Grid Generation and Flow Calculations for Aircraft Geometries

N. P. Weatherill* and C. R. Forsey*
Aircraft Research Association Limited, Bedford, UK

A method for calculating the flowfield around complex aircraft configurations based on a multiblock grid-generation approach coupled with an Euler flow algorithm is presented. In this approach the flowfield is subdivided into a set of nonoverlapping blocks. Grids are generated simultaneously in all of the blocks using an elliptic grid-generation method. Appropriate boundary conditions on the block faces ensure that grid lines pass smoothly between adjacent blocks leaving a grid that is globally smooth except for a few isolated geometric singularities. A geometry package for use with the grid generator based on bicubic surface patches and with component intersection capability is briefly described and the generation of grids on the configuration surfaces is treated in some detail. An explicit finite volume Euler algorithm has been developed for use with the multiblock grids and is described herein. Some preliminary results obtained using the multiblock system are presented.

Introduction

THE rapid advance in computer technology coupled with improved flow algorithms has brought the possibility of flow calculations for realistic aircraft configurations much nearer to practicality. However, to date, most flow simulations have been restricted to relatively simple configurations, such as wing/body combinations,[1] because of the difficulty of generating grids around more complex configurations. Most of the production codes now widely in use in industry are based on ad hoc grid-generation procedures, such as conformal mapping coupled with stacked two-dimensional grids.[1] While such procedures have proven valuable in producing high-quality grids for a restricted range of problems, their inherent limitations suggest an alternative, more flexible approach is necessary for general aircraft configurations. Also, while most current production codes employ the full potential formulation, new algorithms for the efficient solution of the Euler equations have now been developed.[2] Since the Euler equations represent an improved flow model compared with the full potential equation, it seems appropriate that this model should be incorporated in any method developed for more general configurations. In this paper, an approach to calculating flows around complex configurations consisting of a flexible grid generator for producing body-conforming grids coupled with an up-to-date Euler algorithm for the flow solver is described.

It is now generally recognized that the fundamental problem associated with grid generation for general configurations is that each component in the configuration (wing, body, etc.) has its own natural type of grid topology (such as an O or C grid for a wing, O grid for a body, etc.), however, these topologies are usually incompatible with each other. This has led to the idea of generating grids whose topology is locally consistent with each component but with some global means of connecting the grids. Such approaches are usually described as multiblock or zonal approaches because the complete flow region is broken down into blocks or zones local to each component.

The mechanism by which the regions are connected leads to a variety of approaches. Atta and Vadyak,[3] using the full potential equation, and Benek et al.,[4] using the Euler equations, generate the local component grids completely independently and use grid overlapping together with interpolation of the flow variables to transfer information between the grids. Although this appears to work very well for the full potential equation, Benek et al.[4] report some problems with conservation in the interpolation schemes used for their Euler solutions. Alternatively, Lee[5] divides the flow region into a set of blocks and generates the grids essentially independently in each block subject to the constraint that the grid lines meet point for point at the block boundaries. This approach leads to grids that may be nonsmooth at these boundaries so that extra requirements are imposed on the full potential flow solver used. Furthermore, the particular way chosen to divide the flow region into blocks, although simple and consistent for the user, produces irregular points, referred to as fictitious corners, at some grid points on the surfaces of the configuration again imposing further requirements on the flow solver. Miki and Takagi,[6] using the block concept, describe a scheme whereby grid lines remain smooth and continuous across block boundaries and apply these techniques to internal flow geometries.

The approach adopted herein is similar in concept to the preceding methods in that the flow region is divided into a series of blocks chosen to match the general topology of the configuration and to give the particular grid structure felt to be most appropriate for each component. However, the grids are not generated independently in each block, instead, they are generated simultaneously in all blocks using an elliptic grid generator. The mechanics of this multiblock approach together with the geometry package, surface and field grid generators are described and some preliminary results obtained from the Euler code are presented. A general review of elliptic grid-generation methods, which may help to clarify some of the details of the particular approach used herein, is given by Thompson and Warsi.[7]

Multiblock Approach—General Principles

Central to the multiblock approach adopted herein is the idea that the whole flowfield between the surfaces of the con-

Presented as Paper 84-1665 at the AIAA 17th Fluid Dynamics, Plasma Dynamics, and Lasers Conference, Snowmass, CO, June 25-27, 1984; received Aug. 1, 1984; revision submitted April 15, 1985. Copyright © American Institute of Aeronautics and Astronautics, Inc. 1984. All rights reserved.

