①

AGARD-CP-464

AGARD

ADVISORY GROUP FOR AEROSPACE RESEARCH & DEVELOPMENT

7 RUE ANCELLE 92200 NEUILLY SUR SEINE FRANCE

AGARD CONFERENCE PROCEEDINGS No.464

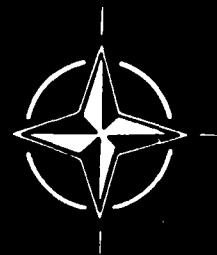# Applications of Mesh Generation to Complex 3-D Configurations
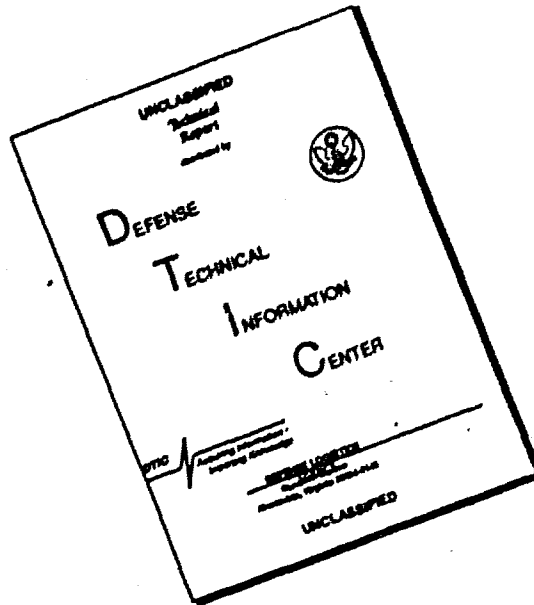
DTIC
ELECTE
MAY 17 1990
S E D

NORTH ATLANTIC TREATY ORGANIZATION

**DISTRIBUTION AND AVAILABILITY
ON BACK COVER**

90 05 16 197

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

AGARD-CP-464

NORTH ATLANTIC TREATY ORGANIZATION

ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT

(ORGANISATION DU TRAITE DE L'ATLANTIQUE NORD)

AGARD Conference Proceedings No.464

APPLICATIONS OF MESH GENERATION

TO COMPLEX 3-D CONFIGURATIONS

THE MISSION OF AGARD

According to its Charter, the mission of AGARD is to bring together the leading personalities of the NATO nations in the fields of science and technology relating to aerospace for the following purposes:

— Recommending effective ways for the member nations to use their research and development capabilities for the common benefit of the NATO community;

— Providing scientific and technical advice and assistance to the Military Committee in the field of aerospace research and development (with particular regard to its military application);

— Continuously stimulating advances in the aerospace sciences relevant to strengthening the common defence posture;

— Improving the co-operation among member nations in aerospace research and development;

— Exchange of scientific and technical information;

— Providing assistance to member nations for the purpose of increasing their scientific and technical potential;

— Rendering scientific and technical assistance, as requested, to other NATO bodies and to member nations in connection with research and development problems in the aerospace field.

The highest authority within AGARD is the National Delegates Board consisting of officially appointed senior representatives from each member nation. The mission of AGARD is carried out through the Panels which are composed of experts appointed by the National Delegates, the Consultant and Exchange Programme and the Aerospace Applications Studies Programme. The results of AGARD work are reported to the member nations and the NATO Authorities through the AGARD series of publications of which this is one.

Participation in AGARD activities is by invitation only and is normally limited to citizens of the NATO nations.

FOREWORD AND CONCLUSIONS

by

Wolfgang Schmidt
Dornier Luftfahrt GmbH
Friedrichshafen
Federal Republic of Germany

During the past years the important role of adequate mesh generation for Computational Fluid Dynamics (CFD) has been fully recognized. Accurate mesh generation has emerged as an indespensible tool in Euler and/or Navier-Stokes calculations. It has been amply demonstrated that the viability of a numerical solution depends directly on the quality of the mesh and surface representation as measured by its spacing and resolution. Of particular interest is the mesh generation for complex configurations, such as advanced fighter or transport aircraft, missiles, or space vehicles, where complex geometries and/or complex flowfields have to be analysed.

There exist numerous reports and proceedings on various methods with many examples such as in References [1] and [2]. The present AGARD Specialists' Meeting on Mesh Generation has been directed towards the application of the different techniques available and the problems encountered if applied to complex cases.

Since two papers could not be given orally, Professor J.Steger agreed to present his paper "Generation of 3-D Body Fitted Grids by Solving Hyperbolic Partial Differential Equations" and Dr Paul Kutler gave a presentation on "CFD at NASA AMES". Professor Steger's presentation has been described in detail in References [4]–[6].

The meeting has been structured in five sessions, giving general surveys by papers one and two in session one, four papers on algebraic grid generation in session two, session three with four papers on block tructured meshes, session four with six papers on multiblock and/or adaptive meshes, and session five with five papers on unstructured meshes. The final paper was an invited paper from the Electromagnetic Wave Propagation Panel showing the mutuality between the computation of electromagnetic wave propagation and CFD, especially in mesh generation.

A detailed evaluation of the material presented has been prepared by the technical evaluator J.Steger. His results are published under separate cover as AGARD-AR-268 [3].

References

[1]   J.F.Thompson, J.L.Steger:
      Three Dimensional Grid Generation for Complex Configurations — Recent Progress. AGARDograph AG-309, 1988

[2]   S.Sengupta, J.Häuser, P.R.Eisemann, J.F.Thompson (Editors):
      Numerical Grid Generation in Computational Fluid Mechanics '88. Pineridge Press, 1988, ISBN 0-906674-68-9

[3]   J.L.Steger:
      Technical Evaluation Report on the Fluid Dynamics Panel Specialists' Meeting on Application of Mesh Generation
      to Complex 3-D Configurations. AGARD-AR-268, 1989

[4]   P.G.Buning, I.T.Chiu, S.Obayashi, Y.M.Rizk, J.L.Steger:
      Numerical Simulation of the Integrated Space Shuttle Vehicle in Ascent. AIAA-88-4359-CP, August 15–17, 1988

[5]   J.L.Steger:
      Generation of Three-Dimensional Body-Fitted Grids by Solving Hyperbolic Partial Differential Equations.
      NASA TM 101069, January 1989

[6]   J.L.Steger, J.A.Benek:
      On the Use of Composite Grid Schemes in Computational Aerodynamics Computer Methods in Applied Mech. and
      Eng. 64, pp.301–320, 1987

AVANT-PROPOS ET CONCLUSIONS

Au cours des dernières années, l'importance de la génération de maillages adéquats pour l'aérodynamique numérique (CFD) a été universellement reconnue. La génération de maillages fidèles apparaît aujord'hui comme un outil indispensable pour la résolution des équations d'Euler et/ou de Navier-Stokes. Il a été amplement démontré que la viabilité des solutions numériques dépend directement de la qualité de la représentation du maillage et de la surface aérodynamique, telle que définie par son pas et sa résolution. La génération de maillages pour des configurations complexes, telles que des aéronefs de combat ou de transport évolués, des missiles, ou des véhicules spatiaux, impliquant l'analyse de géométries et/ou de champs d'écoulement complexes, est d'un intérêt tout particulier.

Il existe de nombreux rapports et comptes-rendus sur les différentes méthodes, contenant beaucoup d'exemples, tels que [1] ou [2]. Cette réunion AGARD de spécialistes, sur la génération des maillages, est orientée vers la mise en application des différentes techniques disponibles et les problèmes rencontrés lorsque ces techniques sont appliquées à des cas complexes.

Deux des conférenciers n'ayant pas pu participer à la réunion, le Prof. J.Steger a bien voulu accepter de présenter une communication sur "La génération de grilles tridimensionelles, pour une représentation affinée de la cellule de l'aéronef, par la résolution d'équations différentielles partielles hyperboliques" et le Dr Paul Kutler a présenté une communication sur "L'aérodynamique numérique (CFD) à NASA AMES". Une description détaillée de la présentation du Professeur Steger est donnée réf. [4] et [6].

La réunion a été organisée en cinq séances, selon le programme suivant:

— séance 1   deux communications servant d'introduction au sujet

— séance 2   quatre communications sur la génération de grilles algébriques

— séance 3   quatre communications sur les maillages structurés par blocs

— séance 4   six communications sur les maillages multiblocs et/ou adaptatifs

— séance 5   cinq communications sur les maillages non structurés

La dernière communication, présentée par le Panel sur la propagation des ondes électromagnétiques, a démontré la relation entre le calcul de la propagation des ondes électromagnétiques et le CFD, en particulier pour la génération des maillages.

Une évaluation détaillée des textes présentés a été réalisée par l'expert en la matière, J.Steger, et ses conclusions ont été publiées sous la forme du document AGARD-AR-268 [3].

FLUID DYNAMICS PANEL

Chairman: Mr D.H.Peckham
Superintendent AE2 Division
Royal Aerospace Establishment
R141 Building
Farnborough Hants GU14 6TD
United Kingdom

Deputy Chairman: Dr W.J.McCroskey
Senior Staff Scientist
US Army Aeroflightdynamics Directorate (AVSCOM)
NASA Ames Research Center N258-1
Moffett Field, CA 94305-1099
United States

PROGRAMME COMMITTEE

Dr W.Schmidt (Chairman)
Deputy Director — DORNIER 328 Program
Dornier GmbH, EY
Postfach 1420
D-7990 Friedrichshafen 1
Federal Republic of Germany

M.J.Verriere
Aérospatiale, Service A DET EG AERO
316 Route de Bayonne
B.P. 3153
31060 Toulouse
France

Professor D.Papailiou
Department of Mechanical Engineering
University of Patras
Rio 26001
Patras
Greece

Prof. M.Onorato
Dipartimento die Ingegneria Aeronautica e Spaziale
Politecnico di Torino
Corso Duca degli Abruzzi 24
10129 Torino
Italy

Professor Ir J.W.Slooff
National Aerospace Laboratory (NLR)
Anthony Fokkerweg 2
1059M Amsterdam
The Netherlands

Professor H.Nørstrud
Division of Hydro- and Gas Dynamics
Norwegian Institute of Technology
N-7034 Trondheim
Norway

Mr P.R.Bignell
BAe PLC, Sowerby Research Centre
Naval Weapons Division FPC 067
P.O. Box 5
Filton
Bristol BS12 7QW
United Kingdom

Dr R.A.Graves
Director, Aerodynamics Division
Mail Code RF
NASA Headquarters
Washington DC 20546
United States

PANEL EXECUTIVE

Mail from Europe:
Mr M.C.Fischer
AGARD OTAN
7 rue Ancelle
92200 Neuilly sur Seine
France
Tel. (1) 4738 5775
Telex 610176 (France)
Telefax (1) 4738-5799

Mail from U.S. and Canada:
Mr M.C.Fischer
AGARD/NATO
APO New York 09777

CONTENTS

SESSION V — UNSTRUCTURED MESHES
Chairman: R.A.Graves

RECENT DEVELOPMENTS IN GRID GENERATION

by

J.Häuser and A.Vinckier
Aerothermodynamics Section
ESA — European Space Research and Technology Center
P.O. Box 299, 2200 AG Noordwijk
The Netherlands

ABSTRACT

This paper gives an overview on recent developments in grid generation with emphasis on the results presented in the proceedings of the Second International Conference in Numerical Grid Generation in Computational Fluid Mechanics '88. Grid generation is essential for the solution of all kinds of fluid physics problems. It also reports briefly about the grid generation activities pursued by the authors, mainly to be used for Hermes. It is particularly important in cases where very different length scales are present, as, for example, in the case of turbulence, which can be considered as the pacing item in present day fluid physics. The main issue of this work deals with multi-block grids in 3-D, but unstructured grids are also briefly mentioned. The important topics in multi-block grid generation are outlined and various approaches to their solution are discussed. The following main building blocks have been identified: Topology of the grid, that is how neighboring blocks are identified and what their relative orientation is to each other; patched (nonoverlapping grid that has grid line continuity only) or matched grids (grids with slope continuity, i.e. continuous tangent vector); block decomposition, which has to be automated if hundreds or thousands of blocks are being used; surface grid generation and analytical description of smooth surfaces to avoid the generation of shocks or expansion fans; grid point clustering (static grids); grid adaptation (dynamic grids) according to specified gradients or function values performed either by redistribution or by local enrichment; postprocessing of grids to visualize and to achieve a specified grid quality at certain points or along certain lines or planes (interactive process).

1. Introduction

Before a numerical solution can be computed, a proper grid must be generated around the vehicle or body of interest. Depending on the physical phenomena during the flight regime, the grid point distribution can vary substantially. Numerical grid generation therefore demands a great flexibility in distributing grid points within the solution domain. The goal is to place grid points at locations where the physics is changing to achieve highest accuracy while only a minimal set of grid points is used. This can be done by using structured grids (a curvilinear CS (Coordinate System) is defined) or by using an unstructured grid where the neighboring grid points are identified by a table of nearest neighbors. The most widely used solution technique is the same for both type of grids -as far as compressible flow is concerned- namely the FV (Finite Volume) technique, which ensures the conservation of the physical quantities and allows for weak solutions.

In the following a brief discussion of the merits and demerits of structured and unstructured grids is given, and recommendations are presented when to use either type of grid. However, this is based on a personal view and there may be different opinions.

2. Structured versus Unstructured Grids

Regarding high speed flow past 3-D objects many flow situations can be encountered where the flow in the vicinity of the body is aligned with the surface, i.e. there is a prevailing flow direction. This is especially the case in hypersonic flow. The use of a SG (Structured Grid), also called body fitted or boundary fitted grid, allows the alignment of the finite FVs in that direction, resulting in locally 1D flow. Hence, numerical diffusion is reduced, i.e. better accuracy is achieved when compared to an UG (Unstructured Grid). Second, SGs can be made orthogonal at boundaries facilitating the implementation of BCs (Boundary Condition) and also increasing the numerical accuracy at boundaries. Furthermore, orthogonality increases the accuracy when algebraic turbulence models are employed. In the solution of the N-S (Navier-Stokes) equations, the BL (Boundary Layer) must be resolved. This demands that the grid is closely wrapped around the body to describe the physics of the BL (some 32 layers are used in general for SGs or UGs). Here some type of SG is indispensable. In addition, to describe the surface of the body a structured approach is necessary. The resolution of the BL leads to large anisotropies in the length scales in the directions along and off the body. Since the time-step size in an explicit scheme is governed by the smallest length scale or, in the case of chemical reacting flow, by the magnitude of the chemical production terms, there will be extremely small time steps necessary. This behavior is not demanded by accuracy but to retain the stability of the scheme. Thus, implicit schemes will be of advantage. In order to invert the implicit operator, factorization is generally used, resulting in two factors if LU decomposition (that is factoring in the direction of the plus and minus eigenvalues) is employed or in three factors if the coordinate directions are used. For the unstructured approach there is no direct way to perform this type of factorization. Moreover, the use of the so called thin layer approach, that is retaining the viscous terms only in the direction off the body, cannot be used. A reduction of 30% in computer time has been reported. Since there are no coordinate lines in the UG, this simplification is not possible. Moreover, the flow solver based on the UG is substantially slower than for the SG. This is due to the more complicated data structure needed for UGs. Factors of 3 [38] and by some authors of up to 10 [39] have been given in the literature.

Although it might be thought that CPU time is no longer a critical item with the next generation of computers, this will not be true if turbulent flows and transition phenomena are to be modeled. Suppose there is a Cray4 that is a 100 times more powerful than the present Cray2 and suppose we need a factor of three more in CPU time and memory based on 10 Million grid points. This would amount to additional 299 present day Cray2s, and using 100 words per gridpoint would demand additional 16 Gbytes of memory. Since transition and

turbulence are the driving force for future applications in aerospace, any additional increase in computing speed and memory has to be used to improve these solution of the physical phenomena.

An important point for the accuracy of the solution is the capability of grid point clustering and solution adaptation. In general, SGs provide sophisticated means both for clustering and adaptation using redistribution or local enrichment techniques. A comparison of these two approaches is given in [14] where local enrichment gives somewhat better results. However, it is much more costly to use. The highest degree of freedom of course is obtained in UGs. We feel, however, that mesh redistribution is totally adequate for the major part of the flow situations encountered in external flows, especially in aerodynamics. If a very complex wave pattern due to special physical phenomena evolves, for example, generating dozens of shock waves, then the UG has advantages [30]. In addition, the coupling of SGs with UGs is possible as has been shown in [17]. Such a grid is called a hybrid grid.

Attention should also be given to the use of parallel computers. Massively parallel systems with several hundred of processors will soon become available. Here block-structured grids can be very important, since, in principal, each processor can iterate the solution for each block and then update the boundaries of the neighboring blocks by passing a message. Overlapping block-structured grids are very well suited for this type of computer. However, in 3-D, overlaps should be restricted to one face only to avoid the storage of grid points that are not active, (see Fig.3). It is important to note that the solution procedure for storage coupled or private memory machines should allow the application to general geometries, i.e. in the case of multi-block (see Chap.3) the CP does not have a regular geometry but can be quite fragmented. This poses difficulties for machines using the wave concept. First results for general 2-D geometries are presented in [4].

3. Multi Block versus Single Block

If the topology of the SD and the domain in the CP are not of the same order, certain grid line configurations cannot be realized (see lower part of Fig.1b). Thus a n-fold connected region in the PP should be mapped onto a n-fold connected region in the CP. This is achieved by using the so called multi-block approach (Fig.2). A grid can be comprised by matching blocks (see Fig.2) where slope continuity is provided, i.e. neighboring blocks overlap by one row or one column. Looking at such grids, block-boundaries are not visible, which is the most suitable case for the flow solver, since, of course, the flow solution must not depend on the blocking of the SD. This can be avoided by the following construction process: first a fairly coarse grid utilizing the overlapping approach is constructed, and, in a second step, the grid is doubled for selected blocks only, without changing the positions of the grid points on the coarse mesh. This approach naturally leads to a local refinement that easily ensures flux conservation and also allows the direct application of the multigrid-technique. A simpler requirement is to demand grid line continuity only and the next step is to also give up that feature, which results in more freedom in grid generation, but demands substantial efforts in the flow solver to guarantee the conservation of fluxes across block boundaries. The best approach, of course, is depending on the physics that has to be modeled.
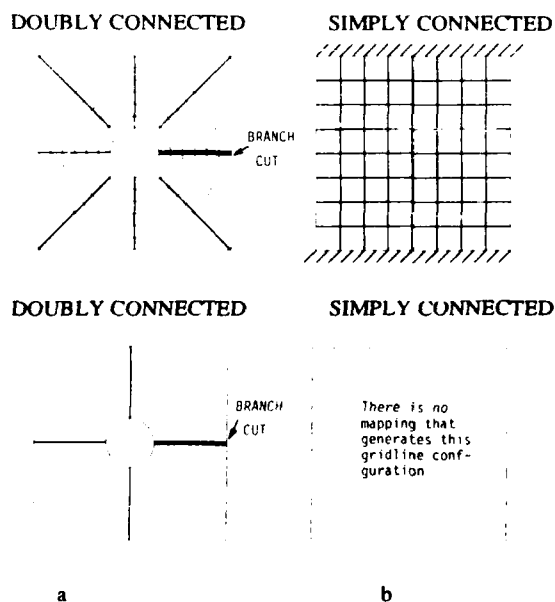
Fig.1: The upper part of (a) shows an O-type grid which, by a branch cut, can be mapped onto a single rectangle in the CP. There are situations where an H-type grid (lower part of (a)) is advantageous, e.g. flow past a cylinder. This grid line configuration cannot be obtained by a branch cut. Using cuts always maps a n-fold connected SD onto a simply connected domain in the CP. Here the idea of multi-block grids is essential in generating a mapping retaining the connectivity of the original SD. Hence, multiple cuts and multi-blocks are not equivalent.
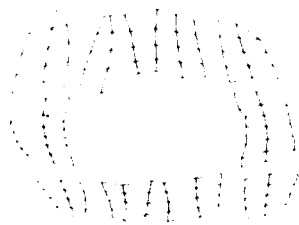
Fig. 2. Grid with slope continuity. Eight blocks are necessary if the same number of grid points is required at neighboring edges. The multi block idea therefore is to map the SD onto a set of connected rectangles in 2-D or boxes in 3-D. The same approach can be used for curved surfaces in 3-D, which can be thought to be patched or matched by a set of charts forming an atlas where each chart is a manifold, i.e. is locally equivalent to a plane surface.

4 Key Issues of 3-D Multiblock Grids

In general, geometry data is given by some type of CAD system. This data is to be processed to generate the required surface grid. The surface grid serves as a type of boundary for the volume grid to be constructed, and thus strongly influences the overall grid quality. The following gives an overview of the key issues in 3-D multi-block grids. In Table 1 the complete process for the design of a vehicle of complex geometry is outlined, starting from the lower left corner. For example, at present, a large amount of time has to be spent to process the data from e.g., CATIA or Euclid to construct a computational grid for the Hermes Space Plane. In the following we briefly present the key issues of 3-D multi-block grid generation. Of course, not all the topics mentioned below can be treated in depth in this paper. One of the major problems encountered so far is the large amount of human interaction to first produce a surface grid and the subsequent generation of the multi-block grid. Moreover, the blocking of the grid is not straightforward. Here, some type of knowledge based system could be useful [1] or the technique described in [24] could be employed, which can be considered to be a major step forward in automating the decomposition of the SD.

(i)   Body Geometry Definition and Geometric Modeling

- Construction of Coons patches for analytical surface description, or similar technique

- Graphics editor

(ii)   Surface Grid Generation

- 2-D grid topology

- 2-D multi block grids on curved surfaces

- direct projection method

  generalized 2-D elliptic grid equations

- database for 2-D surface patches

- analytical definition (e.g. body of revolution)

(iii) Data Structures for 2-D and 3-D Grid Topology (Block Connection)

- interconnection field

- pointer arrays

- Gray Code

(iv)   Grid Generation Techniques

- algebraic

- PDE (Elliptic)

- iterative (SOR etc.)

- multigrid

- hyperbolic (outer boundary not specified)

(v)   Grid Point Clustering and Adaptive Grids

- orthogonality

- distance control

- solution adaptation

- local enrichment

(vi)   Interactive Grid Generation

(vii)   Grid Visualization

(viii)   Grid Generation on Parallel Computers

   - Storage coupled computers (≤ 16 Processors)

   - Massively parallel systems (MIMD, ≥ 16 Processors)

(ix)   Expert Systems

   - reduction of human interaction

   - reduction of turn around time

(x)   Computer Languages for Grid Generation

   - F77, C, C++ for complex data structures
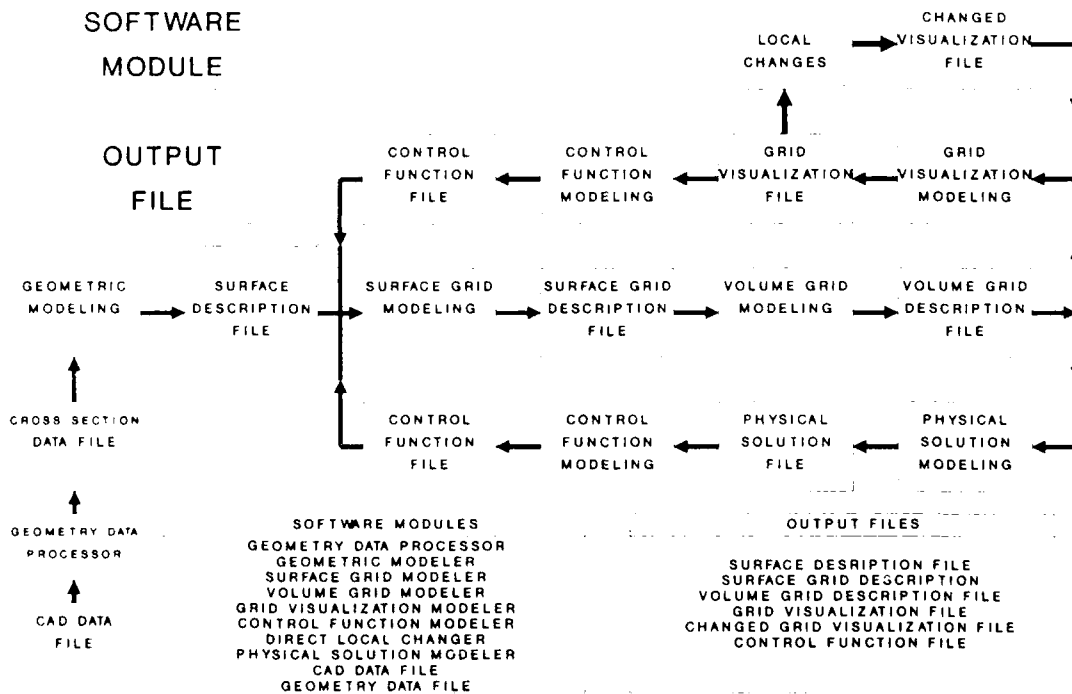
GENERAL 3-D MULTIBLOCK ADAPTIVE GRID GENERATION

Table 1   Flow chart for the flow solution process starting from the CAD data file (lower left corner) using a modular approach The output file of one module is the input file of the following one, similar to the Unix pipe concept
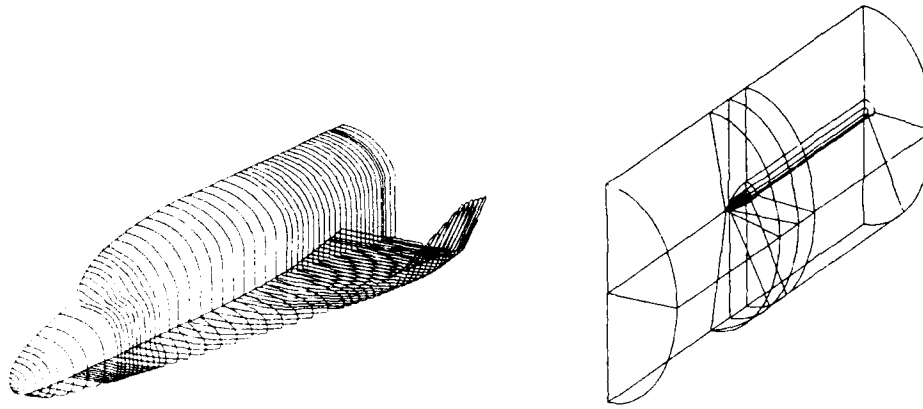
Fig 3 The upper part shows the cross sections for the Hermes Space Plane. The original tape contains spline coefficients from CATIA. This information then is processed to obtain the cross sectional curves. Grid points can be distributed by a user specified weight function, e.g. curvature. It is essential to construct a smooth surface to prevent the generation of shocks or expansion fans. Commercial packages, e.g. PATRAN, can therefore not be used to generate a proper surface. The lower part shows the block structure as described in [22]. A software package was written that generates a sequence of cross sections from CATIA data generating the desired grid point distribution on each cross section for the Hermes space plane.

5 Surface Grid Generation

Eq.(1) gives the general description of a parametric surface needed to describe the surface of a vehicle (body):

$$x^i = x^i(u^1, u^2); \, i=1,2,3 \tag{1}$$

In component form using (u,v) as independent variable this can be written as

$$x = x(u,v) \, ; \, y = y(u,v) \, ; \, z = z(u,v) \, . \tag{2}$$

Let the surface of the body be the plane ξ = constant, two mappings have to be considered for the surface grid: From the physical space to parameter space and from there to the Computational Space with uniform grid spacings $\Delta\xi = \Delta\eta = \Delta\zeta = 1$. The surface grid can then be generated in various ways. For example, if Eqs.(2) are known in form of a set of bicubic splines (Hermite, Bezier, B-Splines), the general form of 3-D elliptic grid generation equations can be used taking into account the constraints represented by Eqs.(2). We start from Eqs.(9) (see Chapter 7 for details), which are the 3-D grid generation equations

$$g^{ik} \frac{\partial^2 x^i}{\partial \xi^i \partial \xi^k} = g^{nn} P_n \frac{\partial x^i}{\partial \xi^n} \, ; \quad i=1,2,3 \tag{2}$$

Using (u^1, u^2) instead of (u,v), inserting the parametric representation in Eqs.(2) in Eqs.(1), one obtains:

$$x^i = x^i(u^1(\xi,\eta,0), u^2(\xi,\eta,0)) \, ; \, i=1,2,3 \tag{3}$$

Employing the chain rule for first derivatives

$$x^i_\xi = x^i_u u_\xi + x^i_v v_\xi \, ; \, i=1,2,3$$

$$x^i_\eta = x^i_u u_\eta + x^i_v v_\eta \, ; \, i=1,2,3$$

(second derivatives are derived in a similar way), results in the following form of the final grid generation equations

$$\alpha_1 u_{\xi\xi} + \alpha_2 J^2 P u_\xi - 2\beta u_{\xi\eta} + \gamma_1 u_{\eta\eta} + \gamma_2 J^2 Q u_\eta + J^2 \vec{C} \left[\alpha' e_u - \beta' e_v \right] |e_u \times e_v|^2 = 0 \tag{4}$$

$$\vec{C} := \alpha' e_{uu} - 2\beta' e_{uv} + \gamma' e_{vv}$$

$$\alpha' := e_v e_v \, ; \, \beta' := e_u e_v \, ; \, \gamma' := e_v e_v$$

A similar equation holds for v. Vectors e_u, e_v denote tangent vectors. It should be noted that Eqs.(4) ensure that points will move on the surface and that grid point distribution control functions P,Q,R can be determined for example from the boundary point distribution or by additional requirements that can be imposed on them (see below).

In order to improve the accuracy of the solution near the body surface, additional constraints are imposed on the grid on the surface:

It is however mandatory to have control over the grid quality near the boundaries, therefore the following constraints are imposed where the body surface is assumed to be the plane $\xi = 0$

(i) Orthogonality [2]:

$e_\zeta.e_\xi = 0$; $e_\zeta.e_\eta = 0$: $Plane$: $\zeta = 0$

$e_{\bar{\gamma}} := \partial x/\partial \xi$; $x = (x,y,z)$

(ii) Cell size control:

$e_\zeta.e_\zeta = (h(\zeta))^2$ = Height of a cell, i.e. distance of next layer of grid points from body surface.

(iii) Vanishing of principal curvature of ζ-directed lines, smoothness property.

$e_\xi.e_{\zeta\zeta} = 0$; $e_\eta.e_{\zeta\zeta} = 0$

(iv) Iterative calculation of control functions [6], for specification of angles of intersection with vertical gridlines and body surface, here for angle α (Q determines angle β)

$$P^{n+1} = P^n + \Delta P \quad ; \quad \Delta P = -arctan\left[\frac{\alpha_{req} - \alpha}{\alpha_{req}}\right]$$

where α_{req} is a user specified angle and α denotes the current angle. (ii) and (iii) are special cases of (iv).

6. Data Structures for 2-D and 3-D Grid Topology

In a multi-block grid each block has its own local coordinate system. In the matching of neighboring blocks there are the following possibilities for 2-D and 3-D SDs, see Figs.3 and 4, respectively.

(i) 2-D: Each side of the current block can be matched with a side of a neighboring block in two ways. Hence, there are 4×2 possibilities of matching a side of a block with a specific side of a neighboring block.

(a)

(b)
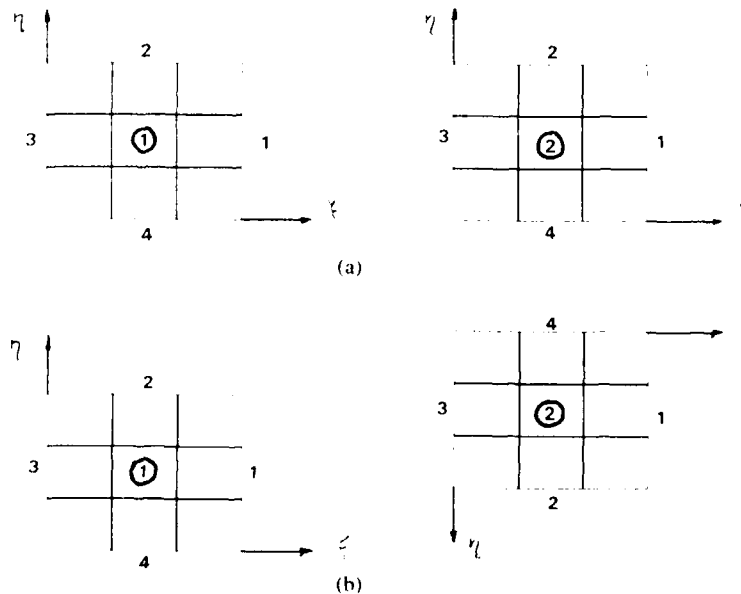
In Fig (4) side 1 of block 1 is matched to side 3 of block 2 in the same direction; i.e. points with the same η values are matched. In (b) the matching direction is reversed. This freedom is absolutely necessary to construct flexible grids.

(ii) 3-D Each face of the current block can be matched with the face of a neighboring block in 4 ways. Hence, there are 6×4 possibilities of matching a face of a block with the specific face of a neighboring block.

(a)

(b)
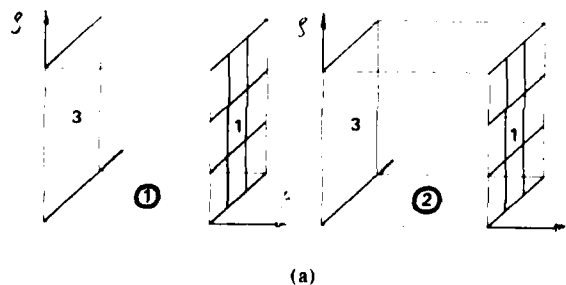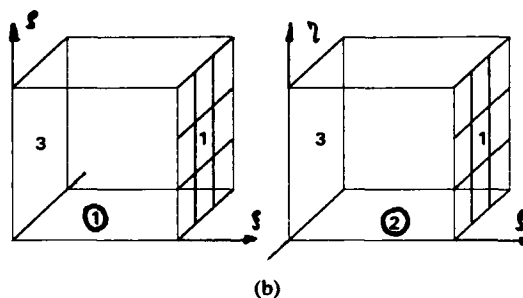
In Fig (5a) face 1 of block 1 is matched to face 3 of block 2 in the same direction. (b) shows the matching where block 2 was rotated about the ξ-axis by an angle of $\pi/2$.

The appropriate data structure describing neighboring blocks along with the matching sides has to be designed; e.g. [40]. Here, a modern high level language like C or C++ is advantageous, since it allows the definition of complex data structures. A short piece of code, taken from [41], shows how the orientation of neighboring blocks with respect to each other is determined. Values rot_x, rot_y, rot_z varying from 0 to 3, determine rotation of a face from 0 to $3/2\pi$. The routine returns values for ni, nj, nk that describe the orientation of the local CS of the neighboring block with respect to the neighboring block. In C, variables provided with an asterisk, denote so called pointers, i.e. indirect addressing is used.

```
void orient(rot_x,rot_y,rot_z,ni,nj,nk)
unsigned rot_x,rot_y,rot_z;
unsigned *ni,*nj,*nk;
{
  void mul();
  static int MZ[3][3]={0,-1,0,1,0,0,0,0,1},MY[3][3]={0,0,1,0,1,0,-1,0,0},
      MX[3][3]={1,0,0,0,0,-1,0,1,0},v[3]={1,2,3};
  int i,j,k;
  for (i=0;i<rot_x;i++)
    mul(MX,v);
  for (j=0;j<rot_y;j++)
    mul(MY,v);
  for (k=0;k<rot_z;k++)
    mul(MZ,v);
  *ni=v[0];*nj=v[1];*nk=v[2];
}

void mul(A,v)
int A[][3],v[];
{
  int C[3];
  int i,j;
  for(i=0;i<3;i++)
    {C[i]=0;
      for (j=0;j<3;j++)
        C[i]+=A[i][j]*v[j];}
  for(i=0;i<3;i++)
    v[i]=C[i];
}
```

*Fig 6 : Illustration of determination of orientation of local CS of neighboring block, using the C-language. Among others advantages, pointers (denoted by *) allowing indirect addressing can be used.*

In Fig.7 we see the block structure for a 2-D multi-block grid for an airfoil where an H- and C- type grid are merged. As can be seen, FVs with more than four vertices can be generated, which can easily be handled by the flow solver. The blocking here is not so straightforward since common edges must have the same number of grid points overlapping by one row or one column.
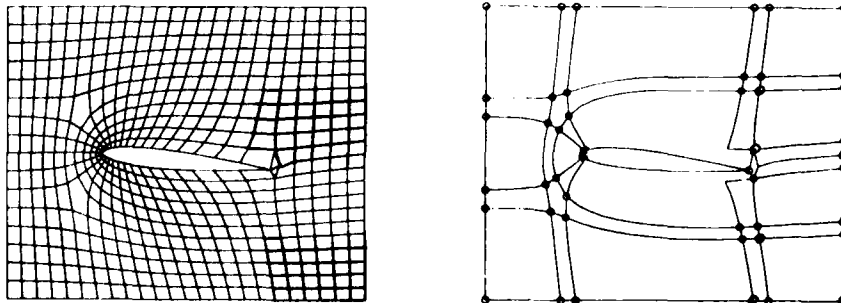
Fig 7: The block structure is shown for an airfoil generating a grid which is both of H- and C- type structure.[3]
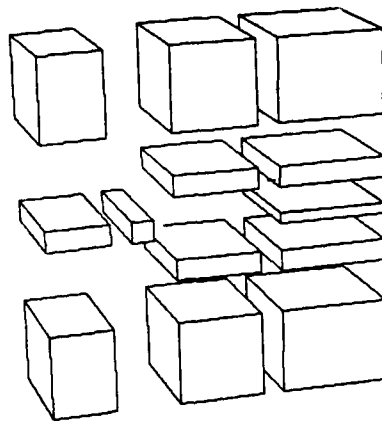
Fig 8: Block structure for the 3-D grid, the fuselage-wing intersection shown in the first part of Fig.8.[40]
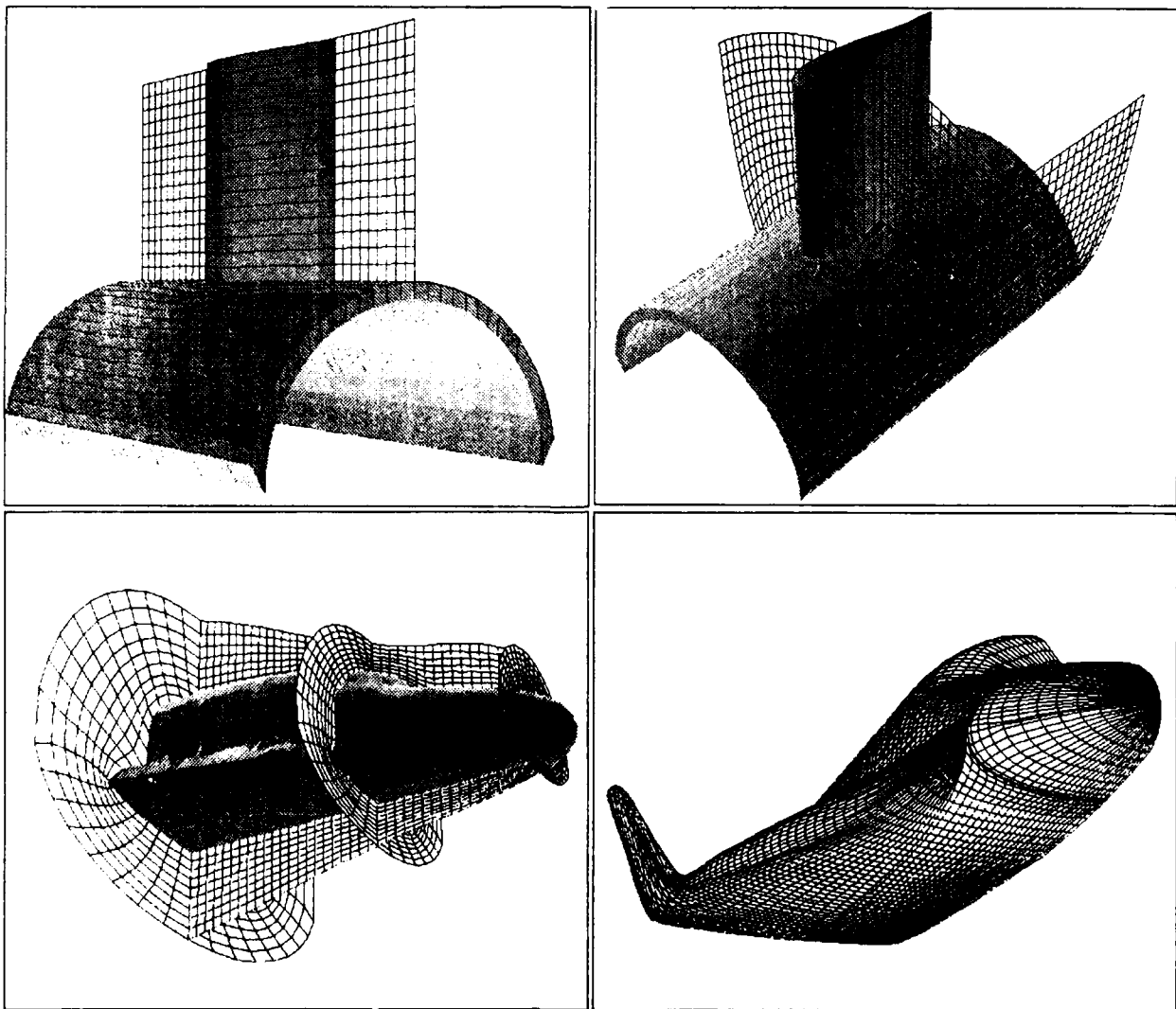
Fig 9: The first two pictures show the 3-D grid for a fuselage-wing configuration. The other two figures show a grid around a generic body, having some similarity with the space shuttle. The figure on the lower right depicts a surface grid for the Hermes space plane using about 10000 points. The purpose of this figure is to demonstrate the importance of the combination of sophisticated visualization software together with advanced supergraphics workstations, that are indispensable for work in aerodynamics.

7. The Grid Generation Procedure

In general, generating the grid point positions for 2-D and 3-D grids has to be done by numerical techniques. This can be done by algebraic grid generation [12] (transfinite interpolation) or by solving PDEs which can either be of hyperbolic [19] or elliptic [8] type. Using PDEs demands first the establishment of the system of equations to be solved (one equation for each coordinate direction). Hyperbolic equations do not need an outer boundary, rather the grid can be started from the surface of the body and then be advanced into the SD. The procedure is stopped if the grid is far enough from the vehicle, that is all the important physics is contained in the gridded region. Of course, this procedure only works if the outer boundary is not prescribed. In such a way an orthogonal grid can be constructed, along with the specification of the grid cell size. Transfinite interpolation, which is multi-directional interpolation, has its name from the fact that it matches the function on the entire boundary, i.e. at a nondenumerable number of points. Interpolation is normally very fast in generating a grid, which is however, not necessarily smooth. If splines are to be used as blending functions, a grid with continuous second derivatives is obtained, that is more costly to generate. The multi-surface interpolation is a unidirectional interpolational procedure where interpolation from a vector field along with vector normalization at each interpolation point is used in order to match boundaries. Both techniques, algebraic and hyperbolic grid generation normally need some smoothing, which is done using an elliptic technique. The use of elliptic PDEs gives rise to the solution of a boundary value problem, i.e. the boundaries along with appropriate BC's have to be prescribed. Therefore we will outline in some detail the essentials of elliptic grid generation. Recalling an example from electrostatics, namely the picture produced by electric field lines and equipotential lines formed by a set of two charged capacitors, one obtains a smooth mesh where grid lines are concentrated automatically in regions of higher curvature. Since the physics is described by the Laplace equation it is natural to use this equation together with a nonzero RHS, to provide additional control over the grid line distribution.

In the Physical Plane (PP) the elliptic grid generation has the form

$$\xi_{xx} + \xi_{yy} + \xi_{zz} = P$$
$$\eta_{xx} + \eta_{yy} + \eta_{zz} = Q \qquad (5)$$
$$\zeta_{xx} + \zeta_{yy} + \zeta_{zz} = R$$

Using a more concise notation

$$\xi^1 = \xi \; ; \; \xi^2 = \eta \; ; \; \xi^3 = \zeta \; ; \; x^1 = x \; ; \; x^2 = y \; ; \; x^3 = z$$

one can write

$$\nabla^2 \xi^i = g^{ii} P_i(\xi^1, \xi^2, \xi^3) = P^i \; ; \; i = 1, 2, 3 \qquad (6)$$

where g^{ii} denote the contravariant components of the metric tensor g (see below). All equations, that is the grid generation and the physical equations are solved in the CP and therefore have to be transformed along with their BCs. The transformation rule for the Laplacian applied to a scalar function $\Phi(x)$ where $x = xi + yj + zk$ is the position vector, is of the form

$$\nabla^2 \Phi = g^{ii} \Phi_{\xi^i \xi^i} + \left[\nabla^2 \xi^i \right] \Phi_{\xi^i} \qquad (7)$$

with

$$g^{ii} = \frac{Det \; (ij \; minor \; of \; g \;)}{Det \; g}$$
$$g = (g_{ik}) = (e_i.e_j)$$
$$e_i = \frac{\partial x}{\partial \xi^i}$$

Let $\Phi := x^l$ and note that $\nabla^2 x^l = 0$. Hence

$$g^{ij} x^l_{\xi^i \xi^j} + \left[\nabla^2 \xi^k \right] x^l = 0 \qquad (8)$$

where Einstein's summation convention is used; double indices are summed over. Substitutions of Eq.(6) into Eq.(8) results in

$$g^{ij} x^l_{\xi^i \xi^j} + P^k x^l_{\xi^k} = 0 \; ; \; l = 1, 2, 3 \qquad (9)$$

Rewriting the latter for computational efficiency gives

$$a_{11} x^l_{\xi\xi} + a_{22} x^l_{\eta\eta} + a_{33} x^l_{\zeta\zeta} + 2 \left[a_{12} x^l_{\xi\eta} + a_{13} x^l_{\xi\zeta} + a_{23} x^l_{\eta\zeta} \right] + J^2 \left[P x^l_{\xi} + Q x^l_{\eta} + R x^l_{\zeta} \right] = 0 \; ; \; l = 1, 2, 3 \qquad (10)$$

$$a_{ij} = A_{mi} A_{mj}$$
$$A_{mi} = (-1)^{m+i} \left[mi \; minor \; of \; \left[\frac{\partial x^i}{\partial \xi^k} \right] \right]$$
$$J = Det \left[\frac{\partial x^i}{\partial \xi^k} \right]$$

The source functions, P_n, are used to influence the distribution of the coordinate lines or surfaces in the physical domain. This system of equations is discretized and solved numerically. Very often this is done by iteration. If multigrid techniques are used, a substantial speed up can be obtained. In that case, elliptic grid generation will be competitive with the algebraic technique.

8. Mesh Redistribution and Local Enrichment

The RHS of the grid generation equations (see Chap.7) contains the so called control functions, P,Q, and R. These functions can be used to redistribute the mesh point configuration according to the grid point distribution specified on the boundaries, which is used to calculate the values of the control functions and then interpolates them into the SD. Hence, the grid in the interior shows the same grid point distribution as on the boundary. Fig.10 shows the effect of the control function on the grid point locations and Fig.11 shows a redistributed mesh for a cross section of the Space Shuttle.
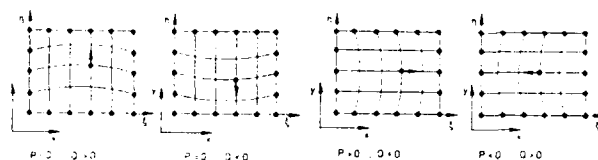
Fig 10: This picture, taken from [6], shows the impact of control functions on the grid point distribution. Control function values larger than 0 move the grid lines to higher coordinate values. Movement in the opposite direction is achieved for negative control function values.
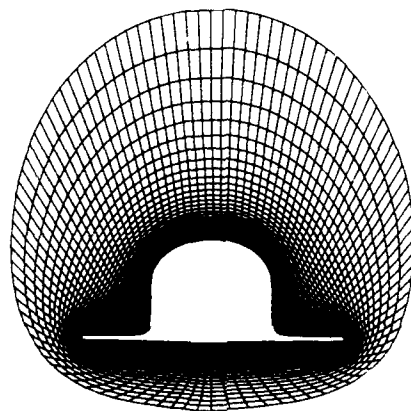
Fig 11: Redistributed mesh for a space shuttle cross section via control functions

In Fig.12 the effect of control functions P,Q on the angles β and α is depicted for the plane ζ = const. Function R controls the distance of the next layer of points. In order to prevent the influence of these control functions into the SD, they are multiplied with an exponentially decreasing function. In order to reduce the number of iterations, a good initial guess for P,Q,R should be supplied [6].
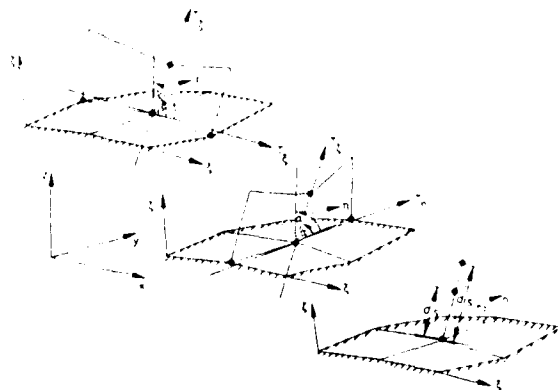
Fig 12: Control functions as used in [6] for the control of angle for grid lines in the direction off the body, i.e. angles α and β can be specified. The third control function controls the distance of the next layer of grid points from the surface. Control functions are calculated iteratively. This approach allows a good control of grid point distribution.

Figs 13 and 14 show a comparison of grid redistribution and local enrichment. The grid is redistributed according to the change in the physical solution, e.g. pressure or density using some type of either finite approximation of the gradient or the second derivative. It is important to ensure that grid lines cannot cross over. Details can be found in [14].

(a) Redistributed grid

(b) Embedded grid

(a) Redistributed grid

(b) Embedded grid
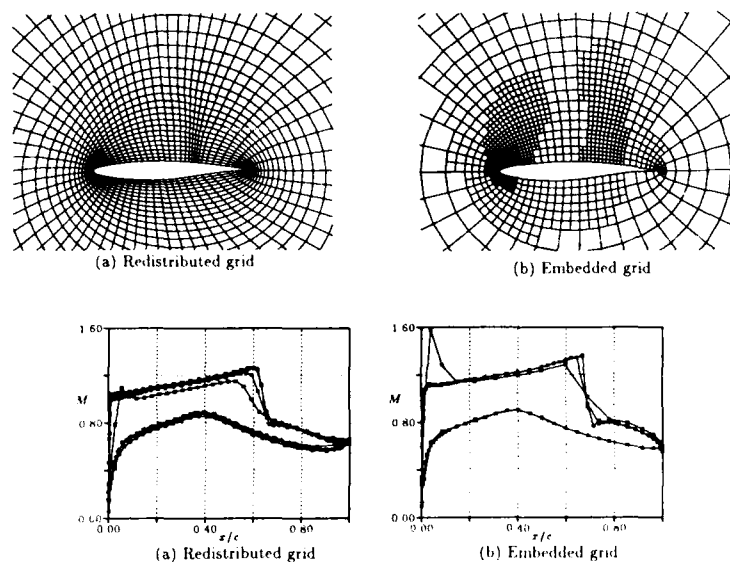
Surface Mach number distributions for the RAE-2822 airfoil at $M_\infty = 0.75$

Fig 13: Grid point redistribution versus local enrichment [8]. Although enrichment results in a somewhat sharper shock, the data structure is more complicated if multi-block grids are used. However, mesh redistribution results in a simpler code. An alternative is to use enrichment for complete blocks only so flux conservation across neighboring faces is easily achieved.

A Comparison of Two Adaptive Grid Techniques

(a) Mach number contours on initial, non-adapted grid, $\Delta M = 0.10$

(b) Redistributed grid

(d) Embedded grid

(c) Mach number contours on redistributed grid, $\Delta M = 0.10$

(e) Mach number contours on embedded grid, $\Delta M = 0.10$
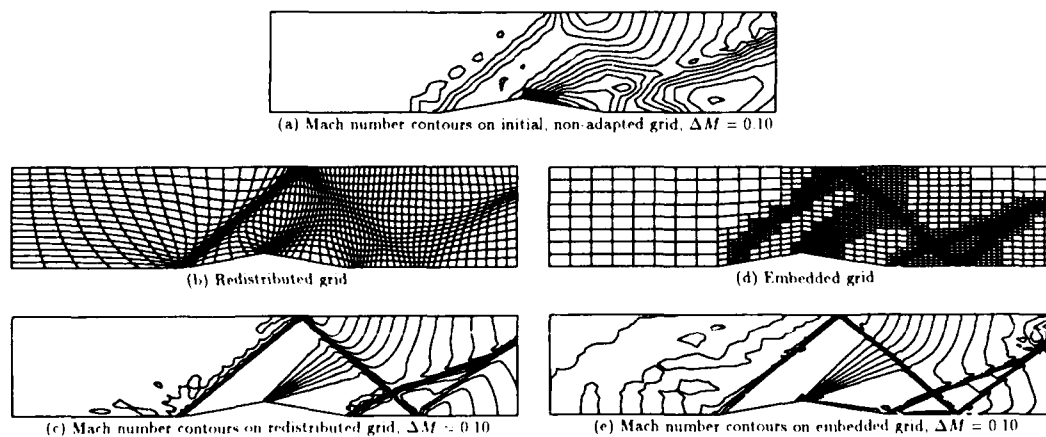
Adapted solutions for $M_\infty = 2.00$ flow in a duct with $10°$ wedges

Fig 14: Grid point redistribution versus local enrichment for SGs taken from [8]. Shock capturing features of both techniques are very good where local enrichment gives somewhat better results.

9. Unstructured Grids

In the past, UGs have been used mainly in structural mechanics, in connection with the finite element method (FEM). Recently UGs have been applied to fluid dynamics, and some impressive results have been obtained, see, e.g., Fig.15. This is not in contradiction to our discussion in Sec.2.
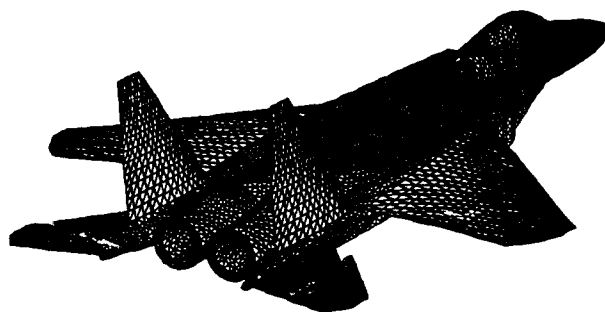
Fig. 15· Detail of surface triangulation from a tetrahedral mesh around the F15 aircraft, taken from [29]

The major concern is that UGs using the FEM will demand much more hardware resources and higher computation times. Most UGs are generated using a Delaunay triangulation that will produce the most equiangular triangles possible. For viscous flow this is not well suited since a directionally refined mesh is needed. The generation of this type of mesh is described in [28]. However, UG generation is at least an order of magnitude slower compared to the block-structured technique.

10. Grid Generation on Parallel Computers

In the preface of the proceedings of the First Grid Generation Conference [43], the possible impact of parallel computers was briefly discussed, and the importance of parallel algorithms for CFD was stressed. In a recent survey paper by Holst [42], the number of floating point operations for the solution of the 3-D N-S equations exceeds 10^{13} for the hypersonic case with a grid of some 10^6 points. This estimate is valid for the stationary case only and does not include the solution of equations needed to account for chemical reactions. Thus, for realistic hypersonic equations where up to some 10^7 grid points may be necessary, and, including transition and turbulence phenomena as well as thermo- chemical nonequilibrium, 10^{15} or even 10^{16} floating point operations have to be performed. Clearly, this tremendous number of floating point operations cannot be handled by a single processor in a time and cost effective manner. The solution therefore lies in the use of a computer architecture employing a set of processors working in parallel.

Using the overlapping multi-block approach, it is naturally to ascribe one or more blocks to each processor on which to solve the Poisson equation (for grid generation) or the physical equations. This process is called DC (Domain Decomposition). Grid generation is a good example for DC, since exactly the same structure is used for the flow solver, which is computationally much more expensive. However, it is important to parallelize the code in such a way that there are no dependencies on the shape of the SD, i.e. in some papers the SD was assumed to be of rectangular shape, which was then subdivided. For the general case, the SD is a set of connected rectangles or boxes and its shape must not influence the speedup of the parallel algorithm [4].

We concentrate here on massively parallel systems, e.g. Intel's hypercube or the Suprenum machine [37], which are private memory (several MB), message passing systems where each processor has a power of several MFLOPs.

It is often stated that scientific programs have several percent of serial computational work, s, that limits the speedup, a, of parallel machines to an asymptotic value of $1/s$ according to Amdahl's law where $s+p=1$ (normalized) and n is the number of processors:

$$a = \frac{s+p}{s+p/n} = \frac{1}{s+p/n} \tag{11}$$

This law is based on the question: Given the computation time on the serial computer how long does it take on the parallel system? However, the question can also be posed in another way: Let s', p' be the serial and parallel time spent on the parallel system then $s'+p'n$ is the time spent on a uniprocessor system. This gives an alternative to Amdahl's law and results in the speedup which is more relevant in practice:

$$a = \frac{s'+p'n}{s'+p'} = n+(1-n)s' \tag{12}$$

It should be noted that DC does not demand the parallelization of the solution algorithm but is based on the partitioning of the SD; i.e. the same algorithm on different data is executed. With that respect, the serial parts s or s' can be set to 0 for DC and both formulas give the same result. The important factor is the ratio r_{CT} (see below), which is a measure for the communication overhead. In general, if the selection algorithm is parallelized, Amdahl's law gives a severe limitation of the speedup, since for $n \rightarrow \infty$ a is equal $1/s$. If, for example, s is two percent and n is 1000, the highest possible speedup from Amdahl's law is 50. However, this law does not account for the fact that s and p can be functions of n. As described below the number of processors, the processor speed and memory are not independent variables, which simply means, if we connect more and faster processors, a larger memory is needed, leading to a larger problem size, and thus reducing the serial part. Therefor speedup increases. If s' equals two percent and $n = 1024$, the scaled sized law will give a

speedup of 980, which actually has been achieved in practice. However, one has to keep in mind that s and s' have different values. If s' denoted the serial part on a parallel processor in floating point operations, it is not correct to set $s = s'n$ since the solution algorithms on the uniprocessor and parallel system are different in general.

For practical applications the type of parallel systems should be selected by the problem that has to be solved. For example, for routine applications to compute the flow around a spacecraft on a grid of 10^7 grid points, needing some 10^{14} floating point operations, computation time should be some 15 minutes. Systems of 1000 processors can be handled, so each processor has to do some 10^{11} computations, and therefore a power of 100 MFlops per processor is needed. Assuming that some 100 words, 8 bytes/word, are needed per grid point, the total memory amounts to 8 GB, that means 8 MB of private memory for each processor, resulting in 22 grid points in each coordinate direction. The total amount of processing time per block consists of computation and communication time:

$$t_p = N^3 * 4000 * t_c + 6N^2 * 10 * 8 * t_T \tag{13}$$

where we assumed that 4000 floating point operations per grid point are needed, and 10 variables of 8 byte length per boundary point have to be communicated. Variables t_c, t_T are the time per floating point operation and the transfer time per byte, respectively. For a crude estimate, we omit the set up time for a message. Using a bus speed of 250 MB/s (quite high), we find for the ratio of computation time and communication time.

$$r_{CT} := \frac{N^3 * 4000 * 250}{6N^2 * 10 * 8 * 100} \approx 20N \tag{14}$$

That is for N = 22, communication time per block is less than 0.25% of the computation time. In that respect, implicit schemes should be favoured, because the amount of computation per time step is much larger than for an explicit one.

In order to achieve the high computational power per node a MIMD/SIMD (Multiple Instruction Multiple Data; Simple Instruction Multiple Data) architecture should be chosen; that means, the system is of massively parallel architecture, e.g. Suprenum, and each processor itself is equipped with a pipelined floating point processor. It should be noted that even if $r_{CT} \gg 1$ this is not sufficient, since, if the computation speed of the single processor is small, e.g. 0.1 MFlops, this will lead to a large speedup which is, of course, somewhat misleading because the high value for r_{CT} only results from the low processor performance. In conclusion, it is believed that the concept of MIMD/SIMD is the most promising for computationally intensive applications in fluid physics and DC will be a powerful concept to tackle problems demanding excessive number crunching.

11. Computer Issues in Grid Generation

Fortran 77 is the language most widely used in science and engineering. This language has quite a number of deficiences as will become clear obvious when compared with C. It should be kept in mind that Fortran (or Ratfor) is a subset of C, and that using this part of C will only take a couple of days for an experienced F77 programmer. If the more advanced parts of C are going to be used, a longer training period is needed, resulting in higher productivity and a safer code. In the following a brief comparison between F77 and C is given:

- C contains all F77 possibilities
- C has advantages with respect to automatic debugging (function prototyping)
- C allows for dynamic storage allocation
- C allows for the definition of complex data structures
- C programs are much shorter and well structured
- C has an interface to many graphics packages
- C allows recursive programming
- C is portable
- F77 programs can be called from C

The recommendation is: If there are no historical constraints, the C language should be preferred, in particular for 3-D multi-block grids where the data structure is more complicated and especially for UGs, which have a much more complex data structure. With the growing popularity of the UNIX operating system, a C compiler will be available from a simple PC to the largest supercomputer, even DEC and IBM have now joined in the UNIX market.

The second important part is visualization of 3-D grids and also of the flowfield properties. Very powerful graphics superworkstations are now available, e.g. Silicon Graphics, Ardent, Stellar, Alliant etc. which provide very sophisticated graphics hardware that frees the user from the burden of low level graphics programming, for example, hidden surface detection by the use of bitplanes, which results in a speedup of several orders of magnitude compared with dumb workstations capable of drawing lines only, while computations are performed on a mainframe.

Adding interactiveness to the grid generation process, in combination with the new type of workstations most

likely will be the way to reduce the grid preparation time for complex configurations from months to days.

REFERENCES

[1] Andrews,A.E.,1988 : Knowledge-Based Flow Field Zoning ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp. 13-22.

[2] Sorenson,R.L.,1988 : Three-Dimensional Zonal Grids About Arbitrary Shapes by Poisson's Equation ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.75-84.

[3] Clark,G.L.,Ankeny,L.A.,1988 : Grid Generation Software Engineering at Los Alamos ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.95-104.

[4] Gentzsch,W.,Häuser,J.,1988 : Mesh Generation on Parallel Computers ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.113-124.

[5] Eyler,L.L.,White,M.D.,1988 : Surface Constrained Grid Generation with Langrange Multipliers ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.125-136.

[6] Hilgenstock,A.,1988 : A Fast Method for the Elliptic Generation of Three Dimensional Grids with Full Boundary Control ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.137-146.

[7] Hoffman,K.A.,Rutledge,W.H.,Rodi,P.E.,1988 : Hyperbolic Grid Generation Techniques for Blunt Body Configurations ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.147-156.

[8] Jones,G.A.,Thompson,J.F.,Warsi,Z.U.A.,1988 : Surface Grid Generation for Composite Block Grids ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.167-176.

[9] Hoffman,K.A.,Chiang,T-L,Bertin,J.J.,1988 : Effect of the Grid System on the Solution of Euler Equations ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.185-194.

[10] Wang,Y,Eiseman,P.R.,1988 : Patch Structured Surface Grid With Dynamic Curvature Clustering ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.195-206.

[11] Warsi,Z.U.A.,Tiarn,W.N.,1988 : Surface Grid Generation Through Elliptic PDE's ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.207-216.

[12] Zhu,J.,Rodi,W.,Schoenung,B.,1988 : Algebraic Generation of Smooth Grids ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.217-226.

[13] Djomehri,M.J.,Deiwert,G.S.,1988 : Three-Dimensional Self-Adaptive Grid Method for Complex Flows ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.277-288.

[14] Dannenhoffer III,J.F.,1988 : A Comparison of Two Adaptive Grid Techniques ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.319-328.

[15] Catherall,D.,1988 : Solution-Adaptive Grids for Transonic Flows ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.329-350.

[16] Arina,R.,1988 : Adaptive Orthogonal Surface Coordinates ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.351-360.

[17] Shaw,J.A.,Georgala,J.M.,Weatherill,N.P.,1988 : The Construction of Component-Adaptive Grids for Aerodynamic Geometries ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.383-394.

[18] Nielsen,P.,Skovaard,O.,1988 : A Depth Adaptive Grid Using a Control-Function Approach ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.435-442.

[19] Klopher,G.H.,1988 : Solution Adaptive Meshes with A Hyperbolic Grid Generator ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.443-454.

[20] Gu,C-Y.,Fuchs,L.,1988 : Zonal Grid Applications to Computations of Transonic Flows ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.465-484.

[21] Holcomb,J.E.,1988 : Requirements For The Adaptive Grid Navier-Stokes Analysis of Complex 3-D Configurations and Flowfields ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.495-504.

[22] Seibert,W.,1988 : A Graphic-Iterative Program-System to Generate Composite Grids for General Configurations ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.517-528.

[23] Miki,K.,Tago,K.,1988 : Three-Dimensional Composite Grid Generation by Domain Decomposition and Overlapping Technique ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.549-558.

[24] Allwright,S.E.,1988 : Techniques in Multiblock Domain Decomposition and Surface Grid Generation ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.559-568.

[25] Amdahl,D.J.,1988 : Interactive Multi-Block Grid Generation ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.579 588.

[26] Kennon,S.R.,Anderson,D.A.,1988 : Unstructured Grid Adaptation for Non-Convex Domains ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.599-610.

[27] Buratynski,E.K.,1988 : A Three-Dimensional Unstructured Mesh Generator for Arbitrary Internal Boundaries ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.621-632.

[28] Holmes,D.G.,Snyder,D.D.,1988 : The Generation of Unstructured Triangular Meshes Using Delaunay Triangulation ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.643-652.

[29] Baker,T.J.,1988 : Generation of Tetrahedral Meshes Around Complete Aircraft ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.675-686.

[30] Löhner,R.,Parikh,P.,Gumbert,C.,1988 : Interactive Generation of Unstructured Grids for Three Dimensional Problems ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.687-698.

[31] Baum,J.D.,Löhner,R.,1988 : Numerical Simulation of Shock-Box Interaction Using An Adaptive Shock Capturing Scheme ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.699-708.

[32] Sonar,T.,Radespiel,R.,1988 : Geometric Modelling of Complex Aerodynamic Surfaces and Three-Dimensional Grid Generation ;in : Numerical Grid Generation in Computational Fluid Mechanics '88 Pineridge Press,pp.795-804.

[33] Abolhassani,J.S.,Smith,R.E.,1988 : Multiple-Block Grid Adaption for an Airplane Geometry ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.815-824.

[34] Soni,B.K.,1988 : GENIE: Generation of Computational Geometry-Grids for Internal-External Flow Configurations ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.915-924.

[35] Jacobs,J.M.J.W.,Kassies,A.,Boerstoel,J.W.,Buijsen,F.,1988 : Numerical Interactive Grid Generation for 3D-Flow Calculation ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.925-944.

[36] Bernard,R.S.,1988 : Grid-Induced Computational Flow Separation ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.955-964.

[37] Giloi,W.K.,1988: Suprenum a trendsetter in modern supercomputer development; in Parallel Computing 7,pp.283-296.

[38] Baker,T.J.,1989 : private communication.

[39] Thacker,W.C.,1977: Irregular Grid Finite Difference Techniques; Simulations of Oscillations in Shallow Circular Basins, J.Phys.Ocean 7, pp.282-292.

[40] Coleman,R.,1988: Adaptive techniques for Boundary Grid Generation ;in : Numerical Grid Generation in Computational Fluid Mechanics '88, Pineridge Press,pp.339-350.

[41] Häuser,J. et al,1989: GRID-3D : A General 3-D Multi-Block Grid Generator, to be published.

[42] Holst,T.,1987: Numerical Solution of the Navier Stokes Equations about 3-D Configuration. A Survey ; Supercomputing in Aerospace, pp281-293 (Ed.P.Kutler), NASA Ames Symposium

[43] Häuser,J.;Taylor,C,1986 : Numerical Grid Generation in Fluid Dynamics, Pineridge Press

GENERAL STRUCTURED GRID GENERATION SYSTEMS

by
Joe F. Thompson
Professor
Mississippi State University
Department of Aerospace Engineering
Drawer A
Mississippi State, MS 39762
USA

ABSTRACT

Current techniques for the generation of composite-block structured grids for general 3D configurations are discussed. The various aspects of grid generation involved are noted, and their incorporation in general codes is cited. Current techniques for adaptive grids for general configurations are also discussed.

1. INTRODUCTION

Grid generation has progressed to the stage now of large general code development. The basic mathematical techniques involved have, to a large degree, been settled on and developed to an adequate level for common use. This is not to say that no further theoretical developments are needed, and advances continue to be reported in the literature, but major effort is now focusing on implementation in comprehensive codes. The principal concern is, in fact, now the development of automated procedures and codes with effective user interfaces.

The major emphasis now is on the treatment of very complex 3D configurations and on dynamically-adaptive grids coupled with solution codes. In regard to the former, although general codes are now available, it can still take a man-month to generate a grid for a complex new configuration. Although more computer time is typically required for a flow (or other PDE) solution, more man-time is generally spent on the grid. This is particularly a problem as flow (PDE) codes are becoming available that can be run effectively by design engineers, while the effective use of the grid codes still requires considerable expertise and experience. Adaptive grid techniques are not as well settled, and a number of approaches are still being considered. Some trends are emerging, however, and the utility of grid adaptation is clear.

This paper delineates the essential grid generation techniques that are incorporated in general codes, takes note of some such codes, and comments on some of the more promising adaptive approaches. This is not, however, a current survey of grid generation, and no effort is made to cover all the literature on the subject or to cite all works in the area. In particular, no conscious attention is given to new generation techniques as such.

2. GRID TYPES

The three basic grid approaches are a rectangular or Cartesian-like grid, a structured curvilinear body-conforming grid, and an unstructured triangularized grid. Each grid type has advantages and disadvantages. The **rectangular** grid is well-ordered, trivial to generate, readily allows accurate interior difference approximations, and the representation of a difference approximation requires the minimum work per step. However, boundary representation requires special logic, is generally of poor accuracy, and the grid does not cluster to efficiently resolve viscous boundary layers on curved boundaries. The **curvilinear** body-conforming grid is also well-ordered, allows higher-order difference approximations, permits simple and accurate boundary difference approximations, and can be clustered into gradient regions. It is especially well suited for viscous boundary layer approximation. However, the governing equations are more complex to difference on a curvilinear grid (although body-conforming grids often permit use of additional approximations), and grid generation, while not difficult for simple bodies, is no longer trivial. The **unstructured** triangularized mesh has good grid concentration (i.e., triangles can be readily deleted in smooth gradient regions) and the shape of the boundary curve is readily conformed to. However, such a mesh is poorly ordered and is therefore less amenable to the use of certain algorithms (e.g. ADI) and vectorized computers. The unstructured mesh requires less ingenuity to devise (though not necessarily to code) for complicated regions than does the structured mesh, but requires considerably more computer time and storage, as well as a much more involved data handling procedure. Moreover, unstructured meshes have not been used to a significant degree for resolving high Reynolds number viscous boundary layers of practical interest. Combinations of structured and unstructured meshes can also be used, with structured meshes near the boundaries connected by unstructured meshes.

For a simple body shape, the use of a single body-conforming curvilinear mesh leads to the most efficient solution procedure. As a result most current aerodynamics solution codes employ a body-conforming structured, curvilinear grid. Considerable effort is now underway to extend these procedures for complex three-dimensional configurations, generally by using composite grid techniques.

3. COMPOSITE GRIDS

Terminology

The use of composite grids has been the key to the treatment of general 3D configurations with structured grids. Here in general, "composite" refers to the fact that the physical region is divided into subregions (domain decomposition), within each of which a structured grid is generated. These sub-grids may be patched together at common interfaces, may be overlaid, or may be connected by an unstructured grid. Considerable confusion has arisen in regard to terminology for composite grids, making it difficult to immediately classify papers on the subject.

Composite grids in which the sub-grids share common interfaces are referred to as "block", "patched", "embedded", or "zonal" grids in the literature. The use of the first two of these terms is fairly consistent with this type of grid ("patched" comes from the common interfaces, "block" from the logically-rectangular structure), but the last two are sometimes applied to overlaid grids. Overlaid (overset) grids are called "chimera" grids after the composite monster of Greek mythology, but may also be said to be "overlapped". Unfortunately, the common interface grids can also be said to overlap since they typically use surrounding layers of points to achieve continuity. Embedded grids can be most anything, and the term is probably best avoided. The use of "zonal" comes mostly from CFD applications where the suggestion of applying different solution equation sets in different flow regions is made. Perhaps "block" or "patched" would be best for the common interface grids, "chimera" for the overlaid (avoiding "overlapped") grids, and "hybrid" for the structured-unstructured combinations.

Forms

With this terminology adopted, the **block** (or patched) grids may be completely continuous at the interfaces, have slope or line continuity, or be discontinuous (sharing a common interface but not common points thereon). ("Block" seems to cover all of these possibilities, but "patched" is being stretched a bit in the latter case.) Complete continuity is achieved through a surrounding layer of ("image", "phantom") points at which values are kept equal to those at corresponding "object" points inside an adjacent block. This requires a data indexing procedure to link the blocks across the interfaces. With complete continuity, the interface is not fixed (not even in shape), but is determined in the course of the solution. This type of interface necessitates an elliptic generation system. Slope continuity requires that the grid generation procedure incorporate some control over the intersection angle at boundaries (usually, but not necessarily, orthogonality), as can be done through Hermite interpolation in algebraic generation systems or through iterative adjustment of the control functions in elliptic systems. In this case the points on the interface are fixed, and the sub-grids are generated independently except for the use of the common interface points and a common (presumably orthogonal) angle of intersection with the interface. The CFD coding construction is greatly simplified with either complete or slope continuity, since no algorithm modifications are necessary at the interfaces.

The **chimera** (overlaid) grids are composed of completely independent component grids which may even overlap other component boundary elements, creating "holes" in the component grid. This requires flagging procedures to locate grid points that lie out of the field of computation, but such holes can be handled even in tridiagonal solvers by placing ones at the corresponding positions on the matrix diagonal and all zeros off the diagonal. These overlaid grids also require interpolation to transfer data between grids, and that subject is the principal focus of effort in regard to the use of this type of composite grid.

The **hybrid** structured-unstructured grids avoid this interpolation by replacing the overlaid region with an unstructured grid connecting logically-rectangular structured component grids. This can require modification of solution codes, however.

Codes

A number of codes, some quite general, based on the **block** structure have emerged, allowing CFD applications on full-aircraft and other general configurations. Some of these block codes are EAGLE (Refs. 1,2), 3DGRAPE (Refs. 3,4), NUGGET (Ref. 5), GRID-3D (Ref. 6), GRIDGEN (Refs. 7,8), GENiE (Ref. 9), and the codes of Refs. 10-25. The principal **overlaid** grid code is still that which coined the chimera name (Ref. 26), but this approach is also applied in Refs. 27 and 28. Some applications of **hybrid** grids have been made to general configurations, cf. Refs. 29,30.

Techniques

Algebraic grid generation today is generally based on transfinite interpolation, which provides a general mechanism for combining one-dimensional interpolation formulas into a 3D framework which matches all the boundaries of a region. With transfinite interpolation, a grid is generated in the interior of a region algebraically by interpolation from the entire (closed) boundary of the region, and perhaps from some interior specified or control surfaces. The basis of this multidimensional interpolation can be either Lagrange (linear) or Hermite (cubic) interpolation, the latter allowing boundary orthogonality. The blending functions which accomplish the interpolation can be linear, but a better choice is based on normalized arc-length distribution, itself interpolated from the boundaries by transfinite interpolation of one less dimensionality. This latter procedure allows boundary point distributions to be reflected throughout the field. Spline blending functions defined by specified point distributions are also used. Transfinite interpolation can produce boundary or grid-line overlapping, particularly with Hermite interpolation, for some configuration shapes, and interior surfaces (actual or control) are employed to prevent such overlap. Another device used to prevent grid overlap, and also to control the orientation and spacing of grid lines, is the division of blocks into sub-blocks for the purpose of the interpolation only, not affecting the block data structure. Values on the sub-block interior boundaries can either be specified for explicit control, or can be interpolated from the edges by transfinite interpolation of one less dimensionality. Grid overlap is a problem with algebraic grids, particularly when Hermite interpolation is used for boundary orthogonality. Overlap can also occur with elliptic grids, but often the elliptic system can unravel an overlapped algebraic grid.

Transfinite interpolation is the fundamental basis of the composite-block codes of Refs. 19-21, 24-25, and 9, and is also commonly used to generate an initial algebraic grid in codes based on elliptic generation, e.g. Ref. 1. The algebraic codes often use an elliptic system as a smoother, cf. Refs. 24,25, and 9. Additional control of the algebraic grid in the interior of the field can be exercised by including interior "support" surfaces in the interpolation as in Refs 24 and 25. Related to this is the use of spline blending functions employing interior points as in Ref. 31. Other algebraic approaches that have been used in composite-block grids include the curve-based system of Refs. 22 and 23, and interpolation from nodes, Ref. 32, following finite element technology.

Elliptic grid generation involves the solution of a set of partial differential equations for the grid, usually by point SOR iteration for general configurations. With point SOR iteration, locally-optimum acceleration parameters should be used for robustness, and directed (based on the sign of the control functions) one-sided differences should be used for the first derivatives. These equations involve control functions which control the orientation and spacing of the grid, and which can enforce boundary (or interface) orthogonality. These control functions can be evaluated automatically by transfinite interpolation from the boundaries of the region, or can be evaluated from an initial algebraic grid and then smoothed. In the latter case, the smoothing should be done only in the two directions other than that of the control function. With control functions determined from the algebraic grid in this manner, the elliptic system produces a grid that has the same general distribution as the algebraic grid but which is smoother. With the control functions evaluated by interpolation from the boundaries, the spacing and curvature parts of the control function should be evaluated on different boundaries and interpolated separately into the field. The spacing component is evaluated on the sides logically parallel to the direction of the control function, and interpolated by transfinite interpolation of one less dimensionality, while the curvature term is evaluated on the other two sides and is interpolated one-dimensionally. This approach allows the boundary point distribution to be reflected into the field. Boundary orthogonality is achieved by iterative adjustment of the control functions. All of these techniques for elliptic generation systems are discussed in detail in Ref. 33.

An elliptic generation system is the fundamental basis of the composite-block codes of Refs. 1-18 and 34-35. Of these, the code of Refs. 3 and 4 is based on slope continuity at block interfaces. The rest use a surrounding layer of points around each block to achieve complete continuity at the interfaces. Many of these codes incorporate iterative adjustment of control functions for boundary ortho- gonality, and also for use on interfaces in order to allow more control of the grid. Still more control is possible with the provision of applying this feature on interior surfaces within blocks as in Refs. 1,2 and 15,16. Various forms of interpolation of the boundary control functions into the field in the course of this iterative adjustment are in use. The original source of the procedure, Refs. 3,4, uses exponential interpolation, while Ref. 36 uses a power-law decaying interpolation with subsequent modifications of the control functions to improve smoothness and reduce skewness. The code of Refs. 1,2 does not interpolate the control functions from the boundary, but rather applies the orthogonality on each successive surface off the boundary to a decaying degree. In any case, it is important to have good control over the extent of the orthogonality from the boundary since different configurations and different applications, e.g. Euler, Navier-Stokes, call for different extent of orthogonality into the field. Ref. 35 uses the biharmonic equation, which admits an additional boundary condition that can be used for boundary ortho-gonality.

Surface Grids

The specification of the boundary point distribution is a two-dimensional grid problem in its own right, which can also be done either by transfinite interpolation or an elliptic generation system. In general, this is a 2D boundary value problem on a curved surface, i.e., the determination of the locations of points on the surface from specified distributions of points on the four edges of the surface. This is best approached through the use of **surface parametric coordinates** (cf. Refs. 37,38,1-2, and 14) whereby the surface is first defined by a 2D array of points, e.g. a set of cross-sections. The surface is then splined (bi-cubic), and the spline coordinates (surface parametric coordinates) are then made the dependent variables for the interpolation or the elliptic generation system. The generation of the surface grid can then be accomplished by first specifying the boundary points on the four edges of the surface grid, converting these Cartesian coordinate values to parametric coordinate values on the edges, then determining the interior values of the parametric coordinates from the edge values by the generation system, and finally converting these parametric values to Cartesian coordinates. The surface grid generation system can operate with the surface parametric coordinates taken as arc lengths along the two defining directions on the surface, but this has the disadvantage of requiring a search to locate the particular spline patch. The need for this search can be eliminated by taking the point indices on the surface definition as the surface parametric coordinates.

Point Distributions

The starting point of grid generation is the setting of a point distribution on a curve, and this can be done by splining the set of points defining the curve and then placing the desired number of points on the spline curve according to a relative arc length distribution. The point distribution on curves is probably best done in terms of normalized arc length using the **hyperbolic functions** that have been shown to reduce the truncation error induced by the unequally spaced points (Refs. 39,40. cf. also 1,2,41). These functions can also be employed as the blending functions in transfinite interpolation by interpolating normalized arc-length distributions from the boundaries (cf. Refs. 1,2). Another approach to point distribution is that based on **exponential functions**, with variable exponents or with exponents of exponents for additional control (cf. Refs. 42 and 3). These functions are used in Ref. 42 to control orthogonality and grid overlap as well.

Orthogonality

Coordinate systems that are orthogonal, or at least nearly orthogonal, near the boundary make the application of boundary conditions more straightforward. Although strict orthogonality is not necessary, the accuracy deteriorates if the departure from orthogonality is too large. The implementation of algebraic turbulence models is more reliable with near-orthogonality at the boundary, since information on local boundary normals is usually required in such models. The formulation of boundary-layer equations is also more straightforward and unambiguous in such systems. It is thus better in general, other considerations being equal, for grid lines to be nearly normal to boundaries.

Construction

The construction of a block grid normally begins with the specification of the four edges of a logically-rectangular surface patch. These edges are either defined directly as space curves or as parametric curves on a splined curved surface. The surface grid on the patch then is generated by either transfinite interpolation or from the elliptic system. In the case of generation on a curved surface, the interpolation or the elliptic solution is done in terms of the parametric coordinates, after which the Cartesian coordinates are recovered from the splined surface. The surface grid generation may itself be done in a block format, with complete continuity across the patch interfaces. When surface grids have been generated on all six sides of each block, the 3D grid in the block

is generated by transfinite interpolation or from the elliptic system, taking account of continuity connections across the interfaces.

The composite-block grid construction thus basically involves the generation of block edges, then faces, and finally volumes. The six faces bound the volume, and four edges bound a face. The **edge** generation is simply the placing of a point distribution on a curve, which may either be defined and then splined, or may be constructed geometrically. Definition may be by input cross-sections, plane-patch intersections (Ref. 35), by a curve generated parametrically on a specified spline curved surface (cf. Refs. 1,2), or by other means. With the four edges in place, the **face** can be generated as a defined surface or may be constructed in free form. With a defined surface, the generation can be done in terms of surface parametric coordinates on the splined defined surface by transfinite interpolation or by an elliptic system (Refs. 1,2,38). The use of surface parametric coordinates allows general surfaces to be treated. By contrast, the use of projection onto planes, or the use of a functional relationship for one Cartesian coordinate in terms of the other two, restricts the generality and requires the provision for rotations to put the surface in an orientation that can be treated. Therefore generation in terms of surface parametric coordinates is the preferred method of surface grid generation.

Surface definition is a significant problem in itself. Some approaches are the various forms of patching, e.g. Coon's patches (cf. Ref. 41), B-spline patches (Ref. 43), and transfinite interpolation (cf. Refs. 1,2,31). Assemblies of cross-sections are also used, either input directly (cf. Refs. 44,45) or formed as plane-patch intersections (Ref. 35). Considerable effort is now being directed at surface definition for grid generation directly from CAD constructions (cf. Ref. 8).

Surface intersections can be done by splining the intersected surface, and one family of lines on the intersecting surface, and then using a three-variable Newton iteration to determine the intersection curve (cf. Refs. 1,?,38, and 41).

The collapse of block faces into lines or even points greatly increases the generality of the composite-block structured grids, and such **degenerate faces** do not require any special treatment in finite volume flow codes. This feature is probably employed in most codes, and Refs. 1,2,4,11, and 13 are representative. This face collapse also allows axis singularities to be included. Ref. 46 introduces a bifurcation singularity for a branching pipe that allows the use of a cylindrical-type grid in the branched system.

User Interface

The man-time involved in grid generation is being greatly reduced by graphical user interfaces whereby the user operates the code and constructs the grid piece-by-piece on the workstation. To be really effective, however, the interface must build a file of input commands which can be subsequently edited or submitted again to reproduce the grid. Otherwise, the user must start again from the beginning, and although the grid is saved, the construction process is not. It is really the construction process that must be saved, since the process can be quite lengthy and will undoubtedly be useful again when modified or when used as the basis for yet another problem.

The interface must have some mechanism for identifying important points, curves, surfaces and spacings, and some means of incorporating these in the saved construction process that will allow changes in the number of points or spacing on a segment, or the position of a segment, to be automatically propagated throughout the construction process.

A complete grid package will contain a front-end boundary code to prepare the boundary segments for the grid code. This front-end code may be an interface from a CAD/CAM system, or may be a curve and surface construction mechanism in itself. The definition of boundary geometry for real configurations continues to be a significant problem. It should be noted that many solid modelers are meant to produce surfaces, not to reproduce existing surfaces with exactitude.

Considerable effort is now being directed at the development of efficient and effective **interactive interfaces** for grid generation systems, in order to reduce the still considerable man-time required for grid generation for complex configurations. Interfaces for surface construction are included in Refs. 8,43, and 47. Ref. 18 includes an interface from PATRAN, and Ref. 48 redefines a grid algebraically in terms of control points which can then be interactively moved to alter the grid. Ref. 49 notes a general interface system that has been attached to several general composite-block grid codes, in particular those of Refs. 1,2, and the 2D form of Ref. 5. A number of other user interfaces have also been reported, cf. Refs. 7,9,16,20-25, and 50, clearly indicative of the importance ascribed to interactive graphical interfaces to grid generation.

Of particular importance is the addition of **automatic topology generation** for the construction of the block structure. Refs. 10 and 11 use a graphically-interactive interface with topology defined by basic slits in the computational field corresponding to the various components of the physical boundary. A similar approach is taken in Ref. 13, but with the inclusion of a hypercube topology generation to form the composite-block structure. These systems allow various block topologies to be included by providing for block faces to be collapsed to lines or even points. Considerable interest is being expressed in the use of **artificial intelligence** to form the block structure, but only Ref. 51 reports any working system, and that is in 2D.

Parallel Processors

Interest is naturally developing in applications of grid generation on parallel processors since the composite-block structure forms a natural domain decomposition within which the various blocks can be assigned to different processors, or can be treated as different computing objects. Ref. 34 reports such an application on a shared memory system of an elliptic generation system using the surrounding layer of points for complete continuity.

4. ADAPTIVE GRID SCHEMES

Finally, dynamically-adaptive grids continually adapt to follow developing gradients in the physical solution. This adaption can reduce the oscillations associated with inadequate resolution of large gradients, allowing sharper shocks and better representation of boundary layers. Another advantageous feature is the fact that in the

viscous regions where real diffusion effects must not be swamped, the numerical dissipation from upwind biasing is reduced by the adaption. Dynamic adaption is at the frontier of numerical grid generation and may well prove to be one of its most important aspects, along with the treatment of real three-dimensional configurations through the composite grid structure.

Adaptive Strategies

There are three basic strategies that may be employed in dynamically adaptive grids (cf. Ref. 52) coupled with the partial differential equations of the physical problem. Combinations are also possible, of course:

(1) Redistribution of a fixed number of points.

In this approach, points are moved from regions of a relatively small error or solution gradient to regions of large error or gradient. As long as the redistribution of points does not seriously deplete the number of points in other regions of possible significant gradients, this is a viable approach. The increase in spacing that must occur somewhere is not of practical consequence if it occurs in regions of small error or gradient, even though in a formal mathematical sense the global approximation is not improved. The redistribution approach has the advantage of not increasing the computer time and storage during the solution, and of being straightforward in coding and data structure. The disadvantages are the possible deleterious depletion of points in certain regions, and the possibility of the grid becoming too skewed.

(2) Local refinement of a fixed set of points.

In this approach, points are added (or removed) locally in a fixed point structure in regions of relatively large error or solution gradient. Here there is, of course, no depletion of points in other regions and therefore no formal increase of error occurs. Since the error is locally reduced in the area of refinement, the global error does formally decrease. The practical advantage of this approach is that the original point structure is preserved. The disadvantages are that the computer time and storage increase with the refinement, and that the coding and data structure are difficult, especially for implicit flow solvers.

(3) Local increase in algorithm order.

In this approach, the solution method is changed locally to a higher-order approximation in regions of relatively large error or solution gradient without changing the point distribution. This again increases the formal global accuracy, since a local increase is achieved without an attendant decrease in formal accuracy elsewhere. The advantage is that the point distribution is not changed at all. The disadvantage is the great complexity of implementation in implicit flow solvers. This adaptive approach has not had any significant application in CFD in multiple dimensions.

Redistribution

Adaptive redistribution of points traces its roots to the principle of equidistribution of error by which a point distribution is set so as to make the product of the spacing and a weight function constant over all the points. A competitive enhancement of grid smoothness, orthogonality, and concentration can be accomplished by representing each of these features by integral measures over the grid, and minimizing a weighted average of the three. The one-dimensional form of this leads, in fact, to the equidistribution principle. A second approach is to note the correspondence between the equidistribution principle and the one-dimensional form of the commonly-used elliptic grid generation system. This leads to a connection between the control functions in the elliptic system and the derivatives of the weight function. This control function adaptive approach has the significant advantage of being based on the same elliptic generation equations that are in common use in grid generation codes, and the adaptive control functions can be added to those already evaluated from the configuration geometry.

Recent applications of the **variational form** of adaptation are given in Refs. 53-55, but none of these use partial differential equations. The last of these bases the formulation on principles of continuum mechanics.

The **control function form** of grid adaptation is used in Ref. 56 for a general composite-block structure. Other applications appear in Refs. 57-60. A block-structured adaptive grid is also given in Ref. 61, but with adaptation to follow streamlines.

One-dimensional adaptation, generally applied in **alternating directions** in multiple dimensions, is used in Refs. 62-64. Here both tension and torsion spring analogies are used to control both the grid concentration and skewness.

Refinement

The addition of points can be used with structured quadralateral grids, as well as with unstructured grids, and Refs. 65 and 66 are recent examples. The latter reference compared grid refinement and redistribution, finding little difference in the 2D transonic Euler solution considered.

Other Approaches

Adaptation is included in a hyperbolic generation system in Ref. 67, and in a parabolic system in Ref. 59. Finally, new approaches to grid adaptation are given in Refs. 68 and 69, the former based on harmonic maps and the latter on parametric mapping on a surface.

5. CONCLUSION

Among the advantages to be cited for the composite grid approach are the following:

(1) ease of treatment of complex configurations.
(2) capability for local refinement and modification.
(3) reduced core storage.
(4) natural use of different flow equations in different regions.
(5) grid singularities can be placed on block boundaries.

A second point is that because of the emphasis on composite grids, the tasks of subdividing the grids, generating surface grids, and providing interfaces have become more time consuming and critical than the task of generating the interior ' rids. The papers on composite grids in Refs. 70 and 71 either strongly hint at, or explicitly note, that how a grid `' ɔuld be subdivided depends on the geometry, the numerical algorithm used, the flow features, etc. So, given a limited computer resource, the sub-grids of a composite grid must be selected with care. This implies a learning process and a need for human interaction. Like geometry definition, the tasks of subgriding, interfacing, and surface grid definition are being assigned to interactive workstations. Various levels of sophistication in treating these problems in this way are evident in the papers. What is strongly implied is that these are not simple tasks or ones for which off-the-shelf software is available. This is evidently a pacing area of research in complex grid generation.

Surface grid generation is seen to have a dominant effect on the quality of the volume grid, to be very time-consuming, and to be in considerable need of improvement in regard to the specification of boundary data sets and the interactive manipulation thereof. Surface definition continues to be a pacing problem. There is a feeling that more emphasis should be put on the development of CAD geometry tools especially suited to the needs of CFD.

The topological definition of the block structure is seen to require considerable experience and to be difficult to teach. There is need for automation of this process, perhaps through the use of artificial intelligence or other means.

The critical need for graphical interaction, especially in regard to surface grid generation, block definition, and grid control is evident. Codes should have an efficient and effective user interface with error-checking and on-line instruction. The process of grid generation for complex configurations still requires too large an amount of man-time.

6. REFERENCES

1. Thompson, J.F., "A Composite Grid Generation Code for General 3D Regions--the EAGLE Code", AIAA Journal, Vol. 26, No. 3, p. 271, March 1988.

2. Thompson, J.F. and Lijewski, L.E., "Composite Grid Generation for Aircraft Configurations with the EAGLE Code", p. 85 of Ref. 70.

3. Sorenson, R.L., "Three-Dimensional Zonal Grids About Arbitrary Shapes by Poisson's Equation", p. 75 of Ref. 71.

4. Sorenson, R.L., "Three-Dimensional Elliptic Grid Generation for an F-16", p. 23 of Ref. 70.

5. Coleman, R.M., "NUGGET: A Program for Three-Dimensional Grid Generation", DTRC Report DTNSRDC-87/036, Sept. 1987.

6. Miki, K. and Tago, K., "Three-Dimensional Composite Grid Generation by Domain Decomposition and Overlapping Technique", p. 549 of Ref. 71.

7. Amdahl, D.J., "Interactive Multi-Block Generation", p. 579 of Ref. 71.

8. Steinbrenner, J.P., Karman, S.L., Jr., and Chawner, J.R., "Generation of Multiple Block Grids for Arbitrary 3D Geometries", p. 40 of Ref. 70.

9. Soni, B.K., "GENIE: Generation of Computational Geometry-Grids for Internal-External Flow Configurations", p. 915 of Ref. 71.

10. Shaw, J.A., Georgala, J.M., and Weatherill, N.P., "The Construction of Component-Adaptive Grids for Aerodynamic Geometries", p. 383 of Ref. 71.

11. Weatherill, N.P. and Shaw, J.A., "Component Adaptive Grid Generation for Aircraft Configurations", p. 29 of Ref. 70.

12. Holcomb, J.E., "Development of a Grid Generator to Support 3-D Multizone Navier-Stokes Analysis", AIAA-87-0203, AIAA 25th Aerospace Sciences Meeting, Reno, 1987.

13. Allwright, S.E., "Techniques in Multiblock Domain Decomposition and Surface Grid Generation", p. 559 of Ref. 71.

14. Woan, C.J., "Three-Dimensional Elliptic Grid Generations Using a Multi-Block Method", AIAA-87-0278, AIAA 25th Aerospace Sciences Meeting, Reno, 1987.

15. Sonar, T. and Radespiel, R., "Geometric Modelling of Complex Aerodynamic Surfaces and Three-Dimensional Grid Generation", p. 795 of Ref. 71.

16. Radespiel, R., "Grid Generation Around Transport Aircraft Configurations Using A Multi-Block Structured Computational Domain", p. 139 of Ref. 70.

17. Deng, G.B., Lecointe, Y., Piquet, J., and Visonneau, M., "Multiblock Grid Generation for Afterbody Problems", p. 529 of Ref. 71.

18. Jacobs, J.M.J.W., Kassies, A., Boerstoel, J.W., Buijsen, F., "Numerical Interactive Grid Generation for 3D-Flow Calculations", p. 925 of Ref. 71.

19. Abolhassani, J.S. and Smith, R.E., "Three-Dimensional Grid Generation About a Submarine", p. 505 of Ref. 71.

20. Smith, R.E. and Everton, E.L., "Interactive Grid Generation for Fighter Aircraft Geometries", p. 805 of Ref. 71.

21. Abolhassani, J.S. and Smith, R.E., "Multiple-Block Adaption for an Airplane Geometry", p. 815 of Ref. 71.

22. Seibert, W., "A Graphic-Iterative Program-System to Generate Composite Grids for General Configurations", p. 517 of Ref. 71.

23. Fritz, W., Haase, W., and Seibert, W., "Mesh Generation for Industrial Application of Euler and Navier Stokes Solvers", p. 106 of Ref. 70.

24. Ozell, B. and Camarero, R., "CAGD in Turbomachinery", p. 865 of Ref. 71.

25. Lauze, Y., Camarero, R., and Pelletier, D., "Interactive Design of 3-D Grids for Propellers", p. 875 of Ref. 71.

26. Benek, J.A., Donegan, T.L., and Suhs, N.E., "Experience with Three-Dimensional Composite Grids", p. 124 of Ref. 70.

27. Gu, C.Y. and Fuchs, L., "Zonal Grid Applications to Computations of Transonic Flows", p. 465 of Ref. 71.

28. Mastin, C.W., "Fast Interpolation Schemes for Moving Grids", p. 63 of Ref. 71.

29. Weatherill, N.P., "On the Combination of Structured-Unstructured Meshes", p. 729 of Ref. 71.

30. Nakahashi, K. and Obayashi, S., "FDM-FEM Zonal Approach for Viscous Flow Computations Over Multiple-Bodies", AIAA-87-0604, AIAA 25th Aerospace Sciences Meeting, Reno, 1987.

31. Zhu, J., Rodi, W., and Schoenung, B., "Algebraic Generation of Smooth Grids", p. 217 of Ref. 71.

32. Ecer, A., Spyropoulos, J.T., and Bulbul, E., "Application of a Three-Dimensional Finite Element Grid Generation Scheme for an F-16 Aircraft Configuration", p. 741 of Ref. 71.

33. Thompson, J.F., "A General Three-Dimensional Elliptic Grid Generation System on a Composite Block Structure", Computer Methods in Applied Mechanics and Engineering, Vol. 64, p. 377, 1987.

34. Gentzsch, W. and Hauser, J., "Mesh Generation on Parallel Computers", p. 113 of Ref. 71.

35. Eberle, A. and Schwarz, W., "Grid Generation for an Advanced Fighter Aircraft", p. 65 of Ref. 70.

36. Chen, S.C., "Three-Dimensional Adaptive Grid Generation for Body-Fitted Coordinate System", p. 309 of Ref. 71.

37. Warsi, Z.U.A. and Tiarn, W.N., "Surface Grid Generation Through Elliptic PDE's", p. 207 of Ref. 71.

38. Jones, G.A., Thompson, J.F., Warsi, Z.U.A., "Surface Grid Generation for Composite Block Grids", p. 167 of Ref. 71.

39. Vinokur, Marcel, "On One-Dimensional Stretching Functions for Finite-Difference Calculations", Journal of Computational Physics, 50, 215, 1983.

40. Thompson, J.F. and Mastin, C.W., "Order of Difference Expressions in Curvilinear Coordinate Systems", J. Fluids Eng., 107, 241, 1985.

41. Luh, R.C.C., "Surface Grid Generation for Complex Three-Dimensional Geometries", p. 85 of Ref. 71.

42. Moitra, A., "Quasi-Three-Dimensional Grid Generation by an Algebraic Homotopy Procedure", p. 41 of Ref. 71.

43. Coleman, R.M., "Adaptive Techniques for Boundary Grid Generation", p. 339 of Ref. 71.

44. Sobieczky, H., "Analytical Surfaces and Grids", p. 96 of Ref. 70.

45. Melton, J.E. and Langhi, R.G., "Surface Grid Generation for Advanced Transport Configurations", p. 751 of Ref. 71.

46. MazHer, A.H., "An Algebraic Procedure to Generate 3D Grids for Complex Arterial Flow Geometries", p. 51 of Ref. 71.

47. Klunover, A., Kao, T.J., and Yu, N.J., "Application of Multiblock Grid Generation Approach to Aircraft Configurations", p. 569 of Ref. 71.

48. Choo, Y.K., Eiseman, P.R., and Reno, C., " Interactive Grid Generation for Turbomachinery Flow Field Simulations", p. 895 of Ref. 71.

49. Clark, G.L. and Ankeny, L.A., "Grid Generation Software Engineering at Los Alamos", p. 95 of Ref. 71.

50. Rout, R.K., "Application of I-DEAS Grid Generator for Three-Dimensional Transonic Flow Analysis", p. 761 of Ref. 71.

51. Andrews, A.E., "Knowledge-Based Flow Field Zoning", p. 13 of Ref. 71.

52. Thompson, J.F., "A Survey of Dynamically-Adaptive Grids in the Numerical Solution of Partial Differential Equations", Applied Numerical Mathematics, Vol. 1, p. 3, 1985.

53. Kumar, A. and Kumar, N.S., "A New Approach to Grid Generation Based on Local Optimization", p. 177 of Ref. 71.

54. Castillo, J.E., "A Direct Variational Grid Generation Method: Orthogonality Control", p. 247 of Ref. 71.

55. Jacquotte, O.P. and Cabello, J., "A Variational Method for the Optimization and Adaptation of Grids in Computational Fluid Dynamics", p. 405 of Ref. 71.

56. Kim, H.J., and Thompson, J.F., "Three Dimensional Adaptive Grid Generation on a Composite Block Grid", AIAA-88-0311, AIAA 26th Aerospace Sciences Meeting, Reno, 1988, to appear in AIAA Journal 1989.

57. Hsu, C.C. and Yang, S.C., "On An Adaptive Grid Generation Technique for Transonic Turbulent Projectile Aerodynamics Computation", p. 415 of Ref. 71.

58. Nielsen, P. and Skovgaard O., "A Depth-Adaptive Grid Using a Control-Function Approach", p. 435 of Ref. 71.

59. Parpia, I.H. and Noack, R.W., "Solution Adaptive Parabolic Grid Generation in Two and Three Dimensions", p. 475 of Ref. 71.

60. Steinbrenner, J.P. and Anderson, D.A., "Three-Dimensional Parametric Block Grid Regeneration with Localized Solution Adaption", p. 539 of Ref. 71.

61. Holcomb, J.E., "Requirements for the Adaptive Grid Navier-Stokes Analysis of Complex 3-D Configurations and Flowfields", p. 495 of Ref. 71.

62. Bockelie, M. and Eiseman, P.R., "Grid Adaptivity with Evolutionary Control", p. 257 of Ref. 71.

63. Djomehri, M.J. and Deiwert, G.S., "Three-Dimensional Self-Adaptive Grid Method for Complex Flows", p. 277 of Ref. 71.

64. Catherall, D., "Solution-Adaptive Grids for Transonic Flows", p. 329 of Ref. 71.

65. Thomas, J.W. and McKay, S.M., "Generation of FAC Patched Grids", p. 1 of Ref. 71.

66. Dannenhoffer, J.F., III, "A Comparison of Two Adaptive Grid Techniques", p. 319 of Ref. 71.

67. Klopfer, G.H., "Solution Adaptive Meshes with A Hyperbolic Grid Generator", p. 443 of Ref. 71.

68. Dvinsky, A.S., "Adaptive Grid Generation from Harmonic Maps", p. 299 of Ref. 71.

69. Lee, K.D., Loellback, J.M., and Pierce, T.R., "Solution-Adaptive Grid Generation Using a Parametric Mapping", p. 455 of Ref. 71.

70. Thompson, J.F. and Steger, J.F., (eds.) Three Dimensional Grid Generation for Complex Configurations-- Recent Progress, AGARD-AG-309, 1988.

71. Sengupta, S., et. al. (eds.) Numerical Grid Generation in Computational Fluid Dynamics 1988, (Proceedings of the Second International Conference), Pineridge Press, 1988.

ALGEBRAIC BLOCK-STRUCTURED GRID GENERATION
BASED ON A MACRO-BLOCK CONCEPT

by

Lars-Erik Eriksson
Volvo Flygmotor AB
S-461 81 Trollhättan, Sweden

and

Erland Ørbekk
Division of Hydro & Gas Dynamics
NTH-The Norwegian Institute of Technology
N-7034 Trondheim, Norway

ABSTRACT

An algebraic technique for generating block-structured grids of arbitrary topology is des-cribed. The method is based on a macro-block concept which allows the usage of large blocks with partial block boundary interfacing. Various spline procedures are used for curve gene-ration and for constrained surface grids whereas all unconstrained surfaces and volumes are gridded by transfinite interpolation. Instead of using derivative information to control the grid in each block the present method is based on the idea of generating as many inte-rior guiding surfaces as required for grid control and gridding each resulting sub-block independently of other sub-blocks. The resulting metric discontinuities, both inside blocks and between blocks, are smoothed either by a local algebraic smoothing procedure or by a global elliptic type smoothing procedure or combinations of both. The complete method has been coded in a highly modular way and includes all graphics routines. Several 3-D multi-block grid examples are presented and discussed.

