of the design of computational grids. It is necessary to design grids that provide good alignment with the changing flow variables. Calculation of element densities specified at the regions of high flow gradients, for example, may become responsible for the accurate solution of the whole domain in transonic flows. For this particular problem, relatively fine grids are required around the leading and trailing edges of the wing profile. The above example illustrates how this can be achieved by local refinement of the grid. Each of the critical regions can be refined locally by the user, while they are automatically connected to the surrounding blocks.

The above example shows the necessity of providing sufficient grid refinement of all flow regions around the wing-body configuration. On the other hand, the inefficiency of an overall grid refinement by the use of equally spaced grid points is also apparent. In this particular example, the ratio of spacing between the nodes around the leading edge to the nodes in larger elements at the far field is about 1:80.

For solving a practical three-dimensional problem, one may start with a basic grid and even obtain some preliminary computational results. This grid can then be refined for better accuracy. The comparison of results from the basic and modified grids will indicate the accuracy of the obtained solution, even without the aid of experimental results. The developed finite element scheme allows the user to concentrate on a particular block of the basic grid and modify it with minimum effort without changing the remaining portions of the computational grid.

Use of Wedge-Type Elements

As discussed previously, the use of wedge-type elements and blocks can considerably reduce the number of elements used in the analysis. For the test case, wedge-type elements were introduced as shown in Fig. 12b. Use of this feature reduced the number of elements from 6720 to 5265, with no appreciable changes in the flow regions where more accuracy is needed. The wedge-type elements were employed in this procedure. The grid pattern was modified only in the radial direction. The same procedure can be applied in other directions to obtain a further reduction in the number of grid points. To further illustrate the importance of such an application, the modified grid was refined three times in each direction, resulting in a grid of 181,660 elements and 191,590 grid points. With a similar application of wedge-type elements, the grid was reduced to 122,580 elements and 127,836 grid points.

Generation of a Finite Element Grid for an Aircraft Configuration

The case of the isolated wing with a cylindrical body illustrates the basic capabilities of the developed grid generation scheme. To demonstrate the applicability of the scheme to more complex geometries, modeling of a complete aircraft geometry was considered. An F-16 aircraft configuration was chosen for this purpose. Although the exact geometry was not employed it provides necessary details for demonstrating the flexibility of the developed scheme.

The first step in the grid generation scheme is the description of the aircraft geometry itself. Although this task is trivial for simple shapes like the wing-body example, it becomes time-consuming in the case of a complete aircraft. For this part of the analysis, a general-purpose geometric modeling program (GEOMOD) was employed.[8] This program was originally developed to perform different types of geometric modeling tasks encountered in CAD/CAM activities. It is an interactive program, with which the user can specify the geometry by digitizing points on several sections of an aircraft body as shown in Fig. 13. To digitize the full aircraft, one can digitize each of the components, such as the body, engine, and wings, as separate objects with their own reference coordinate systems. The geometric modeling package employs a series of B-splines to connect the digitized points by a continuous curve in space, again as shown in Fig. 13a. These sections are then combined to produce a three-dimensional definition of the aircraft volume as shown in Fig. 13c. As shown, the description of the body in the computer takes two forms: the surfaces are represented both by a series of splines and by straight facets. The facet representation is generally employed for illustrations, while the spline representation is used for computations. One can refine a facet representation of a body to desired accuracy based on the information stored in the form of splines. The combination of several objects involves a series of Boolean operations and can be executed in the geometric modeling package in a general fashion, with each of the components of the aircraft described separately and joined together as shown in Fig. 13. The intersection of the body and the wing, for example, is automatically calculated during a Boolean operation.

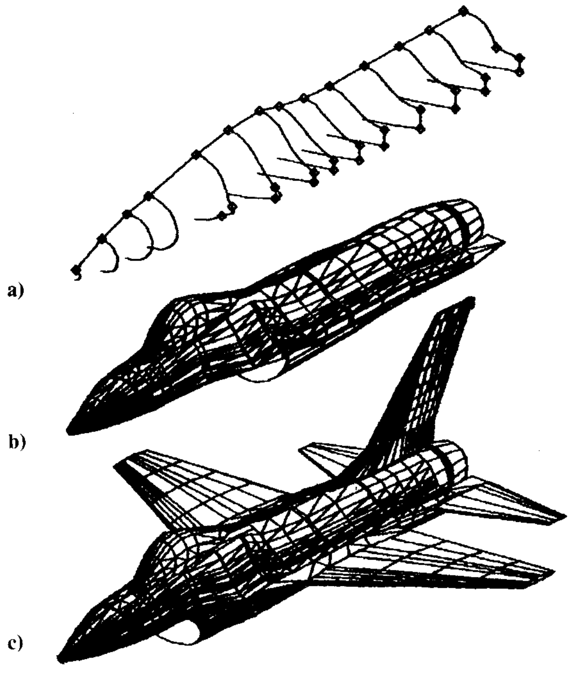

Fig. 13 Construction of the geometric description of the aircraft in three steps.