*Senior Project Supervisor, Aerodynamics Department.

figuration and some outer far-field boundary is conceptually broken down into a set of blocks. The union of these blocks fills the entire flowfield without holes or overlaps. Each block is chosen to be topologically equivalent to a cuboid in that it has six faces and eight corners and, therefore, in principle, can be mapped into a unit cube in computational space without change in topological structure. The general process is illustrated in Fig. 1. Cartesian grids in the unit cubes in computational space map back to curvilinear grids in physical space.

The actual procedure used to perform the mapping is the elliptic grid-generation approach of Thompson et al.[8] In this approach the coordinates (x,y,z) defining the grid nodes in physical space are obtained as the solution of nonlinear elliptic partial differential equations (derived from Laplace's equation) in which the independent variables are the computational coordinates in the unit cubes. Boundary conditions must be imposed on each face of each block both for this grid-generation procedure and in the subsequent flow calculations. A single boundary-condition type on each face both in the grid-generation and flow calculations is chosen herein, although the actual boundary condition used may be different in both steps. This can lead to a rather large number of blocks, but greatly simplifies the internal program logic. The actual boundary conditions used on each face depend upon the particular grid topology.

In the grid generation some block faces represent parts of the configuration surface or parts of the outer boundary. On such faces, referred to as Dirichlet faces, the grid distribution is calculated by the surface grid-generation procedure described subsequently and, hence, can be treated as fixed data for the field grid generator. Other faces represent the junction of two blocks and, as such, are purely notional boundaries which, provided the grid lines pass smoothly through the junction, have no physical significance. On such boundaries, referred to as continuity boundaries, the grid-generation equations are solved at each point as though the points were inside the block since program logic is used to access the coordinates of points in neighboring blocks required in the solution procedure. Hence, the grid is as smooth at these boundary points as at any point inside a block. Boundary conditions similar to those given above can be defined for the flow calculations.

In order to keep the system flexible, the information needed to define the block structure and the interrelationship between blocks is supplied to the system by the user in a "topology file." This consists of a list of all faces of all blocks giving for each face the type of boundary conditions for the grid-generation and flow calculations and, where appropriate, the adjacent block number, face number, and orientation of the adjacent face relative to the current face. Eight possible orientations of one face relative to another can be defined, but only four of these are topologically valid for any particular pair of faces. To help clarify these ideas, a schematic diagram for one of the grids generated is presented subsequently.

At present, this file is set up manually by the user although having once been set up for a particular configuration (e.g., wing/canard) it can be used for all future configurations of this type and also as one part of a more complex configuration. Eventually, an automated procedure such as that described by Roberts[9] may be adopted.

Geometry Handling

Although many geometry packages are now commercially available, they are generally tailored to CAD/CAM applications rather than the special requirements of grid generation. Hence, a surface geometry package capable of simple input format and component intersections, based on bicubic surface patches, has been developed as part of the total system.

For the user the main requirement in geometry definition is simplicity. Hence, each component is defined separately by an arbitrary number of cross sections with an arbitrary number of points on each section. However, a consistent definition of all surfaces is required so that points on the surfaces other than those input can be easily derived and component intersections can be incorporated into the surface definition.

Following the ideas of Coons,[10] each component is represented by a network of parametric bicubic patches. Any patch can be described by the matrix equations

$$AMB^{-1} = X$$

where $X=(x,y,z)$, $A=(s^3,s^2,s,1)$, $B=(t^3,t^2,t,1)$, and M is a matrix containing the parametric derivatives of X and some blending functions. In practice, the parameters s and t are normalized surface distances along and across the input cross sections. Parametric cubic spline curve fits are used to interpolate the input data to the nodes of the patch network and to calculate the parametric derivatives at these nodes for use in matrix M.