INTRODUCTION

In recent years a number of general purpose computer codes for the generation of 3D multi-block structured grids with arbitrary topologies have been presented in the literature[1,2,3,4]. Although much work evidently has been spent on making these codes as easy to use as possi-ble, the problem of automating various features such as the block-structuring process for arbritrary geometries is still unsolved. Since a satisfactory solution to these automation problems seems to be several years ahead in time, it is fair to say that the existing grid codes are really tools with which the experienced user generates whatever grid he can envi-sion and is able to create. In other words, the user's experience and skills with any given grid generation tool are just as important as the actual tool. The tools may differ in terms of how quickly the average user learns to use them satisfactorily or in terms of what the experienced user can accomplish with them, but in all cases the end result depends to a large extent on the user's skills, motivation, experience, knowledge of the overall problem, etc. It seems unlikely that this situation will change significantly in the near future.

The purpose of this paper is to present the authors' own version of a 3D multi-block grid code (G3DMESH) and to demonstrate some of its capabilities. It is similar to other such codes in that it is essentially a grid generation tool. In fact, it is more like a toolbox with all the tools necessary for defining block structure, generating grid curves, grid surfaces, grid volumes, exercise grid smoothing, metric checking, graphics, block redefinition, disc storage, etc. Although some features of the code are based directly on the first author's previous work in this area[5,6,7], many new features are also included. In the following para-graphs we first outline the overall method, then give some grid examples, and finally some concluding remarks.

OUTLINE OF METHOD

Block structure

The present multi-block grid generation scheme is based on a "macro-block" concept in which block sizes are not constrained by block interfacing. This feature is achieved by allowing "partial boundary interfacing", i.e. any part of one block boundary may be connected to any compatible part of another block boundary. In most applications this flexibility allows the user to define much fewer and larger blocks than would be the case with "complete boundary interfacing". An illustration of this effect is given by a 2D example, a C-type grid around two airfoils (Fig 1). Here it is seen that with complete boundary interfacing a minimum of 16 blocks is required whereas only 3 blocks are needed when partial boundary interfacing is allowed. This difference is even more pronounced in 3D applications.

The advantages of the macro-block approach are obvious, not only for the grid generation phase itself but also for the equation solution phase. There are fewer blocks to keep track of, a smaller number of interfaces to be defined, less boundary overhead and more efficient vectorization in the equation solver. A potential disadvantage of the macro-block approach is the fact that partial boundary interfacing is slightly more complex than complete boun-

dary interfacing. However, in the present method a very efficient partial boundary inter-
face scheme has been devised, both for the grid generation code and for the equation sol-
ver, which effectively solves this problem. Each interface is defined by two sets of data,
one for each side of the interface. These data sets consist of eight integers; four inte-
gers to define a reference corner point, two integers to define two directions and two
integers to define two dimensions (Fig 2). In special interface routines this data is used
to set up a complete mapping between the reference grid system and a local interface grid
system which spans across the interface in question. Any operation which needs to be car-
ried across the interface is easily done so using this mapping.

Grid generation

The present grid generation procedure is based on the idea of "piece by piece" building of
the desired grid blocks[6,7]. That is, each grid block may be built up in several steps where
each step constitutes a specific operation such as curve generation, surface generation,
volume generation, etc. The tools which perform these operations are completely general
and may be applied to any grid block or part of a grid block. In other words, grid curves
and grid surfaces can be defined in the interior of a block as well as on its boundaries.
This feature is very useful since it allows the user to introduce any desired grid control
in each block.

Constrained grid curves are here generated using cubic spline interpolation whereas uncon-
strained curves are usually generated by two-point splines with optional direction control
at the end points. As mentioned above a grid curve can be defined in any desired location
in any desired block.

Constrained surface grids can either be defined using bicubic spline interpolation or else
read from external files. The latter option is often used for more complex geometries where
intersections between various independently defined surfaces must first be determined. In
the present version of G3DMESH there are no such CAD/CAM tools included. Unconstrained sur-
face grids are generated either as multiple curves or by transfinite interpolation, depen-
ding on whether two or four boundary curves are defined. In the latter case arc-length
weighted blending functions[7] are used for optimal results. No derivative information is
used in the present version, i.e. only the bounding curves are used in the transfinite
interpolation procedure.

Volume grids (fully defined blocks or sub-blocks) are here generated by transfinite inter-
polation, again using arc-length weighted blending functions. Two different options are
implemented, one for the case when only four bounding surfaces are defined and one for the
case when all six bounding surfaces are defined. As in the surface generation case no deri-
vative information is used in the transfinite interpolation procedure, only the grid points
of the bounding surfaces. At first sight this might be seen as a severe limitation since
the use of derivative data is a very effective way of controlling the grid[5]. However, the
possibility of generating interior grid surfaces in any desired block and applying the vol-
ume grid generator in between these guiding surfaces makes the present approach just as
powerful as any method using derivative data.

Grid smoothing

Due to the "piece by piece" building of the complete grid system there will in general be
several metric discontinuities, both inside blocks and between blocks. If the psysical
boundaries have metric discontinuities such as edges or corners these will also spread
into the domain. This situation is usually unacceptable, at least for finite-difference or
finite-volume equation solvers, and some smoothing has to be applied to these metric dis-
continuities. In the present method there are two alternative "tools" for grid smoothing;
a local algebraic smoother and a global elliptic type smoother. The local smoother is a
one-step projection type procedure based on transfinite interpolation with cubic blending
functions in the direction normal to the discontinuity[7]. It is a computationally efficient
smoother which is easy to apply for "concentrated" metric discontinuities, i.e. disconti-
nuities that are confined to certain grid surfaces or block boundaries. A disadvantage
with this smoother is that the user has to specify the region where it is to be activated,
but the great advantage with it is the fact that it is entirely local and thus does not
alter anything outside the active region.

The other grid smoother is, as was mentioned above, a global elliptic type smoother. It is
based on the standard spatial operator used for elliptic grid generation[8]

$$D \vec{r} = \alpha \vec{r}_{\xi\xi} + \beta \vec{r}_{\xi\eta} + \gamma \vec{r}_{\xi\zeta} + \delta \vec{r}_{\eta\eta} + \epsilon \vec{r}_{\eta\zeta} + \phi \vec{r}_{\zeta\zeta} \qquad (1)$$

where

$$\vec{r} = (x, y, z)$$

$$\alpha = (DF - E^2) \quad ; \quad \beta = 2(EC - BF) \quad ; \quad \gamma = 2(BE - CD)$$

$$\delta = (AF - C^2) \quad ; \quad \epsilon = 2(BC - AE) \quad ; \quad \phi = (AD - B^2)$$

$$A = \vec{r}_\xi \cdot \vec{r}_\xi \quad ; \quad B = \vec{r}_\xi \cdot \vec{r}_\eta \quad ; \quad C = \vec{r}_\xi \cdot \vec{r}_\zeta$$

$$D = \vec{r}_\eta \cdot \vec{r}_\eta \quad ; \quad E = \vec{r}_\eta \cdot \vec{r}_\zeta \quad ; \quad F = \vec{r}_\zeta \cdot \vec{r}_\zeta$$

together with a Jacobi iteration scheme

$$\vec{r}^{(n+1)} = \vec{r}^{(n)} + \lambda (D \ \vec{r})^{(n)} \tag{2}$$

where the locally determined relaxation parameter is chosen such that high spatial frequencies are damped as quickly as possible. The purpose of this procedure is not to try to solve the nonlinear elliptic equations, only to take enough iteration steps to damp out metric discontinuities. In the present version no source terms are used which means that usually about 20 steps can be taken before undesirable changes in the grid begin to appear. However, in all cases tried so far this has been more than sufficient to eliminate all visible discontinuities. This smoother is simpler to apply than the local one (only the block interfacing data is needed) and has the additional advantage that it usually corrects any local grid inversions that may have been created in earlier stages. However, it is not as computationally efficient as the local smoother.

Additional features

There are of course many additional "tools" in the G3DMESH grid code. For example, there are checking routines which "measure" various quantities such as cell volumes, grid spacings, directions, etc. Several graphics routines are available, both for plotting curves and surfaces with any desired viewing angles. Special routines are also included with which the user can duplicate curves/surfaces and translate/rotate them in any desired manner. Block interfaces with periodicity conditions are also included so that grids for turbomachinery applications can be generated. One of the latest additions to the code is a "reblocking" option, i.e. a possibility of changing the block structure for a given grid system. Since most grid systems have non-unique block structure (at least in the macro-block case) it is often advantageous to use one block structure for the grid generation phase and another one for the equation solution phase. A 2D example which demonstrates this is shown in Fig 3; a C-type grid around an airfoil/flap combination. A five-block grid is the natural choice for generating the grid (with the present method, not necessarily with other grid codes) whereas a three-block grid is more natural for the flow solver (at least for the authors' flow solver).

Code structure

The grid code G3DMESH is built around a global data base containing the block structure and grid points. All arrays are one-dimensional and a pointer system is used to convert between block mode addressing (which is all the user needs to worry about) and sequential mode addressing of grid points. Each function or tool which the user can call upon consists of a subroutine which accepts certain input from the user and in return works upon the data base and performs whatever task it is designed for. This highly modular code structure makes it easy to include new or improved subroutines whenever the need arises without invalidating the old functions.

EXAMPLE GRIDS

The present grid code, G3DMESH, has been applied to a number of different cases and has turned out to be a very powerful and versatile grid generation tool. As a first demonstration we present the case of a simple wing-body combination (Fig 4). Here we have used a two-block grid which is of O-type around the fuselage and H-type around the wing. This grid system was built by generating four interior grid surfaces in each block in addition to the block boundary surfaces, a task which was easily and quickly done by using the curve generators and transfinite surface interpolators. The purpose of these interior surfaces was to control the grid around the wing in terms of spacing and orthogonality. A concentration at the leading edge, trailing edge and tip of the wing was desired and achieved through this technique. In this case only the local algebraic grid smoother was used, mainly to demonstrate that it is possible to achieve smooth grids with purely algebraic methods. However, the global elliptic smoother described in the previous section performs equally well, with less input data but with more CPU time. Typical execution times when running the complete session file from start to finish are about 30 CPU-seconds on a VAX 8700 with the algebraic smoother and about 5 CPU-minutes with the elliptic smoother.

Next we present two more "realistic" cases; the proposed HOTOL aerospace plane and an old version of the HERMES reentry vehicle (Fig 5). The same type of two-block grid structure as in the previous case was used for the HOTOL geometry whereas a single block grid was used for the HERMES geometry. Only the algebraic smoother was applied in these cases. Flow solutions (Euler computations) have been obtained on these grids, using our own multi-block time-marching Euler solver G3DEUL, and various aspects of the computed flow fields such as shocks and total pressure loss indicate that the presented grids are realistic in terms of relative spacings and orthogonality at surfaces. An item which may be of interest in this context, although not directly grid related, is the performance obtained with the Euler code on this type of macro-block grid. In several cases, involving single-block grids up to six-block grids and grid sizes from 50000 to 150000 points, the multi-block Runge-Kutta cell-centered finite-volume Euler code G3DEUL has achieved about $25 * 10^{-6}$ CPU-seconds / grid point / time step on a Cray X-MP computer (single processor mode). This figure is only about 30% higher than that for the corresponding old special purpose single-block code. We feel confident that this relatively small overhead is due to the macro-block technique and the efficient partial block boundary interfacing.

The next example is an isolated supersonic intake geometry designed for a free stream Mach number of 2.75 (Fig 6). The reason for choosing this case is two-fold: Firstly the flow in the near vicinity of and inside the intake is important in itself and a good grid in this area is always desirable. Secondly the interference between fuselage / wings and the intake is also important and a grid structure which allows the gridding of the complete fuselage / wing / intake geometry is thus desirable. The grid presented here was primarily designed for the first task but with the second task in mind, i.e. it is compatible with such a "total" grid. As may be seen in the grid plot the interior of the intake is gridded as one block with a wedge-type singularity at the beginning of the compression ramp. The exterior region is then decomposed into three blocks, giving an H-type grid. Euler computations verify that this grid structure is sound and gives an accurate representation of the geometry.

Finally, we present a more complex geometry; a schematic wing / body / air intake combination (Fig 7). Although not very realistic in terms of scale, this model has many of the features of a real world fighter aircraft. The supersonic intake with boundary layer diverter is here an important characteristic which the grid system must be able to cope with. We have here chosen a 12-block grid to model the complete geometry. One block covers the diverter region and a wedge-shaped region upstream of it, another block covers the interior of the intake (as in the previous case), five blocks cover the near region around the fuselage / wings / intake and finally another five blocks cover the remaining region out to the outer boundary. Only the global elliptic smoothing was used here and the intersection plots clearly show the effects of the smoothing between grid blocks.

CONCLUSIONS

The grid examples presented in this paper demonstrate that the algebraic method described above is a powerful grid generation tool. The combination of a user-controlled "piece-by-piece" building approach, transfinite interpolation and local/global grid smoothing procedures makes it possible to input, in a very direct and easily understood manner, any degree of grid control needed and to smooth out any metric discontinuities, inside blocks or between blocks. A highly modular code structure ensures that improved or new functions can be added to the code without invalidating previous functions. The generality of the data base used in the code also ensures that any new or improved future techniques can be easily implemented.

ACKNOWLEDGEMENT

REFERENCES

1. Thompson, J.F., "A Composite Grid Generation Code for General 3-D Regions", AIAA-87-0275, 1987.

2. Allwright, S.E., "Techniques in Multiblock Domain Decomposition and Surface Grid Generation", in: Numerical Grid Generation in Computational Fluid Mechanics '88, eds. S. Sengupta, J. Haüser, P.R. Eiseman, J.F. Thompson, Pineridge Press, 1988.

3. Shaw, J.A., Georgala, J.M., Weatherill, N.P., "The Construction of Component-Adaptive Grids for Aerodynamic Geometries", in: Numerical Grid Generation in Computational Fluid Mechanics '88, eds. S. Sengupta, J. Haüser, P.R. Eiseman, J.F. Thompson, Pineridge Press, 1988.

4. Seibert, W., "A Graphic-Interactive Program-System to Generate Composite Grids for General Configurations", in: Numerical Grid Generation in Computational Fluid Mechanics '88, eds. S. Sengupta, J. Haüser, P.R. Eiseman, J.F. Thompson, Pineridge Press, 1988.

5. Eriksson, L.-E., "Practical Three-Dimensional Mesh Generation Using Transfinite Interpolation", SIAM J. Sci. Stat. Comput., Vol. 6, No. 3, pp 712-741, July 1985.

6. Eriksson, L.-E., "Flow Solution On a Dual-Block Grid Around an Airplane", Computer Methods in Applied Mechanics and Engineering 64 (1987), pp 79-93.

7. Eriksson, L.-E., "A 2D General Purpose Grid Generator Based on Transfinite Interpolation", HOG Report 1987:103 (A), Div. of Hydro- & Gas Dynamics, NTH-The Norwegian Institute of Technology.

8. Thompson, J.F., Thames, F.C., Mastin, C.W., "Automatic Numerical Grid Generation of Body-Fitted Curvilinear Coordinate System for Field Containing Any Number of Arbitrary Two-Dimensional Bodies", J. Comp. Physics, Vol. 15, pp 299-319, 1974.
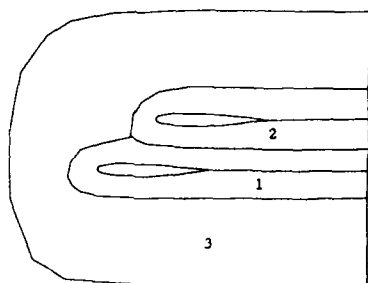
Fig 1a.

C-type grid around two airfoils
using the macro-block concept
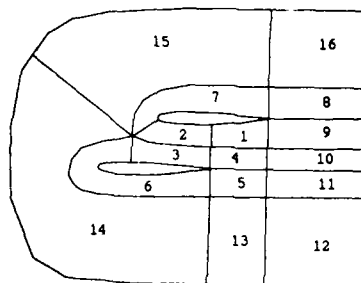(partial block boundary inter-
facing).

Fig 1b.

C-type grid around two airfoils
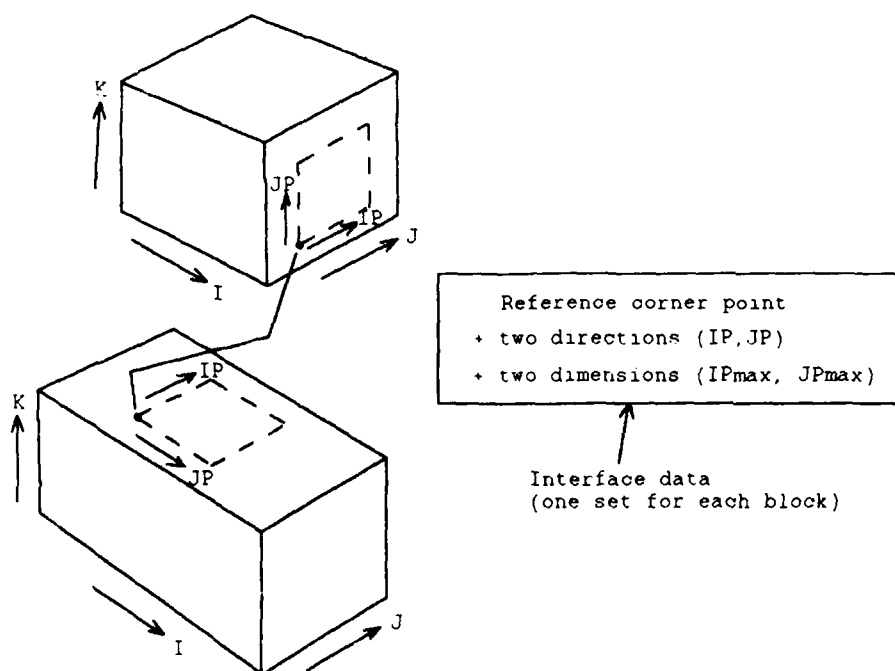using the micro-block concept
(complete block boundary inter-
facing).

Reference corner point

+ two directions (IP, JP)

+ two dimensions (IPmax, JPmax)

Interface data
(one set for each block)

Fig 2.

Interface data which defines a partial block boundary interface.
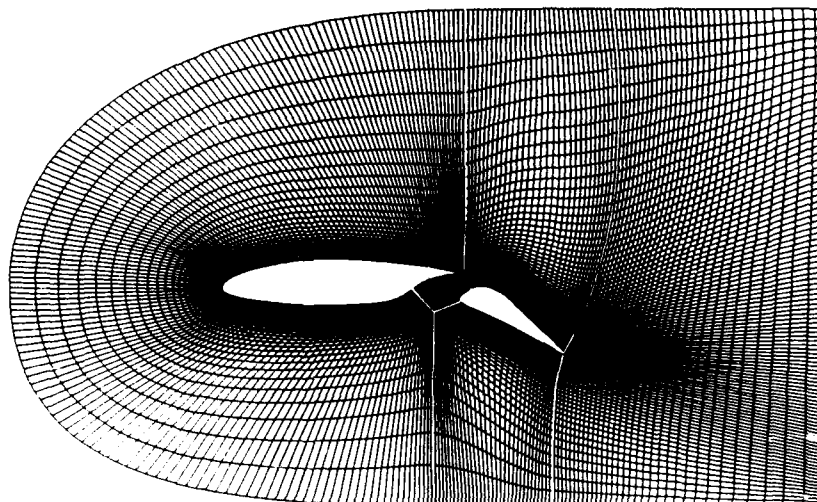
Fig 3a

C-type grid around airfoil/flap geometry Most natural block
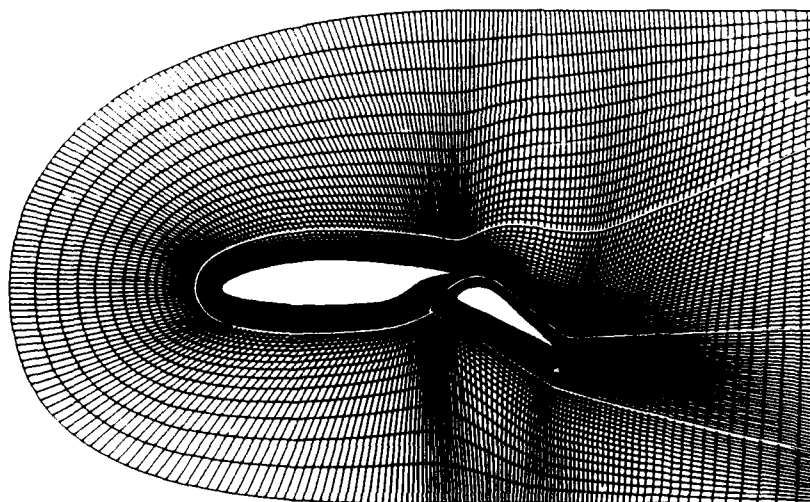structure for grid generation

Fig 3b

Same as above but with reblocked grid to obtain block structure
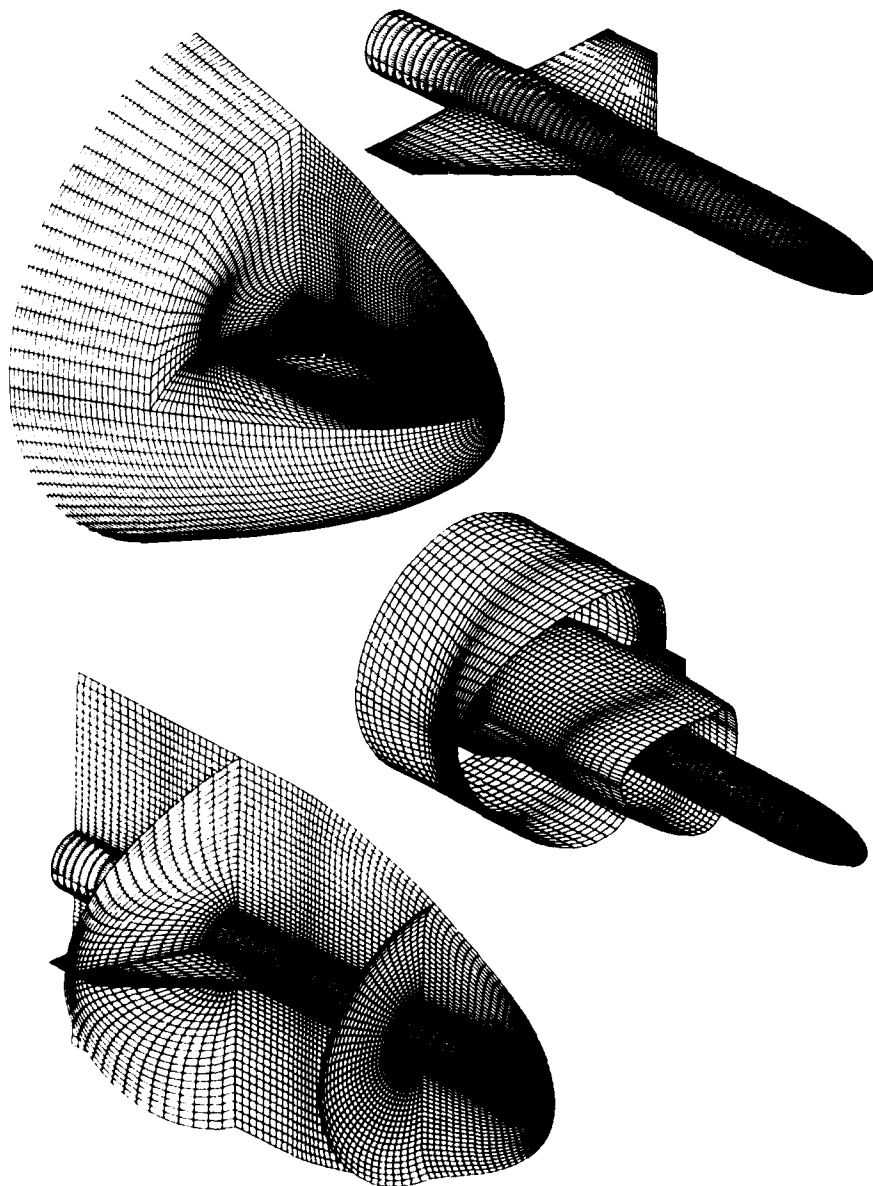more natural for flow solution

Fig 4

Two-block grid around schematic wing-body geometry   Only local
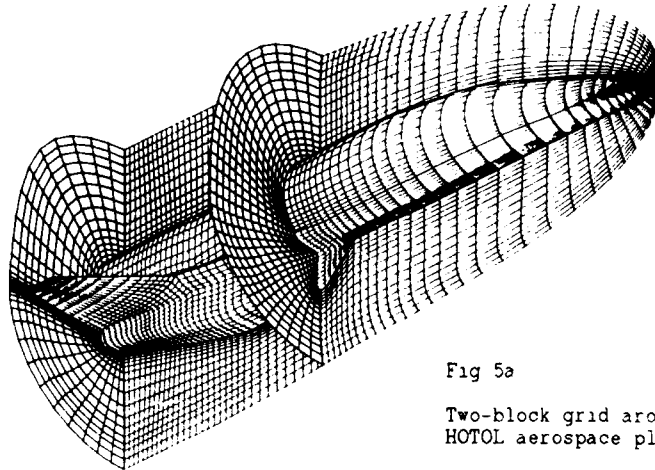algebraic smoothing used

Fig 5a

Two-block grid around the
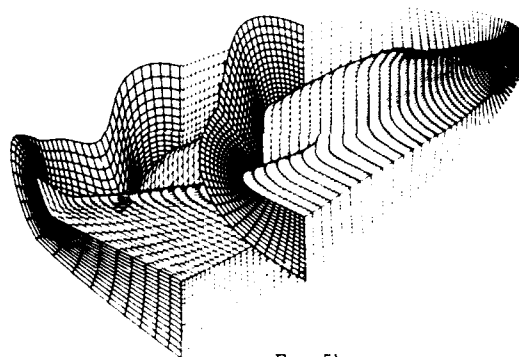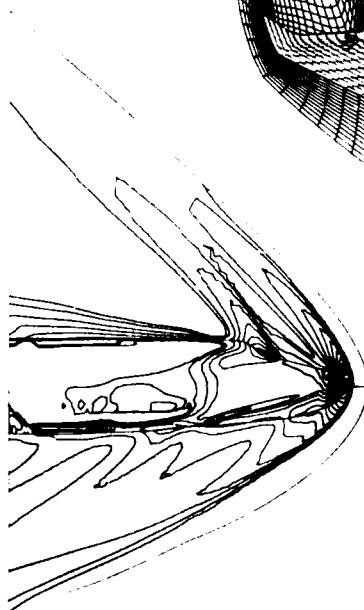HOTOL aerospace plane

Fig 5b

Single-block grid around
the HERMES (old version)
reentry vehicle

Mach contours of Euler
solution on surface and
in symmetry plane for
freestream Mach number 2 0
and 10 degrees incidence

3-D BLOCK-STRUCTURED GRID FOR SUPERSONIC INTAKE GEOMETRY
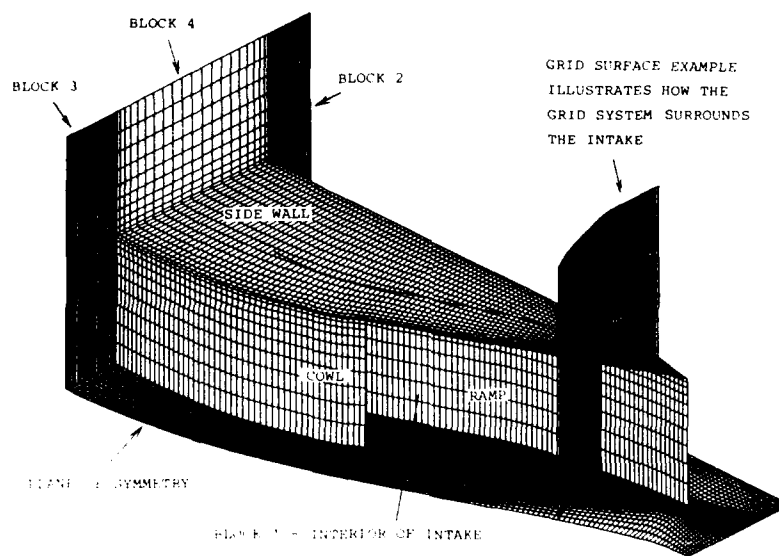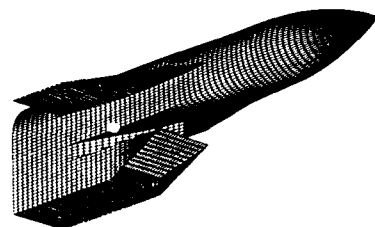
NO. OF BLOCKS: 4

Fig 6

Four-block grid around and inside a 2D supersonic intake. Interior
block has a wedge-type singularity along the leading edge of the
ramp. Outer grid blocks are compatible with a total grid around
complete fuselage/wing/intake

Surface grid
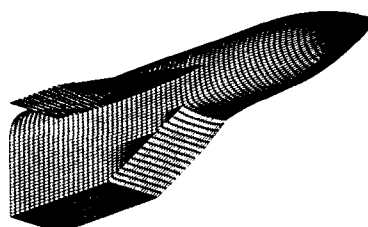
Interior of inlet and boundary
layer diverter region gridded
(2 blocks)
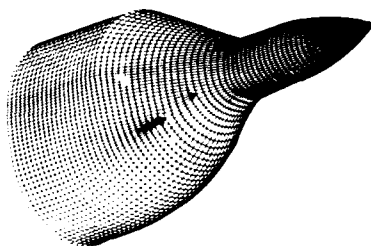
Inner region gridded
5 blocks

Outer region gridded
(5 blocks)

Fig 7a

12-block grid for schematic wing/
fuselage/inlet combination Only
global elliptic grid smoothing used
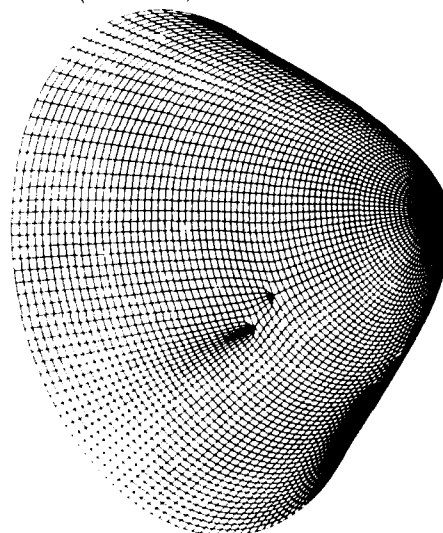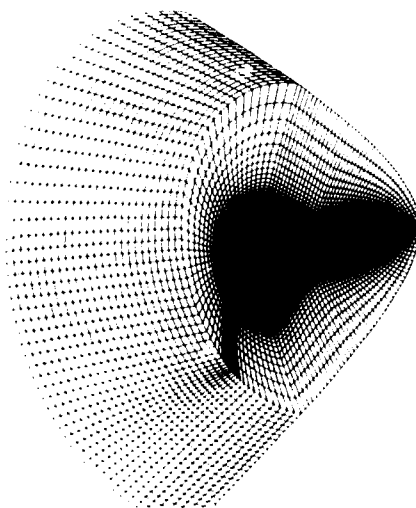
View of complete grid
system

Section in upstream extension of
diverter region

Section downstream of inlet ramp
leading edge

Section just upstream of inlet cowl

Section just downstream of inlet cowl

Fig 7b

Details of various grid sections for same grid as in Fig 7a.

APPLICATIONS OF ALGEBRAIC GRID GENERATION

Peter R. Eiseman
Columbia University
New York, NY 10027

Robert E. Smith
NASA Langley Research Center
Hampton, VA 23665

1 SUMMARY

Techniques and applications of algebraic grid generation are described. The techniques are univariate interpolations and transfinite assemblies of univariate interpolations. Because algebraic grid generation is computationally efficient, the use of interactive graphics in conjunction with the techniques is a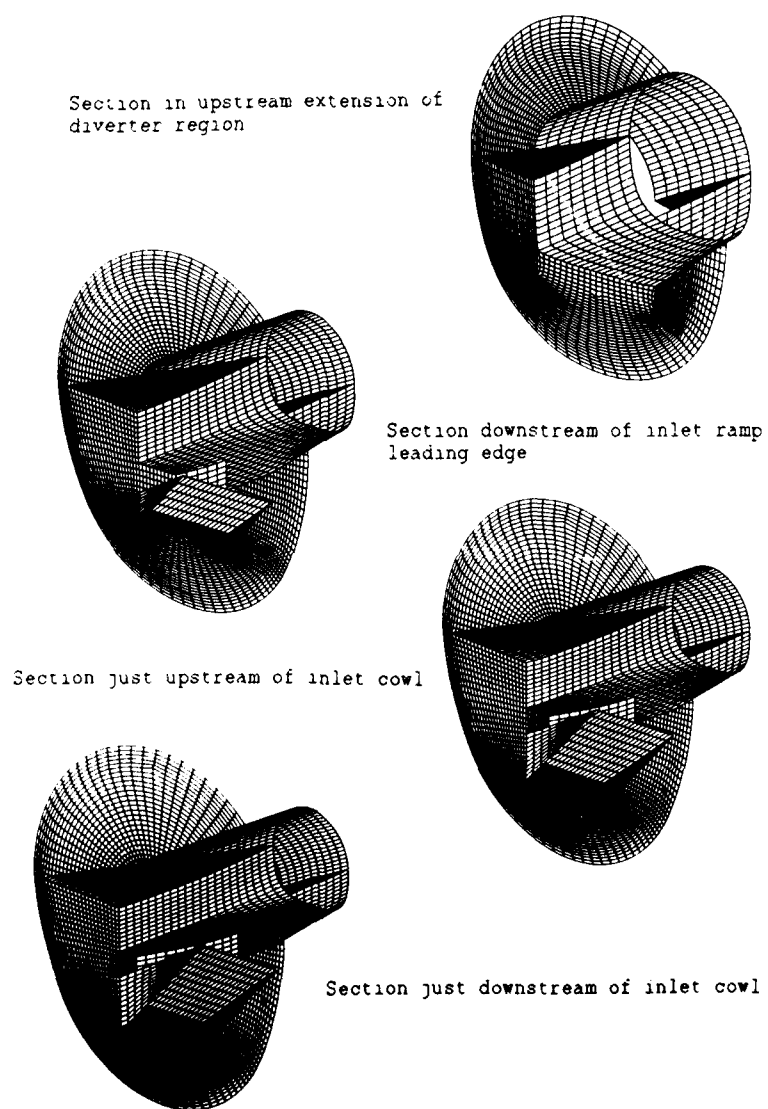dvocated. A flexible approach, which works extremely well in an interactive environment, called the **Control Point Form of Algebraic Grid Generation** is described. The applications discussed are three-dimensional grids constructed about airplane and submarine configurations.

2 INTRODUCTION

The numerical solution of fluid flow problems is directly dependent on the discrete representation of the solution domain. A discrete representation is called a **grid** or **mesh**, and the term **grid** will be used throughout this paper. A grid is a set of points and an implied rule or an explicit table specifying the connectivity of the points. If the implied rule is an index system[1] which is associated with the computational domain, the grid is said to be structured, and the neighboring points of a given point are implied by the preceding and following indices. If the points are irregularly distributed over the solution domain. and there is no rule implying which points are neighbors, then the grid is said to be unstructured. In this case, a table called a **connectivity table** must be created to specify which points are neighbors.

A numerical solution technique for the governing equations does not have a preference for how a grid is generated, but solution software must be developed in accordance with the topology of the solution domain and the structure of the grid. A grid covering a solution domain can consist of several structured blocks, in which case, the connectivity of the blocks must be specified. If the grid is unstructured, then the connectivity of the individual points must be specified. It is also reasonable that a solution domain be covered by a combination of structured and unstructured grids[2]. In any event, it is quite important for the accuracy of a solution that there is an adequate number of grid points to cover the solution domain, that the grid is boundary-fitted. and that the grid points are concentrated in regions where there are high gradients in the solution[3]. It is also important that the spacing between points varies smoothly, and that the the skewness not be excessive[4].

Algebraic grid generation techniques are interpolation or approximation procedures that relate a computational domain, which is a rectangular parallelepiped (a square in two dimensions and a box in three dimensions), to an arbitrarily-shaped physical domain with corresponding sides[3]. A side in the computational domain can map into a line or point in the physical domain, in which case, a singularity occurs in the mapping. Singularities do not pose a problem to finite-volume techniques[5], which dominate current solution approaches for fluid flow, nor do singularities affect solution techniques that use unstructured grids. The interpolations are univariate functions of the individual coordinates in the computational domain, which are combined in a Boolean sum to create the complete transformation. Often, for a particular application, a higher order and more sophisticated interpolation is used in one coordinate direction, which we will call the primary coordinate direction; and low order interpolation, such as linear interpolation, is used in the remaining coordinate directions.

There are as many ways to generate algebraic grids as there are interpolation methods. It is impossible to cover all methods, but general characteristics of transfinite assemblies of univariate interpolation are briefly reviewed. A newly-introduced method called the **Control Point Form of Algebraic Grid Generation** is described.

Algebraic grid generation methods are very efficient and work very well in conjunction with interactive computer graphics. The application of the **Two-Boundary Technique** in an interactive environment is discussed. Also, the **Control Point Form of Algebraic Grid Generation** is advocated for interactive applications.

Applications of algebraic grid generation are varied. Herein, we present applications of the **Two-Boundary Technique**, **Control Point Form of Algebraic Grid Generation** and **Lagrangian Interpolation** in the context of **Transfinite Interpolation**. The applications are three-dimensional grids about airplane and submarine configurations.
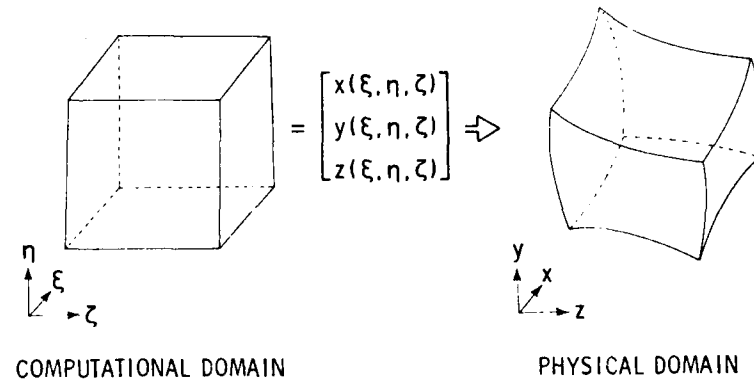
COMPUTATIONAL DOMAIN   PHYSICAL DOMAIN

Figure 1: Transformation Between Computational and Physical Domains

3 TRANSFORMATIONS AND GRIDS

Generally speaking, all algebraic grid generation techniques can be thought of as transformations from a rectangular computational domain to an arbitrarily-shaped physical domain. This is shown schematically in Figure 1 and as a general equation:

$$\mathbf{X}(\xi, \eta, \zeta) = \begin{bmatrix} x(\xi, \eta, \zeta) \\ y(\xi, \eta, \zeta) \\ z(\xi, \eta, \zeta) \end{bmatrix} , \qquad \cdot$$

$$0 \le \xi \le 1, \qquad 0 \le \eta \le 1, \qquad 0 \le \zeta \le 1.$$

A discrete subset of the vector-valued function: $\mathbf{X}(\xi_i, \eta_j, \zeta_k)$ is a structured grid for

$$\xi_i = \tfrac{i-1}{L-1}, \qquad \eta_j = \tfrac{j-1}{M-1}, \qquad \zeta_k = \tfrac{k-1}{N-1},$$

where $i = 1, 2, 3 \cdots, L$, $j = 1, 2, 3, \cdots, M$, and $k = 1, 2, 3, \cdots, N$. The relationship between the indices i, j, and k and the computational coordinates ξ, η, and ζ uniformly discretize the computational domain and imply a relationship between discrete neighboring points. The transformation to the physical domain produces the actual grid points, and the relationship of neighboring grid points is invarient under the transformation.

An unstructured grid can be defined on the computational domain by subdividing the domain into triangles in two dimensions and tetrahedrons in three dimensions. A tetrahedron is defined by four points and has four faces. The ith tetrahedron is denoted by

$$T_i \quad \equiv \quad [F_1, F_2, F_3, F_4]_i, \ ,$$

where $i = 1, 2, 3, \ldots, I$, and the four faces are each defined by three points

$$F_{ji} \quad \equiv \quad \mathbf{X}_{ji}(\xi_k, \eta_k, \zeta_k) \,,$$

where $i = 1, 2, 3, \cdots, I$, $j = 1, 2, 3, 4$, and $k = 1, 2, 3$. A table must be created such that

$$F_{ji} \quad \Longleftarrow \quad TABLE \quad \Longrightarrow \quad F_{ji}$$
$$i \ne \dot{i}.$$

If the computational domain is discretized in an unstructured manner, then the grid points in the physical domain have the same unstructured relationship. This is because of the invariance of the relation between points under the transformation expressed in the general form of the first equation. Thus, algebraic techniques are capable of producing either a structured or an unstructured grid.

For complex physical domains where structured grids are employed, it is most often necessary to create many blocks[1], where each block has the general form of Eq. 1. In this case, the connectivity of the blocks must be specified. That is, an explicit table is generated which denotes the blocks and the corresponding grid points at block interfaces. Two approaches that are also multi-block, but do not use exact grid point overlap are; the **Conservative Interface Approach**[6] and the **CHIMERA**[7] grid scheme. In the conservative interface approach, grid points from different blocks meet at common interface surfaces but the grid points do not coincide (Fig. 2). Variables in a flow solution are transferred from one block to another in such a manner that the variables are conserved. In the CHIMERA scheme, grid blocks overlap, and in a flow field solution, variables are simply interpolated from one block to another without assurance of a conservation property.

The primary advantage of structured grids is simplicity. When the number of structured grid blocks becomes large, the simplicity is lost. In order to maintain simplicity where it is needed, such as in a boundary layer, and to accommodate geometric complexity, it is likely that combinations of structured and unstructured grids will prevail.
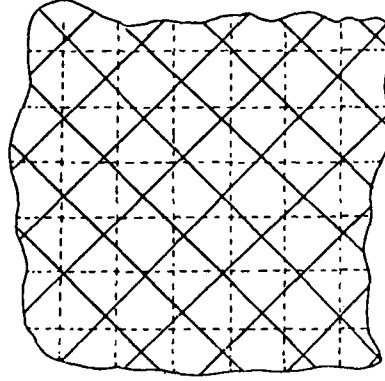
Figure 2: Discontinuous Grid Interfaces

4 TECHNIQUES

The interpolation methodology that dominates algebraic grid generation is called transfinite interpolation. This methodology was first described for grids by Gorden and Hall [8], and the advantage is that it provides complete conformity to boundaries. Transfinite interpolation has since been described many times[1,3,8,9]. However, a basic description is included herein so that specific grid generation techniques that are characterized by this methodology can be discussed.

The ingredients of transfinite interpolation are univariate interpolations in the computational coordinate directions defined by

$$\mathbf{U}(\xi, \eta, \zeta) = \sum_{i=1}^{L} \sum_{n=0}^{P} \alpha_i^n(\xi) \frac{\partial^n \mathbf{X}(\xi_i, \eta, \zeta)}{\partial \xi^n} ,$$

$$\mathbf{V}(\xi, \eta, \zeta) = \sum_{j=1}^{M} \sum_{m=0}^{Q} \beta_j^m(\eta) \frac{\partial^m \mathbf{X}(\xi, \eta_j, \zeta)}{\partial \eta^m} ,$$

$$\mathbf{W}(\xi, \eta, \zeta) = \sum_{k=1}^{N} \sum_{\ell=0}^{R} \gamma_k^\ell(\zeta) \frac{\partial^\ell \mathbf{X}(\xi, \eta, \zeta_k)}{\partial \zeta^\ell} ,$$

$$(D^n \alpha_i^n)(\xi_i) = \delta_{ii} \delta_{nn}, \qquad (D^m \beta_j^m)(\eta_j) = \delta_{jj} \delta_{mm}, \qquad (D^\ell \gamma_k^\ell)(\zeta_k) = \delta_{kk} \delta_{\ell\ell},$$

$$\bar{i} = 1, 2, \cdots, L, \qquad \bar{j} = 1, 2, \cdots, M, \qquad \bar{k} = 1, 2, \cdots, N,$$

$$\bar{n} = 0, 1, \cdots, P \qquad \bar{m} = 0, 1, \cdots, Q, \qquad \bar{\ell} = 0, 1, \cdots, R.$$

where D is a derivative operator. The tensor products are

$$\mathbf{UW} = \mathbf{WU} = \sum_{i=1}^{L} \sum_{k=1}^{N} \sum_{\ell=0}^{R} \sum_{n=0}^{P} \alpha_i^n(\xi) \gamma_k^\ell(\zeta) \frac{\partial^{\ell+n} X(\xi_i, \eta, \zeta_k)}{\partial \zeta^\ell \partial \xi^n} ,$$

$$\mathbf{UV} = \mathbf{VU} = \sum_{i=1}^{L} \sum_{j=1}^{M} \sum_{m=0}^{Q} \sum_{n=0}^{P} \alpha_i^n(\xi) \beta_j^m(\eta) \frac{\partial^{n+m} X(\xi_i, \eta_j, \zeta)}{\partial \eta^m \partial \xi^n} ,$$

$$\mathbf{VW} = \mathbf{WV} = \sum_{j=1}^{M} \sum_{k=1}^{N} \sum_{m=0}^{Q} \sum_{\ell=0}^{R} \beta_j^m(\xi) \gamma_k^\ell(\zeta) \frac{\partial^{\ell+m} X(\xi, \eta_j, \zeta_k)}{\partial \zeta^\ell \partial \eta^m} ,$$

$$\mathbf{UVW} = \sum_{i=1}^{L} \sum_{j=1}^{M} \sum_{k=1}^{N} \sum_{\ell=0}^{R} \sum_{m=0}^{Q} \sum_{n=0}^{P} \alpha_i^n(\xi) \beta_j^m(\eta) \gamma_k^\ell(\zeta) \frac{\partial^{\ell+m+n} X(\xi_i, \eta_j, \zeta_k)}{\partial \zeta^\ell \partial \eta^m \partial \xi^n} .$$

The commutativity in the above tensor products is assumed in many practical situations, but in general, it is not guaranteed! It generally depends upon the commutativity of the mixed partial derivatives. For the grid generation techniques that are characterized by transfinite interpolation, they can be written as the Boolean sum

$$\mathbf{X}(\xi, \eta, \zeta) = \mathbf{U} \oplus \mathbf{V} \oplus \mathbf{W} =$$

$$\mathbf{U} + \mathbf{V} + \mathbf{W} - \mathbf{UV} - \mathbf{UW} - \mathbf{VW} + \mathbf{UVW}.$$

In the above equations, $\alpha_i^n(\xi), \beta_j^m(\eta)$ and $\gamma_k^\ell(\zeta)$ are interpolation functions subject to δ function conditions. The defining parameters $\frac{\partial^{\cdots}X(\xi_i, \eta_j, \zeta_k)}{\partial\zeta^\ell \partial\eta^m \partial\xi^n}$ in the equations are positions (when $i = m = n = 0$) and partial derivatives (otherwise) in the physical domain which are user specified. In this definition, the implicit assumption is that coordinate curves are to be **interpolated** along with their derivatives. This occurs through a network of intersecting surfaces that must be specified. By taking derivatives of this interpolation-theoretic framework, the interpolation functions, then specified, will have an effective meaning only for derivatives - not for the surfaces. The result will be an **approximation** rather than an interpolation.

Under the umbrella of transfinite interpolation, there are many possible algebraic grid generation techniques. The most successful techniques, however, have been those that provide adequate orthogonality control and grid spacing control with acceptable functional complexity. For instance, the two-boundary technique described by Smith[10] uses Hermite cubic interpolation functions in one coordinate direction between two opposing boundary surfaces. In the two remaining coordinate directions, linear interpolation between opposing boundaries is specified. In a similar philosophy, Eriksson, in many of his applications of transfinite interpolation[11], has used Lagrangian interpolation functions, where two, three, or four surfaces are specified in each coordinate direction. He has also used positions and derivatives at opposing boundaries[9].

A popular grid generation technique is the multisurface method described by Eiseman[12-15]. It is a very flexible univariate scheme which is similar to Bézier and B-Spline approximation [16-17], where the parameters defining a curve are not on the curve. In the latest version of the multi-surface method, the blending functions $\alpha_i(\xi), \beta_j(\eta)$ and $\gamma_k(\zeta)$ are nontrivial only over a local region. This means that the position parameters inside the region affect the grid in a local manner.

Recently, Eiseman introduced the **Control Point Form of Algebraic Grid Generation (CPF)**[18] which is a multiple variable multi-surface transformation. In this approach, a sparse grid, denoted by $\mathbf{q}_{i,j,k}$, is first generated. $\mathbf{q}_{i,j,k}$ can be obtained quite simply by a transformation defined by linear interpolation functions in the transfinite interpolation methodology which blends specified boundary data into the interior region. Alternatively, it can be obtained by attachment to any given grid, regardless of how the grid was generated. Attachment is simply the process by which control points are placed in order to essentially reproduce a given transformation or grid. The chosen number of control points for each direction is dictated solely by the amount of control that a user wishes to specify. It is independent of the chosen number of grid points.

In keeping with the previous notation and interpolation structure, the univariate multi-surface transformations in the three coordinate directions are

$$\frac{\partial \mathbf{U}}{\partial \xi} = \sum_{i=1}^{L} (D\alpha_i^0)(\xi) \left. \frac{\partial \mathbf{X}}{\partial \xi} \right|_{\xi = \xi_i},$$

$$\frac{\partial \mathbf{V}}{\partial \eta} = \sum_{j=1}^{M} (D\beta_j^0)(\eta) \left. \frac{\partial \mathbf{X}}{\partial \eta} \right|_{\eta = \eta_j},$$

$$\frac{\partial \mathbf{W}}{\partial \zeta} = \sum_{k=1}^{N} (D\gamma_k^0)(\zeta) \left. \frac{\partial \mathbf{X}}{\partial \zeta} \right|_{\zeta = \zeta_k},$$

where

$$\alpha_i^n = 0 \text{ for } n \neq 0, \quad \beta_j^m = 0 \text{ for } m \neq 0, \quad \gamma_k^\ell = 0 \text{ for } \ell \neq 0.$$

In this situation, it is the coefficients of the **X** derivatives, which are specified to interpolate those derivatives at successive stations in the curvilinear variables. For example, in the first equation, the specification is for $(D\alpha_i^0)$ rather than α_i^0. It is simply the derivative which gets the Krouecker delta condition rather than the function α_i^0. In formal terms, this is stated as

$$(D\alpha_i^0)(\xi_m) = \delta_{im}, \quad (D\beta_j^0)(\eta_m) = \delta_{jm}, \quad (D\gamma_i^0)(\zeta_m) = \delta_{km}.$$

Upon integration together with the requirement that opposing boundaries are precisely adhered to, we get a sequence of intermediate surfaces for each direction that appear in the form

$$\mathbf{U}(\xi, \eta, \zeta) = \mathbf{A}_1(\eta, \zeta) + \sum_{i=1}^{L} \alpha_i^0(\xi)[\mathbf{A}_{i+1}(\eta, \zeta) - \mathbf{A}_i(\eta, \zeta)],$$

$$\mathbf{V}(\xi, \eta, \zeta) = \mathbf{B}_1(\xi, \zeta) + \sum_{j=1}^{M} \beta_j^0(\eta)[\mathbf{B}_{j+1}(\xi, \zeta) - \mathbf{B}_j(\xi, \zeta)],$$

$$\mathbf{W}(\xi, \eta, \zeta) = \mathbf{C}_1(\xi, \eta) + \sum_{k=1}^{N} \gamma_k^0(\zeta)[\mathbf{C}_{k+1}(\xi, \eta) - \mathbf{C}_k(\xi, \eta)].$$

where

$$\alpha_i^0 = \frac{\phi_i(\xi)}{\phi_i(\xi_L)} \qquad with \qquad \phi_i(\xi) = \int_{\xi_i}^{\xi} (D\alpha_i^0)d\sigma \ ,$$

$$\beta_j^0 = \frac{\theta_j(\eta)}{\theta_j(\eta_M)} \qquad with \qquad \theta_j(\eta) = \int_{\eta_i}^{\eta} (D\beta_j^0)d\sigma \ ,$$

$$\gamma_k^0 = \frac{\psi_k(\zeta)}{\psi_k(\zeta_N)} \qquad with \qquad \psi_k(\zeta) = \int_{\zeta_i}^{\zeta} (D\gamma_k^0)d\sigma \ ,$$

and where the **constructive surfaces** are related to the grid array $\mathbf{X}$ by

$$\mathbf{A}_\ell(\eta,\zeta) = \mathbf{X}(\xi_1,\eta,\zeta) + \sum_{i=1}^{\ell-1} \phi_i(\xi_L) \left. \frac{\partial \mathbf{X}}{\partial \xi} \right|_{\xi=\xi_i} \ ,$$

$$\mathbf{B}_m(\xi,\zeta) = \mathbf{X}(\xi,\eta_1,\zeta) + \sum_{j=1}^{m-1} \theta_j(\eta_M) \left. \frac{\partial \mathbf{X}}{\partial \eta} \right|_{\eta=\eta_j} \ ,$$

$$\mathbf{C}_n(\xi,\eta) = \mathbf{X}(\xi,\eta,\zeta_1) + \sum_{k=1}^{n-1} \psi_k(\zeta_N) \left. \frac{\partial \mathbf{X}}{\partial \zeta} \right|_{\zeta=\zeta_k} \ .$$

More details on this process can be found in Eiseman [15].

The primary control over the grid is exercised by the prescription of the surfaces $\mathbf{A}_i, \mathbf{B}_j$ and $\mathbf{C}_k$. This is done for boundary conformity and for the shape of curves connecting opposing boundaries. The specification of surfaces, however, requires a substantial amount of data, and accordingly, there is the problem of efficiently creating or manipulating them for the purpose of generating grids. This problem is inherent in any algebraic technique that requires data in addition to the specification of the boundaries. This includes the specification of derivative data only at the boundaries; for there, a vector specification must be dealt with at each boundary point - an equivalent problem to that of specifying an entire surface.

While such specifications may not present an extreme burden in unidirectional techniques, they grow substantially with a straight-forward multivariate assembly represented by the process of transfinite interpolation. This burden is particularly accute in three dimensions where the specifications are for surfaces rather than for curves which would be the requirement in two dimensions. In the straight-forward assembly, we must deal with the art of constructing surfaces or curves in each coordinate direction together with the details of consistency between the surface or curve parameters for the distinct directions.

To overcome the burdensome constraints of dealing with constructive surfaces or curves in a consistent manner, we are lead to the **The Control Point Form of Algebraic Grid Generation (CPF)**. The central idea is to replace the surface or curve specifications with **sparse** arrays of control points that can be used to generate the required surfaces or curves with the same scheme that is employed between surfaces. Moreover, a control point array is used to generate the requisite surface or curve data in all directions. This, accordingly, removed the need to deal with a consistency problem.

In our discussion, we will follow the development in Eiseman [18], but will depart from that discourse by first casting the multisurface transformation in a surface weighted format that comes from a minor shuffle of terms. For the ξ-direction construction, we have

$$\mathbf{U}(\xi,\eta,\zeta) = [1 - \alpha_1^0(\xi)]\mathbf{A}_1(\eta,\zeta) + [\alpha_1^0(\xi) - \alpha_2^0(\xi)]\mathbf{A}_2^0(\eta,\zeta)$$

$$+ [\alpha_2^0(\xi) - \alpha_3^0(\xi)]\mathbf{A}_3(\eta,\zeta) + \cdots + \alpha_L^0(\xi)\mathbf{A}_{L+1}(\eta,\zeta),$$

and by the same shuffle we get the parallel expressions for the η and ζ directions of $\mathbf{V}$ and $\mathbf{W}$ respectively. Altogether, we arrive at the form

$$\mathbf{U}(\xi,\eta,\zeta) = \sum_{i=1}^{L+1} \alpha_i(\xi)\mathbf{A}_i(\eta,\zeta),$$

$$\mathbf{V}(\xi,\eta,\zeta) = \sum_{j=1}^{M+1} \beta_j(\eta)\mathbf{B}_j(\xi,\zeta),$$

$$\mathbf{W}(\xi,\eta,\zeta) = \sum_{k=1}^{N+1} \alpha_k(\zeta)\mathbf{C}_k(\xi,\eta),$$

where

$$\alpha_i(\xi) = \begin{cases} 1 - \alpha_1^0(\xi) & \text{for } i = 1 \\ \alpha_{i-1}^0(\xi) - \alpha_i^0(\xi) & \text{for } i = 2,\ldots,L \\ \alpha_L^0(\xi) & \text{for } i = L+1 \end{cases},$$

$$\beta_j(\eta) = \begin{cases} 1 - \beta_1^0(\eta) & \text{for } j = 1 \\ \beta_{j-1}^0(\eta) - \beta_j^0(\eta) & \text{for } j = 2,\ldots,M \\ \beta_M^0(\eta) & \text{for } j = M+1 \end{cases},$$

$$\gamma_k(\zeta) = \begin{cases} 1 - \gamma_1^0(\zeta) & \text{for } k = 1 \\ \gamma_{k-1}^0(\zeta) - \gamma_k^0(\zeta) & \text{for } i = 2,\ldots,N \\ \gamma_N^0(\zeta) & \text{for } k = N+1 \end{cases}.$$

In this format, the conditions for matching the faces of opposing boundaries are expressed by the pairs of equations:

$$\begin{cases} \mathbf{A}_1(\eta,\zeta) = \mathbf{X}(\xi_1,\eta,\zeta) \\ \mathbf{A}_{L+1}(\eta,\zeta) = \mathbf{X}(\xi_L,\eta,\zeta) \end{cases},$$

$$\begin{cases} \mathbf{B}_1(\xi,\zeta) = \mathbf{X}(\xi,\eta_1,\zeta) \\ \mathbf{B}_{M+1}(\xi,\zeta) = \mathbf{X}(\xi,\eta_M,\zeta) \end{cases},$$

$$\begin{cases} \mathbf{C}_1(\xi,\eta) = \mathbf{X}(\xi,\eta,\zeta_1) \\ \mathbf{C}_{N+1}(\xi,\eta) = \mathbf{X}(\xi,\eta,\zeta_N) \end{cases}.$$

That is, one pair of opposing boundaries is matched for each respective ξ, η and ζ directional constructs as independently represented by $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$.

To develop the **CPF**, we assume that there is a sparse array of control points

$$\{\mathbf{q}_{i,j,k} : i = 1,2,\ldots,L+1, \quad j = 1,2,\ldots,M+1, \quad k = 1,2,\ldots,N+1\},$$

where immediately we see that any normal sequence of control points can be used to generate a curve which connects the first and last control points of the sequence. Those curves are given by

$$\mathbf{a}_{j,k}(\xi) = \sum_{i=1}^{L+1} \alpha_i(\xi)\mathbf{q}_{i,j,k},$$

$$\mathbf{b}_{i,k}(\eta) = \sum_{j=1}^{M+1} \beta_j(\eta)\mathbf{q}_{i,j,k},$$

$$\mathbf{c}_{i,j}(\zeta) = \sum_{k=1}^{N+1} \gamma_k(\zeta)\mathbf{q}_{i,j,k}.$$

By using these curves, however, we can continue and generate surfaces which match control points at their corners. These surfaces are given by

$$\mathbf{A}_i(\eta,\zeta) = \sum_{j=1}^{M+1} \sum_{k=1}^{N+1} \beta_j(\eta)\gamma_k(\zeta)\mathbf{q}_{i,j,k},$$

$$\mathbf{B}_j(\xi,\zeta) = \sum_{i=1}^{L+1} \sum_{k=1}^{N+1} \alpha_i(\xi)\gamma_k(\zeta)\mathbf{q}_{i,j,k},$$

$$\mathbf{C}_k(\xi,\eta) = \sum_{i=1}^{L+1} \sum_{j=1}^{M+1} \alpha_i(\xi)\beta_j(\eta)\mathbf{q}_{i,j,k},$$

where now we use this notation also for the end condition ($i = 1$ or $L+1$, $j = 1$ or $M+1$, $k = 1$ or $N+1$) rather than the specifications in the original statement of the multisurface transformation. The multisurface transformations are now written as

$$\mathbf{U}(\xi,\eta,\zeta) \;=\; \alpha_1(\xi)\mathbf{X}(\xi_1,\eta,\zeta) \;+\; \sum_{i=2}^{L}\alpha_i(\xi)\mathbf{A}_i(\eta,\zeta) \;+\; \alpha_{L+1}(\xi)\mathbf{X}(\xi_L,\eta,\zeta),$$

$$\mathbf{V}(\xi,\eta,\zeta) \;=\; \beta_1(\eta)\mathbf{X}(\xi,\eta_1,\zeta) \;+\; \sum_{j=2}^{M}\beta_j(\eta)\mathbf{B}_j(\xi,\zeta) \;+\; \beta_{M+1}(\eta)\mathbf{X}(\xi,\eta_M,\zeta),$$

$$\mathbf{W}(\xi,\eta,\zeta) \;=\; \gamma_1(\zeta)\mathbf{X}(\xi,\eta,\zeta_1) \;+\; \sum_{k=2}^{N}\gamma_k(\zeta)\mathbf{C}_k(\xi,\eta) \;+\; \gamma_{N+1}(\zeta)\mathbf{X}(\xi,\eta,\zeta_N).$$

When using only the surfaces $\mathbf{A}_i, \mathbf{B}_j$ or $\mathbf{C}_k$ in the construction of the multisurface transformation, we get a transformation that is solely determined by the control points. It is called the **tensor product transformation** and is given by

$$\mathbf{T}(\xi,\eta,\zeta) \;=\; \sum_{i=1}^{L+1}\sum_{j=1}^{M+1}\sum_{k=1}^{N+1}\alpha_i(\xi)\beta_j(\eta)\gamma_k(\zeta)\mathbf{q}_{ijk}.$$

This determines the volume grid for the control points. By inserting it into the the multisurface constructs just given, we get

$$\mathbf{U}(\xi,\eta,\zeta) \;=\; \mathbf{T}(\xi,\eta,\zeta) \;+\; \alpha_1(\xi)[\mathbf{X}(\xi_1,\eta,\zeta) - \mathbf{A}_1(\eta,\zeta)] + \alpha_{L+1}(\xi)[\mathbf{X}(\xi_L,\eta,\zeta) - \mathbf{A}_{L+1}(\eta,\zeta)],$$

$$\mathbf{V}(\xi,\eta,\zeta) \;=\; \mathbf{T}(\xi,\eta,\zeta) \;+\; \beta_1(\eta)[\mathbf{X}(\xi,\eta_1,\zeta) - \mathbf{B}_1(\xi,\zeta)] \;+\; \beta_{M+1}(\eta)[\mathbf{X}(\xi,\eta_M,\zeta) - \mathbf{B}_{M+1}(\xi,\zeta)],$$

$$\mathbf{W}(\xi,\eta,\zeta) \;=\; \mathbf{T}(\xi,\eta,\zeta) \;+\; \gamma_1(\zeta)[\mathbf{X}(\xi,\eta,\zeta_1) - \mathbf{C}_1(\xi,\eta)] \;+\; \gamma_{N+1}(\zeta)[\mathbf{X}(\xi,\eta,\zeta_N) - \mathbf{C}_{N+1}(\xi,\eta)].$$

This form is particularly instructive since we can now explicitly view the issue of boundary conformity in each direction as being controlled by an adjustment term for each face of a cube. That adjustment represents the deviation from a pure control point representation to one of an exact boundary specification.

In a similar spirit, we wish to examine the tensor product of any two of the constructs with the given adjustment terms. To avoid repetitive manipulation, the tensor product **UV** will be expanded and will set the pattern for **UW** and **VW**. The expansion proceeds by first applying **U** to **V** instead of **X** whereby **UV** becomes

$$\mathbf{T}(\xi,\eta,\zeta) \;+\; \alpha_1(\xi)\left\{\mathbf{V}(\xi_1,\eta,\zeta) - \mathbf{A}_1(\eta,\zeta)\right\} \;+\; \alpha_{i+1}(\xi)[\mathbf{V}(\xi_L,\eta,\zeta) - \mathbf{A}_{i+1}(\eta,\zeta)].$$

With some algebraic manipulation and relabeling of control point entities, we have

$$\mathbf{UV} \;=\; \mathbf{T}(\xi,\eta,\zeta) \;+\; \alpha_1(\xi)\beta_1(\eta)\left\{\mathbf{X}(\xi_1,\eta_1,\zeta) - \mathbf{c}_{1,1}(\zeta)\right\} \;+\; \alpha_1(\xi)\beta_{M+1}(\eta)\left\{\mathbf{X}(\xi_1,\eta_M,\zeta) - \mathbf{c}_{1,M+1}(\zeta)\right\}$$
$$+\alpha_{L+1}(\xi)\beta_1(\eta)\left\{\mathbf{X}(\xi_L,\eta_1,\zeta) - \mathbf{c}_{L+1,1}(\zeta)\right\} \;+\; \alpha_{L+1}(\xi)\beta_{M+1}(\eta)\left\{\mathbf{X}(\xi_L,\eta_M,\zeta) - \mathbf{c}_{L+1,M+1}(\zeta)\right\}.$$

which, as in the case of boundary faces, explicitly separates out the boundary edge blending terms relative to the pure control point dependency represented by the tensor product **T**. Those edges are the cube edges in ζ that are transverse to the tensor product in ξ and η represented by **U** and **V** as **UV**.

By applying the established pattern to **UW** and **VW** we can evaluate the Boolean sum

$$\mathbf{U} \oplus \mathbf{V} \oplus \mathbf{W} = \mathbf{U} + \mathbf{V} + \mathbf{W} - \mathbf{UV} - \mathbf{UW} - \mathbf{VW} + \mathbf{UVW},$$

which reduces to a tensor product core represented by **T** along with a simple adjustment term for each face or edge of the grid block. Each adjustment term appears as a blending function times the difference between the specified boundary part and the corresponding control point representation for the same part. When the part is an edge in one variable, the blending function is the product of the closest surface coefficients in the remaining two variables. When the part is a face in two variables, the blending function is just the coefficient for that face in the expansion for the remaining variable.

When the adjustment terms corresponding to a combination of edges and faces is dropped, the effect is a dependency only upon control points for those corresponding parts of the boundary. The practical implication is that **any** combination of specified and free formable boundaries can be employed. This is in sharp contrast to traditional transfinite methods.

One final observation is evident from the surface weighted format employed in our discussion of the control point formulation. It is simply that the weighting functions $\alpha_i, \beta_j,$ and γ_k could have been chosen arbitrarily rather than in the careful way done here. That care comes from the multisurface construct which amounted to the interpolation being applied to the tangent vectors for our curve and, accordingly, accounting for curvature control in a direct manner. The common Bézier and B-Spline methods, by contrast, only have a convex hull property. This is essentially a much weaker form of curvature control. Nonetheless, we can also use such methods in the control point form. To use the Bézier functions (Bernstein Polynomials), we need only set

$$\alpha_i(\xi) = \begin{pmatrix} L+1 \\ i \end{pmatrix}\xi^i(1-\xi)^{L+1-i}, \qquad \begin{pmatrix} L+1 \\ i \end{pmatrix} = \frac{(L-1)!}{(L+1-i)!i!},$$

and $0 \leq \xi \leq 1$. Similar expressions would result for β_j and γ_k. In continuation, Lagrangian interpolation can be applied, or some mix of various interpolation and approximation types can be used for the distinct directions represented by $\alpha_i, \beta_j,$ or γ_k.
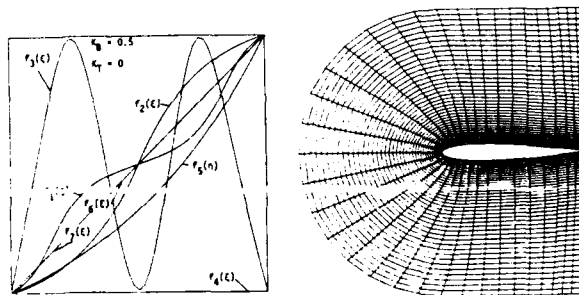
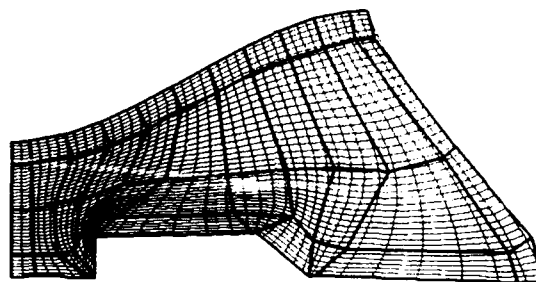Figure 3: Control Functions for Two-Boundary Technique

Figure 4: **CPF** Sparse/Resultant Grid

5 INTERACTIVE GRID GENERATION

As it has been previously stated, algebraic grid generation techniques are computationally efficient. They are, therefore. ideally suited for an interactive environment. The two-boundary technique [19] and the **CPF** [18,21]technique have been cast in this environment for two-dimensional and quasi three-dimensional applications. In the two-boundary technique, displays and control functions (Fig. 3) are interactively created and the grid and grid characteristics are computed in sequence and visually inspected.

In the case of the **CPF** approach. a nominal grid is displayed and a sparse control net is bold-faced and superimposed onto the nominal grid (Fig. 4). Using an interactive device (mouse and cursor) a particular point in the sparse net is identified and moved to another position. In sequence, a new primary grid is computed and displayed. The creation of control functions in the interactive two-boundary technique and the movement of control points in the **CPF** technique can be performed in **real time** using a **state of the art** workstation such as the IRIS 3030. That is, a response to input is computed and displayed as fast as the user can change the input. As workstations become faster and frame buffers[21] connected to supercomputers become available, the entire interactive grid generation process will likely be in **real time**.

6 APPLICATIONS

The generation of structured grids about three-dimensional configurations such as airplanes or submarines requires several planning and construction steps[22]. We assume that there is some original definition of the configuration, such as component cross sections or patch data base[23]. Given the configuration, the first step is planning the topology, which includes the number, location and connectivity of grid blocks. The second step is the determination of a suitable grid on the surface of the configuration. The third step is the construction of intermediate and far field surface grids that correspond to block faces. The fourth step is the interior grid generation for the blocks.

The tools of algebraic grid generation are techniques as described above and software designed to apply the techniques in a **specific** setting. The terminology **specific** is used to indicate that the software can be applied to a particular configuration or class of configurations without changing the source code. There is general software for grid generation, for instance, the EAGLE code[24]. The EAGLE code authored by Joe Thompson et. al. encompasses both transfinite interpolation and differential methods [25]. Boundary definition is interpolated to the interior of blocks using Lagrangian interpolation functions, and the resulting algebraic grid can be smoothed using differential methods. It is the authors' conjecture, however,
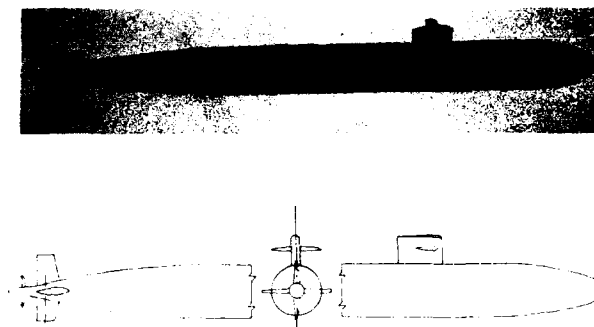
Figure 8: Submarine Configuration

TOP
WING–FUSELAGE GRID

BOTTOM
WING–FUSELAGE GRID

TOP–FRONT GRID

BOTTOM–FRONT GRID

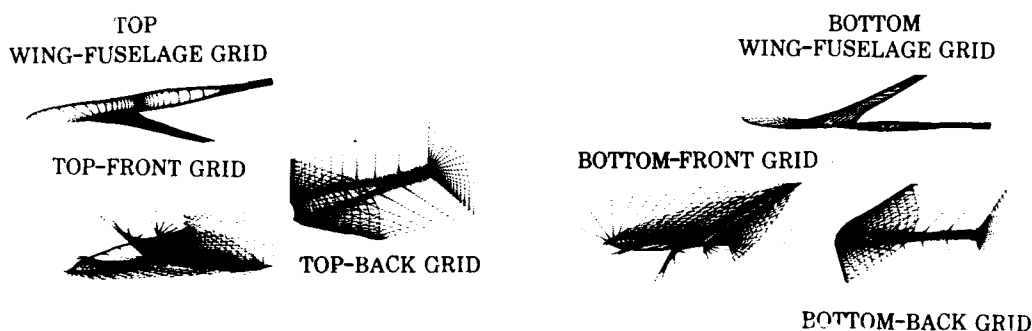TOP–BACK GRID

BOTTOM–BACK GRID

Figure 9: Wing-Fuselage Grid Surfaces

to generate a grid on the four blocks as seen in Figure 9 after a surface grid has been obtained. Reference 22 provides details on this grid generation.

H-type topology is more suitable for grids about fighter aircraft configurations with lifting surfaces that have sharp leading and trailing edges. It is also desirable to concentrate grid points about the edges. Eriksson [11] has proposed a **dual-block** topology for fighter airplanes with highly-swept cranked wings. After generating the surface grid, custom software, based on transfinite interpolation and Lagrangian interpolation functions, has been written by Eriksson for the configuration in Figure 6. Grid surfaces about this configuration are shown in Figure 10. More information about the grid generation and incompressible flow about the configuration can be found in References 11 and 26. The same topology has been been applied by Smith and Everton in an interactive environment to a modified F-18 configuration. In addition to Lagrangian interpolation with exponential controls, intermediate and far field boundaries are determined interactively. Figure 11 shows grid surfaces about the F-18 configuration, and more detail can be found in Reference 27.

Figure 10: Grid Surfaces About a Cranked-Wing Configuration

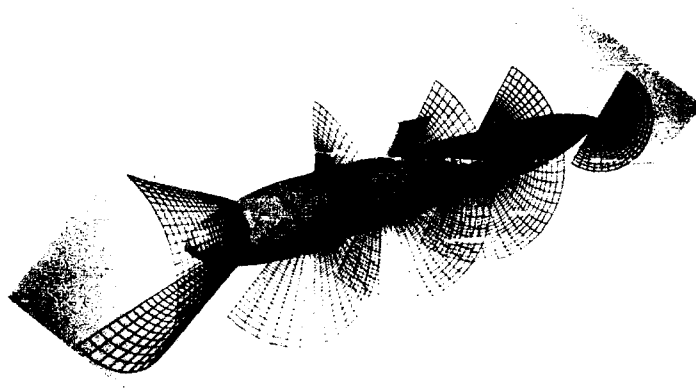Figure 11: Grid Surfaces About a Modified F-18 Configuration

Figure 12: Grid Surfaces About A Submarine Configuration

The last application is grid generation about submarines. This application is similar to the transport airplane configuration where the hull corresponds to the fuselage, and the sail corresponds to a shortened wing. In addition, there are sail planes and stern components in a submarine configuration. Abolhassani and Smith have proposed a topology for submarine configurations[28] and a sample grid is shown in Figure 12.

7 CONCLUSIONS

Algebraic grid generation is a very powerful and a flexible way of discretizing flow field domains. The concept of forming a boolean sum of univariate interpolations is the basic methodology underlying algebraic grid generation methods. Either structured grids or unstructured grids can be generated with algebraic methods. Algebraic techniques work well in conjunction with interactive computer graphics. The **CPF** method, in particular, is highly flexible and suitable for an interactive environment. Three-dimensional applications of algebraic grid generation require several steps, but algebraic techniques are capable of producing discretizations of virtually any domain, given enough blocks. The problem that arises and merits consideration is the trade-off for using many structured blocks, a single unstructured representation, or a combination of structured and unstructured blocks.

8 REFERENCES

1. SMITH, R. E. and ERIKSSON, L.-E. "Algebraic Grid Generation", *Comput. Meths. Appl. Mech. Engrg.* 64, pp 285-300, 1987.

2. WEATHERHILL, N. P. "The Combination of Structured- Unstructured Meshes", *Numerical Grid Generation in Computational Fluid Mechanics'88*, Pineridge Press,pp. 729-739, 1988.

3. SMITH, R. E., "Algebraic Grid Generation", *Numerical Grid Generation*, North Holland, pp. 137-176, 1982.

4. MASTIN, C. W., "Errors Induced by Coordinate Systems", *Numerical Grid Generation*, North Holland, pp. 31-40, 1982.

5. ERIKSSON, L.-E., *Transfinite Mesh Generation and Computer-Aided Analysis of Mesh Effects*, Ph. D. Dissertation, University of Uppsala, Sweden, 1984.

6. KATHONG, M., SMITH, R. E., and TIWARI, S. N. "A Conservative Approach for Flow Field Calculation on Multiple Grids", AIAA Paper 88-0224, 1988.

7. BENEK, J. A., BUNING, P.G., and STEGER, J. L., - "A 3-D Chimera Grid Embedding Technique", AIAA Paper 84-0164, 1984.

8. GORDON, W. N. and HALL, C. A., "Construction of Curvilinear Coordinate Systems and Application to Mesh Generation", *International Journal for Numerical Methods in Engineering*, Vol. 7 461-477, 1973.

9. ERIKSSON, L.-E., "Generation of Boundary Conforming Grids Around Wing-Body Configurations Using Transfinite Interpolation", *AIAA Journal*, Vol. 20, pp. 1313-1320, 1982.

10. SMITH, R. E. and WEIGEL, B. L. "Analytic and Approximate Boundary-Fitted Coordinate Systems for Fluid Flow Simulations", AIAA Paper 80-0192, 1980.

11. ERIKSSON, L.-E., "Flow Solution on a Dual-Block Grid Around an Airplane", *Comput. Meths. Appl. Mech. Engrg.* 64. pp. 79-93, 1987.

12. EISEMAN, P.R., "A Multi-Surface Method of Coordinate Generation" , *J. Comp. Phys.*, Vol. 33, No. 1, pp. 118-150, 1979.

13. EISEMAN, P. R., "Coordinate Generation with Precise Control Over Mesh Properties", *J. Comp. Phys.*, Vol. 47, No. 3. pp. 331-351, 1982

14. EISEMAN, P. R.,"High Level Continuity for Coordinate Generation with Precise Controls", *J. Comp. Phys.*, Vol. 47, No. 3. pp. 352-374, 1982.

15 EISEMAN, P.R., " Grid Generation for Fluid Mechanics Computations", *Annual Review of Fluid Mechanics*, Vol. 17. pp. 487-522, 1985.

16. BÉZIER, P., *Numerical Control: Mathematics and Applications*, John Wiley and Son, 1970.

17. ROGERS, D.F. and ADAMS, J.A., *Mathematical Elements for Computer Graphics*, McGraw-Hill, 1976.

18. EISEMAN, P. R , "A Control Point Form of Algebraic Grid Generation",*Intl. J. Numerical Methods in Fluids*, Vol. 8. pp. 1165-1181, 1988.

19. SMITH, R. E. and WIESE, M. R., "Interactive Algebraic Grid Generation Technique", NASA TP 2533, 1986.

20. CHOO, Y., EISEMAN, P.R. and RENO, C., "Interactive Grid Generation for Turbomachinery Flow Field Simulations", *Numerical Grid Generation in Computational Fluid Mechanics '88*, Pineridge Press. pp. 895-964, 1988.

21. FELDMAN, L. A. and RAPAGNANI, N. L., "Fast Interactive Graphics and the Numerical Electronic-Wind Tunnel (FIG-NEWTUN)", AFWL-TN-87-33, 1987.

22. SMITH, R. E., KUDLINSKI, R. A., EVERTON, E. L., and WIESE, M. R. "Algebraic Grid Generation About Wing-Fuselage Bodies",*J. Aircraft*, 24 (12) pp. 868-872, 1987.

23. CRAIDON, C. B., "A Computer Program for Fitting Smooth Surfaces to an Aircraft Configuration and Other Three-Dimensional Geometries", NASA TM X-3206, June 1975.

24. THOMPSON, J. F. - *Program EAGLE Numerical Grid Generation System User's Manual*, Vol. II and III, AFATL-TR-87-15, 1987.

25. THOMPSON, J. F., WARSI, Z.U.A. and MASTIN, C.W., *Numerical Grid Generation-Foundations and Applications*. North-Holland, 1985.

26. ERIKSSON, L.-E., SMITH, R. E., WIESE, M.R., and FARR, N., "Grid Generation and Inviscid Flow Computation About Cranked- Winged Airplane Geometries", *J. Aircraft*, Vol. 25, No. 9 pp. 820-826, 1988.

27. SMITH, R. E., and EVERTON, E. L., "Interactive Grid Generation for Fighter Aircraft Configurations", *Numerical Grid Generation in Computational Fluid Mechanics '88*, Pineridge Press, pp. 805-814 1988.

28. ABOLHASSANI, J. S. and SMITH, R. E., " Three-Dimensional Grid Generation About a Submarine", *Numerical Grid Generation in Computational Fluid Mechanics '88*, Pineridge Press, pp. 505-515, 1988.

GEOMETRIE ET MAILLAGE DE CONFIGURATIONS COMPLEXES POUR LES CALCULS AERODYNAMIQUES

par :

Gilbert Ranoux, Jérôme Lordon et Jean Diet

----

aerospatiale - Division Engins Tactiques
2, rue Béranger - 92320 Chatillon - France

RESUME

Cet article décrit les méthodes utilisées à **aerospatiale** Division Engins Tactiques pour calculer numériquement l'aérodynamique des missiles dans un cadre industriel. L'accent est mis principalement sur les problèmes de liaison entre géométrie et maillage. Après quelques considérations sur la Conception Assistée par Ordinateur (CAO) et son emploi dans une procédure de maillage, deux chaînes de calcul sont présentées. La première, basée sur le principe de la marche en espace, est destinée aux calculs supersoniques et la seconde propose une approche multidomaine pour les calculs subsoniques et transsoniques. Des exemples de maillage et des résultats de calcul pour diverses configurations sont produits pour illustrer les deux procédures.

ABSTRACT

This paper describes the methods which are used at **aerospatiale** Tactical Missiles Division in order to calculate missile aerodynamics numerically in an industrial way. The problems in linking geometry and mesh are stressed. After a few comments about Computer Aided Design (CAD) and its use in a mesh generation scheme, two procedures are presented : the first one, based on the space marching principle, is designed for supersonic flows and the second one proposes a multizonal approach for transonic and subsonic flows. Mesh examples and flow results are given for different types of configurations in order to illustrate the two procedures.

1 INTRODUCTION

L'aérodynamique des missiles tactiques est fortement tridimensionnelle et essentiellement non linéaire : la géométrie d'un missile étant très ramassée autour de son axe longitudinal, il se produit de fortes interactions entre les différents éléments, et des décollements importants et de nombreuses ondes de choc peuvent apparaître (Référence 1). De tels écoulements sont restés longtemps hors de portée des méthodes numériques si bien qu'elles sont apparues plus tardivement chez les missiliers que chez les avionneurs. En outre, comme les formes des missiles sont restées simples très longtemps (axisymétriques et cruciformes) et leur structure restée rustique (peu d'optimisation), les méthodes semi-empiriques alliées aux essais en soufflerie se sont avérées suffisantes pour répondre aux besoins des projets.

La nécessité de méthodes numériques n'est vraiment apparue chez les missiliers qu'au début de la décennie pour prédire l'aérodynamique globale des formes compliquées (missiles aérobies, par exemple) et l'aérodynamique répartie, inaccessibles par les méthodes semi-empiriques. Pour des considérations de coût de calcul, ce sont d'abord des méthodes de singularités qui ont été utilisées, mais la très forte non-linéarité de l'aérodynamique des missiles limite par trop leur domaine de validité. C'est grâce d'une part aux progrès accomplis dans le domaine des super-ordinateurs, et d'autre part au développement de la technique d'accélération de convergence par marche en espace que les codes Euler sont devenus accessibles industriellement : ainsi la procédure développée autour du code Euler FLU3C décrite au paragraphe 3 nous permet actuellement d'intervenir rapidement à tous les stades du projet pour le domaine supersonique. En ce qui concerne le subsonique et le transsonique, une procédure multidomaine répondant aux mêmes exigences de performance est en phase finale de développement. Elle fait l'objet du paragraphe 4.

2 CAO ET CALCUL AERODYNAMIQUE DES MISSILES

La géométrie des missiles s'est énormément compliquée depuis quelques années, principalement pour des considérations de propulsion (entrées d'air sur l'ASMP ou l'ANS: figure 1) et de furtivité (lissage des formes). Dans le même temps, la précision des méthodes numériques de dynamique des fluides s'est suffisamment accrue pour permettre la prise en compte de détails de géométrie ignorés jusqu'alors (gouttières, capots...). Cette double évolution a rendu indispensable l'emploi de systèmes de CAO non seulement pour assurer la définition rigoureuse de la géométrie, mais aussi pour faciliter son maillage pour des calculs numériques. Aussi, l'aéronuméricien se doit aujourd'hui de posséder une compétence CAO.

En effet, la plupart des géométries que nous avons à calculer sont définies et modifiées par des spécialistes de la CAO et sont destinées à être exploitées dans les domaines les plus divers. Elles ne sont donc soumises à aucun critère restrictif d'élaboration qui pourrait nous en faciliter le maillage. Par exemple, les différents éléments d'une configuration sont le plus souvent juxtaposés sans souci de représentation des intersections, certaines surfaces de définition peuvent se superposer en tout ou partie et même être légèrement disjointes. Ce sont autant d'imperfections que nous devons en premier lieu éliminer. D'autre part, nous devons structurer la configuration à mailler, c'est à dire construire les frontières des domaines de calcul, car en général elles ne coïncident pas avec les frontières des surfaces de définition géométrique. Enfin, nous avons très souvent à modifier la configuration à traiter, en particulier pour des séries de calculs paramétriques de positionnement et dimensionnement de surfaces portantes ou encore pour l'estimation d'efficacités de gouvernes.

Tous ces éléments nous ont montré la nécessité de disposer d'un logiciel de géométrie performant, même si nous n'assurons pas la tâche de conception dans son intégralité. Plutôt que de développer un logiciel aux fonctionnalités CAO, nous avons préféré choisir un système déjà existant et éprouvé et l'interfacer en fonction de nos tâches spécifiques de manière à ne pas transformer l'aérodynamicien en spécialiste CAO malgré lui. Le logiciel ICEM2 de Control Data Corporation sur station de travail IRIS de Silicon Graphics nous a paru répondre à cette exigence : d'abord parce qu'il est extrêmement convivial et facile d'emploi pour des utilisateurs qui ne sont qu'occasionnels et qui en ont un emploi marginal, et ensuite parce qu'il intègre un langage de programmation graphique très puissant (Graphics Programming Language) permettant d'accéder directement la base de données et d'utiliser la plupart des fonctionnalités mathématiques du système. L'écriture de modules en GPL nous a permis d'automatiser le traitement des géométries les plus complexes que nous ayons eues à calculer, comme nous le verrons par la suite.

Parmi les nombreuses fonctionnalités et la souplesse d'emploi offertes par cette solution CAO, il faut souligner la possibilité de traiter les bases de données géométriques provenant des autres systèmes par l'intermédiaire de la base de données normalisée SET (Système d'Echange et de Transfert) via une interface de traduction. De cette manière, nous pouvons traiter les fichiers géométriques provenant de tous les logiciels de CAO (Catia, Cadds, Aérolis...), pourvu qu'ils soient interfacés avec SET.

3 PROCEDURE POUR LA MARCHE EN ESPACE

Lorsque l'écoulement est supersonique dans une direction donnée, les équations d'Euler sont hyperboliques dans cette direction et autorisent une résolution de proche en proche. Cette méthode dite de marche en espace s'applique naturellement aux équations d'Euler stationnaires mais aussi aux équations d'Euler instationnaires. Le maillage est alors généralement constitué d'une succession de coupes planes (2D) orthogonales à la direction de la marche.

La procédure de maillage que nous avons développée s'articule en deux phases bien distinctes (voir l'organigramme de la chaîne de calcul en figure 2) :
- si la géométrie est analytique (par exemple un missile axisymétrique et cruciforme), elle est générée à l'aide du module géométrique d'un logiciel interactif appelé PRECET, et maillée à l'aide du module de maillage du même logiciel qui est organisé autour d'un mailleur bidimensionnel elliptique dérivé de GRAPE (Référence 2);
- si la géométrie est une base de données CAO (par exemple un missile aérobie), elle est d'abord conditionnée au sein du système de CAO par exécution d'une chaîne de programmes GPL interactifs et batch. Le maillage est ensuite réalisé à l'aide du module de maillage du logiciel PRECET.

3.1 Définition de la géométrie
Définition analytique
Le préprocesseur pour le calcul des missiles PRECET (PREprocesseur pour Calcul d'Engins Tactiques), programme graphique interactif écrit en FORTRAN, possède un module qui permet de construire très rapidement une géométrie simple (analytique) en combinant un certain nombre d'éléments (ogives, cylindres, surfaces portantes avec ou sans épaisseur, rétreints...) définis de façon paramétrique. Ceci s'applique très facilement aux missiles de forme classique tel que l' ASTER (Figure 6).

Définition via la CAO
Comme il a été dit précédemment, la configuration à mailler n'est pas définie sous CAO spécialement à notre intention. Ainsi, même si la géométrie est "propre", on ne peut généralement pas utiliser telles quelles les surfaces de définition car elles ne correspondent pas à celles que l'on souhaite mailler (Figures 3a et 3b). La plupart du temps elles ne présentent pas de frontière commune même en les regroupant. La première étape consiste donc à définir les surfaces à mailler avant même de vouloir récupérer la définition géométrique. Or, dans le cadre de la marche en espace où il s'agit de mailler l'espace par une succession de plans perpendiculaires à l'axe de la configuration, les surfaces à mailler présentent deux frontières perpendiculaires à cet axe. Nous avons donc choisi de tirer profit de cette caractéristique en modélisant la géométrie par une série de coupes dont certaines correspondent aux frontières des surfaces à mailler. Le problème de structuration de la géométrie s'en trouve ainsi considérablement simplifié.

Le traitement de la géométrie se déroule comme suit :

- redéfinition de la configuration par une série de coupes perpendiculaires à son axe directeur : l'utilisateur doit demander un nombre suffisant de coupes pour avoir une définition précise de la géométrie dans sa direction longitudinale et doit spécifier les positions des coupes particulières correspondant aux frontières des surfaces à mailler. Ces coupes limites sont choisies généralement pour marquer un changement de topologie de la configuration ou pour autoriser de part et d'autre de celles-ci des maillages de types différents ou encore pour marquer une discontinuité longitudinale. Une fois les plans de coupes définis, un programme calcule les intersections avec les surfaces de définition (Figure 3c); .

- "nettoyage" et préparation des coupes : l'utilisateur spécifie s'il veut traiter la configuration entière, la moitié ou encore le quart, et le programme modifie les entités en conséquence (suppression, coupure ou extension) en éliminant au passage toutes les entités parasites générées lors des coupes (en général des splines cubiques). Il visualise par des points la trace sur les coupes des frontières des surfaces de définition;

- choix des frontières des surfaces à mailler : l'utilisateur doit d'abord désigner les coupes définissant les frontières transversales (limites des blocs), puis pour chacun des blocs ainsi définis désigner les frontières longitudinales (Figure 3d). Pour ce faire il lui suffit de sélectionner avec la souris de sa station de travail certains des points visualisés lors de l'étape précédente. Le choix des frontières longitudinales correspond le plus souvent à une discontinuité qui doit être absolument respectée lors du maillage final mais peut être aussi une simple ligne de maillage que l'utilisateur veut imposer pour, par exemple, pouvoir y effectuer un raffinement local. La qualité du maillage final dépend en majeure partie de cette étape;

- enregistrement de la géométrie des surfaces à mailler : muni des informations que l'utilisateur lui a données précédemment, le programme ordonne automatiquement les entités de chacune des coupes, y répartit un nombre de points fixé par un critère de densité donné et les stocke de manière ordonnée. On dispose ainsi à l'issue de cette étape d'un fichier contenant les surfaces à mailler de la configuration sous la forme d'une grille structurée de points de définition. Cette grille sera interpolée par splines cubiques lors du maillage (Figure 3e). En général, l'interpolation est suffisamment précise pour pouvoir se contenter des points paroi obtenus de cette façon. Cependant, l'utilisateur dispose d'un programme lui permettant en dernier lieu de projeter automatiquement tous les points paroi sur les surfaces géométriques initiales. Ainsi, la géométrie est rigoureusement respectée, même si elle a été maillée à partir d'une représentation approchée.

Dans le cas ou une coupe a donné lieu à plusieurs contours disjoints, l'utilisateur doit les relier les uns aux autres de manière à n'en obtenir qu'un seul, comme le réclame le mailleur. Ceci revient à créer des surfaces à mailler artificielles dites de "transparence" qui seront traversées par l'écoulement lors du calcul comme si elles n'existaient pas (Figure 4). Il faut procéder de même lorsque la configuration présente une partie émoussée et donc un écoulement localement subsonique : pour que le premier plan du domaine qui la contient soit supersonique, il est nécessaire de l'avancer et pour ce faire de construire une surface artificielle qui reproduise la topologie de l'élément.

3.2 Génération du maillage

Préprocesseur

Le préprocesseur PRECET permet de définir un maillage tridimensionnel à partir de maillages bidimensionnels par plans. La géométrie est définie analytiquement de façon interne ou bien à partir d'un fichier de points résultants de la démarche CAO décrite ci avant. L'utilisateur définit d'abord les abscisses limites des domaines de calcul ainsi que le nombre et la répartition des plans de calcul. Ensuite, pour chaque domaine, il définit le maillage transversal en fixant quelques paramètres (nombres de points radiaux et orthoradiaux...) qui seront valables pour tout le domaine; il peut visualiser le maillage résultant dans n'importe quel plan de calcul. La frontière extérieure qui doit englober le choc frontal est fixée de manière empirique en fonction de l'écoulement amont. PRECET écrit des fichiers de type maillage, des fichiers de données pour les calculs FLU3C et un fichier de cartes de contrôle que l'utilisateur n'a plus qu'à soumettre au super-ordinateur.

Mailleur

La méthode de génération de maillage repose sur la résolution des équations de Poisson. Les fonctions de contrôle sont déterminées automatiquement, selon le schéma mis au point par Steger et Sorenson (Référence 2), par spécification de l'espacement et de l'orientation des mailles au voisinage des frontières. Cette méthode a été choisie pour sa capacité à traiter des domaines de formes variées tout en produisant des lignes de coordonnées assez lisses. Les équations, une fois discrétisées, sont résolues par un algorithme SLOR appliqué d'abord sur une grille grossière, où l'on ne prend qu'un point sur trois, puis sur la grille complète. De cette façon le temps de calcul est divisé par 5, en moyenne. Egalement dans le but d'améliorer les performances on a ajouté une option qui opère à partir d'une solution calculée pour le plan précédent dans le cas où les deux domaines sont géométriquement très proches. Le gain obtenu se situe entre 3 et 10.

En général, la configuration étudiée présente un plan vertical de symétrie, de sorte que le domaine à mailler est topologiquement équivalent à un rectangle dont les quatre frontières sont : la frontière intérieure constituée d'une demi section du missile, la frontière extérieure englobant la trace du choc et les traces du plan de symétrie à l'intrados et à l'extrados du missile. Pour les configurations qui n'admettent pas de plan de symétrie, le domaine est topologiquement équivalent à une couronne et on choisit l'option "maillage en O".

Le programme a été adapté pour prendre en compte automatiquement les points anguleux (extrémité de voilure...) et les points confondus (apex d'une aile en flèche). Les données à fournir sont réduites au minimum : nombre de points de maillage dans les deux directions, coordonnées (x,y) des nœuds sur les frontières intérieures et

extérieures et indication de l'espacement désiré au voisinage des frontières. Implicitement, le programme essaye de générer des mailles orthogonales aux frontières. Sur les frontières latérales (plan de symétrie) on peut au choix imposer les nœuds (conditions de Dirichlet) ou laisser les points "flotter" (conditions de Neumann) pour satisfaire une condition d'orthogonalité.

On trouvera en figure 5 un exemple de maillage transversal pour le missile aérobie ASMP.

3.3 Calculs aérodynamiques : code FLU3C

FLU3C est un programme de résolution des équations d'Euler tridimensionnelles instationnaires sous forme conservative qui résulte d'une coopération étroite entre l'ONERA et **aerospatiale** (Référence 3). Le schéma de type volumes finis est explicite et donc soumis à un critère de CFL en temps. La formulation décentrée du calcul des flux, suivant l'approche de Van Leer, assure à la méthode une grande robustesse. La précision d'ordre deux en temps et en espace est assurée par l'utilisation d'un schéma prédicteur-correcteur de type MUSCL. Un limiteur de pentes, qui agit uniquement dans les régions de fort gradients, prévient les oscillations en réduisant l'ordre de précision spatiale près des discontinuités. Pour la recherche de solutions stationnaires, deux méthodes d'accélération de convergence sont utilisées :

- la méthode de pas de temps local; chaque point évolue avec sa propre échelle de temps jusqu'à convergence,
- la méthode de pseudo marche en espace; lorsque la composante de la vitesse suivant une direction privilégiée est supersonique dans tout le domaine de calcul, on fait converger en temps, l'un après l'autre, chaque plan de calcul orthogonal à cette direction en utilisant uniquement les variables situées à l'amont. Cette procédure minimise le nombre d'itérations nécessaires ainsi que le volume d'entrées-sorties.

Le coût de cette méthode explicite restreint son utilisation dans un cadre industriel aux écoulements supersoniques. Le calcul d'un missile est en fait constitué de plusieurs calculs FLU3C (le dernier plan de calcul N sert de plan initial pour le calcul du plan N+1). Une condition nécessaire est que les plans d'entrée et de sortie de chaque calcul FLU3C soient supersoniques. Cependant, cela n'interdit pas la présence de poches subsoniques comme c'est le cas dans les calculs de missiles avec jets latéraux (Référence 4). Le calcul d'une configuration réaliste en écoulement supersonique avec un maillage comprenant 300 000 points est d'environ 20 mn CPU sur le CRAY X-MP.

3.4 Exemples
Missile ASTER 30
La figure 6 montre un exemple de maillage pour l'anti-missiles ASTER 30. Le maillage surfacique comprend environ 14 000 nœuds pour une demi-configuration et 400 000 nœuds au total. La géométrie de ce missile classique est définie de façon analytique avec PRECET et tient compte du profil des voilures, ce qui est indispensable pour la bonne prédiction des moments de charnière (Référence 5). La figure 6 montre également une répartition de nombre de Mach pariétaux pour un nombre de Mach infini de 2,6 et une incidence de 10 degrés, obtenus avec le code FLU3C. Les chocs et détentes liés aux surfaces portantes sont clairement visibles. Le programme FLU3C a été utilisé intensivement dans le cadre du projet de missile ASTER.

Missile ANS
L' ANS est un projet de missile aérobie étudié conjointement avec MBB pour succéder à la famille de missiles anti-navires EXOCET. Il est équipé d'un stato-réacteur avec 4 entrées d'air de révolution. Le maillage surfacique présenté figure 7 comprend environ 16 000 nœuds pour une demi-configuration. La géométrie a été définie sous CAO. Le résultat de calcul (coefficients de pression pariétaux) correspond à Mach 2 et 4 degrés d'incidence. Les entrées d'air sont ouvertes et supposées en fonctionnement supercritique.

Navette HERMES sur lanceur ARIANE 5 [*]
La navette spatiale HERMES sur le lanceur ARIANE 5 est l'une des plus grosses configurations que nous ayons traitées (Figure 8). Elle comprend environ 17 000 nœuds sur la paroi pour une demi-configuration et illustre bien l'utilisation des surfaces transparentes (jonction des boosters et du corps central). Le nez de la navette est maillé avec une singularité d'axe alors que le reste de la configuration est maillé par plan. La figure 8 présente des nombres de Mach pariétaux à Mach 1,5 et 3 degrés d'incidence.

Avion de transport ATSF [**]
L'ATSF est un projet d'avion supersonique civil destiné à succéder à CONCORDE. La géométrie est plus simple que dans les exemples précédents. Le maillage surfacique présenté figure 9 comprend 11 000 nœuds environ. La figure 9 donne également la répartition des nombres de Mach sur la paroi à Mach 2 et 4 degrés d'incidence.

4 PROCEDURE POUR LE MULTIDOMAINE

La procédure marche en espace présente certaines limitations, tant au niveau géométrie-maillage qu'au niveau du calcul. Bien que des configurations très diverses aient pu être maillées et calculées aisément, elles restent cependant limitées en complexité. De plus, quelle que soit leur topologie, toutes les configurations sont maillées en "O-H", alors que certaines parties comme par exemple un bord d'attaque arrondi nécessiteraient un maillage de

type "O-C". Cela peut nuire à la bonne représentation géométrique de l'élément ou conduire à un nombre de points de calcul trop élevé. En outre, certains contours ne peuvent pas être maillés sans recourir à une simplification de la géométrie. Enfin, la convergence d'un calcul transsonique sur un tel maillage n'est pas assurée, en particulier au niveau des raccords entre sous-domaines (non correspondance des nœuds). C'est pour cette raison qu'il nous est apparu nécessaire de développer une procédure tridimensionnelle générale. Nous avons choisi une approche multidomaine sans recouvrement avec possibilité de continuité des lignes de maillage entre domaines pour deux raisons :
- le non recouvrement facilite la gestion du multidomaine,
- la continuité des lignes de maillage facilite le traitement conservatif des raccords.

Cette approche permet en outre, dans sa restriction surfacique, d'obtenir quasi-automatiquement un maillage rigoureusement jointif pour une méthode de singularités.

4.1 Formes simples

Pour traiter les configurations de forme simple (axisymétrique, cruciforme), nous avons développé un préprocesseur FORTRAN appelé PREMICE (PREprocesseur de Maillage Interactif pour Calcul d'Engins) permettant la définition et le maillage interactifs de la géométrie. Celle-ci est définie à partir d'une bibliothèque de formes simples (ogives, troncs de cônes, ailes de différents types...). Le domaine de calcul est décomposé automatiquement en blocs topologiques qui sont ensuite réunis en sous-domaines. Le maillage volumique est effectué à l'aide du mailleur bidimensionnel présenté dans le paragraphe 3.2 par rotation de plans de maillage autour de l'axe de la configuration. Les effets tridimensionnels (tuilage ou encore épaisseur des ailes) sont obtenus par déformation des plans de maillage. Un exemple de maillage multidomaine du missile ASTER est produit en figure 10.

4.2 Formes définies par CAO
Définition de la géométrie
La démarche adoptée ici est sensiblement la même que pour la marche en espace en ce sens qu'elle s'articule en deux parties distinctes, l'une relative au traitement de la géométrie est effectuée au sein de ICEM, l'autre concernant le maillage est effectuée hors de ICEM. Cependant, le traitement de la géométrie requiert cette fois-ci une connaissance plus poussée du système de CAO, puisque l'aérodynamicien doit construire lui-même les frontières des sous-domaines.

La première tâche à accomplir est la décomposition du domaine physique en un certain nombre de sous-domaines faciles à mailler; la plupart des auteurs préconise d'effectuer une partition du domaine en blocs topologiques de type "éléments finis" (Références 6 et 7) : l'intersection de deux blocs ne doit être constituée que d'un sommet, d'une arête complète ou d'une face complète. Les blocs obtenus peuvent être facilement maillés de façon structurée, puis réunis pour former les sous-domaines de calcul. Cette méthode permet une gestion logicielle très simple de la connectique, mais est très contraignante car elle conduit à un grand nombre de blocs dès que le domaine à mailler devient complexe (souvent plus d'une centaine de blocs). L'approche que nous avons choisie est plus simple pour l'utilisateur : le domaine physique est directement décomposé en sous-domaines de calcul (de l'ordre d'une dizaine) qui sont choisis en fonction de la topologie rencontrée. Chacune des faces des sous-domaines est découpée en un ensemble de fenêtres correspondant en général au type de condition limite à y appliquer lors du calcul, ceci de telle manière que les sous-domaines de calcul communiquent entre eux par une fenêtre, un bord de fenêtre ou un coin de fenêtre. L'intérêt de cette approche vient du fait que la structure d'une face de sous-domaine n'a pas besoin d'être reproduite sur sa face opposée, ce qui évite une inflation de blocs. La configuration reste ainsi très lisible au niveau géométrique comme au niveau connectique (Figures 11a et 11b).

La décomposition du domaine en sous-domaines de calcul et des faces en fenêtres est faite par l'utilisateur sous CAO et requiert une certaine connaissance des fonctionnalités du système. L'emploi d'un système de CAO s'avère ici indispensable par comparaison à l'écriture d'un logiciel spécifique qui n'aurait pu en aucun cas offrir la même gamme de fonctionnalités.