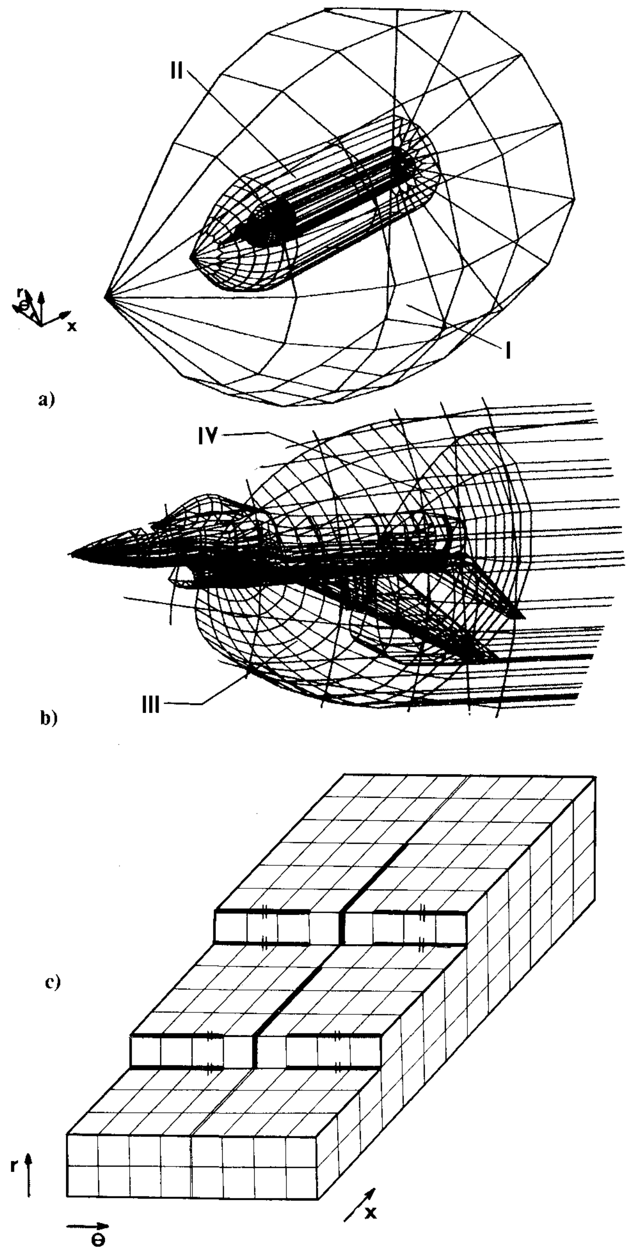

Fig. 14 Geometric description of the block structure around the aircraft: a) definiton of subvolume; b) details of subvolumes around wings; c) overall block structure.

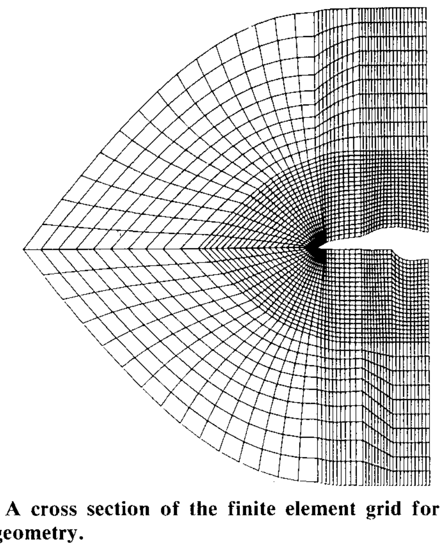

Fig. 15 A cross section of the finite element grid for part of the aircraft geometry.

Once the geometry of the aircraft is defined, the description of the blocks surrounding the aircraft is required. This again becomes a complicated geometry problem for an aircraft configuration. Also, as discussed above, the design of the grid is basically controlled by the shapes and sizes of the blocks. One has to be able to define these blocks in three dimensions without excessive effort. In the case of the sample problem, the block structure around the aircraft body was defined in the following form.