Once all of the surfaces have been patched in this fashion, intersections between any two surfaces can be calculated. At any point on the line of intersection of two patches the equations for both patches yield the same value of X, hence,

$$AMB^{-1} - CND^{-1} = 0$$

where $A=(s^3,s^2,s,1)$, $B=(t^3,t^2,t,1)$, $C=(u^3,u^2,u,1)$, $D=(v^3,v^2,v,1)$, and s,t and u,v are the parametric coordinates in the two patches. Arbitrarily fixing one of these parametric coordinates gives three nonlinear algebraic equations to be solved for the other three parametric coordinates. These three equations can be solved by any convenient method. A Newton-Raphson library routine[11] is used herein, which is found to give rapid convergence in most cases, especially since each intersection point acts as a good initial approximation for the next point.

Once the parametric coordinates of a line of intersection have been calculated, the intersecting surfaces are repatched to eliminate the parts of each component inside the other. The far-field boundary is patched and treated in the same fashion except that this surface is initially defined algebraically rather than by input data. The whole procedure leaves a consistent mathematical model of the complete configuration and its outer boundary for use in the surface grid-generation step.

Surface Grids

Surface grid generation is, in itself, one of the most difficult and yet important aspects of the total grid-generation problem. The surface grid influences the field grid close to the configuration, the very region where flow gradients are important and need to be resolved accurately. Surface grids have the same requirement for smoothness and continuity as the field grids for which they act as boundary conditions but, in addition, they are required to conform to the configuration surfaces, including lines of component intersection, and to model regions of high surface curvature. A number of approaches have been suggested in the literature and the one adopted herein is based on surface parametric coordinates.

The parametric approach utilizes the fact that a surface can be expressed in terms of two parametric coordinates, $S=(s,t)$. Surface grid generation can then be viewed as a transformation, $S \rightarrow X$, where the grid is generated in two-dimensional parametric space and mapped to physical space by an appropriate transformation. The mapping function adopted herein between S and X is the Coons patch for consistency with the geometry definition.

Our procedure for surface grid generation involves extensive data handling and forms an integral part of the geometry package. From the discrete input data which defines the geometry a new distribution of sectional data is derived to ensure that the patched data that describes the continuous surface is smooth. A field topology structure appropriate to the configuration is defined in the manner described earlier. This topology is then scanned for edges of blocks that form the intersections of two components and information extracted on the number of points required on the intersection line. These coordinates are then obtained from the Newton-Raphson intersection routine and used as Dirichlet conditions in the surface grid generation equations. The topological structure of the surface grids is automatically derived from the field topology, thus ensuring that surface and field grid structures are consistent.

The elliptic equations of Thompson et al.,[8]

$$g^{ij}S_{\xi^i\xi^j} = -p^iS_{\xi^i} \tag{1}$$

where g^{ij} are the metric terms, p^i the control functions, and ξ^i the computational coordinates with the tensor notation of i, j taking values 1 and 2 are solved for the parametric coordinates using a successive over-relaxation point-iterative scheme. The solution procedure involves the update of S in a block to iteration level $n+1$ from level n. No special treatment of the equations is required on continuity block boundaries since the points from neighboring blocks contained within the computational molecule are accessed using a pointer system. In this way the positions of the block boundaries in physical space evolve with the iterative procedure. The grid topology is analyzed for singular points at which the s,t values are obtained as an appropriate average of neighboring points. The solution values are mapped via the Coons patches to physical space to yield the surface grid.

Since, in general, equal intervals in parametric space do not correspond to equal intervals in physical space, a necessary condition for a smooth surface grid is that the transformation between X and S is smooth, i.e., the patched transformation must be derived from smooth sectional data. If the patched data are derived from highly stretched sectional data, then distorted grids will be produced on the surface. Ideally, the patched data should be computed from sections at equal intervals in surface distance, however, in regions of rapid changes in geometry the distance between sections must be small implying an unmanageably large number of patches over the entire component.

It follows from the basic nature of the parametric approach that all surfaces map to a rectangle in parametric coordinates. As a consequence, the generation of a continuous grid on a surface must include the capability of admitting discontinuities in parametric coordinates within the solution procedure. For example, a wing patched chordwise from the trailing edge around the section results in a discontinuity in parametric space at the trailing edge. Information necessary for the program logic to incorporate a jump within the solution domain is contained in the topology file.