Une fois effectuée la construction géométrique des arêtes des fenêtres, l'utilisateur est guidé dans sa démarche par un programme GPL auquel il doit donner les renseignements nécessaires pour que celui-ci retrouve et enregistre la connectique des sous-domaines et leur géométrie. Le déroulement des opérations est alors le suivant :
- désignation par l'utilisateur de toutes les fenêtres par sélection des entités géométriques les constituant (Figure 12a). Le programme visualise les fenêtres ainsi définies (Figure 12b);
- sur la modélisation par fenêtres, désignation des faces par sélection de fenêtres. Le programme représente graphiquement les faces ainsi définies (Figure 12c);
- sur la modélisation par faces, désignation des sous-domaines par sélection de faces;
- enregistrement d'un fichier de connectique : la connectique est assurée par l'intermédiaire des fenêtres : à chaque entité géométrique entrant dans la définition d'une fenêtre est associé un "label" unique. La comparaison de ces "labels" permet de savoir si deux fenêtres sont adjacentes ou pas, et donc de structurer les faces;
- enregistrement d'un fichier géométrique : la géométrie des fenêtres est enregistrée sous forme de séries de points répartis sur les entités constituant leurs arêtes. En outre, pour chacune des fenêtres paroi, une grille structurée de points intérieurs est générée à partir des surfaces de définition et leur répartition évolue en fonction de celle des arêtes correspondantes (Figure 13).

Génération du maillage

Au sein d'un domaine, il faut assurer la correspondance des mailles d'une fenêtre à l'autre pour obtenir un maillage structuré. De même, on peut vouloir assurer la continuité des mailles d'un domaine à l'autre. Pour cela, il suffit de d'assurer pour chaque fenêtre l'égalité des nombres de points sur deux arêtes opposées. On obtient ainsi un système d'équations linéaires dont la résolution par une méthode de pivot détermine le maillage des arêtes à partir de la donnée d'une densité de points et/ou d'un certain nombre de ces inconnues (le même procédé est utilisé sous CAO pour fixer les nombres de points du fichier géométrique). Le maillage des faces est obtenu par la réunion des maillages des fenêtres. Les points de maillage des fenêtres de paroi sont interpolés à l'aide d'une représentation par carreaux de Coons bicubiques et les autres fenêtres sont maillées par interpolations transfinies. Le maillage par fenêtre permet de bien respecter les discontinuités de pentes de la paroi qui sont généralement prises comme limites de fenêtres. Enfin, les domaines sont maillés par interpolations transfinies à partir des faces. Un exemple de maillage est présenté sur les figures 14 et 15 à partir de la décomposition multidomaine présentée sur les figures 11a et 11b.

Comme nous l'avons précisé en introduction, cette procédure est encore en développement. Ainsi, un certain nombre de fonctionnalités manquent pour l'instant :

- une fonction de répartition de points est associée à chaque entité géométrique pour permettre les raffinements de maillage, mais cette fonctionnalité n'est pas encore implantée,
- le processeur de maillage est encore rudimentaire,
- nous ne disposons pour l'instant d'aucune technique d'optimisation/adaptation de maillage : aussi nous nous orientons vers l'implantation d'un optimiseur basé sur la minimisation de la fonctionnelle présentée récemment par O. P. Jacquotte (Référence 8).

Nous envisageons également d'étudier une procédure d'aide à la décomposition en blocs de la géométrie pour la génération des sous-domaines.

4.3 Exemples

Pour illustrer la procédure multidomaine, deux exemples ont été évoqués au cours du chapitre précédent. Ils sont détaillés dans ce paragraphe :

- missile classique : la figure 10 présente le maillage multidomaine du missile ASTER ailes minces. Trois sous-domaines de calcul définissent le domaine total : un à l'extrados, un à l'intrados et un entre les surfaces portantes. Ils ont été générés avec PREMICE par rotation autour de l'axe de symétrie du plan de maillage visualisé sur la figure (dimensions : 127x34). Le maillage total comprend 150 000 nœuds;
- missile aérobie : la figure 11a présente la décomposition multidomaine d'un missile générique aérobie semblable à celui de la figure 3a. Le domaine de calcul a été découpé en 15 sous-domaines. La figure 11b donne une vue éclatée des 9 sous-domaines intérieurs dont 6 seulement possèdent une fenêtre de type paroi. Le maillage du sous-domaine n° 4 (extrados du fuselage arrière, dérive et empennage) est visualisé en figure 14 (dimensions : 17x21x69). Il s'agit là d'un maillage brut n'ayant encore fait l'objet d'aucune optimisation et généré directement à partir des données issues de la CAO. Le maillage surfacique (15 000 nœuds) et quelques plans de maillage sont présentés en figure 15.

5 CONCLUSION

Deux procédures de traitement de géométries et de maillage pour les calculs aérodynamiques ont été décrites. La première est orientée vers les calculs Euler avec une technique de marche en espace et privilégie une direction particulière pour définir la géométrie et mailler l'espace de calcul. Elle est très largement utilisée à **aerospatiale** Division Engins Tactiques dans un contexte industriel et son application à des géométries aussi diverses que celles des missiles aérobies ASMP et ANS, de la navette spatiale HERMES et de l'avion de transport supersonique ATSF prouve l'intérêt de cette approche. La seconde est une généralisation de la première avec une orientation calculs multidomaines. Les deux procédures sont intimement liées à l'utilisation par l'aéronuméricien d'un système CAO interactif, auquel nous avons ajouté des modules, pour définir les domaines de calcul et récupérer la géométrie sous une forme qui permette de générer un maillage de manière simple et interactive. Le temps nécessaire pour traiter une géométrie complexe et lancer le premier calcul est de un à deux jours pour la première procédure. Le temps prévu pour la seconde, qui est en cours d'achèvement, est d'environ une semaine. Ceci est rendu possible par la minimisation du nombre de structures a gérer par l'utilisateur et l'automatisation des principales tâches.

---

* : Calculs effectués par M. Mortel de la Division Spatiale dans le cadre du programme ARIANE 5 à la Division Engins Tactiques avec la procédure décrite ici.
** : Calculs effectués par M. Carlier de la Division Avions à la Division Engins Tactiques avec la procédure décrite ici.

6 REFERENCES

[1] R.G. Lacau : "A Survey of Missile Aerodynamics". Nielsen Missile Aerodynamics Conference - Monterey Ca, 1988.

[2] R.L. Sorenson : "A Computer Program to Generate Two-Dimensionnal Grids about Airfoil and Other Shapes by the Use of Poisson's Equations". NASA TM-81198,1980.

[3] M. Borrel, J.L. Montagné, J. Diet, P. Guillen, J. Lordon : "Upwind Scheme for Computing Supersonic Flows around a Tactical Missile". La Recherche Aérospatiale 1988 n° 2.

[4] M. Dormieux, C. Mahé : "Calculs Tridimensionnels de l' Interaction d'un Jet Latéral avec un Ecoulement Supersonique Externe". AGARD Conference - Lisbon 1988.

[5] P. Guillen, J. Lordon : "Numerical Simulation of Separated Supersonic Flows around Tactical Missile Bodies". AGARD Conference - Lisbon 1988.

[6] S.E. Allwright : "Techniques in Multiblock Domain Decomposition and Surface Grid Generation". Grid Generation in Computationnal Fluid Dynamics Conference - Miami 1988.

[7] J.W. Boerstoel : "Preliminary Design and Analysis of Procedures for the Numerical Generation of 3D Block-Structured Grids". NLR TR 86-102 U.

[8] O.P. Jacquotte, J. Cabello : "A Variationnal Method for the Optimization and Adaptation of Grids in Computational Fluid Dynamics". Grid Generation in Computationnal Fluid Dynamics Conference - Miami 1988.

ASMP : Missile Air-Sol Moyenne Portée

ANS : Missile Anti-Navire Supersonique
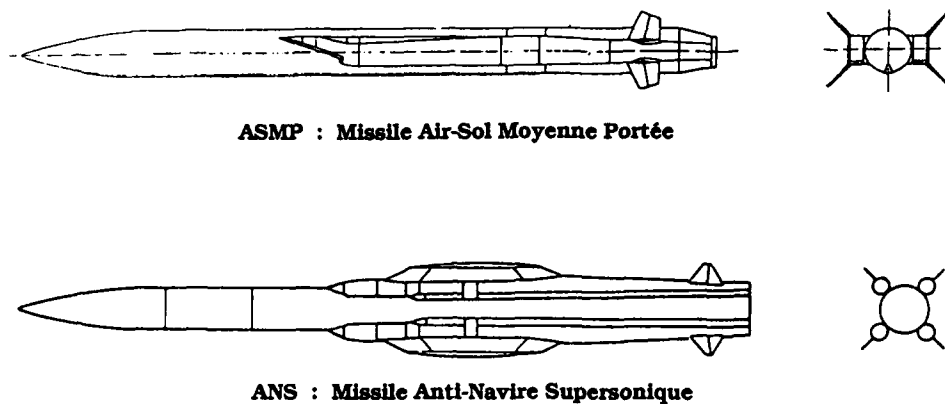
Fig. 1 : Missiles aérobies aerospatiale
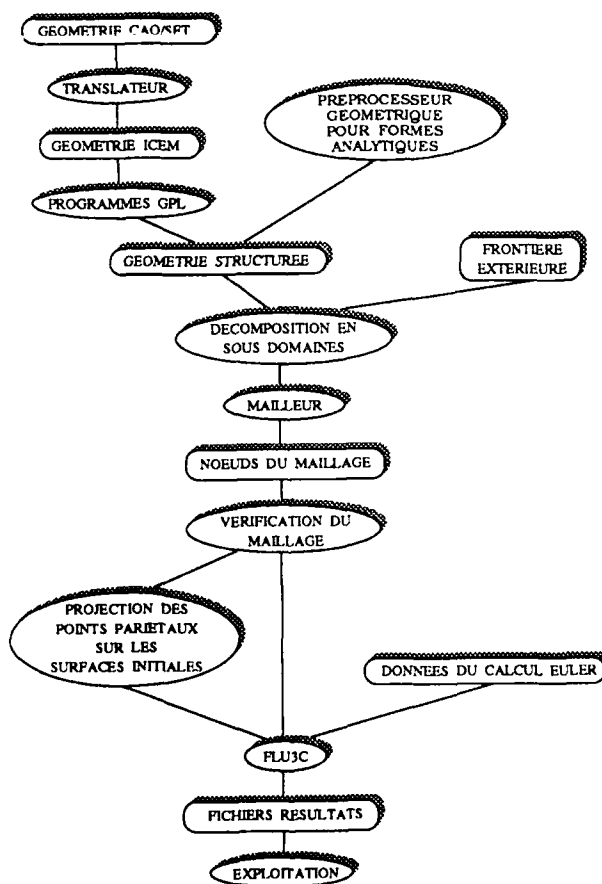
Fig. 2 : Chaîne de calcul pour la procédure de marche en espace
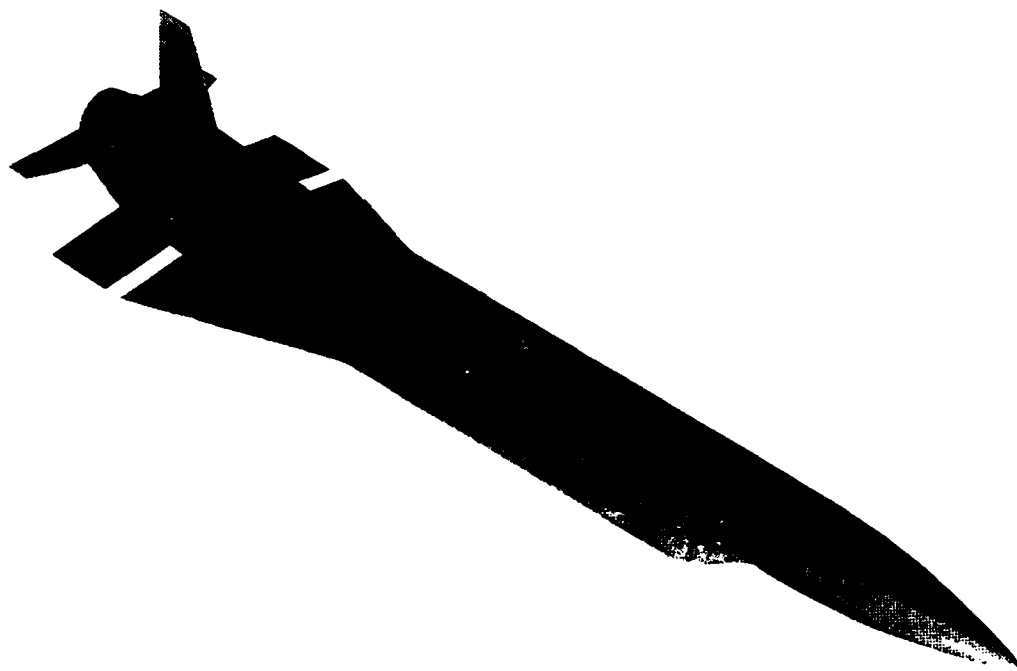
Fig. 3a : Missile générique aérobie
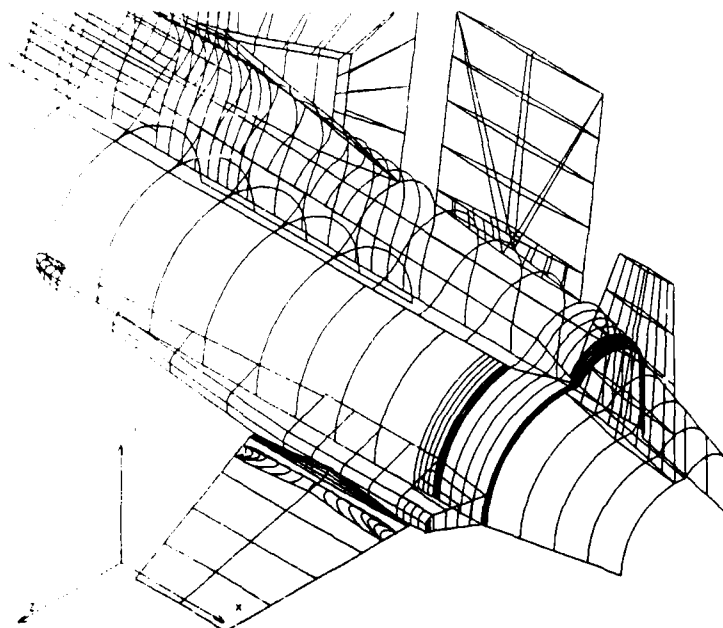
Fig. 3b : Géométrie surfacique CAO de la partie arrière
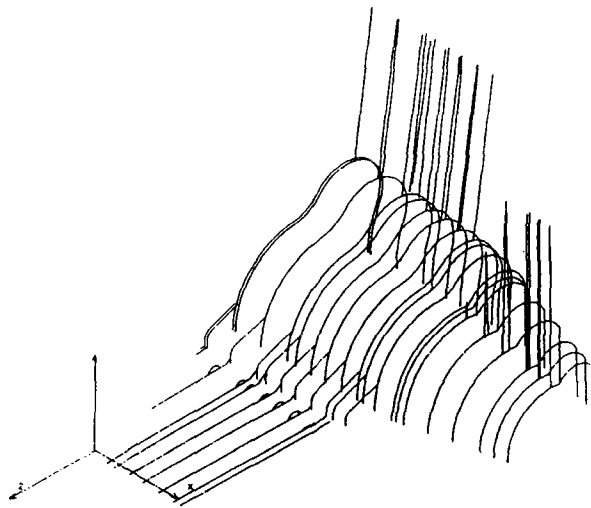
Fig. 3c : Définition minimale par coupes avant "nettoyage"
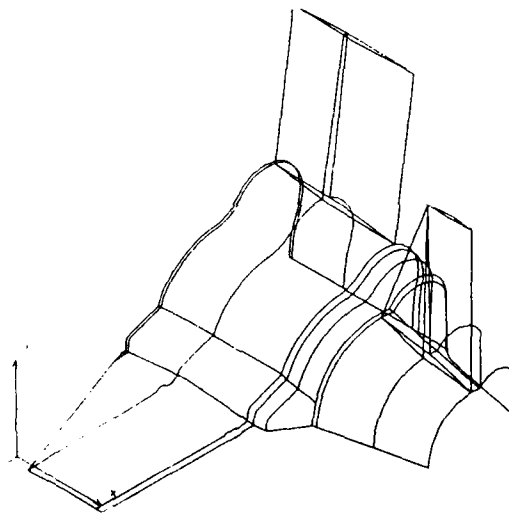
Fig. 3d : Frontières longitudinales et transversales des surfaces à mailler
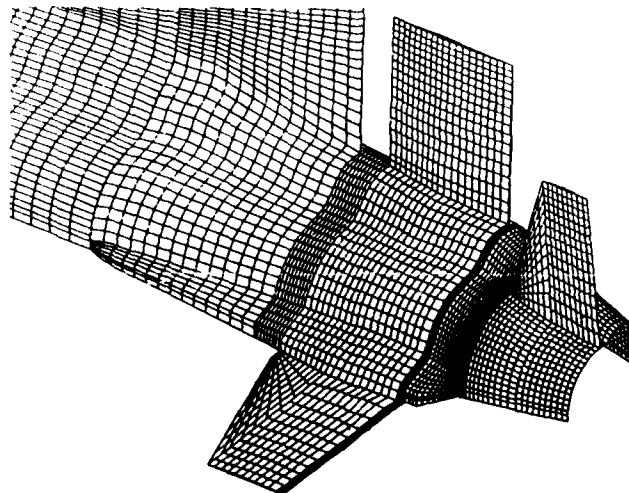
Fig. 3e : Trace surfacique du maillage
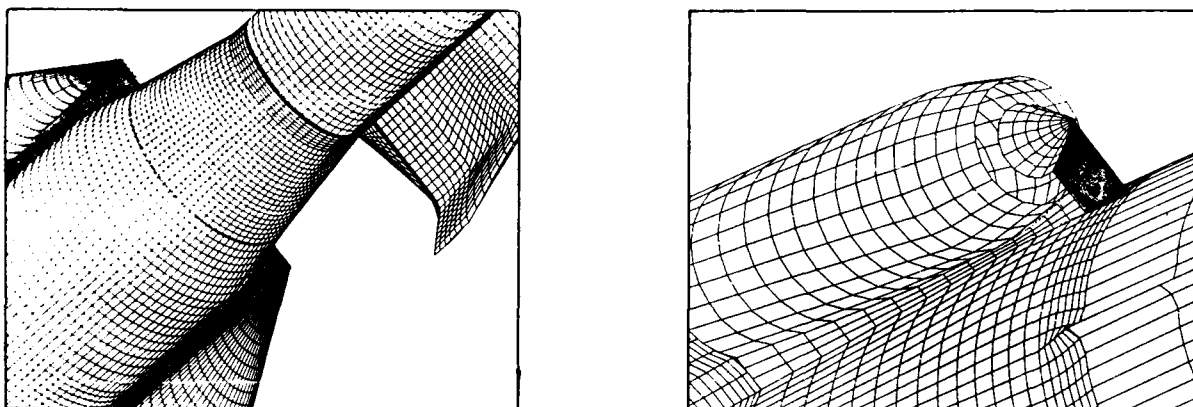
Fig. 4 : Exemples de surfaces transparentes
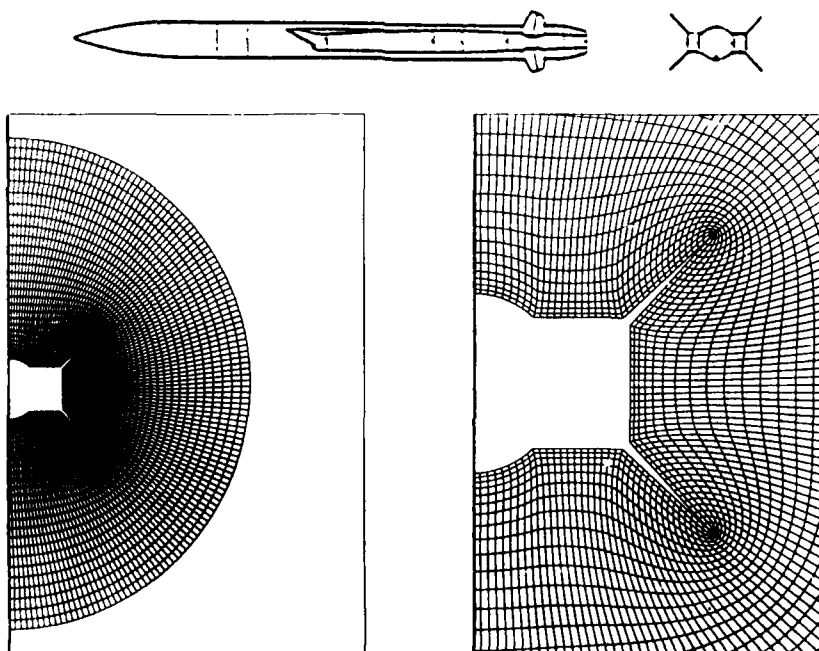
Fig. 5 : Exemple de maillage transversal - Missile ASMP

aerospatiale
DIVISION ENGINS TACTIQUES

Maillage et résultat de calcul (nombre Mach) surfaciques
- Missile ASTER 30 (Mach 2,6 - incidence 10 °) -
Fig. 6

aerospatiale
DIVISION ENGINS TACTIQUES

Maillage et résultat de calcul (coefficient de pression) surfaciques
- Missile ANS (Mach 2 - incidence 4 °) -
Fig. 7

Maillage et résultat de calcul (nombre Mach) surfaciques
- Ariane 5 + Navette Hermes (Mach 1,5 - incidence 3 °) -
Fig. 8

Maillage et résultat de calcul (nombre Mach) surfaciques
- Avion de transport supersonique (Mach 2 - incidence 4 °) -
Fig. 9

Fig. 10 : Maillage multidomaine d'un missile conventionnel - Missile ASTER
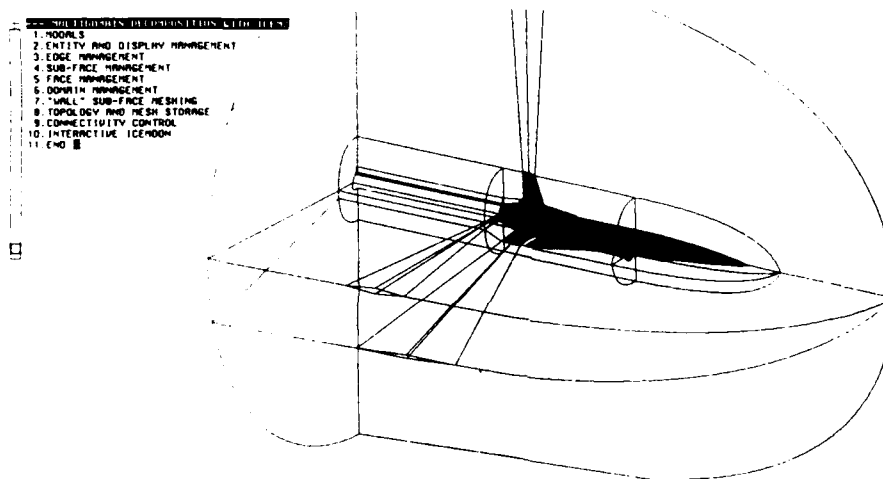
Fig. 11a : Décomposition multidomaine d'un missile non conventionnel - Missile générique aérobie
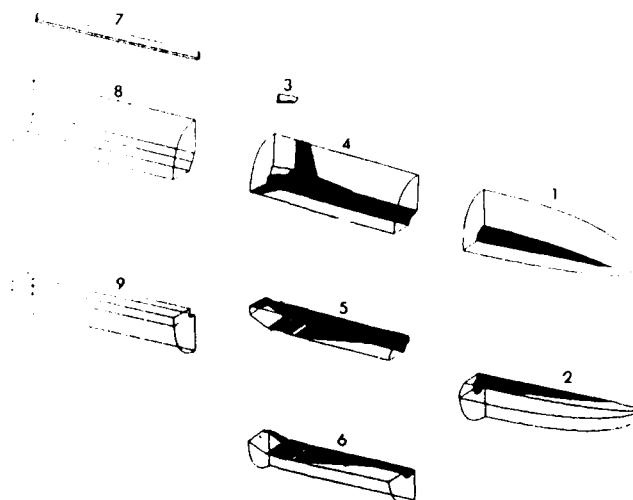
Fig. 11b : Vue éclatée des sous-domaines internes
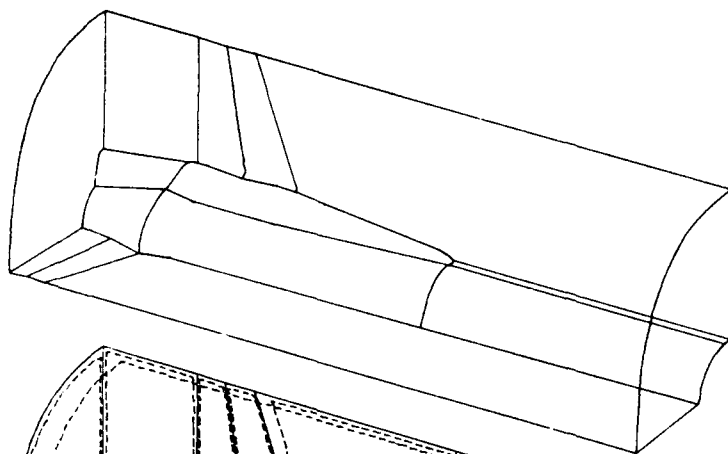
Fig. 12a : entités géométriques
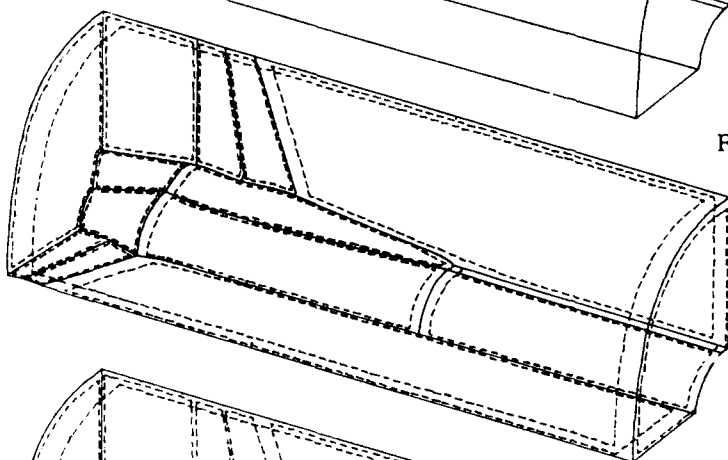
Fig. 12b : entités géométriques et sous-faces
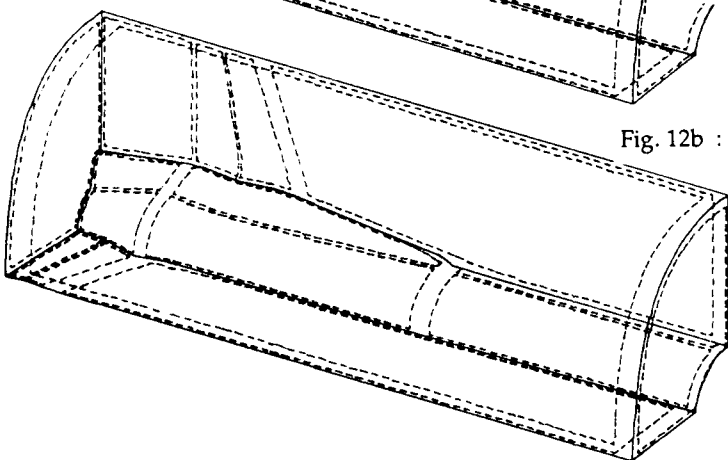
Fig. 12c : sous-faces et faces
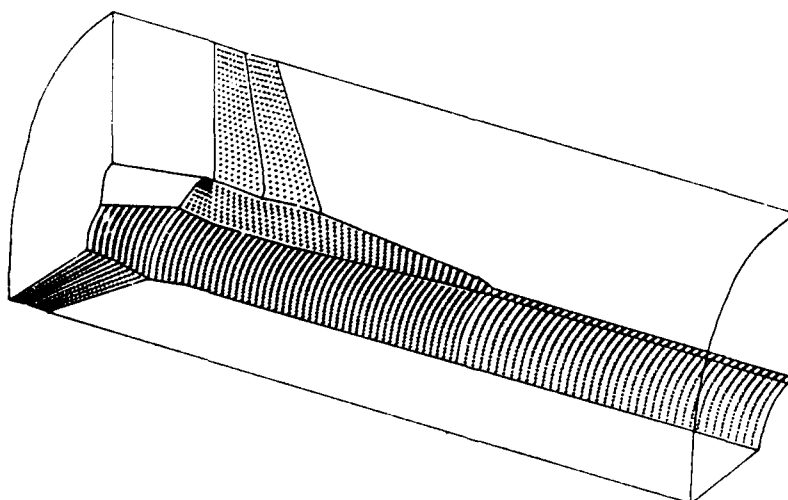
Fig. 13 : entités géométriques et points de définition surfacique

Bloc n°4
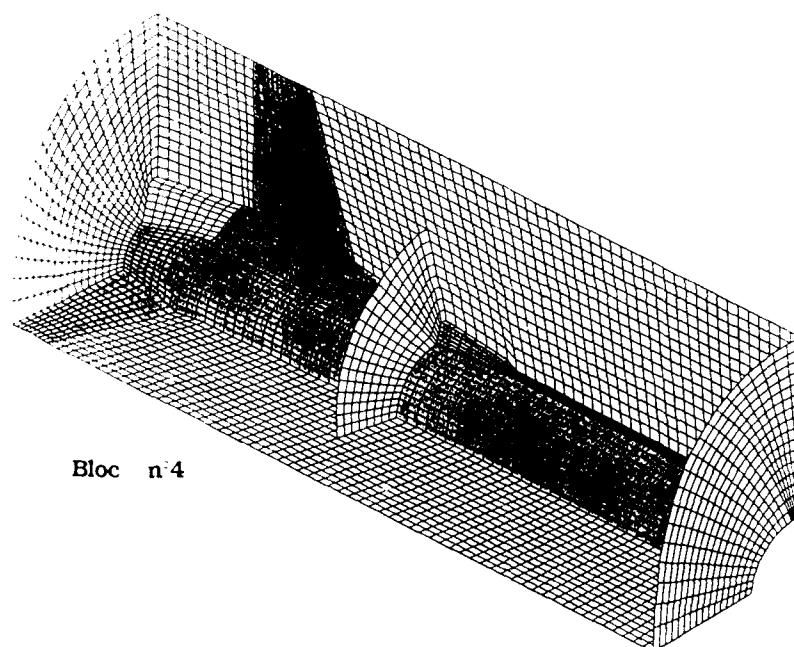
Fig. 14 : Maillage d'un domaine de calcul - Missile générique aérobie
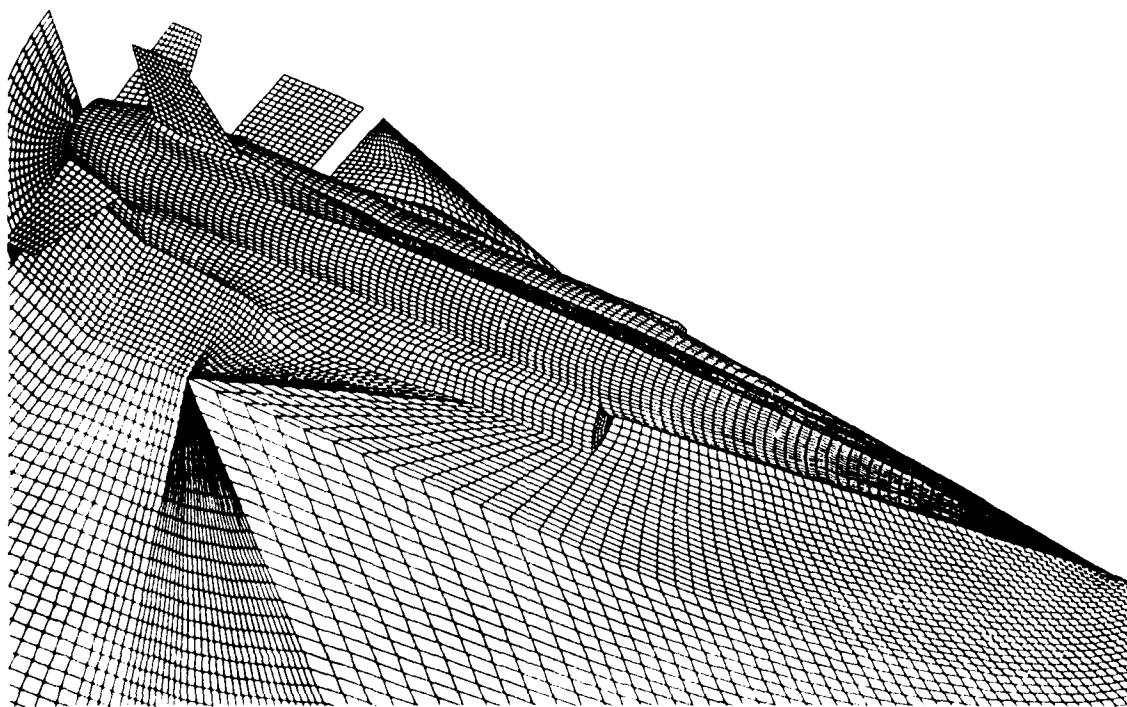
Fig. 15 : Surfaces du maillage multidomaine complet - Missile générique aérobie

MESH GENERATION FOR FLOW COMPUTATION IN TURBOMACHINE

by

M.Goutines and G.Karadimas
SNECMA — Villaroche
77550 Moissy Cramayel
France

C.Hah
General Electric Company
United States

ABSTRACT

This paper deals with building grids for flow computation in turbomachine applications. H,I,C, and O types are discussed for 2D or 3D, inviscid or viscous flow cases. The given examples concern 3D Euler application on a fan with part-span damper and splitter, 2D Navier-Stokes on turbine and compressor transonic cascades and 3D Navier-Stokes on a transonic fan

1 - INTRODUCTION

In recent years air flow computation in turbomachines have permitted more efficient fan and turbine designs [1]. In addition these computations reduce the number of time and cost consuming tests. All modern numerical codes use a grid as a space discretisation.

Grid generation is a primary part of obtaining numerical solutions to the two or three-dimensional inviscid or viscous flow inside turbomachinery. Computation of three-dimensional viscous flow inside cascade of airfoils with arbitrary endwalls or complex geometries such as dampers requires many additional constraints for grid generation. For the current study, only structured grids without any solution adaptation will be discussed. Both unstructured grids and grid adaptation provide additional ways to reduce overall grid size for a given physical problem, their application to three-dimensional viscous flows in a turbomachinery needs further development.

2 - PARTICULARITIES OF TURBOMACHINE MESH

Most of flow computation methods in turbomachine use cylindrical coordinates (R, radial coordinate ; Θ , tangential coordinate ; Z axial coordinate). Then the mesh is drawn on an axisymetric sheet for the two-dimensional case (Θ ; m, meridional coordinate). For the classical three-dimensional case, mesh is build by stacking several two-dimensional meshes. For complex geometries (such as non axisymetric hub, non axisymetric dampers, etc.) more efficient mesh generators are needed. Rotors or stators have many airfoils. In order to save computation time and simplify the topology the mesh is limited to one airfoil channel and a periodicity condition is used in Θ—direction (fig.1). Upstream and downstream flows (Z - direction) are assumed to have axisymetric field for some aerodynamic parameters but with non-zero radial and axial gradients. This is why upstream and downstream boundaries are axisymetric surfaces and very often they are a plane normal to the turbomachine rotation axis. This choice makes easier the treatment of periodicity conditions. Conditions on hub and tip surfaces are classical wall conditions. Last characteristics of turbomachine airfoils are the high level of camber, specially for turbine blades, or the high stagger, specially for transonic fans.

The suitable mesh qualities for flow computation in turbomachine are :
- good regularity
- good orthogonality
- easy introduction of periodicity condition

3 - DISCUSSION ABOUT MESH TYPES

An H-grid (fig.2) has been used very widely for inviscid and viscous flow computations. Although H-grid has very good far field properties and is easy to apply to periodicity condition, the grid tends to skew significantly when applied to transonic fans and highly cambered turbine blades. With the recent improvement in grid generation technique (either elliptic or algebraic), most leading edge and trailing edge problems can be avoided. However, the overall skewness of the grid remains still too high for three-dimensional viscous flow computation. For the three-dimensional Euler equations, using a good stability solver such as Lax-Wendroff-Ni finite volumes scheme, H-grid can be performed on transonic fans with splitter and dampers (see chapter 4).

In viscous flow computation, near the endwall region, a good quality grid is necessary in the middle of passage as well as near the blade. A single O-grid or C-grid (fig.2) provides good grid resolution around the blade and in the wake (C-grid). But the single O-grid or C-grid becomes very skewed at the inflow and periodic boundaries due to the constraint of spacial periodicity

An I-grid which can be treated as a generalized H-grid, provides reasonably good grid resolution near the blade as well as the far field. With the I-grid spatial periodicity of the grid at the periodicity surface is no more forced and the physical periodicity condition is handled inside the code by higher order interpolation of variables, good orthogonality of the grid can be maintained near the blade surface and at periodic surfaces with most elliptic or algebraic grid generation. Although a single I-grid can be used efficiently for most regular blade row description, significant portion of grids can be saved by wrapping an O-grid or C-grid inside the I-grid when large number of grids are required near the blade for boundary layer resolution.

For two-dimensional flow calculation, different physical assuptions can be also applied on the domain surrounding the profile, a full Reynolds-averaged Navier-Stokes equation can be integrated while inviscid assumption can be made on the remaining domain for economy of solution. However, these assumptions cannot be applied for three-dimensional flow calculation because of viscous flow effects of endwalls. In chapter 5 we provide various examples of using combined I, C, O-grids and subdomains technique

4 - EXAMPLES OF 3D EULER COMPUTATIONS

The 3D Euler code here discussed has been firstly developped for single smooth compressor blades [2, 3], then part span damper was introduced [4] and finally we added downstream splitter for fan applications.

The sheets supporting channel grids are axisymetric stream tubes provided by a through flow calculation (fig.3). A simple algebric H-grid is used with a regular tangential distribution of points. The periodic boundaries start at the leading edge (or trailing edge) with the airfoil meanline slope and become meridional lines in the far field. In the axial direction particular lines are fitted with the meridional projection of leading and trailing edge curves. In upstream and downstream region grids lines are spaced according constant rate increase instead of equal spacing which is used inside the blade row.

The use of compatibility equations to impose boundaries conditions [2] permits arbitrary distribution of the points on the periodic boundaries and tangential overlaping is not needed. Grid orthogonalisation, using Poisson's equation, has been tested but in many case it's not necessary.

To compute velocity field around fan blade including part span damper and downstream splitter effects we use multi-domain technique (fig.4). One domain is extending from the hub to the streamline of the splitter lower surface, the second from the streamline of the splitter upper surface to the part-span damper streamline and the third from the part-span damper streamline to the external casing. Grid are generated on each domain by a technique similar to that used for smooth blades. Figure 5 shows an example of such kind of mesh, the total number of mesh points is 22000. For both cases (blade only and blade + part-span + splitter), figures 6 and 7 gives the isomach lines on the suction surface view and on a blade-to-blade view. A shock appears clearly on the damper and the splitter effect can be notted on the lower part of the blade

5 - EXAMPLES OF 2D AND 3D NAVIER-STOKES COMPUTATIONS

In figure 8 an I-O type grid is shown for a transonic turbine rotor blade. Detailed grid near the trailing edge is shown in Figure 9. Various sizes of O-grid can be wrapped around the airfoil inside the I-grid for the resolution of boundary layer growth. As shown in Figure 9, good orthogonality of grid is obtained near the trailing edge where a trailing edge shock system is anticipated. The computed static pressure contours are shown in Figure 10 and detailed trailing edge region is given in Figure 11. The results in Figure 10 and 11 are based on a grid size of 7000 nodes. The accuracy of the solutions are good for the given grid size. Detailed velocity near the trailing edge is shown in Figure 12. The numerical solution was obtained with a upwind relaxation method The laminar-to-turbulent transition and turbulence is modeled with a two-equation turbulence model with a low Reynolds number modification. The details of the numerical scheme is given in [5].

The next examples are a solution of the two-dimensional Reynolds-averaged Navier-Stokes equations completed by the mixing length turbulence model. Numerical scheme is the explicit Lax-Wendroff-Ni finite volumes technique. More details about applications to cascade airfoils are in Ref.[6]. Multigrid steps an local time step are used in order to reduce computation time. We use a multi-domain technique and compatibilities equations are employed on all boundaries.

The first application uses navier-Stokes solver on two I-grids put on both profile sides (Fig.13). An Euler solver is applied on the remaining I-grid in the upstream domain. The boundary slope between the two Navier-Stokes domain is continuous with meanline profile slope and upstream and downstream domain shape is the classical one adopted for H or I-type grids. Normaly to the profile,meshes are spaced according to a geometrical progression. This permit to have a mesh size near the blade consistent with the thickness of the viscous wall layer without unacceptable increase of the number of nodes. On the far wake wider meshes are allowed. The axial spacing is similar to those used in 3D Euler application. This kind of grid does not take in account the actual leading edge shape and bow-shock cannot appear. However most of the flow field is accurately computed and a solution is given in fig.14 for a grid having 18963 nodes. The lambda-shape on the shock/boundary layer interaction zone is clearly shown.

The second application uses C-grid (Navier-Stokes solver) around the profile and a I-grid for the remaining upstream region (Euler solver). In fact the C-grid is built by adding two I-grids at a small C-region around the leading edge. Then a orthogoralisation method with relaxation is applied to this domain and the nodes are renumbered. Fig.15 shows an example of such grid with 20507 nodes. On the boundary between C and I domain the nodes are the same for both grids The numerical solution is given on Fig.16 and one can see on a zoom the accurate computation of the oblique shock becoming bow-shock near the leading edge. The remaining flow field is nearly the same as that computed with I-grid.

The last example is obtained with an I-O type grid (Fig.17) for a modern transonic fan [7]. The grid shows good orthogonality near the leading edge at the blade tip where a leading edge shock system is expected. The computed endwall static pressure contours are compared with measured data and inviscid Euler solution in Figure 18.

6 - CONCLUSIONS

Several structured mesh-types are used in turbomachine applications.

For the current 3D Euler codes H or I types are widely employed. They are often sufficient for accurate computation and they have good qualities in the upstream and downstream zones.

For 2D or 3D Navier-Stokes codes, C or O types permit a good representation of the complex transonic viscous flow at the leading edge or at the trailing edge and in the wake. So, several types of grids are used together in order to combine their qualities. I-O or I-C grids have been tested and the results are presented in this paper.

Acknowledgements

REFERENCES

[1] G. KARADIMAS
"Design of High Performance Fans Using Advanced Aerodynamic Codes"
ASME Paper, n°88-GT-141, june 1988

[2] H. VIVIAND, J.P. VEUILLOT
"Methodes pseudo-instationnaires pour le calcul d'écoulements transsoniques"
ONERA.Publication 1978-4 (English translation : ESA TT 561)

[3] J. BROCHET
"Calcul numérique d'écoulements internes tridimensionnels transsoniques"
La Recherche Aérospatiale, Vol 5, sept. 1980, pp 301-315 (English translation : ESA TT 673)

[4] T. DERRIEN
"Calcul tridimensionnel dans les aubages de compresseurs munis de nageoires"
AGARD Conference Proceedings n°401, Sept. 1986, Paper n°14

[5] C. HAH
"Calculation of Three-Dimensional Viscous Flow in Turbomachinery With an Implicit Relaxation Method"
AIAA Journal of Propulsion and Power 3 (5), pp 415-422, 1987

[6] L. CAMBIER, J.P. VEUILLOT
"Computation of Cascade Flows at High Reynolds Number by Numerical Solution of the Navier-Stokes Equations"
AIAA, 26th Aerospace Sciences Meeting, Reno-Nevada (US)

[7] A.J. WENNERSTROM and S.L. PUTERBAUGH
"A Three-Dimensional Model for the Prediction of Shock Losses in Compressor Blade Rows"
Journal of Engineering for Gas Turbines and Power, Vol.106, April 1984, pp 295-299.
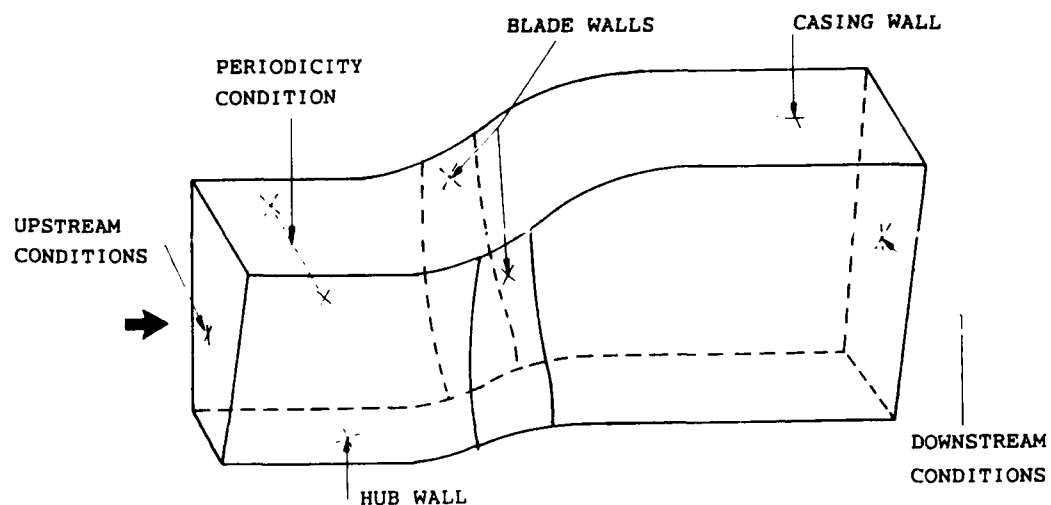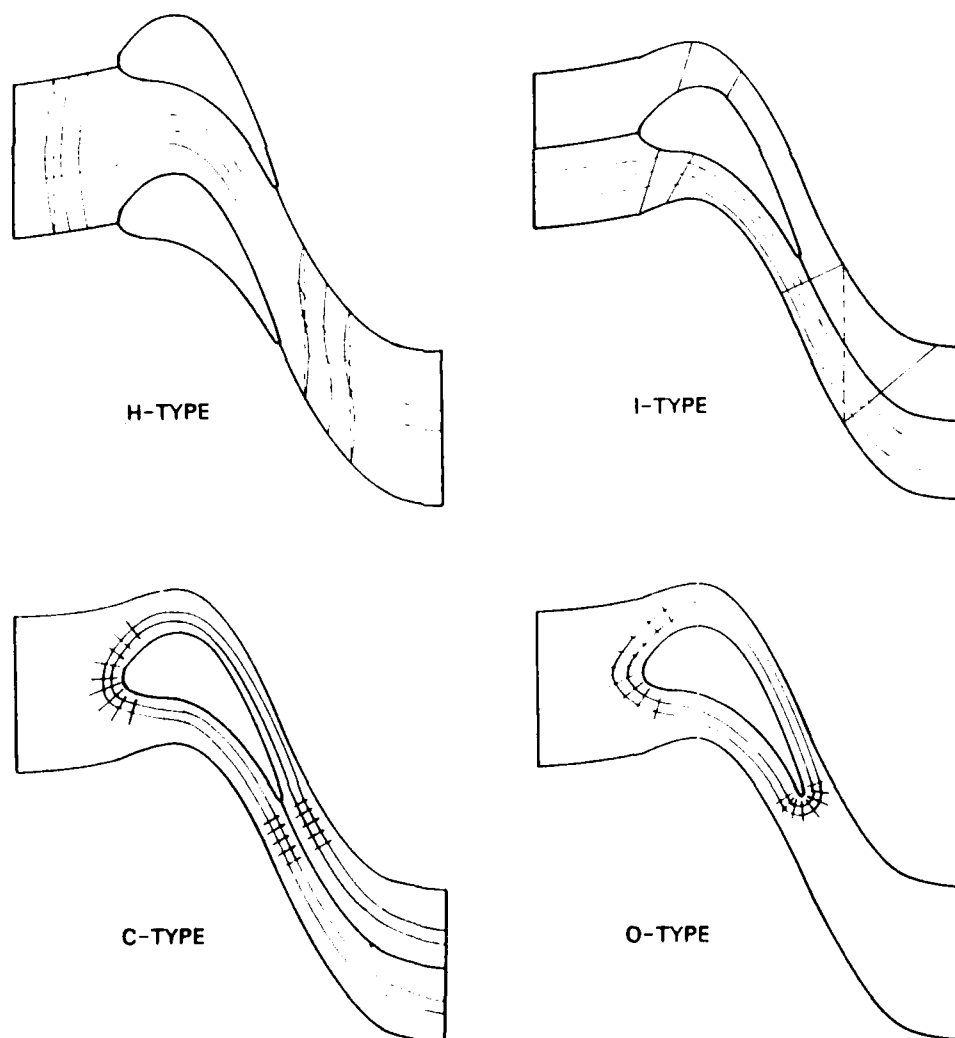
BLADE WALLS

CASING WALL

PERIODICITY CONDITION

UPSTREAM CONDITIONS

DOWNSTREAM CONDITIONS

HUB WALL

FIG.1. COMPUTATION DOMAIN FOR 3 D. FLOWS

H-TYPE

I-TYPE

C-TYPE

O-TYPE

FIG.2. DIFFERENT MESH TYPES

MERIDIONAL SECTION

UPSTREAM FACE

BLADE-TO-BLADE SECTION

FRONT VIEW
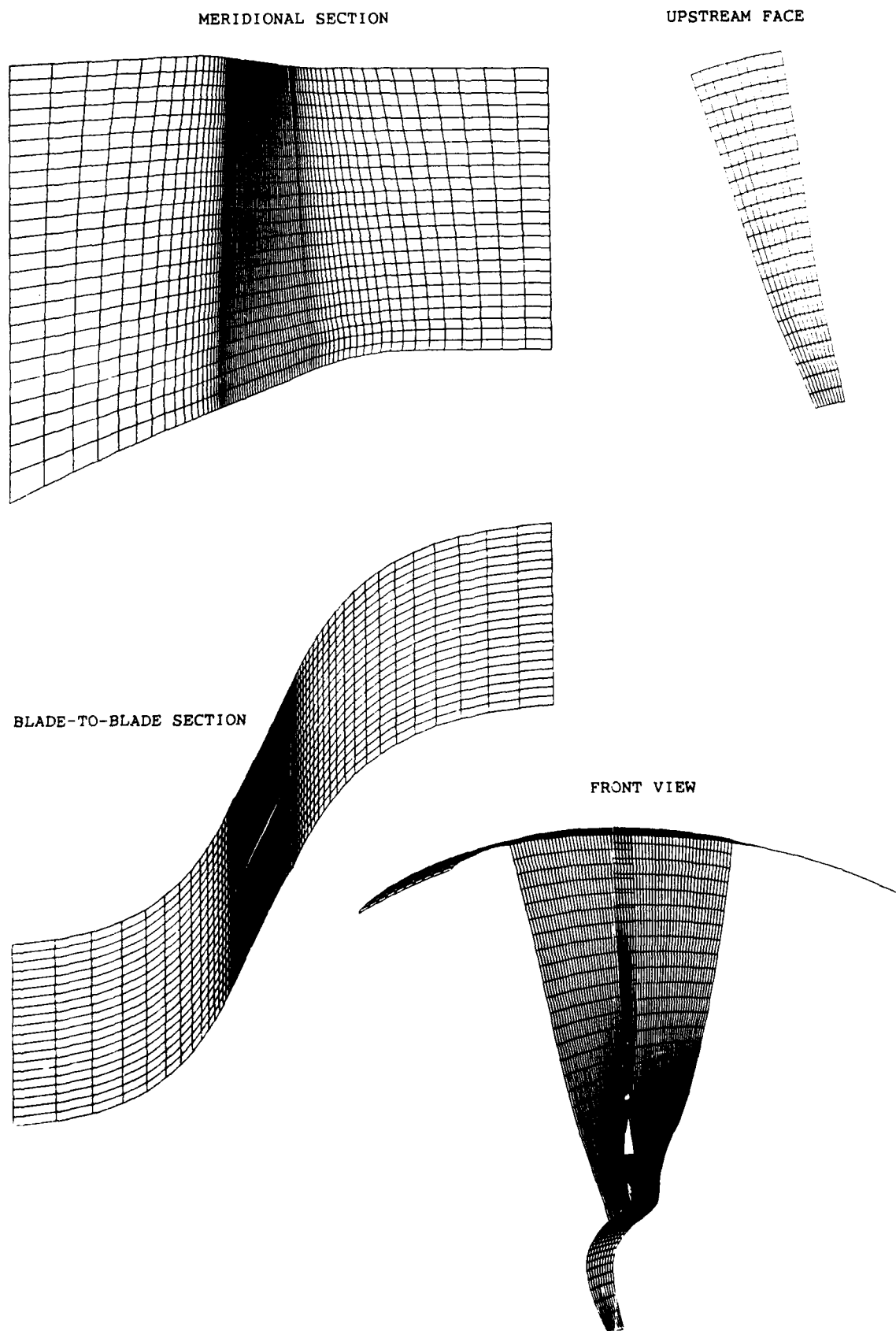
FIG.3. MESH USED FOR 3 D. EULER
SMOOTH FAN BLADE

CASING

③

PART-SPAN DAMPER

②

SPLITTER

①

HUB WALL
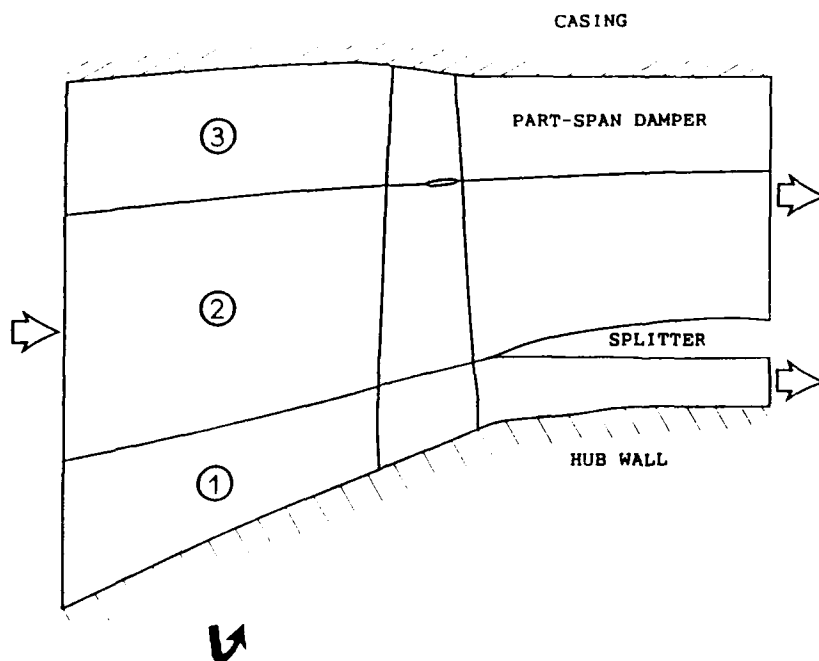
FIG.4. TRANSONIC FAN BLADE + PART-SPAN DAMPER + SPLITTER
SPLITTING IN 3 SUBDOMAINS

MERIDIONAL SECTION

UPSTREAM FACE

BLADE-TO-BLADE SECTION

FRONT VIEW
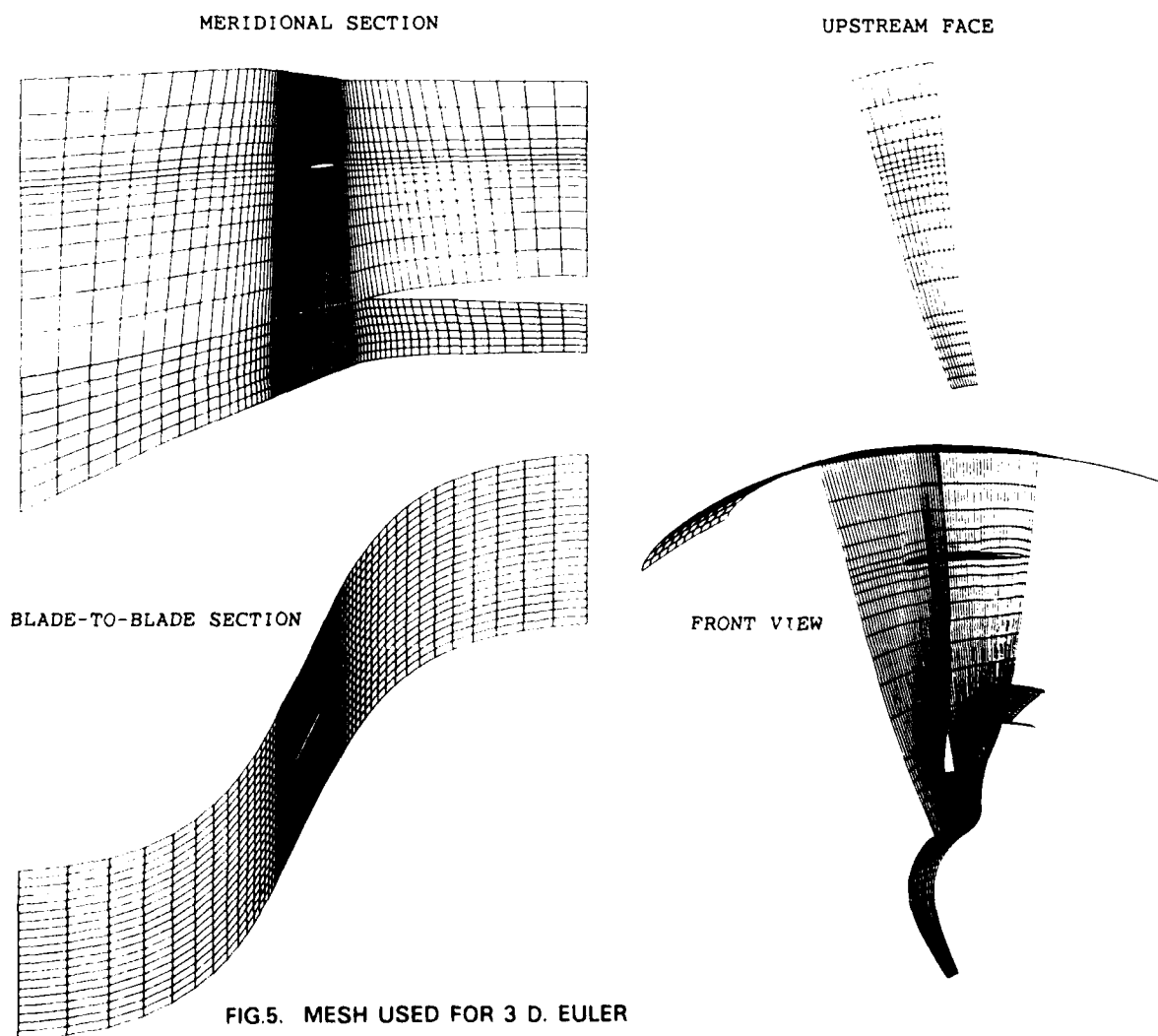
FIG.5. MESH USED FOR 3 D. EULER
TRANSONIC FAN BLADE + PART-SPAN DAMPER + SPLITTER

SMOOTH BLADE   BLADE + DAMPER + SPLITTER
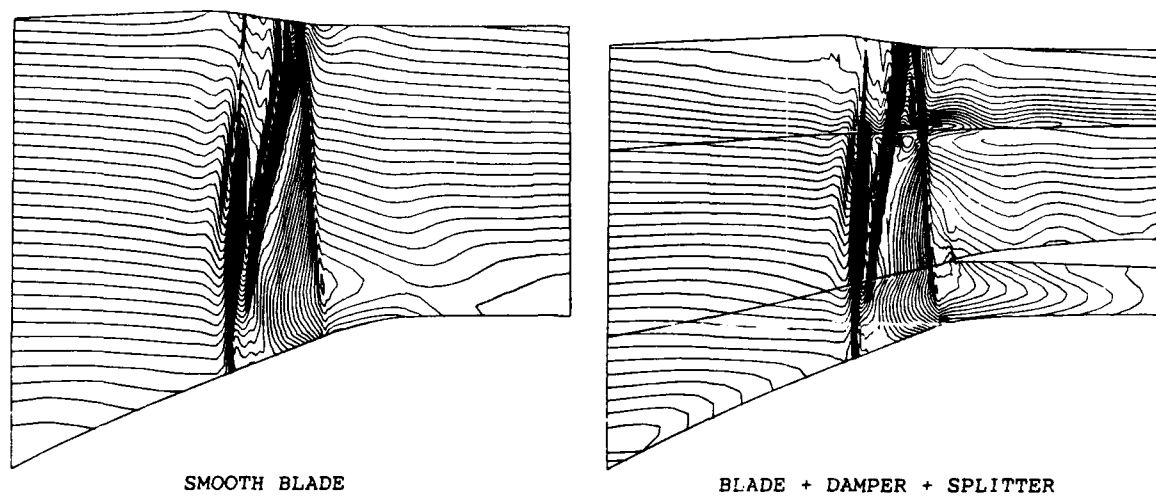
FIG.6. 3 D. EULER - ISOMACH LINES ON SUCTION SIDE
        TRANSONIC FAN BLADE

WITHOUT DAMPER   WITH DAMPER
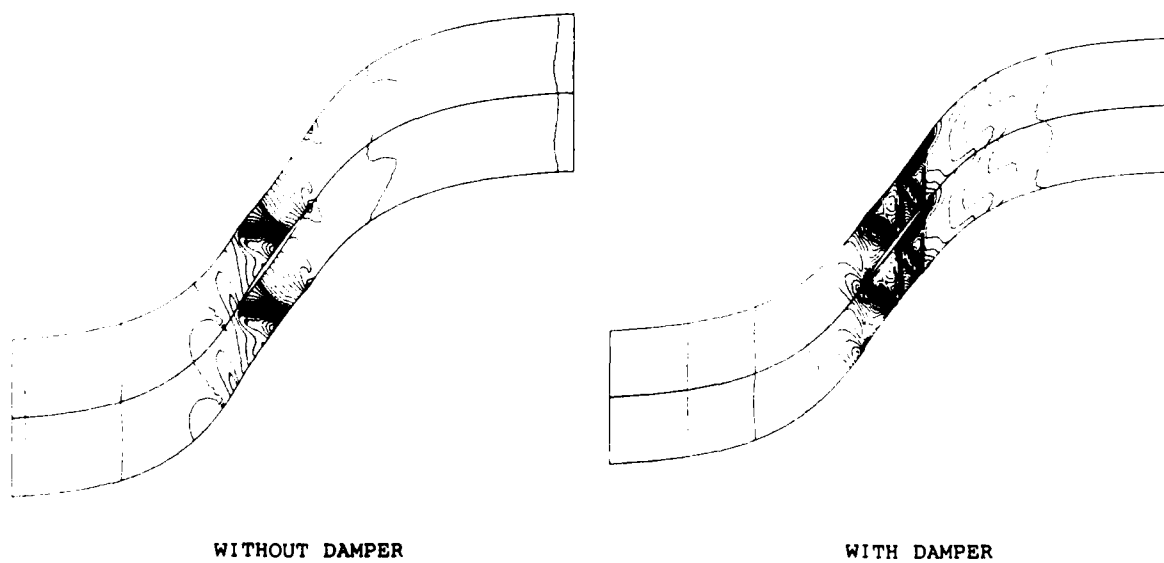
FIG.7. 3 D. EULER - BLADE-TO-BLADE ISOMACH LINES
        SECTION SLIGHTLY ABOVE THE DAMPER
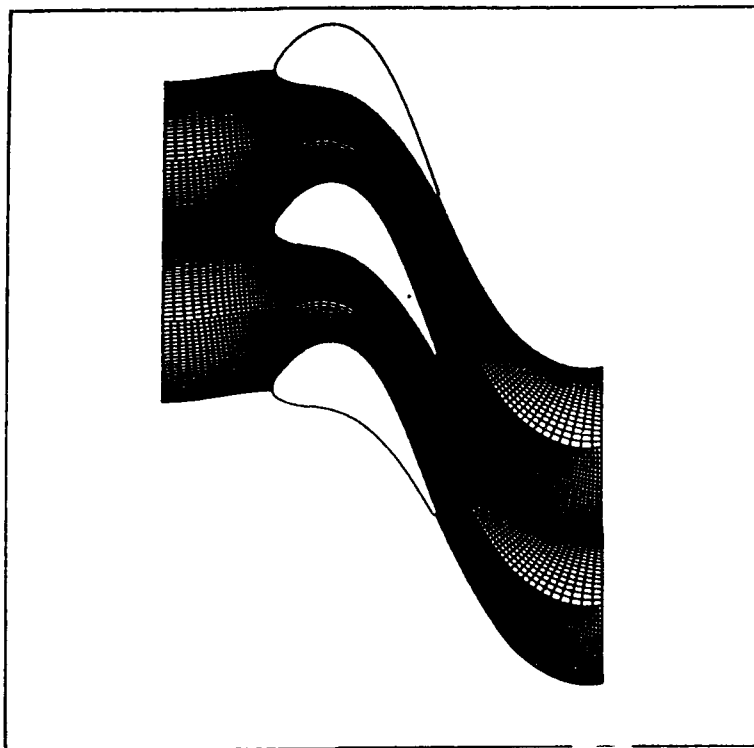
FIG.8. COMPUTATIONAL GRID FOR 2 D. NAVIER-STOKES APPLICATION
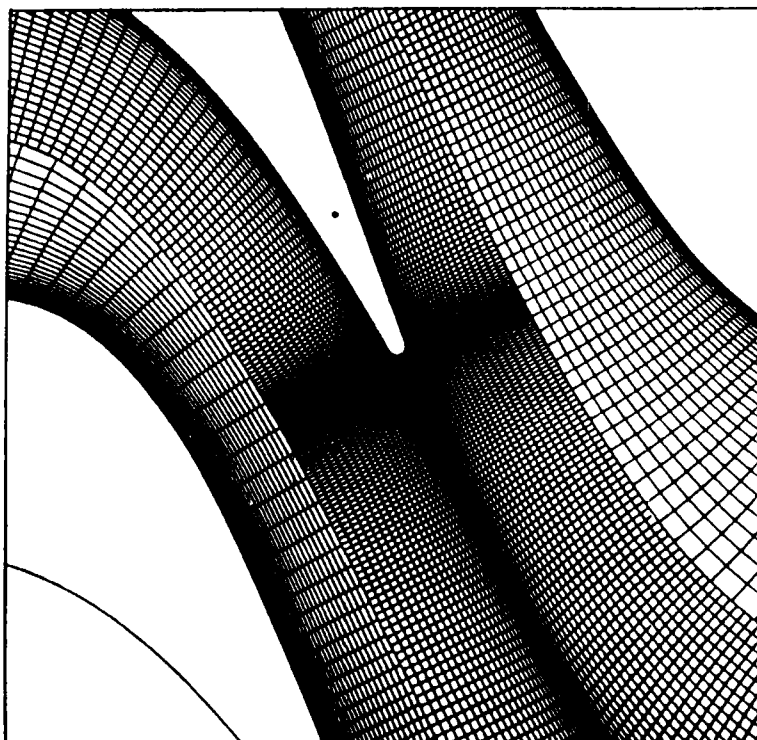ON TRANSONIC TURBINE

FIG.9. COMPUTATIONAL GRID NEAR THE TRAILING EDGE
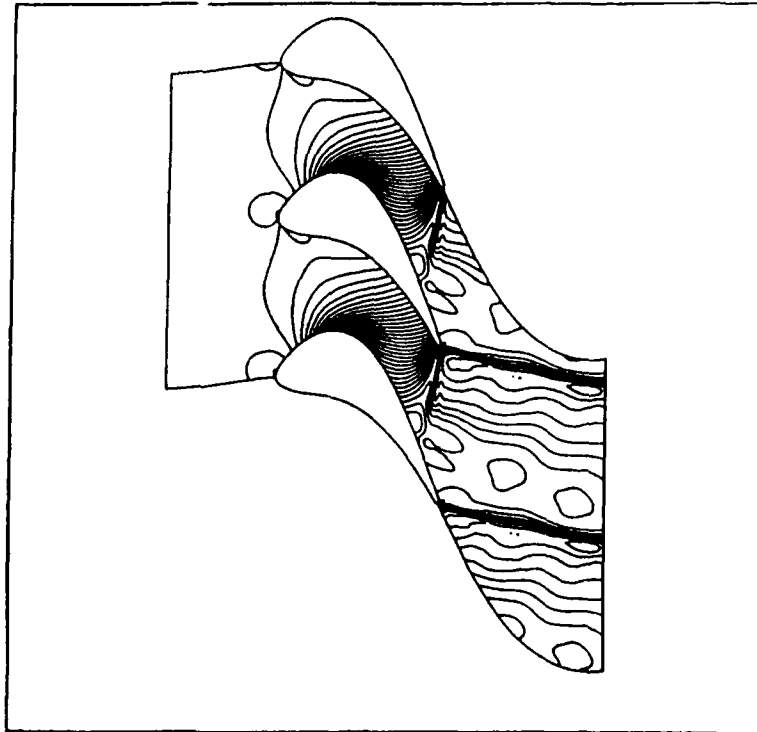TRANSONIC TURBINE APPLICATION

FIG.10. CALCULATED STATIC PRESSURE CONTOURS
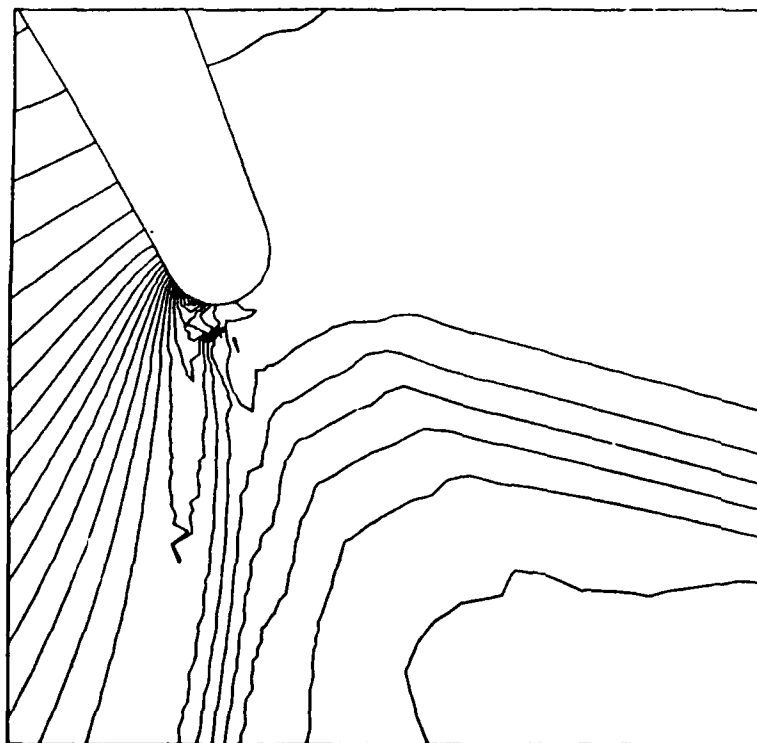TRANSONIC TURBINE APPLICATION

FIG.11(A). CALCULATED STATIC PRESSURE CONTOURS
NEAR THE TRAILING EDGE
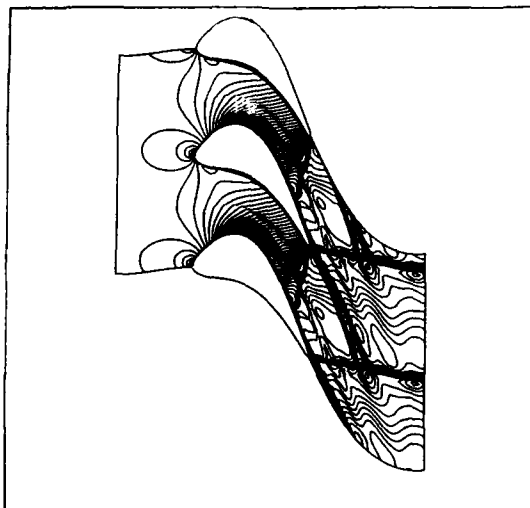TRANSONIC TURBINE APPLICATION

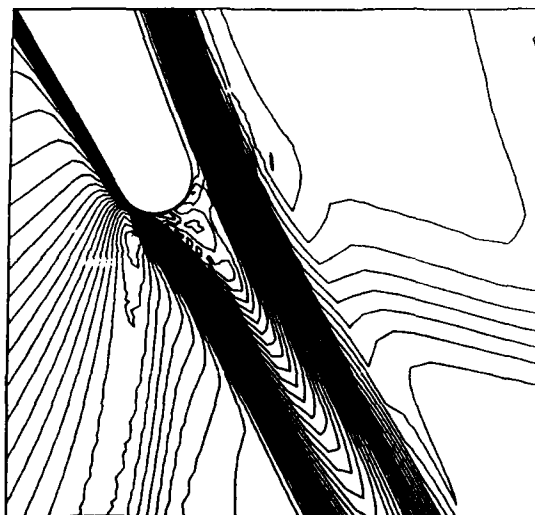FIG.11(B). CALCULATED MACH NUMBER CONTOURS
TRANSONIC TURBINE APPLICATION

FIG.11(C). CALCULATED MACH NUMBER NEAR THE TRAILING EDGE
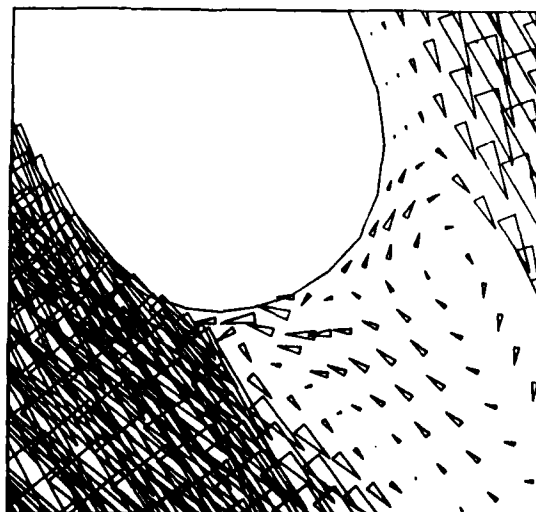TRANSONIC TURBINE APPLICATION

FIG.12. CALCULATED VELOCITY VECTORS NEAR THE TRAILING EDGE
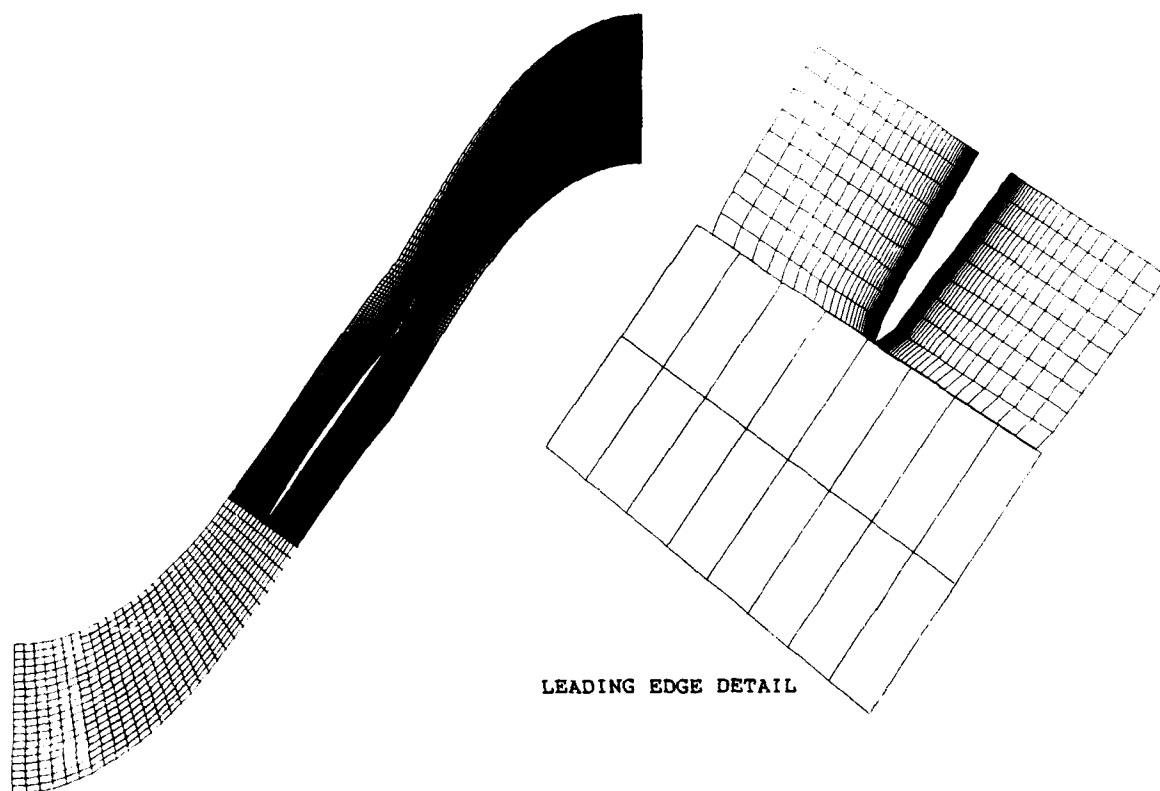TRANSONIC TURBINE APPLICATION

LEADING EDGE DETAIL

FIG.13. I-MESH FOR NAVIER-STOKES CALCULATIONS
18.963 POINTS

FIG.14. NAVIER-STOKES SIMULATION WITH A I-MESH
STATIC PRESSURE DOWNSTREAM / TOTAL PRESSURE UPSTREAM = 0.528
UPSTREAM MACH NUMBER = 1.36

LEADING EDGE DETAIL

FIG.15. C-MESH FOR NAVIER-STOKES CALCULATIONS
20.507 POINTS

LEADING EDGE DETAIL
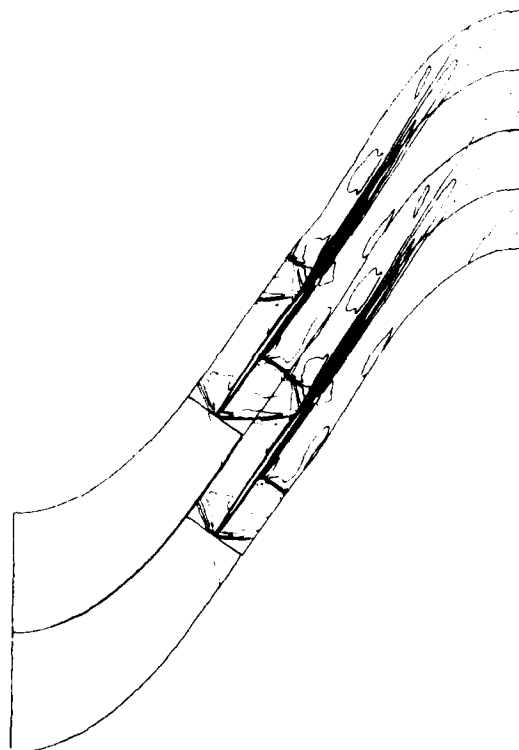
FIG.16. NAVIER-STOKES SIMULATION WITH A C-MESH
STATIC PRESSURE DOWNSTREAM / TOTAL PRESSURE UPSTREAM = 0.528
UPSTREAM MACH NUMBER = 1.36

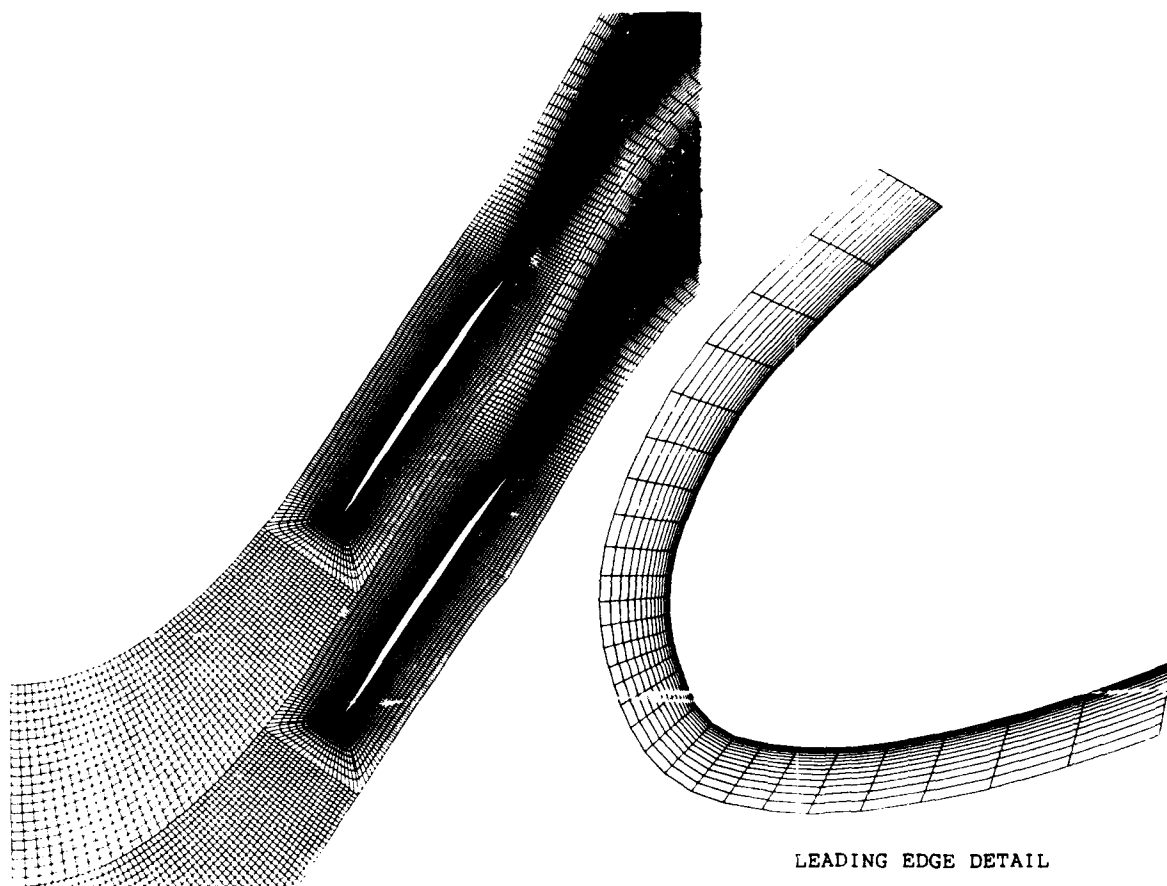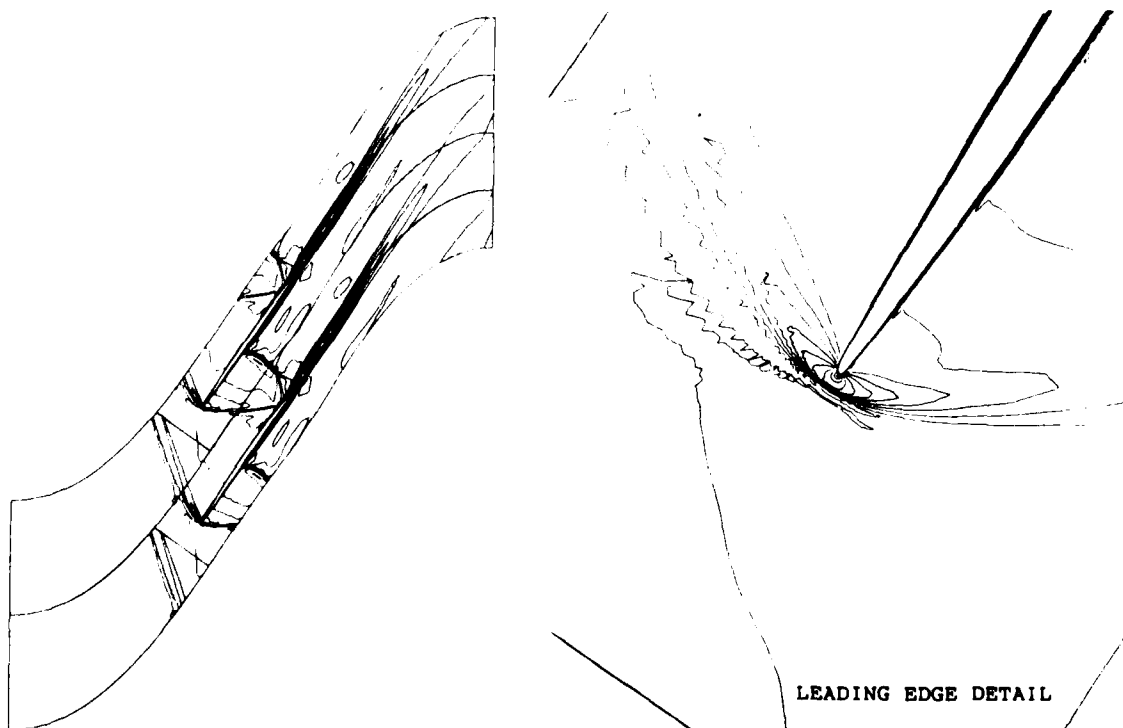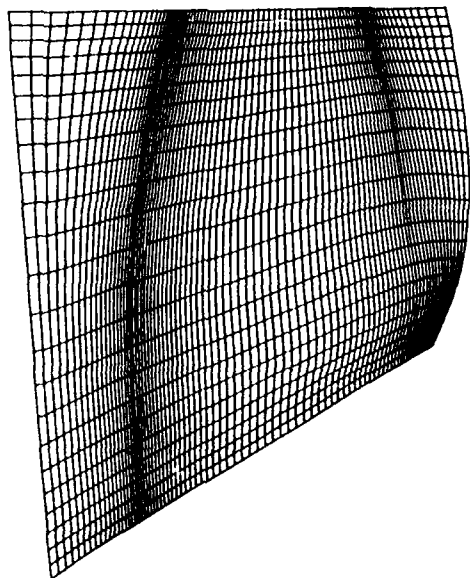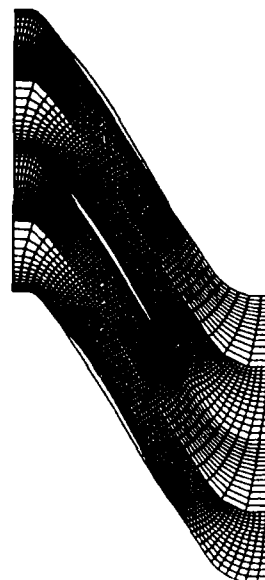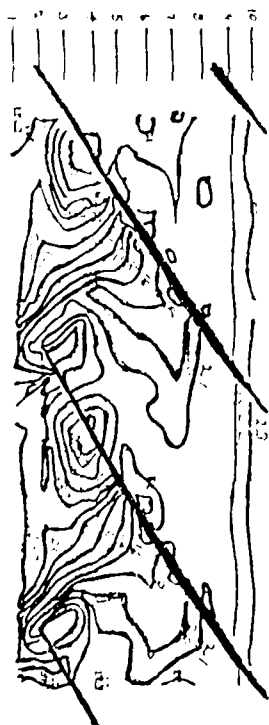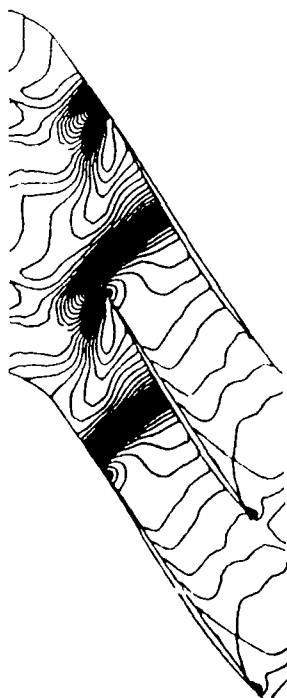HUB-TO-TIP   BLADE-TO-BLADE
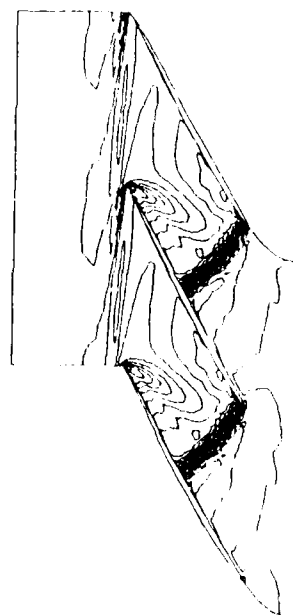
FIG.17. COMPUTATIONAL GRID FOR A SUPERSONIC FAN

KULITE MEASUREMENT   VISCOUS FLOW SOLUTION   EULER SOLUTION

FIG.18. COMPARISON OF SHROUD STATIC PRESSURE CONTOURS

Unsteady Euler Solutions on Dynamic Blocked Grids
for Complex Configurations

by
David L. Whitfield
Professor
J. Mark Janus
Research Engineer
Abdollah Arabshahi
Post Doctorate Assistant
Mississippi State University
Department of Aerospace Engineering
Drawer A
Mississippi State, MS 39762
USA

ABSTRACT

A numerical method for solving the three–dimensional unsteady Euler equations on dynamic multiblocked grids about complex configurations in transonic flow is presented. Two configurations are considered. The first is a wing–pylon–store configuration with the store in the captive position, and then vertically launched from the wing–pylon. The second is a counter–rotating unducted propfan. The numerical results are validated by comparisons with available experimental data.

1. INTRODUCTION

Computational fluid dynamics (CFD) has matured to the point that steady–state numerical solutions have been obtained for the flow about complex three–dimensional configurations. These steady–state solutions are computationally expensive, but unsteady solutions for complex configurations are so much more expensive that they become totally impractical for most of the fluid community. Part of the reason for the enormous expense of unsteady solutions is that the permissible time step is usually restricted by the numerical algorithm and not the physics of the flow considered. Small computational cells, even if there are relatively few in number such as near a solid surface, can greatly increase the number of time steps required to complete the motion. By removing or reducing the time step restriction in the numerical algorithm, far fewer time steps can be used while still capturing the essential physics of the flow. The purpose of this paper is to address steady and, in particular, unsteady solutions of the Euler equations for relatively complex three–dimensional configurations. The numerical algorithms will be discussed with regard to stability and the corresponding allowable time steps. A method used to generate three–dimensional blocked grids, such as used in this work, is described in another paper at this Specialists' Meeting. Consequently, other than techniques used to manipulate blocked grids that move relative to one another, the emphasis here will be on the method used to obtain steady and unsteady flow solutions and not on grid generation. Selected numerical results will be presented and compared with experimental data, where available, for the following configurations in transonic flow: (1) wing–pylon–store and (2) counter–rotating unducted propfan.

2. EQUATION FORMULATION AND NUMERICAL ALGORITHM

Conservative Model

The ultimate goal in computational fluids is to minimize the approximations to the most fundamental modeling equations, presumably to salvage most of the physics, while attaining modest execution times on the available equipment. Traditionally the analysis of rotating machinery begins with the casting of the modeling equations in a cylindrical reference frame in an effort to benefit from the time–asymptotic steady–state solutions which exist for particular configurations. One of the goals here, however, is to produce field simulations of general complex rotating configurations, including those containing interacting components. Hence, the solutions sought are of genuine unsteady flowfields. In the interest of computational brevity, an assumption of a nonconducting, inviscid, perfect gas with no body forces will be made. It is anticipated that viscous flowfield simulations can be produced with only minor modifications to the procedures outlined herein. Efforts toward this end are presently underway. In this light, the unsteady three–dimensional Euler equations in conservative differential form are transformed from a Cartesian reference frame to the time–dependent body–fitted curvilinear reference frame[1]

$$\xi = \xi(x, y, z, t)$$

$$\eta = \eta(x, y, z, t)$$

$$\zeta = \zeta(x, y, z, t)$$ (1)

$$\tau = t$$

yielding

$$\frac{\partial Q}{\partial \tau} + \frac{\partial F}{\partial \xi} + \frac{\partial G}{\partial \eta} + \frac{\partial H}{\partial \zeta} = 0$$ (2)

where

$$Q = J \begin{bmatrix} \varrho \\ \varrho u \\ \varrho v \\ \varrho w \\ e \end{bmatrix} \qquad F = J \begin{bmatrix} \varrho U \\ \varrho u U + \xi_x p \\ \varrho v U + \xi_y p \\ \varrho w U + \xi_z p \\ U(e + p) - \xi_t p \end{bmatrix}$$

$$G = J \begin{bmatrix} \varrho V \\ \varrho u U + \eta_x p \\ \varrho v V + \eta_y p \\ \varrho w V + \eta_z p \\ V(e + p) - \eta_t p \end{bmatrix} \qquad H = J \begin{bmatrix} \varrho W \\ \varrho u W + \zeta_x p \\ \varrho v W + \zeta_y p \\ \varrho w W + \zeta_z p \\ W(e + p) - \zeta_t p \end{bmatrix}$$

with the contravariant velocities

$$U = \xi_x u + \xi_y v + \xi_z w + \xi_t$$

$$V = \eta_x u + \eta_y v + \eta_z w + \eta_t$$

$$W = \zeta_x u + \zeta_y v + \zeta_z w + \zeta_t$$

The Jacobian of the inverse transformation, i.e. $\partial(x, y, z)/\partial(\xi, \eta, \zeta)$, is given by

$$J = x_\xi(y_\eta z_\zeta - z_\eta y_\zeta) - y_\xi(x_\eta z_\zeta - z_\eta x_\zeta) + z_\xi(x_\eta y_\zeta - y_\eta x_\zeta)$$

and the metric quantities are

$$\xi_x = J^{-1}(y_\eta z_\zeta - z_\eta y_\zeta) \qquad\qquad \xi_z = J^{-1}(x_\eta y_\zeta - y_\eta x_\zeta)$$

$$\xi_y = J^{-1}(z_\eta x_\zeta - x_\eta z_\zeta) \qquad\qquad \xi_t = -x_\tau \xi_x - y_\tau \xi_y - z_\tau \xi_z$$

$$\eta_x = J^{-1}(z_\xi y_\zeta - y_\xi z_\zeta) \qquad\qquad \eta_z = J^{-1}(x_\zeta y_\xi - y_\zeta x_\xi)$$

$$\eta_y = J^{-1}(x_\xi z_\zeta - z_\xi x_\zeta) \qquad\qquad \eta_t = -x_\tau \eta_x - y_\tau \eta_y - z_\tau \eta_z$$

$$\zeta_x = J^{-1}(y_\xi z_\eta - z_\xi y_\eta) \qquad\qquad \zeta_z = J^{-1}(x_\xi y_\eta - y_\xi x_\eta)$$

$$\zeta_y = J^{-1}(x_\eta z_\xi - z_\eta x_\xi) \qquad\qquad \zeta_t = -x_\tau \zeta_x - y_\tau \zeta_y - z_\tau \zeta_z$$

The approach taken here is based on the integration of the Euler equations in conservation law form over discrete contiguous volumes in computational space. This formulation, commonly referred to as a finite volume method, yields the following discretized integral expression for a three-dimensional computational space with finite volume (cell) centers denoted i,j,k:

$$\frac{\partial Q}{\partial \tau} + \frac{\delta_i F}{\Delta \xi} + \frac{\delta_j G}{\Delta \eta} + \frac{\delta_k H}{\Delta \zeta} = 0 \qquad (3)$$

or with $\Delta \xi = \Delta \eta = \Delta \zeta = 1$

$$\frac{\partial Q}{\partial \tau} = -(\delta_i F + \delta_j G + \delta_k H)$$

where $\delta_*(\cdot) = (\cdot)_{*+1/2} - (\cdot)_{*-1/2}$

In this expression the components of the dependent variable vectors, $Q_{i,j,k}$, represent average values for the i,j,k cell. It is therefore evident that some method must be devised to accurately represent the vector-valued flux functions F(Q), G(Q), and H(Q) on the bounding surfaces (faces) of the cell. One of the methods used in this study is based on a one-dimensional analysis of the Riemann problem local to each interface, established by the discontinuous nature of the dependent variable vector Q within the spirit of a finite volume field. To facilitate the introduction of the method used in this study, a digression to a one-dimensional Cartesian space is in order.

First Order Flux Formula

The analogous (to Eq. (3)) discretized integral form of the Euler equations written for one spatial dimension in Cartesian space appears as

$$\frac{\partial q}{\partial t} + \frac{\delta f}{\Delta x} = 0 \tag{4}$$

Godunov[2] proposed a procedure to obtain a global solution to Eq. (4) by solving the set of Riemann problems presented by the interface discontinuities. The Riemann problem local to each interface is costly to solve exactly, due to the necessary iteration. Many investigators[3,4,5,6] have made attempts to lessen this computational expense by approximating the solution of the Riemann problem. In essence these methods yield an approximate solution to the exact equation, and hence are dubbed 'approximate Riemann solvers'.

In [7], Philip Roe suggests an alternate procedural choice. Roe proposed to obtain the exact solution to an approximate equation. The cleverness of Roe is evidenced by his choice of approximate equation. Consider the quasilinear form of Eq. (4)'s parent conservation law,

$$\frac{\partial q}{\partial t} + \bar{A}(q_L, q_R)\frac{\partial q}{\partial x} = 0 \tag{5}$$

where $\bar{A}(q_L, q_R)$ is a constant matrix representative of local interface conditions. Matrix $\bar{A}$ is chosen to have the following specific list of properties, which Roe "christened Property U (since it is intended to ensure uniform validity across discontinuities)":

1. It constitutes a linear mapping from the vector space q to the vector space f.

2. As $q_L \to q_R \to q$, $\bar{A}(q_L, q_R) \to \bar{A}(q)$, where $\bar{A} = \frac{\partial f}{\partial q}$.

3. For any q_L, q_R, $\bar{A}(q_L, q_R) \cdot (q_R - q_L) = f_R - f_L$.

4. The eigenvectors of $\bar{A}$ are linearly independent.

Restricting $\bar{A}$ to the satisfaction of Property U results in a special, unique[8] averaging process for the dependent variables from which $\bar{A}$ is constructed. Referred to as "Roe averaged", the dependent variables are given by the following expressions:

$$\varrho = (\varrho_L \varrho_R)^{1/2} \tag{6a}$$

$$u = \frac{\varrho_L^{1/2} u_L + \varrho_R^{1/2} u_R}{\varrho_L^{1/2} + \varrho_R^{1/2}} \tag{6b}$$

$$H = \frac{\varrho_L^{1/2} H_L + \varrho_R^{1/2} H_R}{\varrho_L^{1/2} + \varrho_R^{1/2}} \tag{6c}$$

where the total enthalpy, H, is defined

$$H = \frac{1}{\varrho}(e + p) \tag{6d}$$

The interface flux difference can be expressed as

$$df = f_R - f_L = \bar{A} \cdot (q_R - q_L) = \bar{A} \cdot dq \tag{7}$$

where $\bar{A}$ is constructed with "Roe averaged" variables. Armed with the eigensystem of $\bar{A}$ and the knowledge that the interface differential dq is proportional to the right eigenvectors of $\bar{A}$ (as shown in [9]), the interface flux difference can be written relative to the right eigenvector basis of $\bar{A}$ as

$$df = \sum a_j \lambda^{(j)} r^{(j)} = \sum{}^+ a_j \lambda^{(j)} r^{(j)} + \sum{}^- a_j \lambda^{(j)} r^{(j)} = df^+ + df^- \tag{8}$$

Physically, the flux difference is shown to be the composition of a collection of waves. In Eq. (8), $r^{(j)}$ is a right eigenvector of $\bar{A}$; a_j is the strength of the jth wave (the jump in the characteristic variable across it), $\lambda^{(j)}$ is an eigenvalue of $\bar{A}$ (the speed of the jth wave); and $\sum{}^-$ and $\sum{}^+$ denote summation over the negative and positive wave speeds, respectively.

The interface flux can be computed from either of the following formulae:

$$f_{i+1/2} = f_L + \sum{}^- a_j \lambda^{(j)} r^{(j)} \tag{9a}$$

$$f_{i+1/2} = f_R - \sum{}^+ a_j \lambda^{(j)} r^{(j)} \tag{9b}$$

$$f_{i+1/2} = \frac{1}{2} [f_L + f_R - \sum a_j |\lambda^{(j)}| r^{(j)}] \tag{9c}$$

This first-order interface flux formula, commonly referred to as flux difference splitting (FDS), which was developed for the one-dimensional equations can be used in a multidimensional space provided the assumption is made that all waves travel normal to their respective interfaces. Also, the special averaging process follows directly for each component of the velocity vector in multidimensional space.

Higher Order (TVD) Flux Formulas

In order to provide solutions of higher spatial accuracy, a family of schemes[10] can be represented by the addition of a corrective flux to the first-order interface flux, Eq. (9), produced in the preceding analysis. Hence the higher order interface flux is given by:

$$f_{i+1/2} = \hat{f}_{i+1/2} + \frac{(1+\phi)}{4}[df^+_{i+1/2} - df^-_{i+1/2}] + \frac{(1-\phi)}{4}[df^+_{i-1/2} - df^-_{i+3/2}] \tag{10}$$

The principal part of the truncation error for this flux formula is found to be

$$TE = -\frac{(\phi - 1/3)}{4}(\Delta x)^2 \frac{\partial^3}{\partial x^3}f(q) \tag{11}$$

The details (common names, order of accuracy, etc.) of the members of this family can be found in [11,12]. Two members, third-order ($\phi = 1/3$) and fully upwind second-order ($\phi = -1$) were, by choice, exclusively used (examined) in this work.

The discussion of a higher order scheme inherently involves a method used to control spurious oscillations, i.e. dispersive errors. The method, actually methods, used in this work concern limiting components of the interface flux to produce total variation diminishing (TVD) schemes, i.e. nonoscillatory schemes. The following formulas have theoretical development as TVD schemes only in scalar nonlinear equations and systems of linear equations in one-dimension. The use of "limiters" yields the following expressions for the corrective flux terms:

$$df^+_{i+1/2} = \sum{}^+ L_j(1, -1)r^{(j)}_{i+1/2}$$

$$df^+_{i-1/2} = \sum{}^+ L_j(-1, 1)r^{(j)}_{i+1/2}$$

$$df^-_{i+1/2} = \sum{}^- L_j(1, 3)r^{(j)}_{i+1/2} \tag{12}$$

$$df^-_{i+3/2} = \sum{}^- L_j(3, 1)r^{(j)}_{i+1/2}$$

One of the limiters studied, referred to as a minmod limiter, is implemented by way of the following definition of the L function

$$L_j(m, n) = minmod(\sigma^{(j)}_{i+m/2}, \beta\sigma^{(j)}_{i+n/2}) \tag{13}$$

with $\sigma^{(j)}$, a parameter proportional to the change in dependent variables across nearby interfaces, defined by

$$\sigma^{(j)}_{i+\frac{(\cdot)}{2}} = \lambda^{(j)}_{i+\frac{1}{2}} a_{j,i+\frac{(\cdot)}{2}} = \lambda^{(j)}_{i+\frac{1}{2}} \ell^{(j)}_{i+\frac{1}{2}} dq_{i+\frac{(\cdot)}{2}} \tag{14}$$

where $\ell^{(j)}$ is a left eigenvector of $\overline{A}$. The minmod limiter is then defined

$$minmod|x, y| = sign(x) \max\{0, min[|x|, y\ sign(x)]\} \tag{15a}$$

and the parameter β is a "compression" parameter given by

$$1 < \beta \le \frac{3-\phi}{1-\phi} \tag{15b}$$

In this work the maximum β was used in all cases.

Another limiter studied, this one credited to Roe[13], is called Superbee. It is implemented as follows

$$L_j(m, n) = cmplim(\sigma^{(j)}_{i+m/2}, \sigma^{(j)}_{i+n/2}) \tag{16}$$

where

$$cmplim|x, y| = sign(x) \max\{0, min[|x|, \beta y\ sign(x)], min[\beta|x|, y\ sign(x)]\} \tag{17}$$

and another compression parameter β, differing from that defined by Eq (15b), is taken here to be two.

Approximately Factored Implicit Scheme

In light of the fact that Eq. (3) has yet to be integrated in time, no mention has been made as to what time level the numerical interface fluxes appearing on the RHS are evaluated. The underlying theory of the approximate Riemann solver presented thus far is based on explicit concepts which result in an unattractive, rather stringent time-step restriction. Equation (3) can be written in a linearized discrete-integral delta form to cover a broad class of explicit and implicit schemes[14]:

$$[I + \frac{\theta \Delta \tau}{1 + \psi} M^n]\Delta Q^n = - \frac{\Delta \tau}{1 + \psi} R^n + \frac{\psi}{1 + \psi} \Delta Q^{n-1} \tag{18}$$

Some of the implicit time differencing schemes represented are $(\theta = 1, \ \psi = 1/2)$ three point backward, $(\theta = 1, \ \psi = 0)$ backward Euler, and $(\theta = 1/2, \ \psi = 0)$ trapezoidal.