First, four egg-shaped regions (subvolumes) were defined around the aircraft as shown in Fig. 14a. The two outer subvolumes were designed for generating proper grids around the nose section of the aircraft. In this region, one would like to provide a radial grid where the grid lines remain normal to the aircraft surface. Further away from the body, these directions were modified. In the present example, subvolumes I and II are employed for this purpose. Subvolumes III and IV are designed to model the flow around the two pairs of the wings along the aircraft as shown in Fig. 14b. In each case, the objective was to obtain a C-type grid around the wing. Once these subvolumes are defined using the geometric modeling package, they are further divided to define the blocks. Finally, the aircraft is subtracted from the subvolumes to leave only the air space behind. It is obvious from the above procedure, that the design of the grids will still involve considerable engineering judgment.

The block structure for the developed aircraft grid is shown in Fig. 14c. The blocks are defined in a cylindrical coordinate system: radial, circumferential, and longitudinal. As can be seen from this figure, the inner two subvolumes produce a smaller number of blocks than the outer subvolumes. In this case, 336 blocks were generated for the sample problem. The above block structure is then defined as an input to the present grid-generation program. A sample of the grid generated around the nose is shown in Fig. 15. Once the grid is generated, the user can inspect each of the blocks individually and make modifications. The modifications can occur at the grid generation level or, through changes in the geometry of the blocks, before the grid generation is performed.

IV. Conclusions

The main objective of the present development is to provide a flexible and efficient grid-generation procedure which can be used in modeling complex aircraft configurations. The developed procedure is capable of handling arbitrary geometries and is not restricted to modeling single shapes. Some of the important considerations in attempting to develop such a grid generation system are listed at the beginning of this paper. In describing the method, we have tried to show how each of these problems can be treated. The use of general-purpose geometric modeling packages developed for CAD/CAM applications is also demonstrated in this paper.

It is important to note that the procedure is called a finite element grid generation scheme since it is aimed at utilizing the flexibilities of the finite element method as a solution procedure. The use of wedge-type or higher-order elements in designing irregular grids is an important advantage of the finite-element method. Any grid generated by the present scheme provides the input data for a finite-element program with no specific changes required for the analysis. The real advantages of the finite element method become more apparent in a design environment in which geometries become more irregular and several changes from a standard configuration have to be made during the analysis.

Acknowledgments

This research was supported by AFOSR-Grant No. 80-0258. The authors express their appreciation to the project monitor Dr. James P. Wilson for his valuable suggestions during the course of this investigation. Computer services provided by IUPUI Computer Services are also acknowledged. Finally, the authors would like to thank A. L. Klosterman and G. Nay of SDRC for the preparation of the geometric model of the aircraft geometry.

References

[1]Thacker, W. C., "A Brief Review of Techniques for Generating Irregular Computational Grids," *International Journal for Numerical Methods in Engineering*, Vol. 15, Sept. 1980, pp. 1335-1341.

[2]Zienkiewicz, O. C., *The Finite Element Method*, 3rd Ed., McGraw-Hill Book Company, Ltd., London, 1977.

[3]Irons, B. M., "A Frontal Solver Program for Finite Element Analysis," *International Journal for Numerical Methods in Engineering*, Vol. 2, No. 1, 1970. pp. 5-32.

[4]Ecer, A., "Analysis of Three-Dimensional Transonic Potential Flows Using an Optimum Grid," Final Report, AFOSR-80-0286, Dec. 1982.

[5]Hinson, B. L. and Burdges, K. P., "Acquisition and Application of Transonic Wing and Far-Field Test Data for Three-Dimensional Computational Method Evaluation," Final Report, Lockheed-Georgia Company, AFOSR-TR-80-0421, March 1980.

[6]Ecer, A., Citipitioglu, E., and Bhutta, B. A., "Design of Finite Element Grids for the Computation of the Three-Dimensional Transonic Flow Around a Wing," AIAA Paper 82-1019, June 1982.

[7]Ecer, A., Spyropoulos, J., and Tuncer, I. H., "A Block Structured Finite Element Grid Generation Scheme for the Analysis of Three-Dimensional Transonic Flows," AIAA Paper 84-0004, Jan. 1984.

[8]Klosterman, A. L., Ard, R. H., and Klahs, J. W., "A Geometric Modeling Program for the System Designer," Conference on CAD/CAM Technology in Mechanical Engineering, MIT, Cambridge, Mass., March 1982.