The surface grid over an aircraft with a variety of components is shown in Fig. 2. Although simple Cartesian-type grids have been used, the figure illustrates the ability to build up grids on complex configurations using the principles outlined. Another example of a surface grid is shown in Fig. 3 for a wing/body/canard/fin configuration. In this case, the surface grid structure has been derived from an 180 block field topology which consists of a C grid with a Cartesian structure spanwise along the wing and canard. The grid on the body is formed from 42 rectangles in computational space, while the wing and canard grids are derived from 4 and 2 rectangles, respectively.

The flow algorithm for the Euler equations is a finite volume formulation and, hence, the flow domain is truncated a finite distance from the configuration. The generation of a grid on the outer boundary surface is achieved using the same method as for a component surface. The topological structure of such a grid is obtained from the three-dimensional topology for the field and the grid, generated in parametric coordinates, is then mapped onto the surface of a semiellipsoid. Once the outer boundary and the components that intersect the symmetry plane have been discretized, the necessary Dirichlet data are available to compute the grid on the plane of symmetry. An example of a wing with plane of symmetry and outer boundary is given in Fig. 4.

Field Grids

At this stage the finite flow domain has been enclosed by a net of grid points. These fixed nodes are used as Dirichlet conditions for the three-dimensional grid-generation elliptic equations that are equivalent to Eq. (1), where S is replaced by X and the tensor summation is for i, j over the values 1,2, and 3. In a manner similar to the surface grid technique, the equations are solved iteratively with a point over-relaxation scheme using block-by-block update. Singular lines and points in the field are fixed by averaging neighboring points. At continuity block boundaries it is necessary to access points in the adjacent blocks. This is achieved by increasing the dimensions of

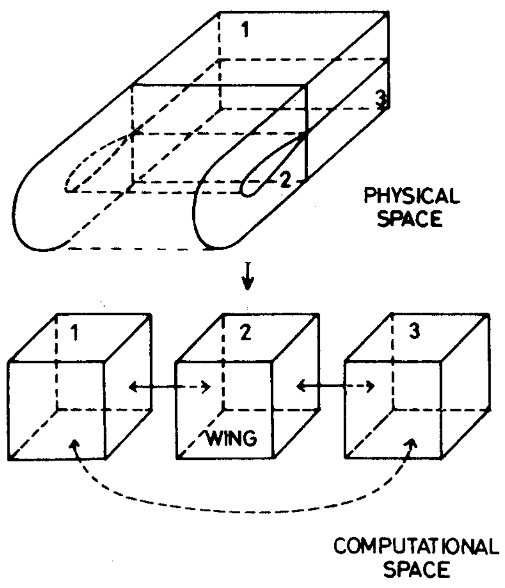

Fig. 1 Multiblock representation of the field around a wing.

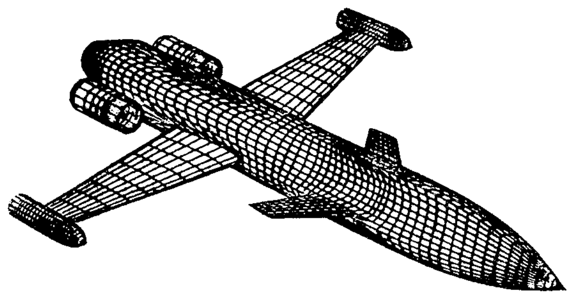

Fig. 2 Surface grid over an aircraft configuration.

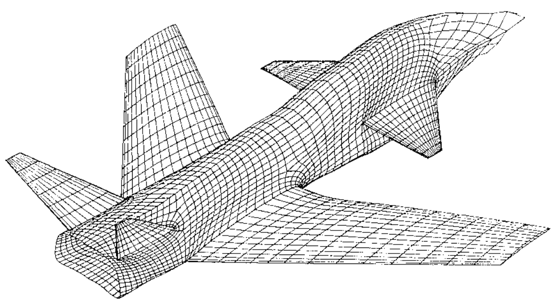

Fig. 3 Surface grid over a wing/body/canard/fin configuration.