Formally, all terms appearing in this equation should result from a single flux formulation. Superior results have been obtained, though, by evaluating the residual term R^n with flux difference split theory, and the left-hand-side (LHS) operator with flux vector split (FVS) theory, see [15 and 16]. The rationale behind this is presently attributed to the more dissipative nature of the FVS theory. The following expressions, Eq. (19 and 20), complete Eq. (18) for this hybrid scheme

$$M^n = \delta_i A^+ + \delta_i A^- + \delta_j B^+ + \delta_j B^- + \delta_k C^+ + \delta_k C^- \tag{19}$$

with

$$A^+ = (\frac{\partial F^+}{\partial Q})^n$$

$$A^- = (\frac{\partial F^-}{\partial Q})^n$$

$$B^+ = (\frac{\partial G^+}{\partial Q})^n$$

$$\cdot$$
$$\cdot$$
$$\cdot$$

where F^+, F^-, G^+, ... result from Steger-Warming flux vector split theory with the elements of A^+, A^-, B^+, ... being given in [17], also

$$R^n = \delta_i F^n + \delta_j G^n + \delta_k H^n \tag{20}$$

where, F, G, and H result from the flux difference split theory discussed herein.

The LHS of Eq. (18) tends to be cumbersome and difficult to invert, not to mention very costly. In light of this, the LHS was approximately factored into the product of two operators, each of which involve the passage of selected information. Here a forward and backward operator are used (block LU factorization), yielding the following two step (LU) scheme:

$$[I + \frac{\theta \Delta \tau}{1 + \psi} M^+] [I + \frac{\theta \Delta \tau}{1 + \psi} M^-]\Delta Q^n = - \frac{\Delta \tau}{1 + \psi} R^n + \frac{\psi}{1 + \psi} \Delta Q^{n-1} \tag{21}$$

or

$$[I + \frac{\theta \Delta \tau}{1 + \psi}(\delta_i A^+ + \delta_j B^+ + \delta_k C^+)]\Delta Q^* = - \frac{\Delta \tau}{1 + \psi} R^n + \frac{\psi}{1 + \psi} \Delta Q^{n-1} \tag{22a}$$

$$[I + \frac{\theta \Delta \tau}{1 + \psi}(\delta_i A^- + \delta_j B^- + \delta_k C^-)]\Delta Q^n = \Delta Q^* \tag{22b}$$

$$Q^{n+1} = Q^n + \Delta Q^n \tag{22c}$$

Although factoring has been shown to degrade the unconditional stability of Eq. (18),[18] it has been our experience that the (LU) scheme apparently retains this touted attribute. Equations (22) are in the final form of the mathematical model developed for the time-accurate analysis.

Boundary Conditions

Since the approximate Riemann solver is a characteristic based scheme, the characteristic variable boundary conditions developed in [1] relative to a three-dimensional time-dependent body-fitted reference frame for inflow, outflow, and impermeable boundaries are employed where applicable. As in [1], phantom cells are utilized to implement these boundary conditions. The changes in dependent variables (ΔQ^n), and ΔQ^*, are set to zero in the phantom cells for inflow, outflow, and impermeable boundaries.

3. BLOCKED GRIDS

As mentioned in the introduction, the scope of this work can encompass extremely complex flowfields as well as complex geometries. At times, in order to adequately resolve these flowfields, an enormous number of cells are required. With the present formulation, approximately 190 vital pieces of information must be known for each cell (up to 115 simultaneously). Bearing this in mind, it is easy to see how the vast majority of present-day supercomputers are unable to support such calculations due to insufficient internal (primary) memory. In addition, the mesh (grid) for most complex geometries is more easily generated in pieces, where each piece generally conforms to a single component of the overall configuration. These are but a couple of the reasons which can be cited for the segmenting of one virtually insurmountable flow environment into several, smaller, more manageable, intercommunicating flow environments.

This segmenting is commonly referred to as composite gridding the field, of which there exist three common methods: overlaid, patched, and blocked. Examples include: the chimera (overlaid) scheme of Benek, Buning, and Steger[19], the zonal (patched) scheme of Rai[20], and the dynamic block scheme of Belk[17]. The approach taken here, a dynamic blocked grid method, is similar to that taken by Belk. In [17], Belk investigated many of the dilemmas posed when attempting time-accurate simulations of flowfields while using a blocked grid structure of a dynamic (moving) mesh. His emphasis was on the formidable task of developing a computer algorithm to handle a completely arbitrary arrangement of generally dissimilar blocks. Belk's general approach was used here for the wing-pylon-store computations. Unfortunately, this general approach adds to the complexity of the code.

For the case of turbomachinery, the nature of the geometry suggests possible block arrangement and characteristic restrictions which can yield significantly simpler algorithm logic. The block structure proposed here for the specific case of dynamic cylindrical geometries (generally, bladed or finned bodies of revolution) will be referred to as selected similarity mapped multiblock. For details of the block arrangement for the special case of turbomachinery, see [21] and [22]. For a discussion of the general case, see [17] and [23].

4. RESULTS

Numerical results are presented for store aerodynamics involving the mutually interfering transonic flow about a wing-pylon-store configuration, in both the captive and launch positions. Also, results are presented for rotating machinery involving a counter-rotating unducted propfan propulsion system.

Wing-Pylon-Store

Steady state multiblock solutions are demonstrated by computing the flow about the wing-pylon-store configuration with the store in the captive position. Unsteady dynamic multiblock solutions are demonstrated by computing the flow about the complete multibody configuration as the store moves away from the parent wing-pylon configuration through a vertical launch trajectory. Unfortunately, no experimental data is available for comparison with the unsteady moving store solution; however, experimental data is available for the captive position and is compared with the numerical solutions.

The wing-pylon-store configuration considered was the same as that used in wind tunnel experiments. The basic configuration is shown in Fig. 1 with the store in the captive position and in Fig. 2 with the store located two store diameters below the pylon. The wing was a symmetrical airfoil and the leading edge was swept 45 degrees. The store was an ogive-cylinder with a cylindrical sting joined to the store boattail. The pylon was a biconvex airfoil shape, and a small gap existed between the store and pylon in both the experimental and computational configuration. The complete grid was composed of 30 blocks.

The numerical solution was run for a freestream Mach number of 0.85 and zero degree angle of attack. Numerical and experimental surface pressure distributions on the outboard and inboard sides of the store in the captive position are shown in Figs. 3 and 4, respectively. Notice that there is a large lower pressure region on the inboard side of the store (Fig. 4) than on the outboard side of the store (Fig. 3). Figure 5 is included to show that the same thing happens, computationally and experimentally, on the pylon. The result of this pressure differential would be that a released store would have an initial side force that would push the store toward the fuselage rather than away from the fuselage.

The reason for the pressure being lower on the inboard side of the store and pylon is attributed primarily to the presence of the store. Figure 6 is used to argue this point. This figure compares computations corresponding to the store dropping through a point two store diameters away from the pylon with steady state experimental data for the wing and pylon only (no store) at the same flow conditions. Notice that the inboard and outboard pressures on the pylon without the store present are now much closer to the same values. (One should note that it is dangerous to compare unsteady computations with steady state experimental data, but unsteady experimental data are not available and the assumption is made that the store being two diameters away will not significantly influence the unsteady flow about the wing and pylon.)

Counter-Rotating Propfan

The configuration considered is the GE UDF8-8, a counter-rotating unducted fan immersed in an oncoming $M_\infty = 0.7$ axial flow, see Fig. 7. The configuration has two fan rows with eight blades per row. The fore row rotates clockwise and the aft row rotates counterclockwise. Both blade rows rotate with an advance ratio, J, of 2.8. The highly swept, tapered, twisted, thin blades are designed to reduce the axial Mach number through the blading to alleviate compressibility losses.

The $\alpha = 0°$ solutions appearing herein and in [24] were obtained using only two blocks, one per blade passage (benefitting from solution symmetry). Although only two blocks were used, axial interblock communication was implemented with a full buffer ring (temporary storage area for injected or extracted data). The procedure involves extracting data, imaging the data to form a full 360° communication buffer ring, then allowing the appropriate data to be injected based on the positional relationship between the blocks and the buffer ring. Each block mesh was H-type in all directions and contained 56x21x10 (i,j,k) cells.

To begin the transition from the first-order time FVS solutions presented in [24] to the FDS solutions presently available, consider the comparison of the local relative Mach number of [24] to that of second-order time FVS with block-block interfaces maintained to the interior level of spatial accuracy (up to second-order for FVS), as shown in Figs. 8. At first glance it is quite noticeable that the second-order solutions do not expand nearly as much as the first-order. Presently the cause of this anomoly is under investigation. It is not known whether this is due to the modifications made to the block-block interface or to the use of three point backward (second-order) time differencing.

With this noted and under investigation, the transition is completed with the local relative Mach number comparison between FVS second-order space and FDS third-order space (minmod) both with three point backward time differencing, as shown in Figs. 9. The increase in spatial resolution due to FDS is evident with the sharper shock definitions and the ability to resolve (to some extent) the geometric subtleties of the blade geome-

try. Any enhanced resolution would be welcomed considering the extreme coarseness of the blade chordwise mesh (only 10 cells from leading to trailing edge). Also, an increase in the tip loading is noticed with the FDS method.

In Figs. 10, the unsteady behavior or lack thereof as predicted by the FDS method is presented. The curves are of time-averaged freestream relative pressure coefficient, C_p, with the minimum and maximum local cell values indicated by the fluctuation bars. From this plot one can get a feel for the regions of greatest fluctuations and their magnitude. There exists evidence of the inherent unsteady behavior of the flow, though it is by far not prominent. Figures 10 support the comments in [24] regarding lack of variation in the blade surface relative Mach number for both fore and aft blade rows. Though little variation is shown overall, there is more variation in the aft blade row, as expected. The only C_p variations of any significance occur on the pressure side of the aft blade near the root and midspan.

As a final note on the UDF8-8 comparison, the present FDS integrated performance parameters of power coefficient, efficiency, and torque ratio, $(C_{pw}, \eta, Q_2/Q_1)$ respectively, are compared in Figs. 11 to that of FVS second-order time and measured data as reported in [25]. Reasonable agreement is shown to exist with the measured data viewing the inviscid nature of these calculations. Also, the intuitive trends regarding the less numerically dissipative nature of the FDS method compared to the FVS method is a plausible explanation of the relative position of the time-averaged data with respect to the measured data. That is to say, one might expect to see stronger, farther aft (chordwise) shock patterns with FDS resulting in higher compressibility losses; consequently, more power-in for less thrust-out (lower efficiency), as shown in Figs. 11a and 11b would not be unusual.

5. CONCLUSIONS AND COMMENTS

A numerical scheme was presented for solving the three-dimensional unsteady Euler equations on dynamic multiblock grids for complex configurations, and comparisons were made with available experimental data. The numerical formulation used permits extremely large time steps, such that the time step size selected can be established by the physics of the flow being solved and not not the numerics of the algorithm used. This is particularly important for Navier-Stokes calculations where extremely small cells with high aspect ratio (similar to this piece of paper on which these words are printed) could severely restrict the time step for most algorithms. Navier-Stokes calculations on extremely fine grids with high aspect ratio cells of this type, have been successfully carried out by Simpson[16] for maximum Courant numbers greater than 10^4.

REFERENCES

1. Janus, J.M., "The Development of a Three-Dimensional Split Flux Vector Euler Solver with Dynamic Grid Applications", MS Thesis, Mississippi State University, August 1984.

2. Godunov, S.K., "Finite Difference Method for Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics", Mat. Sbornik, Vol. 47, No. 3, pp. 271-306, 1959. Translated as JPRS 7225 by U.S. Dept. of Commerce, November 1960.

3. Glimm, J., "Solutions in the Large for Nonlinear Hyperbolic Systems of Equations", Communications on Pure and Applied Mathematics, Vol. 18, pp. 697-715, 1965.

4. Chorin, A.J., "Random Choice Solution of Hyperbolic Systems", Journal of Computational Physics Vol. 22, pp. 517-533, 1976.

5. Harten, A., and Lax, P.D., "A Random Choice Finite Difference Scheme for Hyperbolic Conservation Laws", SIAM Journal of Numerical Analysis, Vol. 18, pp. 289-315, 1981.

6. Sod, G.A., " A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws", Journal of Computational Physics, Vol. 27, pp. 1-31, 1978.

7. Roe, P.L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes", Journal of Computational Physics, Vol. 43, pp. 357-372, 1981.

8. Roe, P.L., and Pike, J., "Efficient Construction and Utilization of Approximate Riemann Solutions", in Computing Methods in Applied Sciences and Engineering, ed. R. Glowinski and J.L. Lions, 6:499-518, Amsterdam: North Holland, 1984.

9. Jeffrey, A., Quasilinear Hyperbolic Systems and Waves, Pitman Publishing, San Francisco, 1976.

10. Osher, S., and Chakravarthy, S., "Very High Order Accurate TVD Schemes", ICASE Report No. 84-44, Sept. 1984.

11. Chakravarthy, S.R., "A New Class of High Accuracy TVD Schemes for Hyperbolic Conservation Laws", AIAA Paper No. 85-0363, Jan. 1985.

12. Chakravarthy, S.R., Szema, K.Y., Goldberg, U.C., Gorski, J.J., and Osher, S., "Application of a New Class of High Accuracy TVD Schemes to the Navier-Stokes Equations", AIAA Paper No. 85-0165, Jan. 1985.

13. Roe, P.L., "Some Contributions to the Modelling of Discontinuous Flows", Large Scale-Computations in Fluid Mechanics, edited by B. Engquist, S. Osher, and R. Somerville, Lectures in Applied Mathematics, Vol. 22, part 2, pp. 163-193, 1985.

14. Beam, R.M., and Warming, R.F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations", AIAA Journal, Vol. 16, No. 4, pp. 393-402, April 1978.

15. Whitfield, D.L., Janus, J.M., and Simpson, L.B., "Implicit Finite Volume High Resolution Wave-Split Scheme for Solving the Unsteady Three-Dimensional Euler and Navier-Stokes Equations on Stationary or Dynamic Grids", Mississippi State Engineering and Industrial Research Station Report No. MSSU-EIRS-ASE-88-2, February 1988.

16. Simpson, L.B., Unsteady Three-Dimensional Thin-Layer Navier-Stokes Solutions on Dynamic Blocked Grids, PhD Dissertation, Mississippi State University, December 1988.

17. Belk, D.M., Three-Dimensional Euler Equations Solutions on Dynamic Blocked Grids, PhD Dissertation, Mississippi State University, August 1986.

18. Anderson, W.K., Implicit Multigrid Algorithms for the Three-Dimensional Flux Split Euler Equations, PhD Dissertation, Mississippi State University, August 1986.

19. Benek, J.A., Buning, P.G., and Steger, J.L., "A 3-D Chimera Grid Embedding Technique", AIAA Paper No. 85-1523-CP, July 1985.

20. Rai, M.M., "An Implicit, Conservative, Zonal-Boundary Scheme for Euler Equation Calculations", NASA CR-3865, February 1985.

21. Janus, J M , and Whitfield, D.L., "A Simple Time-Accurate Turbomachinery Algorithm with Numerical Solutions of an Uneven Blade Count Configuration", AIAA Paper No. 89-0206, January 1989.

22. Janus, J.M., Advanced 3-D CFD Algorithm for Turbomachinery, PhD Dissertation, Mississippi State University, May 1989.

23. Arabshahi, A., A Dynamic Multiblock Approach to Solving the Unsteady Euler Equations about Complex Configurations, PhD Dissertation, Mississippi State University, May 1989.

24. Whitfield, D.L , Swafford, T.W., Janus, J.M., Mulac, R.A., and Belk, D.M., "Three-Dimensional Unsteady Euler Solutions for Propfans and Counter-Rotating Propfans in Transonic Flow", AIAA Paper No. 87-1197, June 1987.

25. Smith, L.H., Jr., "Aerodynamic Performance Tests of Unducted Fan Models", Advanced Turboprop Workshop Presentation, NASA Lewis Research Center, Cleveland, Ohio, Nov. 5-6, 1986.

Figure 1. Wing–Pylon–Store Configuration with Store in Captive Position

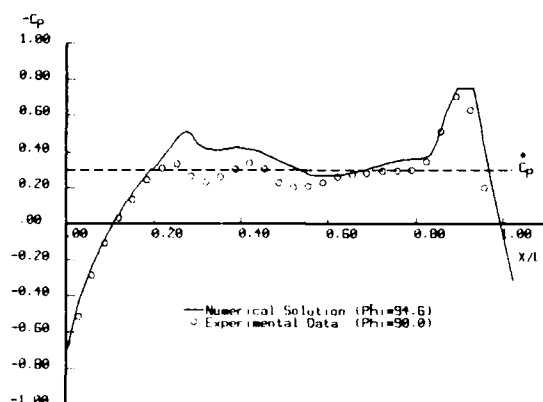Figure 2. Wing–Pylon–Store Configuration with Store Located Two Store Diameters Below the Pylon

Figure 3. Numerical and Experimental Surface Pressure Distribution on the Outboard Side of the Store M = 0.85, Alpha = 0.0
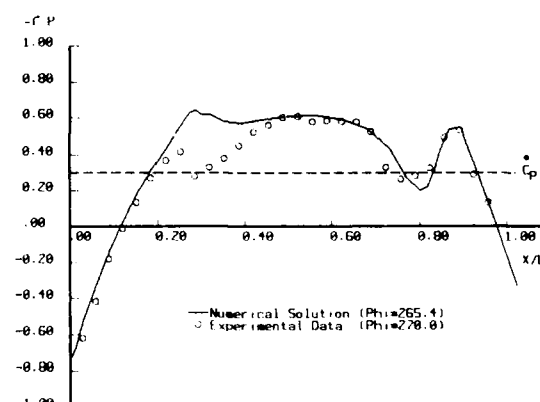

Figure 4. Numerical and Experimental Surface Pressure Distribution on the Inboard Side of the Store at M = 0.85, Alpha = 0.0
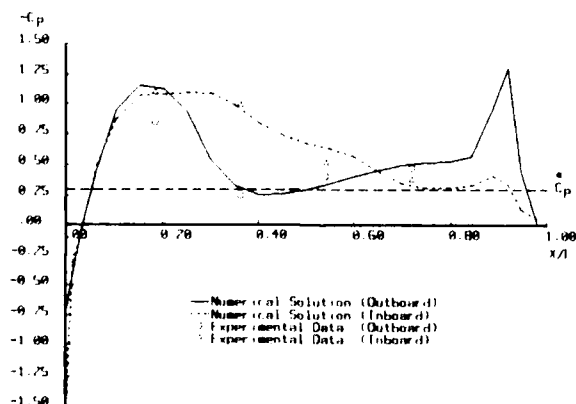

Figure 5. Numerical and Experimental Surface Pressure Distribution on the Pylon (Upper Row, Y = 1.17) at M = 0.85, Alpha = 0.0

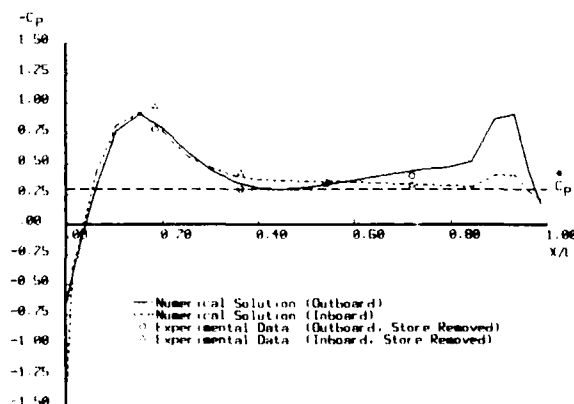
Figure 6. Numerical and Experimental Surface Pressure Distribution on the Pylon (Upper Row, Y = 1.17) at M = 0.85, Alpha = 0.0
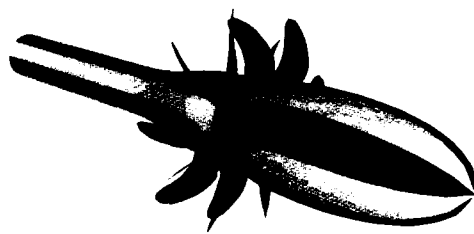
Figure 7. UDF8-8 Geometry

fore blade

aft blade

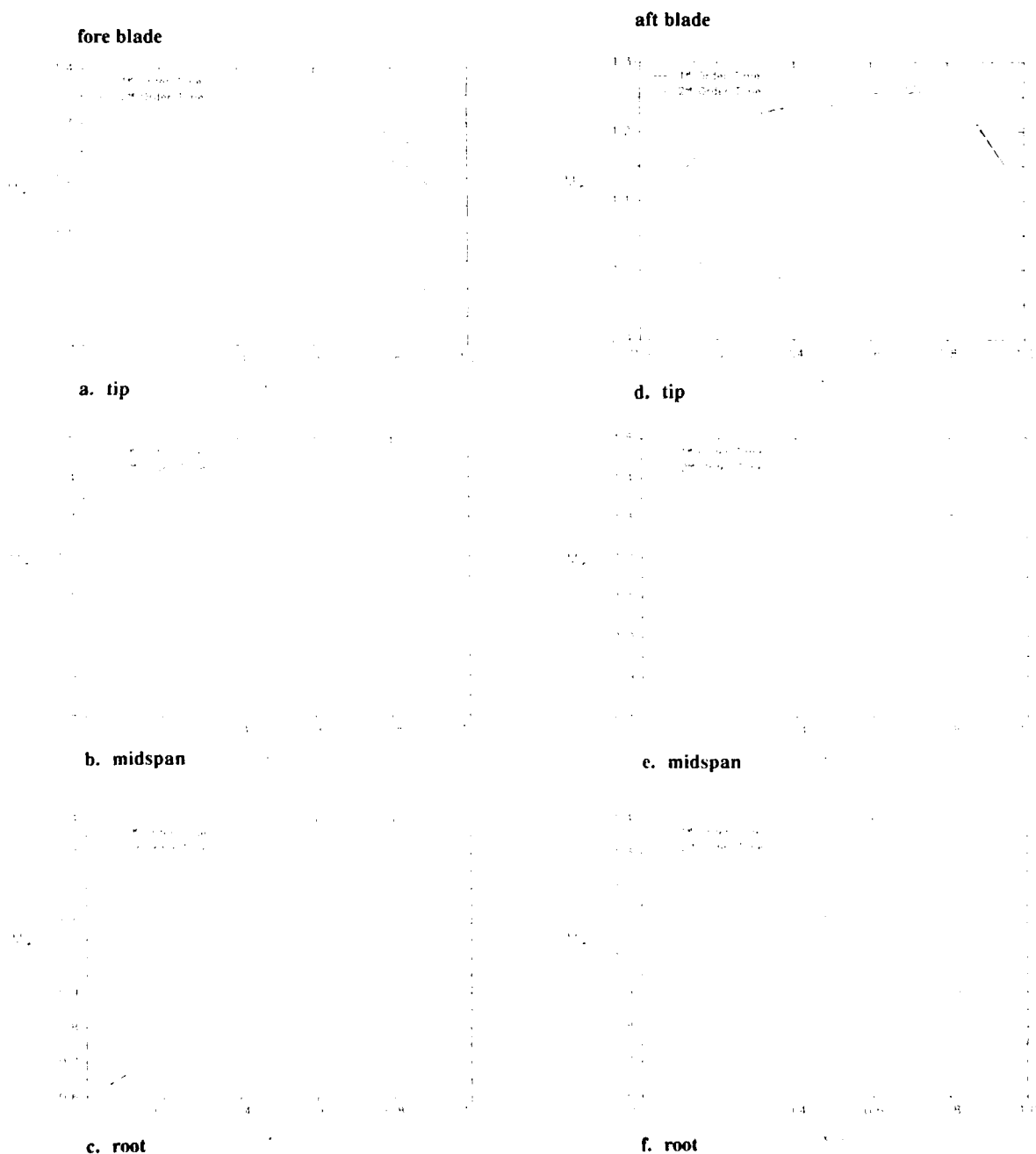a. tip

d. tip

b. midspan

e. midspan

c. root

f. root

Figure 8. FVS Blade Surface Local Relative Mach Number
(time-averaged)

fore blade

aft blade

a. tip

d. tip

b. midspan

e. midspan

c. root

f. root

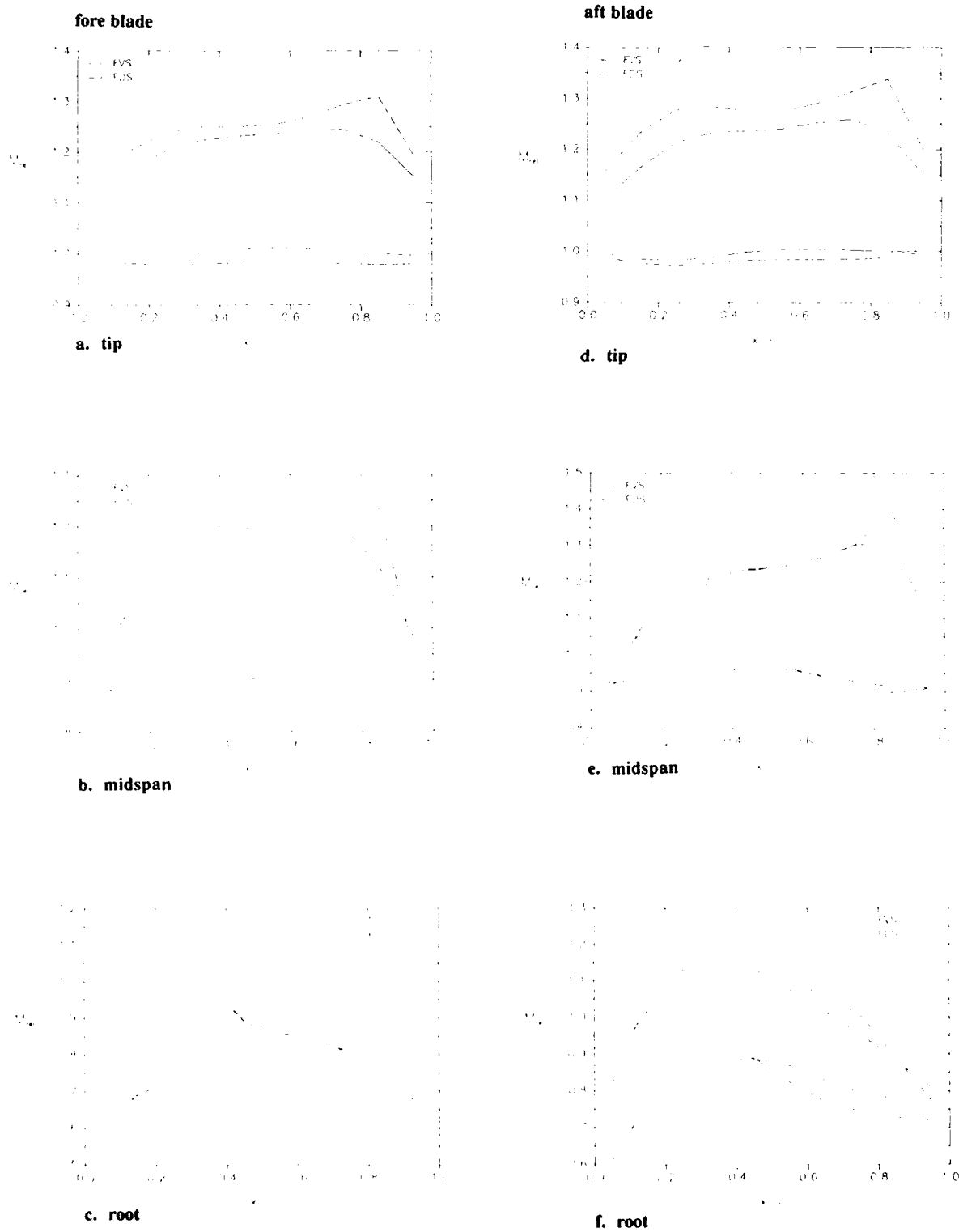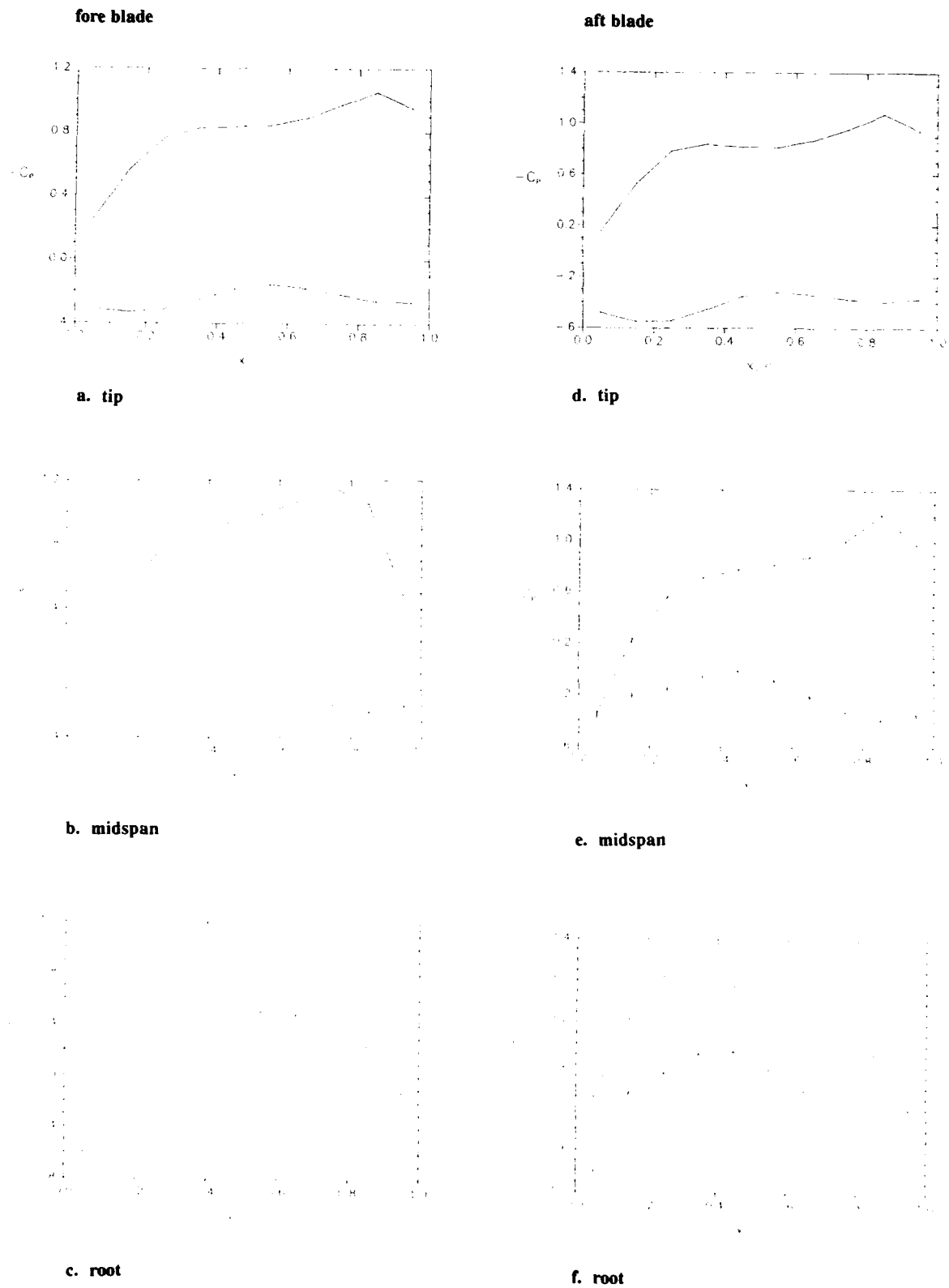Figure 9. Second-Order Time Blade Surface Local Relative
Mach Number (time averaged)

fore blade
aft blade

a. tip
d. tip

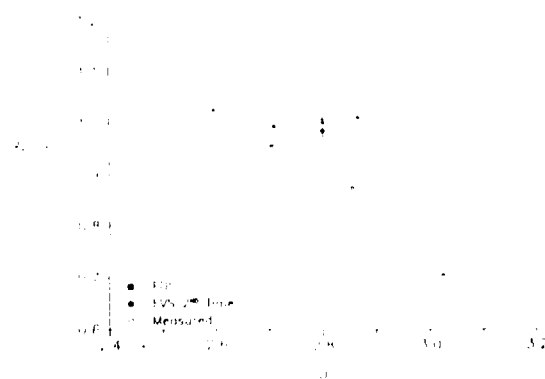b. midspan
e. midspan

c. root
f. root

**Figure 10. FDS Freestream Relative Pressure Coefficient
(time averaged with fluctuation bars)**
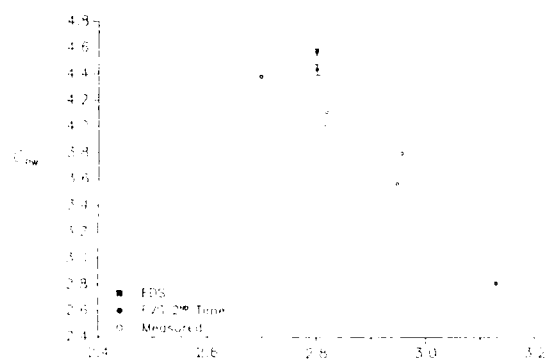
a. Power Coefficient
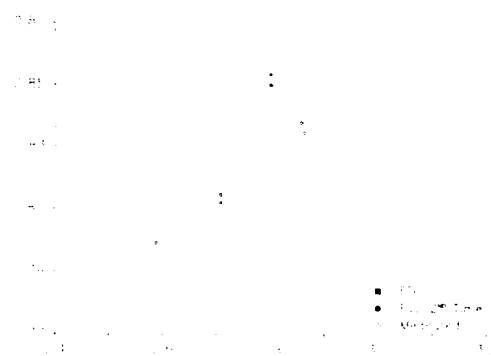
b. Efficiency

c. Torque Ratio

Figure 11. Integrated Performance Parameters
(time averaged with fluctuation bars)

A Structured Approach to Interactive Multiple
Block Grid Generation

J.P. Steinbrenner, J.R. Chawner and C.L. Fouts

Computational Fluid Dynamics Group
General Dynamics Fort Worth Division
P.O. Box 748, Fort Worth, Texas, 76101, USA

SUMMARY

The sheer variety of problems found in Computational Fluid Dynamics (CFD) has dictated a need for grid generation methods of the utmost generality. Experience has shown that user interaction and graphical feedback are two necessary features of a successful method as well. Employing these ideas, a structured method of grid generation has been developed, allowing a grid system to be constructed through the use of four specialized codes, accessed on two machines. These codes are based on the multiple block concept, whereby the flowfield domain is decomposed into a number of contiguous subdomains, allowing for efficient grid generation and flowfield solution. The first of these codes aids the user in inspecting the flow domain and in developing a suitable blocking strategy for the block system. A second code is then used to establish the exact connections between abutting blocks and to set flow boundary conditions on all surfaces of the block system. This connectivity and boundary condition data is accessed in the final two codes to construct the grid itself. The third code generates the surfaces of each block in the system, and the final code distributes grid points on the block interiors. The intricacies of these codes are explained along with an example, conclusions and projections for further work.

INTRODUCTION

Since its inception at the turn of the decade, the idea of multiple-block grid schemes has become so widely used and accepted that its justification hardly seems necessary today. Nevertheless, the advantages of a multiple-block scheme over a single block method deserve mention. The foremost advantage is that by reducing the flow domain to a number of subdomains, complex geometrical shapes may be modelled more easily and with greater numerical accuracy than with a single block. One need only compare the geometrical shapes generated routinely today with the more difficult single block configurations of a few years ago to realize the increased geometrical complexity that multiple block methods have permitted. Multiple block methods also increase the maximum total allowable size of the computational problem, since only one or a few blocks need reside in core memory at any given time, with the remaining blocks residing on an external device. This allows multiple block grids with millions of grid points to be generated and used on computers with enough internal memory for only a small percentage of the total grid size.

Multiple block flow solver methods have been developed and used at General Dynamics Fort Worth Division (GD) since 1985. Each Euler and Navier-Stokes solver in use at GD is structured with the same basic architecture (ref. 1, 2). Each solver is written for use on a Cray supercomputer, and each retains only a single block in memory at a time. To make full use of the vectoring capabilities on a Cray, the total grid system is normally divided into as few a number of blocks as possible, with each of the blocks being approximately equal in size as permitted. This constraint forces an added generality to the boundary conditions (bc's) - the allowance for several bc types on a given wall of a block. The solvers described above are used for a variety of computational flow analyses, and as might be expected, the configurations these codes see are as varied as the flow conditions, usually involving several geometrical length scales in a single problem. Furthermore, these analyses are performed in a design environment, such that rapid turnaround time is always a high priority.

In view of these constraints, a list of requirements for a multiple block grid generation package was not difficult to formulate. In 1985, the three most crucial requirements for a multiple block grid system were generality, speed of generation, and accuracy. Coupled to these was the need to apply the grid methods to all in-house CFD codes and the ability to specify boundary conditions on the cell, rather than face, level. These considerations served as guidelines for the initial development of the grid generation system (ref. 3). The requirement for geometric generality was met by making the codes highly interactive. In combination with the interactivity, extensive graphics capabilities were included to ensure the desired level of accuracy. By making the system interactive, the time-consuming duplication of effort associated with batch codes was avoided.

After over four years in development, GD now has a grid generation system which satisfies all of the requirements established in 1985. Today's software allows the user to generate multiple block grids for generalized configurations and generalized flow conditions. The volume grid file and the flow solver-specific boundary condition file created with the codes are directly accessed by the flow solver. The entire software package is shared between two computers - Silicon Graphics IRIS Workstations and CRAY supercomputers. On the IRIS, three codes have been developed for decomposing the

domain into blocks, specifying interblock connections and flow boundary conditions, and generating the block surface grids. Three separate programs were necessary due to the memory constraints of the IRIS, but the added overhead has not proven to be significant. The CRAY-based software consists of a single code for the generation of the block interior points. This code could equivalently be housed on the IRIS, but the need to run the flow solver on the CRAY and the CRAY's superior computational power make it the natural choice.

The intent of this paper is to outline the procedure routinely used at General Dynamics for the generation of multiple block grids. Successful generation is contingent upon delivery of a usable description of the configuration, and so in-house techniques for conversion of geometric models is explained first. Next, the four-step procedure, encompassing domain decomposition, connectivity generation, surface grid generation, and volume grid generation is overviewed, explaining both existing and developing capabilities. Each of these steps is applied towards the generation of a multiple block grid around the forebody of an F-16 for further illustration of the methods. Finally, an analysis of the current methodology is made to help project the future direction of this interactive grid generation system.

GEOMETRY PREPROCESSING

Computer aided design (CAD) has become a standard tool in the aerospace industry for geometrically defining developing configurations, and the majority of advanced designs which are used in CFD studies have indeed been created on CAD systems. In many of these systems, the geometry is represented as a combination of planar cuts and surface splines. At GD, several CAD packages with differing geometry representations are utilized daily on various projects. The differences in the various CAD systems, however, have made it necessary to establish one universal format for direct use in the interactive grid generation software. In the resulting format, the entire configuration surface is represented by a number of patches, each patch containing an m by n well-ordered array of physical points. We refer to each patch as a network, and to the collection of networks as a database. There is reasonable generality of the types of suitable networks, in that networks may overlap or abut with point, slope or no continuity. For ease of grid generation, however, it is desirable to have point continuous networks, although totally discontinuous database networks will not decrease the range of program applicability. Since spline equations are not stored in the resulting database, it is necessary to have a sufficient number of points in each network to resolve the important geometric properties of the configuration. A suitable network database for the F-16 forebody is shown in Figure 1. Fourteen networks are used to define this model, with computational dimensions varying from 3x10 to 28x25. Notice that the areas of higher resolution in this model generally correspond to regions of large surface curvature.
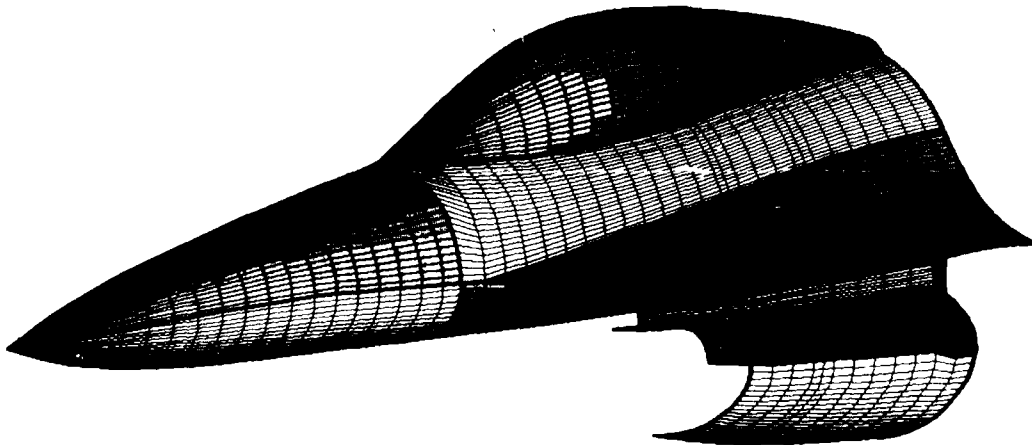
Figure 1. Fourteen Network Database about F-16 Forebody

Traditional methods for downloading CAD models to the correct grid generation format were at best cumbersome, and often required the CAD operator to sort data files by hand, to respline curves and surfaces, and to rely on visual inspection. It was not unusual for the operator to invest a week or more to accomplish this task. Consequently, a program was written in 1988 to streamline the creation of databases on the CATIA CAD system. In this batch code, a given surface model is automatically broken into networks, placing network edges at lines of intersection in the surface model. For simple

configurations, the user time has been reduced to an hour or so, with the total throughput time on the order of four hours. The database creation time for more complex configurations is not expected to be much more than that required for simpler configurations. Although this program is currently compatible only with the CATIA system, work is underway to extend it to other CAD systems as well. This capability will greatly enhance the interface between the design and the CFD groups.

DOMAIN DECOMPOSITION

Before a grid system can be generated, an appropriate blocking structure must be devised to divide the problem domain into smaller, more manageable sub-domains. Many factors are considered when decomposing the problem domain, including the computer hardware, the flow solver software, and the geometry. The maximum computational size of a block is restricted by the amount of core memory available on the computer. The optimal size of a block is also machine dependent. On a vector machine like the CRAY, a small number of large blocks is preferred; on a parallel computer, however, a large number of small blocks is more efficient. Furthermore, the manner in which blocks interconnect, such as overlapping or point to point matching, is flow solver dependent, also impacting the blocking strategy. Finally, geometric complexities of the configuration being analyzed may impose a practical lower limit on the number of blocks required; certain complex geometries just cannot be represented with only a few blocks.

Restrictions on the blocking structure imposed by geometrical complexities are typically difficult to visualize, and are therefore difficult to detect. This difficulty is usually tempered if a physical model of the configuration is available, but physical models are expensive, time-consuming to make, and are not easily modified to reflect geometrical changes. Although still in a development stage, GRIDBLOCK, the first of the three interactive IRIS codes, offers an alternative solution to this problem by providing the capability for real-time graphical manipulation of the 3-d configuration. More importantly, however, it provides an efficient means of investigating and generating blocking schemes.

A typical GRIDBLOCK screen is shown in Figure 2. Shown is the F-16 forebody database network with a preliminary blocking structure around the configuration. Both mouse and keyboard input control the options in GRIDBLOCK. The main options of the code are displayed in a menu on the lower left side of the screen. This menu changes during execution for continual display of currently available options. Above the menu area, a diagnostic window displays statistics pertinent to the current function. While defining lines, for example, the current 3-D coordinates of the cursor are displayed. The small window above the diagnostic window is used for typed user input such as file names and numerical data. The window above this is used to display step-by-step instructions for the current function, which leads the first time user through the program and serves as a reminder for the experienced user.
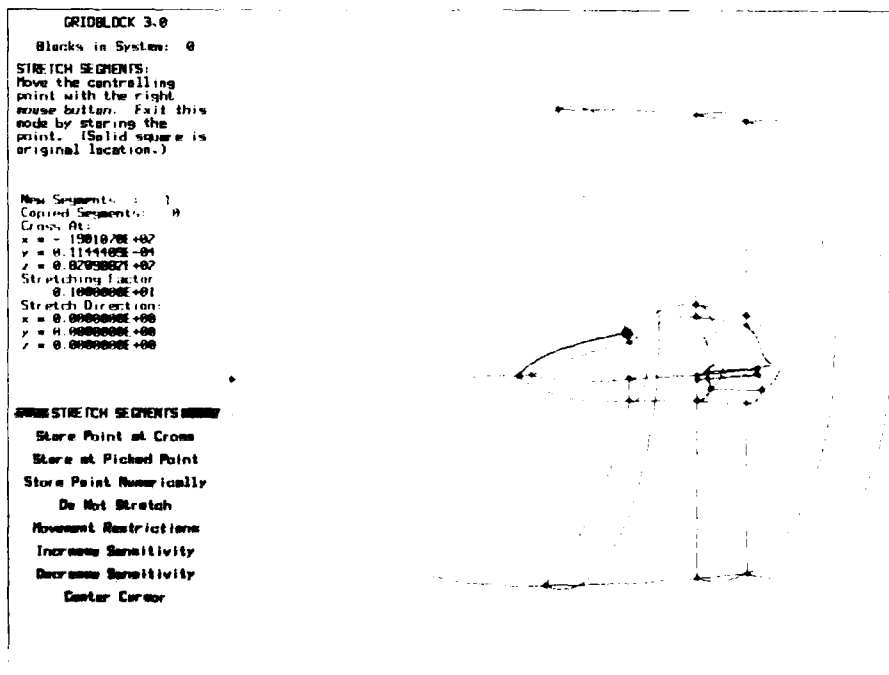

Figure 2. Typical GRIDBLOCK Screen

GRIDBLOCK begins with a display of the 3-D wire-frame model of the problem configuration, which is then rotated, scaled, and translated for further examination. As ideas for dividing the domain are formulated, three-dimensional lines representing the bounding edges of the blocks in the structure are drawn by the user about the database. These lines can be created in a variety of ways, and once created, can be further edited and modified as becomes necessary. It has been observed that a good understanding of the entire block can be derived by displaying only the edges of the block. This simplifies the work in GRIDBLOCK by eliminating the need to define and display surface shapes.

The most straightforward method of composing a multiple block structure in GRIDBLOCK is to build each block one edge at a time. These edges, called connectors, are composed of one or more line segments. Several types of line segments are available: straight lines, elliptic arcs, and smooth cardinal spline curves. These simple segments can be quickly combined to create virtually any 3-d curve. Each connector terminates in two nodes, one at each end. Any number of connectors may share a node, and connectors may be linked only at nodes. As a connector is defined, it is not immediately assigned to a block edge, but is simply a reference line, which is surprisingly helpful when first developing a blocking scheme. Connectors can later be edited (by moving individual points, stretching, translating, etc.) or even deleted. If a node location is changed, all connectors attached to that node are modified to maintain the connectivity. Nodes are drawn as large dots in Figure 2.

Once several connectors are added to the system, they may be grouped together and assigned to blocks. When twelve connectors are assigned to a given block, the user is prompted to define computational directions and dimensions on that block. Improper assignments are not allowed and are indicated to the user. For the F-16 forebody illustrated in Figure 2, the flow domain has been divided into a total of seven blocks.

Although the simplest, defining blocks one connector at a time is the most time consuming and monotonous mode of creation. Thus, work on GRIDBLOCK is underway to provide facilities for creating an entire block (i.e., all twelve edges) at once. Several basic shapes will be available, including rectangular blocks (for H-grids), half cylinders (for C-grids), and full cylinders (for O-grids). These shapes will then be translated, rotated, and scaled to position in the block structure. All commands available in the connector mode described above will be carried to this full-block mode, so that a rough block shape may first be generated, with further connector refinement performed later as needed. Again, when the block definition is completed, the user will be asked to supply computational coordinates and dimensions.

Many of the connectors defined in a typical system are shared by more than one block. A procedure is under development which will determine the user-drawn interblock connections automatically, and will normally be accessed after all blocks have been defined. Upon exiting the program, all block to block connections established automatically will be stored in a file referred to as the BOCON file. The file at this point will also contain cartesian definitions of the connectors and the composition of each block (connectors, dimensions, etc.). The set of data defining interblock connections is then the starting point for the second IRIS code, GRIDBOUND, which allows the user to complete interblock connections and to set the flow boundary conditions on all surfaces of the blocks. The cartesian connector information is later accessed in the surface grid generation routines, and the block composition data is used when restarting the GRIDBLOCK program. The manner in which the GRIDBLOCK code and BOCON files fit into the master plan of grid generation is explained in Figure 3.

CONNECTIVITY GENERATION

Once the domain blocking strategy has been developed, the straightforward yet tedious task of assigning flow boundary conditions and determining remaining interblock connections must be performed. This is essentially a bookkeeping task which is easily managed with the second IRIS code in the process, GRIDBOUND. Output from GRIDBOUND is the completed connectivity table (BOCON file), which is then used in GRIDGEN2D and GRIDGEN3D to maintain grid point continuity across connected block faces. After the multiple block system is completed, the connectivity data is converted in GRIDGEN3D to a format for direct use in the flow solver. (see the FLOWCON file in Figure 3)

Like GRIDBLOCK, GRIDBOUND is an interactive graphics code with input controlled through pull-down menus, making the program simple to use for even the novice. The user's initial task in GRIDBOUND is to specify the intended flow solver, so that the correct set of flow boundary conditions are made available in the interactive session. Currently three different solvers may be chosen within GRIDBOUND, but the effort required to permit additional solvers is minimal. As stated earlier, the GRIDBLOCK code is under development to write out a preliminary BOCON file containing interblock connections determined in that code. Currently, however, blocks are created in GRIDBOUND through pull-down menus designed for block definition, and each block is initialized with no boundary conditions.

All blocks are schematically drawn in GRIDBOUND in computational space. Figure 4 shows the block system after several different bc's of the seven block F-16 forebody have been specified. Note that bc types are differentiated in GRIDBOUND by color. Each block can be rotated independently so that each of the six faces can be examined.
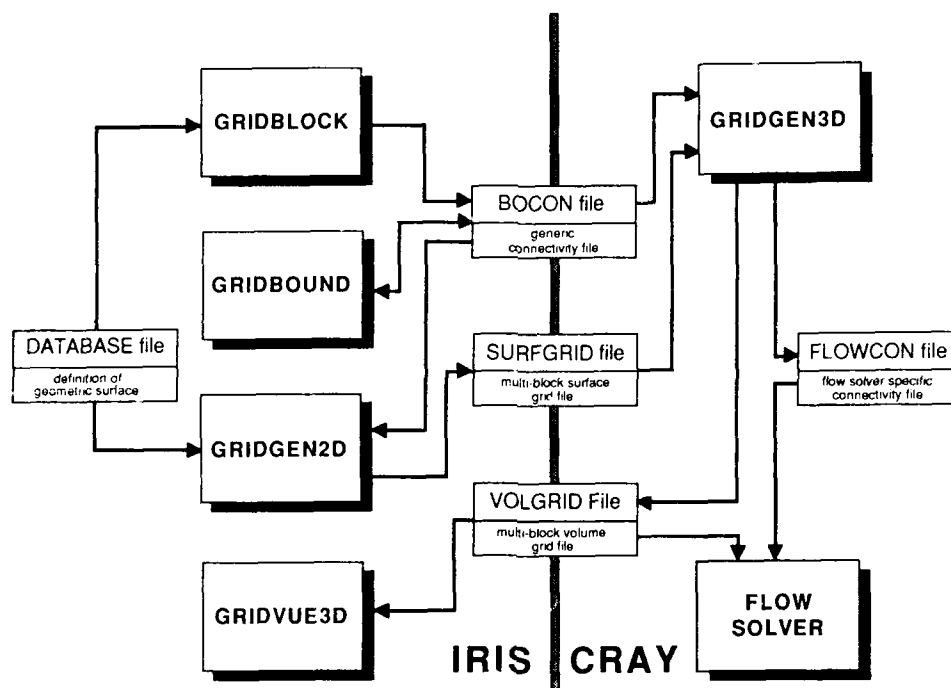
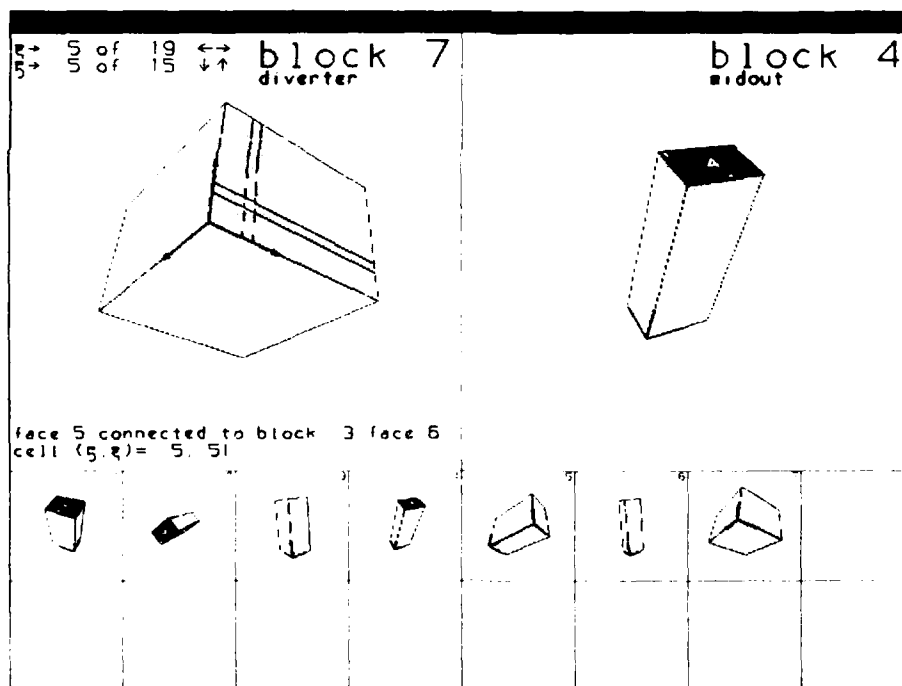Figure 3. Schematic Drawing of Grid Generation Process

Figure 4. Typical GRIDBBOUND Screen

Boundary conditions are set by selecting the block, the face number and the b.c. type for processing. The user is then prompted for the range of points to which that boundary condition is to be applied (the source range). The program checks this region, erasing any existing bc's and removing any connections (on both this block and the connecting block), before storing the new bc at each cell face in the region. For flow solver bc's, each cell in the source range is shaded with the corresponding bc color, and the specification is complete. For interblock connections, the intended range of points on the connecting face (the target range) is input, with the program rejecting improper connections. Allowable connections are controlled by the type of flow solver selected at the beginning of the problem. Currently, GRIDBOUND supports both point-to-point matching and ordered-subset matching (where one region is an ordered subset of the other), and connections may be established on the cell level and on the full face level.

GRIDBOUND provides several methods for ensuring that all necessary bc's have been specified as desired. High-light bars can be scanned over each face, with the cell bc at the intersection of the bars listed at the bottom of the block window. For connecting bc's, a cell cursor appears at both the high-lighted source cell and the corresponding target cells. By moving the high-light bars over the connection region, it is easy to verify that connections are oriented correctly. It is also possible to make all regions of a specified bc type blink. This tool is especially helpful for quick identification of regions with no set bc at all, which is usually the final check before exiting the program.

GRIDBOUND offers a number of block editing commands as well. Dimensions of existing blocks may be changed by adding or removing planes of points anywhere in the block. Blocks may also be split into smaller blocks, with connections at newly created surface points on the dividing plane set automatically. This allows blocks which are continuous in space to be generated as a single block, to be later divided to satisfy the equal block size objective. Blocks can also be added later or removed entirely. Changes such as these affect the overall blocking structure established originally in GRIDBLOCK, causing the structure to be no longer compatible with surface or volume grids created in the two codes described next. Therefore, a mechanism to forward the GRIDBOUND changes to GRIDGEN2D files has been included in GRIDBOUND.

SURFACE GRID GENERATION

The two IRIS codes explained above allow the user to establish the topology of the system, the interblock connections, and the flow boundary conditions. These codes may be thought of as grid generation preprocessors, since they do not determine actual grid point locations. The grid itself is generated in components, starting with block edges. The block edges are then used as boundary conditions for the block faces, and the resulting block faces are used as boundary conditions for the block interior. Both edges and surface grids are generated in GRIDGEN2D, an IRIS code initiated in 1985 (ref. 4), and the subject of this section. Since expanded to more than 40,000 lines of Fortran, GRIDGEN2D now has extensive plotting capabilities, and offers diagnostic windows to aid the user. A typical interactive window from GRIDGEN2D is displayed in Figure 5.

Starting

As illustrated in Figure 3, both the database and the BOCON file are used as starting points for surface grid generation. Upon choosing a block and a face to create, the user divides the face into any number of subfaces. Each subface may span a portion or the entirety of the face. The subface feature is demonstrated in Figure 6, which depicts the downstream planes of blocks three and four which abut blocks five through seven and the inlet face. The block 4 face has been divided into three subfaces, and note that three of the edges of subface 3 run along the inlet lip. By defining the edges of each subface, points interior to the entire face may be explicitly specified. The structure of the subface is easily modified, and in many cases, only a single subface is necessary.
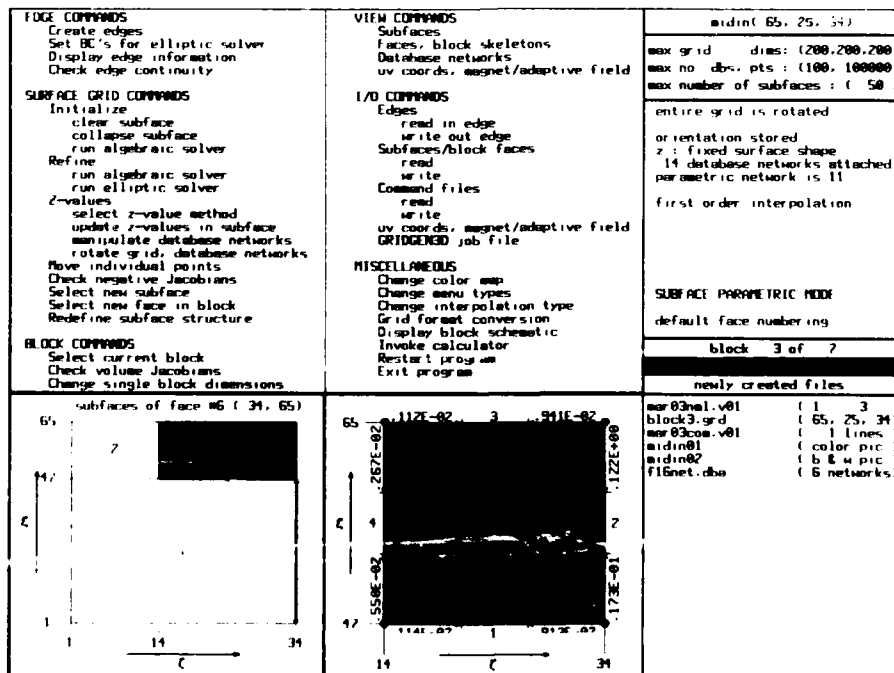
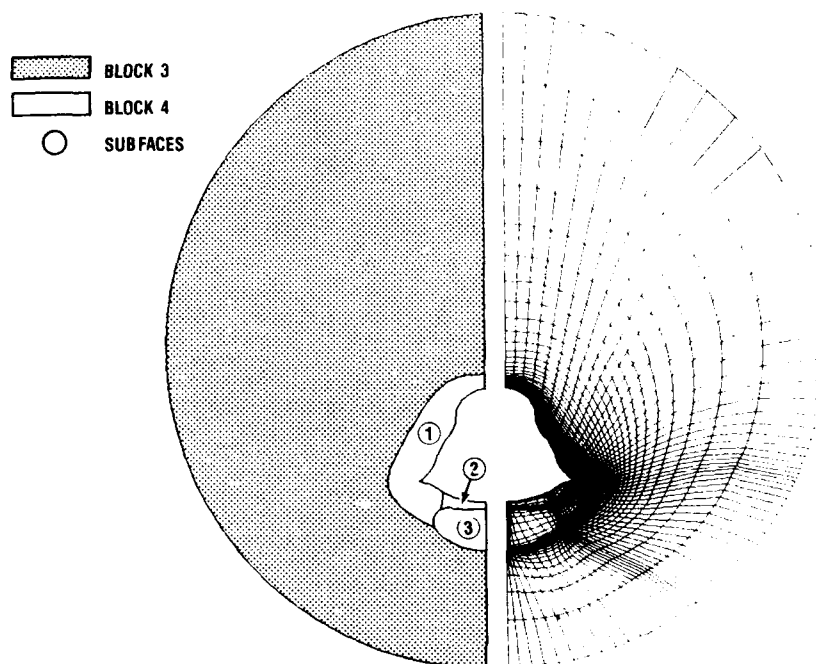

Figure 5. Typical GRIDGEN2D Screen

Figure 6. Faces of Blocks 3 and 4

Edge Distribution

Each of the four edges of a subface is generated before proceeding with interior point placement.   An edge is created in three stages: edge definition, breakpoint placement, and grid point distribution. The edge definition stage consists of defining the 3-d shape of the edge by piecing together a number of segments interactively defined with the keyboard or graphically. Available segment types include edges defined earlier by GRIDBLOCK, segments traced from the database, simple geometrical curves, curves stored in exterior files, user-defined curves in 3-d space, and user-defined curves constrained to a database or grid surface. The first of these segment types, which allows the edge definitions to be read in directly as defined in GRIDBLOCK, eliminates much of the duplication and confusion inherent with GRIDBLOCK to GRIDGEN2D transition. The last of these segment types is particularly useful when distributing interior surface points via the parametric elliptic modes, as explained later.

In stage 2, breakpoint distribution, the newly defined edge is broken into a number of subedges, separated at interactively-chosen locations referred to as breakpoints. Grid points are automatically placed at breakpoints and at the beginning and end points of the edge. Breakpoints provide a convenient means of clustering grid points at various locations along the edge.

In the final stage, the edge grid points (as specified by the block dimensions) are distributed between breakpoints as the user requires.   Initially grid points are distributed at equal arc increments, but the number of grid points and their relative distribution between breakpoints may be user-set. In addition to equal spacing, grid points along an edge may be distributed by either a two-sided hyperbolic tangent function (ref. 5), a one-sided geometric stretching function, by clustering to edge curvature, or by scaling a distribution function defined elsewhere in the grid. This latter distribution method is an easy way to maintain the same general clustering at opposite edges of a given face.

The entire edge generation process in GRIDGEN2D is menu-driven with a continuous graphics display. Any of the three stages above may be entered or reentered during edge construction. Graphical editing is added to allow for minor changes without duplication of effort, and on-screen help-menus are provided to assist the unfamiliar operator. The availability of linear, cubic and exponential splines (ref. 6) provides an additional degree of numerical control, and these edge splines may be applied to x, y or z coordinates, as well as to the arclength.

Interior Point Distribution

Grid points interior to a subface are initialized by interpolating from the subface's four edges.   Several algebraic methods are available for initialization, including transfinite interpolation with linear and arclength based (ref.   7) interpolants, and polar interpolation, which distributes points around a user-chosen axis in 3-d space.

Occasionally the algebraic initialization schemes are sufficient for the user's

purposes, but the majority of algebraic grids require further refinement. The initialization methods described above interpolate solely from the edges, and generally do not affix interior points on the intended shape of the surface, defined either by database networks, by interpolation, by planar surfaces, or by LaPlace's equation. When the surface shape is defined by database networks, therefore, the provisional surface must be interpolated onto the intended surface. In GRIDGEN2D, the interpolation proceeds in only the z physical coordinate, with the x and y coordinates held fixed. This means that it is often necessary to rotate the grid before the interpolation takes place, so that the optimal orientation of interpolation is aligned with the z physical axis. This rotation is done graphically so that the user can be confident of the chosen orientation for interpolation.

Interpolation of the z-values guarantees the correct surface shape, but makes no guarantees about the distribution of points on that surface. When the interior point distribution after interpolation is not acceptable, an elliptic solver is utilized. Elliptic solvers redistribute interior grid points to satisfy a set of elliptic partial differential equations (pde's), specifically a transformed set of Poisson equations. Solutions to elliptic equations of this form are generally smooth, and obey a maximum principle (no grid crossing) in the case of LaPlace's equation, so they are natural candidates for grid generation.

There are four modes of elliptic solvers in GRIDGEN2D. The first two solve directly for the new grid point locations (x, y and z) in an iterative process. For planar or nearly planar problems (where surface shape is not crucial, as in block interfaces), the 2-d equations of Thompson (ref. 8) are solved. For 3-d surfaces where shape is important, the modified Thomas (ref. 9) equations are solved. This latter set accounts for the curvature of the surface in the driving equations, essentially solving the equivalent Thompson equations on the 3-d surface. In each of these methods, only x and y values are calculated by the pde's, with z values updated through database interpolation (when applicable) as described above. Both methods also require that the surface first be rotated to a single-valued orientation in the z direction. When no suitable rotation exists, as in cylindrically shaped grids, it is necessary to divide the face into several subfaces, and to work on one subface at a time (see the beginning of this section).

The dissection of a given face into single-valued subfaces and the corresponding solution of the elliptic equations on these subfaces tends to be a tedious task. For this reason, two other elliptic solver types are included in GRIDGEN2D. Both techniques incorporate the same driving equations as above, but solve the grid in parametric coordinates, rather than in cartesian coordinates. These two parametric solvers employ a modified form of the equations popularized by Warsi (ref. 10), and differ only in their choice of parametric variables.

The easier to use of the parametric solvers we refer to as the subface parametric mode. Here, parametric coordinates are initially set equal to the computational indices of the existing face, and the elliptic equations are solved on this initial grid. This method is useful for subfaces which maintain the correct surface shape but have an unacceptable distribution of points on that shape.

The second parametric mode is referred to as the database parametric mode. In this mode, edges of the face are first generated in terms of the computational coordinates of a selected database network. With this technique, parametric coordinates are then interpolated onto the face interior, with Cartesian coordinates in turn interpolated from parametric coordinates. The elliptic solver is then run in a manner similar to the subface parametric mode. Although this solver is the most difficult of the four to use, the difficulty lies in understanding the concept, and not in the mechanics of running the solver. Because this elliptic solver mode requires that only one network be used to define the surface, it was necessary to equip GRIDGEN2D with an extensive database network manipulation routine. In this routine, networks may be scaled, translated, reduced and duplicated in a graphical, interactive environment.

The idea and mechanics of the GRIDGEN2D parametric elliptic solvers are explained in Reference 11. Because this method uses the standard Thompson (ref. 8) equations in a transformed parametric space, the resulting grid will be independent of the actual parametric representation. This is in contrast to other parametric methods (c.f. ref. 12) employing an abbreviated form of the parametric equation, which require a smooth distribution of parametric coordinates in physical space to insure a smooth surface grid. The biggest advantage of parametric solvers is that there is no need to rotate the surface grid to a single-valued orientation, as is often the case for the standard elliptic solvers. This permits the very easy generation of generalized surfaces. Drawbacks of the method are that set-up time is a little longer, and the numerical scheme is slightly slower.

Embedded in the standard elliptic grid generation equations is a set of functions which influence the distribution of grid points on the interior. Various forms of these control functions have been proposed over the years, and three of the more proven methods have been incorporated into GRIDGEN2D. These methods include those of Thomas and Middlecoff (ref. 13) and Sorenson (ref. 14) and each method is extended in GRIDGEN2D for use on 3-d surfaces, rather than 2-d planes. The third method backs out the control functions from the existing grid, and then smooths the values, so that kinks in the original grid will be removed while maintaining the important features of the grid.

When engaging the subface parametric solver described above, it is possible within GRIDGEN2D to enhance the control functions for a further degree of grid point control. By attaching a solution array obtained from an external source (such as a flow solution), grid points may adapt towards gradients in the solution by the method described in Reference 15. A magnet function may also be applied, in which the user graphically defines curve and point magnets on the surface where additional grid point clustering is desired. Both of these methods create components which are added to the control functions described above.

Several additional features in the elliptic solvers deserve mention. First, there is an interactive display of the surface grid during the execution of the solver, allowing the user to visualize the grid's iterative convergence. Further, the view of the grid may be manipulated between iterations. Secondly, any number of subfaces may be loaded into the elliptic solvers at a time, so that the entire face may converge uniformly. Finally, edge boundary conditions other than Dirichlet (fixed) may be selected for each edge of each subface. Orthogonality conditions will slide edge points along the original shape to enforce an orthogonal intersection with the edge. Abutting conditions allow for differencing across edges which are adjacent in physical space but not in computational space. This latter feature, along with the multiple subface solving is equivalent to a multiple block capability on the surface, rather than the volume level.

Finishing Up

The surface grid generation procedure explained in this section is repeated for each subface in the face, and for each face in the block. When a given block is completed, the next block to be generated is loaded into memory. In so doing, all interblock connections from the previous block are updated to the remaining blocks in the system. This procedure insures that each block interface need be generated only once. Once all faces of each block have been created, the surface grid generation is completed. Several faces of the completed seven block F-16 test case are shown in Figure 7. The final step in GRIDGEN2D is to save the multiple block surface file, referred to in Figure 3 as the SURFGRID file, and to run the GRIDGEN3D preprocessor. This preprocessor is accessed from the GRIDGEN2D menu and creates the job control language (JCL) and input data needed to start the volume grid generation on the CRAY.
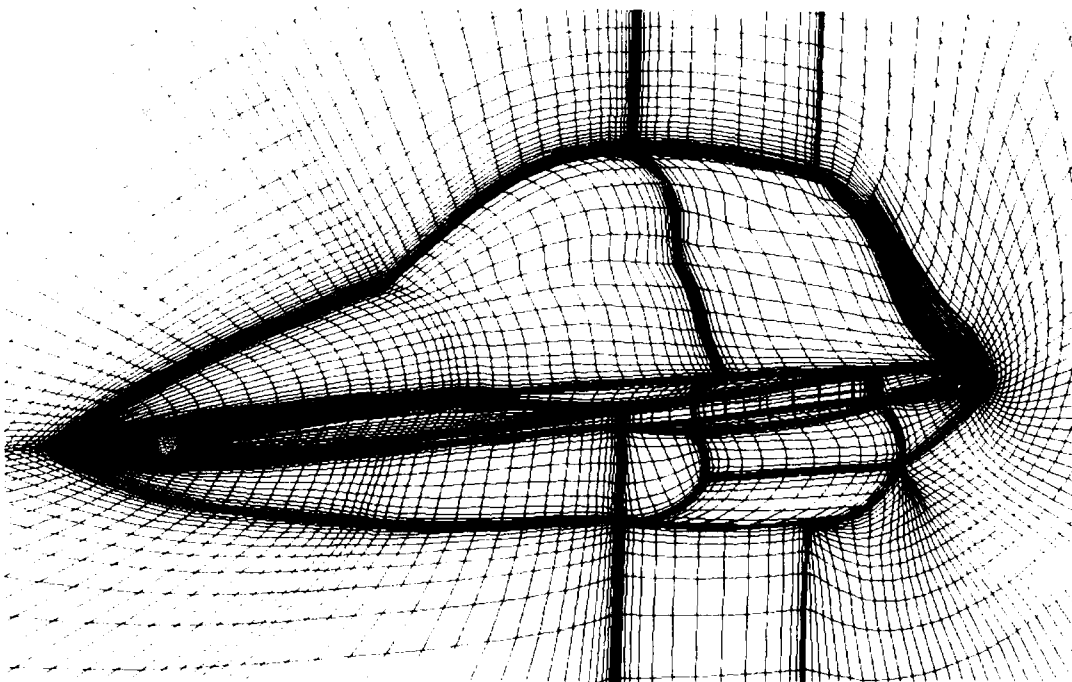

Figure 7. Completed Seven Block Grid about F-16 Forebody

VOLUME GRID GENERATION

The fourth and final step of the grid generation process is the distribution of grid points within the interior of each block. This task is performed with the batch procedure GRIDGEN3D, the only one of the four codes written for a CRAY supercomputer (see Figure 3). The philosophy behind the development of this batch procedure has been to utilize as much of the high speed and large core memory of the CRAY as possible, since these features aren't available on today's affordable workstations. Thus, the number of blocks is minimized in order to maximize the vector length of each calculation.

An additional benefit of using fewer and larger blocks is that the number of block interfaces is reduced. When the blocks sizes are large, however, the surface grids must be of sufficiently high quality, so that special treatment of the block interior points is not required. GRIDGEN3D, therefore, operates on the premise that the user has carefully utilized GRIDGEN2D to produce good surface grids. When a small number of blocks is used to discretize a domain, it also becomes necessary to allow for multiple boundary-condition and interface types on each face, which adds to the complexity of the code. We have considered this problem to be tractable, however, in light of the computational efficiency offered with large blocks.

Naturally other researchers have taken different approaches to volume grid generation. Seibert (ref. 16), for example, manages to perform edge, surface and volume grid generation within a single code. The advantage of this approach is clear; both overlap time and the transfer of files between codes is eliminated, greatly simplifying the process. On the other hand, Thompson's EAGLE code (ref. 17) was originally designed as a CRAY batch procedure for both elliptic and algebraic surface and volume grid generation. The advantages of EAGLE are that the batch procedure is more easily understood and applied, and that the computational time required for grid generation is reduced. The GD grid programs are a compromise between these two disparate approaches; the user intensive tasks such as surface grid generation are performed using interactive procedures on a workstation and the number crunching of volume grid generation is performed on a supercomputer using GRIDGEN3D.

User-friendliness is as much a part of GRIDGEN3D as it is a part of the interactive grid programs. In fact, it was easier to build user friendliness into GRIDGEN3D because it is a batch procedure with a uni-directional flow of operations. GRIDGEN3D was written in as general a manner as possible for easy application to a wide variety of configurations. No assumptions are made in the code concerning the grid topology or the orientations of the computational indices. A variety of robust grid generation methods, similar to those used in GRIDGEN2D, are available in GRIDGEN3D, and each method has been formulated to account automatically for point or line singularities. A great deal of the user friendliness of GRIDGEN3D comes from the minimal amount of input needed to compute a volume grid. As mentioned earlier, a preliminary GRIDGEN3D JCL file can be created while still in GRIDGEN2D. Default values for all inputs are based on data from the BOCON file and each input is checked for consistency with its related input variables. CRAY procedures have been written for the JCL to minimize the mechanics of running the code; only six or seven lines of JCL are currently required.

The grid generation methods in GRIDGEN3D are the volume equivalents of the surface methods employed in GRIDGEN2D. Algebraic transfinite interpolation with either linear or arclength (ref. 7) based interpolants is used to provide an initial volume grid. In many cases the volume grid produced using the arclength interpolants is sufficient (since the surface grids are generated with careful attention to quality) for flow calculations.

In the event that further refinement of the provisional volume grid is necessary, an elliptic pde grid solver is available in GRIDGEN3D. This solver may be run with any of four different control function types, in order to help the user meet his requirements for smoothness, clustering, and orthogonality. A smooth grid may be obtained using either the LaPlace control functions or by smoothing the control functions in the existing grid. The control functions of Thomas and Middlecoff (ref. 13) are used whenever interior clustering is required. Finally, the Sorenson control functions (ref. 14) can be used to maintain orthogonality and clustering at user-specified faces.

When running the elliptic solver, grid line slope continuity is maintained at block interfaces by moving the face, edge, and corner points where indicated by bc data from the BOCON file. To provide boundary point movement, a single layer of ghost points is saved around each grid block. On the portions of the faces corresponding to connections, ghost points take on the coordinate values of the grid points immediately inside the connecting block. This allows central differences to be used when grid points on the interface are moved. Edge and corner movement also uses the ghost point coordinates, but employs a one-sided approximation for the mixed derivative terms.

A parametric elliptic solver similar to the one in GRIDGEN2D has also been developed for volume grid generation (ref. 11). In this method, new grids are generated in terms of the computational indices of the original grid, which lends itself to adaptive grid applications quite naturally. To generate an adaptive grid, flowfield data from a partially converged steady state solution is used to form a control function that clusters grid points to flow field gradients, such as shock waves. This method is still under development, but is planned to be incorporated into GRIDGEN3D in the near future.

GRIDGEN3D reports the quality of the volume grid to the user by tallying the number of positive, skewed, and negative volume cells in the grid and by computing a measure of the grid quality as defined by Strang (ref. 18). Using this data the user can plan further GRIDGEN3D runs. The volume grid can also be transferred from the CRAY to the IRIS so that it may be visually examined using the GRIDVUE3D graphics program (see Figure 3). Within GRIDVUE3D, planes of constant computational coordinate may be scanned in real-time, providing a first hand confirmation of grid quality. A unique feature of GRIDGEN3D is its ability to translate the BOCON data into a boundary condition file (known as the FLOWCON file) for use with one of its associated flow solvers. Since setting up the boundary conditions is an error-prone and man hour intensive task, this feature of GRIDGEN3D results in a large time savings. This FLOWCON file and the resulting volume grid (VOLGRID) file are generally the only significant inputs needed to begin running the flow solver (See Figure 3).

FUTURE PLANS

The grid generation methods described above have proven to be effective tools in the application of CFD to the aircraft design process. For example, the sample grid shown throughout this paper could be generated today in about a week. Although this time is a vast improvement to the state of a couple of years ago, it is still significantly longer than that needed for complete integration of CFD into the design environment. In order to meet this goal, several improvements are planned for our methods. The near term improvements will fit within the four code framework we have established and are described below.

GRIDBLOCK, the most preliminary of the four codes, is considered a major key to faster turnaround. Improved drawing and edge manipulation commands are continually implemented. Nevertheless, complicated blocking structures are still difficult to visualize because of the complexities of typical aircraft geometries. Therefore, semi-automated domain decomposition methods are being investigated. Automated domain decomposition methods have already been proven (ref. 19) for certain applications, and it is believed that such methods in GRIDBLOCK would reduce time spent in that code considerably. Work is already underway to allow GRIDBLOCK to determine complete blocking structures automatically for certain classes of airplane topologies (e.g., wing, wing-body, etc.). This capability will be most instructive to the novice user, who may have no idea how to devise a suitable blocking scheme. After a blocking structure is automatically generated, the user can interactively modify the structure, adding refinements or changes as required.

The GRIDBOUND code has proven to be an easy code to understand and to use. However, since it is sometimes difficult to recognize blocks by their computational representations (see Figure 4), the GRIDBLOCK and GRIDBOUND codes will eventually be combined into a single code. The user will be able to toggle between physical $(x,y$ and $z)$ and computational representations of the multiple block systems. As edges are combined to form blocks in GRIDBLOCK, they will automatically be written to the BOCON arrays. By examining how edges are shared between blocks, the majority of block connections can be detected automatically by the code.

GRIDGEN2D is clearly the most extensive of the four codes used for multiple block grid generation. There are no specific improvements planned for GRIDGEN2D, although slight modifications are continually being made on user suggestions.

GRIDGEN3D will continue to be the batch volume equivalent of GRIDGEN2D. However, the volume grid generation process will be greatly simplified by removing the CRAY specific coding from GRIDGEN3D. This will allow GRIDGEN3D to be run on a workstation along with the three other grid codes, eliminating the need for file transfer and conversion procedures. The simplification of the overall four step grid generation process should more than make up for the loss in computing speed. In order to enhance the grid generation procedures within GRIDGEN3D, the adaptive methods described in the body of this paper will also be incorporated into the code.

The long term plans for our grid generation methodology are much harder to project because they are dependent upon advances in computer hardware and software. Already the dividing lines between the codes are overlapping. For example, definition of edges can be performed both in GRIDBLOCK and GRIDGEN2D. Eventually, this gray area will be removed by combining the four codes into one unified, interactive code. This implies, of course, that workstations will have grown enough to allow core memory to contain all of the blocking, connectivity and grid point data simultaneously. Also, the workstation CPU's will have to become fast enough to make interactive volume grid generation practical. A single unified code will significantly streamline the process, allowing the user to maintain his train of thought by eliminating much of the dead time needed now for transfer of files between codes.

As intimated earlier, the biggest challenge in generating a multiple block grid system is in suitably decomposing the domain into blocks. In fact, in many applications, a sizable portion of the grid generation process can be automated once the blocking structure is determined. It will therefore be prudent to investigate methods for automating the domain decomposition process as well. Artificial intelligence (AI) methods continue to gain in popularity and applicability, and have recently been used for two-dimensional domain decomposition (ref. 20). Progress in this field is certainly worth watching carefully. AI methods may also find their way into the volume grid generation arena, whereby an expert system could control the elliptic pde solver to obtain grids which meet or exceed predefined quality measures. Finally, as new control function formulations, pde solution algorithms, and adaptive grid strategies are developed, they will be incorporated into the code.

CONCLUSION

A structured, four code approach to multiple block grid generation has been developed for use in an aircraft design environment. Three of the four codes are interactive graphics procedures which give the user the ability to decompose a flowfield domain into multiple blocks, specify inter-block connections and flow solver boundary conditions, and generate grids on each of the six faces of each block. Generation of grids on the interior of each block is performed using a batch procedure that is run on a

supercomputer. These four codes generate multiple block grids for use in solving the full Navier-Stokes equations about complex aircraft. Their greatest utility is their emphasis on interactivity and graphical feedback which allow high quality grids to be generate with a minimal effort. Improvements are continually sought to increase the speed of the grid generation process. Most of these improvements involve methods of grid generation automation, and are expected to reduce the time needed to develop a multiple block grid system even further.

REFERENCES

1. Karman, S.L., Jr., Steinbrenner, J.P., and Kisielewski, K.M., "Analysis of the F-16 Flow Field by a Block Grid Euler Approach", AGARD-CP-412, 1986.

2. Reed, C.L., and Karman, S.L., Jr, "Multiple-Block Grid Method Applied to Complex 3D Geometries", presented at SIAM 1986 National Meeting, July, 1986.

3. Steinbrenner, J.P., Karman, S.L., Jr., and Chawner, J.R., "Generation of Multiple Block Grids for Arbitrary 3D Geometries", AGARD-AG-309, 1988.

4. Steinbrenner, J.P., "GRIDGEN2D, Interactive Elliptic Surface Grid Generation", General Dynamics report No. CFD 063-4-8601, June, 1986.

5. Vinokur, M., "On One-Dimensional Stretching Functions for Finite-Difference Calculations", NASA CR-3313, October, 1980.

6. McCartin, B.J., "Theory, Computation, and Application of Exponential Splines", DOE/ER/03077-171, October, 1981.

7. Soni, B.K., "Two- and Three-Dimensional Grid Generation for Internal Flow Applications of Computational Fluid Dynamics", AIAA 85-1526, 1985.

8. Thompson, J.F., Thames, F.C., and Mastin, C.M., "Automatic Numerical Generation of Body Fitted Curvilinear Coordinate Systems for Fields Containing any Number of Arbitrary Two-Dimensional Bodies", Journal of Computational Physics, Vol. 15, 1974.

9. Thomas, P.D., "Composite Three-Dimensional Grids Generated by Elliptic Equations", AIAA Journal, Vol. 20, 1982, pp. 1195-1202.

10. Thompson, J.F., Warsi, Z.U.A., and Mastin, C.W., Numerical Grid Generation Foundation and Applications, North-Holland, 1985.

11. Steinbrenner, J.P. and Anderson, D.A., "Three-Dimensional Parametric Block Grid Regeneration with Localized Solution Adaption", Numerical Grid Generation in Computational Fluid Mechanics '88, S. Sengupta et.al., eds., Pineridge Press, 1988.

12. Weatherill, N.P., Forsey, C.R., "Grid Generation and Flow Calculations for Complex Aircraft Geometries Using a Multi-Block Scheme", AIAA-84-1665, June, 1984.

13. Thomas, P.D. and Middlecoff, J.F., "Direct Control of the Grid Point Distribution in Meshes Generated by Elliptic Equations", AIAA Journal, Vol. 18, 1979, pp. 652-656.

14. Sorenson, R.L., "A Computer Program to Generate Two-Dimensional Grids about Airfoils and Other Shapes by the Use of Poisson's Equations", NASA TM-81198, 1980.

15. Anderson, D.A., Steinbrenner, J., "Generating Adaptive Grids with a Conventional Grid Scheme", AIAA-86-0427, Reno, Nevada, January, 1986.

16. Seibert, W., "A Graphic-Interactive Program-System to Generate Composite Grids for General Configurations", Numerical Grid Generation in Computational Fluid Mechanics '88, S. Sengupta et.al., eds., Pineridge Press, 1988.

17. Martinez, A., Mounts, J.S., "Program EAGLE Numerical Grid Generation System User's Manual", AFATL-TR-87-15, Vols. I, II, and III, Air Force Armament Laboratory, 1987.

18. Strang, W.Z., "QBERT: A Grid Evaluation Code", AFWAL-TM-88-193, Air Force Flight Dynamics Laboratory, July 1988.

19. Allwright, S.E., "Techniques in Multiblock Domain Decomposition and Surface Grid Generation", Numerical Grid Generation in Computational Fluid Mechanics '88, S. Sengupta et.al., eds., Pineridge Press, 1988.

20. Andrews, A.E., "Knowledge-Based Flow Field Zoning", Numerical Grid Generation in Computational Fluid Mechanics '88, S. Sengupta et.al., eds., Pineridge Press, 1988.

ACKNOWLEDGEMENT

DESIGN AND TESTING OF A MULTIBLOCK GRID-GENERATION PROCEDURE
FOR AIRCRAFT DESIGN AND RESEARCH

J.W. Boerstoel, J.M.J.W. Jacobs, A. Kassies
National Aerospace Laboratory NLR
P.O. Box 90502
1006 BM AMSTERDAM; The Netherlands

A. Amendola, R. Tognaccini, P.L. Vitagliano
Aeritalia/Gruppo Aerei da Trasporto/Ufficio di Aerodinamica
80038 Pomigliano d'Arco (Napoli); Italy

Summary

A multiblock grid-generation procedure embedded in a numerical flow simulation system is described. Major features of the grids are: suitable for complex aerodynamic configurations; grid lines continuous, in particular, over block faces; grid lines not slope-continuous over block faces; topology and geometry of block decomposition first specified, and then grid-point distributions; application of transfinite interpolation and elliptic techniques.

It is possible to construct multiblock grids around complex configurations with 250-1000 blocks, and to compute (Euler) flows on such grids.

New technical concepts are proposed, to improve the accuracy of the flow simulation results, and to reduce manhour investments in the construction of multiblock grids. These concepts concern a. the use of compound faces and edges, and b. the application of grid refinement per block and per coordinate direction, to remove the constraining effect of grid-line continuity on grid-point-density control; c. the use of new techniques for analytic aerodynamic geometry modeling, to reduce the dependence on non-CFD geometry software packages; d. the control of grid quality and acceptability with weight functions in the independent variables of the 3D vector functions defining the geometrical shape of edges, faces and blocks, and e. use of hyperblocks to speed up the block decomposition.

1. Introduction

The purpose of this paper is to present the current status of the grid-generation research and development work at NLR and AIT, to report about practical experiences with the current grid-generation procedure, and to outline improvement plans now in execution.

The aim of the current grid-generation activities at NLR and AIT (NLR cooperates also closely with Fokker) is to provide computer codes for the construction of grids for Euler-flow calculations around 3D complex aircraft configurations. Propeller-slipstream simulation for transport aircraft is of particular interest. Ensuring growth-potential of the grid-generation codes for NaS (Navier-Stokes) flow calculations is also of importance.

The material of this paper is presented as follows.
In section 2, grid-generation is analyzed as a subtask in CFD calculations for industrial aerodynamic design tasks or for aerodynamic research tasks. Here, user-requirements for a grid-generation procedure are analyzed from this viewpoint.
Section 3 is a technical description of our current grid-generation procedure.
Section 4 is a report of important experiences with this grid-generation procedure.
Section 5 is a summary of new technical concepts that may be used to improve the grid-generation procedure.
Section 6 is a summary of conclusions.

The grid generator described here produces multiblock grids, and not unstructured grids. The choice issue involved here, is briefly discussed in section 2.4 (Rationalisation of technical choices), and in the references mentioned there. Further, our expectation that algebraic mesh tuning techniques can perhaps be made more efficient than elliptic mesh tuning techniques is discussed in section 5. Adaptive grid-generation is not discussed here, except that from the remarks in section 5 conclusions can be drawn about the way how growth potential in this direction is secured.

2. The design of a grid-generation procedure

2.1 Grid-generation as a subtask in CFD calculations

The construction of grids is the first subtask in a numerical flow simulation. In our case, the subtasks of a numerical Euler (or NaS)-flow simulation are those of figure 2.1,
.   topological block decomposition
.   geometrical block decomposition   grid construction around a given aerodynamic configuration,
.   generation of multiblock grid
.   flow simulation, and
.   visualisation of flow.
The grid-generation procedure is thus subdivided in three closely related subtasks. At the input side, this procedure accepts arbitrary aerodynamic configurations, e.g. aircraft configurations for propeller-slipstream simulations. At the output side, grids that can be accepted for numerical flow simulations have to be produced.

2.2 User-requirements modeling for grid generators

When a grid-generation procedure must be designed, the first step is to carefully prepare a list of user-requirements for this procedure. This user-requirements list is a basis for the formulation of rational balanced answers to a variety of questions. These user-requirements are a specification of:

- which aerodynamic grid-generation <u>functions</u> have to be possible with the grid-generation procedure.
- what classes of aerodynamic configurations are specified as <u>input</u> (e.g. aerofoils, wings, aircraft parts, complete aircraft), how these are defined (e.g. point collections with/without interpolation rules), where they are generated, etc..
- which structure and properties the grids should have that are produced at the <u>output</u> side, how their <u>quality</u> is defined and measured.
- what <u>acceptance criteria</u> must be satisfied (e.g. manhour investments required for the construction of a grid of acceptable quality, grid-smoothness properties, etc.).

The analysis of these user requirements involves an estimate of what <u>technical resources</u> are available to implement the required functions in a grid-generator code. These resources are e.g. numerical grid-generation methods, and available computer/network/workstation configurations.

2.3 User-requirements for grid-generators

Figure 2.1 illustrates that, at the <u>input</u> side of the grid generation task, one has to obtain given aerodynamic surface configurations, including those for complete aircraft. These surfaces are smooth, except at edge lines, and a few vertex points.

Aerodynamic-surface data can be delivered (via file interfaces) from a large variety of sources, e.g. data bases in different software packages (SIGMA, AEROLIS, GAMMA, COMPVIS, CATIA, PATRAN), which are operational on different computer systems with different operating systems. Further, these data on files are usually in various formats. This wealth of possible input sources has requirements as a consequence.

- <u>Any aerodynamic 3D configuration</u> must be acceptable as input. In particular, the grid generator code must have excellent means for the representation of the geometric shape of aerodynamic aircraft surfaces.
- This input must be acceptable from any <u>source</u>, via a <u>standard input file</u>.

This standard input file has a very simple format, so that writing small conversion programs for bringing geometrical data into the format of this file requires negligeable effort.

At the <u>output</u> side of the grid-generation task, multiblock grids are produced. User requirements for this output are as follows.

- The grid boundaries should approximate with sufficient accuracy the <u>geometric shape of the flow boundaries</u>.
- The <u>quality</u> of the grids should be such that <u>discretisation errors of flow-simulation results</u>, which are due to the grid, can be made smaller than an arbitrary user-specified upper limit.
- The grid-generation procedure should be <u>loosely coupled</u> to the flow simulators (Euler, NaS).

Starting point for a definition of <u>grid quality</u> and <u>acceptability</u> is the control of discretisation errors of flow-simulation results. Ideally, discretisation errors should be smaller than user-defined upper-limit values, and the grid should allow this. This leads to quality and acceptability criteria for grid data, like

. <u>mesh sizes</u> (in principle, it should be possible to make mesh sizes arbitrarily small, by <u>enrichment</u>),
. <u>grid-variation</u> (including slenderness and shewness variation), and
. <u>local grid smoothness</u> in the flow domain.

Two conclusions follow from simple asymptotic error analyses of discretisations of equations.

- To permit, that local discretisation errors can be made smaller than any small predefined upper limit, <u>enrichment</u> (increase of the total number of grid points in a fixed flow volume, e.g. a block) has to be possible, to any degree. A grid is said to have <u>sufficient quality</u> if, everywhere on the grid, the local discretisation errors can be made smaller than a small predefined limit value.
- Further, for a given fixed total number of grid points in a given fixed volume, the grid is acceptable from the point of view of discretisation efficiency, if the local cell shapes vary over the volume in such a way that a roughly uniform distribution of discretisation errors of simulation results is obtained, because usually this will be optimal. Optimal error variation in a given flow domain volume for a given number of grid points is thus an <u>economic acceptability criterion</u>.

The notion of local grid smoothness deals with the conditions under which discrete equations are second-order accurate. Usually, during a consistency and error analyses of discrete equations, it is required that the grid is sufficiently smooth. It has to be so smooth that the local 2nd-order accuracy of discrete conservation equations and boundary conditions is not destroyed by too much variation of the grid. This sets limits to mesh stretching and a few other functions describing 2nd-order behaviour of the grid. When grids are varying, <u>sufficient grid smoothness</u> is thus a second <u>acceptability criterion</u>.

An other important <u>acceptability requirement</u> is that it should be possible to produce blocked grids around complete aircraft by an experienced CFD specialist in about two working days. This requirement is not yet met by us with our current means, see table 2.1.

subtask in Euler-flow simulation	(CODE)	NASA-Langley wing-nacelle-propeller	F100	F50	G222
. topological block decomposition	(PATRAN, GAMMA)	4 weeks	3 weeks	2 weeks	8 weeks
. geometrical block decomposition	(GAMMA, AEROLIS)	4 weeks	3 weeks	2 weeks	8 weeks
. generation of multiblock grid	(EGRID)	4 weeks	2 weeks	2 weeks	8 weeks
. one Euler-flow simulation	(ESOLV)	½ week	½ week	½ week	-
. one visualisation of results	(VISU3D)	½ week	½ week	½ week	-

Table 2.1   Rough estimates of amounts of manpower investments, spent to subtasks in executed Euler-flow simulations

This table explains one reason why our current development and research efforts are concentrated on improvement of the Euler-flow simulation system in the area of aerodynamic-geometry handling and grid generation. Grid design requires too much manpower compared to the other subtasks in numerical flow simulation.

Other <u>acceptability requirements</u> for the grid generator procedure are the following ones.
- Interactive options for the manipulation of aerodynamic surfaces in the grid-generation procedure (grid points on configuration surfaces should remain on those surfaces when their position is shifted).
- Tuning of the quality of acceptable grids on coarse grids on workstations, and tuning of corresponding fine grids in batch on a supercomputer.

2.4 Rationalisation of technical choices

Based on user requirements formulated in section 2.3 (in fact, these define only what input and output of the grid generation process is desired), a preliminary technical design of a grid generation procedure was prepared, [1,2]. Thereby, a choice between various alternative grid-generation techniques and related technical issues had to be rationalized. Our considerations and decisions were summarized in [3], see also [1].

2.5 Test library

To test the grid generation procedure under development, a library of test cases is defined. This library consists of a few aerodynamic configurations which complexity is estimated to be representative for what one may encounter in aircraft-design and in aerodynamic research environments. Included in this library are a few full transport aircraft configurations or important parts of them,
- the NASA-Langley propeller-nacelle-wing configuration, [4],
- the F100 and F50 aircraft configuration (c.f. [9]), and
- the AIT G222 configuration.
See figure 2.2 - 2.4.

3 Technical description of the grid-generation process

The grid generation procedure is based on the application of the following technical concepts.
1. Flow domains are made finite. Grids are made boundary conforming.
2. A multiblock approach is applied.
3. The block decomposition of the flow domain around a given 3D aerodynamic configuration is done in two steps.
   a. First, the topology of the block decomposition is defined. Purpose of this step is the construction of topology tables that describe how the vertices, edges, faces, and blocks in a block decomposition are connected to each other, and what are the positive directions of curvilinear 3D, 2D, or 1D, boundary-conforming coordinate systems in blocks, in faces, and in edges, respectively, see [5,6]. In this stage, it is possible to work with very rough geometric approximations of the true geometric shape of aerodynamic configuration surfaces, so that e.g. the geometry software package PATRAN can be used. (However, see the remarks in section 4.) Data sizes are relatively small, in this stage.
   b. In the second step, given these topology tables, the corresponding geometric shape of the blocks and its faces and edges, and the position of the vertices are defined.
   In particular, the geometric shape of block faces on the aircraft configuration are defined from the given input of the grid generation procedure. This geometric shape is stored in the form of function prescriptions (not: points to be interpolated in an unspecified way).
   Logically, we consider these two steps as strictly sequential, so that data for the geometric shape of block faces, edges, and vertices are defined on top of data for the topology stored in topology tables. Below it will become clear that this hierarchical data structuring is used to obtain simple geometry-definition procedures for large numbers of block vertices, edges, and faces in the flow domain.
4. Blocks are packed block-face to block-face, without gaps or overlaps. Algorithms in flow simulators for the coupling of flows over block faces may thus be based on 2D data structures, which offers good opportunities for vectorisation and parallellisation of algorithms.
5. The geometric shape of each block face (including those on given aerodynamic configuration surfaces) is defined by given functions of two parameters. These functions define smooth block-face surfaces. The function prescriptions are required in the grid generator, when grid points in given block-face surfaces are computed.
6. Transfinite interpolation is used to initialize grid-point distributions, see point 9 below.
7. Elliptic grid-generation, with user-controlled scaled dimensionless weight functions for mesh-size tuning, is used to tune a grid, see [1,3,5,6], and point 9 below.
8. Grid lines are made continuous over block faces and over block edges, but not slope-continuous, see figure 3.1. When the grid would be required to have more smoothness over block-faces, this would simplify the grid-generation and flow simulator algorithms, because a number of special algorithms at block faces are then no longer needed. However, in such cases, the control of the acceptability and quality of the grid near corners like that of figure 3.2 will usually become a problem. A rigorous solution is not to require slope-continuity. The corresponding complications in the numerical algorithms in the flow simulator can be solved by standard numerical techniques, [7,8].
9. A further advantage of requiring only grid-line continuity over block faces and face edges is that the grid-generation procedure may be decomposed into a sequence of three substeps. This decomposition offers good options for controlling grid quality and acceptability (as defined in section 2.3) locally, in a sence made precise in point 10.f. The substeps are as follow.
   a. First, the grid in the interior of each block-edge curve is constructed, with the two given vertex points kept fixed. For each edge, a 3D vector function defining the geometric shape of the edge curve, is given. The grid points on the edge interior are usually defined by an elliptic technique. Thereby the grid points are shifted along the edge curve to desired locations, specified by a given, user-defined, positive weight function $w(\xi)$.
   b. In the second step, the grid in the interior of each block-face surface is constructed, with the grid in each edge curve known now. For each face surface, a 3D vector function defining the geometric shape of the face surface, is given. To obtain an initial grid in the face, transfinite

bilinear interpolation in the two independent variables of this vector function is applied. Thereby the independent-variable values at the given grid points on each of the four face edges are bilinearly interpolated to independent-variable values at interior grid points in the face. These are subsequently substituted in the vector function prescription to obtain the grid points on the face. Usually, in a large percentage (say, 80%) of the block faces, this transfinite interpolation procedure produces in faces acceptable grids of sufficient quality.

When the grid in a face produced by the transfinite bilinear interpolation is unacceptable or of insufficient quality, elliptic mesh tuning is subsequently applied to define in the face the grid point distribution, using two partial differential equations (these are particular forms of spring-analogy equations) to define the required independent-variable distributions.

   c. In the third step, the grid in the interior of each block volume is constructed, with the grid in each of the six blocks faces known from step b. In each block, to obtain an initial grid, trilinear transfinite interpolation in the given grid points on the six block faces (defined in step b) is applied, with the computational coordinates $(\xi, \eta, \zeta) \in [0,1]^3$ in the block volume as independent variables. Usually, this produces in nearly each block acceptable grids of sufficient quality. In blocks, where this is not the case, the grid points are defined by solving subsequently three elliptic partial differential equations (c.f. [3,6,7] for details about their definition).

10.Analysis of the structure of this 3-step grid-construction process, first in edges, then in faces, and finally in blocks, shows that various easy understandable mechanisms are available to control the acceptability and the quality of the multiblock grids to be produced.

   a. For a given block-decomposition topology, the position (in the flow or on the aerodynamic configuration) of each vertex point can be arbitrarily defined within usually wide limits.

   b. The geometric shape of each edge curve segment can be arbitrarily defined, between its two vertex points. In our algorithms, a straight line segment between the two vertex points is the default edge shape.

   c. The geometric shape of each face surface segment can be arbitrarily defined, between its four edge curves. In our algorithms, a bilinear transfinite fit between the four edge curve segments (these can thus be completely arbitrary continuous curves) is the default face shape.

   d. Given the topology of a block decomposition of a flow domain, there is thus much flexibility in using the block-decomposition geometry for the control of the acceptability and quality of the multiblock grid constructed on top of it.

   e. Because it is required that grid lines are continuous over block faces and over face edges, there exist tight relations between the total number of grid lines in the various computational coordinate directions in the blocks, the faces, and the edges, which constrains the control over grid quality.

   f. However, in each edge, for each given total number of grid points in the edge, the distribution of the grid points can be completey freely chosen. This offers thus good options for local mesh tuning in edges.

Similarly, in each face, the distribution of the grid points in the face interior has only to be matched smoothly to the grid point distributions in the four edges. And in each block, the distributions of the grid points in the block interior has only to be matched smoothly to the grid-point distributions in the six faces.

   g. A grid in an edge defines only the grid in each face and block which has that edge on its boundary, but the grid in the other blocks and faces is completely independent of the grid in that edge. This property offers the posibility to adapt grids only locally, by modifying the geometric shape of that edge, and/or the grid-point distribution in that edge.

Similarly, modifications in the geometric shape of a face, and/or the grid-point distributions in a face or in a block, change a grid only locally. This 'locality' property is extremely useful when improving grids, by manual interaction for example.

11.A summary of the various mechanisms for the control of the acceptability and quality of grids is listed in the table below.

- Vertex-point positions,   ⎤

                               ⎪   manipulation of block-decomposition

    edge-curve shapes,   ⎬

                               ⎪   geometry (of given topology).

    face-surface shapes,   ⎦
- Numbers of grid points, per coordinate direction (constraining effect of grid-line continuity).
- Grid-point-distribution control from edges into faces into blocks, using user-defined positive dimensionless weight functions, in edges, and in faces and blocks only if required.
- Three-step procedure: edges → faces → blocks.
- Grid-control is local.

4. Experiences with the grid-generator procedure

From experiences with the grid-generation procedure with the aerodynamic configurations listed in table 2.1, it was found that the procedure satisfies most requirements of section 2.3 (User requirements for grid generators.) It is possible to construct multiblock grids around complicated 3D aerodynamic configurations, and Euler-flow calculations including propeller-slipstream simulations can be made with success.

A few examples of constructed grids are presented in figures 4.1a through d.

Strong points of the grid-generation procedure were found to be the following.

1. The face-to-face packing of blocks (without gaps or overlaps) and the continuity of the grid lines over block faces and over face edges leads to simple data structures in the grid-generator code and in the flow-simulator code.

2. The advantages of requiring only grid-line continuity over block faces and over face edges, and no more smoothness, seem to outweight the disadvantages. Advantages are the points 3-7 below.

3. Control of grid acceptability and grid quality (resolution), by manipulation of the given geometrical shape of faces and for edges, and of the position of vertices, has been used several times, and turned out to be useful.

4. Similarly, control of grid acceptability and quality by the weight functions in the elliptic grid-generation procedure produces usually acceptable grid-point distributions.
5. The default rules for the definition of the geometric shapes of block edges and of block faces (section 3, points 10.b, 10.c), which do not require user-input of geometric data, are in practice often applicable, and thus very useful.
6. In most faces and in most blocks, bi- and trilinear transfinite interpolation produces acceptable grids of good quality, when the grid points in edges have been correctly distributed by an elliptic method. Here, most faces and blocks means of the order of 80% of the faces and all blocks. This procedure is succesful when independent variables in the function prescriptions for edges, faces, and blocks are correctly tuned to each other. Grid tuning with the comparitively expensive elliptic methods can thus be avoided in the vast majority of blocks and faces.
7. The 'locality' property (section 3, point 10.g) of the control mechanisms for the constrcution of acceptable grids of sufficient quality turned out to be very useful, when grids had only locally to be improved.

   The grid generation procedure needs also improvement.
1. Definition, modification, and other manipulations of the topology and geometry data of a block-decomposition of a flow domain around a complex 3D aerodynamic configuration turned out to require much manpower, and to contribute significantly to long turnaround times for grid generation. The software for topology definition (PATRAN) and for subsequent geometry definition (SIGMA, AEROLIS, CATIA, etc.) can be used. However, PATRAN cannot handle aerodynamic surfaces with sufficient accuracy, and slight topology changes produce large changes in topology data. The other geometry software packages were found to have a high user threshold for CFD specialists. It was concluded that much shorter turnaround times for the construction of block decompositions will require special software for the integrated definition of topology and aerodynamic geometry data, to be handled by CFD specialists.
2. The requirement of face-to-face packing of blocks leads, in most applications with complicated aerodynamic configurations, to a large number of blocks with relatively small block volumes. For a complete aircraft, the number of blocks is of the order of 500 - 1000. The manipulation of this amount of blocks (and of a similar amount of vertices, edges, and faces) during block decomposition and grid generation is cumbersome. Further, because block volumes are relatively small, even on fine grids the total number of grid points per block becomes also relatively low (say, of the order of 20^3), so that vector operations (usually, only inner loops of nested loops are auto-vectorized) become slower than desired. Hence, it is desired to enlarge block volumes, to enhance vector performance in the flow simulator, and to reduce at the same time the total number of blocks, faces, edges, and vertices, to reduce the administrative tasks of a grid designer.
3. Further it was found that the mechanisms for the control over the quality of grids requires considerable improvement, to eliminate accuracy and approximation-efficiency problems with flow-simulation results. To explain how this arises, observe first that the requirements of face-to-face patching of blocks and of continuity of grid lines has as a consequence that, in complex 3D block decompositions, many grid lines are continuous through chains of blocks from e.g. flow boundary to flow boundary. Of course, grid-cell volumes are chosen small in blocks with high flow gradients (near wing leading and trailing edges, inlet lips, etc.), which leads to high grid-point densities in blocks covering such regions. But because grid lines are continuous over block faces, this high density is also propagated into other blocks where this may not be required, e.g. outer-flow regions near infinity. In order to prevent too dense grids in such outer blocks, designers make compromises by accepting, in blocks which should have dense grids, grids which are locally too coarse, so that here flow-simulation results are locally to inaccurate. Hence, there is a need for grid refinement/coarsening over block faces, like illustrated in figure 4.2, where grid lines can terminate on block faces, so that they do not propagate into blocks where a smaller grid-point density is sufficient for good grid quality.
4. When the grid-point densities in different blocks are strongly related to each other due to a continuity requirement of grid lines at block faces, grid designers spend (too) much manhour time in manually optimizing stretch-factor behaviour and other second-order behaviour of the grid-point distributions, to prevent large variations in grid-point densities over a block volume.
5. Sometimes grid folding is encountered. Efficient repair by an ad hoc procedure is usually possible, if the folded region can be quickly detected and visual inspection means are available for analysis.
6. The coupling between the grid generator code and the visualisation code can be made loose (two different computer codes in one job-control loop, coupled by files). This situation is excellent from the point of view of task decomposition, but a more tight coupling is necessary from the point of view of the man at the workstation/terminal.

5. Improvements

5.1 Introduction

   The block-decomposition/grid-generation procedures, and the corresponding procedures for the manipulation of surfaces on aerodynamic configurations and in the flow, are given the desired properties by introducing new data structures for multiblock grids, and new algorithms for block decomposition and grid generation. The new technical concepts are sketched in sections 5.2 - 5.6 below.

5.2 Elementary and compound blocks, faces, and edges

   When blocks are packed block-face to block-face, a collection of blocks, faces, and edges is obtained. These blocks, faces and edges may be called elementary, to distinquish them from another kind of blocks, faces, and edges to be introduced below. Such collections of elementary blocks can be described both topologically and geometrically by simple mathematical constructions, which can easily be mapped in computer codes. Exactly this simple structuring is also the reason why elementary-block subdivisions are insufficiently flexible for control of the accuracy of flow-solver results, and why vector lengths are relatively short. Is is thus necessary to generalize, to eliminate these problems.

A mathematically well-structured approach is obtained when unions of elementary blocks, faces, or edges are allowed to be combined into a new so-called compound block, face, or edge. Moreover, the concepts of an elementary block, face, and edge have to be generalized somewhat. The definitions of these entities are now as follows.

a. An elementary edge is a curve segment in space, with the topology of the unit interval, with two vertex points as its end points.

b. A compound edge is the union of two (elementary and/or compound) edges, joined together at a common interior vertex point.

c. An elementary face is a surface segment in space, with the topology of the unit square, with four (elementary and/or compound) edges as its boundary.

d. A compound face is the union of two (elementary and/or compound) faces, joined together at a common interior (elementary or compound) edge. This union should also have the topology of the unit square, and have four edges.

e. An elementary block is a volume segment in space, with the topology of the unit cube, and with six (elementary and/or compound) faces as its boundary.

f. A compound block is the union of two (elementary and/or compound) blocks, joined together at a common interior (elementary or compound) face. This union should also have the topology of the unit cube, and have six faces.

Conceptually, the generalisation is nothing else than that each union of elementary blocks, having also the topology of a unit cube, is accepted as a new compound block. Further, each union of elementary faces having the topology of a unit square, is also accepted as a new compound face, and each union of edges having the topology of the unit interval is also accepted as a new compound edge.

It will be evident from the definitions that each compound entity (block, face, edge) may be decomposed into a union of elementary corresponding entities via binary-tree data structures, with elementary entities at the leaves, and compound entities at the nodes.

Using compound entities, it is possible to greatly reduce the chaining effects in elementary-block decompositions, because a block face may now terminate in an interior edge of a compound face of an adjacent block, without propagating into that block due to a requirement of a block-face to block-face packing. This feature allows grid designers considerably better control over the geometrical modeling of blocks and the positioning of blocks with respect to each other.

5.3 Grid refinement per block and per computational-coordinate direction

Compound blocks are necessary but not sufficient to realise sufficient control of grid point densities. In addition it is necessary to allow that, in each block, the grid point density can be defined practically independently from that in adjacent blocks.

This requirement raises the question how the grids and the flow states in two different adjacent blocks should be related to each other at a common face. To analyse this question, two extreme cases are considered first.

a. When each grid line is continuous from one block over the common face into the other block, the data structures for grid geometry and flow states in each of the two blocks are closely related to each other. They can in fact be related to each other in computational space using topology data, because the grid-point multi-indices in the two blocks can be mapped onto each other by a linear relation. However, the advantage of the simplicity of this mapping is at the same time a constraining factor, because, near the common face, the grid-point densities in the two blocks are about equal due to this linear mapping.

b. When the grids in the two blocks at each side of the common face would be completely unrelated, grid-point multi-indices of the two grids must be related to each other in physical space via the geometrical position of grid points in the common face. This involves complex search procedures, and produces a, in general very nonlinear, relation between the grid-point indices of the two blocks. The advantage of complete freedom in defining grid-point density in each block is thus obtained at the cost of this nonlinear multi-index relation.

A reasonable compromise between these two unattractive alternatives is to keep the property that grid-point multi-indices of the two grids are mapped onto each other by a linear relation, but to allow grid-point densities over faces to increase or decrease, by applying grid coarsening/refinement over block faces. See figure 4.2 for an illustration. The coarsening/refinement is allowed independently in each of the three computational directions. Of coarse this presumes that in the solver special block-coupling algorithms are available, that are conservative, and in general second-order accurate.

5.4 Numerical aerodynamic geometry modeling

As discussed in section 4, it became recently clear that numerical aerodynamic geometry modeling in the block-decomposition/grid-generator codes is required because, for CFD specialists, the user threshold of current geometry software packages is very high, and the required functionality for Euler and NaStokes flow calculations is not available (e.g. double curved surfaces in the flow).

Face surfaces are defined by control-point distributions, arranged in 2D arrays, which are interpolated to a unit-normal continuous face surface, by patching together transfinite bicubic Hermite polynomials, one for each patch spanned by four control points.

It is important to observe that a unit-normal continuous surface may be represented by an interpolating function with weaker continuity than C^1-continuity. This fact may be exploited to optimize the qualitative behaviour of the curvature of the surface represented by the Hermite polynomials. This optimalization is aerodynamically useful.

This approach, whereby a face surface is represented by patched bicubic transfinite Hermite polynomials interpolating a 2D array of face-surface control points, may be compared to approaches, in which an underlying analytic surface definition from a geometry software package is used, which must then not be compromised during multiblock grid generation. We consider our approach as more efficient and flexible because, in the flow solver, aerodynamic configuration surfaces are approximated anyhow by grid point distributions. It has then no sense to stick to a unique aerodynamic-surface definition in the block-decomposition/grid-generation process when this is cumbersome, provided the CFD specialist is offered full information about and control over the geometrical approximations he is making during the manipulation of aerodynamic surfaces.

Another important topic concerning aerodynamic geometry modeling is that of natural (i.e. physically relevant) independent variables (parameters) in the 3D vector functions defining the geometrical shape of an edge curve or of a face surface.

It was found that, during grid generation in a face, usually good grids could be obtained, using only transfinite bilinear interpolation, if the grid-point distribution in each of the four edge curves of the faces was appropriately defined. In each of the four edge curves, the corresponding distribution of the independent variables in the 3D vector functions for the face geometry are then also known. A grid may then be defined in two steps. First, in the face interior, a uniform distribution of the two independent variables between their edge values is defined using bilinear transfinite interpolation in the independent variables on the four edges. Next, each pair of independent-variables values of this distribution is substituted in the 3D vector function for the face geometry, to produce a grid point in space on the face.

This procedure fails when the independent variables in for example two opposite edges are differently defined, for example, arc length in one edge, and control-point index in the other edge with a very non-uniform control-point distribution. In such a case, the above procedure may produce unexpected distributions of grid points in the face.

In order to prevent such unexpected behaviour, it is necessary to use geometrically meaningful independent variables. They are called here natural variables (or parameters). Arc lengths along parameter lines are expected to be a good choice in general, but other choices may occasionally be also useful.

The use of natural parameters and multi-linear transfinite grid generation for the final generation of complete multiblock grids is now under development.

5.5 Control of grid quality and acceptability

It may be expected that CFD specialists will have considerably better options for controlling the quality and acceptability of a multiblock grid, when the following technical means are available.
- Both, compound and elementary faces and edges available for use. (Compound blocks are not needed.) If desired, enclose in special blocks regions requiring extra dense grids, or allowing coarser grids. Use of unit-normal continuous surfaces defined by patches of bicubic transfinite Hermite polynomials, to position complicated block faces in the flow.
- Grid-point density defined per block, under the constraint that computational multi-indices of grid points in each common face of two blocks can be mapped onto each other by a linear transformation. Grid refinement/coarsening over block faces allowed (grid-point density control here).
- Grid tuning has the 'locality' property discussed above. Grid tuning, using in subsequently each edge, each face, and each block, the chain of mappings:
  : computational coordinate(s) in edge, face, or block →
  → weight function(s) for grid tuning (grid smoothness control here):
  : natural independent variable(s) in edge, face or block →
  → 3D vector function defining the geometrical shape of the edge, face, or block:
  : grid-line coordinate(s) in edge, face, or block.
  Bi- and trilinear transfinite interpolation is used to define weight functions in faces and in blocks from those in the edges and faces, respectively.

5.6 Hyperblocks

A hyperblock is a collection of blocks, arranged to form a desired topological structure, and having a rough geometrical shape that may be defined in detail as desired. Examples are the Cartesian, Cylindrical, and Spherical hyperblock structures of figure 5.1.

Hyperblocks may become useful to speed up the interactive block-decomposition process, by inserting automatically around a given aerodynamic configuration part a large number of blocks, faces, edges, and vertices in one hyperblock with one hyperblock command. It remains to be investigated whether this feature is really needed in practice, because the use of compound faces and edges may in practice eliminate the need for hyperblocks, in particular when graphical workstations are fast enough.

6. Concluding discussion

The scientific technical issues to be reported here are what may be learned from past experiences, and to conclude from them how to proceed in near future with the grid-generation discipline.

Multiblock grids can be used with success for numerical flow simulations around complex aerodynamic configurations (section 4). This may be achieved by:
. integrating development work in grid generation with corresponding work in flow-solver development (section 2.1, 2.3-2.5),
. developing new mathematical theory for the description and construction of the topology, the geometry, and the grid-point distributions for numerical multiblock-flow simulation, and by
. developing new corresponding block-coupling algorithms in the flow solver [7,8].

In section 5, five technical concepts for the improvement of the multiblock grid generation procedure are proposed. They may in particular be used to improve the accuracy of the flow simulation results and their stable computation (sections 4, 2.1), and to reduce manhour investments in the construction of multiblock grids. These five concepts are:
1. compound blocks, faces, and edges (section 5.2),
2. grid refinement per block (section 5.3),
3. new aerodynamic-geometry modeling techniques (section 5.4),
4. control of grid quality and acceptability with various new concepts (section 5.5), and
5. hyperblocks (section 5.6).

It may be expected that the incorporation of these concepts in the multiblock flow simulator codes will greatly improve the system in the desired directions. Because the concepts are introduced now, the firm proof of this expectation cannot yet be presented here.

7. References

1. Boerstoel, J.W.   Preliminary design and analysis of procedures for the numerical generation of 3D block-structured grids, NLR TR 86102U (1986).

2. Boerstoel, J.W.   Progress report of the development of a system for the numerical simulation of Euler flows, with results of preliminary 3D propeller-slipstream/exhaust-jet calculations, NLR TR 88008L (1988).

3. Boerstoel, J.W.   Numerical grid generation in 3D Euler-flow simulation, in: Numerical Methods for Fluid Dynamics, Ed. Morton, K.W., Baines, M.J., Clarendon Press, Oxford (1988).

4. Bartlett, G.W.   An experimental investigation of propfan installations on unswept supercritical wing at transonic Mach numbers, NASA CR 172605 (1985).

5. Amendola, A.   User guide for the generation of three-dimensional block-structured grids, NLR TR 88176L (1988).

6. Amendola, A.   A multiblock elliptic grid generator for general three-dimensional configurations; Methods and software, NLR TR 88175L (1988).

7. Amendola, A.,   Validation of a multiblock Euler flow solver with propeller slipstream flows,
   Tognaccini, R..   in AGARD Conf. Proc. 437, Validation of Computational Fluid Dynamics, Lisbon
   Boerstoel, J.W.,   (May 1988).
   Kassies, A.

8. Kassies, A.,   to be published.
   Tognaccini, R.

9. Jacobs, J.M.J.W.,   Numerical interactive grid generation for 3D-flow calculations, in: Numerical
   Kassies, A.,   Grid Generation in Computational Fluid Mechanics '88, Sengupta, S.,
   Boerstoel, J.W.,   Häuser, J., Eiseman, P.R., Thompson, J.F. (ed.s), Peneridge Press (1988);
   Buysen, F.,   NLR MP 88055U (1988).
   Kuyvenhoven, J.L.
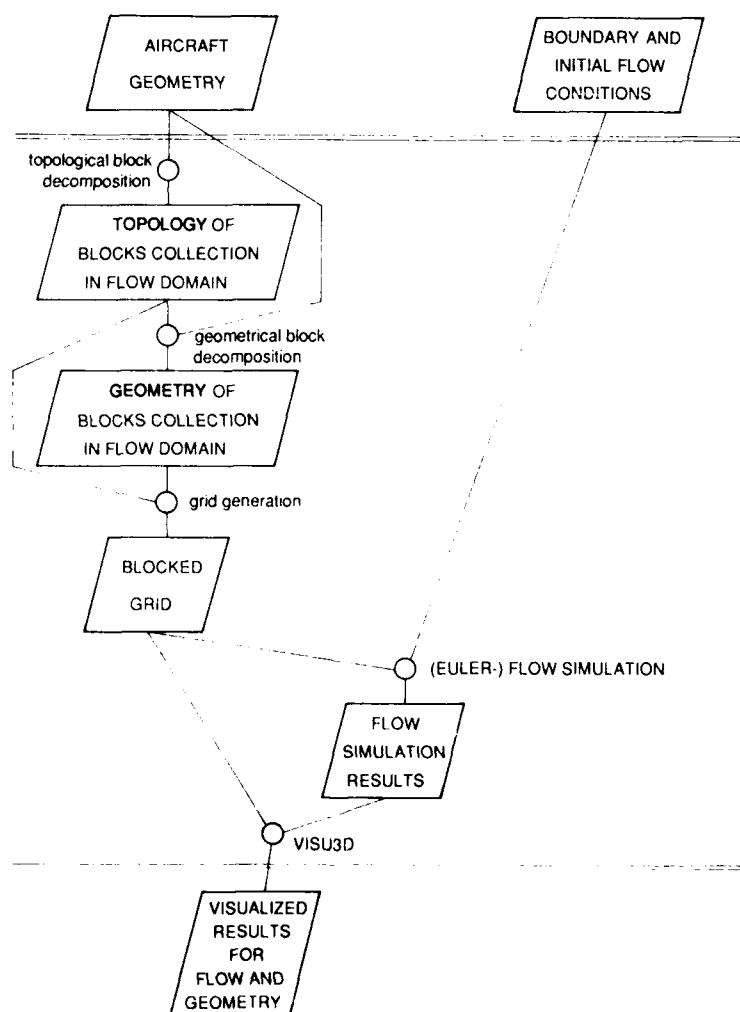
Fig. 2.1 Block decomposition and grid generation as subtasks in a numerical flow simulation.
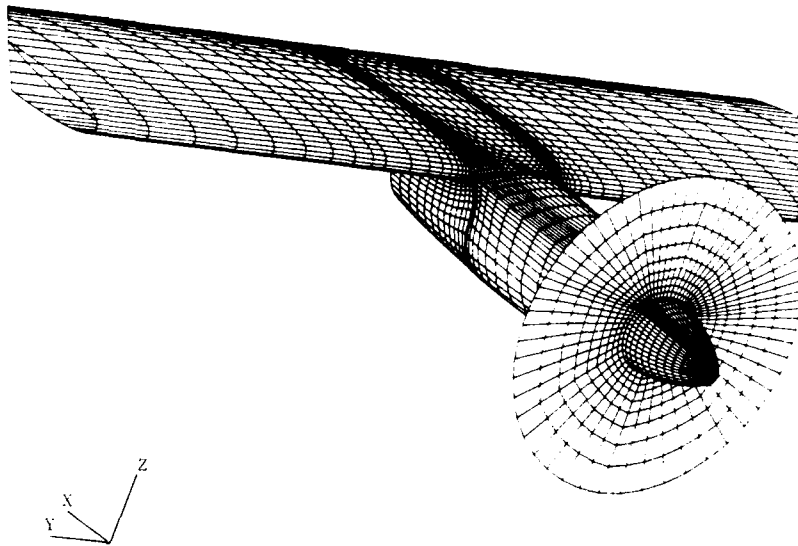
Fig. 2.2 NASA-Langley wing-nacelle-propeller configuration.

10 Blocks

20 Blocks
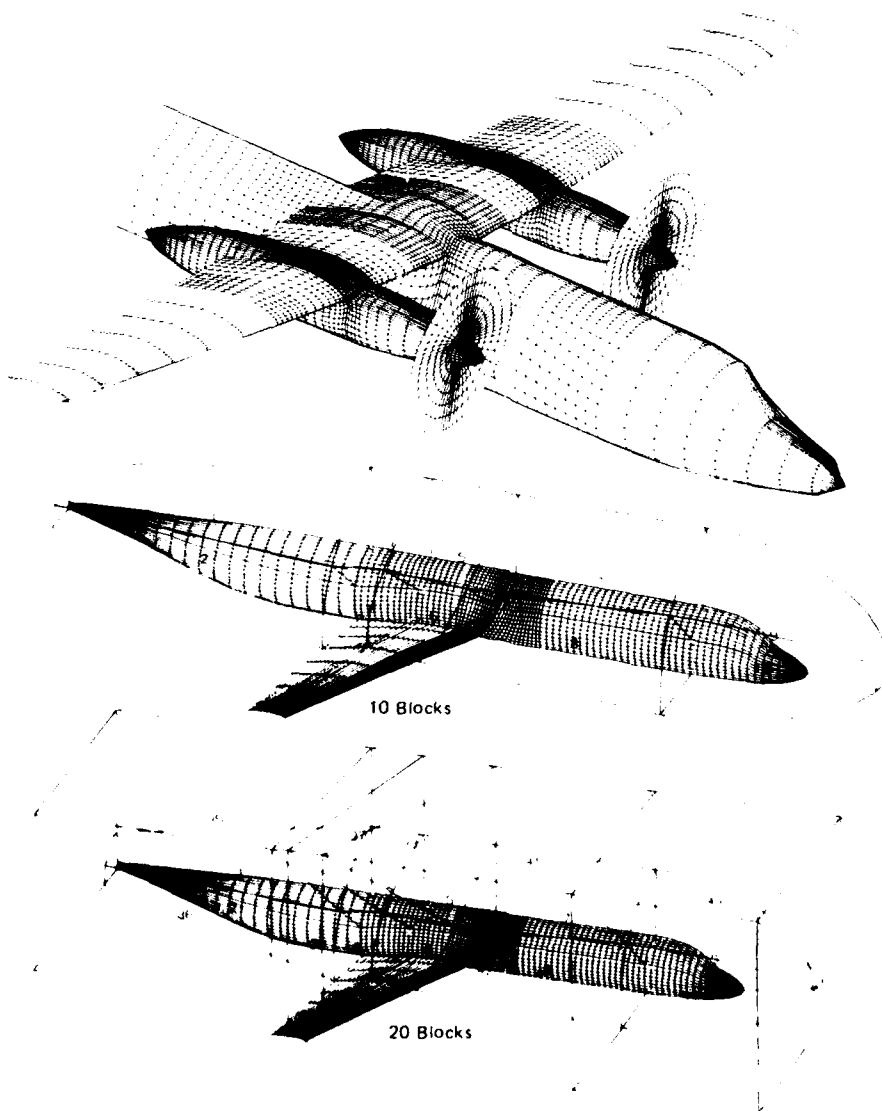
Fig. 2.3 Fokker 50 and Fokker 100 configurations.
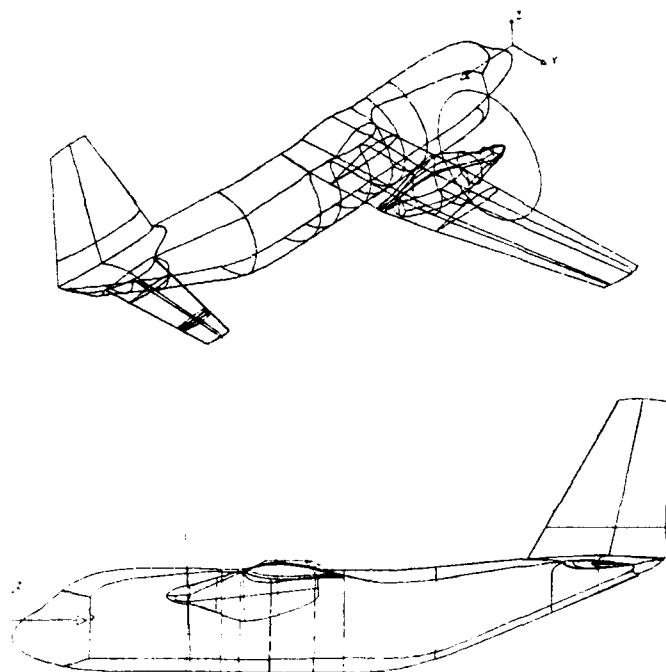
Fig. 2.4   Aeritalia G222 configuration.

propeller disk

wing
l.e.
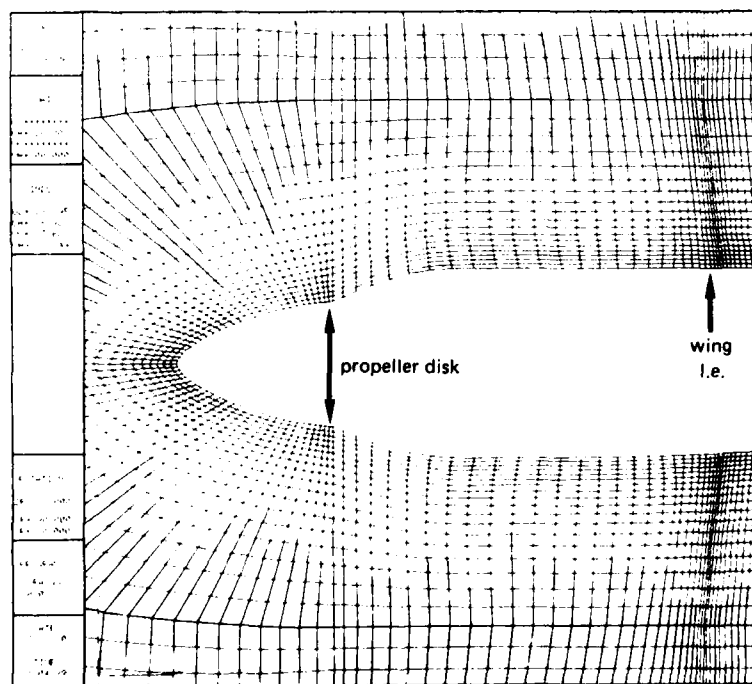
Fig. 3.1   Example of slope-discontinuity of grid lines at block faces, edges, and vertices.
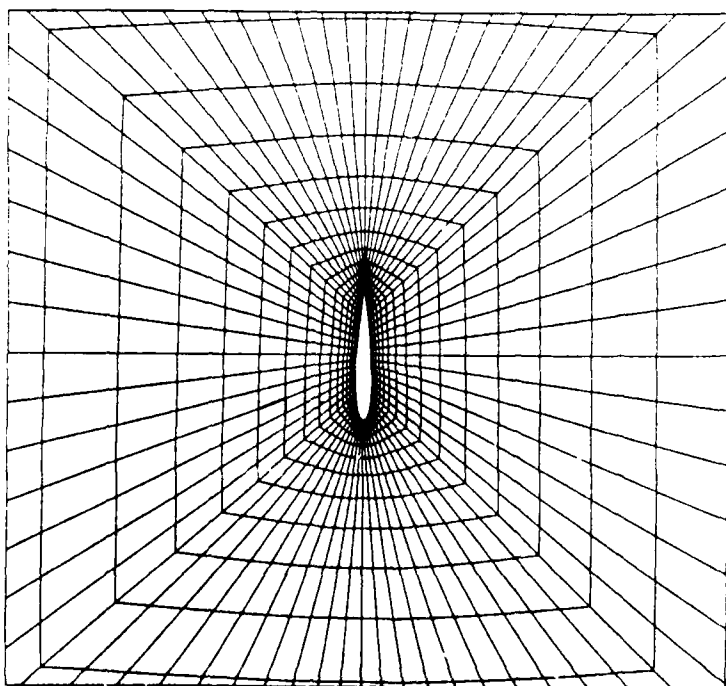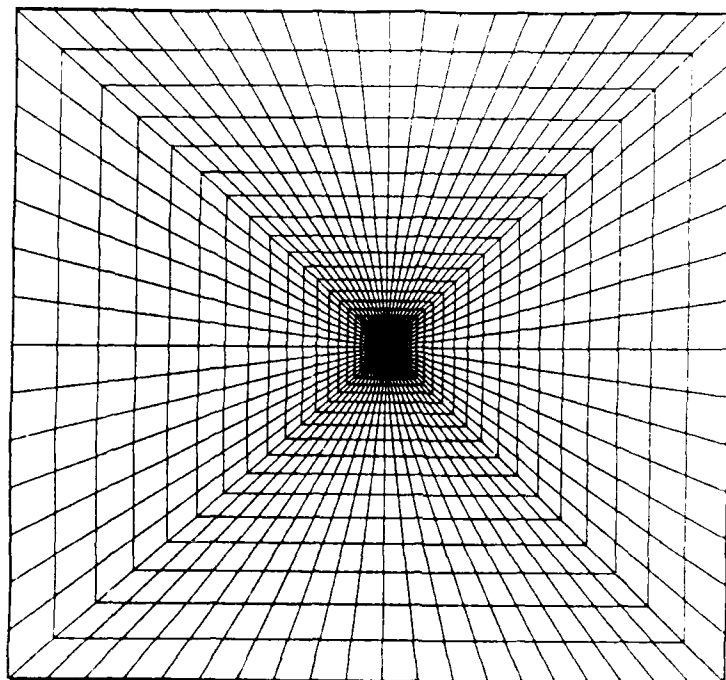
Fig. 3.2 Example of well-defined grid at corners of flow domains.
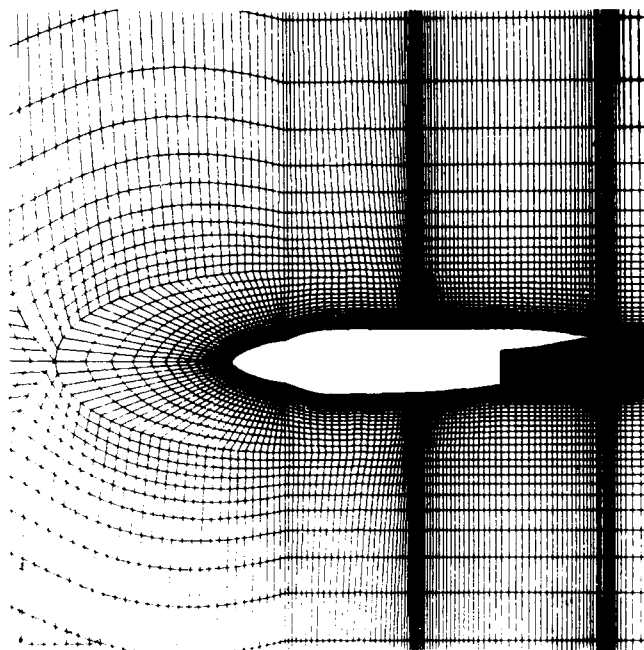
Fig. 4.1a Example of collection of grid planes.
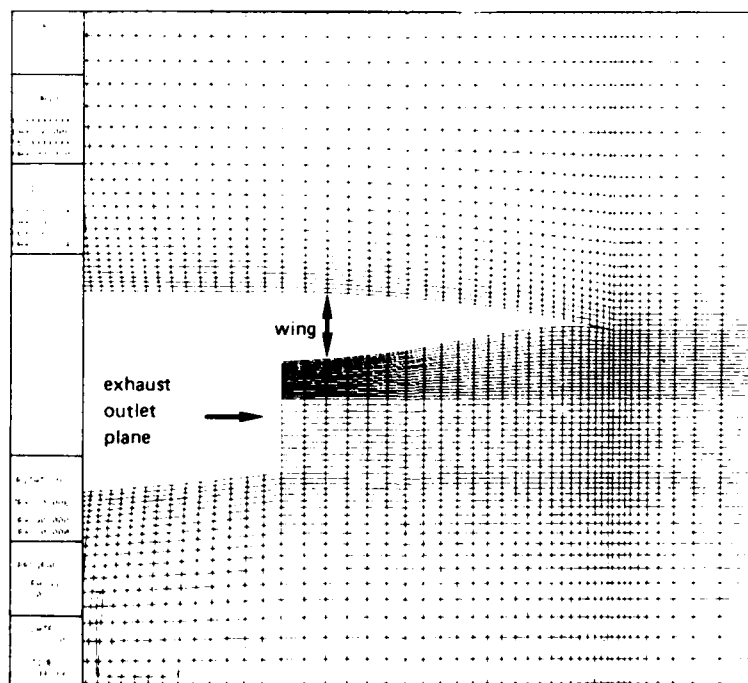
Fig. 4.1b Example of collection of grid planes.
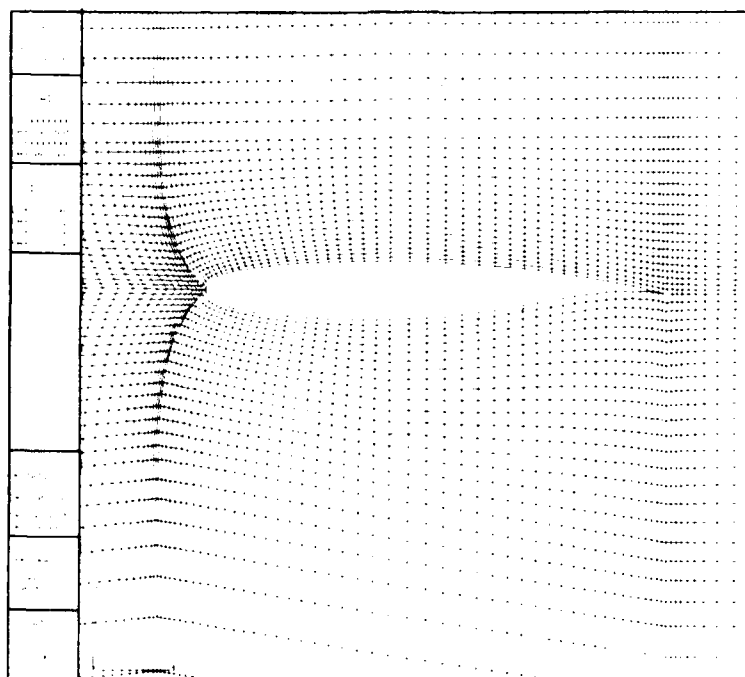
Fig. 4.1c Example of collection of grid planes.

enlargement
of

BLOCK FACE
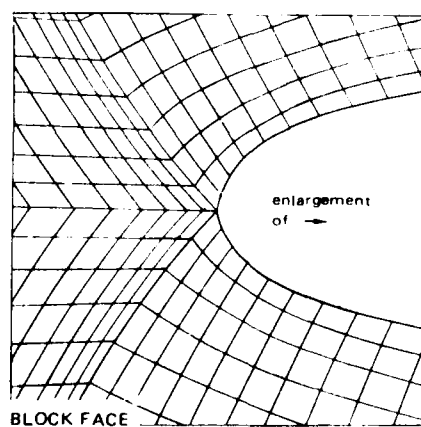
Fig. 4.1d Example of collection of grid planes.
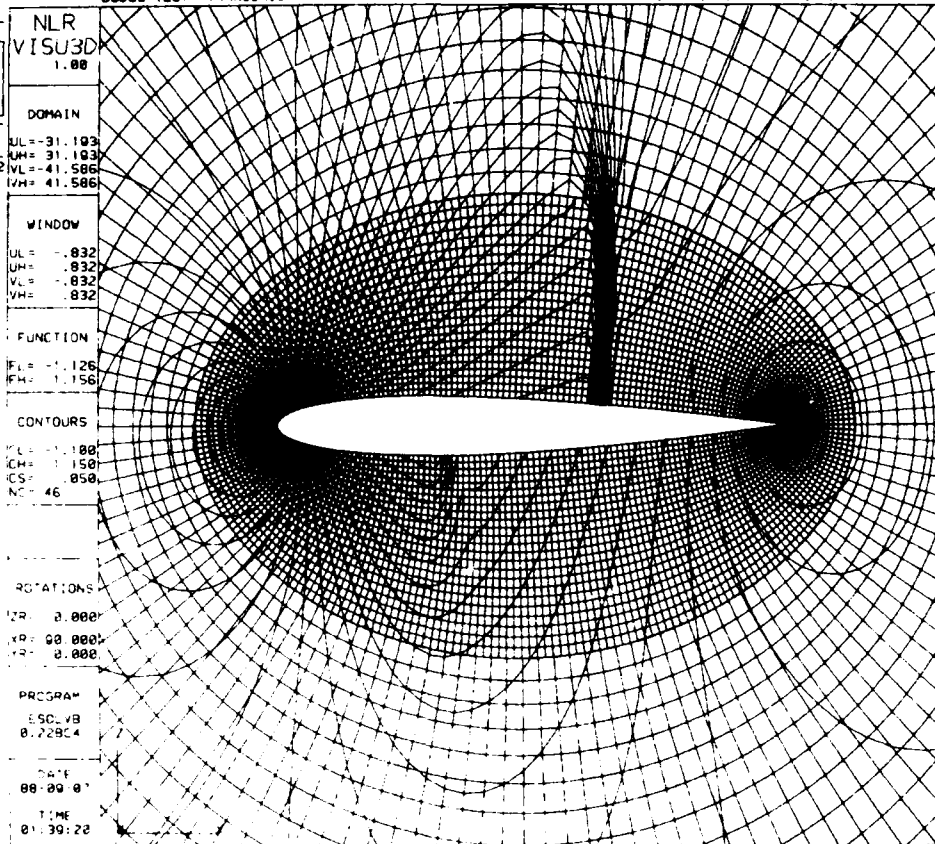
Fig. 4.2 Grid refinement/coarsening over block faces, for grid-point density control.

Fig. 9.2 continued

CARTESIAN HYPERBLOCK

CYLINDRICAL HYPERBLOCK

SPHERICAL HYPERBLOCK
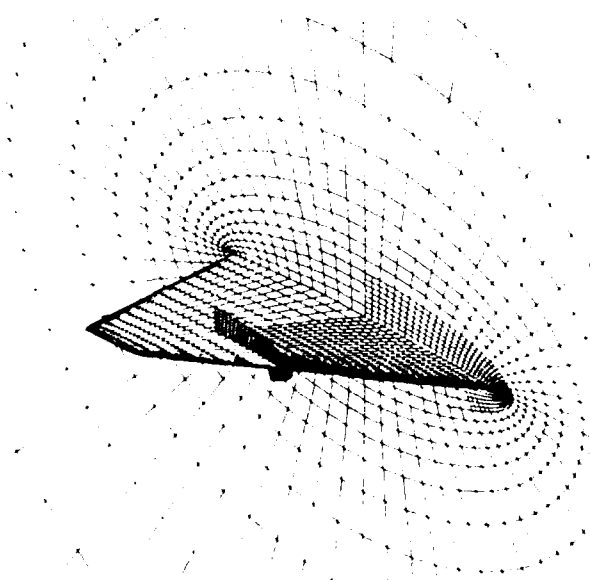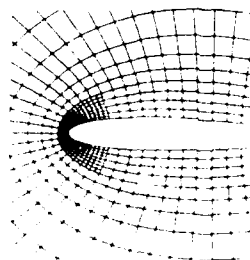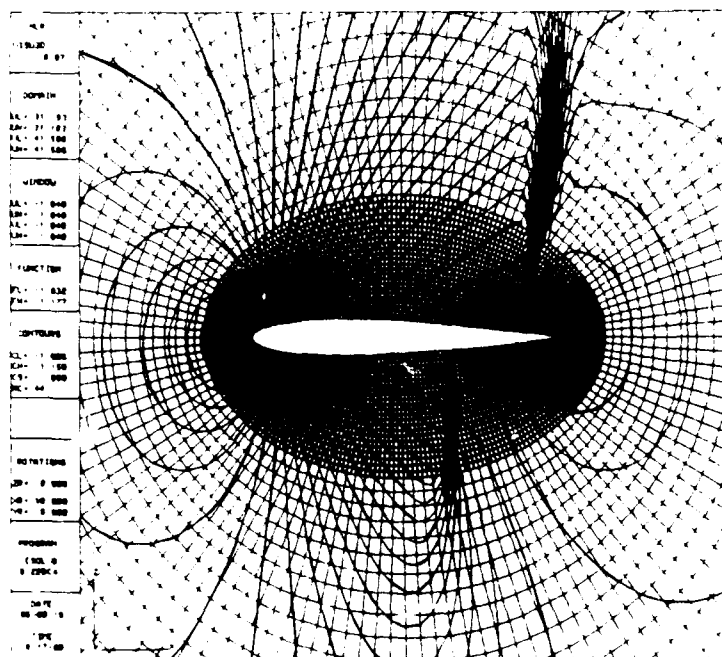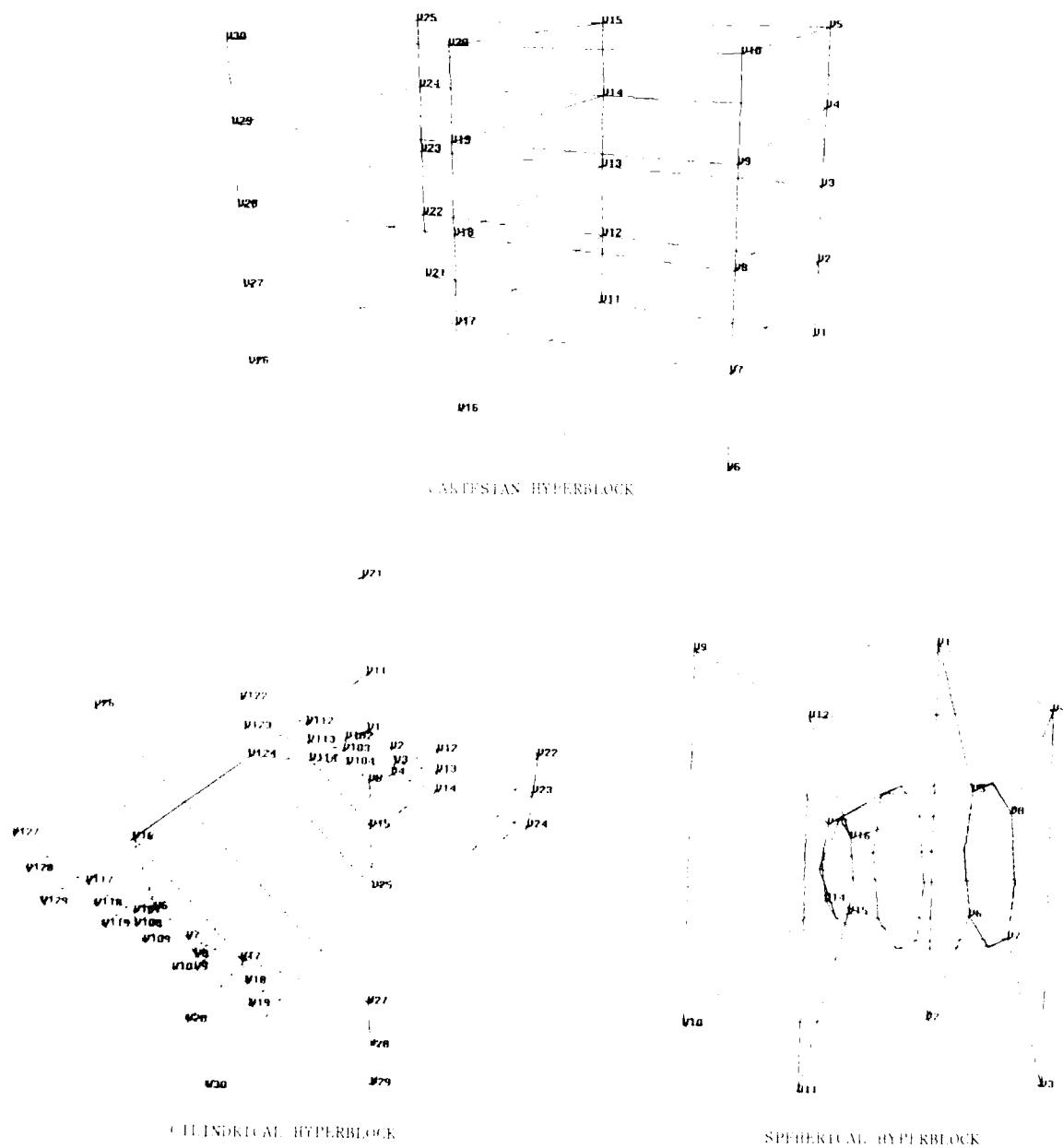
Fig. 5.1 Hyperblocks.

GRID PATCHING APPROACHES FOR COMPLEX THREE-DIMENSIONAL CONFIGURATIONS

W.Schwarz*, G.Hartmann**, M.A.Schmatz*,
K.M.Wanie*, M.Pfitzner**
Messerschmitt-Bölkow-Blohm GmbH
Postfach 80 11 60
D-8000 München 80
Federal Republic of Germany

SUMMARY

Three examples of different grid generation procedures are reported in this paper. The first one is based on a single-block approach but nevertheless it is able to handle very complex aircraft configuratons and requires only a minimum of user input. This system was the base for the development of the following patched grid method. The next example shows the application of the patched grid technique for the zonal solution of Euler, boundary--layer and Navier-Stokes equations and demonstrates the ability of this method to achieve the necessary local grid refinement for viscous calculations. Finally an application of a patched grid method for an Euler code with a shock fitting approach for supersonic calculations is presented. Here the grid near the body surface is fixed whilst the grid in the outer region is moved so that it can be adapted to the location of the bow shock.

1. INTRODUCTION

Grids which are used for Euler calculations are usually continuous over the whole computational domain because this is the most simple way for the implementation of the flow solver. For the solution of the Navier-Stokes equations however, it may be impossible to generate a suitable continuous grid, at least for more complex configurations. In this case the use of patched grids is a very attractive way to solve the problem. But this approach is not limited to Navier-Stokes solutions because the use of patched grids offers the possibility to introduce local grid refinement and/or local grid adaption easily.

For the patched grid approach the compucational domain is divided into different regions. In each region the grid may be adapted to the special requirements of the flow solver. These grids are finally patched together to cover the whole flowfield. This may be done in two different ways. In the blocked approach the grids fit together so that there is no overlapping. Another method is to overlay the different grids so that the grid lines are really overlapping. This grid is much simpler to generate but problems in the flow solver may arise because of the necessary interpolation work and because a conservative discretization of the governing equations is very difficult. Applications of both methods are included in this report.

The paper starts with a short discussion of the advantages of CAD systems for surface grid generation. This is followed by an overview of the basic grid generation algorithms that are used. The major part is the description of various grid generation methods. This includes grids for complex configurations suitable for Euler solutions, patched grids for zonal solutions (a coupling of Euler/boundary-layer/Navier-Stokes equations) and adaptive meshes for a shock fitting Euler code.

2. SURFACE GRID GENERATION

The surface geometry of new aerodynamic configurations is usually developed nowadays using CAD/CAM systems (Computer Aided Design/Computer Aided Manufacturing), in which the geometry is represented by mathematical functions. The CAM part of them allows direct programing of NC-machines (Numerically Controlled) and thus represents the connecting link between construction and production. Since the constructed geometry is represented best in the CAD system it is the natural way to use CAD also for the generation of surface grids required for numerical simulations and to link construction and numerical simulation in this way. In contrast to the usual proceeding, where the geometry is transfered to the grid generation algorithm by means of point coordinates which are to be interpolated, a deformation of the geometry is completely avoided. Due to the interactive working technique and the plotting devices inherent in CAD systems, grid generation can be done ve.y efficiently. Only few additional features compared to standard CAD systems are to be provided, which are known mainly from conventional grid generation, like input/output routines, routines to compute point distributions along lines etc.

In the following the surface grid generation for a hypersonic forebody configuration using the CAD system CATIA will be described exemplary. The lines i=const. (i-lines) are situated in cross section planes. Starting point for the surface grid generation is the CAD surface model (Fig. 2.1a). The first step is to define the cross section planes for the i-lines. For this purpose a suited point distribution along, say, the x-axis is

---

* Military Aircraft Division, Theoretical Aerodynamics
** Space Communications and Propulsion Systems Division, Aerodynamics

created and planes normal to the x-axis are constructed at these points. These planes are intersected with the surface geometry to create the i-lines of the surface grid (Fig. 2.1b). Since the surface usually is represented in the CAD system by various overlapping elements, the i-curves are obtained as overlapping elements, too, which have to be trimmed and concatenated to get a single, unique curve. The generation is finished with the creation of suited point distributions along the i-lines (Fig. 2.1c).

The described process, of course, is just an example. The inherent flexibility of CAD systems allows a generation very closely adapted to the various problems. Furthermore the application of CAD is not limited to surface grid generation. Also two-dimensional space grids can be build up very easy. The probably more interesting further application, however, is the construction of outer boundary conditions for the two- and three-dimensional elliptic grid generation procedures described in the following chapters.

Once the surface grid is generated its quality should be assessed by analysing the metric properties [1], which are requiried to be sufficiently smooth. Figure 2.1d shows as an example the distribution of the metric tensor component a_{11}, a further analysis can be found in [2].

3. BASIC GRID GENERATION SYSTEMS

The grid generation systems used for the cases described in this paper are all based on elliptic systems, namely on some formulation of Poisson's equation. The reasons for this are the well known advantages of elliptic partial differential systems (discussed for example in [3]), in particular the extremum principles to guarantee a non-overlapping grid, the smoothness of the resulting grid and the possibility to specify the points on the entire boundary.

3.1 ORIGINAL POISSON SYSTEM

The most commonly used form of a general Poisson-type grid generation system is

$$\xi_{xx} + \xi_{yy} + \xi_{zz} = P,$$

$$\eta_{xx} + \eta_{yy} + \eta_{zz} = Q, \qquad (3.1)$$

$$\zeta_{xx} + \zeta_{yy} + \zeta_{zz} = R.$$

Grid control is exercised via the so called "control functions" P, Q, R. For the examples shown in this paper, these functions where used to attract grid lines towards other specified lines or points like described in [4]. The range and intensity of this attraction may be chosen. The attraction in ξ-direction is controlled by the P function which takes the following form in 2D:

attraction towards a line $\xi = \xi_i = const.$:

$$P = A_i \ sign(\xi_i - \xi) \ exp\left(- \frac{|\xi_i - \xi|}{B_i}\right), \qquad (3.2)$$

attraction towards a point (ξ_i, η_i):

$$P = A_i \ sign(\xi_i - \xi) \ exp\left(- \frac{\sqrt{(\xi_i - \xi)^2 + (\eta_i - \eta)^2}}{B_i}\right), \qquad (3.3)$$

where the subscript i denotes a particular line $\xi = const.$, A_i is the intensity of the attraction and B_i is a decay factor which limits the range of the attraction effect.

The Q function works in a similar form for the attraction in η-direction with ξ and η interchanged. Eq. (3.1) has to be transformed in the computational domain and the resulting quasi-linear equation is solved by a Gauss-Seidel iteration scheme. An application of this type of grid generation system is shown in Chapter 6 for the HERMES reentry vehicle.

3.2 BIHARMONIC SYSTEM

This grid generation method is also based on Poisson's equation, but this time the formulation in computational space as explained in [5] is used:

$$x_{\xi\xi} + x_{\eta\eta} + x_{\zeta\zeta} = P(\xi, \eta, \zeta),$$

$$y_{\xi\xi} + y_{\eta\eta} + y_{\zeta\zeta} = Q(\xi, \eta, \zeta), \qquad (3.4)$$

$$z_{\xi\xi} + z_{\eta\eta} + z_{\zeta\zeta} = R(\xi, \eta, \zeta).$$

Therefore no transformation between physical and computational space is necessary. The values of the control functions P, Q, R are determined by the solution of Laplace's equation to assure a smooth distribution of the source terms over the whole computational

domain:

$$P_{\xi\xi} + P_{\eta\eta} + P_{\zeta\zeta} = 0,$$

$$Q_{\xi\xi} + Q_{\eta\eta} + Q_{\zeta\zeta} = 0, \qquad\qquad (3.5)$$

$$R_{\xi\xi} + R_{\eta\eta} + R_{\zeta\zeta} = 0.$$

Together, these two sets of equations form the biharmonic equation $\nabla^4 x^i = 0$. Due to the stability problems of central difference approximations of higher-order derivatives, this fourth-order equation is implemented as a system of the two second-order equations (3.4) and (3.5). The discretization of the derivatives by centered differences leads to linear algebraic equations in physical space which can be rearranged easily to yield the final equations for the grid point coordinates. For the numerical solution of these equations a simple Gauss-Seidel iteration scheme is used.

The boundary conditions for the coordinates may be of Neumann or Dirichlet type. The boundary conditions for the control functions P, Q, R are calculated using the current coordinates of the points near the boundaries according to Eq. (3.4). As the location of these points changes during the iteration procedure, the values of the control functions at the boundaries do not remain constant but are updated continuously.

There are different ways to calculate the boundary values of the control functions, depending on the type of boundary. To obtain a concentration of grid lines towards inner boundaries (i.e. surface of the configuration), one or several grid lines away from the surface must be generated by some algebraic grid generation technique. At the last grid surface determined in this manner the boundary values of the control functions are determined using the coordinates of the surrounding points as indicated in Figure 3.1a. At the outer boundaries where freestream flow conditions are imposed, the values of the control functions are put to zero forcing a Laplace type grid in this region (Fig. 3.1b). In a multi-block approach, the boundary between two blocks is treated with a similar technique as the inner boundaries. The control functions at the block boundary are determined from the coordinates of the surrounding points in both blocks as show in Figure 3.1c.

The next two chapters will show some applications of this grid generation system, both for single-block grids and for patched grids using a multi-block approach.

4. SINGLE-BLOCK GRID GENERATION FOR COMPLEX GEOMETRIES

4.1 GENERAL DESCRIPTION

The application of the biharmonic grid generation system will be demonstrated first for single block grids, as this is the basic grid generation approach. To resolve complex geometries properly, a very flexible H-type topology with several interior branch cuts is used. This leads to a number of singular lines and points on the surface called fictitious corners. So in this concept not the flowfield but the configuration has to be divided into blocks as it is shown in Figure 4.2b for the surface grid of a fighter aircraft. The whole computational domain is a single block and the volumes lying inside the configuration have to be excluded from the calculation.

The grid generator accepts any number and arrangement of configuration blocks. As input the program needs the surfaces of these blocks. So the first step is the division of the whole surface into sub-surfaces which are limited by fictitious corner lines. Then a suitable point distribution on these surfaces is generated as described in Chapter 2. The points on these surfaces must already have the correct __global__ coordinates (ξ,η,ζ) and special care has to be taken at the interfaces between two surfaces to assure continuity of grid lines. Starting with this input the grid generator recognizes the dummy volumes inside the configuration and marks them with a logical flag.

As explained in Chapter 3.2, the first coordinate surface off the boundary has to be calculated by some algebraic method to achieve an attraction of grid lines towards that boundary. In this case this attraction is done in quasi-radial direction by specifying a constant aspect ratio (a/b) like shown in Figure 4.1. For ξ=constant boundary surfaces this attraction is done along ξ-coordinate lines and in an analogous form for the other coordinate directions. In principle it is possible to specifiy different aspect ratios for different surfaces or to specify the distance of the first coordinate surface from the body instead of the aspect ratio.

In the plane of symmetry, the grid may be generated with a 2D version of the grid generation algorithm or by using symmetry boundary conditions for Eqs. (3.4) and (3.5). In the normal case the farfield boundaries form a rectangular box and a perpendicular intersection of grid lines is imposed. It is also possible to rotate any of its faces so that a truncated pyramid is formed (Fig. 4.5b).

The advantage of this single-block grid generation method is that once you have a suitable surface grid with properly defined block boundaries, it is very simple to generate the grid for the flowfield because only few parameters are necessary, namely the number of grid lines and the attraction parameter(s). The disadvantage is that you have only

limited grid control possibilities. Therefore the resulting grids are in general useful only for Euler calculations. The Euler method and some applications are described in [6,7].

4.2 APPLICATIONS

The grid for an advanced fighter aircraft demonstrates that it is possible to generate grids for very complex geometries with this single-block approach. The main features of this configuration (see Fig. 4.2a) are a fuselage with belly intake, a cranked delta wing, canard and two fins. Figure 4.2b shows the surface grid in computational space and illustrates the complicated block structure that is necessary to resolve this configuration properly. For the generation of the volume grid (Fig. 4.3) only one attraction parameter was used for the whole mesh. Details of the grid generation procedure are described in [8]. The results of the Euler calculations done on this mesh are reported in [9].

Another example is the grid for a wing tunnel model which was used for wave drag investigations. The model has a fuselage with a delta wing and plyons with external stores may be added. Figure 4.4 is taken from [10] and shows the surface grid and Figure 4.5 shows some views of the volume grid. As the calculations had to be done only for supersonic test cases, the outer boundaries were adjusted to the presumed location of the bow shock (Fig. 4.5b) in order to minimize the total number of grid points.

5. PATCHED GRIDS FOR ZONAL NAVIER-STOKES APPLICATIONS

5.1 ZONAL SOLUTION METHOD

Together with the development of large vector computers it is possible now to compute the viscous flow about realistic configurations. At MBB the Navier-Stokes solver NSFLEX was developed and applied successfully for a wide range of Mach numbers [11,12]. However, systematic application of Navier-Stokes methods in aerodynamic design is limited up to now by high computer costs. To accelerate the Navier-Stokes method the so-called CCPNS (close coupling procedure for the Navier-Stokes equations) method was designed, see for example [13,14].

The principle is to cover the flowfield with a uniform Navier-Stokes grid. In regions of weak viscous-inviscid interaction the fine grid in the vicinity of the wall is discarted to get an Euler grid there, which is a subset of the governing Navier-Stokes grid. In these weak interaction regions an equivalent inviscid flow is calculated, i.e. a combination of an Euler and a boundary-layer solution. The boundary-layer calculation in these regions delivers the equivalent inviscid source distribution for the inviscid solution as

well as the flow profiles for the coupling of the equivalent inviscid region and the regions of strong viscous interaction where the Navier-Stokes equations are solved. Strong viscous interaction occurs where shocks or separation are located. There the boundary-layer equations are no longer valid. In Figure 5.1 the different zones can be seen for the wing flow application presented here.

To study the effect of using patched grids, the method was coded and extensively tested for Euler flows [13]. There it could be shown that the zonal Euler solution and the global Euler one give the same results even for very different cell sizes at the artificial boundaries if third order accurate fluxes are calculated at these boundaries, too, as in the complete flow field [13].

The present zonal method was applied to several two and three-dimensional flow problems. In comparison with a full Navier-Stokes solution, the CCPNS method requires about half the computer time and yields similar results. Note that the code is highly vectorized. The MBB boundary-layer code SOBOL ([1,15]), which solves the second-order boundary-layer equations is incorporated in the method as a subroutine. The CCPNS code is designed to find the different zones of flow modelling automatically and to rezone them if this is indicated by the boundary-layer method. To get a good vector performance the different zones are chained to one-dimensional arrays plane by plane.

Due to the flow and the grid topology the flowfield is divided in four different zones (Fig. 5.2). In blocks 1,3 and 4 the Navier-Stokes equations are employed and also a Navier-Stokes grid is used. In block 2 the Euler together with the boundary-layer equations are solved. An equivalent inviscid flow is calculated there on a much coarser grid. At the artifical boundaries coupling approaches are necessary at every time step. The boundary-layer solution is recalculated after some time steps.

The surface grid together with the outer grid shell of the Navier-Stokes mesh is shown in Figure 5.3.

5.2 GRID GENERATION FOR A WING FLOW SIMUALTION

From a given surface distribution a C-O-type grid is generated using local monoclinic coordinates at the wall whereever possible, since the boundary-layer theory is restricted to such coordinates. Note that the boundary-layer method works on the same surface grid as the Euler-Navier-Stokes method does and that for a second-order boundary-layer solution an inviscid flow distribution is required. With this surface normal grid an algebraic turbulence model, like the one of Baldwin and Lomax, is easily and accurately to apply. Exponential stretching is used in these algebraic subblocks, that means where viscous effects are predominant. The height of the first cell is designed such that the

dimensionless wall coordinate y⁺ will reach about 2 to 4 in the flow solution to resolve
the viscous sublayer of the overall turbulent flow with at least one cell. After a NSFLEX
solution y⁺ is checked for accuracy reasons.

The inviscid flow grid is found as a subset of the viscous one by simply omitting some
cells. In Figure 5.4 a detail of a η=const. plane shows the Euler grid in comparison with
the Navier-Stokes grid, which differs just near the body. Since the grid is perpendicular
to the wall and since the Euler mesh is a subset of the Navier-Stokes mesh only little
interpolation work is necessary for the CCPNS process.

The elliptic grid generation procedure described in Chap. 3.2 follows algebraic mesh
generation to cover easily the outer region with mesh points. Utilized is the biharmonic
system, Eqs. (3.4),(3.5). Dirichlet boundary conditions are employed at all boundaries of
the computational domain. This way the grid points are attracted towards the algebraic
mesh. To vectorize the grid generation method red-black Gauss-Seidel iteration is used
like in the Navier-Stokes method.

The surface grid for a generic transport aircraft wing is shown in Figure 5.3. To get an
exponentially stretched point distribution in the farfield the whole grid is redistri-
buted along ξ,η=const lines, the sublines starting from the surface, using the curves of the
algebraic and the elliptic subblocks as interpolation paths. The main features of the
C-O-type mesh used can be seen in Figure 5.5, where the symmetry plane, the outer
boundary and the plane where the upper grid joins the lower one is plotted.

The computational domain is devided up into four different blocks (Fig. 5.2). Block 2 is
the equivalent inviscid one, the Navier-Stokes equations are solved only on blocks 1,3
and 4. Since the zonal boundaries are not known a priori it is necessary to allow them to
float during the convergence process. This rezoning capacity is achieved by chaining
planewise the solution vector, the geometry and what else is needed in the different
subroutines in one-dimensional arrays. Then the answer is calculated again on one-di-
mensional array and restored in the original three-dimensional arrays. Thereby only one-
dimensional arrays have to be added to the code in comparison to the mono-block NSFLEX
code. For details see [16].

To demonstrate that both the patched grids and the zonal solution procedure give reason
able results here one application out of [16] is reported. The freestream condition is:
M∞=0.78 and α=2.2 degrees, Re=7,000,000. The governing Navier-Stokes grid consists of
156,000 grid points which is a rather crude grid. In chordwise direction 100 cells are
used, in spanwise direction 40 cells and normal to the wing 39 cells. For the inviscid
part of the grid 10 cells from the Navier-Stokes grid are omitted, see Figure 5.4. This
means that the Euler grid consists of 116,000 cells in total. The resolution at the wall
is rather fine for inviscid calculations. The height of the first Euler cell is designed
to be 0.02 percent of the mean chord length. The boundary layer is calculated in the
region of the equivalent inviscid flow (block 2 in Fig. 5.2) with 50 points normal to the
wall. A much higher resolution is achieved in comparison with a Navier-Stokes solution
especially in regions where the boundary layer is thin since the boundary-layer mesh is
adapted to the boundary-layer thickness.

In Figure 5.6 the isobars on the lower and on the upper side of the wing demonstrate
perfect smoothness across the zonal boundaries. The same behaviour can be found in the
skin friction and the pressure distribution at all spanwise stations [16]. Compared to a
global Navier-Stokes solution the results are nearly the same whilst the computer time is
reduced and the accuracy in the zone of the boundary layer is enforced.

With the zonal method described above all tools are available and verified also for the
use of embedded meshes both for Euler and/or for Navier-Stokes applications. In the
future this will be done to resolve special features of hypersonic flow fields around
complex configurations in more detail.

6. PATCHED GRID FOR THE HERMES REENTRY VEHICLE

A computational grid has to be generated for the HERMES reentry body, which is suitable
for the calculation of the flow field using a shock-fitting-EULER-code (finite differen-
ces), that means good continuity for the metric derivatives and simple and fast adapta-
tion to the location of the bow shock is needed, because the grid has to be adapted after
every timestep [17,18]. An essential point for this is a fixed surface grid on the body.
For a good resolution of the body in a flow calculation it is useful to attract points at
regions of large curvature, and coarsen the grid spacing in areas of small curvature,
that is to cluster the grid points. As measure for the clustering the local radius of
curvature is used (Fig. 6.1), which is smoothed (Fig. 6.2,6.3), because clustering should
occur in the whole neighbourhood of maxima of curvature. Also a constant level is added,
so that the rest of the curve is not too much depleted from grid points (Fig. 6.4). The
coordinate points given in spanwise cross-sections of the body contour are catched with
parametric splines for interpolation. The grid points on the cross-sectional curves are
then chosen such that they divide the area under the clustering function into equal
increments (Fig. 6.5). This procedure results in a special point distribution for the
given ribsections. Now the coordinate points of corresponding intervals are connected by
splines, resulting in grid lines along the body. For the final surface grid (Fig. 6.6)
these grid lines are intersected with the cones of the special coordinate system for the
shock-fitting-algorithm, where the distribution of aperture angles ω are given.

For the generation of the space grid the body is divided into two areas with an overlapping part, to make interpolation of flow variables possible for the calculation. A one-block grid with straight lines to the fitted bow shock and body adjusted angles ω and ♦ (Fig. 6.7) is used for the front part. The distribution of points on the lines is achieved by an exponential function, to enable condensed grid lines at the body (Fig. 6.8).

Such a simple structure of the net is not possible for the rear part (winglet section). The region between body and fitted bow shock is divided into two areas (Fig. 6.9). With the angle δ=70° of the tangent a footline on the wingtip is defined (Fig. 6.10), also a straight line in the x-z-plane of symmetry with the angle ε=45° is chosen (Fig. 6.11). The boundary curve between the two blocks in the ribsections are chosen as half ellipses, which intersects the two fixed lines. The point distribution on the boundary curves (Fig. 6.12) are then done with the weighting function based on the local radius of curvature, like described above.

Since block 2 has a rather complicated boundary and is fixed in time, in this block the grid is generated numerically by an elliptic solver [3,4,19,20]. This 2-D grid generation code [21] distributes grid points by solving Poisson's equation (Chapter 3.1) using Gauss-Seidel overrelaxation. The source terms in the Poisson system admit clustering of grid points along specified lines or points. The strength and range of source terms can also be varied by the user (Eq. 3.2 and 3.3). The grid points on all boundary lines of the domain have to be specified and are (except on symmetry boundaries) not moved by the solver. An attraction line has been placed on the upper wing surface cross-section curve and attraction points are set at the wing root and at the junction point between the wing and the winglet. The strength of the source terms are adjusted such that the space grid appears compatible with the given boundary point distribution in the first grid plane. The source terms are then held constant in the grid generation process of all other grid planes to get a smooth grid in the z-direction. Fig. 6.12 shows the computational grids in two cross-sections. It is nearly body orthogonal and smooth. The resolution of this type of grid is better than the one block grid especially near the wing-body junction.

Due to the time-dependent bow shock boundary, which moves during the relaxation in its stationary location, the outer grid block is time-dependent, in contrast to the inner block, which is fixed. An algebraic grid generation is the most efficient strategy for a time-dependent grid [22,23,24]. The simple geometrical form of block 1 allows to choose straight lines from the inner boundary to the bow shock, in the same manner applied to the forebody. The points along these straight grid lines are distributed with the same exponential stretching function used for the front region. To achieve grid lines as smooth as possible across the block boundary, the angle ♦ of the grid lines has to be matched to the angle distribution of the fixed grid in block 2. First a raw angle distribution is calculated. At the block boundary the angle is taken over as linear extrapolation of the grid lines in block 2. At the body contour of the lower wing surface the angle ♦ is calculated from the body normal projected onto the grid plane (Fig. 6.14). This distribution is smoothed by an IMSL spline, so that monotonous increasing is achieved (Fig. 6.15). At last Fig. 6.16 shows a whole two-block grid plane in a cross-section.

The flow field computation runs in two parts, first the forebody with the one-block structure is calculated. Then the flow variables in the overlapping section are interpolated from the one-block grid into the two-block grid and taken as boundary condition for the winglet region (Fig. 6.17). After convergence is reached the two datasets are connected for analysis. Fig. 6.18 shows the lines of constant Mach number on the surface of the body. In spite of the patched grid the solution shows good continuity at the transition between the different grids, where a better resolution in the section with two-block structure is achieved.

7. CONCLUSIONS

Although the described single-block grid generation system is acceptable for a wide range of Euler applications, its strong smoothing tendencies and limited grid control possibilities make it unsuitable for the generation of Navier-Stokes type grids. This disadvantage could be overcome by the introduction of an algebraic sub-grid which provides a resolution which is sufficently fine for calculations with viscous methods.

An important fact is, that the flow solvers have no problems in handling the boundaries between two different grids. The computed results show no jumps in the flow quantities across the boundaries. Although the patched grid methods shown here have been applied only to specific configurations, the experiences are encouraging. The use of patched grids is an interesting and relatively simple way to introduce local mesh refinement and adaption. This is especially true for viscous flow calculations.

Future work will include the improvement of the single-block system so that it is possible to have more influence on the grid. This includes a posterior grid optimization and also some interactive grid generation strategies. An extension of the basic method to multi-block grids is also planned. The patched grid technique will be applied to other configurations to develop a more general approach, in addition the grid for the HERMES configuration will be adapted to the requirements of Navier-Stokes calculations.

REFERENCES

[1]   Monnoyer, F.: "Calculation of three-dimensional attached viscous flow on general configurations with second-order boundary-layer theory", to appear in ZFW, 1989.

[2]   Wanie, K.M., Schmatz, M.A.: "Numerical analysis of viscous hypersonic flow past a generic forebody configuration", in preparation.

[3]   Thompson, J.F., Warsi, Z.U.A., Mastin, C.W.: "Numerical Grid Generation", North-Holland, 1985.

[4]   Thompson, J.F., Warsi, Z.U.A., Mastin, C.W.: "Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review", Journal of Computational Physics, Vol. 47, 1982, pp. 1-108.

[5]   Schwarz, W.: "Elliptic Grid Generation System for Three-Dimensional Configurations Using Poisson's Equation", Proc. 1st Intern. Conf. on Numerical Grid Generation in Computational Fluid Dynamics, J.Häuser, C.Taylor (eds.), Pineridge Press, Swansea, 1986, pp. 3.1 - 352.

[6]   Eberle, A.: "3D Euler Calculations Using Characteristic Flux Extrapolation", AIAA-Paper 85-0119, 1985.

[7]   Eberle, A., Schmatz, M.A., Schäfer, O.: "High-order solutions of the Euler equations by characteristic flux averaging", ICAS-paper 86-1.3.1, 1986.

[8]   Eberle, A., Schwarz, W.: "Grid Generation for an Advanced Fighter Aircraft", in Three Dimensional Grid Generation for Complex Configurations - Recent Progress, AGARD-AG-309, 1988, pp. 65 - 76.

[9]   Eberle, A., Misegades, K.: "Euler Solution for a Complete Fighter Aircraft at Sub- and Supersonic Speed", in Applications of Computational Fluid Dynamics in Aeronautics, AGARD-CP-412, 1986, pp. 17-1 - 17-12.

[10] Heiß, S.: "FFE-Aufgabe CAE/CAD Integration, Arbeitspaket Oberflächennetzerzeugung", MBB-report MBB-FE122-AERO-MT-826, 1988.

[11] Schmatz, M.A., Brenneis, A., Eberle, A.: "Verification of an implicit relaxation method for steady and unsteady viscous and inviscid flow problems", AGARD Symp. Validation of CFD, Lisbon, 1988.

[12] Schmatz, M.A.: "Hypersonic three-dimensional Navier-Stokes calculations for equilibrium gas", AIAA-paper 89-2184, 1989.

[13] Wanie, K.M., Schmatz, M.A., Monnoyer, F.: "A close coupling procedure for zonal solutions of the Navier-Stokes, Euler and boundary-layer equations", ZFW, Vol.11, 1987, pp. 347-359.

[14] Schmatz, M.A., Monnoyer, F., Wanie, K.M., Hirschel, E.H.: "Zonal solutions of three-dimensional viscous flow problems", in: Zierep, J., Oertel, H. (eds): IUTAM Symposium Transsonicum III, Springer-Verlag, 1989, pp. 65-74.

[15] Monnoyer, F.: "Second-order three-dimensional boundary layers", Notes on Numerical Fluid Mechanics, Vol.13, Vieweg, Braunschweig-Wiesbaden, 1986, pp.263-270.

[16] Schmatz, M.A., Monnoyer, F., Wanie, K.M.: "Numerical simulation of transonic wing flows using a zonal Euler / boundary-layer / Navier-Stokes approach", ICAS paper 88.4.6.3, 1988.

[17] Hirschel, E.H. Weiland, C.: "Hermes Numerical Aerothermodynamics for Phase B2", MBB-Report H-RE-1-0002-MBB, 1988.

[18] Hirschel, E.H., Weiland, C.: "Hermes Numerical Aerothermodynamics for Phase B3", MBB-Report H-RE-1-0016-MBB, 1988.

[19] Thompson, J.F.: "Grid Generation Techniques in Computational Fluid Dynamics", AIAA Journal Vol. 22, No. 11, pp. 1505-1523.

[20] Thompson, J.F.: "Elliptic Grid Generation", in: Numerical Grid Generation, ed. J.F. Thompson, North-Holland, 1982.

[21] Halter, E.: "Programmdokumentation FCBFC Benutzerhandbuch", unpublished report, KFK Karlsruhe, 1985.

[22] Smith, R.E.: "Algebraic Grid Generation", in: Numerical Grid Generation, ed. J.F. Thompson, North-Holland, 1982.

[23] Ives, D.C.: "Conformal Grid Generation", in: Numerical Grid Generation, ed. J.F. Thompson, North-Holland, 1982.

[24] Gordon, W.J., Thiel, L.C.: "Transfinite Mappings and Their Application to Grid Generation", in: Numerical Grid Generation, ed. J.F. Thompson, North-Holland, 1982.
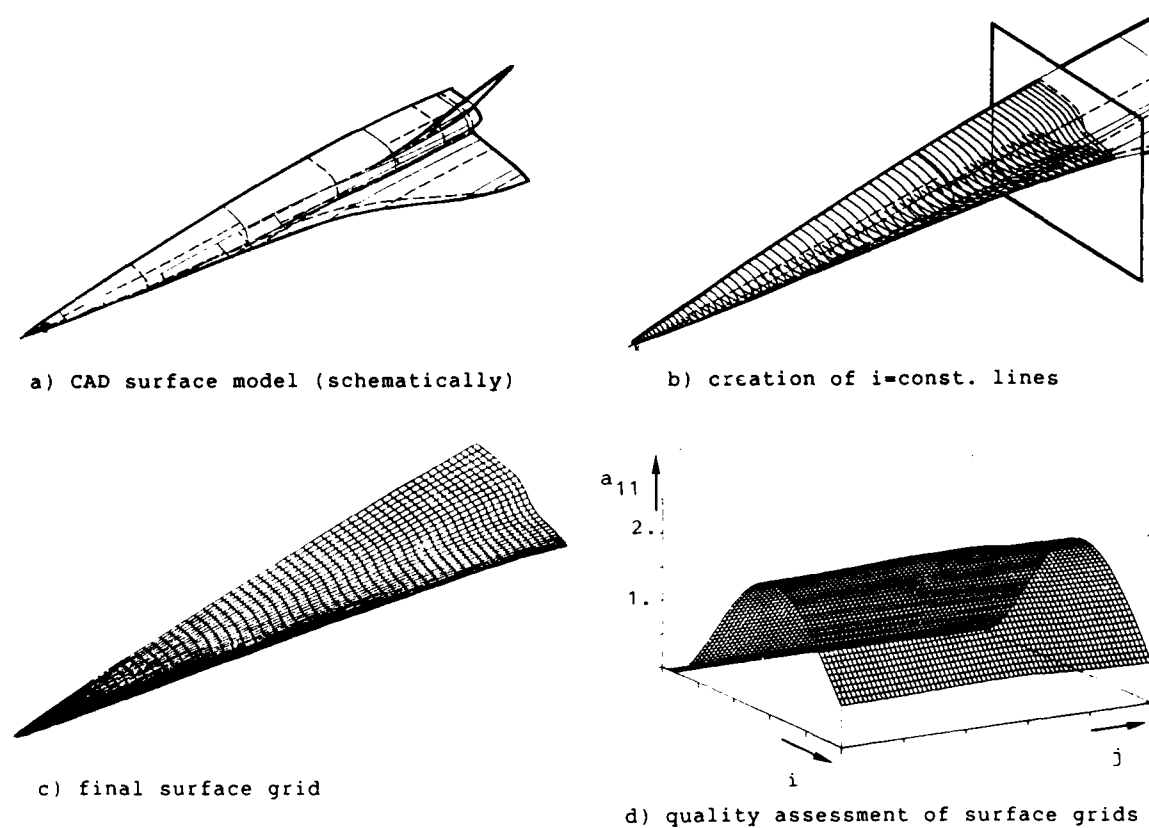
a) CAD surface model (schematically)

b) creation of i=const. lines

c) final surface grid

d) quality assessment of surface grids

Fig. 2.1 Surface grid generation for a hypersonic forebody configuration

a) inner boundary   b) outer boundary   c) block boundary

——— boundary

● points where the control function is calculated

○ points necessary to calculate control function
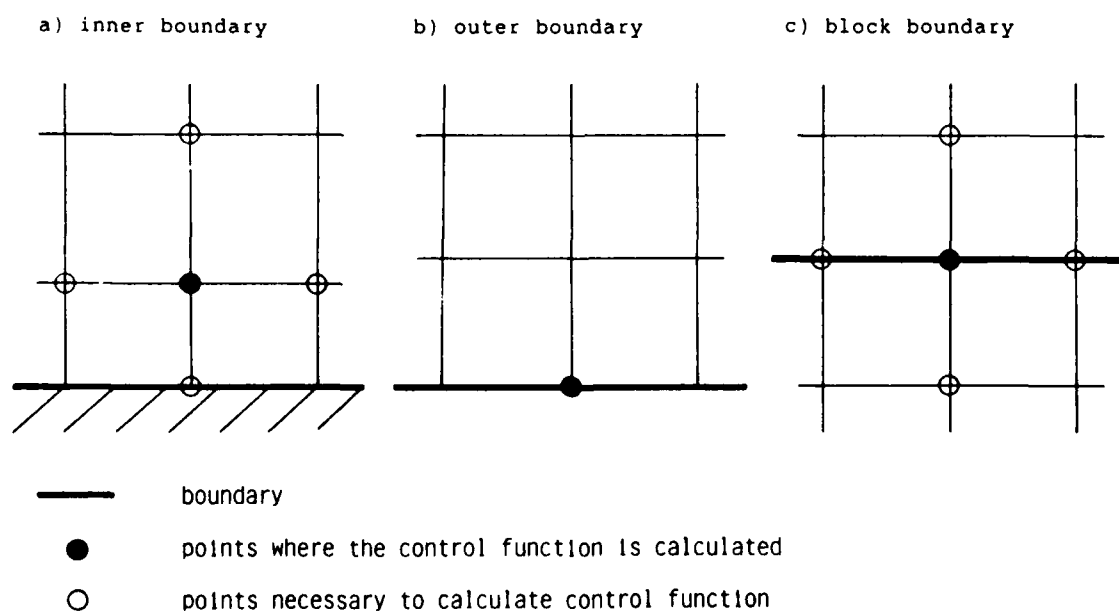
Fig. 3.1 Boundary conditions for control functions
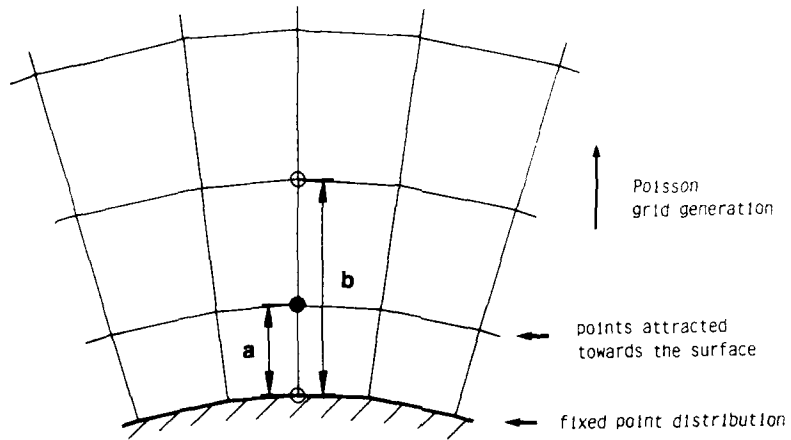
Fig. 4.1 Attraction of grid points towards the body surface

a) in physical space   b) in computational space
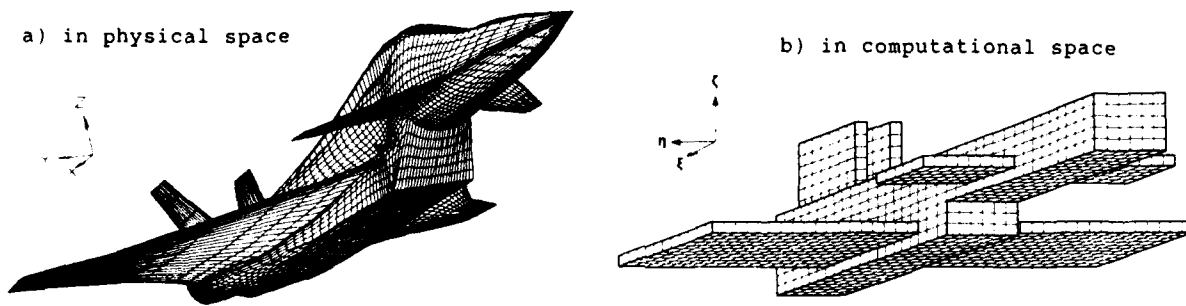
Fig. 4.2 Advanced fighter aircraft, surface grid
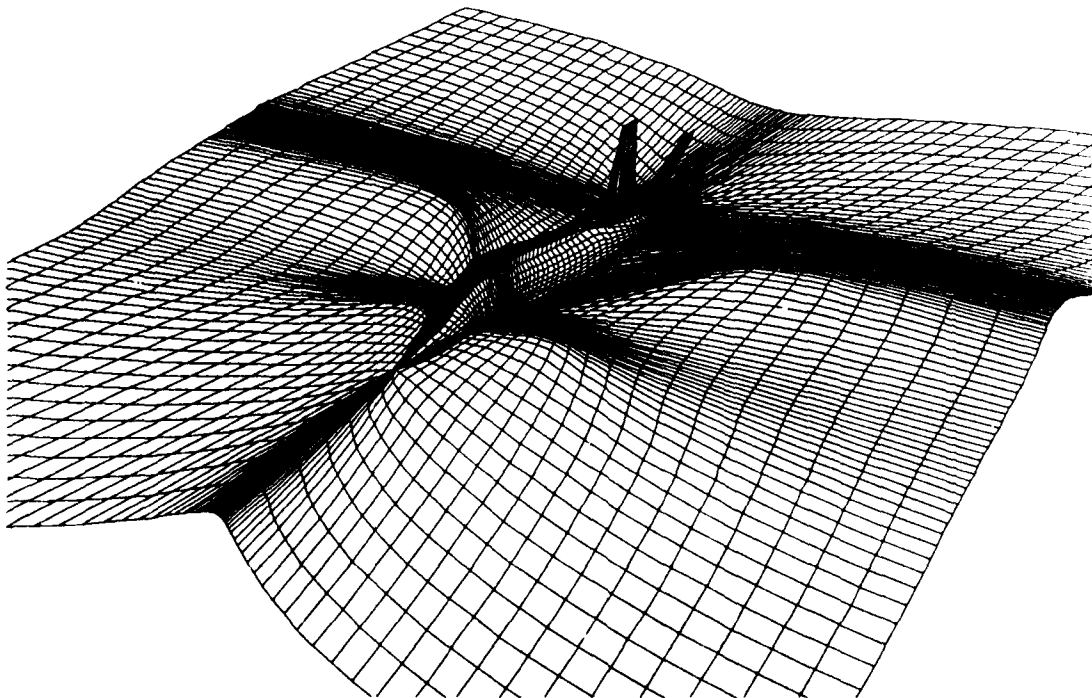
Fig. 4.3 Advanced fighter aircraft, upper surface grid
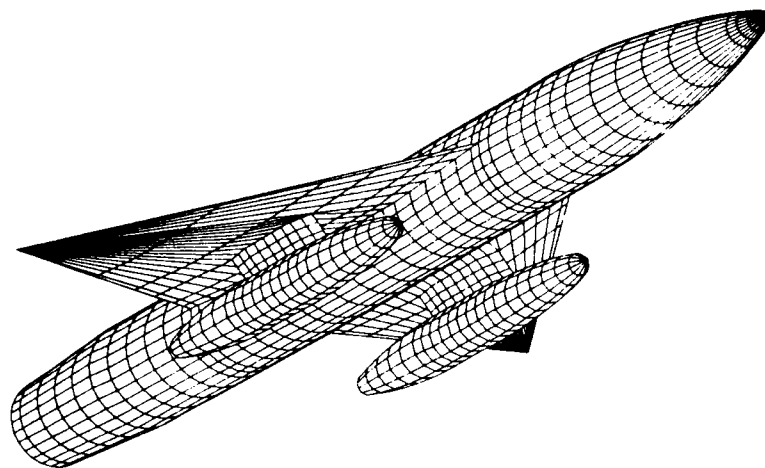
Fig. 4.4 Wing-fuselage-pylon-store - combination, surface grid

a) cross-sectional surface

b) grid surface at wing root
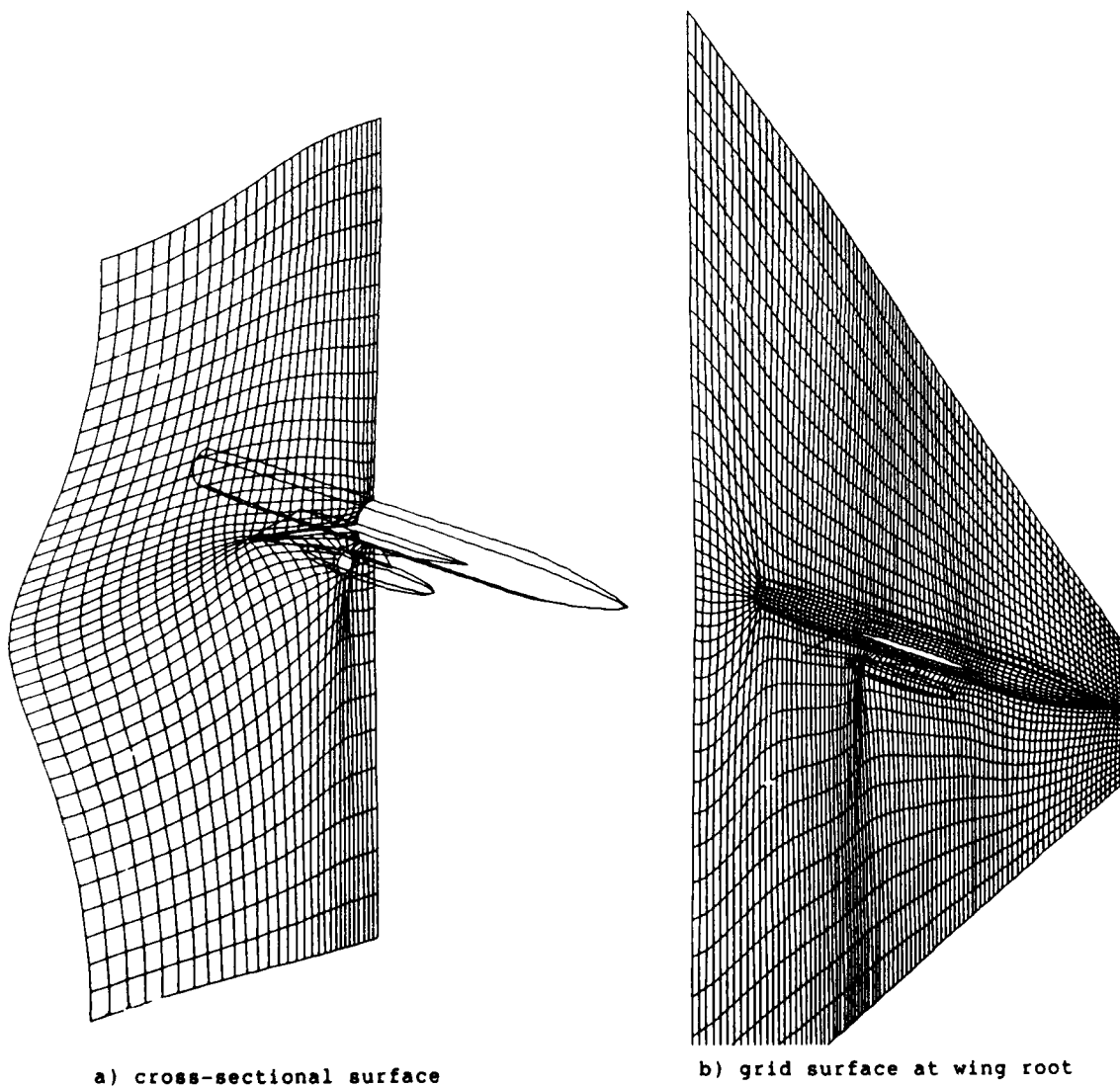
Fig. 4.5 Wing-fuselage-pylon-store - combination, details of volume grid

Fig. 5.1 Zonal Navier-Stokes solution for wing flow problems

Fig. 5.2 Solution adaptive block definition

Fig. 5.4 Patched Euler and Navier-Stokes grid

Fig. 5.3 Wing surface grid and Navier-Stokes zone
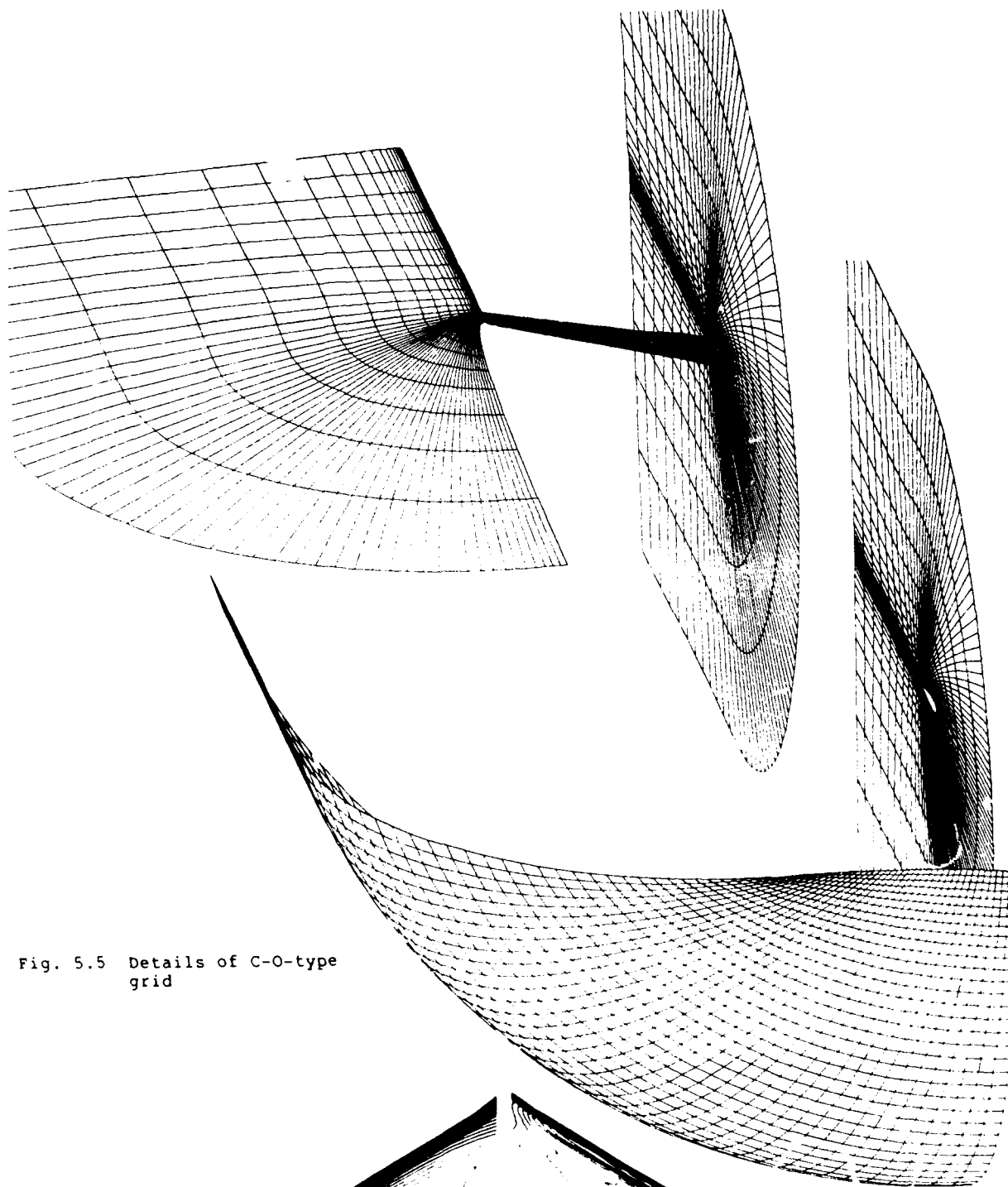
Fig. 5.5 Details of C-O-type
          grid

lower side

upper side

zonal boundary

zonal boundary
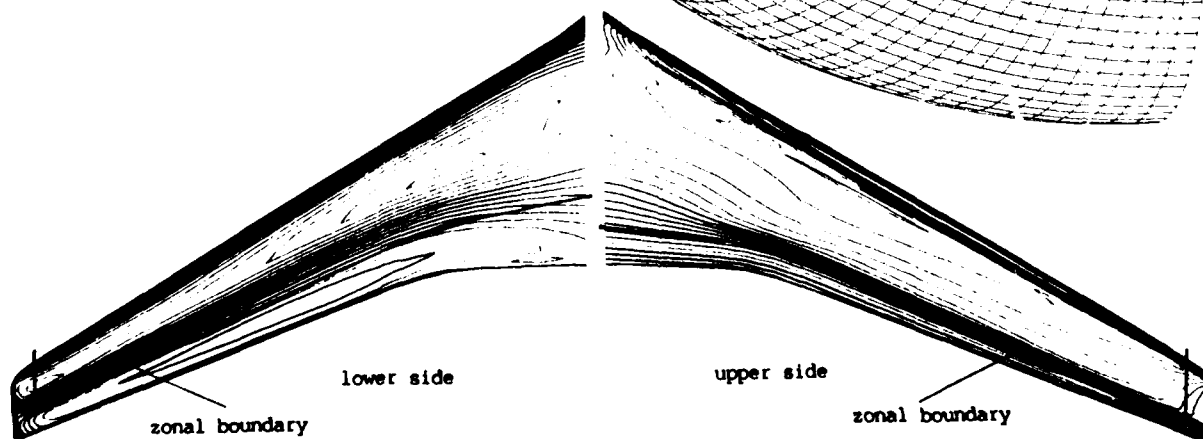
Fig. 5.6 Isobars (CCPNS solution at M_∞=0.78, α=2.2°, Re=7,000,000)
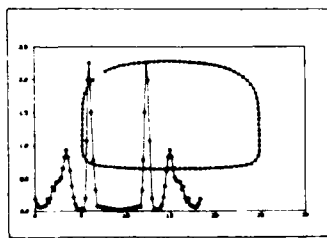
Fig. 6.1 Radius of curvature of body cross section no smoothing
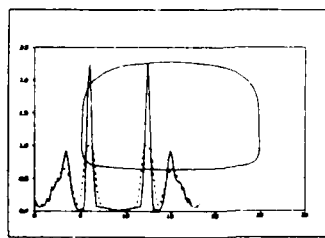
Fig. 6.2 Radius of curvature of body cross section medium smoothing
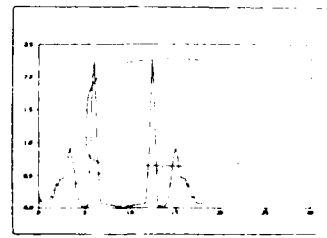
Fig. 6.3 Radius of curvature of body cross section strong smoothing
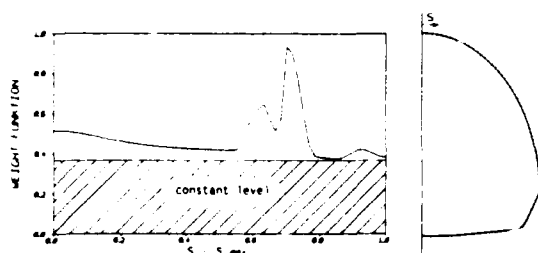
Fig. 6.4 Typical grid point clustering weight function w
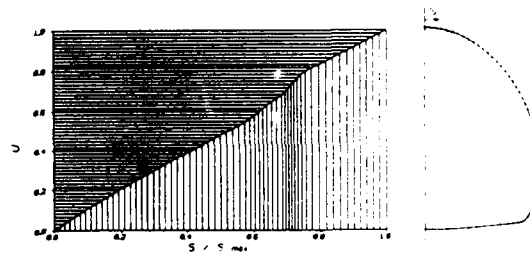
Fig. 6.5 Integral u of weight function for clustering of grid points
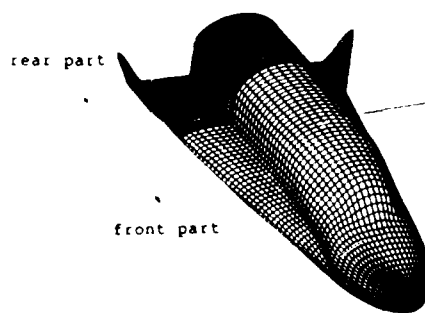
Fig. 6.6 HERMES – Computational grid on body
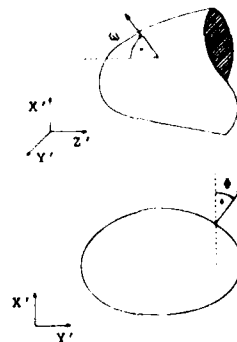
Fig. 6.7 Definition of angle ω and φ
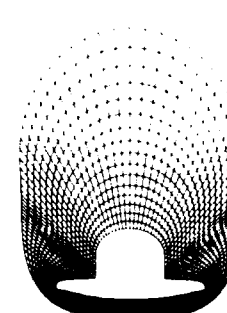
Fig. 6.8 One block grid about HERMES body in spanwise cross section
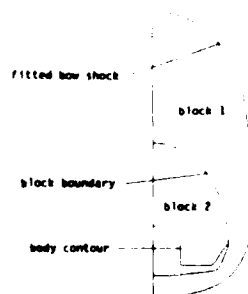
Fig. 6.9 Schematic view of blocks of computational grid
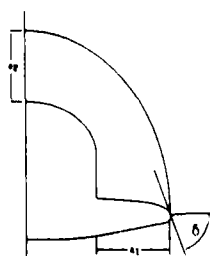
Fig. 6.10 Definition of the axes of elliptic block boundary curve
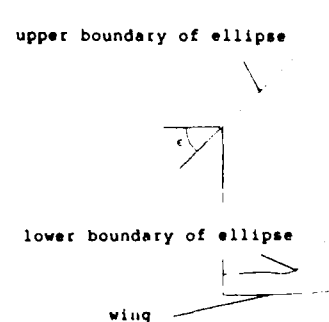
Fig. 6.11 View of block boundary and definition of angle ε
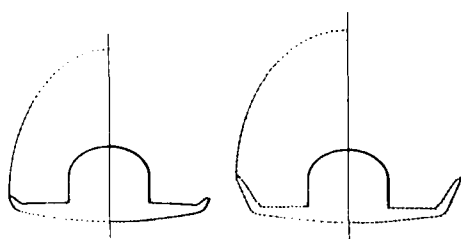
Fig. 6.12 Comparison of the original geometry
points and grid points of computational
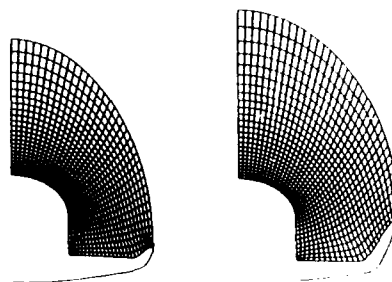grid in two planes

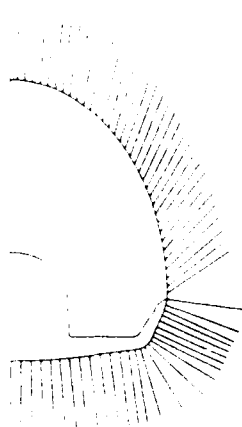Fig. 6.13 Computational grid in block 2
in two planes
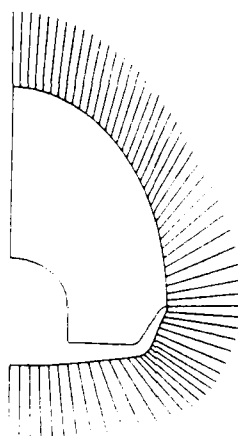

Fig. 6.14 Raw angle
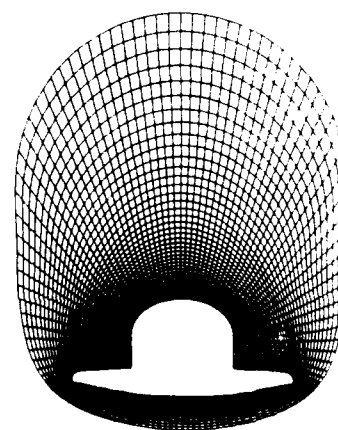distribution

Fig. 6.15 Smoothed angle
distribution

Fig. 6.16 Computational grid of two
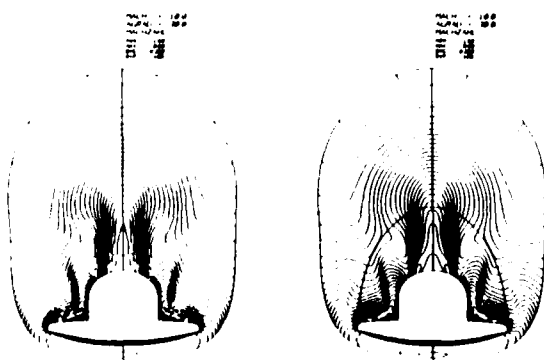block grid in cross section

Fig. 6.17 Comparison of lines of constant Mach number
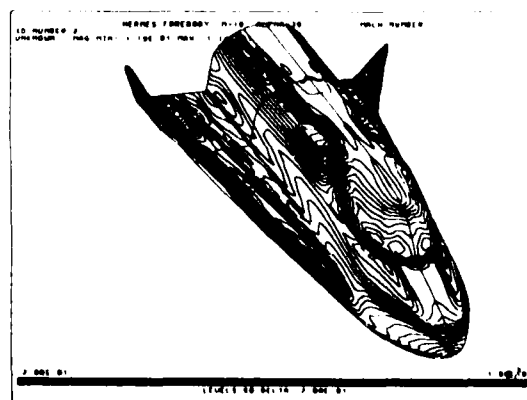on one-block and two-block grid (interpolated)

Fig. 6.18 HERMES forebody and winglet region,
$M_\infty = 10$, $\alpha = 30°$, ideal gas. Lines
of constant Mach number on body

MULTIBLOCK TOPOLOGY SPECIFICATION AND GRID GENERATION
FOR COMPLETE AIRCRAFT CONFIGURATIONS

by

Steve Allwright
Senior Research Engineer
British Aerospace (Commercial Aircraft) Ltd
Airlines Division
Hatfield, Herts AL10 9TL
United Kingdom

Summary

The ability to calculate the flow around complex aircraft geometries is fundamentally controlled by the ability to generate grids of suitable structure and quality around the configurations of interest. This paper discusses the approach to Multiblock topology specification and grid generation pursued within British Aerospace, targeted to make Multiblock flow prediction methods available for use at all stages of the aerodynamic design process. The grids and computed flow solutions for a number of complex geometries are shown, and the capability for rapid systematic analysis of similar configuration geometries is illustrated.

1. Introduction

The aerodynamic optimisation of an aircraft design requires the use of a multitude of computational and experimental techniques. The design of civil aircraft in particular places stringent demands on the accuracy and generality of these techniques. as fractional improvements in aerodynamic performance translate into significant fuel or payload operating benefits. The most fuel efficient powerplants for these aircraft are getting ever larger with ultra-high bypass or unducted fans offering significant performance benefits over conventional turbofans. These engines interact strongly with the aerodynamics of the wing, and as a result, it is no longer possible to consider the optimisation of an aircraft configuration design without conducting complete configuration flow simulations.

These requirements on accuracy of flow simulation and generality of application have governed our approach to computational flow modelling for aircraft configurations. Encouraged by the pioneering work of the Aircraft Research Association (ref. 1) in Multiblock grid generation, and work within British Aerospace (ref. 2) developing an efficient finite volume Euler Multiblock flow solver, an opportunity was seen to develop computational techniques for the systematic analysis of complete aircraft configurations : a method that would allow the optimisation of a wing design fully accounting for engine installation interference effects. Furthermore, the same general purpose Multiblock codes could be used to analyse a wide range of aircraft configurations.

The particular approach to Multiblock that we have pursued, considers multiple blocks of curvilinear grid joined face-to-face without overlap or holes to cover the entire flow domain. In this way, blocks of grid can be joined to form the optimum grid structure for modelling the flow around component geometries : C-grids for wings. O-grids for bodies. etc. (fig. 1).

Multiple blocks of curvilinear grid joined to form the
optimum grid structures for component geometries :

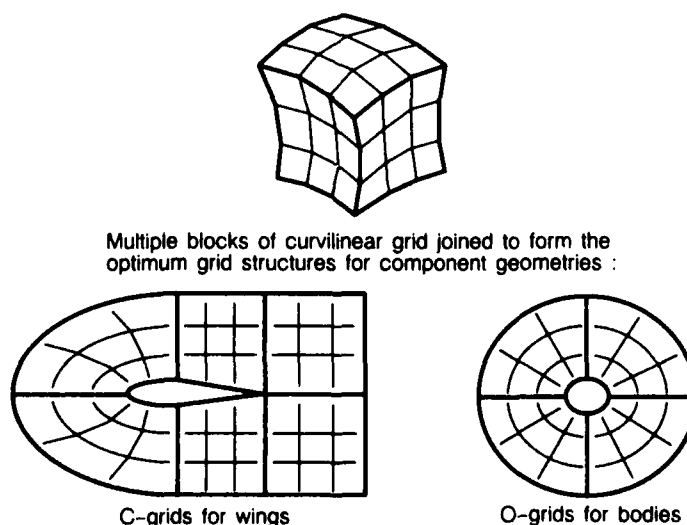C-grids for wings   O-grids for bodies

Figure 1 : The Multiblock Concept

No restriction is placed on the orientation of a block face relative to its neighbouring face, as long as grid points correspond one-to-one across the interface. At the boundary to the flow domain a single boundary condition type is imposed over the block face, according to whether the face adjoins a solid surface, inflow or outflow boundary.