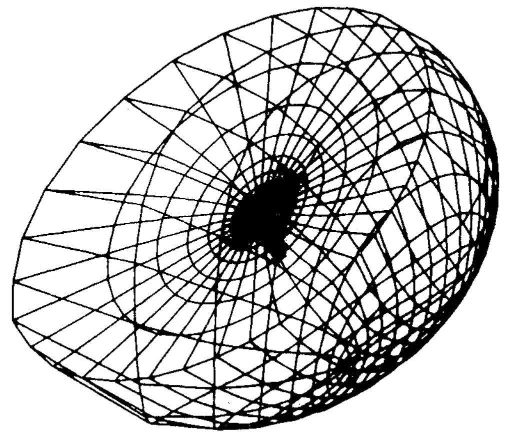
Fig. 4 Grid on a wing, plane of symmetry, and outer boundary.

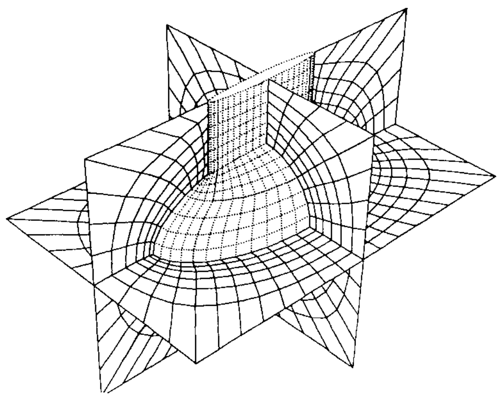
Fig. 5 Field grid around a pylon/tank.

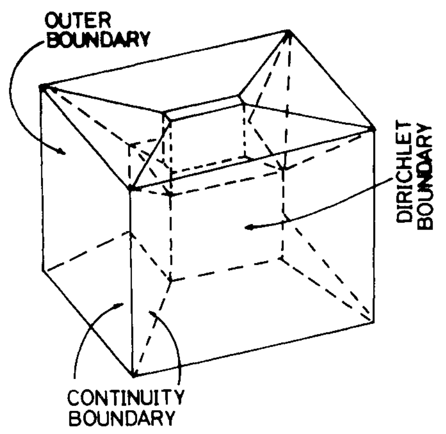

Fig. 6 Schematic of the field grid around a pylon/tank.

the block and the relevant neighboring point coordinates copied into the extra rows created. The solution procedure for points within the block can then be performed without logic checks for block boundaries and, hence, this so-called "halo" approach can use the vector capabilities of modern computers effectively. In this case the halo approach was adopted to provide a comparison with the pointer schemes used in the surface grid and Euler codes. Although computationally efficient, the halo approach, in general, requires more memory than an equivalent pointer scheme.

An example of a field grid around a pylon/tank configuration using a 9-block topology structure is shown in Fig. 5. An O grid structure is evident on each of the three sections through the field. A schematic of the block structure corresponding to this grid is detailed in Fig. 6. Figure 7 shows a 30-block wing-body-tailplane grid with an O-O-O structure that is computationally a very efficient topology, since, for a given number of points, it resolves the details of the wing/wing tip better than any other topological grid structure.

Grid Control

With the present restrictions and cost of computer memory and CPU time, the generation of an optimum grid is of the utmost importance. Research into grid-generation techniques is now widely undertaken, but there exists little information on the quantitative properties necessary for a good grid. Invariably, a grid is judged from visual considerations highlighting the requirement for sophisticated data handling and plotting software.

The ability to control the attraction and repulsion of points to and from particular areas within the flow domain is achieved by the functions p^i of Eq. (1) and its three-dimensional equivalents. Many grid control techniques have been described in the literature, but one that has been implemented into the multiblock logic of the surface grid generator is that due to Thomas and Middlecoff.[12] This approach utilizes the spacing of the fixed boundary data to determine the control functions. These functions are then interpolated throughout the field and act to reflect the spacing on the boundaries throughout the grid. Interpolation from boundary to boundary in a multiblock structure is not always easy and complicated internal program logic is required to ensure that the control functions are smooth and retain the correct sign from block to block.

Figure 8 shows a grid without control where the spacing of the grid lines away from the wing sections is greater than that ideally required for an accurate flow calculation. However, as is shown in Fig. 9, grid control can be used to decrease this spacing.