It should be noted that compared with alternative multiple block techniques, for example those that allow mixed boundary conditions over a block face, the current method makes use of a larger number of smaller blocks. The logical connection between these blocks is however very simple, and this has allowed formulation and implementation of grid generation algorithms that iteratively relax inter-block boundary grid points as the overall grid itself is generated. This ensures smoothness of the grid across the block boundaries, but more importantly from a practical viewpoint, it absolves the user from having to specify the shape of the boundaries. The large number of small blocks also provides an opportunity to exploit parallel processing efficiently in the various Multiblock solution algorithms.

2. Strategy for Multiblock grid generation

While simple in principle, considerable practical difficulties are encountered when the Multiblock grid for a complete aircraft configuration is to be considered. Conceptual problems arise where the grid structures from various component geometries must be integrated, and practical problems arise in handling the "topology" data for the configuration (the specification of how each block adjoins its neighbours). Further problems arise in specifying detailed control of the grids, and these problems are compounded when the requirement for systematic analysis of similar geometries is considered.

At first sight, our approach to Multiblock grid generation might seem heretical, in that we attempt to specify and describe our grid as fully as possible without actually generating it ! However, it is this development of a "geometry independent grid description" that provides the key to use of Multiblock for general aerodynamic design – allowing systematic generation of grids for any number of similar configuration geometries, consistently, quickly and without further user interaction.

The process of developing the Multiblock grid description for a new configuration type, starts with a formal definition of the configuration in terms of how the various component geometries intersect and fit together. A number of issues must then be addressed :-

- definition of the topology of a multiple-block grid structure, with optimum grid structure for modelling the flow over component geometries.

- definition of grid density and grid point clustering for the configuration surfaces, and over key grid control surfaces slicing through the flow domain.

- generation of surface grids for any specific configuration geometry, and generation of the associated field grid.

The following sections discuss our approach to these modelling issues.

3. Topology Definition

The ability to define a consistent multiple-block topology fundamentally controls whether Multiblock methods can be applied to a particular configuration. While endeavouring to develop techniques to facilitate the process of topology definition, it is therefore essential to maintain complete generality, so as not to preclude the modelling of new types of configuration.

It was reasoned that the initial sketching of a block structure on paper could be formalized in the graphic construction of a 3-d wire-frame schematic of the block structure. This would allow any coherent block structure to be specified, and the physically representative nature of the schematic would allow holes or overlap in the block structure to be identified at an early stage. Analysis routines acting on the schematic could then automatically generate the lists of topology indexing data required to formally specify the block structure for the various Multiblock calculation methods.

3.1 Graphical Method

Initially, a direct graphical method was devised to enable construction of a multiple-block wire-frame schematic by means of cursor input to a 3-view representation of the topology (fig. 2). At any time, the cursor can be used to establish a current x, y or z working plane, and the complete ordinates of a point on that plane are then specified by cursor input to the third orthogonal view window. A single block can be defined by tracing out the position of its 8 vertices, or compound block building utilities can be invoked to perform stacking, splitting or mirroring of multiple-block structures. These utilities in particular facilitate the generation of well structured topologies.

Areas of locally detailed block structure can then be introduced into the topology by deleting blocks to create a hole, and mapping in an independently generated detailed sub-topology. In this way for example, the block structure for a single turbofan engine installation could be generated once, and be inserted into both the inboard and outboard engine locations in a wing-fuselage topology.
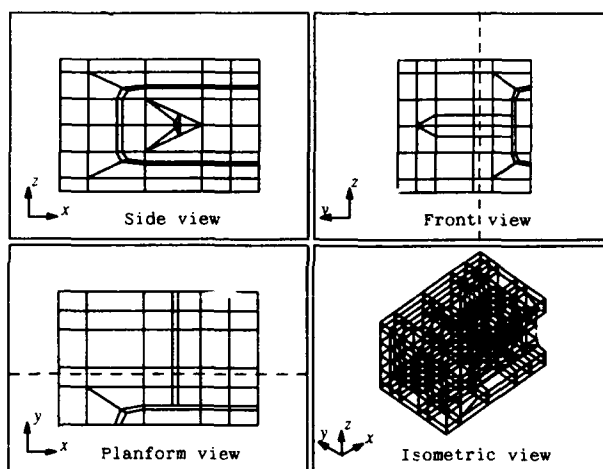
Figure 2 : Wire-frame schematic of wing-fuselage topology.

3.2 Automatic Method

Through the experience gained in graphical block decomposition and in control of the associated grids, it has been possible to devise various rules and strategies for block decomposition. These rules are being progressively implemented in an automated block decomposition method, that generates a wire-frame schematic to represent the complete field grid topology, given just a simple block representation of the configuration to be modelled. The configuration is defined in terms of a collection of cuboidal blocks, positioned in space to represent the relative position of the various geometric components within the configuration. The grid structure local to each component is considered initially as a "hyper-cube" structure (fig. 3) with a block of grid sitting on each face of the component to form a complete 0-0 type wrap-around grid structure. The block of grid on any face can then be collapsed, so for example, if the blocks adjacent to the x_{max}, y_{min} and y_{max} faces are collapsed, then the classic C-H grid structure for wings is established. In this way, the appropriate wrap-around grid structure for any geometric component of arbitrary orientation can be specified : wings, pylons, fuselages, stores, smoothly closed wing-tips etc.
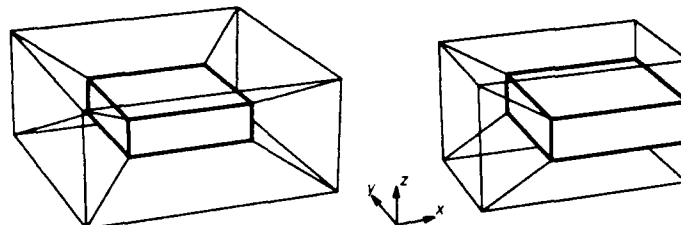

Figure 3 : Initial 'hyper-cube' grid topology, and C-H topology.

The technique has been further generalized to model internal block structures for inlets and through-flow nacelles. All these component grid structures then fit together within a topologically cartesian block matrix, to cover the entire flow field. The 3-d wire-frame topology schematic that is generated can of course be edited graphically so any locally complex grid structure beyond the current scope of automation can be built into the topology interactively.

This automated block decomposition capability is illustrated by the definition of the complete field grid topology for an executive jet configuration, with aft-fuselage mounted turbofan engine installation (fig. 4). In this case, a 7-block representation of the configuration was processed to give a 1697-block grid structure to represent the external flow field including through-flow nacelle representation (see figs. 13 & 14 for grid and flow solution).
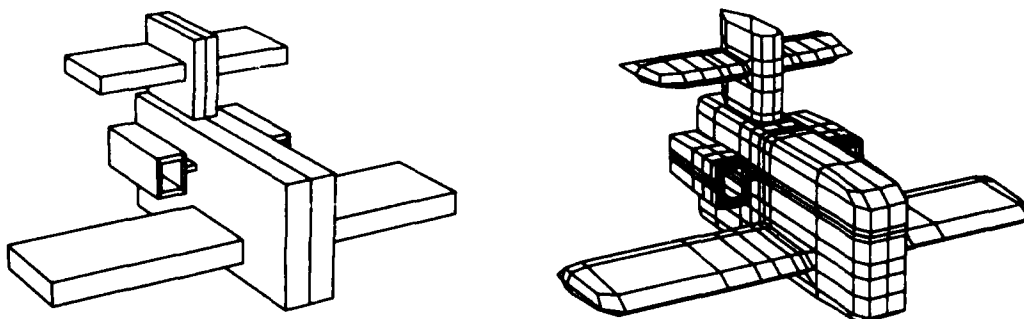

Figure 4 : Automatic block decomposition for executive jet.

4. Grid point control along boundaries

The definition of grid point density and clustering in a Multiblock grid, is achieved through the definition of grid point spacing along key geometric lines on the configuration surface geometry and over selected grid control surfaces slicing through the field grid. These grid control lines will include all significant geometric features (leading and trailing edges, intersection lines, top and bottom fuselage centrelines) allowing control of grid clustering over the configuration surfaces, and will also include grid control lines hanging between the configuration and the outerboundary, allowing control of grid point spacing in the direction normal to the configuration surfaces.

It should be noted that the shape of such grid control lines can be extracted from the geometric database for the configuration, and it is only the form of the distribution of grid points along the control line that need be specified in the grid description for the configuration. Again, the philosophy of specifying these distributions in a form independent of a specific geometry has been pursued, so that the grid description can be applied to any number of geometries of the configuration type in systematic design analysis studies.

It is useful to consider an example to illustrate the mechanisms used to control grid point alignment and clustering along grid control lines. Fig. 5 illustrates Multiblock surface grids for the plane of symmetry and fuselage for a wing–fuselage configuration. Grids with and without grid point control are shown.
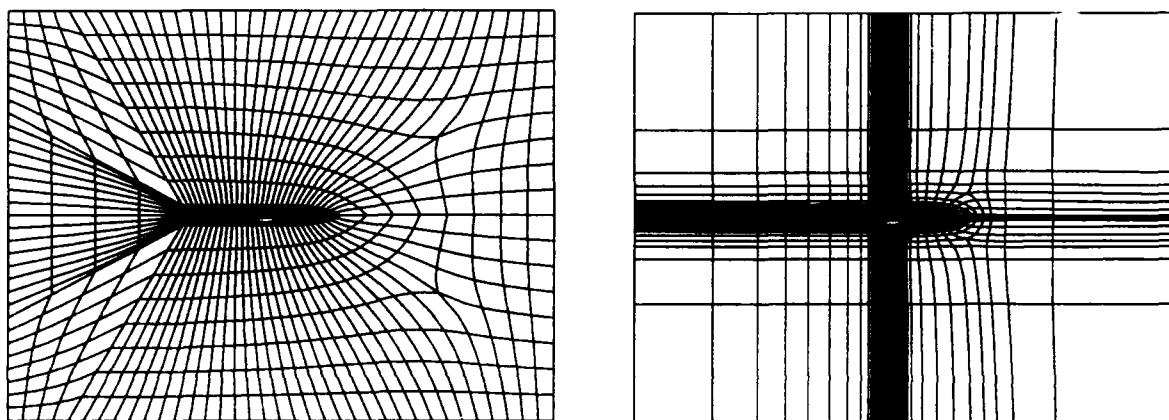

Figure 5 : Fuselage and plane of symmetry surface grids, before and after control of boundary point spacing.

We can describe the features of the controlled grid compared with the default equi-spaced grid, in terms of the positional alignment of key features, and the clustering or spacing of intermediate grid points :-

- Align grid point on fuselage top centreline with the wing trailing edge.
- Cluster grid points on wing–fuselage intersection line to leading and trailing edges.
- Set cell height normal to wing leading edge equal to grid point spacing around wing leading edge.
- Set clustering on fuselage top centreline to match clustering at trailing edge of wing.
- Reduce clustering at rear of fuselage.
- Align grid points on top outerboundary with the wing trailing edge, and rear of fuselage.
- Set grid point spacing on outerboundary to match that along fuselage centreline.
- Cluster grid points towards fuselage.
  ...etc.

These forms of grid control are essentially similar to features that might be coded up in the standalone grid generator for a single-block wing-fuselage grid, or might literally be the actions executed by an engineer generating grids at a CAD workstation building up a grid interactively around a specific geometry.

We have sought to retain the attractive features of both approaches, by devising a set of geometry independent standard format grid point alignment and clustering instructions. These instructions or labels are associated with the grid description or topology data. A custom interactive CAD tool has been developed that allows the setting and manipulation of these labels by graphical picking and movement of grid points. This is closely integrated with the surface grid generation modules allowing the development of the grid control for a complete configuration in a single interactive session.

When satisfactory grids are achieved, the full grid description is saved. This grid description describes the status of the final grid control for the configuration (rather than a list of incremental edit instructions) allowing direct generation of grids for any number of similar configuration geometries, consistently, quickly and without further user interaction.

5. Surface grid generation

The surface grid generator comprises firstly an interpreter that specifies the position of the boundary grid points according to the actual geometry and the grid control labels, and secondly the surface grid generation modules themselves. Elliptic p.d.e. grid generation techniques rather than algebraic or interpolation techniques are used because of the requirements to cater for singular points and to relax inter-block boundaries (ensuring smoothness of grid across interfaces).

A variety of elliptic solvers have been implemented, both to initialise the surface grids (ref. 3) and to calculate the final high quality grids. Hybrid solution procedures can be defined for any surface in the topology to allow solution strategy and final grid control options to be optimised for that surface.

5.1 Thompson method

The primary grid generation method used is that due to Thompson (ref. 4) working in terms of the section/generator definition of the geometric surfaces, or an x-y type parameterisation of outerboundary and control surfaces :-

$$\alpha \left(x_{\zeta\zeta} + \phi\, x_\zeta \right) - \beta\, x_{\chi\eta} + \gamma \left(x_{\eta\eta} + \psi\, x_\eta \right) = 0$$

$$\alpha \left(y_{\zeta\zeta} + \phi\, y_\zeta \right) - \beta\, y_{\chi\eta} + \gamma \left(y_{\eta\eta} + \psi\, y_\eta \right) = 0$$

$$\alpha = x_\eta \cdot x_\eta + y_\eta \cdot y_\eta$$

$$\beta = x_\eta \cdot x_\zeta + y_\eta \cdot y_\zeta$$

$$\gamma = x_\zeta \cdot x_\zeta + y_\zeta \cdot y_\zeta$$

The ϕ and ψ grid control terms along boundaries are derived using the Thomas and Middlecoff formulation (ref. 5) to propagate boundary point spacing through the grid (first term) and with an additional term that is iteratively updated to force orthogonality of the grid to fixed boundaries. This second term is derived from that reported by Thompson (ref. 6) to account for the curvature of the family of grid lines approaching the boundary, but calculating the curvature term using the target grid line slope at the boundary and the slope evaluated at 1/2 a cell out from the wall. This technique has the benefit of using just one point in the field and has proved very robust :-

$$\phi = \left(x_{\zeta\zeta} \cdot x_\zeta + y_{\zeta\zeta} \cdot y_\zeta \right) / \gamma + 2 \left(x_\zeta \cdot x_{\eta 1/2} + y_\zeta \cdot y_{\eta 1/2} \right)$$

$$\psi = \left(x_{\eta\eta} \cdot x_\eta + y_{\eta\eta} \cdot y_\eta \right) / \gamma - 2 \left(x_\eta \cdot x_{\zeta 1/2} + y_\eta \cdot y_{\zeta 1/2} \right)$$

The effect of this orthogonality term is illustrated for a grid control surface through an integrated wing-pylon-nacelle-propeller installation (fig. 6).
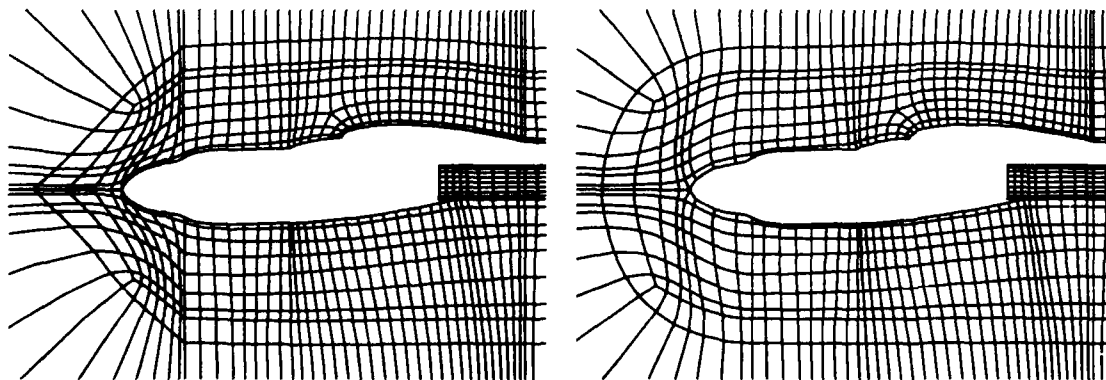

Figure 6 : Effect of grid orthogonality control

The formulation of the grid control and the recommendations for interpolating it through the field apply to a single block curvilinear grid structure. While analogous interpolation techniques can be applied in a multiple-block environment, special consideration must be given to the formulation of grid control along grid lines approaching singular points, where the local grid structure is non-cartesian.

In the same way that the grid control is updated to achieve orthogonality to fixed boundaries, so the grid control along lines approaching singular points can be updated to control the angle between the grid lines at the singular point itself (fig. 7). If full mutual orthogonality is specified for the singular point then an equal angle between grid lines at the singular point will be achieved.
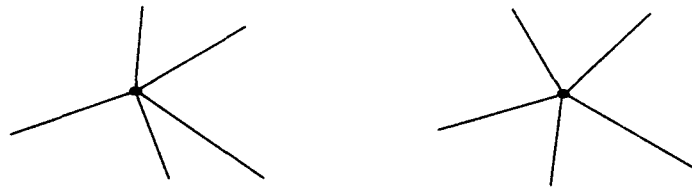
Figure 7 : Update of grid control at singularities
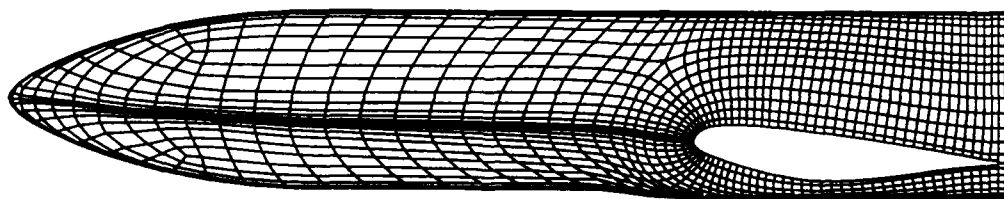
5.2 Curved surface formulation

The use of a Thompson solver working simply in terms of the section/generator definition of a curved surface, is in general quite adequate for the majority of geometric components in an aircraft configuration. However because it does not work in physical space, orthogonality of the grid to fixed boundaries cannot be directly controlled. A further problem can arise in the case of highly stretched and sheered surface definitions. because the grid inevitably depends to a degree on the detail of the section/generator definition.

A curved surface formulation of the Thompson solver has been developed within the B.Ae. Multiblock surface grid generator by Forsey and Billing of the Aircraft Research Association, a development of their work reported in ref. 7. The bi-parametric surface patch definition of component geometries is interrogated throughout the iterative solution strategy, and the surface metrics are used to decouple the surface grids from the detail of surface definition. Expressions analogous to the Thomas and Middlecoff formulation can be derived for grid control over curved surfaces, and a similar strategy for updating the grid control can be applied to achieve orthogonality of the grid to boundaries and to control the grid near singular points.

The effectiveness of this formulation is illustrated in the generation of surface grids for the fuselage of a wing–fuselage configuration. Orthogonality of the grid is now achieved around the wing intersection line, and better control of the three point singularity is achieved on the forward fuselage.
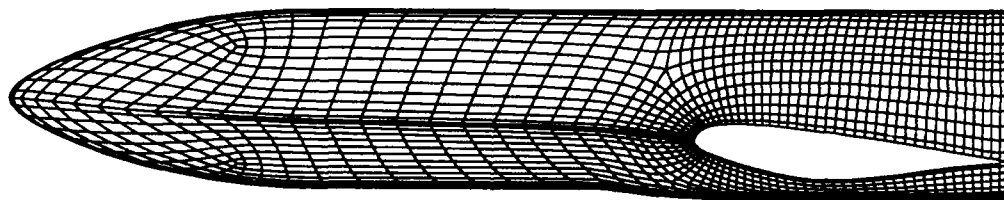
Original section/generator formulation


Curved surface formulation

Figure 8 : Comparison of surface grid generation techniques for fuselage.

6. Systematic analysis of similar geometries

Optimisation of the aerodynamic design of any aircraft configuration requires analysis methods that can be applied systematically in the study of a range of geometries. For example, in the engine installation studies for civil transport aircraft, the effect of varying nacelle incidence and position relative to the wing must be assessed, and the section and camber of the pylon must be controlled to optimise the flow on the undersurface of the wing.

The geometries to be analysed in these parametric studies are essentially similar in form and proportion, however the detailed profile of any part of the geometry may vary between cases. In Multiblock terms, the same topology, grid structure and grid point distribution will be required for all these geometries to achieve consistent flow field simulations. The use of a geometry independent grid description as described earlier ensures this similarity of grid quality between cases, and thus provides the basis for systematic analysis of any number of geometries of the configuration type.

The flow chart below illustrates the processes and user interactions involved in using Multiblock for the systematic analysis of configuration geometries. With the topology database for the configuration type established, there is only minimal user input required to complete the flow simulation :-
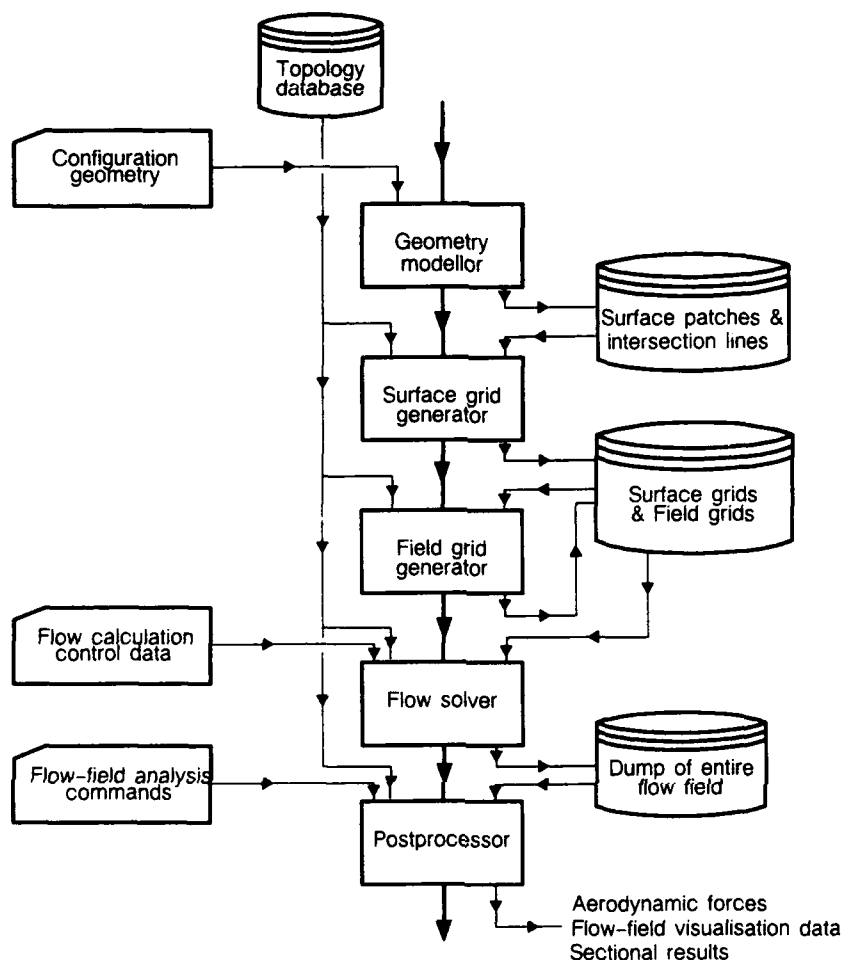
Flow chart summarising analysis of a configuration using Multiblock.

7. Applications

The various graphical and automated techniques for topology definition and surface grid generation have been used within B.Ae. on a number of aircraft projects (fig. 9). The geometric modelling and field grid generation codes developed at A.R.A. Bedford, and the Multiblock Euler code developed at B.Ae. Bristol are used to complete the Multiblock flow field analysis for the configurations. The following subsections discuss selected applications in detail.

7.1 Wide-body civil transport aircraft

A number of Multiblock topologies have been established in support of B.Ae.'s wide-body civil transport design programmes. Under-wing turbofan engine installations are modelled using a graphically generated 368-block 300.000 cell topology (fig. 10). This features C-grid structures around the wing and pylon, and around each section of the nacelle to form a C-O structure. This nacelle O-grid structure adjoins a cartesian block of grid along the nacelle axis (fig. 11) and the tubular grid structure extends upstream and downstream of the nacelle.

Alternative topologies have been generated automatically, that consider the pylon to hang straight below the wing, or to protrude forward from the wing leading edge. This later topology that is more representative of typical configuration geometries, avoids the highly sheered grid structure against the pylon side (fig. 12).

The flow solution for all these topologies compares favourably with wind tunnel results. The different pylon grid structures show differencies in detailed predictions in the vicinity of the pylon, however the effect of pylon grid topology on the gross aerodynamic interaction effects with the wing are minimal. A number of different nacelle pylon geometries have been studied using Multiblock alongside comprehensive wind tunnel test programmes.

7.2 Executive aircraft

The block structure used for modelling the executive jet was specified automatically (fig. 4), and the resulting 1697-block 1,000,000-cell grid structure is illustrated in figure 13. This features C-grid structures around the section of all component geometries – wing, pylon, fin and tail, and around each section of the through-flow-nacelle. As with the civil transport configuration, the tubular grid structure of the nacelle extends both upstream and downstream to the outerboundaries.

The engine installation for this configuration lies just forward of the wing trailing edge so there is a strong aerodynamic interference effect with the wing. This is illustrated by the contours of surface pressure depicted in figure 14, showing the area of decelerated flow at the wing root trailing edge, caused by the blockage effect of the nacelle. The detail of flow in the pylon-nacelle-fuselage gully was also represented by the calculation, and compares favourably with wind-tunnel results.

It is interesting to note in this configuration, that geometrically the leading edge of the tailplane lies behind the trailing edge of the nacelle, however because of the sweep and sheering of the grid structure required to model the leading edge of the fin, the tailplane leading edge is topologically forward of the nacelle trailing edge.

8. Concluding remarks

The combined use of graphical and automated techniques in Multiblock topology definition and surface grid generation, has facilitated the application of Multiblock to a wide range of complex aircraft configurations. In particular, the use of a geometry independent grid description specifying the generic grid for a configuration type, has allowed the systematic analysis of numerous similar geometries in parametric design studies.

References

[1] WEATHERILL, N.P., SHAW, J.A., FORSEY, C.R. and ROSE, K.R. – *A Discussion on a Mesh Generation Technique Applicable to Complex Geometries.* AGARD-CP-412, 1986.

[2] DOE, R.H., PAGANO, A., BROWN, T.W. – *The Development of Practical Euler Methods for Aerodynamic Design.* Proc. Int. Congress of Aerospace Sciences Paper 1.4.2, 1986.

[3] ALLWRIGHT, S.E. – *Techniques in Multiblock Domain Decomposition and Surface Grid Generation.* Proc. 2'nd Int. Conf. on Numerical Grid Generation in Computational Fluid Dynamics, Miami 1988.

[4] THOMPSON, J.F., THAMES, J.P. and MASTIN, C.W. – *Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing any Number of Arbitrary Two-Dimensional Bodies.* JCP, Vol 15, 1974.

[5] MIDDLECOFF, J.F. and THOMAS, P.D. – *Direct Control of the Grid Point Distribution in Meshes Generated by Elliptic Equations.* AIAA 79-1462, 1979.

[6] THOMPSON, J.F. – *Composite Grid Generation for General 3-D Regions.* Proc. 1'st Int. Conf. on Numerical Grid Generation in Computational Fluid Dynamics, Landshut 1986.

[7] FORSEY, C.R. and BILLING, C.M. – *Some Experiences with Grid Generation on Curved Surfaces using Variational and Optimisation Techniques.* Proc. 3'rd Conf. on Numerical Methods for Fluid Dynamics, Oxford 1988.
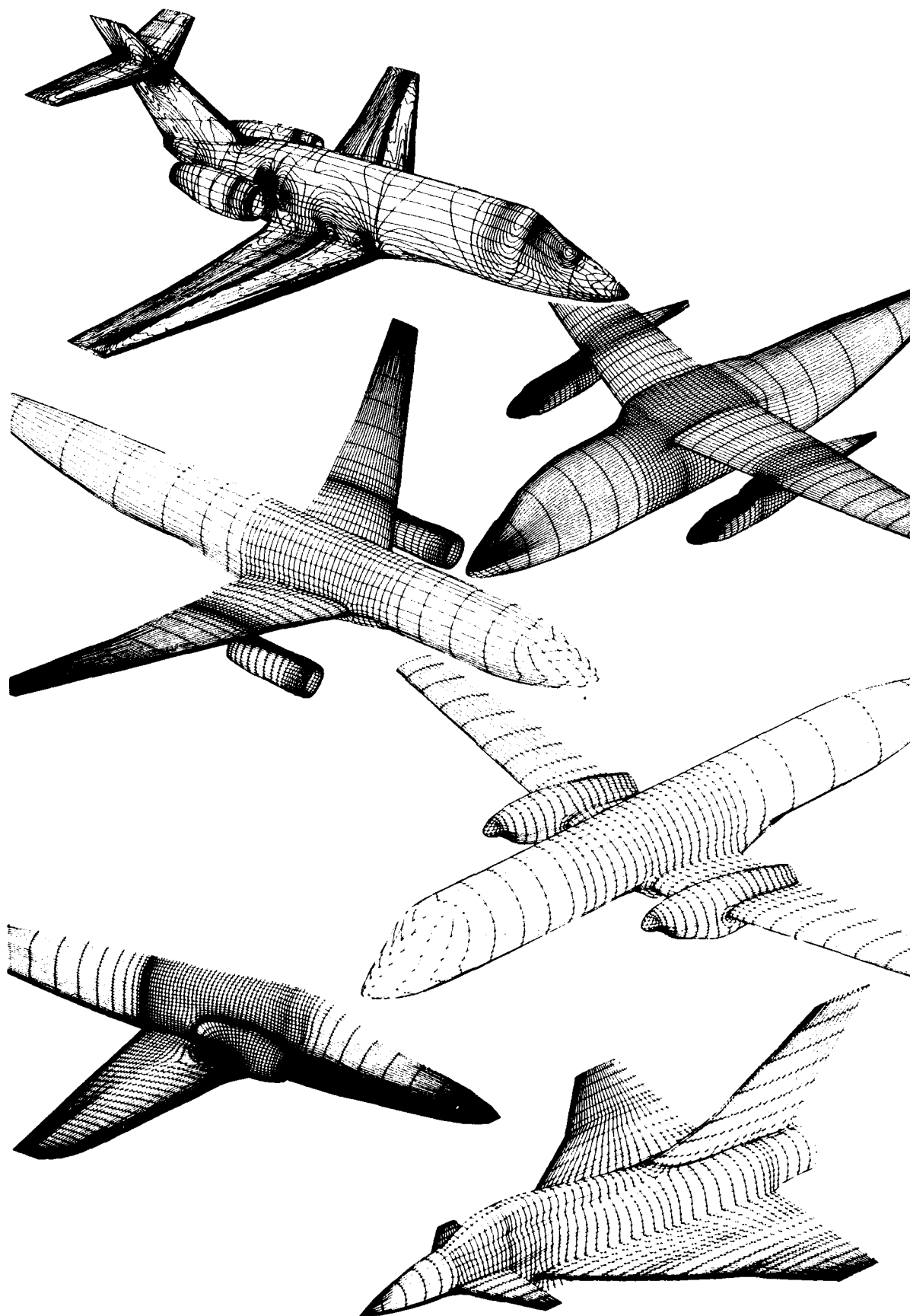
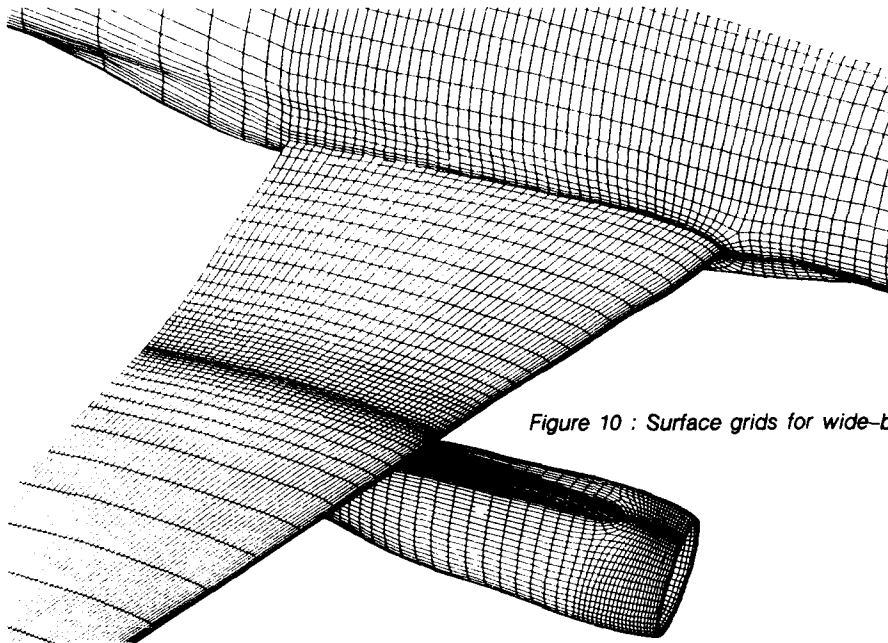Figure 9 : Extent of the application of Multiblock within B.Ae.

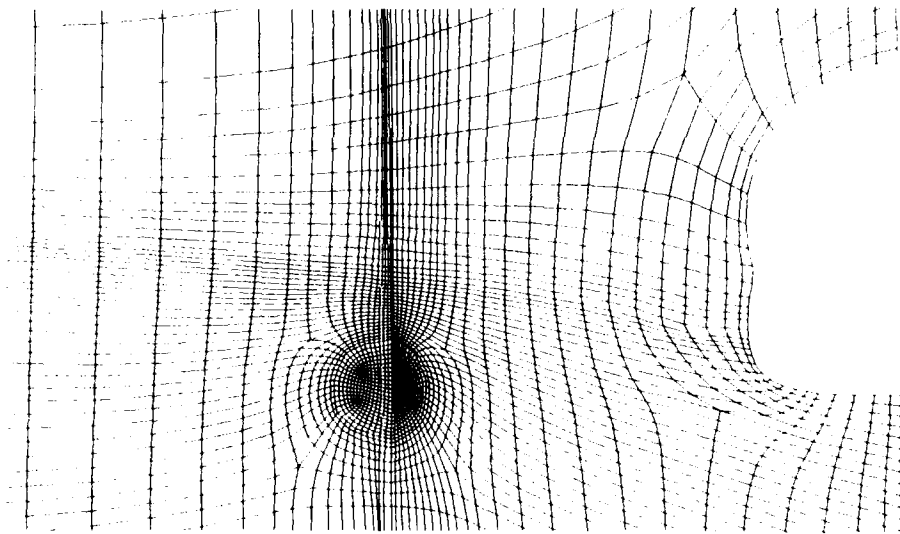Figure 10 : Surface grids for wide–body civil transport aircraft.

Figure 11 : Grid for transverse slice at wing trailing edge
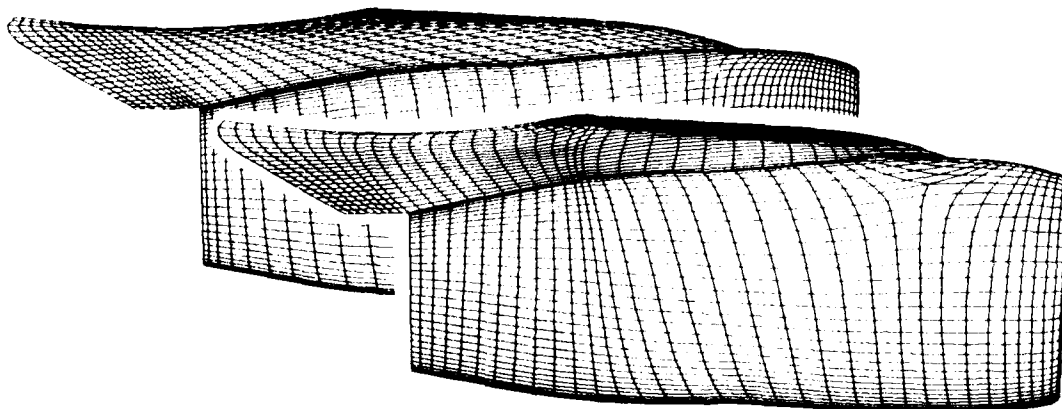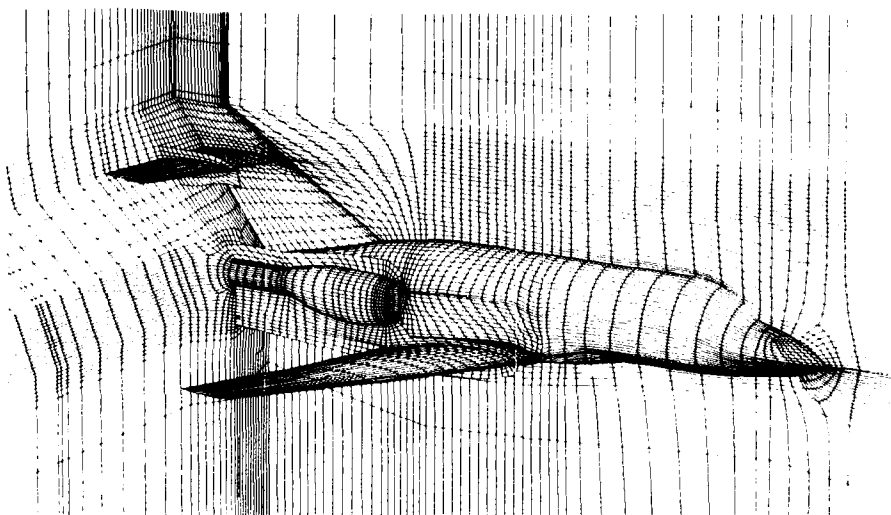

Figure 12 : Alternative topologies for pylon.

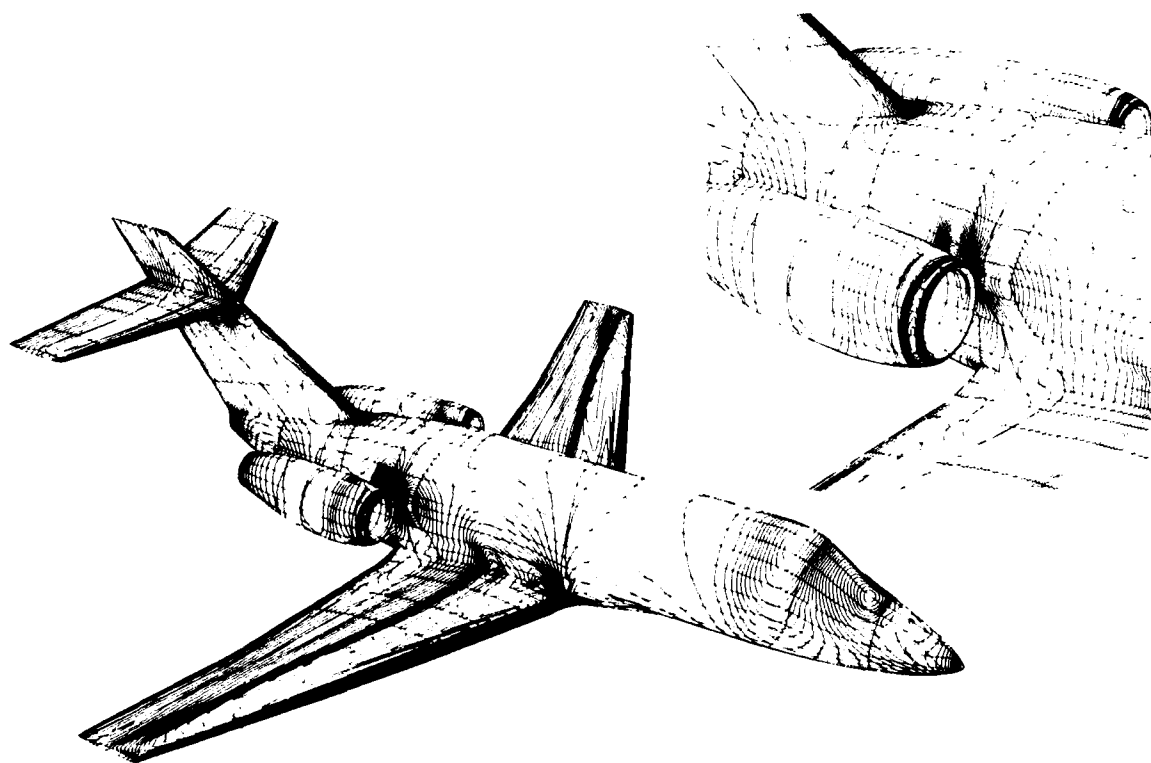Figure 13 : Multiblock surface and field grid for executive aircraft.

Figure 14 : Contours of surface pressure for executive aircraft.

THREE-DIMENSIONAL ADAPTIVE GRIDS WITH LIMITED SKEWNESS

by

Renzo Arina
Department of Aerospace Engineering
Politecnico di Torino
Corso duca degli Abruzzi 24
I 10129 Torino
Italy

Summary

A grid generation technique for curved surfaces and three-dimensional regions is presented. In the two-dimensional case the set of solutions of the proposed grid generator belongs to the class of quasiconformal mappings, and it is shown that under appropriate conditions, it represents unfolded orthogonal coordinates. The isothermic coordinates are a particular case of this wider family of mappings. In three dimensions the solution of the mapping system is harmonic. Different kinds of stretching including an adaptive control of the mesh clustering are presented.

1. Introduction

As numerical algorithms for the solution of the governing equations of fluid-dynamics are progressing, as well as computer resources, there is an increasing demand to simulate flows over more realistic aerodynamic shapes. As the complexity of the flow domain increases, the role of mesh generation within the overall numerical simulation become more important.

The objectives of grid generation are essentially two. Firstly the grid must accurately represent the geometrical boundaries of the domain. Secondly the mesh distribution should resolve all the scale lengths of the flowfield solution. This last constraint is accomplished by an appropriate clustering of the grid points in the parts of the domains where important flowfield gradients are expected. If the positions of the high-gradient regions are unknown or they evolve on time, the mesh generator and the flow algorithm should interact in order to adequately redistribute or add grid points.

Numerical grids can be classified into two main categories: unstructured and structured meshes [1]. Unstructured meshes are formed by a set of points and a connection between them, forming triangles in two dimensions or tetrahedrons in three dimensions. They are very flexible and it is possible to easily treat multiple-connected domains, however the data handling is quite involved and moreover certain efficient numerical algorithms, such as ADI techniques, cannot be implemented.

In body conforming structured grids the connections between points is defined throught the curvilinear coordinate system. The most fruitful strategy for generating structured grids in complex configurations is the multiblock approach. The physical region is segmented into subregions bounded by six curved surfaces. In this way each block is chosen to be topologically equivalent to a cuboid and therefore be mapped into a unit cube in computational space without change in topological structure. Cartesian grids in the unit cubes in computational space map to curvilinear grids in physical space.

By subdividing the flow domain into a set of blocks, it is possible to distinguish the subregions or blocks where an accurate grid is necessary because of geometric constraints or because the flow field developing into the corresponding physical region exhibits important gradients, or boths. In these subregions the grid characterized by an appropriate metric, should minimize the geometric-induced discretization errors associated with the numerical algorithm employed to solve the flow equations. When dealing with finite-difference or finite-volume algorithms it is advisable to have smooth grids, with a limited departure from orthogonality. Moreover a dynamically-adaptive clustering of points in the regions where the flow variables display important gradients may be advisable.

The accurate description of the boundaries of the physical domain is a crucial aspect of the numerical simulation. Moreover the surface grid generation has a dominant effect on the quality of the volume grid. In this work a grid generation technique for generating structured grids on curved surfaces is presented. In two dimensions, the set of solutions of the proposed grid generator belongs to the class of quasiconformal mappings [2], and they represent orthogonal curvilinear coordinates. Moreover it is proved that the differential model admits regular continuous solutions, without local foldings. Subsequently the technique is extended to three-dimensional regions, and it is shown that the solution represents harmonic maps.

2. Curvilinear coordinates in Euclidean spaces

Grid generation of curvilinear coordinates in Euclidean spaces consists in the construction of the coordinate system $\{\xi^i\}$ corresponding to given metric tensor components, that is in calculating the transformation $x^i(\xi^j)$, $\{x^i\}$ being the cartesian frame.

The mapping $x^i = f(\xi^j)$ defined on the domain $\mathcal{D}$ represents a coordinate transformation if the function $f : \mathcal{D} \rightarrow D(\mathcal{D}, D \in R^n)$ is one-to-one in each point $P \in \mathcal{D}$ and has a local inverse which is one-to-one on the image of a neighborhood of P. Then if in $P \in \mathcal{D}$ the Jacobian determinant $J(f) \neq 0$ by the

inverse function theorem it follows that there exists in a neighborhood of P a regular coordinate system without singularities or local foldings. These conditions of local regularity are satisfied by a proper choice of the metric tensor components g_{ij}. However even if $J(f) \neq 0$ at all the points of $\mathcal{D}$, it does not follows that f is one-to-one on $\mathcal{D}$. In order to have a globally one-to-one mapping on $\mathcal{D}$ it is necessary to appropriately specify the physical domain D, by a one-to-one correspondence $g : \partial\mathcal{D} \rightarrow \partial D$.

The knowledge of the metric element $ds^2 = g_{ij}d\xi^i d\xi^j$ at an arbitrary point P of the space, enable us to image a frame $\mathcal{F}$ with origin at this point, with characteristics specified by the components g_{ij}. The local reconstruction of the space consists in localizing with respect to this frame $\mathcal{F}$, the frame $\mathcal{F}'$ relative to a point Q contained in a neighborhood of P. It is then necessary to find a set of equations expressing the characteristics of $\mathcal{F}'$ in function of the known frame $\mathcal{F}$. Being $d\bar{r} = dx^i \bar{e}_i$, by the definition of the vectors $\bar{g}_i$, tangent to the axes of $\mathcal{F}$, and of covariant derivative, with Γ^k_{ij} the connection coefficients, we have

$$d\bar{r} = d\xi^i \bar{g}_i \quad i = 1, n \tag{2.1}$$

$$d\bar{g}_i = \Gamma^k_{ij} d\xi^j \bar{g}_k \quad i, j, k = 1, n \tag{2.2}$$

If the functions $g_{ij}(\xi^k)$ are continuous, the symbols Γ^k_{ij} can be expressed as functions of the derivatives of g_{ij}. Equations (2.1,2.2) solve completely the problem in a neigborhood of P, and form the mapping system.

The conditions of integrability for equations (2.1,2.2) require that, for symmetric symbols $\Gamma^k_{ij} = \Gamma^k_{ji}$, the curvature tensor $R^r_{ijk} = 0$. The functions g_{ij} must satisfy this flatness condition in order to be viewed as the components of the Euclidean metric. Moreover the solution of the mapping system (2.1,2.2) represents the curvilinear system specified by the given metric tensor components, and the functions $x^i(\xi^j)$ are a local coordinate transformation in a neighborhood of P, if $g_{ij} \in C^1$ and the determinant of the matrix g_{ij}, $g \neq 0$ in P (regular metric).

For a regular metric we have $x^i(\xi^j) \in C^2$ at least, then the mapping system (2.1,2.2) can be converted into a system of second-order partial differential equations. Differentiating equations (2.1) with respect to ξ^j, by using equation (2.2), we obtain

$$\frac{\partial^2 x^r}{\partial \xi^i \partial \xi^j} - \Gamma^k_{ij} \frac{\partial x^r}{\partial \xi^k} = 0 \quad r = 1, 3 \tag{2.3}$$

The terms on the left-hand side of equations (2.3) are the components of a symmetric second order covariant tensor, called the second fundamental form of the map $x^i = f^i(\xi^j)$ [3]. The trace of this tensor, obtained by inner multiplication by g_{ij}, is called the tension field of f, and it is formed by the system of second-order partial differential equations

$$g^{ij} \frac{\partial^2 x^r}{\partial \xi^i \partial \xi^j} - g^{ij} \Gamma^k_{ij} \frac{\partial x^r}{\partial \xi^k} = 0 \tag{2.4}$$

System (2.4) is the most general set of equations which can be used for grid generation in Euclidean spaces

A mapping f is defined harmonic if and only if it is an extremal of the energy integral [4]

$$e(f)(\xi) = \frac{1}{2} \int_{\mathcal{D}} g^{ij} \delta_{mn} \frac{\partial x^m}{\partial \xi^i} \frac{\partial x^n}{\partial \xi^j} d\xi^N \tag{2.5}$$

It can be shown that equations (2.4) are the Euler-Lagrange equations arising from the variational problem of the energy integral (2.5), then their solutions $x^i = f(\xi^j)$ represent harmonic maps.

The terminology tension field can be explained by a physical picture of a harmonic map $f^{-1} : D \rightarrow \mathcal{D}$ [4]. Suppose that D is made of rubber and that $\mathcal{D}$ is made of marble: the map f^{-1} constrains D to lie on $\mathcal{D}$. At each point of D there is a vector representing the tension in the rubber at that point. It is seen that f is harmonic if and only if f^{-1} constrains D to lie on $\mathcal{D}$ in a position of elastic equilibrium.

System (2.4) can be recast into the form

$$\frac{1}{\sqrt{g}} \frac{\partial}{\partial \xi^i} \left(\sqrt{g} g^{ij} \frac{\partial x^r}{\partial \xi^j} \right) = 0 \quad r = 1, 3 \tag{2.6}$$

and it follows that the basic mapping system is formed by a set of Laplace-Beltrami equations, which can be viewed as the Euler-Lagrange equations of the variational problem of the integral

$$I = \frac{1}{2} \int_{\mathcal{D}} \sqrt{g} g^{ij} \delta_{mn} \frac{\partial x^m}{\partial \xi^i} \frac{\partial x^n}{\partial \xi^j} d\xi^N \tag{2.7}$$

3. Surface orthogonal coordinates

The development of an appropriate grid generation technique for n-dimensional spaces must begin by specifying the n-independent metric components in order to represent a particular curvilinear coordinate system. Their substitution into system (2.4) leads to a specific mapping system which, with a suitable set of boundary conditions, defines the mapping problem whose solution represents the appropriate

coordinate transformation. As it has been shown previously, the coordinate transformation satisfies the properties of local regularity if the metric is regular: $g_{ij} \in C'^1$ and $g \neq 0$ in $\mathcal{D}$. An explicit consideration of the flatness condition is not necessary. It is a requirement of integrability which is satisfied when the existence of the coordinate transformation $x^i(\xi^j)$ is proved.

A surface in the Euclidean space is individuated by two coordinates, say ξ^1 and ξ^2, holding the third fixed. If in addition the ξ^3-coordinate lines are orthogonal to this surface ($g_{13} = g_{23} = 0$), system (2.6) reads, with $r = 1,3$ and $\alpha,\beta,\gamma = 1,2$

$$\frac{1}{\sqrt{g}}\frac{\partial}{\partial \xi^\alpha}\left(\sqrt{g}g^{\alpha\beta}\frac{\partial x^r}{\partial \xi^\beta}\right) = (K_1 + K_2)n^r \tag{3.1}$$

where K_1 and K_2 are the principal curvatures of the surface, and $n^r\bar{e}_r$ the local unit normal vector forming with the tangent vectors of the surface coordinate lines ξ^1 and ξ^2, a right-handed frame.

For an orthogonal surface grid $\{\xi^\alpha\}$ we have $g_{12} = 0$, then equations (3.1) take the form, with $F^2 = g_{22}/g_{11}$, $\xi^1 \equiv \xi$ and $\xi^2 \equiv \eta$,

$$\frac{\partial}{\partial \xi}\left(F\frac{\partial x^i}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(\frac{1}{F}\frac{\partial x^i}{\partial \eta}\right) = (K_1 + K_2)n^i \tag{3.2}$$

Let $\{u^\alpha\}$ be any arbitrary surface coordinate system such that $x^i(u^\alpha)$. The metric tensor components $g_{\alpha\beta}$ of the surface orthogonal coordinates $\{\xi^\alpha\}$ must satisfy the following relations

$$g_{22} = F^2(\xi,\eta)g_{11} \quad , \quad g_{12} = g_{21} = 0 \tag{3.3}$$

where the distorsion function F is non-zero and continuously differentiable. If $a_{\gamma\delta}$ are the metric tensor components of the u^α-coordinates, by the transformation law for tensor components, equations (3.3) yield the relations

$$F\frac{\partial u^\alpha}{\partial \xi} = \varepsilon^{\alpha\beta}\frac{a_{\gamma\beta}}{\sqrt{a}}\frac{\partial u^\gamma}{\partial \eta} \tag{3.4}$$

where $a = a_{11}a_{22} - a_{12}^2$ and $\varepsilon^{\alpha\beta}$ is the permutation symbol ($\varepsilon^{11} = \varepsilon^{22} = 0, \varepsilon^{12} = -\varepsilon^{21} = 1$). Equations (3.4) are the generalized Cauchy-Riemann equations. The reason for this terminology is that they reduce to the well known Cauchy-Riemann equations of complex analysis when the surface is flat ($a_{\alpha\beta} = \delta_{\alpha\beta}$).

System (3.2) with the cartesian coordinates $\{x^i\}$ as dependent variables, can be reduced to a set of two equations with unknowns the surface coordinates $\{u^\alpha\}$. By the chain rule for differentiating composite functions and by equations (3.4), system (3.2) takes the form [5]

$$\frac{\partial}{\partial \xi}\left(F\frac{\partial u^\alpha}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(\frac{1}{F}\frac{\partial u^\alpha}{\partial \eta}\right) = \varepsilon^{\alpha\beta}\left[\frac{\partial u^\gamma}{\partial \eta}\frac{\partial}{\partial \xi}\left(\frac{a_{\beta\gamma}}{\sqrt{a}}\right) - \frac{\partial u^\gamma}{\partial \xi}\frac{\partial}{\partial \eta}\left(\frac{a_{\beta\gamma}}{\sqrt{a}}\right)\right] \tag{3.5}$$

On flat surfaces it is possible to introduce cartesian coordinates $\{u^\alpha\}$, in this case the rhs of equations (3.5), as well as of equations (3.2), vanishes, and the system takes the same linear form as the mapping system proposed in ref.[6] to generate orthogonal grids on plane domains. Also in the case of isothermic coordinates $\{u^\alpha\}$ the rhs of the mapping system (3.5) vanishes. Thus it is possible to conclude that if the surface is parametrized by isothermic coordinates $\{u^\alpha\}$, then any other orthogonal coordinate system can be obtained as solution of a grid generation technique developed for two-dimensional plane domains.

Equations (3.4) are the Euler-Lagrange equations from the variational problem of the integral

$$I_s = \int_{\mathcal{D}}(Fg_{11} + \frac{1}{F}g_{22})d\xi d\eta \tag{3.6}$$

The transformation $u^\alpha(\xi^\beta)$ for orthogonal coordinates, corresponding to the mapping of a rectangular domain $\mathcal{D}$ defined on the mathematical plane ξ^α onto a domain D, defined on the surface $x^i(u^\alpha)$, is the solution of the boundary-value problem formed by the mapping system (3.5) and by a set of appropriate boundary conditions. The boundary conditions must ensure that the image of the boundary $\partial\mathcal{D}$ coincides with the boundary ∂D on the surface. This correspondence can be obtained by specifying Dirichlet boundary conditions. However in the present case the condition of orthogonality (3.4) must hold through the boundary $\partial\mathcal{D}$. In general Dirichlet boundary conditions are not consistent with these additional constraints. It is possible to overcome this problem by representing the boundary ∂D in parametric form, and to enforce a 'shape correspondence' of ∂D with the image of $\partial\mathcal{D}$, by leaving the boundary grid points to float along the boundary ∂D between the corresponding corner points, in order to satisfy equations (3.4).

If the mapping $u^\alpha = f(\xi^\beta)$ represents an allowable mapping, the Jacobian determinant of the function $f : \mathcal{D} \to$ D must not vanish on $\mathcal{D}$. Orthogonal coordinates satisfy in each point of $\mathcal{D}$ the generalized C-R equations (3.4), then it follows that the Jacobian determinant reads

$$J = \varepsilon_{\alpha\beta}\frac{\partial u^\alpha}{\partial \xi}\frac{\partial u^\beta}{\partial \eta} = \frac{1}{F}\frac{a_{\beta\gamma}}{\sqrt{a}}\frac{\partial u^\gamma}{\partial \eta}\frac{\partial u^\beta}{\partial \eta} \tag{3.7}$$

This quadratic form is positive definite, and it vanishes only if $\frac{1}{F}$ or if both the partial derivatives $\frac{\partial u^i}{\partial \eta}$ are zero. But this cannot be true for a regular solution, then the grid will always be unfolded.

The existence of surface orthogonal grids, solutions of the boundary value problem formed by equations (3.5) and the boundary-point relocation procedure ensuring the orthogonality at the boundaries, can be proved as follows. It is known (Radò theorem [7]) that every orientable surface which carries a metric tensor can be made into a Riemann surface. Then all orientable surfaces with a metric admit a conformal structure carried by the isothermic coordinates, and by the Koebe-Riemann mapping theorem for Riemann surfaces it follows that an arbitrary region D of the surface bounded by a simple closed curve, can be mapped onto a simply-connected plane domain. If the system $\{u^\alpha\}$ is isothermic the rhs of eq.(3.5) vanishes, and the resulting mapping system can be interpreted as a particular case of the complex Beltrami equation [6]. Then from complex analysis it follows that in this case the transformation $u^\alpha(\xi^\beta)$ is a regular quasiconformal mapping [2]. Having proved the existence of isothermic coordinates, the problem of the existence of surface orthogonal coordinates is reduced to the existence of a solution of the Beltrami equation, which is known to exists. Then it is possible to state a mapping theorem for quasiconformal mappings analogous to the Riemann theorem for conformal transformation [2]. In addition, for mappings $u^\alpha = f(\xi^\beta)$ which are quasiconformal on $\mathcal{D}$, it is possible to prove that if the image of $\partial\mathcal{D}$ under f is a one-to-one sense-preserving image which consists of a simple closed curve ∂D, then the mapping f defines a global one-to-one correspondence between $\mathcal{D}$ and D.

The mapping theorem states that the solution is unique fixing the correspondence of three points on ∂D with the images of three points of $\partial\mathcal{D}$. Being interested in quadrilaterals domains D, it follows that it is not possible to map D on 'a priori' specified rectangular regions $\mathcal{D}$. For a given domain D with conformal module $M = m(\mathrm{D})$ [7], the conformal modulus of the rectangle $\mathcal{D}$ is given by the relation

$$m(\mathcal{D}) = \frac{\mathcal{M}}{\mathcal{K}} \tag{3.8}$$

K being the upper bound of the distorsion function F. For conformal mappings F=1, and it follows that the two regions must have the same conformal module [6]. For a rectangular region, the conformal module coincides with the side ratio, then it is possible to normalize the region $\mathcal{D}$ into the unit square $\bar{\mathcal{D}}$ and to treat $m(\mathcal{D})$ as an unknown stretching parameter, by adding the relation

$$m^2 = \frac{\int_{\bar{\mathcal{D}}} F\sqrt{g_{11}/g_{22}}\,d\bar{\xi}\,d\bar{\eta}}{\int_{\bar{\mathcal{D}}} \frac{1}{F}\sqrt{g_{22}/g_{11}}\,d\bar{\xi}\,d\bar{\eta}} \tag{3.9}$$

Equation (3.9) is obtained by integrating equations (3.4).

The numerical algorithm consists of two steps. Firstly, for a fixed boundary point distribution and given conformal module, the finite-difference discretizations of equations (3.5) are solved by an approximate factorization technique. Then the positions of the boundary points are adjusted in order to satisfy equations (3.4), and the conformal module is updated by solving equation (3.9).

The control of the grid spacing is obtained by specifying the distorsion function $F(\xi, \eta)$. For $F = 1$ we have the case of conformal mapping. Along a η-constant coordinate line the differential $d\eta$ vanishes then from the definition of metric, it follows that the incremental arc length, ds is given by the relation

$$ds = \sqrt{g_{11}}\,d\xi \tag{3.10}$$

Similarly along a ξ-constant line, denoting by t the arc length, $dt = \sqrt{g_{22}}\,d\eta$. From the definition of distorsion function (3.3) it follows

$$F(\xi, \eta) = \frac{t_\eta}{s_\xi} \tag{3.11}$$

The functions s_ξ and t_η represent the distribution of the arc lengths along the coordinate lines with respect to constant increments of ξ and η, and can be prescribed by any suitable stretching function. Fig.1 shows an orthogonal grid on a part of an ellipsoide, with stretching of the grid points near the corners, obtained by specifying the functions s_ξ and t_η as exponential stretchings with respect to $\{\xi^\alpha\}$. It is worth noting that this grid cannot be interpreted as the composition of a isothermic transformation and one-dimensional stretchings.

If the attraction line or point is in the interior of the domain, rather than on a boundary, then it is necessary to specify the stretching functions s_ξ and t_η as functions of $\{u^\alpha\}$. A typical control function has the form [8]

$$s_\xi = 1 - |\, \bar{t} \cdot \overline{\tau^{(t)}} \,|\, e^{A|d_{min}|} \tag{3.12}$$

where $\bar{t}$ and $\overline{\tau^{(t)}}$ are the unit vectors tangent to the attraction line and normal to the ξ-coordinate line respectively, d_{min} the shortest distance between the grid point and the attraction line, and A a decay factor. In figure 2 it is shown the grid on a part of a sphere, with concentration of the grid points with respect to a fixed point inside the domain.

From the previous considerations it follows that the most straightforward adaptive specification of the distorsion function F is given by a curve-by-curve approach [9]. Along a η-constant coordinate line the basic differential statement of the equidistribution law is

$$w(\xi, \eta)\,ds = C\,d\xi \tag{3.13}$$

For each given curve, C is a constant. However going from a curve to another curve of the same family, that constant becomes a function of the transverse variable which governs this progression. Comparing

equations (3.10) and (3.13), it follows

$$\sqrt{g_{11}} = \frac{C}{w} \qquad (3.14)$$

A similar result holds for $\sqrt{g_{22}}$ along ζ-constant coordinate lines. For a given weight function, the constant C is determined explicitly by integration of equation (3.13) along the entire curve. The weight function depends upon the physical space $\{u^\alpha\}$, moreover the coordinate line 'moves' during the iterative procedure for solving the boundary-value problem; then the value of C as well as the function g_{11} must be updated at each iteration.

The weight function is formed by a positive scalar of the type $w = 1 + cM$, where M is a non-negative function formed by some error estimate or some normalized gradient of a scalar function representative of the physical problem. And c is a non-negative constant indicating the level of importance attached to M. In the present case we used a function M based on the normalized first derivative along the coordinate line of a scalar test function f.

Figures 3-a,b show the grid around a double-ellipse shape; the grid is clustered along the shock-like fronts simulating the intersection between the bow shock and the shock originating from the canopy . It can be noted how the grid is fitting the shocks, with one family of coordinate lines being aligned to the fronts while the other one is crossing it orthogonally.

4. Three-dimensional curvilinear coordinates

The aim is to develop a grid generation technique for generating structured three-dimensional grids with smoothness, limited departure from orthogonality and adaptive clustering of the mesh points. Imposing the constraints of orthogonality $g_{12} = g_{13} = g_{23} = 0$ the metric tensor components can be expressed as follows

$$g_{ij} = \lambda_{(i)}\delta_{ij} \qquad (4.1)$$

where $\lambda_{(i)}$ is the value of the diagonal metric tensor component g_{ii}. Introducing the following notation, with no summation on repeated indeces,

$$F_{(i)} = \sqrt{g}g^{ii} \qquad (4.2)$$

from the general mapping system (2.6), we obtain the mapping system

$$\frac{\partial}{\partial \xi^i}\left(F_{(i)}\delta^{ij}\frac{\partial x^r}{\partial \xi^j}\right) = 0 \qquad r = 1,3 \qquad (4.3)$$

being $x^1 = x, x^2 = y$ and $x^3 = z$ cartesian coordinates.

As in the case of two-dimensional mappings [5,6], it is possible to obtain from the metric tensor constraints (4.1) a set of relations which represent the conditions of integrability for the mapping system. These conditions can be considered as the extension of the generalized Cauchy-Riemann relations (3.4) to the three-dimensional case, and they can be expressed in the following form, being $\overline{g}_i$ the tangent vector to the ξ_i-coordinate line,

$$\overline{g}_1 = \frac{1}{3F_{(1)}}(\overline{g}_2 \times \overline{g}_3) \qquad (4.4)$$

$$\overline{g}_2 = \frac{1}{3F_{(2)}}(\overline{g}_3 \times \overline{g}_1) \qquad (4.5)$$

$$\overline{g}_3 = \frac{1}{3F_{(3)}}(\overline{g}_1 \times \overline{g}_2) \qquad (4.6)$$

As shown in section 2, the solution of system (4.3) represents a harmonic map, and it is an extremal of the integral (2.7) which in this case reads

$$I_v = \frac{1}{2}\int_D \left(F_{(1)}g_{11} + F_{(2)}g_{22} + F_{(3)}g_{33}\right)d\xi^N \qquad (4.7)$$

The bilinear form $a(u,v)$ corresponding to the integral I_v satisfies the relation

$$a(v_i,v_j) = \sqrt{(g)}g^{ij}v_iv_j \geq \alpha \mid v \mid^2 \qquad (4.8)$$

for every point $\{\xi^i\} \in D$ and for every $u \in R^N$, $\alpha > 0$, being the Beltrami-Laplace operator elliptic. Then the bilinear form is coercive, and it follows that for a domain with a smooth, or a piecewise smooth, boundary, there exists a unique weak solution of the boundary-value problem formed by system (4.3) and a suitable set of boundary conditions [10]. The bilinear form (4.8) is symmetric then the solution minimizes I_v.

Unlike the two-dimensional case, where it was possible to state the condition for the existence of a regular solution, in three dimensions the metric constraints and the relations (4.4,4.5,4.6) form an overdetermined system, then the class of regular harmonic mappings is highly restricted.

The control of the grid clustering can be obtained in a similar way as in the case of two-dimensional orthogonal coordinate. Applying equation (3.10) to the definition of the functions $F_{(i)}$ (4.2), and denoting with s,t and u the arc lengths along the ξ-, η- and ζ-coordinate lines respectively, it follows

$$F_{(1)} = \sqrt{\frac{t_n u_\zeta}{s_\xi}} \qquad (4.9)$$

Similar relations hold for $F_{(2)}$ and $F_{(3)}$.

In fig.4 the grid obtained in a region between two ellipsoids is shown. The grid on the boundary surfaces has been obtained by transfinite interpolation, and kept fixed during the iterative procedure. The interior grid is smooth and with a limited departure from orthogonality. An improvement of the present technique could be obtained by leaving the boundary points to move on the boundary surface in order to satisfy the orthogonality requirements.

5. Conclusions

A grid generation technique for curved surfaces and three-dimensional regions is presented. In two dimensions it has been proved that the mapping represent unfolded orthogonal coordinates. In three dimensions the resulting mapping is harmonic. Clustering control including an adaptive curve-by curve method is presented.

References

1. Thompson, J.F., Steger, J.L. and Yoshihara, H. Three-dimensional grid generation for complex configurations - Recent progress, AGARDograph n.309, March 1988.

2. Letho, O. and Virtanen, K.I. Quasiconformal mappings in the plane. Springer-Verlag, NY., 1973.

3. Willmore, T.J. Total curvature in Riemannian geometry, Ellis Horwood, Chichester, 1982.

4. Eells, J. and Sampson, J.M. Harmonic mappings of Riemannian manifolds, Amer. J. Math., 86, 109-160, 1964.

5. Arina, R. Adaptive Orthogonal Surface Coordinates, Numerical Grid Generation in Comput. Fluid Mechanics, Sengupta and al. eds., proceedings of the 2nd International Conference on Numerical Grid Generation in CFD, Pineridge, 1986.

6. Arina, R. Adaptive Orthogonal Curvilinear Coordinates, Conference on Numerical Methods for Fluid Dynamics, Oxford, Morton K. and Baines M.J. eds., Oxford Univ. Press, 1988.

7. Cohn, H. Conformal Mappings on Riemann surfaces, Dover, New York, 1980.

8. Thompson,J.F.,Warsi,Z.U.A. and Mastin,C.W. Numerical grid generation, foundations and applications, North-Holland, NY, 1985.

9. Eiseman, P.R. Adaptive grid generation, Comput.Meth.in Appl.Mech.and Eng., Vol.64-1/3, pp.321-376, 1987.

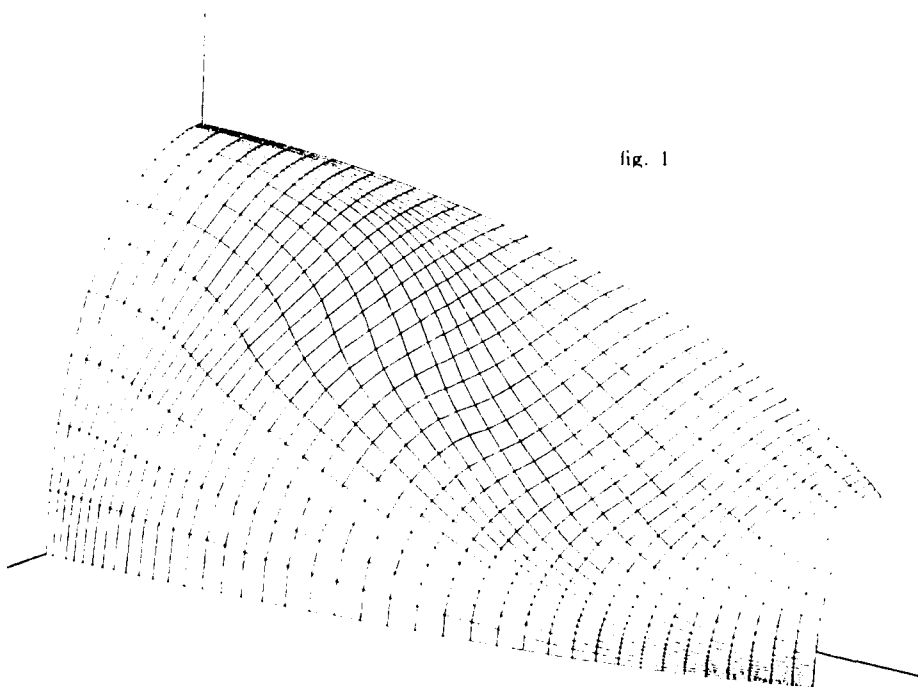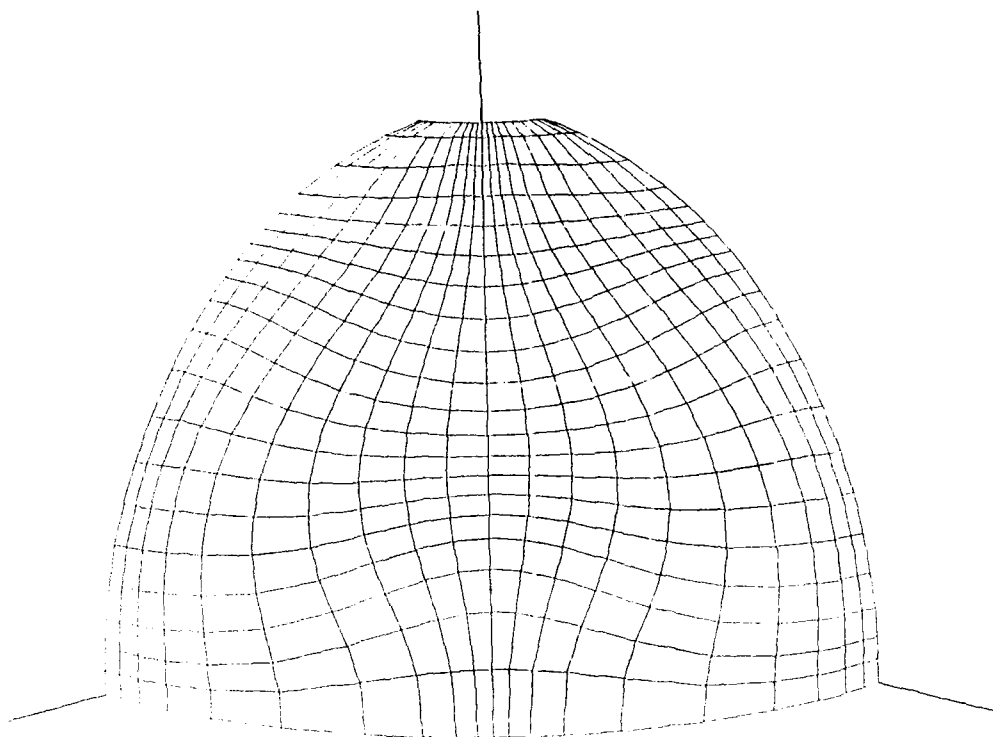10. Brezis, H. Analyse fonctionnelle - Theorie et applications, Masson, Paris, 1983.
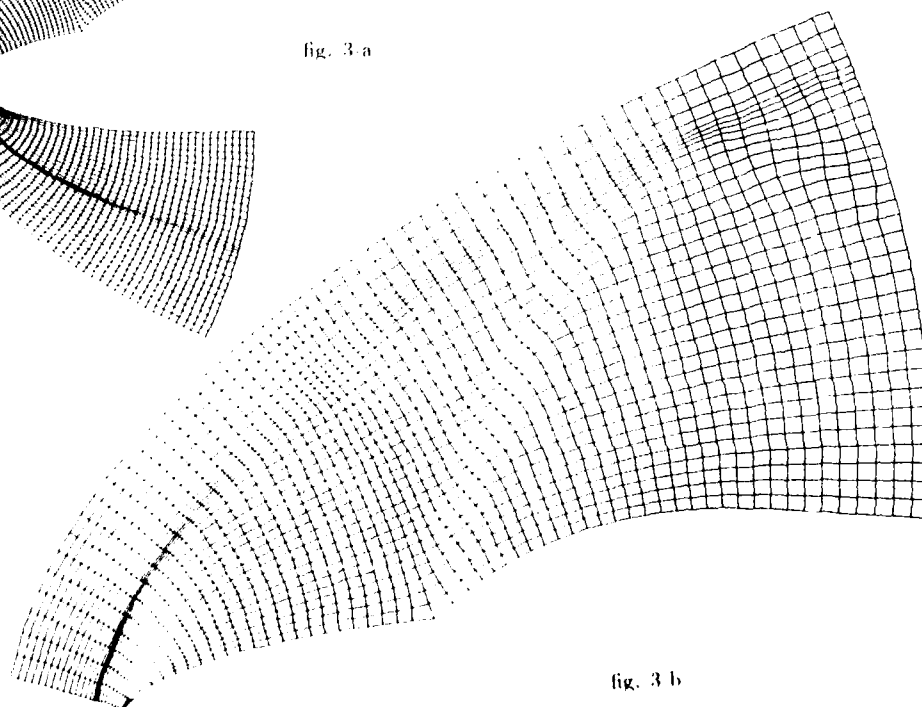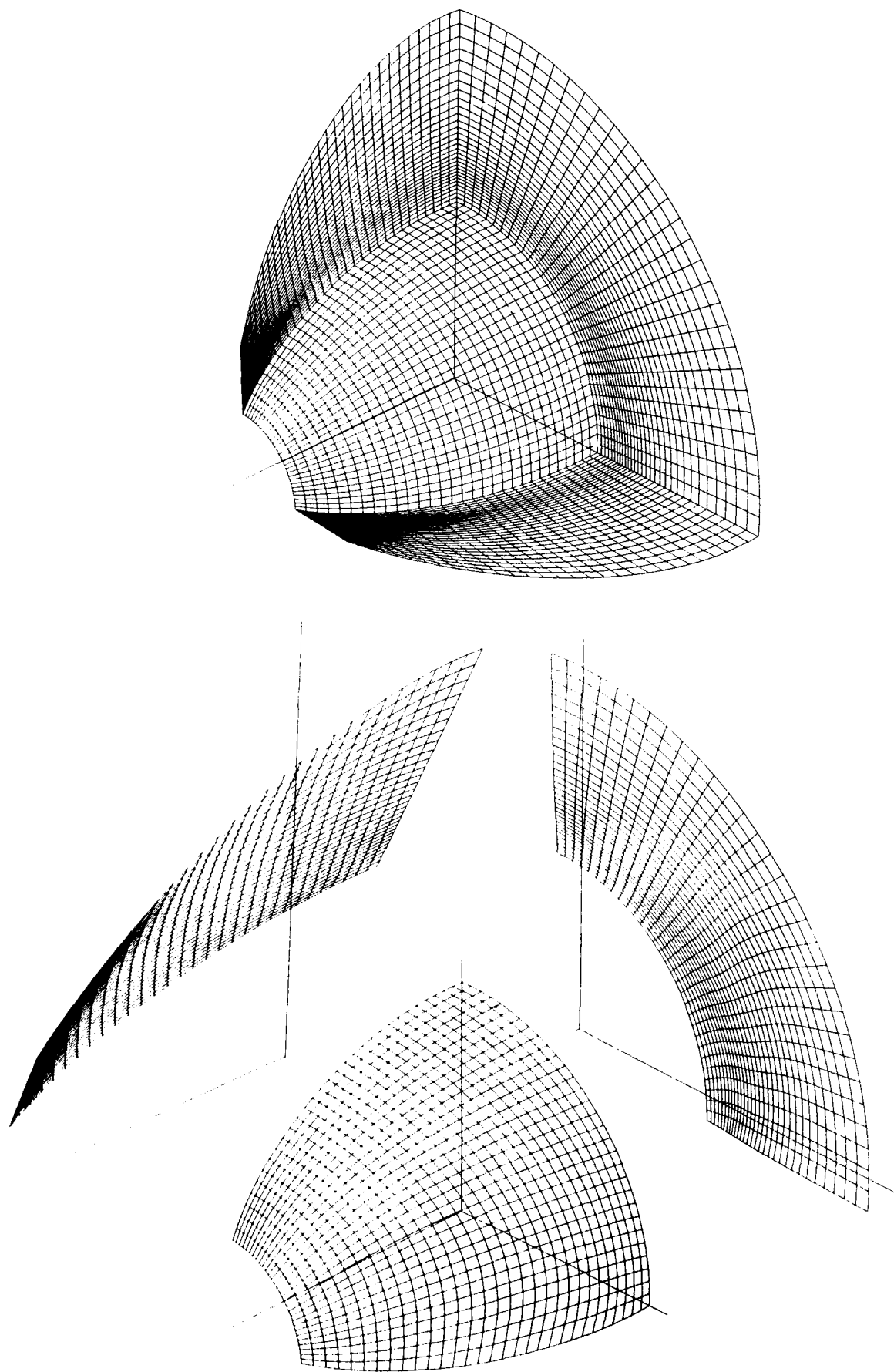
fig. 1

fig. 2

fig. 3 a

fig. 3 b

fig. 1

FEATURE-ASSOCIATED MESH EMBEDDING
FOR COMPLEX CONFIGURATIONS

by

C.M.Albone
Royal Aerospace Establishment
Farnborough
United Kingdom

Gaynor Joyce
Aeronautics Department
The City University
London EC1
United Kingdom

SUMMARY

The mesh-generation scheme described in this paper nas been designed to cope with complex geometric and flow features by employing many regular three-dimensional overlying meshes. Features are classified according to the number of geometric constraints to which they are subject, and each feature has its own purpose-built mesh. Four different mesh topologies are required to deal with all possible geometric and flow features. Progress to date is described and meshes for simple three-dimensional configurations are shown together with results of some Euler flow calculations.

1 INTRODUCTION

The geometric complexity of aircraft configurations presents a major challenge to those involved in generating field meshes upon which the governing flow equations may be discretized and solved numerically. Accurate representation of all parts of the solid surfaces of the aircraft and of the boundary conditions that apply there is crucial to obtaining numerical solutions in which the aircraft designer can have confidence. There is now broad agreement amongst researchers that alignment of the mesh to all solid surfaces is necessary in order to achieve this accuracy. For complex configurations, this requirement alone places numerous constraints upon the mesh and these frequently result in meshes that are of poor quality in some parts of the field. Further constraints arise because the quality of the numerical solution is also strongly dependent upon how well the mesh represents features other than solid surfaces. These are the features of high-speed viscous flow which may have a directional bias, such as shock waves and shear layers, or they may simply be regions of high, but not directionally dominated, flow gradient. As a result of these difficulties, methods for generating meshes have over recent years grown considerably in complexity and sophistication[1]. Usually, the task of generating a mesh of even modest quality for a given complex configuration requires considerable user expertise, much manual intervention and several man-months of work.

Most mesh-generation methods fall into one of three categories: multiblock, unstructured or overlapping, although the last of these may be called overset or overlying. The strategy behind the method[2] described in this paper was inspired by the nature of geometric and flow features that occur in high-speed aerodynamics. The method, known as Feature-Associated Mesh Embedding (FAME), uses many meshes and is of overlapping-mesh type, yet it is more appropriately described as being an overlying-mesh method. It contains many regions of embedded mesh and so is at least in part unstructured, yet it bears little resemblance to so-called unstructured meshes. Its main mesh is synthesized from many sub-meshes called blocks, yet it is not multiblock method. This serves to indicate how difficult it can sometimes be to categorise a method. Before proceeding to a more detailed description of the present method and of the results in sections 3 and 4 respectively, we will attempt to clarify some of the issues regarding the nature of various mesh-generation schemes emphasising where possible the similarities between them. This will comprise section 2.

2 A CONCILIATORY VIEW OF MESH-GENERATION METHODS

In recent years, there has been an unfortunate trend for views to become polarised regarding the merits of the three broad approaches to mesh generation. It may therefore be helpful to spend a while identifying some of the similarities between the approaches. This task is fraught with difficulties; not least of all those of semantics. For example, let us firstly consider how many meshes are employed in the various approaches. Few would dispute that overlapping-mesh methods use several meshes. The common view of multiblock however is that it consists of one mesh decomposed into several blocks. The use of these words tends to give the impression that the two approaches are quite distinct. But are they? If, in one block of a multiblock mesh, we refine (or embed) the mesh in one or more coordinate directions by (say) a factor of two, so that some mesh lines terminate at block faces, do we still have one mesh and several blocks, or is it now more appropriate to consider the finer block to be a separate mesh that overlaps (or perhaps more appropriately overlies) the original mesh? Certainly the treatment necessary at such a block face would have much in common with that used in overlapping mesh schemes. The second problem area concerns use of the words regular and irregular to describe points in a mesh. Again, few would dispute that unstructured meshes consist entirely of irregular points. For a multiblock mesh it is also accepted that so-called singular points at certain block edges and corners are irregular. But what about those points on a face between two blocks having different orientations; should they not be irregular as well? And if they are irregular, why is a multiblock mesh commonly referred to as a structured mesh (a term usually reserved for a mesh consisting entirely of regular points)?

We begin to try and unravel some of this confusion by defining regularity for each point of a three-dimensional mesh at which the flow equations (possibly in conjunction with a boundary condition) are to be discretized. The condition for regularity concerns the addressability of neighbouring mesh points, where the number of these is a function of the size of the stencil (or computational molecule) associated with the discretization scheme. For a point indexed (I,J,K) to be regular, all the points in physical space that are included in the stencil must be addressable as (i,j,k) where i, j and k differ from I, J and K , respectively, by zero or one (or possibly by 2 for some larger stencils). In other words, all points in the stencil must be adjacent to the point in question in computing space as well as in physical space. This condition necessarily excludes points where more than six mesh lines meet. Mesh points that fail to satisfy this condition are defined as being irregular.

Let us now turn our attention to meshes rather than points and deal with the terms structured and unstructured. All meshes possess structure, since without it no spatial relationship would exist between points and so no discretization scheme could be implemented upon them. For so-called unstructured meshes, the structure is defined on a local basis rather than the simpler. global basis used for structured meshes. Here we avoid using both these terms for defining types of meshes since they add little to our understanding of the properties of a mesh. Instead, we define a mesh to be _regular_ if all points at which the flow algorithm is to be discretized are regular. An irregular mesh is defined as one having at least one irregular point. Whilst these definitions follow logically from those for points, they may conflict with some commonly-held views. As an example, an O-mesh or a C-mesh around an aerofoil is usually considered to be regular. However, mesh points on the 'cut line' downstream of the trailing edge are interior in physical space, but are boundary points in computing space at which the flow equations are discretized. If the mesh is extended across the cut line from both sides through use of _haloes_, and the appropriate values of flow variables are set at the halo points, then the mesh is _regular_ by the above definition, since all points at which the flow algorithm is discretized are regular. If no halo is used, then points on the cut line fail to satisfy the condition of regularity and a _pointer scheme_ is usually adopted to address neighbouring points. In this case the mesh is _irregular_. These two alternative ways of viewing such meshes will arise later on in discussion of multiblock meshes. A second example of where our definitions may conflict with commonly-held views is illustrated by the two-dimensional mesh, sketched in Fig 1, which might be perceived as being irregular. If at the boundary points (marked by circles) values of flow variables are given, so that the flow algorithm is discretized only at interior points, then, according to our definition, the mesh will be _regular_ (for a computational molecule no larger than a nine-point symmetric stencil). In this case the irregular shape of the region covered by the mesh does not automatically lead to the mesh being irregular.

It is widely accepted that flow algorithms are more easily implemented on regular meshes than irregular ones due to their global addressing property. This simplicity is exploited by advocates of overlapping meshes where each mesh is usually regular or can be made regular through the use of haloes, but they pay the penalty of having to construct possibly complicated and potentially inaccurate interpolation procedures in order to transfer flow data between meshes. With the new definition to hand, we may therefore state that overlapping mesh methods use _several overlapping regular meshes_.

We now show how multiblock mesh methods can be classified in two completely different ways according to whether the data structure at block boundaries is of halo type or pointer type. When using the halo approach, the data array for discretizing the flow equations at interior points of a block is expanded so that the discretization scheme may be applied at block-boundary points as well. Where, as is usual, each point interior to a block is regular, this procedure renders block-boundary points regular, but now the 'expanded block' is bigger and indeed it _overlaps_ neighbouring blocks. Through using the halo procedure, multiblock is therefore conceptually the same as the overlapping mesh approach and so can be defined as consisting of _several overlapping regular meshes_. Of course there are differences in detail between the two. If at block boundaries, mesh lines are continuous (this necessarily means no embedding) and changes in the slope of mesh lines are small, then points may be taken as coincident in the overlap region so avoiding the need for interpolation. The second approach to data structure at the block boundaries of a multiblock scheme is to use a pointer system. In using this, the block is not expanded and so the irregularity of points on the block boundary is accepted and the pointer system is used to address points in neighbouring blocks. In this case multiblock may be defined as consisting of _one irregular mesh_.

To complete the definition of the various approaches to mesh generation, we may state that so-called unstructured-mesh methods consist of _one irregular mesh_. The observation that this is identical to the definition of multiblock methods when a pointer system is adopted may cause some consternation, since the two approaches are usually viewed as quite distinct. The difference in fact is one of degree (albeit a significant degree) in that _all_ points in an unstructured-method are irregular whereas only a two-dimensional subset of mesh points is irregular in multiblock. The unstructured-mesh approach could therefore be viewed as a multiblock scheme (using pointers) with only one cell in each block, so removing all points interior to a block, which in turn results in there being no regular points.

In trying to break down some of the conceptual barriers between the three approaches, we have used the two interpretations of multiblock to link it to overlapping and unstructured schemes. Why then should multiblock alone have two interpretations? The main reason is that the two possible data structures at block boundaries (which led to the two interpretations) are equally viable alternatives for a multiblock scheme where the

number of blocks is small compared with the number of mesh points. If the number of (regular) interior points per block became very small (resulting in many more blocks), the pointer scheme would be preferable to the use of haloes. This explains why for the unstructured mesh approach (with no interior regular points) a pointer scheme is employed and in consequence why it is considered as a single irregular mesh. It would, of course, be possible in principle to adopt a halo scheme for the unstructured-mesh approach thereby considering it to be a set of overlapping meshes. However this would be most inefficient since the number of meshes would equal the number of points!

Finally then, what are the prospects for an alternative interpretation of overlapping-mesh methods. Here of course mesh overlap is necessary in order to cover the whole of space because of the discontinuity in position and orientation between meshes. We do not have to construct haloes since they already exist (implicitly) via the overlap, and so our interpretation given earlier (several regular meshes) is natural. In order to apply the alternative interpretation, we would have to generate a set of irregular points (to replace the overlap region) which link the neighbouring regular meshes, thereby creating one irregular mesh. This (irregular) interpretation of overlapping meshes appears quite plausible. (It should be noted however that it gives rise to a mesh that is actually different from that obtained with the conventional interpretation. This contrasts with the position for multiblock where the two interpretations are purely conceptual.) The alternative (irregular) approach for overlapping meshes has indeed been adopted, but only (as far as the authors are aware) by treating all field points as irregular[3], so ignoring the regularity of the majority of the points.

It appears that the argument has gone full circle. The conclusion of our conciliatory view of the three types of mesh-generation scheme appears to be that they are indeed similar if one chooses to interpret them as such. The interpretations more commonly adopted seem to accentuate differences.

This brief review has avoided a discussion of the merits of the various approaches to mesh-generation. This omission is deliberate, for such a discussion would have to be based upon many detailed aspects of the actual methods under development by CFD research groups. Some of the issues that would have to be addressed in such a discussion are given below:

(a)   What are the factors affecting mesh quality - cell skewness, mesh expansion ratio, cell aspect ratio, mesh smoothness, appropriate mesh density for flow gradients - are there more?

(b)   In multiblock methods with a high level of continuity at block boundaries, how do the constraints on the mesh impact on mesh quality; is such a level of continuity necessary?

(c)   How good is the control of cell 'aspect ratio' in unstructured-mesh methods; how suitable are they for dealing with regions of strong directionality?

(d)   In overlapping-mesh methods, how much accuracy is lost through interpolation; how complex is the program bookkeeping?

(e)   How easily can each method be extended to include solution adaption?

(f)   How much manual intervention is needed to get the required mesh quality; how expert does the user have to be?

It is beyond the scope of this paper to answer these questions. However some of the issues raised in this section should assist the reader in understanding the philosophy and strategy behind the FAME method reported here.

3   FEATURE-ASSOCIATED MESH EMBEDDING

The long-term aim of this work[2] is to unify the treatment of geometric and flow features through a flexible approach to mesh-generation, although in its current state of development only geometric features are treated. The corner-stone of the method is a classification of features according to the number of directional constraints to which they are subject. The strategy for generating high-quality meshes is then built upon four key ideas. Firstly, in order to minimise constraints upon meshes, many meshes are employed, with one mesh associated with each feature. Secondly, the spatial extent of each feature-associated mesh is limited to the neighbourhood of the feature itself. Thirdly, a main (or background) mesh underlies all other meshes and covers the whole field of interest; it is not aligned to any of the geometric surfaces. Finally, comparability of mesh densities where overlap occurs (namely, where flow data is interpolated between meshes) is achieved primarily through the use of multi-level embedding on the main mesh.

We consider four types of feature; these are classified according to the number, N, of directional constraints associated with the feature and are denoted as being of type N, N = 0, 1, 2, 3. Features of type 1 are associated with surfaces, those of type 2 with lines and those of type 3 with points. Type-0 features have no directional constraints associated with them. It is asserted that each feature merits its own mesh having a topology that is appropriate to the feature. We refer to meshes that are associated with features of type N as being type-N meshes. Each of these types is now discussed in a little more detail starting with type 1.

Features such as body geometry, shock waves and vortex sheets in inviscid flow are associated with surfaces in space; others, such as shear layers in high Reynolds-number flow are associated with thin regions adjacent to surfaces (which may be solid surfaces or surfaces within the fluid). All such features are characterised by the direction of the normal to the associated surface and accordingly these are denoted as being of type 1. Each type-1 feature merits its own mesh which should be orientated according to the orientation of the associated surface. Where, for example, the surface is a solid boundary, this simplifies the task of satisfying the solid-surface boundary conditions. Each type-1 mesh could thereby be constructed with two coordinate directions in the surface and one normal to it. The distribution of mesh points within each mesh of type-1 is not a major consideration here; it could be prescribed by any of the methods currently used for conventional meshes or it could be subject to solution adaption. The novel aspect of the present approach concerns the spatial extent of each type-1 mesh in the direction normal to the associated surface. The only requirement is that the feature is entirely covered by the mesh. In inviscid flows, all type-1 features are simply surfaces and so unless there are good reasons for doing otherwise, meshes of type-1 need only extend as many intervals away from the surface as is necessary to define the flow-algorithm stencil, as sketched in Fig 2 for an extent of two mesh intervals.

For features such as thin shear layers, the need to resolve very high flow gradients normal to the associated surface will require us to use a mesh with many small intervals normal to the surface. Nevertheless, the extent in physical space again need only cover the feature, see Fig 3.

We define features of type-2 to be those associated with lines, examples of which may be the line of intersection of two surfaces or a line across which the surface normal is discontinuous. (Whilst we may sometimes view these as distinct in aerodynamic terms, geometrically they are one and the same.) Whereas features of type-1 are associated with a single surface and characterised by the surface normal, features of type-2 are associated with two surfaces and are characterised by the normals to each surface at their line of intersection. We refer to type-2 features as edge lines (or edges) and consider this term to embrace all lines of intersection of two surfaces irrespective of their sense (either 'convex' like a wing trailing edge or 'concave' like a typical wing-fuselage junction line). The two surfaces concerned may be associated with any geometric or flow features of type-1. We assert here that each feature of type-2 merits its own feature-associated mesh. Such meshes are necessary because each type-1 mesh is designed to cope with only a single surface (one directional constraint). Therefore, close to an edge line where two surfaces intersect (two directional constraints), neither of the type-1 meshes associated with each surface will be suitable. Type-2 meshes, however, are designed specifically for edge lines. In conventional mesh-generation schemes, edges are treated in a variety of different ways according to the sign and magnitude of the discontinuity in the normal. We may view these treatments two-dimensionally by considering planes normal to edge lines and by characterising the edge by an angle θ, $-\pi < \theta < \pi$, which measures the discontinuity in the surface normal, with θ taking positive values for edges of convex type and negative for those of concave type. If we denote by ϵ a positive angle that is small compared with π, we may identify three different edge-line topologies in common use as shown in Fig 4 for $|\theta| < \epsilon$, $|\theta \pm \pi/2| < \epsilon$ and $|\theta \pm \pi| < \epsilon$ respectively. Whilst these topologies are perfectly adequate for the geometries concerned, their use presents difficulties where θ varies significantly along the edge as for example in the case of a wing-fuselage junction where topologies (a) and (b) may occur along the junction line. Here we propose that each type-2 feature irrespective of the sign and magnitude of θ should have its own local mesh of O-H topology which takes the form of a cylindrical-polar-type mesh with its axis running along the edge line. As with meshes of type-1, type-2 meshes need extend only a limited distance into the field. Proposed meshes of types 1 and 2 are sketched for a wing trailing edge (in two dimensions) in Fig 5.

We define features of type-3 to be those associated with points, an example of which is the point of intersection of three surfaces, such as the wing upper surface, the wing lower surface and the fuselage surface where the wing has a sharp trailing edge. Type-3 features may also exist at isolated points on an otherwise smooth surface (at for example the nose of a pointed body of revolution). As with types 1 and 2, type-3 features merit their own feature-associated mesh. The topology of type-3 meshes, being appropriate to a point, should have O-O structure and so take the form of a spherical-polar-type mesh with the origin of the mesh at the point concerned. The radial extent of each type-3 mesh can (as with the other meshes) be limited to the extent of the feature, which in the case of inviscid flow merely needs to be sufficient to allow the definition of the flow algorithm stencil.

Regions of the flow lying away from solid surfaces, shear layers and shock waves are comprised of features of type 0. Such regions could alternatively be termed 'feature-less' and indeed, since we classify features according to the number of directional constraints associated with them, this term is not inappropriate. However the numerical treatment employed for type-0 features is crucial to the accuracy of the flow calculation because these features may include regions of high (but not directionally dominated) flow gradient. Since type-0 features do not have directional constraints associated with them, type-0 meshes can simply be of rectangular-Cartesian type. Further, the absence of any strong directionality in flow gradients means that all mesh cells may be taken as cubes. A patch of mesh consisting of identically-sized cubes constitutes a type-0 mesh. Variation of the size of these cells across the field is achieved through mesh embedding by a factor of two in all coordinate directions so ensuring that all cells remain as cubes, as sketched in Fig 6 for two dimensions. A type-0 mesh of given fineness will, therefore, be embedded within a coarser type-0 mesh. The set of all type-0 meshes covers the whole

field of interest, including regions interior to solid components, and is considered to constitute the main (or background) mesh. This main mesh is synthesized starting with the coarsest type-O mesh and proceeding to finer type-O meshes through an automatic embedding algorithm[2] that may be driven by a variety of mechanisms. The only mechanism employed so far in the present work is that arising from the requirement that, where meshes overlap, their densities should be comparable. Thus for a smooth body, for which we have just one type-1 mesh, the embedding algorithm generates type-O meshes of successively higher density in the region of overlap with the type-1 mesh until the densities are comparable to within a factor of two as shown in Fig 7. (The embedding algorithm could alternatively be driven by the magnitude of local flow gradients, thus providing solution adaption, and it is intended to investigate this mechanism in the next phase of the work.) In order to rationalize the main-mesh data structure, each type-O mesh is itself synthesized from a set of computationally-identical sub-mesh units called blocks. Each block is a cube consisting of n^3 mesh cells, where n is usually taken to be 4, 6 or 8.

In section 2 it was noted that FAME is best viewed as being an overlying mesh method rather than an overlapping one. The reason for this is central to the philosophy of the method. The coarsest type-O mesh covers the whole field and, as far as data structure is concerned, it does not have a 'hole' cut in it where a finer type-O mesh is embedded within it. It underlies all finer type-O meshes. Thus each type-O mesh may be considered to overlie all coarser type-O meshes. This data structure was considered to be relatively simple to construct and it is immediately amenable to the implementation of a main-mesh multigrid scheme. A hierarchy of type-O meshes therefore exists with the coarsest mesh at the bottom and the finest at the top. Relative position within this hierarchy determines which mesh takes precedence and in consequence the order in which computed flow data on one mesh replaces those from another. Meshes of type-1, having been constructed as appropriate to the regions occupied by type-1 features, overlie the main mesh; they are located above type-O meshes in the hierarchy and flow data computed on them replaces those from the main mesh. In turn, meshes of type-2 overlie those of type-1 (and in consequence all those of type-O), and are located further up the hierarchy than those of type-1. Flow data computed on type-2 meshes replaces those computed on all meshes lower down the hierarchy. Finally, type-3 meshes head the hierarchy.

It should be noted that, in the context of this method, the statements 'mesh A overlies mesh B' and 'mesh A is embedded within mesh B'are considered to be equivalent. The manner in which flow data is transferred (using linear interpolation) from one mesh to another is the same whether the transfer is between two meshes of type-O or between those of types 1, 2 or 3 and those of type O; it differs only in detail. For the former data transfer, the interpolation is simple because all type-O meshes have the same orientation, whereas for the latter general three-dimensional interpolation is necessary.

Finally, we can now classify FAME according to the discussion of section 2. All meshes of types 1, 2 and 3 are regular overlying meshes by construction. Whilst the main mesh with its multilevel embedding appears irregular, the data structure that we adopt ensures that this is not so. Our smallest mesh unit, from which the type-O meshes are synthesized, is a cube-shaped block. There are a large number of these and they are computationally identical irrespective of their size in physical space. We treat each block as a regular mesh by expanding its data array to form a halo. (The computational identity of these blocks enables us to take as the 'vector length' the (large) number of blocks rather than the (small) number of points in any coordinate direction.) Accordingly, the main mesh is considered to be a set of regular (and indeed uniform) overlying meshes.

4   DEVELOPMENT STAGES OF FAME AND CURRENT RESULTS

The authors' search for a new mesh generation strategy for complex configurations and flows began in 1985. By the following year, a pilot method[4] in two dimensions had been developed which employed a main mesh, with a rudimentary form of embedding, together with a surface-orientated overlying mesh constructed by dropping normals to the surface from certain main-mesh points. This study served to indicate how not to approach the problem, and as a result the present strategy began to take shape towards the end of 1986. A two-dimensional code, consisting of mesh generator and Euler flow solver, was developed and tested by early 1987. The main and surface-aligned meshes generated by this code for a three-element aerofoil are shown in Fig 8. A close-up of the region covered by the slat and leading edge of the main aerofoil is shown in Fig 9. Meshes for the case of two NACA 0012 aerofoils of different sizes, one above the other, are shown for vertical separations of 0.25 and 0.5 (upper aerofoil chord = 1.0) in Figs 10 and 11 respectively. This configuration has been used as a test case in two dimensions for mesh generation aspects of store release. All the meshes shown in Figs 7 to 11 were generated automatically from given point distributions on the component surfaces and from three simple control parameters; no user expertise was needed.

Surfaces in three dimensions become lines in two dimensions, edge lines become points, and so only features of types 1 and 2 exist in two dimensions. In two respects, the treatment of surface-aligned meshes in the two-dimensional version of FAME lacks the full generality intended for the three-dimensional work. Firstly, a surface-aligned C-mesh was used for each aerofoil so that the aerofoil surface (type-1) and the trailing edge (type-2) were accommodated by a single mesh. This runs counter to the general strategy given in section 3 since one mesh (the C-mesh) deals with two features. Secondly, the extent of each C-mesh normal to the aerofoil surface was limited to just one mesh interval. Whilst this had little impact upon the mesh-generation scheme, it

limited the flow algorithm stencil to just two points normal to the surface, so preventing full second-order accuracy from being achieved. Despite the use of a C-mesh and the partial second-order accuracy, results from the two-dimensional code showed that the strategy and methodology of FAME was very promising in that high quality meshes for multi-element aerofoils could be generated automatically.

Work on the development of FAME in three dimensions began later in 1987, and is still continuing. For this development, the principle of 'one mesh for one feature' is being strictly adhered to. This means, however, that type-2 meshes are required in order to treat a configuration as simple as an isolated wing with a sharp trailing edge. At the present time, the mesh-generation code and an Euler flow solver have been developed and tested only for meshes of types 0 and 1, so that we are currently restricted to smooth, non-intersecting bodies, but there may be any number of these. The incorporation of type-2 meshes into the mesh-generation and flow codes is currently in progress. The flexibility afforded by FAME has enabled us to treat the far-field boundary of the computing region as a type-1 feature and accordingly we give it its own type-1 mesh. This has the advantage that the position and shape of the far-field boundary can, where necessary, be changed whilst leaving the near-to-mid-field meshes unaltered. Further, various forms of far-field boundary-condition treatments can be investigated on a purpose-built, boundary-aligned mesh quite independently of the rest of the solution algorithm. Type-1 meshes are constructed without the limitation on field penetration present in two dimensions in that they may extend more than one mesh interval away from the body surface. For flow calculations using Euler solvers, the number of intervals normal to the body surface is usually taken to be two, so permitting use of a stencil with three points in the normal direction and in consequence allowing for full second-order accuracy. Navier-Stokes flow solvers can be implemented on type-1 meshes since these meshes can now extend many intervals normal to the surface and may be constructed with very high aspect-ratio cells close to the surface.

The Euler method employed on all meshes uses a first-order upwind finite-difference algorithm[5] which is made second-order accurate through use of a deferred-correction scheme[4]. It is ideally suited to use within the complex embedded structure of FAME since it has a compact, seven-point symmetric stencil in three dimensions. This Euler algorithm (but limited to partial second-order accuracy as mentioned earlier) was used as the flow-solution method in the two-dimensional version of FAME. Results of Euler flow calculations for the three-element aerofoil of Figs 8 and 9 have been reported elsewhere[2].

Development and testing in three dimensions of the flow algorithm and of the mesh-generation scheme have been taking place in parallel. Both methods contain many new aspects that have needed careful evaluation and checking. In consequence, we have concentrated on simple ellipsoidal shapes and indeed much of the testing has been carried out for flow past a sphere. Since, however, the main mesh in FAME is not aligned to the configuration surface, the full generality of embedded block structure and of three-dimensional interpolation is necessary for a body as simple as a sphere. Type-1 meshes (around the sphere and the far-field boundary) are of spherical-polar type. The finite-difference Euler algorithm[4,5] on these meshes has, however, been formulated to allow for general three-dimensional, non-orthogonal meshes. Visualization of the field meshes in three dimensions presents difficulties because of the embedded structure of the main mesh. However, inspection of field meshes for quality is far less necessary in FAME than in most other methods since the scope for generating meshes of poor quality (in respect of stretching and skewness) is very limited. Type-1 meshes are by construction nearly orthogonal with direct control over mesh stretching. Type-0 meshes are orthogonal and locally uniform with strict 'factor-of-two' subdivision between embedded levels. This is checked automatically by the embedding algorithm itself within the mesh-generation code.