Flow Calculations

Although methods for solving the Euler equations have been under development for many years, it is only recently that methods which combine satisfactory convergence rates with acceptable accuracy have become available. We use a finite volume explicit Runge-Kutta time-stepping scheme based on the work of Jameson et al.[2] that has been coded to operate within a multiblock framework. A pointer system similar to that described for the surface grid-generation equations has been implemented to enable flux balancing across continuity block boundaries. All variables are stored in one-dimensional arrays and because of the chosen solution procedure of updating variables at all points in a block before advancing to an adjacent block, the code, which is memory resident on a 1 megaword CRAY 1S, can accommodate grids with a maximum of 60,000 points. No special formulation of the Euler algorithm is required at grid singularity points and limited investigations have indicated that the code is insensitive to such points.

As a simple test case for the multiblock code, an Euler calculation was performed using a 4-block C grid for the flow over an unswept panel wing with NACA 0012 sections at 2 deg of incidence and a Mach number of 0.63. The results obtained are compared with a two-dimensional full potential result[13] in Fig. 10. Symmetry conditions applied at the ends of the wing simulate two-dimensional flow in the multiblock calculation. This subcritical result obtained on a grid without grid control and with only 60 points around the airfoil shows reasonable agreement with the full potential result.

Figure 8 shows a section of a 54-block grid about a wing/canard configuration with two NACA 0012 sections, the canard span being half that of the wing. C grids wrapped around each wing are embedded within another C structure, while outboard of each component the grid is collapsed to a

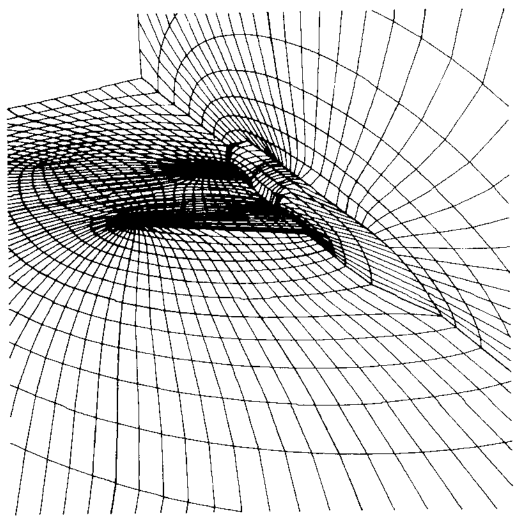

Fig. 7 O-O-O field grid around a wing/body/tail configuration.

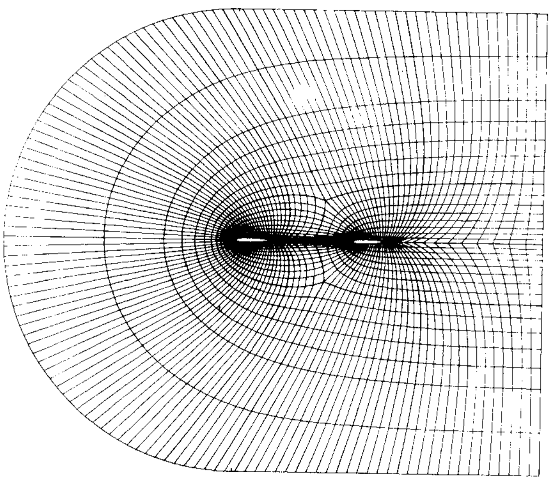

Fig. 9 Section of a grid with control around a wing/canard.

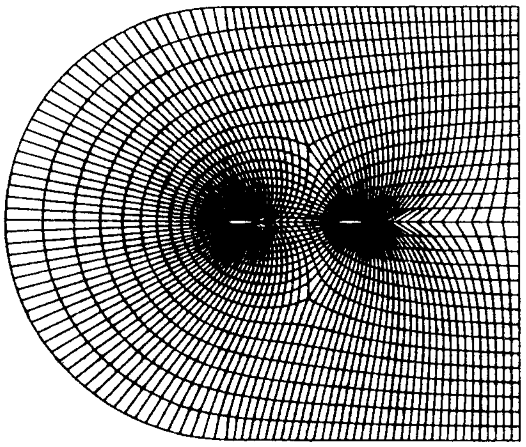

Fig. 8 Section of a grid without control around a wing/canard.