Mesh visualization in certain two-dimensional sections through the field is, however, straightforward, if rather unspectacular for a configuration as simple as a sphere. An example is shown in Fig 12. Fig 13 summarises results of flow calculations for the sphere at a Mach number of 0.4 using various versions of the Euler algorithms[4,5]. These results are presented in the form of a plot of peak suction on the sphere surface against $1/n^2$, where there are $2n$ surface mesh intervals in each meridian plane; values of n of 24, 32, 48 and 64 are used. Curves are shown for results using the first-order upwind algorithm[5], the deferred-correction second-order accurate scheme[4] and the partially second-order accurate scheme that was used in the two-dimensional method. The deferred-correction scheme is clearly shown to have second-order behaviour with increase in mesh fineness since the curve through the four points is virtually a straight line.

Meshes have been generated for an idealized, two-component, wing-store configuration where each component is modelled as an ellipsoid. The wing has a root chord of unity, a span of 4.0 and a thickness of 0.2. Its surface is defined by the ellipsoid

$$\left(\frac{x}{0.5}\right)^2 + \left(\frac{y}{2.0}\right)^2 + \left(\frac{z}{0.1}\right)^2 = 1 \ .$$

The store has a circular cross section of radius 0.05 and a fineness ratio of 6:1. The surface of the store is defined by the ellipsoid

$$\left(\frac{x}{0.3}\right)^2 + \left(\frac{y - 1.0}{0.05}\right)^2 + \left(\frac{z + 0.4}{0.05}\right)^2 = 1 \ ,$$

so that it is located underneath the starboard wing at mid-semi span. Fig 14 shows type-0 and type-1 meshes in the plane $x = 0$. A close-up of the meshes in the same plane in the vicinity of the store is shown in Fig 15. The overl p between the two type-1 meshes presents no problems, since, at its outer edge, each mesh exchanges flow dat with the main mesh. Finally, Fig 16 shows the meshes in the 'wing plane', $z = 0$ for the starboard wing. Due to current memory and cpu limitations, flow calculations have not as yet been run for this configuration.

5   CONCLUDING REMARKS

Feature-associated mesh embedding should develop into a flexible method for generating high-quality meshes for complex aircraft configurations. Development and testing of the mesh-generation and flow codes is a slow and painstaking process, because maximum generality is being built in at all stages. However, when the full package is complete, configuration components may be modified, added or removed without the need for global changes of mesh topology. The approach may be particularly useful for configurations having components in relative motion, such as those associated with store release.

REFERENCES

1 Sengupta, S., Hauser, J., Eiseman, P.R. and Thompson, J.R., (editors), Numerical Grid Generation in Computational Fluid Mechanics '88, pub Pineridge Press Ltd, 1988.

2 Albone, C.M., "An approach to geometric and flow complexity using feature-associated mesh embedding (FAME): strategy and first results", in Numerical Methods for Fluid Dynamics III, edited by K.W. Morton and M.J. Baines, pub Oxford University Press, 1988.

3 Mavriplis, D.J., "Adaptive mesh generation for viscous flows using Delaunay triangulation", in Numerical Grid Generation in Computational Fluid Mechanics '88, pub Pineridge Press Ltd, 1988.

4 Albone, C.M., "A second-order accurate scheme for the Euler equations by deferred correction of a first-order upwind algorithm", RAE Technical Report 88061, 1988.

5 Chakravarthy, S.R., Anderson, D.A., and Salas, M.D., "The split-coefficient matrix method for hyperbolic systems of gas-dynamic equations", AIAA Paper 80-0268, 1980.
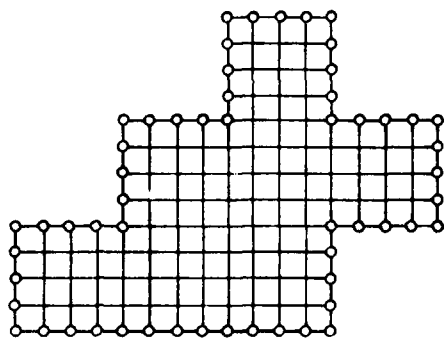
Fig 1 A regular mesh of irregular shape
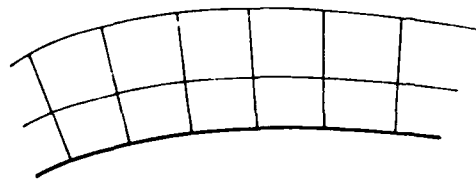
Fig 2 Type—1 surface mesh for inviscid flows
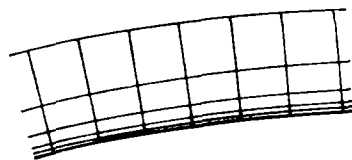
Fig 3 Type—1 surface mesh for viscous flows
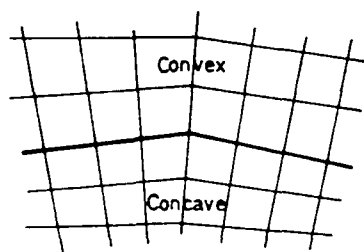
a) $|\theta| < \varepsilon$
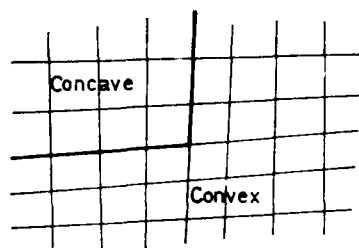
b) $|\theta \pm \pi/2| < \varepsilon$
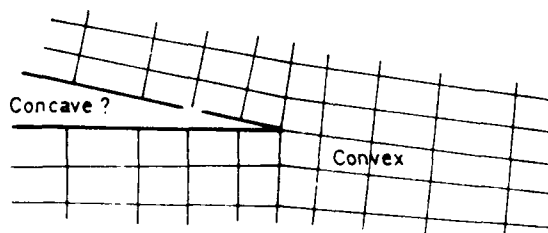
c) $|\theta \pm \pi| < \varepsilon$

Fig 4 Topologies commonly used in the treatment of edges
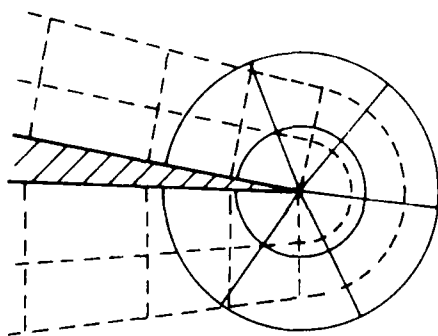
––– Type 1 mesh   ——— Type 2 mesh
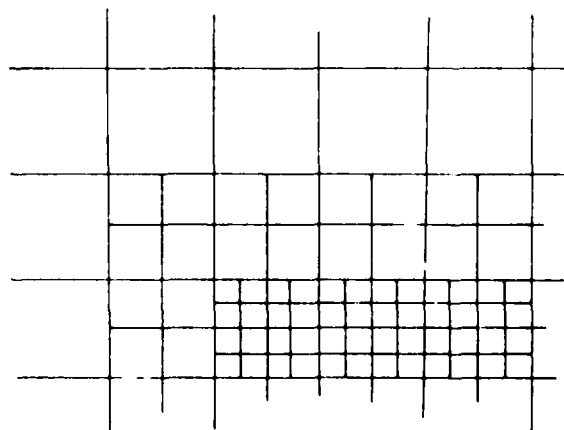
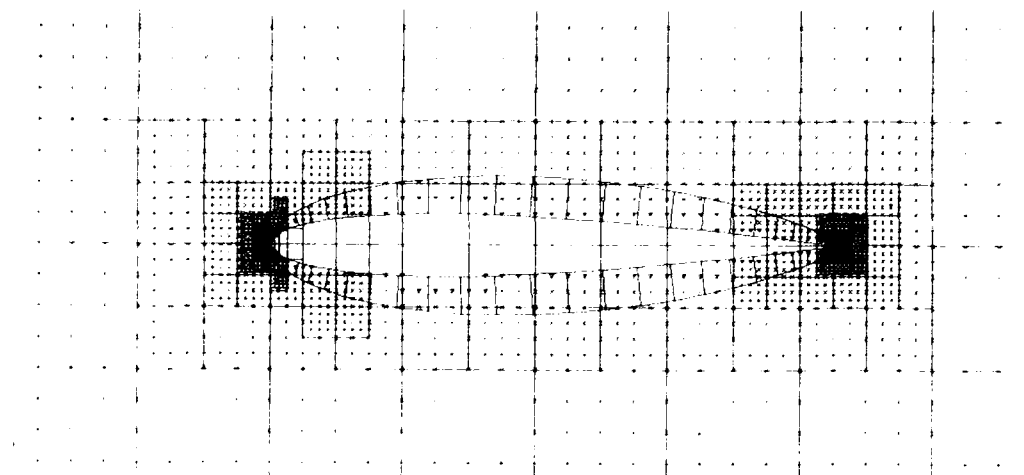Fig 5 Mesh topologies: wing trailing edge

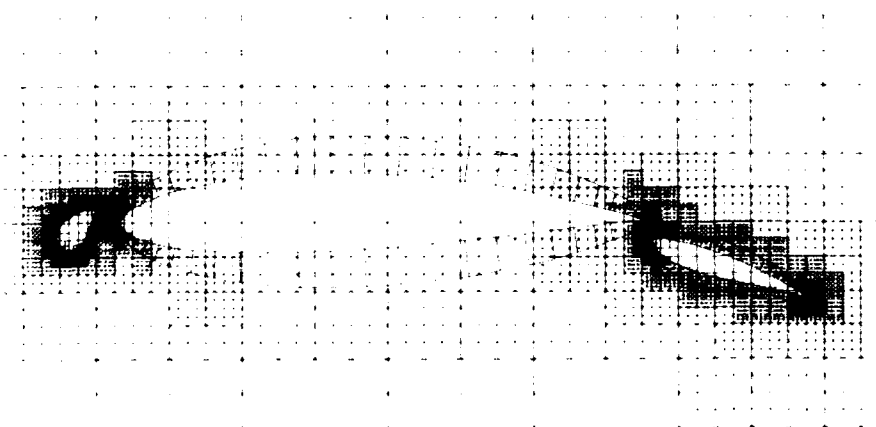Fig 6 Type—0 meshes

Fig 7   Meshes for a NACA 0012 aerofoil

Fig 8   Meshes for a three—element aerofoil
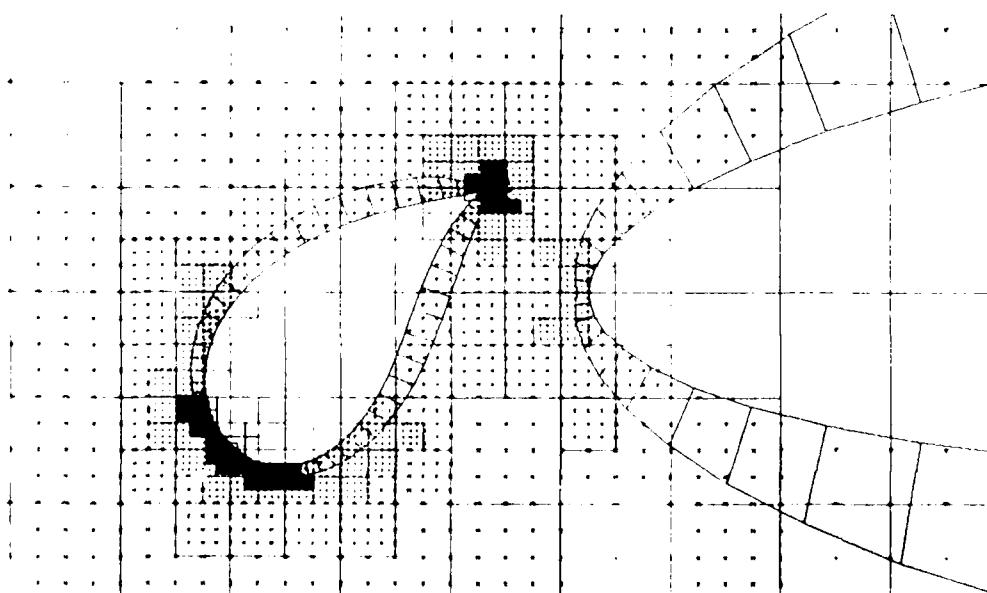
Fig 9   Blow up of Fig 8 in the region of the slat
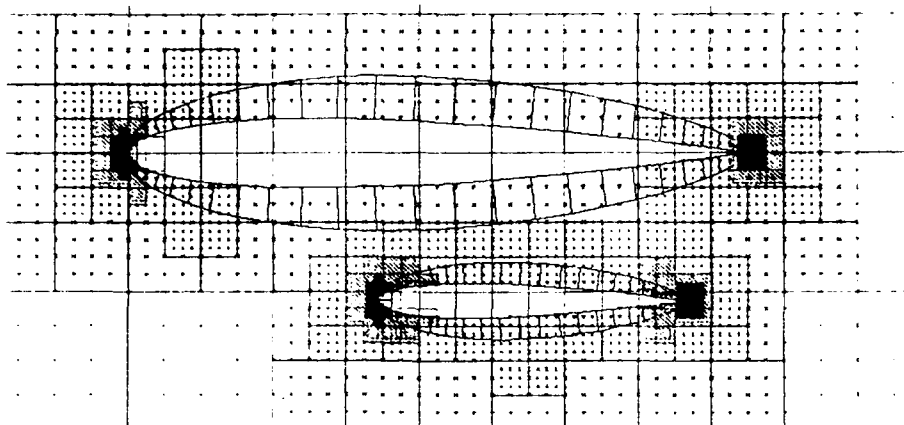
Fig 10   Meshes for two NACA 0012 aerofoils: separation = 0.25
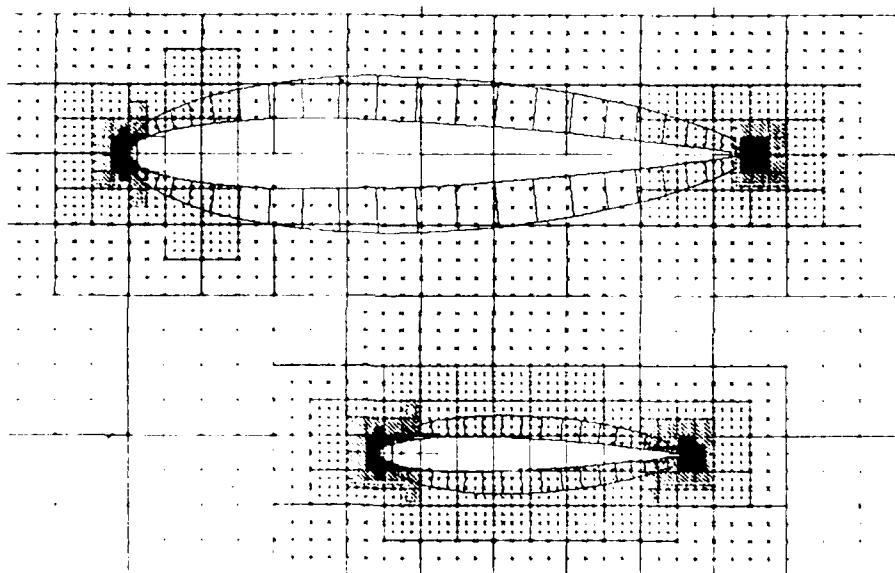
Fig 11   Meshes for two NACA 0012 aerofoils: separation = 0.5
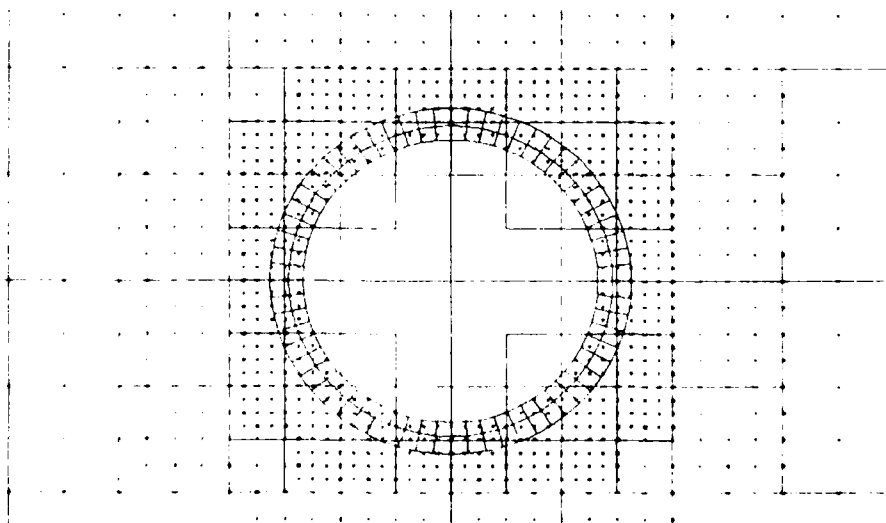
Fig 12   Meshes for a sphere: centreline plane
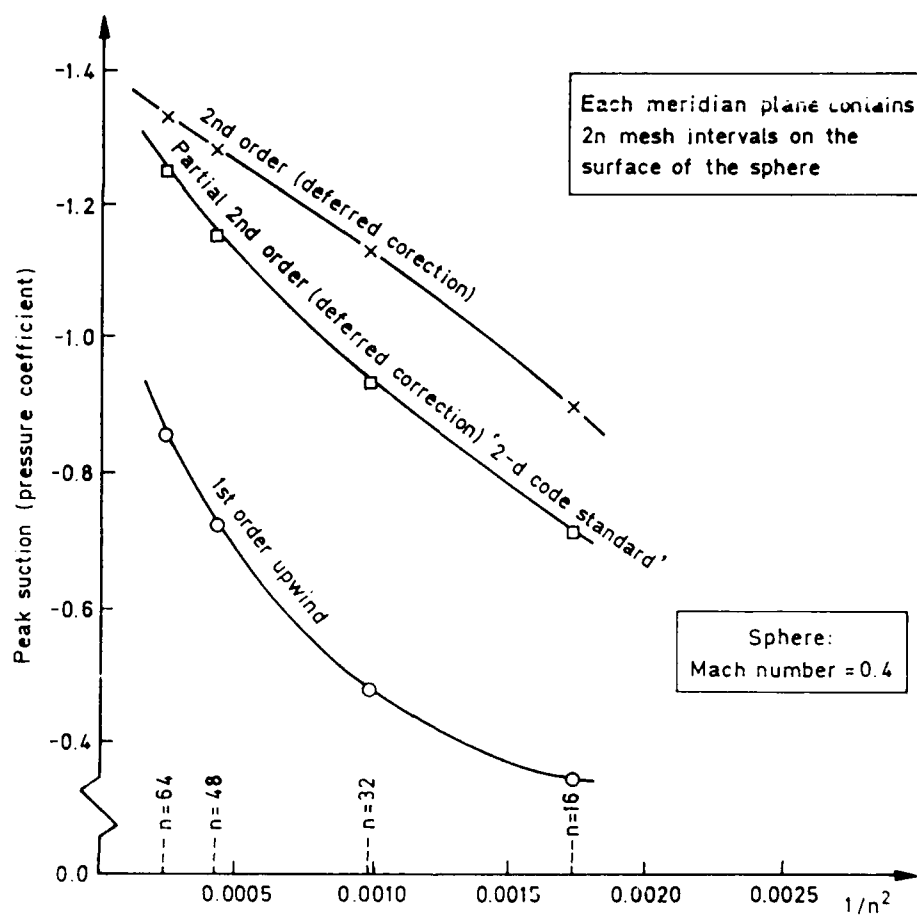
Fig 13 Convergence under mesh refinement for flow past a sphere
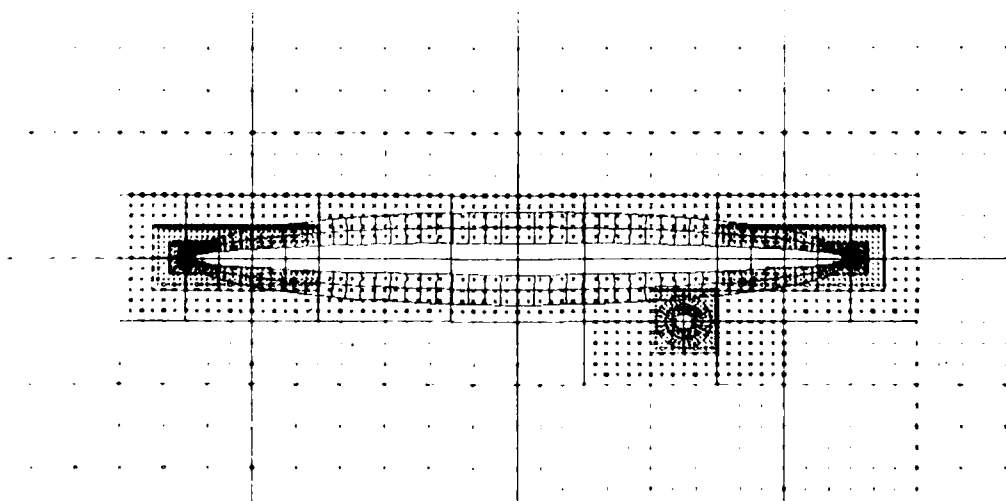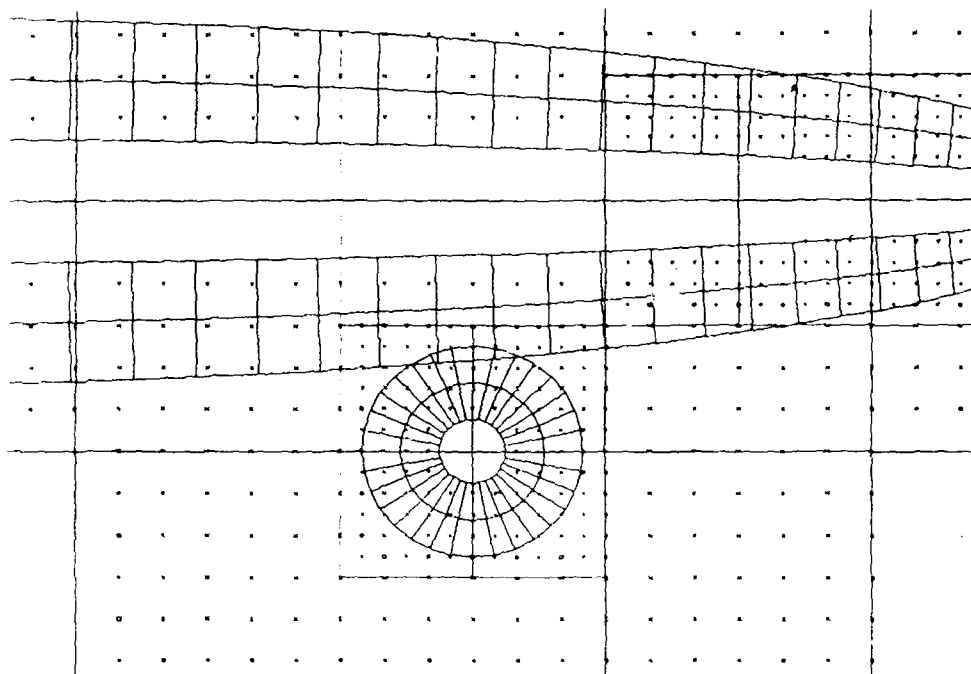

Fig 14 Meshes for an idealized wing—store: plane x = 0
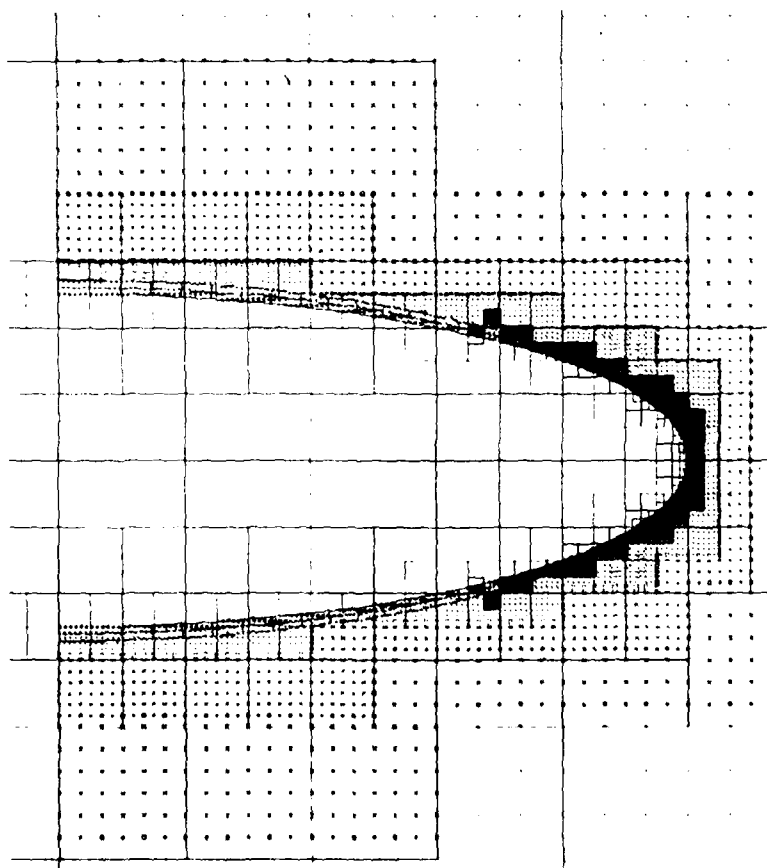
Fig 15 Blow up of Fig 14 in the region of the store

Fig 16 Meshes for an idealized wing—store: plane z = 0

ON THE WAY TO AN INTEGRATED MESH GENERATION SYSTEM
FOR INDUSTRIAL APPLICATIONS

by
W.Seibert, W.Fritz, S.Leicher
DORNIER-LUFTFAHRT GMBH
Computational Fluid Dynamics Group
Postfach 1420, D-7990 Friedrichshafen
Federal Republic of West-Germany

SUMMARY

The main features of some specialized batch modules, which have been developed recently to meet the requirements of a grid generation for complex configurations, are described in brief. One module is a combination of an algebraic grid generator for the determination of a surface grid and the far field boundary, and an hyperbolic grid generator for the sectionwise calculation of the corresponding orthogonal internal grid lines. Two other modules are concentrated on solution adaptive grids - either using algebraic redistribution proportional to the curvature of a typical flow filed describing function, or by solving elliptical partial differential equations resulting from the transformation of the Poisson equation from the physical space into the computational space. Adaption of the grid to pressure gradients and to the total pressure loss is done by replacing the source terms.
The first part of the paper however is the description of the graphic-interactive program-system INGRID, which already comprises several standard techniques to generate composite volume grids around arbitrary complex configurations, and which has the potential to become an integrated system to match the demands for a general productive mesh generation method.
As application examples several grids are shown, illustrating actual problems of external aircraft aerodynamics, ground-vehicle aerodynamics and of internal pipe flow.

1. INTRODUCTION

Within the last years the pretensions concerning the productivity of procedures for the numerical simulation of flow processes raised more and more. A high accuracy of the results is expected, the processed configurations should correspond to the real geometries with idealizations as small as possible. Especially the use of methods based on the Navier-Stokes's equations require a high resolution of details within the discretization of the computing model.

Besides those facts the operational area of the procedures is widened permanently. There is still the classical problem of the simulation of flows around aircrafts, but investigations of the flow behaviour at road vehicles and vessels are accomplished as well. Also the industrially very interesting field of internal flows, i.e. flow through machine parts, tube branches, pipe systems or power plants is accessed more and more by numerical investigations.

This development is supported by increasing computer capacities and performances. The average number of computing points for one investigation could be increased clearly - however the resulting computing times remained further acceptable. As a résumé it can be said, that the computational simulation of flow processes of all kinds has taken an essential role within industrial research and development.

However, there is one non-negligible subtask, which is still solved unsatisfactory up to now. It is the geometry processing and the grid generation, which becomes necessary before a computational analysis. Particularly by the increased abilities of the simulation programs, the weaknesses of the so-called preprocessing has become obvious. Lacking flexibility concerning changes of geometry types as well as the awkward handling of predominantly batch operating grid generators causes a disproportionately high time expenditure. An abundance of more or less user-friendly grid generators with different degrees of automation have been developed, they also work still satisfactorily in practice, but only as long as no substantial changes are made within the task. At the latest then indeed, the input and the program itself has to be modified. Time consuming reprogramming and testing become necessary. Additionally to this modifications, the conventional approach of batch programs in combination with plot procedures does no longer represent the state of the art anyway. Definition of parameters, generation of grids, their visualization and their inspection are single steps within this approach, only their iterative application to a problem can lead to a finally acceptable computing model. With respect to a fast, reliable and highly flexible mesh generation process, the necessity of an integrated system comprising some proved modules but offering also the advantages of the interactive technique is obvious.

A first success in this direction arose with the establishment of the program package INGRID (INteractive GRIDgeneration system) for the generation of blockstructured volume grids. During its application in combination with an (arbitrary) commercial CAD-System, there are no restrictions with resp. t to the geometry to be processed. Grids for internal and external flows, for Euler- and Navier-Stokes-simulations therewith can be be generated, visualized, controlled and modified within shortest times. During construction of the program system, some already proven batch modules have been used. However, as the system is still open with respect to the coupling of further routines, also the two methods which are still under development and which are presented at the end of this report could be connected to the interactive operation mode.

The system INGRID at the present time is allready a tool to be used productively for the generation of blockstructured grids for field methods. Long term aim is the completion to a flexible and versatile supplementary aid, which integrates the diverse common procedures for the necessary preprocessing to only one interactive and user-friendly package. The modules described in chapters 3 and 4 up to now are still batch operated - but within the near future they should also become integrated into the interactive environment.

Indeed all of the presented grid examples were used to calculate flow solutions, but meshes generated the same way could be used also for the predection of radar backscattering. Either surface grids according the concept of the physical optics or the volume grids for the solution of Maxwell's equations (electrodynamics) [1].

2. THE INTERACTIVE APPROACH

The base of any industrial flow simulation is either a projected or an allready existing geometrical shape. "Computer Aided Design"-systems nowadays are installed in most of the companies where geometrical models have to be treated with respect to any development and/or manufacturing. Graphic terminals or even workstations came along with those CAD-systems, and a lot of engineers became familiar with the interactive techniques to communicate with commercial application programs via messages and menus. Additional software libraries became available, which enabled a programmer to write his own custom tailored application interface, where specialized algorithms are combined with the ability to create, display, and interact with graphics data. Then it was at the time to rearrange the traditional batch oriented grid generation procedures according to this popular working method. The advantages are obvious, within a dialog and under permanent visual control step by step (and even backwards) a basic geometry can be upgraded to a final network. Several proven algorithms are available and can be selected and variations can be tried to find the best possible solution. Generation parameters may be modified rapidly to study the mesh behaviour. Routines to check the mesh quality can be used, so that possible mistakes become obvious immediately but with the chance to be corrected without delay. As an intelligent workstation is used, generation algorithms, data administration and interaction control are running on a host-computer while visualization and transformation are downloaded to local

processors, thus saving computing time and accelerating the process. In addition, the comprehensive possibilities of 3D-representation of such a modern workstation, supplies picture sequences, which some years ago were only possible with complex trick film techniques. The overall cycles of geometry changes, parameter variations, mesh control and visualization, formerly taking weeks with previous methods, are compressed to several interactive sessions within days. This technique "allows the user to concentrate on the geometry of the problem rather than on the mechanics of the processing programs" as Eiseman and Erlebacher [2] remarked.

2.1 Preparation of the Geometry

The entire procedure of a gridgeneration, independently of the supplementary aids with which it is accomplished, can be divided into two sub-tasks: geometry-preparation and grid-generation. The first part leads to a configuration description by means of suitable geometrical elements. Since for pure geometrical tasks several interactive program systems are already existing and available within the industry, such a softwarepackage and an appropriate installation can be used. The starting point should be a sufficiently detailed geometry model of the desired configuration available within such a system. As an example a CADAM wire frame model of the coming Dornier utility aircraft Do 328 is shown in fig.1.
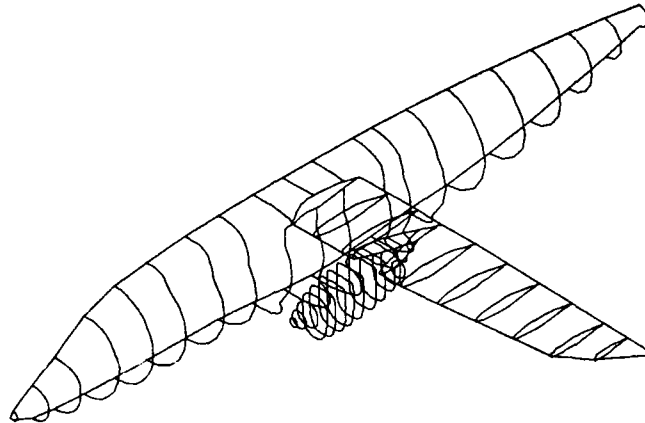

Figure 1. Some sections of a Do 328 CADAM wireframe model description

Using the standard functionality of the CAD-system, the shape and location of the far-field boundary is defined and the block decomposition is carried out. Fig.2 shows the far-field and the overall blockstructure of the example geometry
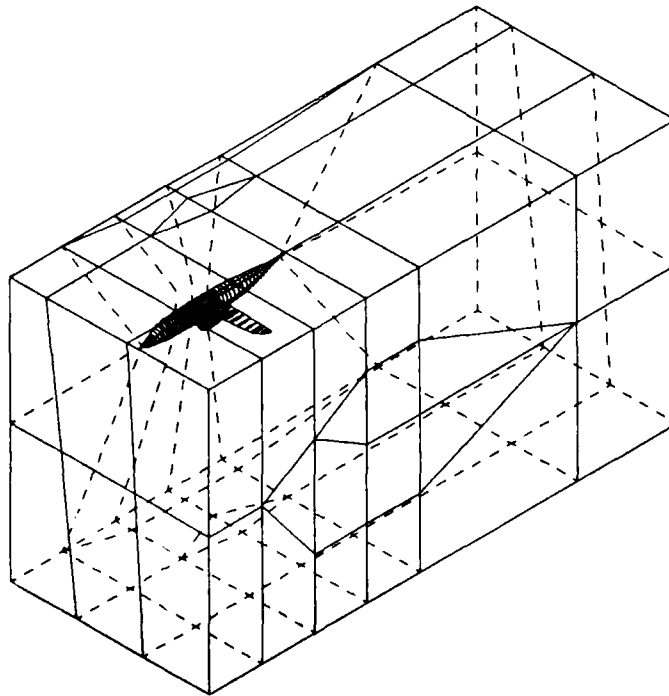

Figure 2. Far field boundary and possible block arrangement

The representation of the body surface is to be subdivided and arranged according to the chosen block-topology. Fig.3 gives a more detailed view of a prepared configuration.
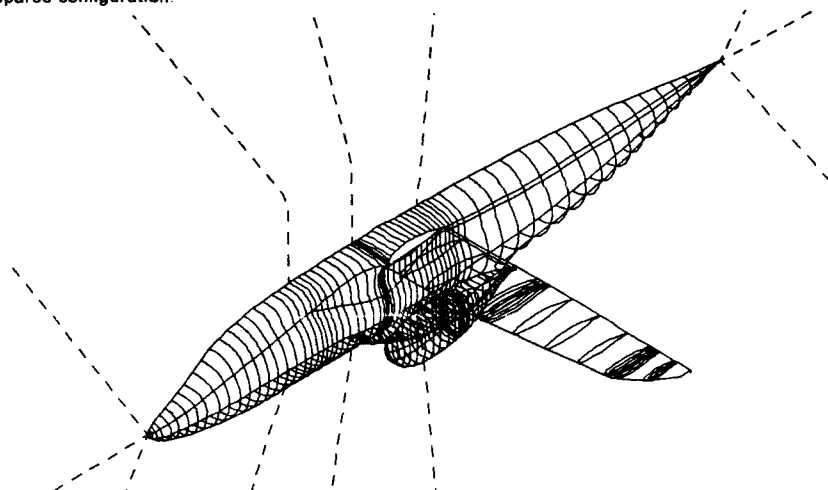

Figure 3. Do 328 - prepared surface representation and some blockboundaries

The result of the first step within the CAD-system and a subsequent interface program is a datafile containing the so called base geometry. This is a collection of geometry elements - more exactly: points and point-sequences - giving the necessary information to provide a sufficient description of the overall geometry. Sufficient in that context means that the interpolation procedure based on a cubic spline formulation and used during grid generation should be able to meet the actual geometry shape within acceptable tolerances

2.2   Grid generation using INGRID

For the second step the user interface application program INGRID containing the descretization algorithms is activated. First of all from the displayed base geometry the elements which should form an active block have to be selected. The order of picking edges implicitly assigns the counter directions I, J and K. Optionally at block faces additional surface lines can be specified - this is necessary especially for all block sides located next to twice curved body surfaces. After the logical connections between all elements of one block are completed, point quantities and up to twelve distribution functions can be specified. All edge elements are treated one after the other. The redistributed nodes are visualized immediately and can be withdrawn if the result is not satisfying. During generation for the very first block of a configuration all these parameters have to be specified by the user - respectively selected out of the possibilities offered within pop-up menus. While treating subsequent blocks, faces of neighbouring and allready existing meshes might be called on to the screen and counters and distribution functions can be transferred - ensuring consistent meshing between adjacent blocks. When the node distribution for all edges is complete, the block surface grid generation is executed. Finally, after specifying the desired type of integer plane, the volume grid generation is performed for the active block and results are shown plane by plane

While executing the step by step generation for edges, faces and internal counter planes forming the volume-grid, at each time the momentary results might be either accepted, optionally be modified or withdrawn. Additional routines optionally can be selected, for tracing coordinates and integer counters, for local and global grid modifications and to hunt up 'negative volumes'. After the active block has been completed, the base geometry is recalled and the procedure restarts with the selection of the geometry elements for the next block. All these actions have to be repeated until all blocks of a complete configuration are processed. Fig 4 shows the final mesh at the aircrafts body surface
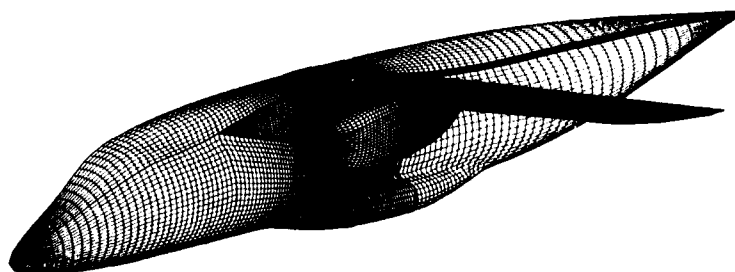

Figure 4. Do 328 - final mesh distribution at the surface

2.3   Implemented Generation Techniques

The basic routines of the program system INGRID are partly common algorithms, proved to be versatile during countless applications within batch oriented procedures, and partly new programmed modules which were tested and modified within interactive sessions. The core of the generation part is a procedure for point redistribution along curves in space: similar procedures are in industrial use to produce the necessary data for NC-milling. Starting with a number of base points, first of all a parameterized cubic spline is evaluated, its formulation allowing only pure interpolation without any smoothing. After that, the generation of intermediate points along those splines is done according to a desired one-dimensional distribution. As there are no restrictions in this method any imaginable point distribution can be achieved

The application of this procedure to block edges leads to the desired grid points. For block surfaces the same procedure is applied along all given surface-lines and at least with two sweeps into the different counting directions. In the case of complicated boundaries one ore two repetitions might become necessary until the changes within the distribution become equal to zero. Finally again the same respline procedure is repeated in the various space directions until the volume discretization is complete. But as block faces and internal integer planes are usually bounded by three or four edges, where in a general case each might have its own distribution function, some blending must be done for the interior. Taking into account the influences of the distributions at the boundary curves is done by a repetitive mapping procedure.
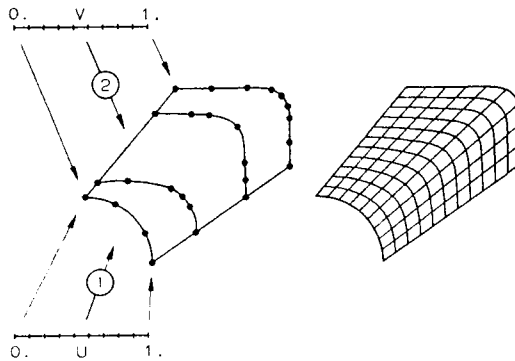
Figure 5.
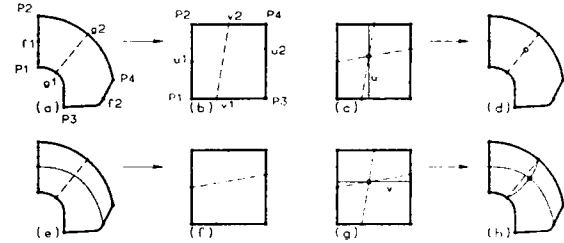Generation of grid points at block surfaces

Figure 6.
Mapping procedure to evaluate internal distribution functions
(2 sweeps)

Consider a four-sided physical region whose sides are f1, f2, g1 and g2 (fig.6a). u1, u2, v1 and v2 represent the normalized parametric distribution functions of the four sides (fig.6b). As the respline procedure needs an initial solution, for the first sweep (in the f-direction) a linear connection between opposite boundaries is assumed. Its direction has to be specified interactively to pay attention to the actual shape' The calculation of the line-intersection within the parametric space (c) leads to a weighted distribution function u. which can be applied to respline the initial solution within the physical space (d). For the second sweep (in the g-direction). using the same steps. the weighted distribution function v is calculated (g) and applied in the transverse direction (h)

The procedure of node redistribution within integer planes can be applied repetitively for arbitrary plane types (I.J.K = const) The more complex the block shape is. the larger the number of sweeps will be necessary to achieve a satisfactory solution.

The method of Thompson et al [3]. wherein the grid is derived from the solution of a set of inhomogeneous Laplace equations is also implemented within INGRID and can be used optionally to do mesh modifications. During applications for various geometric shapes. it turned out that results of the redistribution method could be achieved faster and were often more suitable than the comparable solutions of the differential equation method. Fig.7 gives two examples with different boundary shapes
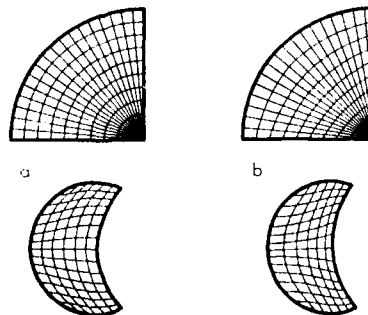
Figure 7. Results of node redistribution (a) and differential equation method (b)

The left hand side shows the solution of the redistribution method achieved with 1 and 3 sweeps. while the right hand side shows the grid generated by solving the Laplace equations. In both cases the interior of the redistributed mesh is oriented much closer to the given distribution at the boundary curves. Without the necessity to adjust control functions, as required if the Poisson equation system is used. the redistribution method immediately results in a smooth mesh. keeping the boundary characteristics proportional throughout the whole field. An additional and essential advantage of the method is its ability to apply it directly to arbitrary body shapes. As it works along real 3D-curves. keeping given shapes all the time. this simple and fast method represents an attractive alternative to the much more complicated formulations of the specified surfaces according to Thomas [4] or the Gaussian surfaces as described by Warsi [5].

2.4 INGRID Environment

During INGRID applications all the graphic support an intelligent workstation being able to give. can be used. Local real-time animation of the wire-frame representation and selective "show/noshow"-procedures of grid planes enable the user. to get and to keep continuously a complete overview of all details of a spatial network. The program development as well as the examples presented here were carried out on a SPECTRAGRAPHICS 1500 workstation connected to an IBM 3090 host computer. The base geometries were established by means of the commercial software packages CADAM and CATIA. The mesh generator with the user interface application program IN-GRID uses the device-specific soft- and firmware called PRISM for graphic access.

2.5 INGRID Application Examples

Fig.8a indicates the block architecture of a local O-mesh imbedded in a global H-structure for a wing with pylon and load. In 8b the grid on the surface is shown as well as within the plane of symmetry. The appropriate computational analysis was done to investigate increases of the drag due to local transonic effects.
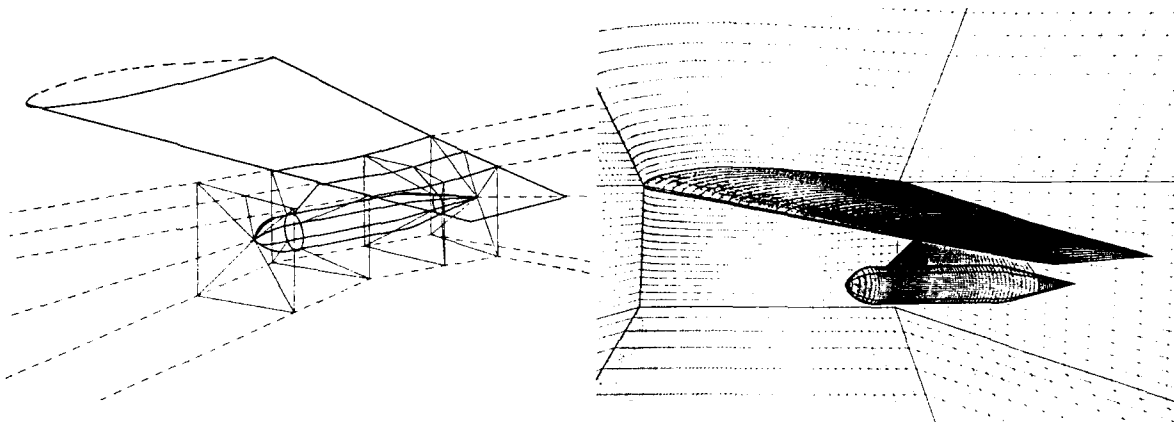

Figure 8. Local O-mesh imbedded into a global H-structure for wing with pylon and load

Fig.9 gives a glimpse of a composite grid arrangement for Navier-Stokes analysis of a car configuration. Blocks with very high resolution are located within the expected boundary layer region surrounded by a global H-structured mesh.
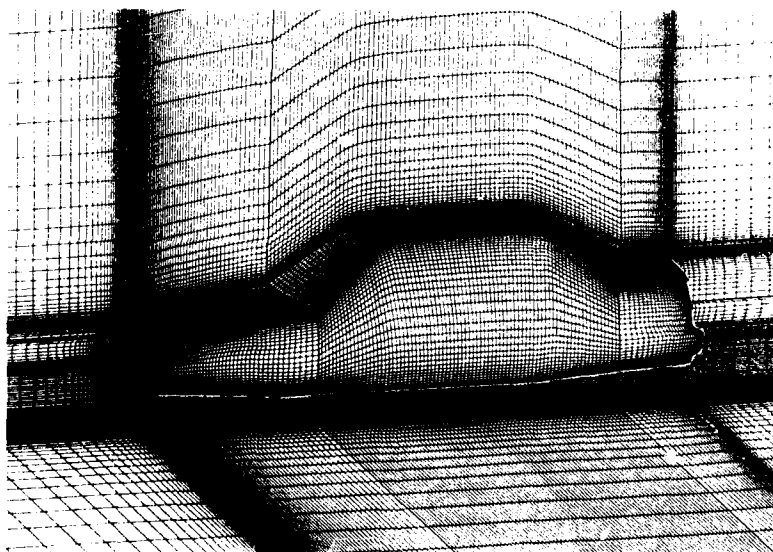

Figure 9. Composite grid for Navier-Stokes analysis of a road vehicle

Fig 10 shows an internal flow problem - a segment bend pierced by a valve lifter (a) gives the original surface model and (b) the base-geometry, both done with CATIA. Views (c) and (d) show some details of the generated grid in the interior and on the surface of the configuration.
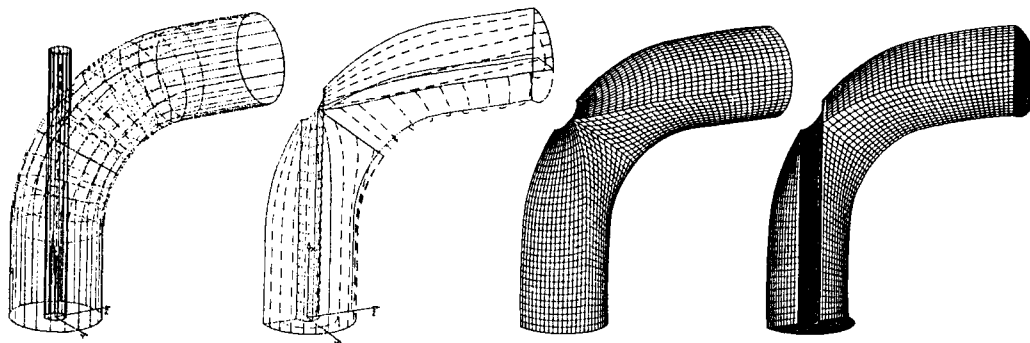
Figure 10. Segment bend pierced by a valve lifter - geometry model and generated grid

More interactively generated examples are given in references [6] and [7].

3.   COMBINED ALGEBRAIC-HYPERBOLIC GRID GENERATION

For the aerothermodynamic flowfield analysis of reentry vehicles within the Mach number region 2 to 20 a special approach has been developed. At the present time this procedure is still batch operated, but because of its modular structure it is well suitable to become integrated soon into the interactive environment of INGRID.

The process features the steps as described in the following. First of all based on a geometry definition a surface grid is determined by the application of an algebraic grid generator. A sectionwise point distribution is generated, where the grid density is adjusted according to the local curvature of the geometry. Determination of the farfield shape is done by prescribing a lower and an upper angle of inclination within the plane of symmetry. Those values are adapted to the Mach number and the angle of attack in order to optimize the grid without wasting points in regions of low interest. The cross section shape of the farfield is that of a general ellipse, where the ratio of the main to the sub-diagonal also depends upon the freestream conditions. After predefinition of such a farfield, it is treated with the same algebraic grid generation process as the surface. The internal grid points are calculated using a 2-d hyperbolic algorithm, which is applied sectionwise. The resulting orthogonal grids are especially desirable for complex shaped cross section contours. As for hyperbolic marching the farfield distance can only roughly be prescribed, however not a specific shape, the farfield definition of the algebraic grid generator is used. The intersection of the radial hyperbolic gridlines with that boundary is determined and the internal hyperbolic grid is then redistributed by a local adaption within each section. Either an affine radial stretching according to the hyperbolic grid can be used whereby the grid height of the first cell is kept. Or a geometrical stretching function with a fixed initial grid size can be applied. A procedure generating a smooth transition from an internal algebraic to the full hyperbolic grid is also available. Within an application for the Hermes configuration. additional stretching functions are used to ensure nearly equidistant distributions in the region of the front shock wave. Other input parameters allow to avoid crossovers in concave regions or for example to even out the grid size in tangential directions in prescribed regions. This sectionwise application of the hyperbolic grid generater as well as the reshaping of the internal grid in some cases may cause minor grid irregularities between neighbouring sections. Therefor subsequently a 2-d and/or 3-d Poission solver or a 3-d smoothing operator can be applied [8].
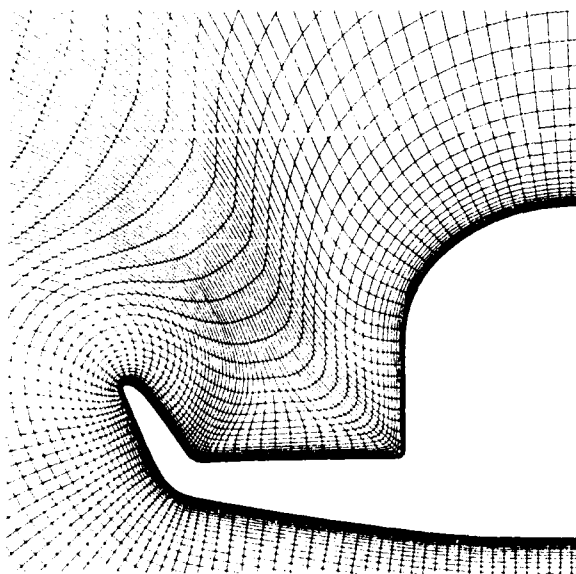

Figure 11
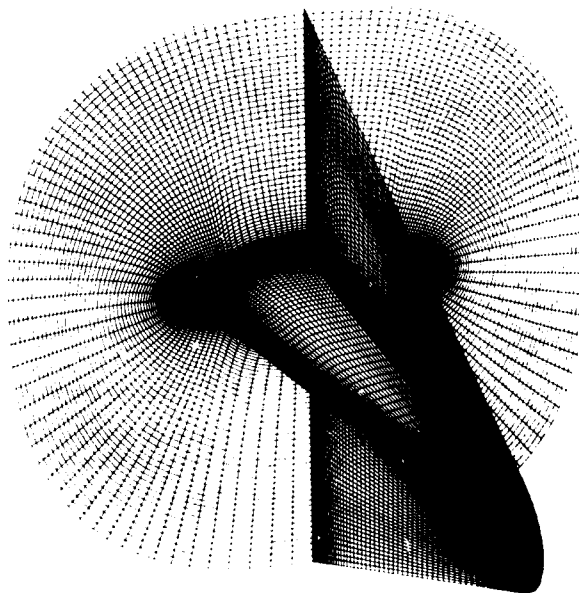*HERMES - Grid within typical cross section*

Figure 12.
SAENGER - Some grid planes and their topology

4. ADAPTIVE GRIDS

It is well known, that the accuracy of a numerical solution depends on the fineness of the mesh - the finer the grid, the more accurate the numerical solution will be. The presence of large gradients causes the error to be large in the the approximation of the derivatives. Especially in the presence of shock waves, more artificial diffusion must be added to retain adequate smootheness of a solution. That is the reason for an urgent need for schemes which are able to resc:/e those large gradients without the necessity of adding additional grid points. An adaptive scheme moves given grid points to regions of high gradients in cases when the locations of these gradients are not known a priori. An adaptive method reduces also the total number of grid points required to achieve a given accuracy. Adaptive grid generation schemes can run in partnership with a flow code and dynamically adapt the grid to the evolving flow field, which is known as dynamic adaption, or they can adapt one grid which will remain invariant for the entire calculation, which is called the static adaption.

4.1 GRID ADAPTION BY ALGEBRAIC REDISTRIBUTION

The first approach described hei 2, is based on an algebraic generation scheme (see for example [11], [13]). The redistribution of grid points along arbitrarily shaped lines according to the curvature of a sensor function can be used for the static adaption of 2-D grids and also for the field adaption of 3-D grids. It is assumed that the redistribution of grid points should be based on the distribution of the curvature of a typical, the flow field describing function u (for example: surface pressure distribution). The curvature is obtained at each point i by the central difference approximation

$$a_i = u_i = \frac{2}{h_3} \left\{ \frac{u_{i+1} - u_i}{h_2} - \frac{u_i - u_{i-1}}{h_1} \right\} + O\left(h^2, h_2 - h_1\right) \tag{4.1}$$

using forward and backward difference operators. For sake of simplicity we may set $a_1 = a_2$ and $a_N = a_{N-1}$. By nomalizing the curvature with the constant step size h,

$$h = \frac{x_N - x_1}{N - 1} \tag{4.2}$$

we obtain a weighted measure k_i of curvature at each point:

$$k_i = a_i \frac{h_i}{h} \tag{4.3}$$

with

$$h_i = x_i - x_{i-1}. \tag{4.4}$$

In order to damp extreme values in curvature and to increase the interval of influence, a new measure of curvature.

$$\alpha_i = \frac{1}{2n + 1} \sum_{j=0}^{2n} k_{i+j-n}, \quad i = n + 1, \dots, N - n, \tag{4.5}$$

is introduced for inner points. At boundaries a similar but one-sided formula is used. In all cases described here, a value of $n = 1$ was used. resulting in smoothing three points.

The transformation function is finally obtained from the integration of alpha

$$S_i = \sum_{j=2}^{i} \alpha_j. \tag{4.6}$$

with $S_1 = 0$. One notices that the transformation function $S(x_i)$ has its maximum slope where the curvature of $u(x_i)$ has its maximum curvature and its minimum slope where the curvature of $u(x_i)$ is also minimal. The table of values obtained from $S = S(x_i)$ can also be used in its inverse form $x = x(S_i)$. By dividing the interval

$$S_N = \frac{1}{h} \int_{x_1}^{x_N} \alpha_i dx \tag{4.7}$$

into N -1 subintervals

$$S_i^+ = S_N \frac{i - 1}{N - 1}, \quad i = 2, 3, \dots N.$$

one can obtain through interpolation the new distribution $\bar{x} = \bar{x}(S_i^+)$. In order to guarantee monotonicity this interpolation must be linear then from the existence theorem the inverse function exists because S_N is continous.

The new step sizes found by the procedure just described depend completely on the behaviour of the function $u(x_i)$. If this function is piecewise linear, some of the α_i become zero. This can lead to uncontrollably large step sizes. Since however, the accuracy of numerical methods always depends on the chosen step size, an additional condition must be introduced, controlling the maximum interval between two adjacent points. The step parameter P is defined as

$$h_{max} = P h \tag{4.8}$$

Where h is again the step size for uniform point distribution. The gradients of S(x) are now compared against a minimum value

$$q = \frac{S_N}{(n - 1)h_{max}} \neq 0 \tag{4.9}$$

which is controlled by P. Therefore it proves neccessary to use an additional linear transformation in order to ensure such a minimum gradient of value q. Figure 13 shows an example for this adaption techique for a C-type mesh around an airfoil.
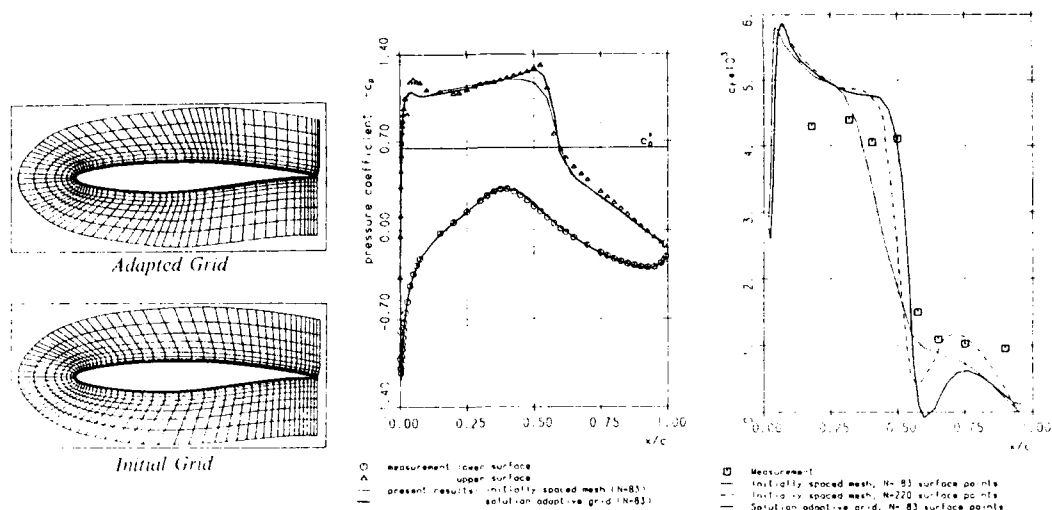
Figure 13. RAE-2822 Airfoil - 2-D grid adaption by algebraic point redistribution

The initial point spacing is already non-uniform, having more concentrated points at the leading and trailing edges, in these regions a pressure distribution is assumed a priori showing larger curvature. The adapted grid is based on the surface pressure distribution calculated by means of the initial mesh. Therefore concentrations of mesh points at the approximate middle of the upper and lower surface as well as at the trailing edge are due to the curvature of the pressure distribution. The effect of the adapted grid can easily be recognized in the surface pressure distribution as well as in the surface skin friction. On the upper surface the shock region is much better represented as well as the pressure plateau in front of the shock. Of significance in the prediction of the aerofoil force coefficients is the calculation of the wall skin friction coefficient c_f. Furthermore, but already indicated by the pressure distribution, a much better representation of of the shock region can be obtained by the use of the adapted grid. More details about this 2-D grid adaption technique are given in [11]

The same technique was applied for the field adaption of the grids for the flow calculations with a Parabolized Navier-Stokes (PNS) method [12]. Within the cross sections in streamwise direction, the grid points have been redistributed along the radial coordinate directions. Figure 14 shows the coordinate systems and the static pressure distribution for an ogive in supersonic flow. The better resolution of the shock can clearly be seen and the adapted grid shows already the position of the shock.
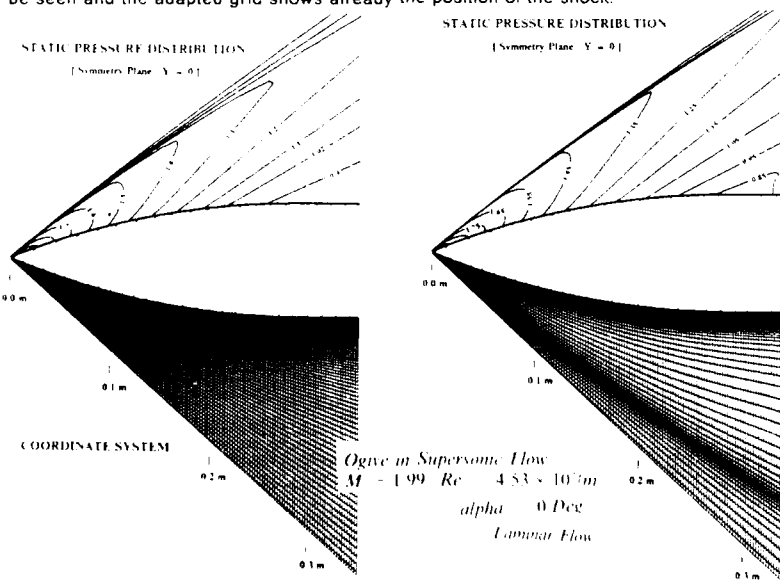

Figure 14. Ogive at supersonic flow - 3-D grid adaption by algebraic point redistribution

4.2 GRID ADAPTION WITH LOCAL REFINEMENT

A second adaption scheme is used for the generation of 2-D block structured grids with local grid refinement. This method is based on the solution of elliptical differential equations including weighting functions from the flow solution. It is suitable for the generation of 2-D adapted block structured grids.

The use of adaptive grids in combination with local grid refinement combines the advantages and cancels the disadvantages of each method. So the use of adaptive grids requires a high number of grid points to avoid jumps in the grid spacing. On the other hand, the use of fine subgrids would be a very good approach for viscous flows, if the boundaries of those subgrids could be adapted to the structure of the flow field. If additionally a block structure, which is adapted not only at the geometric requirements but also at the structure of the flow field (see for example refs. [13] and [14]), is used, we will have a very effective discretization of the flow field, (adaptive grids with local grid refinement) and also a very effective procedure for the flow solution by the use of zonal approach (Euler/Navier Stokes) which due to the block structure can be done very simply.

With the computational cordinates (ξ, η) and the physical coordinates (x,y), the commonly used elliptical partial differential equations suggested by Thompson [15] for grid generation can be written as:

$$A X_{\xi\xi} + B X_{\eta\eta} + C X_{\xi\eta} + QI X_{\xi} + QJ X_{\eta} = 0 \tag{4.10}$$

wherein the $X = (x,y)$ are the cartesian coordinates of the grid points. The mixed derivative $X_{\xi\eta}$ can be neglected without changing the elliptical character of the above equation. The constants A, B and C are fixed by the transfomation operations between the physical and the computational space, only the control functions QI and QJ can be used for grid control and for an adaption of the grid. According to Thompson [16], adaptive grids can be constructed by including weighting functions of the flow solution in those control functions

The condition for a one dimensional adapted point distribution along the ξ- or i- direction in the computational (index) space is described by the relation:

$$W_{(i)}\Delta x_i = const.$$

Where $W_{(i)}$ is any weighting function. In the computational space this yields to

$$W_{(\xi)} x_{\xi} = const.$$

or

$$x_{\xi\xi} + W_{(\xi)} \frac{1}{W_{(\xi)}} x_{\xi} = 0 \tag{4.11}$$

which is a one dimensional Poisson equation. Compared with the equation (4.10), the above equation is identical with the one dimensional elliptical PDE. If in the one-dimensional version of (4.10) which is

$$A X_{\xi\xi} + QI X_{\xi} = 0$$

the source term QI, which is used for grid control is replaced by

$$\overline{QI} = QI + \frac{W_{(\xi)\xi}}{W_{(\xi)}}$$

the one dimensional discretization will be adapted to the weighting function $W_{(\xi)}$. In two dimensions the adaption of the grid to a weighting function can be achieved by replacing the weighting functions in (4.10) by:

$$\overline{QI} = QI + \frac{W_{(\xi)\xi}}{W_{(\xi)}}$$

$$\overline{QJ} = QJ + \frac{W_{(\eta)\eta}}{W_{(\eta)}}$$

where $W_{(\xi)}$ is the weighting function in the ξ or i-direction, $W_{(\eta)}$ that of the η or j-direction. With those modified source terms the PDE (4.10) can generate solution adaptive grids if the weighting functions are taken from the flow solution. The choice of the weighting function depends on the nature of the flow field. In flow direction and along surfaces, the weighting function should be coupled with the pressure distribution and in j direction, which is the direction normal to the main flow direction, the weighting function should be coupled with any indicator for viscous effects. Numerous experiments with different weighting functions have shown that the best weighting function for the computational i-direction is given by the relation

$$W_{(\xi)} = \alpha \frac{\partial p}{\partial x} + \beta \frac{\partial^2 p}{\partial x^2} \tag{4.12}$$

So the weighting function is a combination of the first and the second derivative of the pressure distribution. This gives a grid adaption to pressure gradients and extreme values. α and β are weighting parameters by which the user can make the first or second derivative more ore less dominating. In transonic flow, the gradient of the local Mach number is also a very suitable weighting function adapting the grid to shock waves. Normal to surfaces, which is usually the η or j-direction, possible weighting functions may be the total pressure loss- or the vorticity distribution. It was however found out, that the total pressure loss is the most suitable parameter to drive the grid adaption to any flow field discontinuity, because its values move within a small range whereas the values of the vorticity spread over several powers of ten. So the weighting function for the j-direction has been chosen as

$$W_{(\eta)} = \gamma(1 - \frac{p_t}{p_{t0}}) \tag{4.13}$$

Where γ again is a scaling parameter. Of course the weighting functions have to be smoothed and are normalized with the extreme values. To avoid an "overadaption" of regions with extreme gradients, it is also necessary to damp extreme values in the weghting functions. The grid adaption can be performed in 3 levels. Adaption of the surface point distribution along the surface to the surface pressure distribution, adaption of the field grid points normal to the flow direction, and the adaption of the field grid points in flow direction. The perimeter adaption along the surface is done by the solution of a one dimensional elliptical PDE

$$\frac{W_{(\xi)}}{W_{\xi}} x_{\xi\xi} + x_{\xi} = 0 \tag{4.14}$$

If the above equation is approximated by finite differences in the index space this leads to a simple tridiagonal equation system. The weighting function is given by equation (4.12). For this surface adaption, only the surface pressure distribution or the Mach number distribution along the surface is required. For the field adaptions the weighting functions according to eqs. (4.13) and (4.14) are taken

The local grid refinement is treated as follows. First the uniform, finest grid is generated. Then the coarser blocks are obtained by eliminating each 2, 4, 8 grid point in i- and/or j-direction. For the grid adaption, the weighting functions of the flow solution are interpolated into the uniform fine grid and the grid adaption is performed for this uniform fine grid and finally the coarse subgrids are regenerated

If the flow solution operates in a sequence from coarse to fine grids (multilevel grid technique), at each switch from a coarser to the next finer grid this finer grid can be adapted by the results of the coarser grid. So the grid points are automatically concentrated in regions with highly dominating viscous effects

If once the weighting parameters α, β γ have been calibrated for a certain configuration, the described adaption method is very stable. It is very suitable to all flow fields wich have viscous regions embedded into an inviscid outer region. For internal flows however, the viscous regions can be extended over the complete flow field and the total pressure loss can be no suitable weighting function. In those cases the second derivative of the velocity profiles were a better indicator for separated regions. It was also found, that the field adaption to the field pressure distribution has no advantages as long as there are no pressure discontinuities in the flow field. The adaption of the grid to the surface pressure distribution is sufficient and can be done once at the beginning of the calculation. Figure 15 shows the flow field around a 2-D fast back car body and the adapted grids at 2 different solution levels. In the flow field one can recognize separation at the rear wind shield, the wake region and the boundary layer along the fixed ground plate behind the car. The grid is adapted to the

surface pressure distribution by the solution of equation (4.14) with (4.12) as weighting function. This adaption has been done once at the beginning of the calculation and gives a concentration of grid points in regions with pressure gradients and extreme values. The field adaption was carried out two times during the solution process taking (4.13) as weighting function in η - direction and without field adaption in ξ - direction.
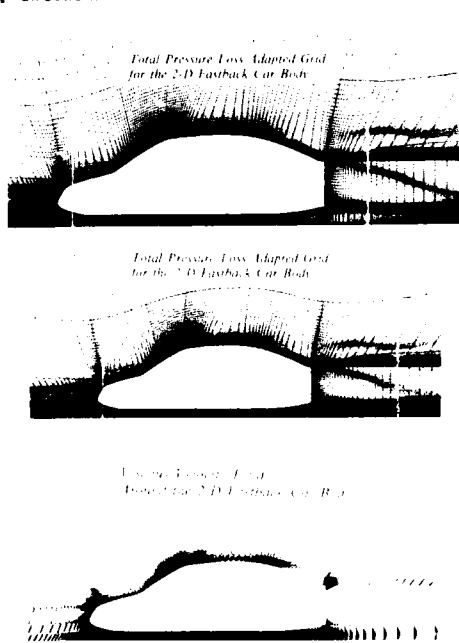

Figure 15
2-D Grid adaption by elliptical PDE's
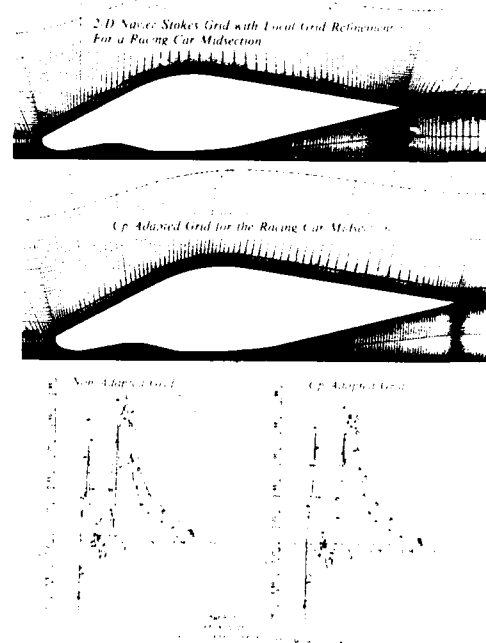
Figure 16
2-D Grid adaption by elliptical PDE's

Figure 16 gives an example for the influence of the grid adaption on the surface pressure distribution. The configuration is an idealized midsection of a racing car and the grid is a H-type grid with local grid refinement in the inner blocks consisting of a total number of 22536 cells. In the initial grid the grid points around the contour are concentrated at the leading- and trailing edge region. In the c_p-apted grid only the point distribution around the contour was adapted to the surface pressure distribution and again the grid points are concentrated in regions with pressure gradients and -extreme values. The surface pressure distribution is the result of a 2-D Navier Stokes analysis at a Reynolds number of $Re = 1.8 \times 10^6$. With the c_p adapted grid, the pressure distribution becomes smoother the gradients and the extreme values are better represented

An example fo the reduction of the numerical error is given in Figure 17

2-D Grid Adaption by Elliptical PDE's

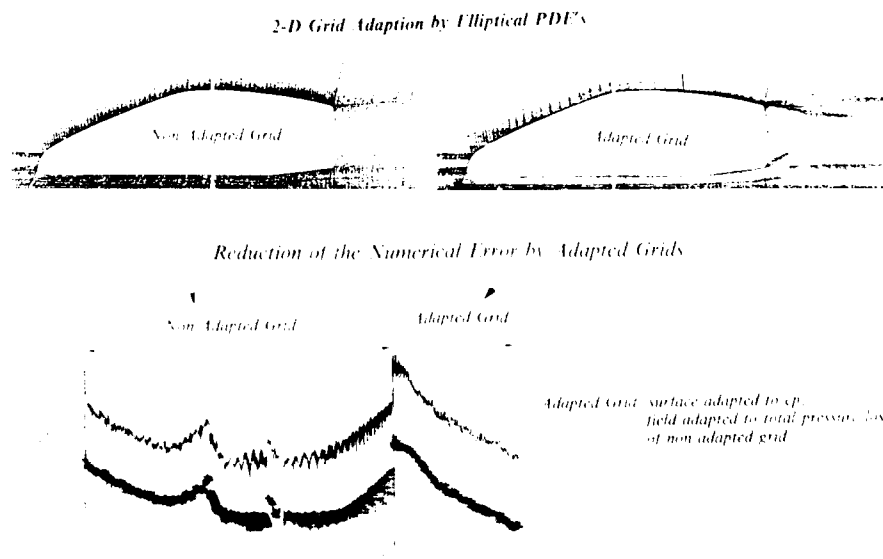### Reduction of the Numerical Error by Adapted Grids

Figure 17 2-D Grid adaption by elliptical PDE's

In the non-adapted coarse grid a 2-D Navier Stokes calculation was started with the aime to get a solution for the adaption of a fine grid. As it can be seen in the plot of the convergence behaviour, it was impossible to get a converged solution in that grid. With the adapted grid (along the surface to the pressure distribution according to (4.12) and normal to the surfaces to the total pressure loss according to (4.13)) a very good convergence rate was achieved

The Figures 18 and 19 and 20 show the use of the static grid adaption scheme within the multilevel grid technique during a 2-D Navier Stokes calculation for an airfoil with a 30 degrees deflected trailing edge flap.
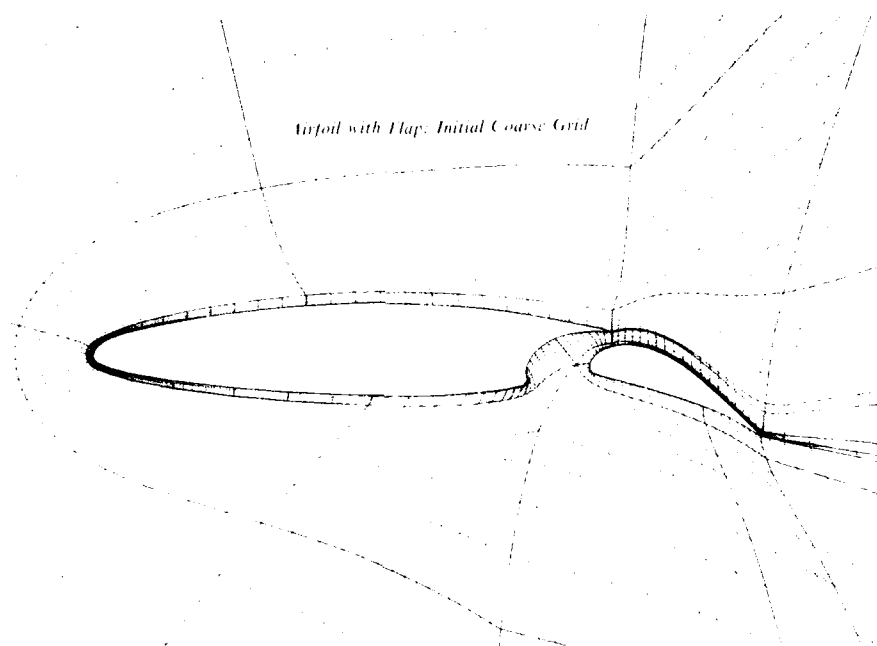

Airfoil with Flap. Initial Coarse Grid

Figure 18   Airfoil with flap - initial coarse grid
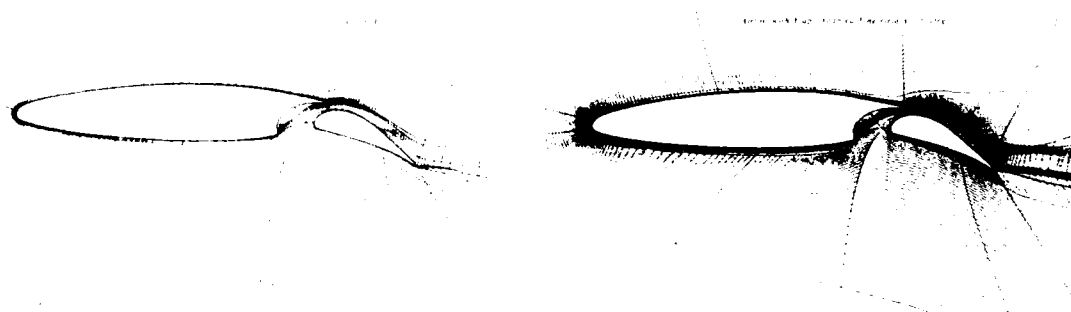

Figure 19
Airfoil with flap - adapted medium grid

Figure 20
Airfoil with flap - adapted fine grid

To avoid the typical singularities of a H-type grid a system of patched block structured C - meshes has been generated for that configuration. Both airfoil and flap have a thick trailing edge and the C - meshes are closed by an additional trailing edge block. The flow solution was started in the coarse mesh. The medium grid of figure 19 is adapted to the results of the coarse grid and the development of the boundary layers and the wakes can be seen in the medium grid where an adapted inner grid is embedded into a coarse global grid. The final fine grid (40000 points) of figure 20 is adapted to the result of the medium grid and shows a very fine resolution of all viscous regions.

5.   CONCLUSIONS

There is still a lot of work which has to be done forming a really flexible mesh generation system capable to handle all the various problems which are of industrial interest. Promising new batch modules as described in chapters 3 and 4 are under development for special applications. But the location where more and more meshes are generated effectively nowadays and in the future is the graphic workstation. CAD-systems and graphic-interactive application programs each provide a great help within its part of the complete working process concerning handling and control of geometries and meshes. The grid generation system INGRID is a step towards the integration of standardized procedures and also new developed modules with the aim to form a user-friendly and productively applicable tool for the mesh generation. Integration of further modules into the current system will enhance its versatily and provide a powerfull collection of tools to match most of the industrial CFD-tasks. The generation of complex grid structures is simplified clearly by the application of such an interactive procedure - various tasks for general configurations become solvable rationally only due to such an approach.

5. REFERENCES

[1] Grashof, J.
FINITE-VOLUME METHOD FOR THE TIME-DEPENDENT MAXWELL EQUATIONS AND
PREDECTION OF THE SCATTERING FROM PERFECTLY CONDUCTING BODIES
The Third International IGTE-Symposium on
Numerical Field Calculation in Electrical Engineering
Graz, Austria, Sept. 1988

[2] Eiseman, P.R., Erlebacher, G
GRID GENERATION FOR THE SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS
ICASE Report No. 87-57, August 1987

[3] Thompson, J.F., Mastin, C.W.
GRID GENERATION USING DIFFERENTIAL SYSTEMS TECHNIQUES
Numerical Grid Generation Techniques
NASA CP 2166, 1980

[4] Thomas, P.D.
CONSTRUCTION OF COMPOSITE THREE DIMENSIONAL GRIDS FROM
SUBREGION GRIDS GENERATED BY ELLIPTIC SYSTEMS
AIAA CFD Conference, Palo Alto, 1981

[5] Warsi, Z.U.A.
BASIC DIFFERENTIAL MODELS FOR COORDINATE GENERATION
in "Numerical Grid Generation", Ed. Joe F. Thompson
North-Holland Publications, 1982

[6] Seibert, W
AN APPROACH TO THE INTERACTIVE GENERATION OF BLOCK STRUCTURED
VOLUME GRIDS USING COMPUTER GRAPHIC DEVICES
Numerical Grid Generation in CFD, Ed. Häuser and Taylor,
Proc. of the Int. Conf. in Landshut, FRG. 1986

[7] Seibert, W
A GRAPHIC-INTERACTIVE PROGRAM-SYSTEM TO GENERATE
COMPOSITE GRIDS FOR GENERAL CONFIGURATIONS
Numerical Grid Generation in CFD, Ed. Häuser and Taylor,
Proc. of the 2. Int. Conf. in Miami Beach, Florida, 1988

[8] Haase W Leicher S Rieger H
HERMES - AEROTHERMODYNAMIC FLOWFIELD ANALYSIS
AT HYPERSONIC SPEEDS
Dornier Report BF 11/88 B, December 1988

[9] W. C. Connet, R. K. Agarwal, A. L. Schwartz
AN ADAPTIVE GRID ALGORITHM FOR THE EULER/NAVIER-STOKES EQUATIONS
AIAA Paper-88-0519

[10] J. S. Abolhassani, S. N. Tiwari
GRID ADAPTION FOR BLUFF BODIES
NASA -CR-180329 1986

[11] W. Haase, K. Misegades, M. Naar
ADAPTIVE GRIDS IN NUMERICAL FLUID DYNAMICS
International Journal for Numerical Methods in Fluids
Vol. 5 1985 pp 515-528

[12] H. Rieger, H. Mathauer
THERMISCHE AUFHEIZUNG VON FLUGKORPERN BEI HOHEN GESCHWINDIGKEITEN
DORNIER Report BF 7/86B, 1986

[13] W. Fritz, W. Haase, W. Seibert
MESH GENERATION FOR INDUSTRIAL APPLICATION OF EULER
AND NAVIER STOKES SOLVERS
Paper in "AGARDOGRAPH on Numerical Grid Generation in CFD" 1987

[14] W. Fritz
NUMERICAL SIMULATION OF 2-D TURBULENT FLOW FIELDS
WITH STRONG SEPARATION
ICAS Paper ICAS-88-4.6.4

[15] J. F. Thompson
A SURVEY OF GRID GENERATION TECHNIQUES
IN COMPUTATIONAL FLUID DYNAMICS
AIAA Paper 83-0447

[16] J. F. Thompson
GRID GENERATION
Lecture at "Navier Stokes Flow Simulations"
AIAA/VKI Study Seminar September 15-16 1986

GENERATION, OPTIMISATION ET ADAPTATION
DE MAILLAGES MULTIDOMAINES
AUTOUR DE CONFIGURATIONS COMPLEXES

Olivier-Pierre JACQUOTTE
Office National d'Etudes et de Recherches Aérospatiales
Direction de l'Aérodynamique
92322 CHATILLON FRANCE

RESUME

Cet article décrit une méthode qui a été développée pour la construction de maillages autour de configurations complexes. Cette méthode permet tout d'abord une génération algébrique de type multidomaine reposant sur une partition du domaine de calcul en sous-domaines hexaédriques, chacun de ces sous-domaines étant maillé de manière structurée; une méthode d'optimisation est ensuite utilisée pour améliorer les qualités du maillage et en particulier ses propriétés métriques, ou pour l'adapter à la solution physique que l'on désire calculer sur ce maillage. La combinaison de ces deux méthodes, multidomaine et variationnelle, est en cours de mise en oeuvre et les premiers résultats obtenus sont présentés.

INTRODUCTION

Depuis plusieurs années, avec l'arrivée des super-calculateurs, des progrès remarquables ont été observés dans les calculs d'écoulements autour de configurations tridimensionnelles. Ces progrès concernent à la fois la complexité des modèles mathématiques utilisés et la complexité des formes géométriques. Les différents modèles mathématiques vont des méthodes de petites perturbations, linéarisées ou transsoniques, aux équations de Navier-Stokes et aux modèles de couplage fluide parfait- fluide visqueux en passant par les équations du potentiel et d'Euler. Par exemple, pour un avion de transport, les configurations étudiées sont passées progressivement d'une voilure isolée à l'avion complet avec les adjonctions successives du fuselage, de nacelles et de mâts. La construction de maillages autour de telles configurations est devenue une tâche de plus en plus compliquée et cruciale pour obtenir de bons résultats.

De nombreuses stratégies ont été développées d'une part pour obtenir ces maillages, et d'autre part pour pouvoir les utiliser dans les codes de calculs existants ou en cours de développement. Plusieurs étapes peuvent cependant être distinguées dans le processus de construction de maillage: une première étape de génération consiste à remplir l'espace tridimensionnel de noeuds de maillage reliés entre eux par un réseau d'arêtes permettant d'identifier chacun de leur voisin (connectique du maillage) et de définir des volumes élémentaires appelés cellules, mailles ou éléments selon les auteurs ou les codes de résolution utilisant ces maillages. De nombreuses classes de maillage peuvent être distinguées: ces maillages peut être de structure régulière ou non, de type multidomaine avec ou sans recouvrement, avec ou sans correspondance des noeuds aux interfaces entre les domaines ... Une deuxième étape consiste à s'assurer que le maillage obtenu possède les propriétés métriques nécessaires pour le calcul de l'écoulement. Ceci peut être a priori réalisé par certaines méthodes de génération ou peut être obtenu a posteriori par déplacement de certains noeuds du maillage initial sans en changer la topologie. Une troisième étape consiste, après un calcul d'écoulement sur un maillage initial, à construire un nouveau maillage mieux adapté à la solution à calculer. Ceci peut être effectué par ajout ou supression de noeuds du maillage avec changement de la topologie, ou par déplacement des noeuds, en conservant la même structure, de manière à resserrer les noeuds dans les régions désirées.

Nous présentons dans cet article une méthode développée pour réaliser certains de ces objectifs. La première partie est consacrée à la méthode de génération choisie qui permet de construire un maillage algébrique multidomaine par bloc structuré sans recouvrement, avec coincidence des noeuds aux interfaces. Nous indiquons dans la deuxième partie comment la structure multidomaine peut pratiquement être construite et comment la géométrie est prise en compte et le maillage construit. Nous rappelons enfin dans la troisième partie les principales caractéristiques de la méthode variationnelle qui a été développée pour l'optimisation et l'adaptation des maillages algébriques précédemment obtenus. Chacune de ces parties est illustrée d'exemples

STRUCTURE MULTIDOMAINE DU MAILLAGE

Nous présentons ici les caractéristiques de la méthode multidomaine développée: celle-ci consiste à tirer parti de la simplification procurée par une numérotation à trois indices "i, j, k" des noeuds du maillages où le triplet (i, j, k) parcourt la boîte $[1,im]$ x $[1,jm]$ x $[1,km]$ de $\mathbb{N}^3$. Cependant un tel ensemble de points constituant un bloc est rarement assez général pour prendre en compte toute la complexité du domaine. Une idée intuitive permettant d'obtenir un maillage complet consiste à diviser de manière arbitraire le domaine en plusieurs sous-domaines hexaédriques et à construire dans chacun d'eux un tel bloc "i, j, k". On parlera alors d'approche multibloc ou multidomaine. Cependant différentes décompositions sont possibles: ces sous-domaines peuvent se recouvrir ou avoir des intersections de volume nul, et, dans ce dernier cas, les noeuds situés de chaque côté des interfaces de deux blocs peuvent être distincts ou au contraire se correspondre. C'est cette dernière approche, compatible avec la Méthode des Eléments Finis, qui est considérée ici. A quelques différences près, cette méthode est d'ailleurs très souvent rencontrée dans la littérature consacrée aux maillages et est la plus fréquemment utilisée [9]. Différentes étapes sont considérées et seront décrites ci-dessous: partition de type "Eléments Finis" du domaine en blocs, maillage "i, j, k" des blocs, regroupement structuré des blocs en sous-domaines de calcul.

Partition du domaine en blocs

Une première partition du domaine de calcul Ω de forme arbitraire en blocs est considérée. Ces blocs sont des hexaèdres curvilignes isomorphes au cube unité $[0,1]$ x $[0,1]$ x $[0,1]$: ils sont caractérisés par 8 sommets, 12 arêtes et 6 faces curvilignes. On suppose que cette partition entre les blocs est de type "Eléments Finis", c'est à dire que:

1) l'intersection de deux blocs dans une telle partition ne peut être constituée que d'un sommet, ou d'une arête complète ou d'une face complète.

2) deux sommets reliés par une arête ne peuvent définir que cette arête et quatre arêtes constituant les côtés d'une face ne peuvent définir que cette face.

Plusieurs caractéristiques du maillage peuvent être déduites de ce choix de partition: en particulier la topologie générale du maillage est entièrement connue dès lors que l'on se donne le nombre de blocs et, pour chaque bloc, ses 8 sommets. Il est en effet possible de connaître:

- le nombre de blocs (donné), de faces, d'arêtes, de sommets;
- pour chacun des blocs: ses 8 sommets (donnés), ses 12 arêtes, ses 6 faces;
- pour chacune des faces: ses 4 sommets ou coins, ses 4 arêtes ou côtés;
- pour chacune des arêtes: ses 2 sommets ou extrémités.

Il est alors possible de numéroter ces entités. Réciproquement, ayant obtenu cette topologie, il est possible de dire si 8 (respectivement 4, 2) sommets donnés définissent un bloc (resp. une face, une arête) et le cas échéant de déterminer lequel (resp. laquelle). Il est alors intéressant de tirer parti de ces propriétés et en particulier du fait que toutes les entités - bloc, face, arête (et plus loin sous-domaine) - peuvent être désignées de manière univoque par les 8, 4, 2 (et plus loin IM x JM x KM) sommets qui les définissent.

Dans chacun de ces blocs hexaédriques, on peut alors obtenir un maillage structuré "i, j, k" par des techniques variées. On parlera alors de blocs de maillage Ω_{mail}. Les éléments finis seront dénotés par Ω_{ijk}. Notons qu'une des caractéristiques de ces blocs de maillages est la valeur des nombres de noeuds sur les 3 familles de 4 arêtes opposées.

Regroupement des blocs de maillage en sous-domaines

La partition de type " Eléments Finis " précédemment décrite pour les blocs de maillages est relativement contraignante et le nombre de blocs augmente très rapidement avec la complexité du domaine. Il peut donc être décidé de regrouper ces blocs en sous-domaines Ω_{calc} sur lesquels les calculs devront être effectués. Afin d'assurer la même structure (i, j, k) à ces sous-domaines, il est nécessaire d'effectuer aussi ce regroupement de manière structurée, c'est à dire de considérer un sous-domaine comme formé de l'empilement dans les 3 directions de $(IM-1) \times (JM-1) \times (KM-1)$ blocs hexaédriques, chaque bloc pouvant être défini par exemple par ses 8 sommets:

$$\{ S_{i,j,k} , i = i, i+1 ; j = j, j+1 ; k = k, k+1 \} \quad \text{pour } i = 1, IM-1 ; j = 1, JM-1 ; k = 1, KM-1.$$

Les sous-domaines sont donc également des hexaèdres et comportent 6 "super-faces" (unions "I, J" des faces élémentaires), 12 "super-arêtes" (unions d'arêtes élémentaires) et 8 sommets. La partition du domaine Ω en sous-domaines de calcul est alors devenue arbitraire.

Différentes partitions du domaine

Les quatre niveaux de maillage $(\Omega, \Omega_{calc}, \Omega_{mail}, \Omega_{ijk})$ se répartissent ainsi:

- Partition de type "Eléments Finis" du domaine en blocs de maillage: $\Omega = \bigcup_{type\ E.F.} \Omega_{mail}$

- Partition arbitraire du domaine en sous-domaines de calcul: $\Omega = \bigcup_{arb.} \Omega_{calc}$

- Regroupement " I, J, K" des blocs en sous-domaines de calcul: $\Omega_{calc} = \bigcup_{IJK} \Omega_{mail}$

- Maillage " i, j, k" des sous-domaines et des blocs: $\Omega_{mail} = \bigcup_{ijk} \Omega_{ijk}$ et $\Omega_{calc} = \bigcup_{ijk} \Omega_{ijk}$

Notion de famille d'arêtes

Les deux caractéristiques principales du maillage sont d'une part la structure "i, j, k" des blocs et d'autre part la correspondance des noeuds de maillage sur les interfaces entre les blocs. Elles permettent de dégager la notion de famille d'arêtes en remarquant tout d'abord que si deux arêtes sont en vis à vis sur une même face, elle ont nécessairement le même nombre de noeuds, puis que cette propriété se transmet de face en face dans les différents blocs du maillage. On dira donc que deux arêtes appartiennent à une même famille s'il existe une face telle que ces deux arêtes sont opposées sur cette face, ou s'il existe une suite de faces permettant de proche en proche de relier ces deux arêtes par des arêtes de la même famille.

Cette notion permet de transmettre d'autres informations que le nombre de noeuds en particulier la répartition des noeuds sur les arêtes d'une même famille. Cette propriété sera utilisée lors de la construction algébrique du maillage.

CONSTRUCTION DU MAILLAGE MULTIDOMAINE

Construction de la topologie et prise en compte de la géométrie

Une question qui apparaît immédiatement après cette description topologique est la construction pratique de la topologie et plus précisément la donnée des huit sommets de chacun des blocs et, pour chaque sous-domaine, la donnée des blocs les constituant. Remarquons tout d'abord que la structure topologique précédemment décrite est totalement indépendante de la géométrie du domaine dans lequel on désire construire le maillage: en effet, par exemple en 2 dimensions, l'union de trois blocs peut être utilisée pour des configurations aussi variées qu'un maillage en H dans un canal inter-aube ou qu'un maillage en C autour d'un profil isolé (fig.1). Inversement, un domaine peut être maillé selon des topologies différentes (cf. 1er exemple) ou encore, un même maillage multidomaine peut être construit en utilisant des découpages différents (fig.2) Ce découpage du domaine est facile en deux dimensions car il est possible de tracer le domaine, les différents blocs et les sous-domaines et plusieurs topologies peuvent facilement être définies et dessinées pour une configuration complexes. Cette opération de conception et de représentation de volumes géométriques complexes devient très difficile en trois dimensions et nécessite l'utilisation d'un outil interactif de visualisation tridimensionnelle. Celui-ci peut en particulier être un logiciel de CAO fonctionnant sur une station de travail. L'utilisation d'un tel logiciel revêt encore plus d'intérêt quand on remarque que les géométries modernes complexes sont, soit issues de la CAO, soit devront être traitées par la CAO après avoir été déterminées par le calcul. Ainsi les géométries soumises à l'ingénieur-numéricien lui sont données sous forme de fichiers CAO et c'est un fichier de ce type qui doit servir de point de départ pour la génération du maillage. Finalement, notons que dans les logiciels de CAO modernes, de nombreux modules ont fait l'objet de développement poussés et peuvent être directement utilisés lors de la construction du maillage: modélisation et conception géométrique et volumique, interpolation et paramétrisation, manipulation et intersection de surfaces...

Le code qui a permis d'obtenir les résultats présentés plus loin n'utilise pas encore les atouts d'un tel logiciel; il permet cependant d'étudier une topologie multidomaine telle qu'elle a été décrite et de construire un maillage à partir de maillages partiels de surfaces et d'arêtes. Les données géométriques consistent uniquement en certaines faces (respectivement arêtes) données sous forme de grilles de $im \times jm$ points (resp. suites de im points) Ainsi les données nécessaires à une configuration simplifiée constituée d'un demi ensemble fuselage-voilure dans un hémisphère sont représentées sur la figure 5: elles sont constituées de 3 grilles définissant le fuselage (et en particulier l'emplanture de l'aile), d'une suite de points définissant l'extrémité de l'aile, ainsi que de certaines arêtes définissant la frontière du domaine.

Ces données géométriques imposées par la CAO ne sont en général pas suffisantes pour construire le maillage: les blocs, afin d'être maillés, doivent être définis par leurs six faces qui peuvent être, soit imposées a priori par la CAO, soit internes au domaine et non nécessairement déterminées. De même il peut être

nécessaire de construire certaines arêtes ou de définir certains sommets intérieurs au domaine ou sur la surface externe de celui-ci. Nous désirons dans une première étape construire algébriquement, rapidement et aussi interactivement que possible un premier maillage. Nous avons choisi de construire un maillage en agissant le plus possible sur le "squelette" multidomaine de celui-ci lors de la construction des sommets et des arêtes des différents blocs: ainsi, un déplacement des sommets, ou un changement de la forme d'une arête permettent de modeler les blocs et procurent un premier moyen souple de contrôler la qualité du maillage. Ceci peut être réalisé en autorisant les quelques manipulations simples envisagées:

- création d'un sommet,
- création d'une arête par la donnée de deux sommets et de deux tangentes ou d'un point intermédiaire,
- déplacement d'un sommet,
- modification de la forme d'une arête.

Ainsi il est possible, comme en deux dimensions, de construire la topologie en modelant les domaines tout en tenant compte de la géométrie imposée. Ces procédures deviennent très utiles dès lors qu'elles peuvent être mise en oeuvre interactivement et graphiquement.

Maillage algébrique

Disposant ainsi d'un squelette multidomaine, il reste à placer les noeuds sur les différentes entités (arêtes, faces et blocs). Rappelons que le nombre de noeuds est déterminé dès lors qu'il est fixé sur un représentant de chaque famille d'arêtes. Tout comme l'étape précédente de définition du squelette, nous choisissons de porter l'essentiel de l'effort sur la répartition des noeuds sur les arêtes dont nous connaissons à ce stade la géométrie: les maillages bi- et tridimensionnels seront ensuite obtenus par interpolation à l'intérieur des entités à partir des valeurs aux bords, avec respect éventuel d'une géométrie.

Les répartitions linéiques peuvent tout d'abord être définies de manière absolue, sans faire référence à aucune autre répartition; dans ce cas on pourra imposer le rapport des deux mailles extrêmes, la répartition sera alors géométrique, ou on pourra imposer la longueur de la première et (ou) de la dernière maille, la répartition sera alors cubique (ou de degré inférieur). Notons que cette dernière manière de définir la répartition doit être utilisée avec précaution car le résultat dépend fortement du nombre de mailles sur l'arête: pour un paramétrage de 0 à 1, une première maille de 1/10ème crée avec 5 mailles un raffinement au voisinage de 0, tandis que le raffinement se trouve au voisinage de 1 avec 20 mailles. Il est préférable de définir des raffinements relatifs qui permettront, par simple changement du nombre de noeuds sur chaque famille, d'obtenir un maillage de même aspect avec la densité de points voulue.

Les répartitions peuvent aussi être définies en référence à une autre arête; deux cas sont particulièrement intéressants et impliquent le transfert d'un certain type d'information d'une arête à une autre. Ayant déterminé une répartition sur une arête par un moyen quelconque, il est souvent utile de prescrire sur l'arête opposée d'une face le même répartition à une homothétie près (même répartition en abscisse réduite) et ainsi transférer cette répartition sur d'autre faces; les arêtes en question appartiennent à la même famille par définition et ont donc le même nombre de noeuds: la construction du maillage sur ces arêtes ne posera donc pas de difficulté et pourra ainsi être envisagé par transfert de répartition sur une famille d'arête. Un autre moyen de définir une répartition est obtenu en remarquant qu'à l'interface entre deux blocs adjacents, il est souvent nécessaire d'imposer des tailles de mailles voisines; lors de la construction du maillage du squelette, cela nécessite en particulier que les longueurs de mailles de part et d'autre d'un sommet soient voisines. Cela fournit un autre moyen de définir une répartition (de type cubique ou de degré inférieur) par transfert d'une information (une taille de maille) d'une arête à une autre à travers un sommet.

Exemples d'illustration

Ces différents types de construction pour la topologie, le squelette multidomaine et le maillage du squelette sont en cours de mise en oeuvre; ces étapes nous fournissent trois niveaux de contrôle lors de la construction du maillage. Les résultats présentés ont cependant été obtenus avec la méthode la plus simple qui consiste à construire entre deux sommets une répartition uniforme de noeuds. Nous rappelons qu'il est important de concevoir un code qui puisse construire le maillage à partir de données de trois types indépendants - géométrie, topologie, répartition linéique - de manière à pouvoir envisager sans trop de modifications des maillages complètement différents.

Le premier exemple illustre cet aspect du problème. Il s'agit de construire un maillage autour d'une nacelle simplifiée axisymétrique creuse ayant été obtenue par rotation d'un profil NACA0012 autour d'un axe, et dont le maillage est une donnée. Plusieurs topologies sont envisagées et les maillages correspondants sont obtenus par simple changement d'un fichier de topologie. On suppose d'une part que

l'on désire obtenir un maillage en H dans ce plan méridien (fig.3). D'autre part, en ce qui concerne les plans radiaux, on considère différentes combinaisons de topologies pour l'intérieur et l'extérieur de la nacelle: maillage en H ou en O. Nous montrons sur la figure 4 les maillages sur certaines faces de sous-domaines obtenus pour les topologies suivantes:

- Topologie 1 : le sous-domaine intérieur est maillé en H et les deux sous-domaines extérieurs sont maillés en O (fig.4a),
- Topologie 2 : le sous-domaine intérieur est maillé en O et les deux sous-domaines extérieurs sont maillés en H (fig.4b).

Le second exemple montre comment il est possible d'obtenir un maillage autour d'une configuration simplifiée (fuselage + voilure) d'un demi avion de transport. La géométrie peut être définie comme suit (fig.5):

- le plan de symétrie de l'avion est le plan xOz,
- la direction du fuselage suit l'axe Ox,
- l'aile est au voisinage du plan xOy et est définie par son emplanture sur le fuselage d'une part, et par son profil d'extrémité d'épaisseur nulle.
- la frontière du domaine donnée par des arêtes construites sur l'hémisphère extérieur.

On a défini 40 blocs s'appuyant sur 8 faces ou unions de faces données sur le fuselage et 31 arêtes ou unions d'arêtes tracées sur une sphère. Les blocs sont regroupés en 3 sous-domaines: un sous-domaine en O autour du fuselage et des deux autres sous-domaines (extrados et intrados) contenant l'aile. Le tracé des arêtes des 3 sous-domaines (fig.7) permet de vérifier la bonne prise en compte de la topologie. Les maillages de certains couples de faces opposées des différents sous-domaines sont montrés sur les figures 7a, b et c. Celles-ci permettent de vérifier le bon fonctionnement du code, mais laissent percevoir la nécessité de mettre en oeuvre les procédures plus générales évoquées plus haut pour la construction géométrique des arêtes et la génération des maillages sur celles-ci, ou pour l'optimisation de maillages par des méthodes variationnelles telles celles décrites dans la partie suivante.

OPTIMISATION ET ADAPTATION VARIATIONNELLE

De nombreuses méthodes variationnelles existent [2, 3] pour obtenir des maillages possédant les qualités de régularité et d'orthogonalité recommandées pour les calculs aérodynamiques. Celles-ci reposent pour la plupart sur des idées intuitives et empiriques qui consistent à considérer le maillage comme un treillis de points reliés par des ressorts et des barres de torsions assurant la régularité et l'orthogonalité. Bien qu'un modèle mécanique soit sous-jacent à ces méthodes, celles-ci ne sont pas entièrement satisfaisantes pour de nombreuses raisons [5]. Nous rappelons ici les principales caractéristiques d'une méthode variationnelle développée pour l'optimisation et l'adaptation de maillages structurés bi- et tridimensionnels. Un modèle mécanique est là encore sous-jacent, mais fait référence à la mécanique des milieux continus par opposition aux méthodes précédemment évoquées. Une mesure de la déformation des mailles peut être définie et permet de quantifier le qualité du maillage; cette quantité est alors optimisée par déplacement des noeuds dans le domaine. Un terme de contrôle de volume apparaît naturellement dans les fonctionnelles obtenues, et est utilisé, conjointement avec la mesure de la déformation, pour l'adaptation du maillage avec contrôle de la déformation des mailles. Plusieurs exemples de maillages optimisés ou adaptés sont présentés et illustrent les possibilités de la méthode.

Une mesure de la déformation du maillage

Nous considérons la méthode de maillage comme la discrétisation d'un problème continu qui consiste à trouver une transformation $x(\xi)$ du cube unité de référence (espace $\xi=(\xi,\eta,\varsigma)$) dans le domaine à mailler (espace $x=(x,y,z)$). Pour cela nous raisonnons en considérant la déformation d'une maille élémentaire cubique en une cellule arbitraire. A partir de quatre axiomes et propriétés [5], il est possible de démontrer qu'une mesure correcte σ de la déformation du cube unité de référence dans une maille courante ne dépend que des invariants I_1, I_2, I_3 du tenseur des déformations C associé à transformation $x(\xi)$:

$$\sigma = \sigma (I_1 , I_2 , I_3)$$

où

$$I_1 = \text{tr } C , \quad I_2 = \text{tr Cof } C , \quad I_3 = \text{det } C$$

avec

$$C = \nabla x^t . \nabla x$$

On impose en outre que σ dépende du sens d'orientation de la cellule; le troisième invariant I_3 n'étant pas sensible à cette orientation, il est alors nécessaire de le remplacer dans l'expression de σ par:

$$J = \det \nabla \mathbf{x}$$

Ainsi, on a:

$$\sigma = \sigma\left(I_1 , I_2 , J \right)$$

La deuxième étape consiste à s'assurer que la minimisation de la fonctionnelle σ est un problème bien posé. Ceci conduit à poser des hypothèses sur σ et sur ses dérivées premières et secondes pour des transformations dites rigides qui conservent la forme de la cellule. Ces transformations vérifient les quatre propriétés équivalentes suivantes:

i) $\mathbf{x}(\boldsymbol{\xi})$ est une transformation rigide ii) $\nabla \mathbf{x}$ est une matrice orthogonale directe

iii) $\mathbf{C} = \mathbf{Id}$ et $J = 1$ iv) $\left(I_1 , I_2 , J \right) = \left(3 , 3 , 1 \right)$

Pour ces transformations, il n'y a pas déformation de la cellule σ est stationnaire. Nous supposons d'autre part que la fonctionnelle est convexe au voisinage des transformations rigides: ces conditions assurent les bonnes propriétés mathématiques au problème et la certitude que les algorithmes usuels de minimisation convergeront efficacement vers une solution unique. Il est alors possible de caractériser la fonctionnelle et d'exhiber des fonctions simples (polynômiales) des invariants [5].