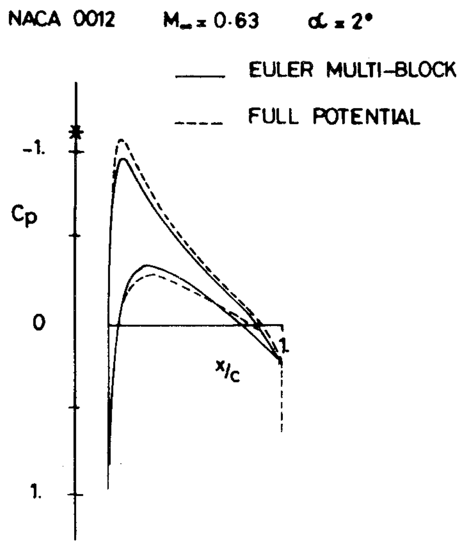

Fig. 10 Comparison between Euler multiblock and full potential approaches.

slit. Automatic grid control was not implemented in this case although to decrease the cell sizes at the trailing edges the point distributions along the wake lines of each component were fixed and the outer boundary specified about eight chords from the configuration. A total of 25,000 nodes were used with 60 points around each wing and 4 and 8 spanwise sections along the canard and wing, respectively. This grid is primarily intended to test various features of the multiblock system rather then as the best grid for such a configuration. The results for the case of Mach number 0.8, 1.25 deg incidence, and 30 deg sweep are shown in Fig. 11. In general, convergence of the Euler algorithm is impaired by poor grid quality so the reduction in the average change of density per time step of three orders of magnitude in 600 time steps while producing pressure coefficients which do not exhibit spurious peaks and oscillations is encouraging. As a consequence of fixing points in the field, discontinuities occur in the grid slope across the surfaces which emanate from the trailing edge of each wing. Since the wing sections are symmetric, no flux crosses these surfaces if the configuration is at zero incidence. However, the results shown in Fig. 11 are for an asymmetric flow where fluxes across these surfaces are nonzero. If the flow algorithm is insensitive to slight discontinuities in the slope of the grid lines, as is the implication from Fig. 11, the possibility arises that the fixing of block boundaries within the field may be an effective technique for providing grid control.

An important aspect of the Euler multiblock structure is the block-by-block update of flow variables. In a single block grid, all variables in the field are updated to an intermediate time level before advancing to the next stage in the Runge-Kutta scheme. However, in the multiblock code the flow variables are updated to the new time level block by block. Consequently, on a continuity block boundary the variables required from an adjacent block could be either frozen at the old level or have already been advanced to the new time level. This update approach is easier to code than sweeping throughout the field before continuing to the next stage of the Runge-Kutta scheme, but the mixing of old and new values could impair the convergence rate as the number of blocks is increased. Figure 12 shows the convergence rate of the Euler code for three contrasting flow calculations both subcritical and supercritical using 18 and 54 blocks. Therefore, it is evident that the convergence rate is not significantly impaired by the number of blocks in the flowfield.

Summary

A general grid-generation method designed to provide grid points around a complex geometry has been described. An established formulation to solve the Euler equations has been structured to accept a multiblock grid, and preliminary flow results indicate that the methods described can successfully form a system that can predict the flow around an aircraft

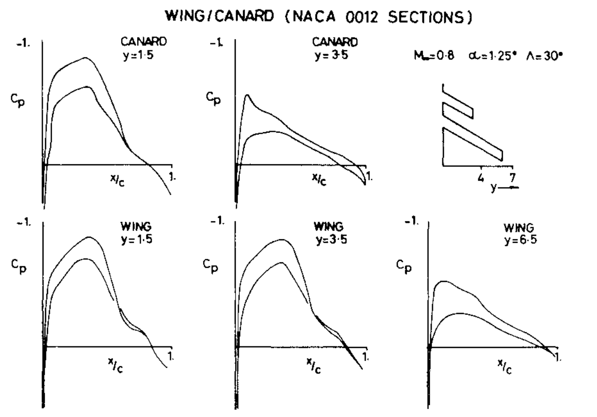

Fig. 11 Pressure coefficients for a wing/canard configuration.

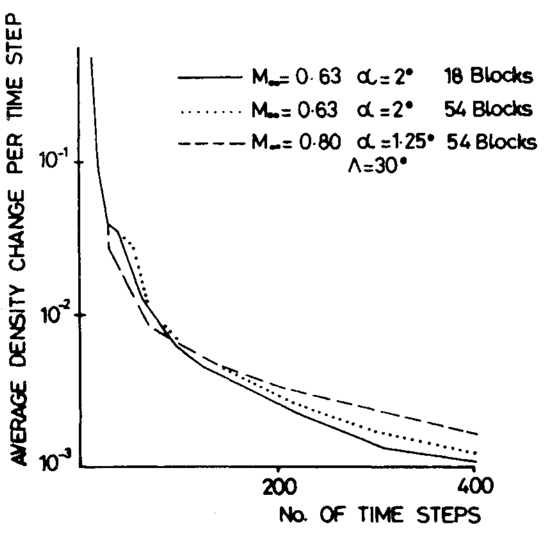

Fig. 12 Convergence history of Euler multiblock approach.

configuration. The system is at the development stage, but it is intended that each part of the multiblock scheme will be integrated together to provide the user with a computer package that requires only the input geometry and topological field structure to enable flow calculations to be performed. Good graphics play an important part in three-dimensional grid generation and flow analysis and a set of graphics tools is being developed concurrently with the main multiblock system. At present, the topological structure of the field grid is derived from freehand sketches of a grid around the configuration. However, this is a daunting task for a complex geometry when perhaps several hundred blocks will be required and the automatic generation of the topology affords a major challenge. Until such a package is available, libraries of topological structures for particular components and configurations can be created that will accommodate most classic configurations. The system has been structured so that new algorithm developments and improvements can be readily incorporated into both the grid and flow codes. For an adequate resolution of the flow, many configurations will require more than 60,000 grid points, implying the need for a much larger computer memory. A backing store version may be an alternative but the practicality of this is presently unclear.

Acknowledgments

This work has been carried out with the support of the Procurement Executive, Ministry of Defence. The authors would like to thank Dr. K. E. Rose for his development work on the plotting software used in the multiblock system and Mr. J. A. Shaw for his contribution to the work on grid control.

References

[1]Holst, T. L., Slooff, J. W., Yoshihara, H., and Ballhaus, W. F., "Applied Computational Transonic Aerodynamics," AGARD-AG-266, 1982.

[2]Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods using Runge-Kutta Time-Stepping Schemes," AIAA Paper 81-1259, 1981.

[3]Atta, E. H. and Vadyak, J., "A Grid Interfacing Zonal Algorithm for Three-Dimensional Transonic Flows about Aircraft Configurations," AIAA Paper 82-1017, 1982.

[4]Benek, J. A., Steger, J. L., and Dougherty, F. C., "A Flexible Grid Embedding Technique with Application to the Euler Equations," AIAA Paper 83-1944, 1983.

[5]Lee, K. D., "3D Transonic Flow Computations using Grid Systems with Block Structure," AIAA Paper 81-0998, 1981.

[6]Miki, K. and Takagi, T., "A Domain Decomposition and Overlapping Method for the Generation of Three-Dimensional Boundary-Fitted Coordinate Systems," *Journal of Computational Physics,* Vol. 53, 1984, pp. 319-330.

[7]Thompson, J. F. and Warsi, Z. U. A., "Three-Dimensional Grid Generation from Elliptic Systems," AIAA Paper 83-1905, 1983.

[8]Thompson, J. F., Thames, F. C., and Mastin, C. W., "Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing any Number of Arbitrary Two-Dimensional Bodies," *Journal of Computational Physics,* Vol. 15, 1974, pp. 299-319.

[9]Roberts, A., "Automatic Topology Generation and Generalized B-Spline Mapping," *Numerical Grid Generation*, edited by J. F., Thompson, North-Holland Publishing Co., New York, 1982.

[10]Coons, S. A., "Surfaces for Computer-Aided Design of Space Forms," MIT MAC-TR-41, 1967.

[11]Powell, M. J. D., UKAEA Harwell Subroutine Library Item No. NS01A, 1968.

[12]Thomas, P. D. and Middlecoff, J. F., "Direct Control of the Grid Point Distribution in Meshes Generated by Elliptic Equations," *AIAA Journal*, Vol. 18, June 1980, pp. 652-656.

[13]Jameson, A. et al., "Accelerated Finite Volume Calculation of Transonic Potential Flows," *Notes on Numerical Fluid Mechanics*, Vol. 3, edited by A. Rizzi and H. Viviand, Vieweg, Germany, 1981, pp. 11-27.