- En 2 dimensions, on considère la fonction:

$$\sigma_{2d} = C\left(I_1 - 2 J \right) + K\left(J - 1 \right)^2$$

Cette expression fait apparaître deux termes précédés de constantes positives. Le second terme, $\left(J - 1 \right)^2$, peut être interprété comme un terme de pénalité, interdisant au volume V de la maille de s'éloigner d'une valeur prescrite de référence V_{Ref} ($J = V/V_{Ref}$), et donc prohibant le déversement des mailles. Le premier terme, $\left(I_1 - 2 J \right)$, peut quant à lui être interprété comme une formulation "moindres carrés" des relations de Cauchy-Riemann qui assurent la conformité du maillage:

$$x_\xi - y_\eta = 0 \quad \text{et} \quad x_\eta + y_\xi = 0$$

Cette fonctionnelle peut aussi être utilisée pour l'optimisation de maillages sur des surfaces gauches [6].

- En 3 dimensions, on considère la fonction:

$$\sigma_{3d} = C\left(I_1 + I_2 - 6 J \right) + K\left(J - 1 \right)^2$$

Cette expression peut être interprété comme une formulation de type "moindres carrés" des propriétés (i-iv). En effet, celles-ci sont équivalentes à:

v) $\mathbf{F} = \mathrm{Cof}\, \mathbf{F}$ et $\det \mathbf{F} = +1$

ce qui conduit directement à l'interprétation annoncée en remarquant que:

$$\| \mathbf{F} - \mathrm{Cof}\, \mathbf{F} \|^2 = I_1 + I_2 - 6 J$$

Outre l'interprétation précédemment donnée pour $\left(J - 1 \right)^2$, on note ici que ce terme complète le premier en imposant que $\mathbf{F}$ soit une matrice orthogonale *directe* ($\det \mathbf{F} = +1$).

- Pratiquement, la méthode ne peut être utilisée telle qu'elle a été décrite, en mesurant la déformation des cellules par rapport au carré ou au cube unité: il paraît en particulier nécessaire de pouvoir imposer des raffinements dans certaines régions du domaine, ou plus généralement de construire une fonctionnelle à l'échelle du domaine. Cela est effectué en prenant comme référence un parallélépipède de côtés a, b et c (soit $0 \le \xi \le a$, $0 \le \eta \le b$, $0 \le \zeta \le c$), où ces coefficients dépendent de la maille et peuvent être choisis par l'utilisateur. Pour chaque maille ou élément, il est donc possible de définir une contribution élémentaire σ^e mesurant sa déformation; celle-ci s'exprime comme une fonction des coordonnées des noeuds de l'élément. Par sommation de ces contributions, on construit une quantité globale Σ mesurant la qualité de déformation du maillage; le maillage est alors obtenu par minimisation de Σ, fonction des coordonnées des noeuds de l'ensemble du maillage.

D'un point de vue numérique, la fonctionnelle s'exprime en fonction des coordonnées des noeuds du maillage comme des polynômes de degré $2N$ (N désignant la dimension de l'espace). Pour la minimisation, on utilise un algorithme de gradient conjugué dans lequel l'étape de descente et la recherche

unidimensionnelle d'un minimum conduit à l'annulation d'un polynôme de degré $2N-1$. Il a bien été vérifié que la propriété de convexité est nécessaire à l'unicité d'une racine réelle pour ce polynôme et donc au bon déroulement de l'algorithme.

Adaptation du maillage

Les fonctionnelles mises en évidence en 2 et 3 dimensions ont en commun de faire apparaître un terme de contrôle de volume de la maille:

$$\sigma_{vol} = K (J - 1)^2$$

Il paraît intéressant d'utiliser ce terme pour adapter le maillage à un phénomène physique étudié, c'est à dire pour raffiner ou appauvrir le maillage dans certaines régions. Ceci peut être fait après le calcul d'une solution obtenue dans un maillage initial, ou itérativement au cours du processus de calcul de la solution. L'adaptation consiste donc à déplacer les noeuds en les resserrant dans certaines régions. Cela est effectué par minimisation des fonctionnelles introduites où l'on a remplacé σ_{vol} par σ_{adapt} avec

$$\sigma_{adapt} = K (\omega J - 1)^2$$

Dans cette expression, ω est un poids qui peut être calculé en fonction d'estimations a posteriori d'erreur afin de minimiser celle-ci dans le domaine. Si on ne dispose pas de telles estimations, le coefficient ω peut être obtenu comme une fonction d'une quantité physique ou de son gradient, par exemple la pression, le nombre de Mach ou l'entropie. C'est cette dernière approche qui a été testée et dont nous montrons les résultats dans le paragraphe suivant.

Afin de garantir les bonnes propriétés métriques du maillage, et contrôler la déformation des cellules au cours de l'adaptation, il est nécessaire de faire intervenir le poids dans le premier terme des fonctionnelles, et de faire un choix approprié pour la cellule de référence. Considérons une cellule dans le maillage initial; cette cellule peut être approchée par un parallélépipède de côtés a, b, et c. La cellule de référence recherchée a un volume égal à ωabc, mais il faut répartir le poids ω entre les trois côtés. Cela est fait en faisant intervenir les cosinus directeurs n_1, n_2 et n_3 du gradient de la quantité physique, calculés par rapport au trièdre formé par les vecteurs $\mathbf{a}$, $\mathbf{b}$ et $\mathbf{c}$; la cellule de référence choisie est un parallélépipède de côtés:

$$a \omega^{n_1^2}, \ b \omega^{n_2^2} \text{ et } c \omega^{n_3^2}.$$

Quand, par exemple, le gradient est parallèle au côté $\mathbf{a}$ ($n_1=1$, $n_2=n_3=0$), c'est dans cette direction que les plus grandes variations du champ physique ont lieu: par le moyen décrit, l'adaptation a donc tendance à raffiner dans cette direction. Le poids choisi ω s'écrit [8]:

$$\omega = \omega_0^{(1 - |\nabla u|^2)} . \ \omega_1^{|\nabla u|^2}$$

où $|\nabla u|$ désigne la valeur ramenée entre 0 et 1 du gradient d'une quantité physique choisie, et ω_0 et ω_1 sont deux constantes: l'une est choisie pour raffiner dans les régions de fort gradient ($\omega_1 \cdot 1$) et l'autre est ajustée de telle sorte que:

$$\int_\Omega \omega(x) \, dx = \int_\Omega dx$$

Résultats

Dans cette section, nous présentons des maillages qui ont été obtenus par la méthode précédemment décrite. Nous mentionnons tout d'abord certaines références [5-8] dans lesquelles la robustesse de la méthode était illustrée pour le cas bidimensionnel, en particulier sa faculté à restituer un maillage orthogonal à partir d'une initialisation tout à fait arbitraire (initialisation aléatoire); en 3 dimensions, la même propriété a été observée pour la fonctionnelle σ_{34}.

La figure 2a représente un domaine autour d'une aube de turbine. La topologie est dite H-C car elle présente deux sous-domaines : un sous-domaine en C autour du profil et un sous-domaine en H en amont. Le maillage présenté a été obtenu par minimisation de la fonctionnelle σ_{24} séparément dans chacun des sous-domaines. Cette minimisation permet, dans un premier temps, en laissant les noeuds mobiles sur les côtés des sous-domaines, de construire deux familles orthogonales; ces familles sont ensuite utilisées comme système de paramétrisation du domaine et permettent par interpolation de construire les lignes de maillage passant par les points voulus sur les frontières. Cette interpolation algébrique permet d'assurer l'orthogonalité aux frontières de chacun des sous-domaines, et en particulier la continuité de pente aux l'interfaces.

La méthode d'adaptation a été utilisée en deux [6] et trois dimensions [7] et a permis d'améliorer

15-8

sensiblement la qualité de solutions d'écoulements obtenues par résolution des équations d'Euler pour diverses configurations: l'adaptation permet d'atteindre avec un maillage moyen adapté une précision qui aurait nécessité un maillage fin non adapté. Nous présentons ici un exemple qui a été traité dans le cadre de la simulation numérique d'un essai en soufflerie, portant sur l'étude du pilotage en force du missile ASTER à très haute altitude [4]. L'interaction d'un jet latéral et d'un écoulement supersonique externe crée, en amont de la sortie du jet, un choc détaché, principal point d'intérêt de cette simulation avec sa réflexion sur la paroi de la soufflerie.

Un maillage initial est construit plan par plan avec les conventions suivantes:
- plan amont : k=1 et plan aval : k=85.
- surface du missile : j=1 et paroi de la soufflerie : j=49.
- plan de symétrie : i=1 et i=82 (dont le plan du jet : i=1).

La densité a été retenue comme paramètre d'adaptation, la résolution des équations d'Euler a été effectuée avec le code FLU3C [1] à l'Aérospatiale. La Figure 8 représente la topologie et le maillage dans un plan perpendiculaire à l'axe du missile (plan k=1) ainsi que les lignes iso-densité dans le plan du jet (plan i=1).

Le maillage a été optimisé et a été utilisé pour calculer une nouvelle solution (Fig.9). L'examen de celle-ci [8] a montré que l'adaptation permet de décrire le pied du choc avec plus de précision et en particulier laisse apparaître une discontinuité de contact non décelée sur le maillage initial; l'adaptation du maillage a également permis de mieux prédire la réflexion du choc sur la paroi de la soufflerie.

CONCLUSION

Une méthode pour la construction, l'optimisation et l'adaptation de maillages structurés tridimensionnels a été présentée. Celle-ci se caractérise par une première étape purement algébrique qui nécessite des moyens de calcul peu puissants, des possibilités graphiques importantes et une forte interaction de l'ingénieur; cette étape permet de construire, rapidement et interactivement, un premier maillage qui peut éventuellement supporter un calcul aérodynamique. La seconde étape variationnelle nécessite des moyens de calcul importants surtout en temps de calcul; elle est par contre automatisée et ne demande pas d'intervention de l'ingénieur. Le maillage obtenu à la suite de cette étape permet de calculer des solutions d'excellentes qualités.

Ces deux étapes sont encore découplées et un imporatnt travail reste à faire afin d'obtenir un outil général. Certains aspects n'ont d'autre part pas été évoqués dans cet article mais font l'objet d'approfondissement et de recherche: interface CAO-maillage, définition et traitement des surfaces, choix du critère d'adaptation.

REFERENCES

[1] M. BORREL, J-L. MONTAGNE, J. DIET, Ph. GUILLEN et J. LORDON, "Méthodes de calcul d'écoulements supersoniques autour de missiles tactiques à l'aide d'un schéma décentré" **La Recherche Aérospatiale**, No 1988-2, p.43-55, Mars-avril 1988.
[2] J.U.BLACKBILL et J.S.SALTZMANN, "Applications and generalizations of variational methods for generating Adaptive meshes", **Numerical Grid Generation.** J.F.Thomson Ed., Elsevier Science Publishing Co., Inc. 1982, pp.865-884.
[3] R.CARCAILLET, "Optimization of three-dimensional computational grids and generation of flow adaptive computational grids", **AIAA Paper** 86-0156, Reno, Nevada, USA, Jan.1986.
[4] M. DORMIEUX et C. MAHE, "Calculs Tridimensionnels d'interaction d'un jet latéral avec un écoulement supersonique externe", 62ème **Meeting AGARD** sur la validation du calcul en Dynamique des Fluides, 5 Mai 1988, Lisbonne, Portugal.
[5] O.-P. JACQUOTTE, "A Mechanical Model for a New Generation Method in Computational Fluid Dynamics", **Comp. Meth. Appl. Mech. Engrg.**, Vol.66, pp.323-338, 1988; aussi ONERA T.P. No 1988-48.
[6] O.-P. JACQUOTTE et J. CABELLO, "A Variational Method for the Optimization and Adaptation of Grids in Computational Fluid Dynamics", **Numerical Grid Generation in Computational Fluid Mechanics '88**, cf.[9]; aussi ONERA T.P. No 1988-174.
[7] O.-P. JACQUOTTE et J. CABELLO, "A New Variational Method for the Generation of Two- and

Three Dimensional Adapted Grids in Computational Fluid Dynamics", **Proceedings, 7th International Conference FINITE ELEMENT METHODS IN FLOWS PROBLEMS**, Huntsville, Alabama, USA, 3-7 Avril 1989.

[8] O.-P. JACQUOTTE et J. CABELLO, "Une Méthode de Construction de Maillages Tridimensionnels Fondée sur un Principe Variationnel", **La Recherche Aérospatiale**, à paraître, 1989.

[9] **Numerical Grid Generation in Computational Fluid Mechanics '88**, Ed. S. Sengupta, J. Haüser, P.R. Eiseman, J.F. Thompson, Pineridge Press, Proceedings, 2nd International Conference on Numerical Grid Generation in CFD, Miami-Beach, Florida, USA, 5-8 Décembre 1988;
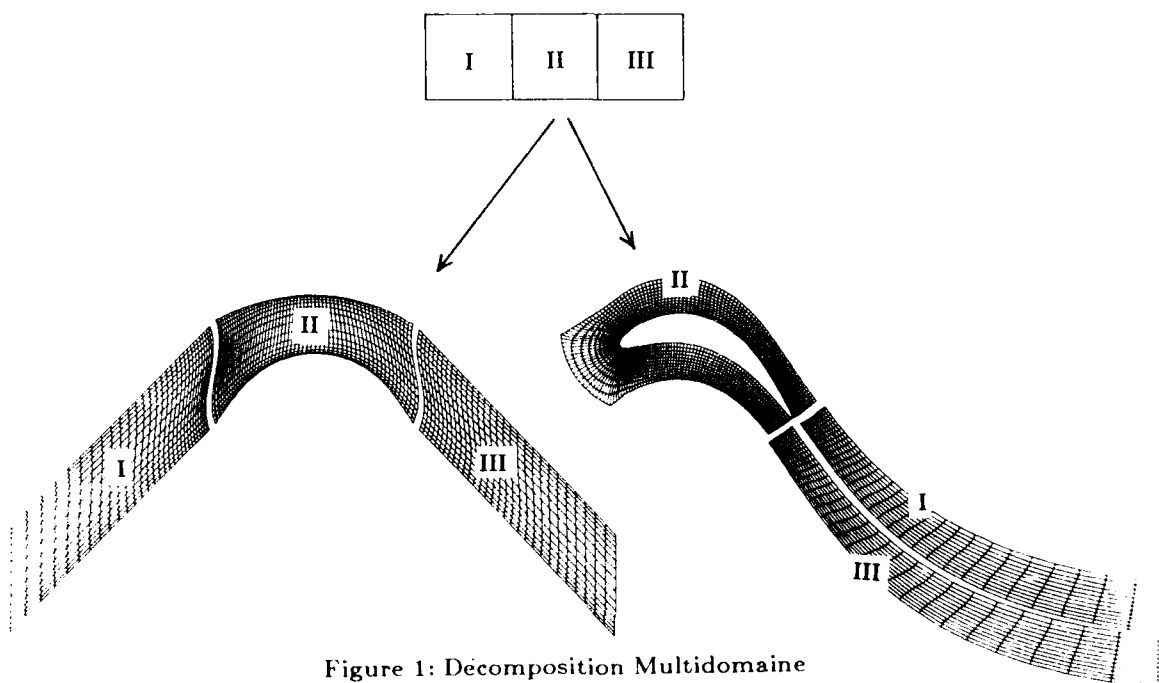
Figure 1: Décomposition Multidomaine
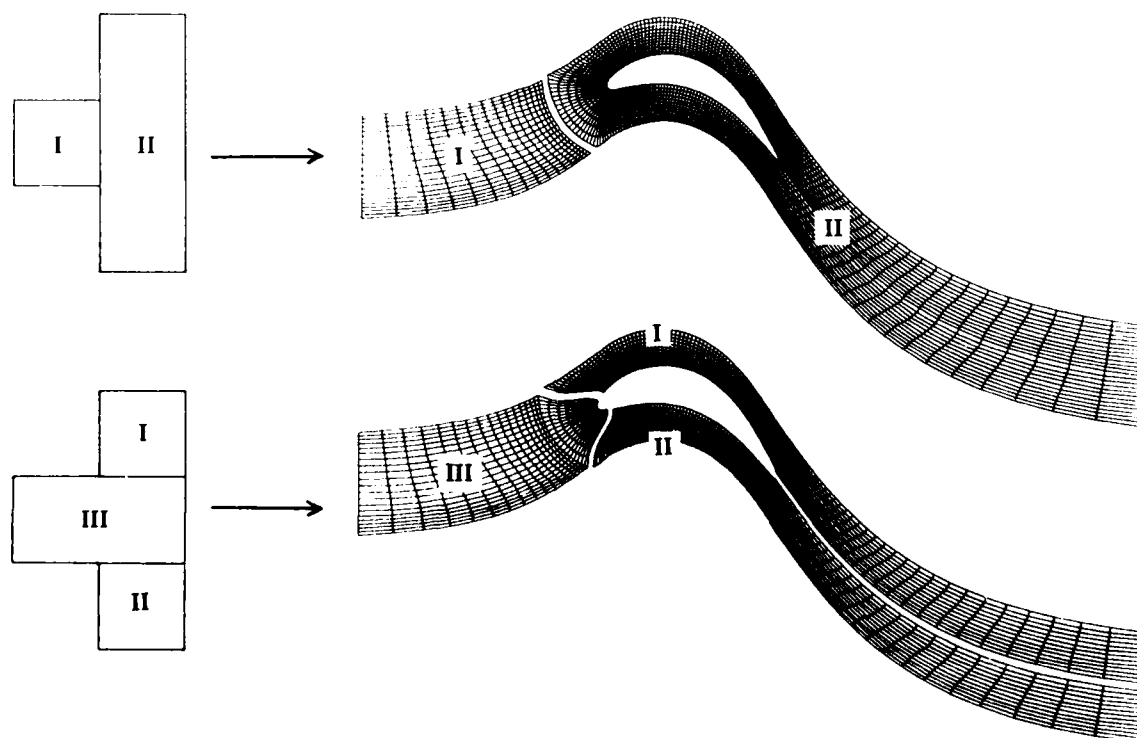
Figure 2: Non Unicité de la Décomposition

a) Plan Médirien

b) Plan Radial:
Topologie H-O

c) Plan Radial:
Topologie O-H
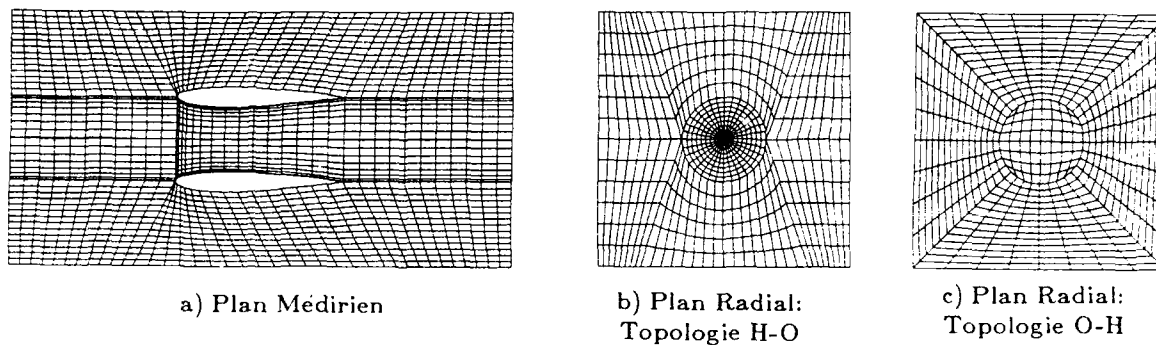
Figure 3: Topologies pour un Maillage autour d'une Nacelle Creuse

a) Topologie O-H

b) Topologie H-O
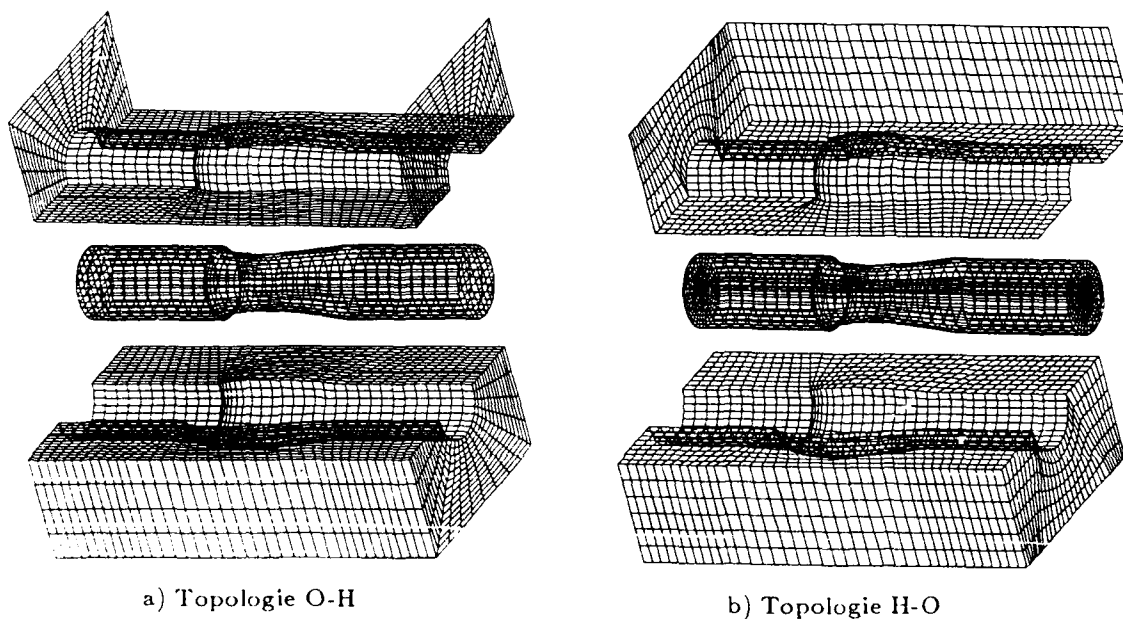
Figure 4: Maillages Multidomaines autour d'une Nacelle Creuse
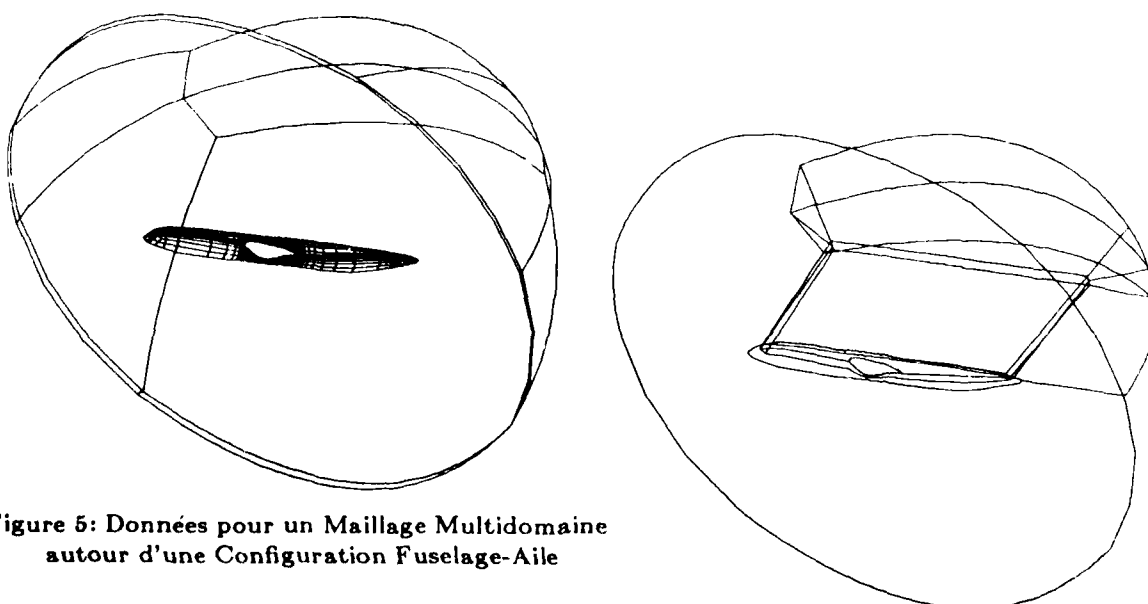
Figure 5: Données pour un Maillage Multidomaine
autour d'une Configuration Fuselage-Aile

Figure 8: Représentation Filaire du Maillage Multidomaine:
Arétes des Sous-Domaines

b) Sous-Domaine Extrados

c) Sous-Domaine Intrados
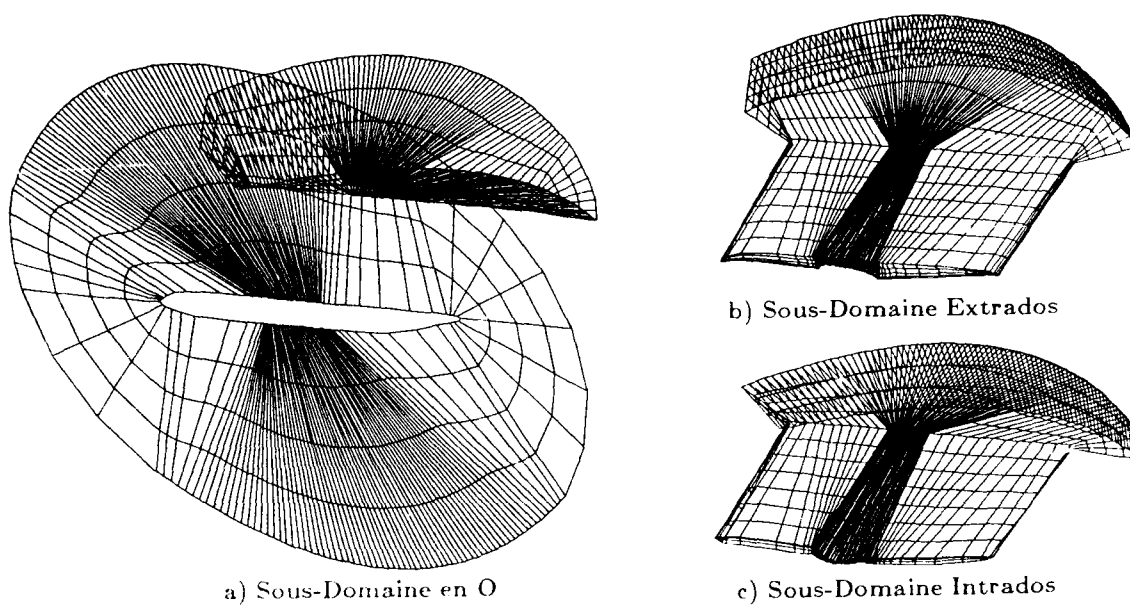
a) Sous-Domaine en O

Figure 7: Faces des Sous-Domaines

Sortie du jet
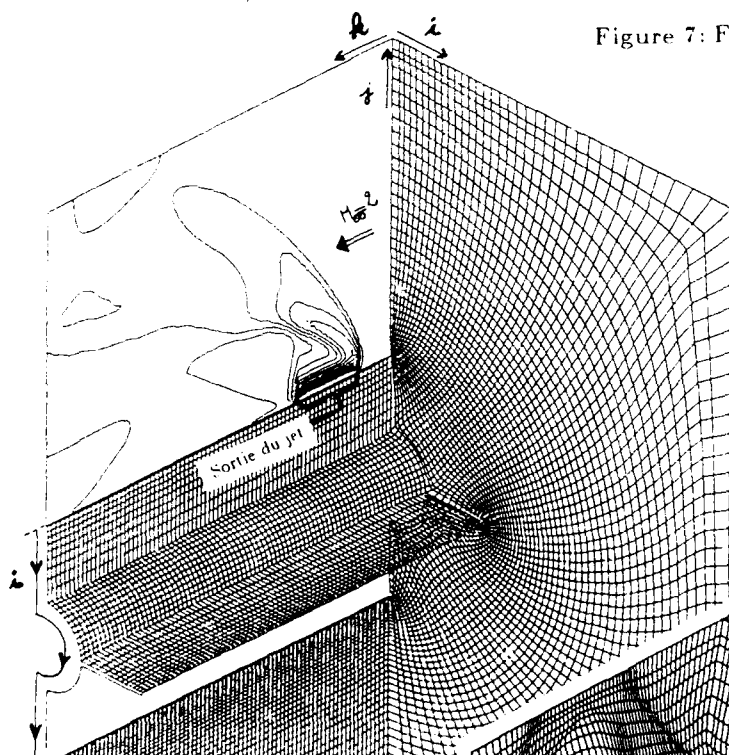
Figure 8:
Calcul autour du Missile ASTER:
Topologie, Maillage
et Solution Initiale
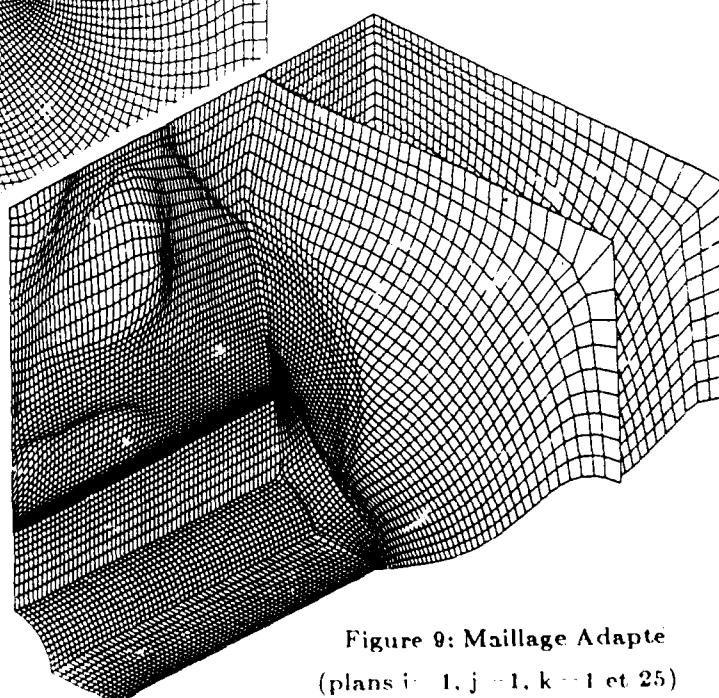
Figure 9: Maillage Adapté

(plans i 1, j 1, k 1 et 25)

A DISCUSSION ON ISSUES RELATING TO MULTIBLOCK GRID GENERATION

J M Georgala, J A Shaw
Aircraft Research Association Limited
Manton Lane, Bedford MK41 7PF, England

SUMMARY

In this paper, efforts aimed at bringing a multiblock grid generation system to the state of maturity necessary for practical use are discussed. Attention is focused upon the interrelated areas of topology generation and grid control. An algorithm for automatically decomposing a flow domain about an aircraft configuration into a component-adaptive topology is given. Two techniques for controlling the resulting grid topology are described. The first automatically produces *default grids*, which will generally be of an acceptable quality. The second is a user-friendly interactive grid editor which allows any deficiencies in the default grids to be rapidly identified and modified. The paper concludes with examples of the multiblock approach applied to a range of aircraft geometries.

1 INTRODUCTION

The analysis of transonic flows about realistic aircraft configurations is of considerable importance in the design of military and commercial aircraft. The importance of such a flow regime to practical aircraft design arises from the need to maintain a high aerodynamic efficiency in the speed regime where compressibility effects become substantial. For transport aircraft this is necessary to ensure efficient high speed cruise, whilst for combat aircraft, it may provide a useful extension to the manoeuvre flight envelope.

As the level of sophistication for mathematically modelling compressible flow has developed, and the effective cost of computing a given solution has reduced, there has been an increasing demand to introduce the use of the best currently available flow model into the design environment. To date, the solution techniques that have been devised for the higher order equations which govern fluid motion require the flow domain to be discretised into a set of points known collectively as a grid. In spite of this, developments in grid generation have continually lagged well behind progress in flow algorithm development. The extent of this can be seen by contrasting the maturity of Euler solvers[1,2] with the isolated reports of grids constructed about complete, powered aircraft.[3] If computational fluid mechanics is to fulfil its potential and become a major tool for use by an aerodynamicist at various stages of a project design, then *mesh generators need* to be brought to a level of maturity whereby it is possible to handle shapes of arbitrary generality with ease. Anything short of this will inevitably affect the long term application of any approach to modelling flow behaviour.

This paper focuses on efforts aimed at bringing a multiblock grid generation system to the state of maturity required for practical use[4,5]. Existing multiblock methods rely heavily on user-interaction. Complete automation of such systems is not feasible since each configuration invariably possesses distinct geometric characteristics necessitating individual attention. However, to ease the burden imposed on the non-specialist, a steady move is being made towards introducing automation to those elements for which such an approach is practical. An algorithm for automating the construction of component-adaptive grid topologies about a practical range of aerodynamic geometries is outlined. A second interfacing algorithm which drives the distribution of points on the boundaries and subsequently within the domain is discussed. Since a sizeable amount of user-interaction will always be necessary in the successful application of multiblock systems, an interactive surface grid editing facility has been designed as an aid to the non-specialist. This graphical mechanism for viewing and modifying grid deficiencies is described. The power of the multiblock technique, which is coupled with an Euler algorithm, is illustrated with grids and flow results for a variety of complex aircraft configurations.

2 GENERAL PRINCIPLES OF MULTIBLOCK GRID GENERATION

The multiblock approach to grid generation which uses the basic concept of block structured grids has been well documented.[6,7,8] Each component of a complex aircraft configuration favours its own natural type of grid structure. For example, a wing may favour a 'C' grid structure whilst for a fuselage a polar 'O' grid structure may be a more suitable choice. The philosophy behind our approach is to incorporate the most natural structures around each component within a global Cartesian structure. The flow domain is decomposed into a set of non-overlapping blocks, with the constraint that a single boundary condition type be associated with each face of each block. Each block maps to a cuboid in computational space. The arrangement of blocks with respect to each other defines the topology. For grid generation purposes, a Dirichlet (fixed) boundary condition is imposed at block faces which lie on the configuration or on the farfield boundary of the flow domain. The other block faces which lie within the interior of the domain are given a continuity condition. Grid points on such faces will be treated in the same manner as points within a block to ensure that grid lines pass smoothly between adjacent block boundaries.

A set of non-linear elliptic partial differential equations (based on the ideas of Thompson, Thames and Mastin[9]) are used to generate grid points within the flow domain. The equations are of the form

$$g^{ij} X_{\xi_i \xi_j} = -p^i X_{\xi_i} \tag{1}$$

where g^{ij} are the metric terms, p^i the control functions, X the physical grid point coordinates, with the tensor notation i,j taking the values 1, 2 and 3.

The geometry for each component of the configuration under consideration is defined separately by an arbitrary set of cross-sections of the component. The Coons bicubic patch technique[10] is used to obtain a continuous description of the surface of each component in terms of its own pair of parametric coordinates (non-dimensionalised surface distances (s,t) along and across the input cross-sections defining the geometry). The equations for a given bicubic patch are of the form

$$X = AMB^T \tag{2}$$

where $X = (x,y,z)$, $A = (s^3 \ s^2 \ s \ 1)$, $B = (t^3 \ t^2 \ t \ 1)$ and M is a matrix consisting of parametric derivatives of X and some blending functions. This continuous description of each surface enables the intersections between adjacent components to be determined using a Newton-Raphson technique.

Surface grids are generated in terms of the parametric representation of each component using the equivalent two-dimensional form of Eq (1). The two-dimensional topological structure associated with each component surface is derived automatically from the global grid topology. The method of Thomas and Middlecoff[11] is employed to control the distribution of grid points on the surfaces. This method of grid control will be discussed further in subsequent sections. Briefly, grid points are fixed in position along the Dirichlet boundaries of each grid, the control functions in equation (1) are determined on the boundaries, and are then interpolated throughout the topological structure of the grid. Thus, the distribution of points on the boundaries directly influences the positioning of grid points within the grid domain. The grids computed on the geometry and farfield boundary surfaces are then mapped back to physical space and used as fixed Dirichlet data for computing the grid point distribution throughout the flow domain.

3 TOPOLOGY CONSTRUCTION

The ideas upon which multiblock is based allow, for a given configuration, a wide variety of topological structures to be assembled. In practice, however, grid topology and grid control are closely related and a poor choice of topology may result in an inferior quality grid. Consequently, alternative grid topologies can have noticeably different effects on flowfield solutions. Isolating which is the most suitable topology for a given geometry is not an easy task. Furthermore, the generation of all of the necessary information for connecting blocks is a tedious job for even the most experienced method developers. Such an exercise is very time consuming and extremely difficult to visualise. As a further complication, a minor alteration to the geometry definition may necessitate major changes to the topology structure.

The information stored within the topology data file controls the grid generation and flow solution processes, and it is therefore vital that the data be set up efficiently and error free. The ability to view some or all of the stages followed in the construction of a suitable topology is essential. To encourage the use of the multiblock technique, it is necessary to steer efforts towards establishing a user-friendly environment in which the topology information can be readily defined and the resulting block structure examined. This requirement has led to the development of a topology generation algorithm[7,8] which has absorbed much of the necessary expertise, leaving the user free from the responsibility of specifying block connectivity data. Component-adaptive topologies can be generated quickly (typically five minutes for the examples shown) to ease the search for the most suitable topology. As the topology generation algorithm decomposes the domain it stores data relating to the relative locations of the individual components within the topological structure. This information controls the boundary grid point spacings which would otherwise require considerable user effort if defined interactively.

3.1 Topology Generation Algorithm

The topology algorithm follows three fundamental steps. Firstly, the domain about a representative schematic of the configuration is split up to define a Cartesian 'H' block structure. As discussed earlier, this decomposition is bound by the constraint that one boundary condition be applied at each block face. Secondly, the layers of blocks lying either side of the individual components are split in two, forming two layers, to enable local grid structures to be embedded around each component. Finally, new blocks are added to the global Cartesian framework as required to create the appropriate 'C' or 'O' structures around each component.

This process is illustrated here in two dimensions (Figure 1). Given an aerofoil AA' within a finite rectangular domain BCDE (Figure 1a), consider the mapping to a computational domain (Figure 1b) with coordinate system (ξ,ζ). The aerofoil profile maps to a horizontal slit AA'. The domain is then subdivided into a Cartesian topology allowing for a single boundary condition at each side of each block. For this case the domain decomposes to a minimum of six blocks (Figure 1c). The layers of blocks lying above and below the aerofoil schematic are split (from H to H' and from G to G') to give twelve blocks (Figure 1d). The addition of two blocks at the leading edge of the slit (at A) produces an embedded 'C' grid structure (Figure 1e). By adding two further blocks at the trailing edge (A'), an 'O' grid structure (Figure 1f) is obtained.

This process transforms readily to a wing, with a Cartesian modelling of the tip, by stacking either of the structures in Figures 1e and 1f in the spanwise direction. Three-dimensional topologies are, however, more usually defined by combining these structures as appropriate.

3.2 Automatic Topology Generation in Three Dimensions

A truly three-dimensional application of the topology generation algorithm is illustrated using the wing-fuselage-three pylon-store configuration shown in Figure 2a. A basic schematic representation of the configuration (Figure 2b) is the only geometric data required. It is used to inform the algorithm of the relative position of components. The schematic is defined in terms of a computational coordinate system representing the flow domain (ξ,η,ζ). The domain itself is represented by a cube of dimensions 1000 x 1000 x 1000. Each component of the configuration is represented as a rectangular

plane in the schematic. The user is required to define each of these planes by specifying the four corner points of the plane in computational coordinates. Since each component is finite in size, each must be defined within the dimensions of the domain. In fact, the only boundary of the domain on which corner points may be defined is the $\eta = 0$ boundary which represents the plane of symmetry. The absolute dimensions of the planes are immaterial. The relative positioning of the schematic planes signifies the relative physical positions of components. Each component is represented by a plane which is constant in the most appropriate coordinate. For example, the fuselage is represented by a plane of constant η whilst a plane of constant ζ corresponds to the wing. Thus, for the components of the configuration shown in Figure 2a, the following information is specified.

a) Fuselage: plane of constant η, corners (200,0,200), (800,0,200), (200,0,800) (800,0,800)
b) Wing: plane of constant ζ, corners (400,0,500), (600,0,500), (400,200,500), (600,200,500)
c) Pylons: planes of constant η, corners (450,η_p,300), (450,η_p,500), (550,η_p,300), (550,η_p,500)
             with η_p = 50, 100 and 150 to represent the differing spanwise positions
d) Store: plane of constant ζ, corners (300,75,300), (300,125,300), (700,75,300), (700,125,300).

The fuselage schematic is defined at $\eta = 0$ since it intersects and is symmetric about the plane $Y = 0$ (ie the plane of symmetry). The schematic plane representing the wing is defined so that it intersects the fuselage plane. Each of the pylons intersect the wing and this is indicated by specifying the chordwise ζ coordinates for each inside the limits of those defined for the wing. The middle pylon also intersects the store, so its chordwise coordinates must also lie inside the range of those defined for the store. The nose of the store extends upstream of the wing leading edge and the rear extends downstream of the wing trailing edge and this is denoted by the relative ζ coordinates of the two components.

Armed with the schematic definition of the configuration, the topology generation algorithm can proceed to construct a component-adaptive topology. A topology consisting of 1246 blocks may be derived from the information provided for this configuration. The Cartesian framework of the topology can be examined during the construction process by viewing the projection of the schematic onto planes constant in one of the computational coordinates (Figures 2c-e). The topology algorithm has added layers of blocks either side of each component to allow the 'O' and 'C' structures to be embedded (section 6.1). The computer code incorporating this allows the optional interactive addition and removal of layers of Cartesian blocks. A limited number of points may be assigned to each block, so the addition of extra layers allows more points to be added in a given direction if found necessary. The facility for removing layers of blocks may be used to remove redundant layers, for example, when an 'H' grid structure is to be used instead of a 'C' grid structure for which extra layers are automatically provided (step 2 of the topology generation algorithm). For example, the arrows in Figure 2c point to two layers of redundant blocks which may be removed from the default Cartesian structure defined for this configuration to produce a 1040 block topology. This configuration is discussed further in section 6.

4 DISCUSSION ON GRID CONTROL

The method of Thomas and Middlecoff[11] is employed to control the distribution of points within the grid domain. On the assumption that grid lines transverse to the Dirichlet boundaries are locally orthogonal to the boundary and have zero curvature at the boundary, limiting forms of Eq (1) allow the control functions p^i to be determined based purely on the distribution of points on the boundary. The control functions are then interpolated throughout the topological structure of the grid to ensure that the grid stretching within the grid domain reflects the spacing on the Dirichlet boundaries.

This method is first used to control the generation of grids on the surface of each component. Surface grid generation reduces to grid generation in two dimensions since the surface of each component maps to a two-dimensional rectangular domain in the surface parametric coordinate system. The method due to Thomas and Middlecoff therefore only requires grid points to be predefined on the rectangular boundary and along the intersection boundaries with other components. A geometry intersection package provides the necessary stretchings along configuration component intersections. The other boundaries correspond to intersections with the flowfield boundaries or to the trailing edges and tips of the component surfaces. When combined, the Dirichlet boundaries on each surface form a number of distinct closed contours.

The process of distributing points along the Dirichlet boundaries depends upon the topology structure and knowledge of the relative positions of other components. This would prove to be a laborious job if undertaken interactively and would require a number of iterative steps before satisfactory grid quality is achieved. To ease this situation, an algorithm has been developed to position points along the Dirichlet boundaries of each component whilst being sensitive to the relative positions of other components. This method forms the basis of the default surface grid generation system.

The topology generation algorithm sets up a list of descriptors which are associated with features of the geometry and topology and assigns them to particular block edges. This information drives the grid control algorithm. It traces each Dirichlet contour, and examines any descriptor associated with a transverse edge. Grid points are then clustered towards the point at which the Dirichlet contour crosses such an edge. The example illustrated in Figure 3, which utilises the two-dimensional topology schematic shown in Figure 1e, indicates how the algorithm is used to control the grid near the trailing edge of an aerofoil. The boundary at the aerofoil surface will already have a fixed point distribution defined. The grid control algorithm therefore only needs to examine the outer Dirichlet contour. Starting at a nominal point downstream of the aerofoil trailing edge, the algorithm follows the contour in an anticlockwise direction until it meets a transverse edge with an associated descriptor. In this example, the first transverse edge met with such a descriptor is a vertical edge above the trailing edge of the aerofoil with the descriptor WINGTEX. This descriptor informs the algorithm to fix a point on the boundary directly above, and also below, the aerofoil trailing edge and gathers points either side of them.

5   INTERACTIVE GRID EDITING FACILITY

The quality of the grids on the surfaces of the configuration and the farfield boundaries have a direct effect on the properties of the grid within the flow domain.   The quality of a grid may be simplistically assessed by examining the local skewness, smoothness, aspect ratio and stretching of grid cells.   Since the surface grids are used as Dirichlet data for the generation of a global three-dimensional grid, any such deficiencies exhibited by them will be propagated into the field, and may have an adverse effect upon the flow solution.

The default grid generation system described above offers a limited number of methods for altering the distribution of points on the surface grids.   The method adopted for controlling surface grids requires that predefined point distributions or stretchings be fixed on the Dirichlet boundaries as already discussed.   A few of these prespecified boundary stretchings may be modified.   For example, points along the trailing edge of the wing can be repositioned using a cubic or linear point distribution. Alternatively points lying along the intersection line between two components may be redistributed by recalculating the intersection line using a Newton-Raphson technique.   The only other means for modifying surface grids is the use of factors by which the strength of the source functions p^1, Eq (1), is multiplied in certain predefined regions.   For example, for the surface grid on a wing-body-foreplane configuration, the 'C' grid structures around the leading edges of the intersections with the wing and foreplane can be controlled by user defined input parameters.   The successful use of this technique is again likely to be an iterative one which can only locally modify grids which already exhibit the basic qualities desired of a grid.

The methods discussed here have been used with considerable success to generate grids on a variety of configurations.   Experience has however shown that these grid control methods have some clearly identifiable restrictions. The grid point distribution algorithm is insensitive to local changes in geometry for a given component.   For instance, in distributing points along the trailing edge of a wing, the algorithm makes no attempt to determine whether the wing has a crank in it. Also, experience has shown that the detailed control of grids which are expected to stretch according to large variations in length scales of a given domain can be a demanding task. This problem is exaggerated if components are closely coupled as will be illustrated for a multi-element aerofoil case.   In addition, poor quality grids may be a result of unsuitable topology structures or insufficient numbers of points in a given direction.   Finally, for a grid generation scheme to be robust it is essential to provide a flexible mechanism for unfolding grids in regions that have crossed over, and for improving where necessary the qualitative features of a grid.

The issues raised above have motivated the development of an interactive surface grid editing facility to complement the default grid generation system.   The system is run in conjunction with the surface grid generator.   A set of default grids is generated on the surfaces of the configuration and the farfield boundaries and the grid generator identifies if any regions of a grid have crossed over.   The grids may be examined in detail using available plotting packages so that the quality of each may be assessed.   If the methods provided by the default grid generation system offer no further improvement to the grids, the interactive facility can then be implemented.   The editing facility is a menu driven program which employs standard graphics software to allow surface grids to be viewed and modified in a suitable format.   The menu provides a selection of options for interactive control of the view and edits.

Surface grids are initially generated in terms of their own parametric coordinate system.   Each component type (eg WING, FUSELAGE) to be processed is defined according to the standard parametric coordinate directions (s,t) with the normal to the surface pointing into the component.   For example, for the wing, the parameter s varies in the chordwise direction starting at the upper surface trailing edge around to the lower surface trailing edge, whilst t varies along the span of the wing.   Although the surface grids are subsequently mapped to the global coordinate system via the Coons patch technique, the interactive editor will operate on the parametric description of each grid.   The parametric description provides an advantageous format for viewing and applying modifications to surface grids due to the comparative ease of plotting two-dimensional grids.   The parametric coordinates for individual grids are output to separate files and the interactive editor will therefore require access to the grid files only for component grids needing alterations.   The surface topologies will already have been derived from the global topology.   When the parametric surface grid files and associated topologies are available, the interactive system may then be run.

5.1 Description of Interactive System

The interactive facility starts by prompting for the component name associated with the surface grid to be edited. The relevant data files are input and the grid is displayed in terms of its own pair of parametric coordinates.

A typical screen display is shown in Figure 4. A default view with only block edges visible is given. The example represents the grid on the surface of a fuselage which is intersected by a wing. Throughout the editing process a variety of colours are used to highlight different topological features such as Dirichlet boundary edges, internal continuity edges and new point distributions. Different line types will be used to illustrate these characteristics in this paper. The blocks are numbered to provide a means for identifying the areas to be operated upon. A main menu of options is displayed below the grid. The available functions fall into the following categories:

a) Editing functions (1,2,3,4 and 9)
b) Informative functions (5 and 6)
c) Viewing functions (7 and 8).

Each of these options yields sub-menus and/or sequences of prompts.   User input is entered either directly at the keyboard or with the use of a cross-hair cursor.   Responses from the user require validation and a variety of error messages are available.

Alternative views of the surface grid may be obtained at any time between edits using the viewing functions (7 and 8). The cross-hair cursor can be used to supply an enlarged view of an area of the grid, or a view of the grid in physical coordinates (x,y,z) from a given viewpoint can be chosen. For the latter case, the system will require the bicubic description of the surface for transforming from parametric grid coordinates to the global coordinate system. Although the default view of the grid only shows block edges, the view can also be changed to include all grid lines.

A number of options are available for gaining information about the grid and its topology (5 and 6). Function 5 provides a sub-menu of options for highlighting selected features of the grid structure such as the current distribution of points along block edges. Function 6 lists a sub-menu of options offering information on the topological structure; for example, a list of the adjacent component grids can be obtained or the axis system local to a given block identified.

The editing options (1 to 4) provide considerable flexibility for altering the distribution of points where required. The distribution on the Dirichlet boundaries is used, as discussed previously, to control the distribution within the grid domain. The default distributions may be changed and this in turn will reflect on the redistribution of grid points within the domain. Internal continuity block edges can also be modified and fixed in position. This provides a direct means for resolving features such as grid cross-over. Although sacrificing slope continuity of grid lines meeting at such edges, otherwise incorrectable deficiencies can be sufficiently improved to allow subsequent computations for the global grid and the flowfield. Finally, option 9 may be used to input edits of types 1 to 4 via an input data file.

5.2 Editing Grids

The editing functions operate upon grid points coinciding with topological characteristics of the grid such as block edges or block corners. Each surface block conforms to a standard numbering system for the relative positions of corners and edges (Figure 5). The block edge and corner numbering provides a mechanism for indicating where changes are to be made. The editing functions can then be applied either to points at block corners (option 1) or to points distributed along a series of one or more consecutive edges known collectively as a path (options 2 to 4). To ease the identification of a block corner to be edited, the corners of any block can be numbered. To identify a series of edges requiring edits, only the block corners lying at the end points of the string of edges need be given, together with the number of the block edge lying first along the path. For edges or corners which lie adjacent to others only one of the coincident edges or corners need be edited since the system will automatically redistribute the points associated with the others.

As functions 2 to 4 indicate, boundary and internal edges of the grid must be edited separately, since they are treated differently. As already discussed, internal continuity edges are fixed in position once edited and thus the boundary condition at such edges becomes Dirichlet. Edges along the boundaries of the domain may have new stretchings applied but the boundary condition remains unchanged.

Various mechanisms are available for allocating new point positions depending on the editing function chosen. Block corners can be moved either by specifying the new position in parametric coordinates, or by using a cursor digitising mechanism to pinpoint where the corner should be relocated on the grid. Points along a series of edges can be repositioned by choosing one of a selection of analytical stretchings, by specifying the new coordinates of each point along the path in turn, or by using the cursor mechanism to pinpoint each point location in turn. The choice of method depends very much on the problem to be resolved. The changes chosen are indicated on the grid with symbols 'X' and '+'. They may be accepted or rejected and the process can be repeated.

When all the necessary edits have been made, the edits may be saved, or completely rejected, leaving the original grid untouched. If edits to the boundary of the grid are saved, then any surface grid sharing the common boundary within the global structure must be modified accordingly to ensure that points on the common boundary match up when the grids are transformed to the physical coordinate system. The system can usually perform these edits automatically for the appropriate adjacent grids with the operator only required to confirm that they are acceptable. The operator can then proceed to edit another grid or exit the system.

When the necessary edits have been made to the set of surface grids, the grids are resubmitted to the surface grid generation process in their parametric form, as an initial guess for the iterative solution of the grid generation equations. The point distributions within the domain of each grid are recomputed based on the updated fixed edge distributions. The resulting grids can be replotted and their quality reassessed. The edit cycle, composed of edits followed by grid regeneration, may be repeated as necessary.

5.3 Examples of Editing Process

a)   Crank wing-fuselage configuration

The grid control algorithm defines a cubic or linear stretching along the full span of the exposed wing trailing edge. Input parameters control the strength of a cubic based on surface distances which is used to gather points towards the wing root and tip. The user is therefore unable to use this mechanism to position a point accurately at the location of the crank. In addition, this mechanism offers no facility for clustering points either side of the crank, and is thus unable to model the discontinuity in sweep at the crank. Figure 6 illustrates changes that can be made using the editing facility to alter the spanwise distribution along such a wing. Both the parametric and physical views of the default grid are shown. The labels ABCDEF highlight the correspondence between the parametric and physical coordinate systems. Examination of the wing geometry definition will give the parametric location of the crank. To control the spanwise positions of all of the grid lines defined around the wing in the chordwise direction, it is sufficient to redistribute points along the fixed spanwise

edges. Therefore, points along edges lying along the upper and lower surface trailing edge are redistributed and since the grid control algorithm also fixes the block edges lying along the leading edge of the wing, these must also be altered accordingly. Two layers of blocks are defined in the spanwise direction, and with appropriate numbers of points in each, it is sensible to try and position the interface between the two layers at the crank position. This can be achieved by positioning the block corners coinciding with both the interfacing edges and the leading and trailing edges of the wing at the crank position. The new position of these corners is marked on both views of the grid with the symbol 'X'. Then, by redistributing points either side of the interface in the spanwise direction, points can be gathered towards the crank as well as the wing root and tip. Again, cubic distributions can be used to cluster points towards these features. The sub-menu below the grid shows the possible analytical stretchings which can be applied. Grid points have been redistributed along the leading edge of the wing and along a section of the lower surface trailing edge. The points outboard of the crank are about to be modified. The end points of the single edge being edited are marked with the symbols 'Δ' and '∇'. Again, a cubic distribution can be used to draw points towards the crank and the wing tip. Figure 7a highlights the problem at the crank position where the grid definition has blended the sections either side of the crank. By moving the appropriate block corners and redistributing points along the leading and trailing edges as illustrated in Figure 6, the grid more accurately models the crank (Figure 7b). The regenerated grid on the surface of the configuration is shown in Figure 7c.

b) Multi-element aerofoil configuration

To illustrate the adverse effect which closely-coupled components can have upon a multiblock grid, a 53 block topology structure was set up for a wing-flap configuration which was treated as having an infinite span. This is essentially a two-dimensional test case with sections stacked in the spanwise direction to provide input to the three-dimensional multiblock Euler code. A number of topology structures were investigated and the structure shown in Figure 8a proved to be the most suitable. The wing is represented by the line A'A and the flap by line B'B. The structure allows for the slight overlap of the components. The minimum of two layers of blocks lying between the components is used so that 'C' grid structures may be embedded around each component whilst minimising the number of grid points lying between the closely-coupled components. An extra layer of blocks is generated below the configuration to provide the interactive editor with additional scope for fixing edges within the grid domain. The default grid based upon this topology (Figure 8b) is very poor in quality with highly stretched and skewed cells and grid lines pulled away from the surface of the configuration. The grid lines around the flap leading edge have crossed over due to the incompatible density of grid lines between the two components and between each component and the farfield. The grid is unsuitable for accurate flow calculation. The deficiencies identified within this grid indicate the difficulties that can be encountered in controlling meshes, about shapes of arbitrary generality, using the minimum specification of boundary data required for solving elliptic problems. In this example, the deficiencies are mainly caused by a combination of the inherent smoothing properties of the elliptic equations and the close-coupling of the elements. Other authors report[12] having to fix the position of grid points on all block boundaries in order to control the grid about a multi-element geometry. Here, the graphical editor is used to additionally constrain the default grid shown in Figure 8b. By modifying a number of the farfield boundary distributions and fixing points within the grid domain, the original deficiencies may be improved (Figure 8c). A number of the boundary edges have been edited to draw grid points closer to the configuration and many internal edges have been fixed to improve the size and shape of cells, particularly those near the surfaces of the configuration. Figure 8d shows pressure distributions on the surface of both components and the comparison with the analytical solution due to Williams[13], indicating that the grid is now reasonable for flow solution purposes but not ideal.

6   GENERAL MULTIBLOCK APPLICATIONS

To illustrate the power of the multiblock approach, four configurations are considered. The grids defined for each case have been generated using the automatic techniques described in sections 3 and 4. The default grids are acceptable for these cases but it is expected that the interactive grid editing facility will prove necessary for other types of geometries. A numerical algorithm for the solution of the Euler equations based on the ideas of Jameson, Schmidt and Turkel[1] has been adapted to enable the coupling with multiblock grids. Results are shown for some of the configurations discussed.

6.1   Wing-Fuselage-Three Pylon-Store Configuration

Firstly, for the wing-fuselage-three pylon-store configuration already considered (Figure 2) a 1040 block component-adaptive topology has been constructed. 'C' grid structures were embedded local to the wing and each of the pylons. A polar 'O' grid was defined around the fuselage whilst a spherical polar grid structure was embedded local to the store. Flowfield solutions are shown for this configuration and some of its simpler derivatives (Figure 9) illustrating the quality of solutions typically obtained.

6.2   Research Civil Wing-Fuselage-Tailplane Configuration

For the civil aircraft configuration shown in Figure 10, two alternative schematics of the geometry were defined for the automatic topology generator, one in which the tailplane had the same elevation as the wing and one in which the tailplane had a higher elevation than the wing. The grid topology resulting from the former schematic was found to be easier to control and computationally more efficient for the same number of surface grid points. The surface grid for this topology is given in Figure 10.

6.3   Military Aircraft Configuration with Propulsion

The complexity of a modern military aircraft configuration with twin engine intakes and afterbody

nozzles provides a good illustration of the applications of the techniques described. Figure 11 shows the grids on the component surfaces together with slices of the grid in the flow domain. The grid planes in the flowfield highlight the different grid structures generated local to the individual components. An 'O' grid structure was embedded local to the fuselage and afterbody nozzles, and 'C' grid structures were constructed around the wing and foreplane. An 'H' grid structure was used to model both the fin and the intake. Reference 8 presents example flowfield calculations on the wing and foreplane surfaces.

6.4 Military Aircraft with Tip Store and Winglets

The final example shows a surface grid for a research aircraft, indicating that the automatic grid generation procedure in the multiblock system can be applied to detailed geometric components.

7   CONCLUDING REMARKS

This paper concentrates on work undertaken in two main areas of multiblock grid generation, namely topology construction and grid control. Effort has been steered towards automation with the aim of transforming the multiblock technique to a more efficient and workable system. Consequently, the level of expertise required from those implementing the system is reduced and this should promote its practical use. To maintain a degree of flexibility within such a system, an interactive surface grid editing facility has been developed to remedy any localised deficiencies. The wide range of example configurations and flow solutions supports the continued use of multiblock techniques as a means of constructing component adaptive grid topologies.

REFERENCES

1   Jameson, A, Schmidt, W, Turkel, E, Numerical Solutions of the Euler Equations by Finite Volume Methods using a Runge-Kutta Time-Stepping Scheme, 1981, AIAA 81-1259.

2   Hall, M G, Cell-Vertex Multigrid Schemes for Solution of the Euler Equations, 1986, Numerical Methods for Fluid Dynamics II, Ed Morton, K W and Baines, M J, Oxford University Press, pp 303-345.

3   Sengupta, S, Häuser, J, Eiseman, P R, Thompson, J F (editors), Numerical Grid Generation in Computational Fluid Mechanics '88, 1988, Proc of Second International Conference on Grid Generation in Computational Fluid Dynamics, Pineridge Press Ltd.

4   Baxendale, A J, Application of Computational Fluid Dynamics to Military Aircraft at Supersonic Speeds, 1988, Proc.Royal Aero.Soc.Symp. on Aerodynamic Design for Supersonic Flight.

5   Fulker, J L, Ashill, P R, A Theoretical and Experimental Evaluation of a Numerical Method for Calculating Supersonic Flows over Wing/Body Configurations, 1988, AGARD FDP Meeting on Validation of CFD, Lisbon.

6   Weatherill, N P, Shaw, J A, Forsey, C R, Rose, K E, A Discussion on a Mesh Generation Technique Applicable to Complex Geometries, 1986, AGARD Symposium on Applications of Computational Fluid Dynamics in Aeronautics, Aix-en-Provence, France.

7   Weatherill, N P, Shaw, J A, Component Adaptive Grid Generation for Aircraft Configurations, 1988, AGARDograph AG-309, Current Practices in Grid Generation by J L Steger and J F Thompson, Ed H Yoshihara.

8   Shaw, J A, Georgala, J M, Weatherill, N P, The Construction of Component Adaptive Grids for Aerodynamic Geometries, 1988, Proc Second International Conference on Numerical Grid Generation in Computational Fluid Dynamics, Florida, Published by Pineridge Press Ltd.

9   Thompson, J F, Thames, F C, Mastin, C W, Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Field Containing any Number of Arbitrary Two-Dimensional Bodies, J Comp Phys, Vol 15, 1974.

10 Coons, S A, Surfaces for Computer-Aided Design of Space Forms, 1967, MID MAC-TR-41.

11 Thomas, P D, 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.

12 Schuster, D M, Generation of Patched Multiple-Region Grids Using Elliptic Equations, 1986, Proc of First International Conference on Numerical Grid Generation in Computational Fluid Dynamics, Landshut, W Germany, Published by Pineridge Press Ltd.

13 Williams, B R, An Exact Test Case for the Plane Potential Flow about Two Adjacent Lifting Aerofoils, 1973, ARC R&M 3717.
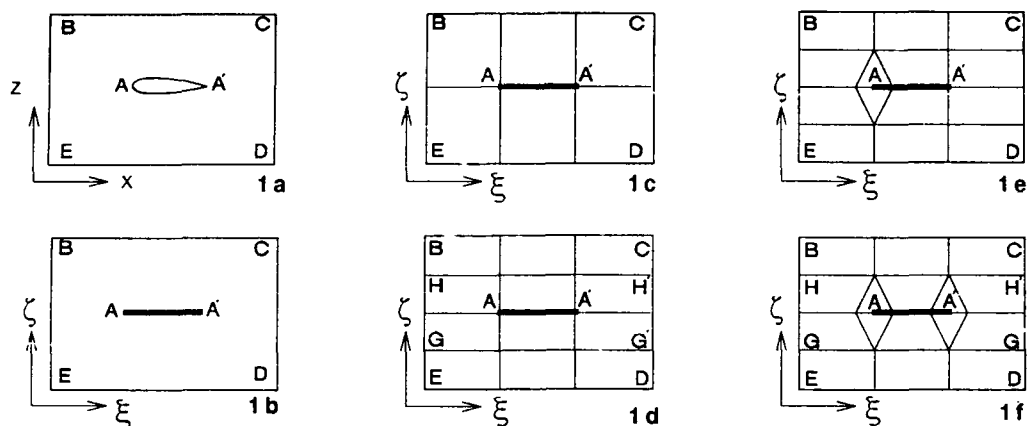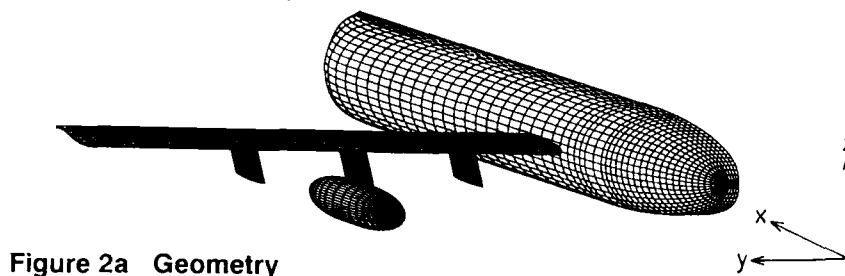
Figure 1   Block Decomposition in Two Dimensions about an Aerofoil
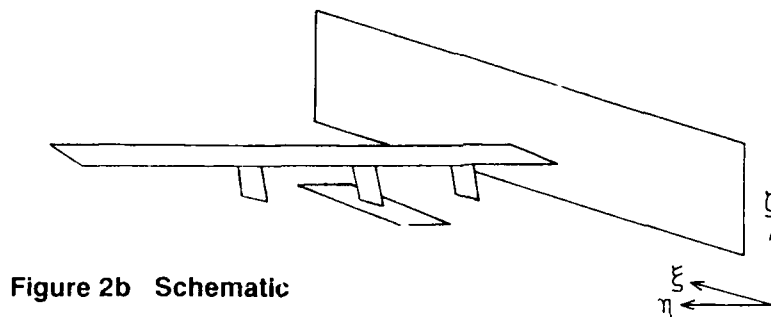

Figure 2a   Geometry

Figure 2b   Schematic
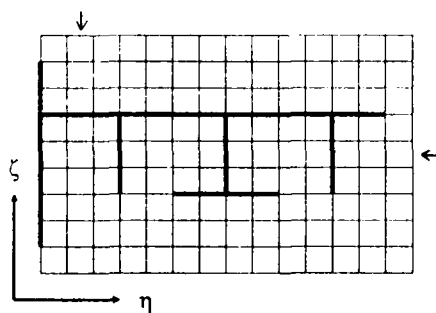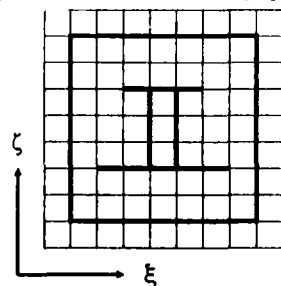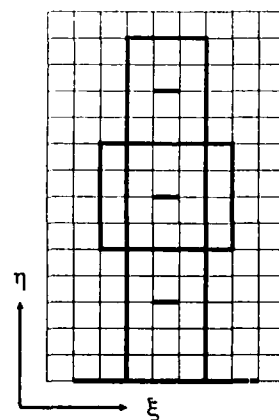

Figure 2c   Constant ξ plane

Figure 2e   Constant ζ plane

Figure 2d   Constant η plane

Figure 2   Wing-Fuselage-Three Pylon-Store Configuration
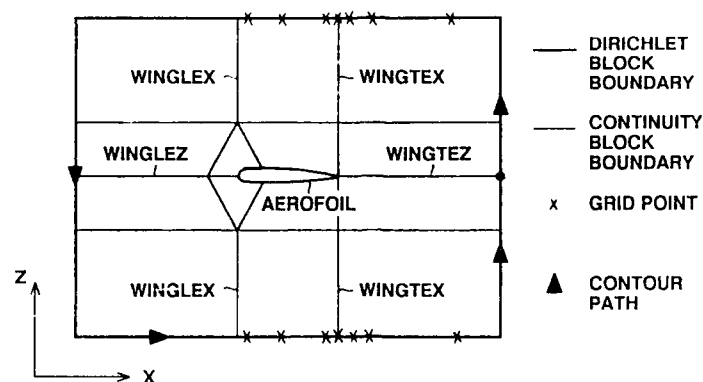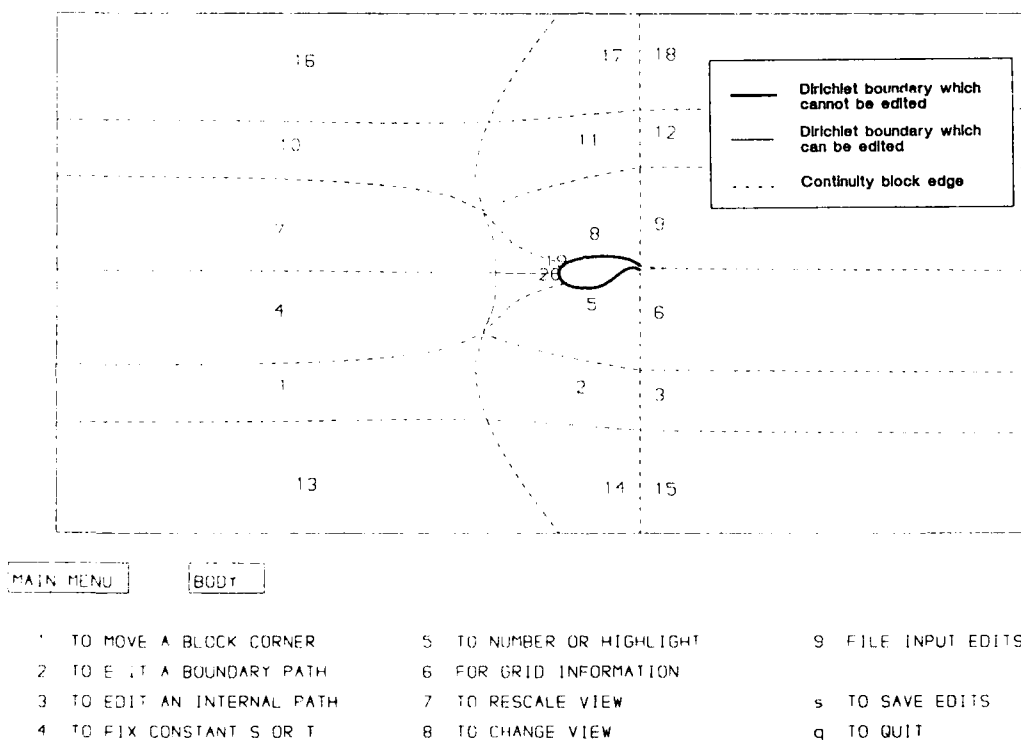
Figure 3   Path Traced by Grid Control Algorithm

```
'  TO MOVE A BLOCK CORNER       5  TO NUMBER OR HIGHLIGHT     9  FILE INPUT EDITS
2  TO E IT A BOUNDARY PATH      6  FOR GRID INFORMATION
3  TO EDIT AN INTERNAL PATH     7  TO RESCALE VIEW            s  TO SAVE EDITS
4  TO FIX CONSTANT S OR T       8  TO CHANGE VIEW             q  TO QUIT
```

Figure 4   Interactive Editing Facility - Typical Screen Display
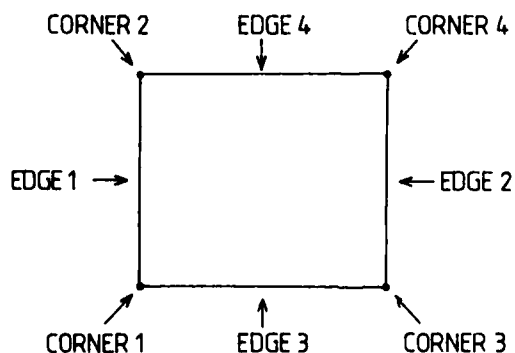
Figure 5   Standard Numbering System for Surface Block Corners and Edges

—— Dirichlet boundary which cannot be edited

x New position of block corner

—— Dirichlet boundary which can be edited

+ + + New distribution of points along a block edge

. . . . Continuity block edge

△ ▽ End points of path being edited

MENU 2   WING

STRETCHING OPTION:

1 LINEAR	4 COSINE 0 to PI/2	7 TANH	q TO QUIT
2 QUADRATIC	5 COSINE 0 to PI	8 GEOMETRIC	
3 CUBIC	6 SINE 0 to PI/2	9 CUBIC (2)	
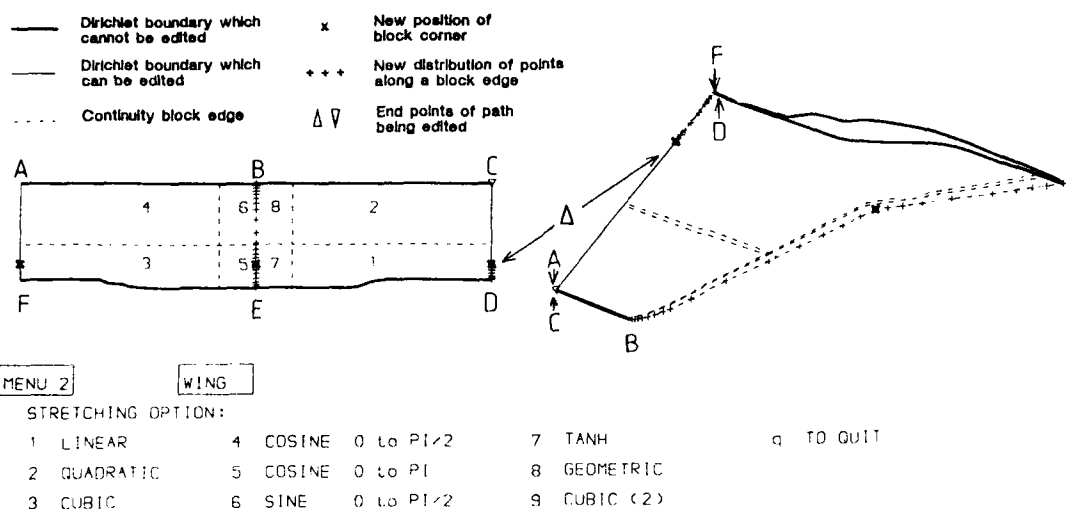
Figure 6   Edits for Modelling the Crank Position on a Wing
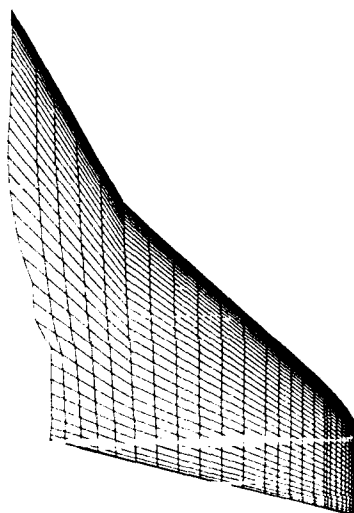
Figure 7a   Distribution of points on a Crank Wing Prior to Editing
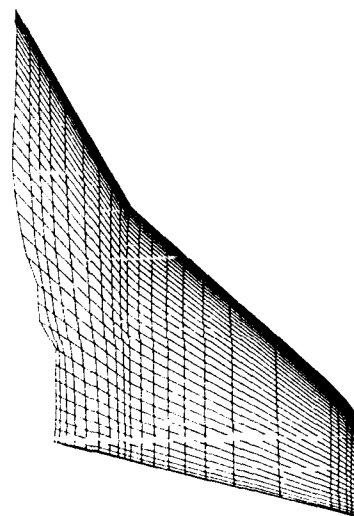
Figure 7b   Distribution of points on a Crank Wing After Editing
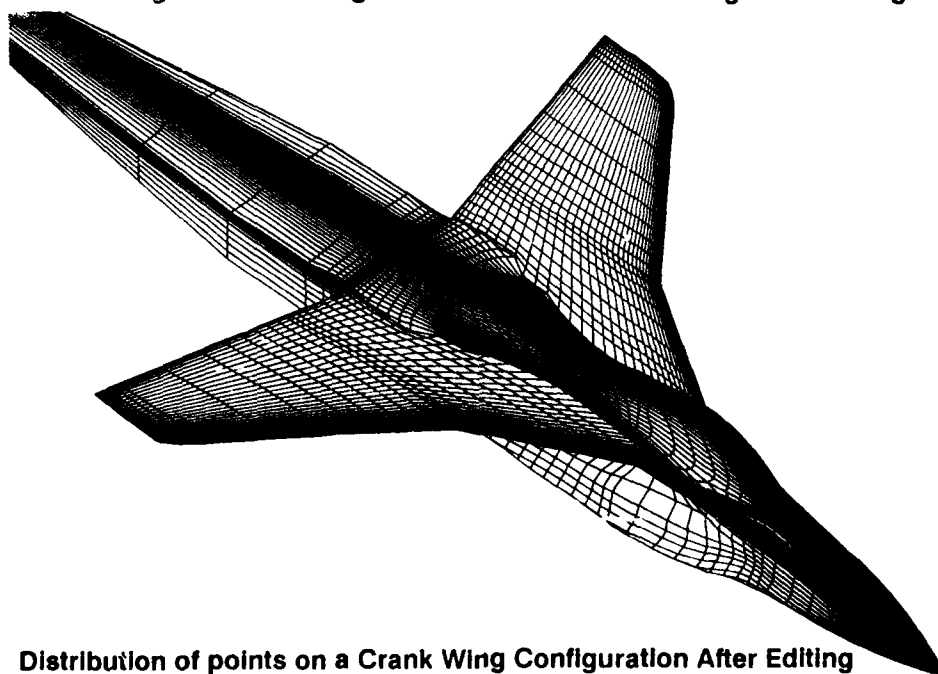
Figure 7c   Distribution of points on a Crank Wing Configuration After Editing
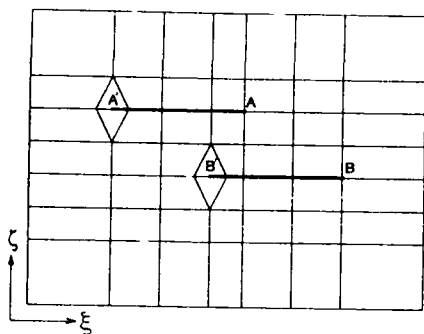
Figure 8a   53 Block Topology for a
Wing-Flap Configuration
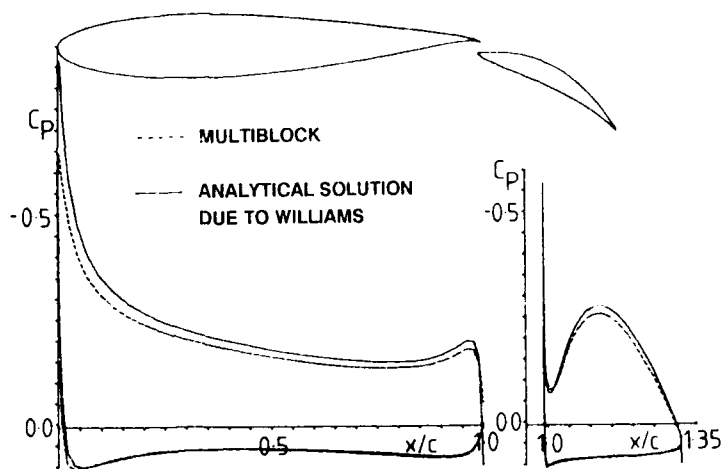
Figure 8d   Chordwise Pressure Distribution
on Component Surfaces
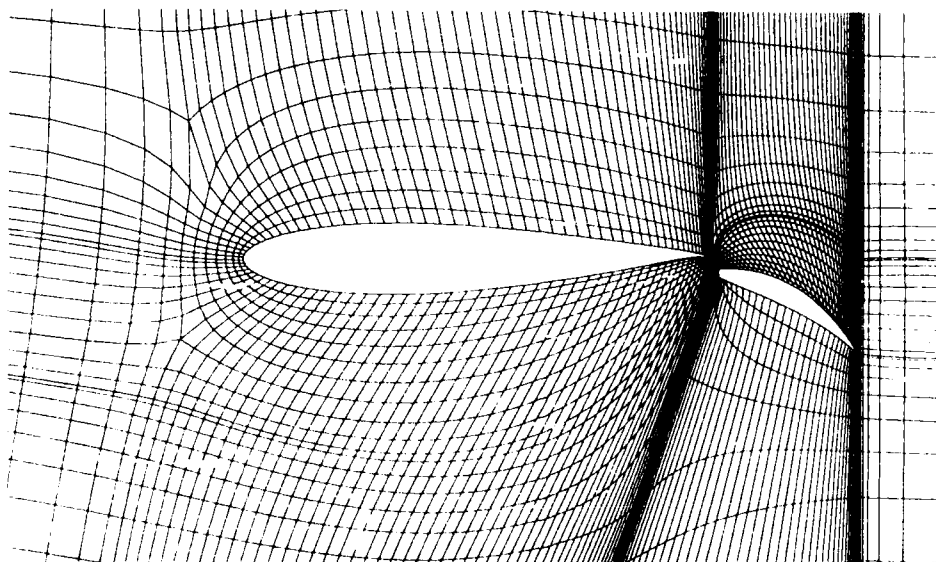
Figure 8b   Grid Structure around Wing-Flap Configuration Prior to Editing
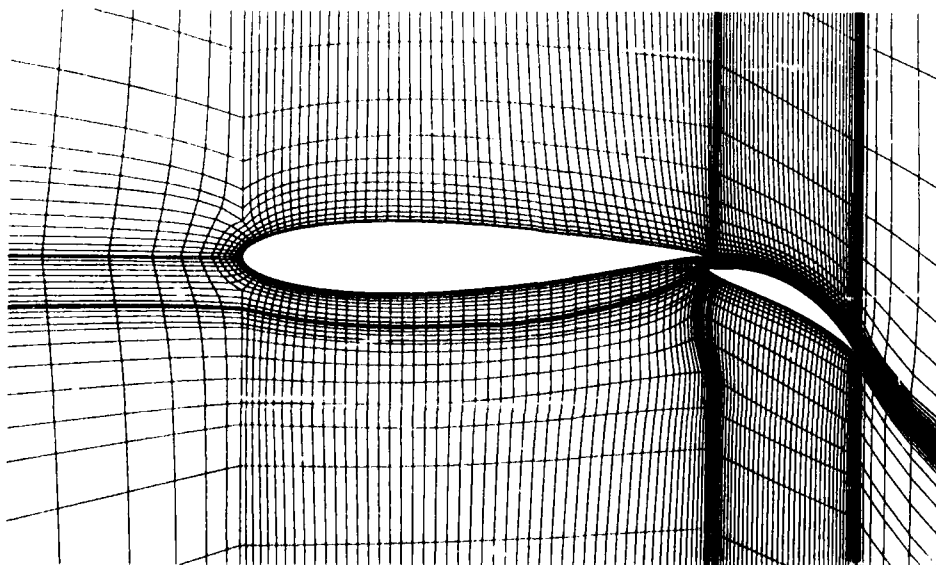
Figure 8c   Grid Structure around Wing-Flap Configuration After Editing

Figure 9b Wing-Pylon-Store

$M_\infty = 0.82$ $\alpha = 0°$
— MULTIBLOCK
△ EXPERIMENT (UPPER SURFACE)
□ EXPERIMENT (LOWER SURFACE)
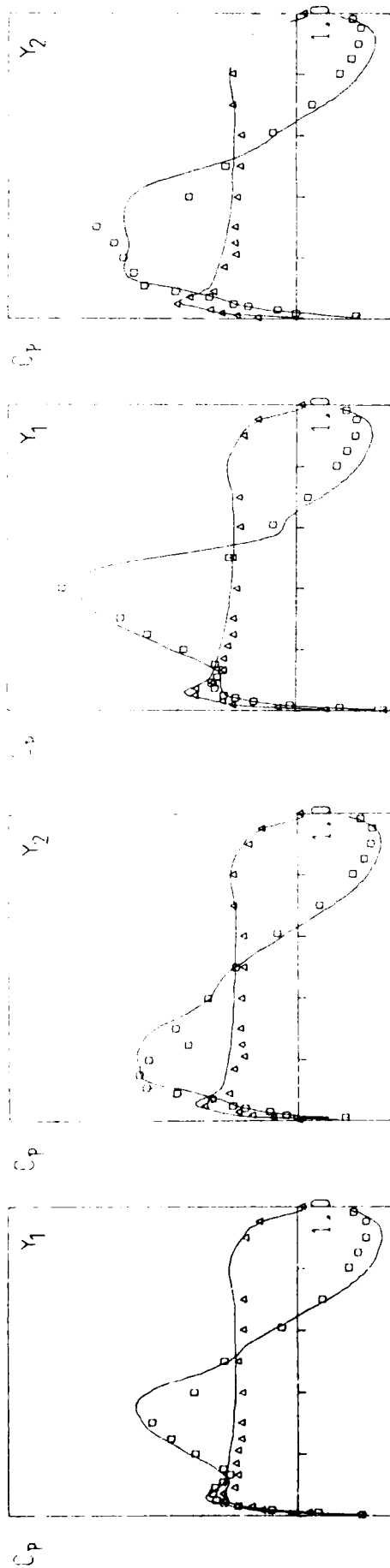
Figure 9a Wing-Pylon
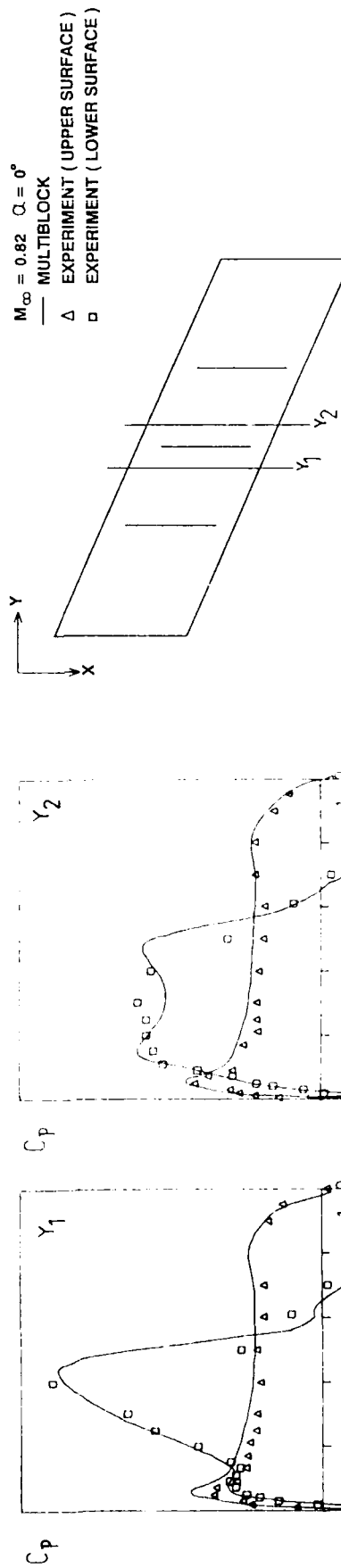
Figure 9c Wing-Three Pylon-Store

Figure 9   Chordwise Pressures on Wing Surface for a Wing-Fuselage-Multiple Pylon-Store Configuration and its Simpler Derivatives
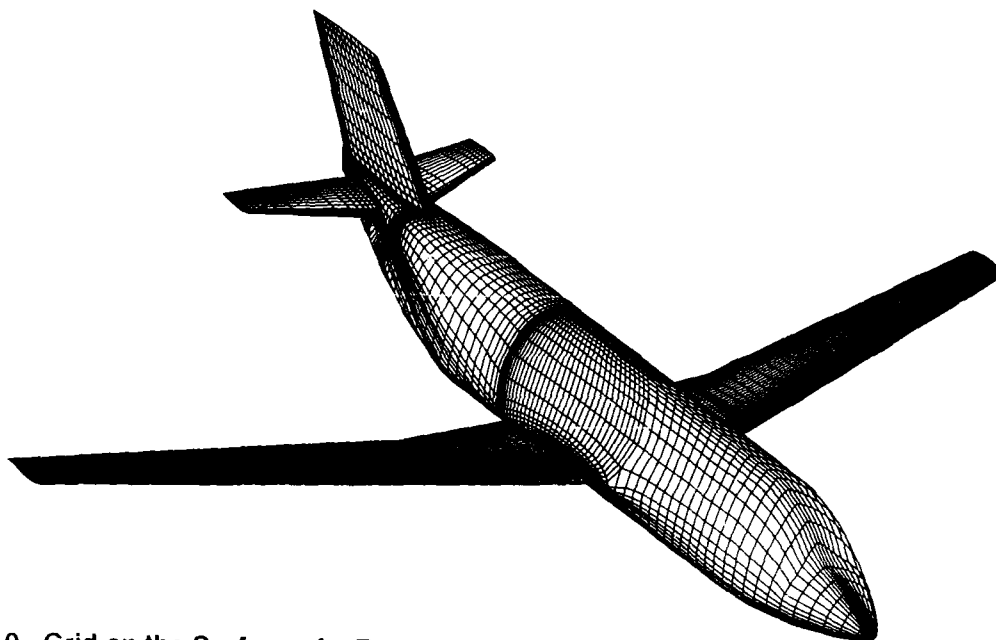
Figure 10   Grid on the Surface of a Research Civil Wing-Fuselage-Tailplane-Fin Configuration
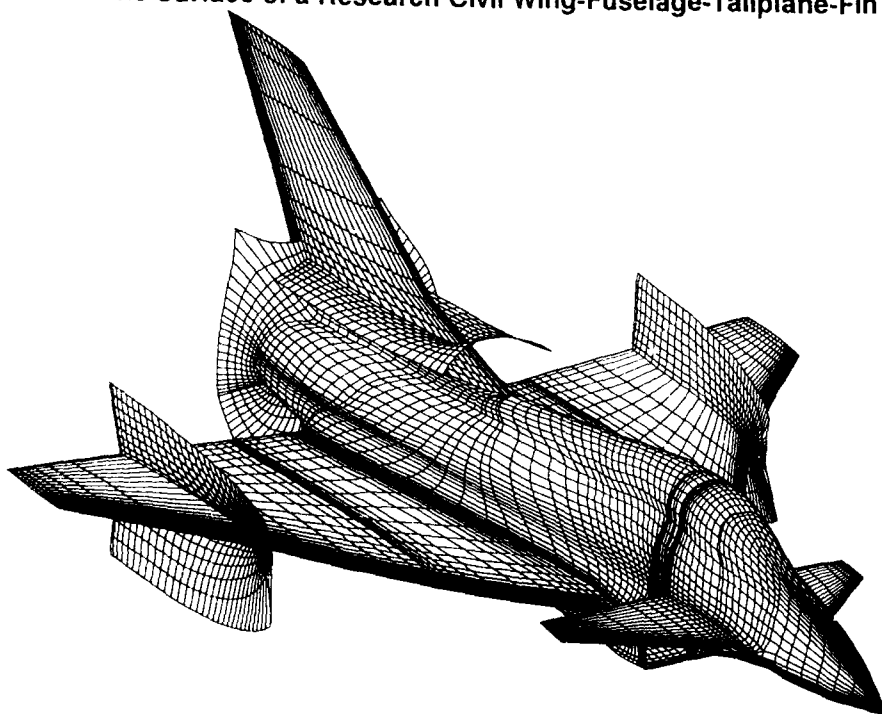
Figure 11   Grid on the Surface of a Powered Military Aircraft with Selected Planes from the Field Grid
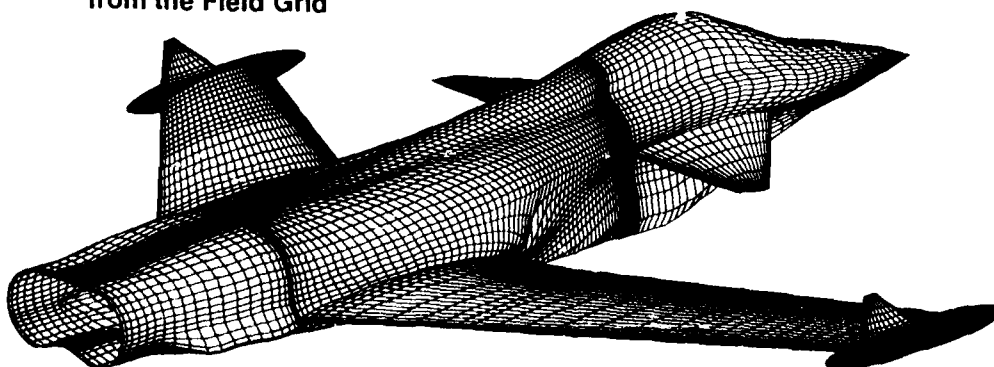
Figure 12   Grid on the Surface of a Research Military Aircraft with Tip store and Winglets

GENERATION DE MAILLAGE AUTOMATIQUE
DANS DES CONFIGURATIONS TRIDIMENSIONNELLES COMPLEXES
UTILISATION D'UNE METHODE DE "FRONT"

par

F.Huet
Departement des Etudes Théoriques Aérodynamiques
Avions Marcel Dassault — Breguet Aviation
78 quai Marcel Dassault
92214 Saint Cloud
France

Présenté à la Conférence AGARD

"Applications of Mesh Generation to Complex 3-D Configurations"

Loen, Norvège, 24-25 mai 1989

RESUME

Parmi les types de génération de maillage, la méthode de construction par "front", partant de limites facettées prédéfinies, paraît la plus prometteuse à l'égard des configurations complexes dans le cadre de la méthode numérique des éléments finis.

Sa souplesse et ses importantes possibilités d'adaptation en font un outil dont le domaine d'application est très large.

L'algorithme présenté fonctionne pour les maillages de configurations complexes rencontrées en aéronautique : tuyère de moteur à flux multiples, navette Hermès, Falcon.

I. INTRODUCTION

Le développement de méthodes numériques performantes pour la simulation de la mécanique des fluides a permis la modélisation de formes de plus en plus complexes, mais dont le délai de discrétisation tridimensionnelle est devenu pénalisant.

Ainsi dans le cadre de la méthode des éléments finis, l'apparition de résolutions numériques fiables des équations d'Euler des écoulements aérodynamiques en 3-D, et maintenant des équations de Navier-Stokes, rend les calculs quotidiens et accroît le nombre de configurations étudiées.

Un effort important a donc porté sur la réalisation de mailleurs tridimensionnels non structurés efficaces, pour des domaines très complexes.

Plusieurs procédés de modélisation non structurée ont été élaborés, en particulier des discrétisations tétraèdriques de l'espace qui offrent la souplesse nécessaire à la modélisation de formes géométriques complexes.

Cette approche favorise aussi l'utilisation de méthodes de raffinement et d'adaptation de maillage.

Cette présentation décrit une approche de la génération de maillages tétraèdriques sur des formes quelconques (en 3-D) par une méthode de front, et ses applications.

II. ALGORITHME DE GENERATION

1. LA GENERATION PAR FRONT

La méthode de création par front est caractérisée par la génération pas à pas de noeuds et d'éléments s'appuyant sur le front.

Par rapport à des mailleurs structurés ou des générateurs globaux type Voronoï, la méthode par front apporte des avantages considérables :

- Une indépendance complète par rapport à la forme discrétisée, quelque soit sa complexité.

- Des possibilités de modélisation importantes, les liaisons tétraèdriques permettant des configurations multiples.

- Une grande souplesse de contrôle de la modélisation permettant un choix local de la tétraèdrisation. Elle offre donc de grandes possibilités d'adaptation de maillage, pour capter finement les écoulements aérodynamiques.

- Un traitement local des problèmes rencontrés, autorisant la multiplication des zones de discrétisation complexe.

2. DISCRETISATION DE LA FRONTIERE INITIALE

La discrétisation triangulaire (cas tridimensionnel) de la frontière initiale est indépendante de la méthode de maillage présentée.

Elle peut donc être réalisée préalablement par des mailleurs surfaciques adaptés à la modélisation désirée aux limites, et permet l'établissement d'une base de donnée de maillages surfaciques.

La planche 1 présente des exemples de maillages utilisés.

3. ALGORITHME DE GENERATION

3.1. Présentation 2-D

3.1.1. Définition du front

Le front initial est une courbe orientée C, définie par une suite de points D.

Il est possible de définir en tout point X de D, par interpolation d'une courbe du deuxième degré au voisinage de x :

- une direction Nx normale,
- une valeur scalaire CVx reflétant la courbure de D en X, appelée aussi concavité.

CVx servira ainsi d'indicateur local de complexité de la forme.

3.1.2. Algorithme

Le générateur optimise la concavité (variable CV) sur les fronts successifs pour obtenir la concavité d'un cercle.

Le principe d'optimisation consiste pour le noeud I de CV maximum à élever des éléments supplémentaires à partir des barres de D voisines de I, en créant un noeud supplémentaire. Il est placé à une distance DIST de I selon la direction Ni.
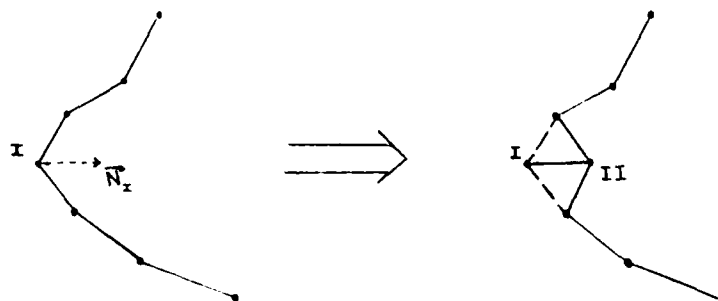
Un nouveau front est ainsi créé en remplaçant I par II.

Le mécanisme d'optmisation est complété en analysant la position relative de II par rapport aux noeuds déjà existants. En cas de forte proximité, il est confondu au noeud le plus proche.

3.1.3. Contrôle du front

Le contrôle du front s'exerce à deux niveaux, celui de la distance DIST (distance de I à II) et celui de la taille des mailles à sa surface.

DIST est calculée de manière à rendre les éléments créés les plus réguliers possibles, multipliée par un coefficient d'allongement QALL.

La surface du front est contrôlée par deux paramètres :

- Un coefficient QMIN définit la taille minimale acceptée des éléments du front, si elle est dépassée, le front est modifié localement pour respecter la limite, soit par suppression de la facette ou permutation de barre, soit par optimisation géométrique.
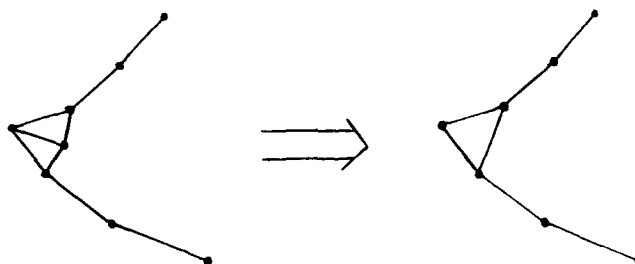
- Un coefficient QMAX déterminant la taille maximale acceptée des éléments du front, si elle est dépassée, l'élément ainsi déterminé est sous-maillé.
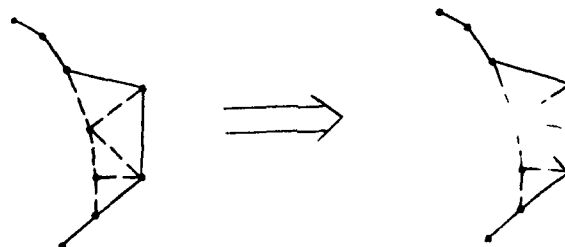
3.2. Généralisation 3-D

3.2.1. Définition du front

Le front initial est une surface orientée S discrétisée en triangles.

Il est possible de définir en tout point x de D, par interpolation des valeurs équivalentes aux notions établies en 2-D :

- Une direction Nx normale,
- Une valeur scalaire CVx reflétant la courbure maxi sur S en X, appelée aussi concavité.

plus

- Une valeur scalaire CXx reflétant la courbure mini sur S en X, appelée aussi convexité.

L'indicateur de complexité de la surface est alors le couple (CV,CX).

La figure 1 présente les valeurs des concavités pour des configurations courantes de l'aéronautique.

Ainsi le bord de fuite d'une voilure sera très convexe et l'emplanture de cette voilure assez concave, l'intersection entre les deux représente un "point selle" dans le maillage, à la fois très concave et très convexe, qui constitue la difficulté majeure de la configuration.

3.2.2. Algorithme

L'algorithme utilisé est identique au cas 2-D, avec la prise en compte du facteur supplémentaire de la convexité dans la méthode d'optimisation de la courbure.

L'optimisation est limitée au noeud I de CV maxi parmi les noeuds de CX faible, en créant alors des tétraèdres ayant pour base les facettes voisines de I sur le front.

Le calcul de DIST et les contrôles du front à l'aide des coefficients QALL QMIN QMAX, sont identiques au cas 2D.

4. ADAPTATION

Le programme est donc entièrement piloté par un nombre de coefficients limités :

- QALL   Allongement des éléments perpendiculairement au front.
- QMIN   Taille minimale autorisée des éléments du front.
- QMAX   Taille maximale autorisée des éléments du front.

qui permettent le contrôle de la répartition des mailles.

Ainsi il est possible de déduire des valeurs discrètes de ces coefficients en chaque noeud du front, pour adapter le maillage à un champ de déformation donné, décrivant la répartition de mailles voulue en un point quelconque de l'espace.

Cette répartition peut être déduite d'un premier calcul sur une modélisation grossière de la forme étudiée, suivant les zones d'intérêt des calculs à effectuer.

Une description simple du champ de déformation peut être fournie par la connaissance en tout point d'un trièdre définissant l'allongement des éléments selon trois directions orthogonales.

Les coefficients de génération peuvent alors être déduits en un noeud I du front :

- QALL est égal au maximum des normes des projections des vecteurs du trièdre selon la direction de la normale Ni du noeud I.

vecteurs du trièdre
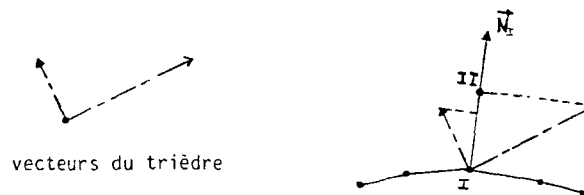
- QMIN et QMAX sont déduits indépendamment pour chaque élément à partir des projections des barres de celui-ci selon les vecteurs du trièdre.
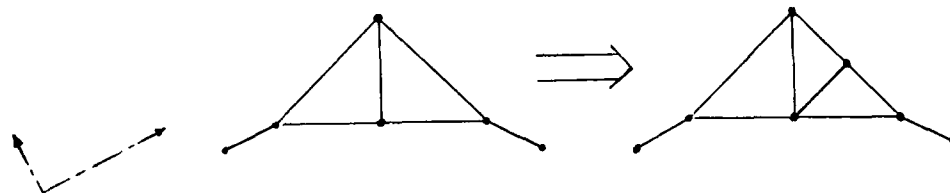
vecteurs du trièdre

III. CONCLUSIONS

En utilisant des structures de données adéquates pour les opérations de recherches géométriques (arborescences), l'algorithme conduit à un temps de création d'ordre o (N * Log(N)) pour un maillage non structuré de N noeuds.

Cette méthode, couplée à un code de résolution des équations d'Euler, a prouvé son efficacité sur de nombreuses configurations courantes de l'aéronautique.

La planche 2 présente plusieurs fronts lors de la création de maillages pour la cabine de l'Hermès, une tuyère de moteur.

La méthode présente en outre une possibilité d'adaptation importante de la discrétisation permettant la réalisation d'itérations sur différents maillages lors d'un calcul numérique.

IV. REFERENCES

1. B. STOUFFLET, J. PERIAUX, F. FEZOUI et A. DERVIEUX,
    "Numerical Simulation of 3-D Hypersonic Euler Flows around Space vehicles using Adapted Finite Elements", publication AIAA 87-0560, 1987.

2. J. PERAIRE, J. PEIRO, L. FORMAGGIA, K. MORGAN et O.C. ZIENKIEWICZ,
    "Finite Element Euler Computations in Three Dimensions", publication AIAA 88-0032, 1988

3. K. NAKAHASI et S. OBAYASHI
    "Viscous Flow Computations using a Composite Grid", publication AIAA 87-1128-CP, 1987

4. A. JAMESON, T.J. BAKER, et N.P. WEATHERHILL,
    "Calculation of inviscid transsonic Flow over a complete aircraft", publication AIAA 86-0103, Janvier 1986

5. K. MORGAN, J. PERAIRE, R.R. THAREJA et J.R. STEWARD,
    "An Adaptative Finite Element Scheme for the Euler and Navier-Stokes Equations", publication AIAA 87-1172-CP, 1987

6. R. LOHNER,
    "Simulation of strongly unsteady flows by the finite element method", publication AIAA 87-0555, 1987

7. A. JAMESON et T.J. BAKER,
    "Improvements to the Aircraft Euler Method", publication AIAA 87-0452, 1987

8. J. PERAIRE, K. MORGAN, J.PEIRO et O.C. ZIENKIEWICZ
    "An adaptive Finite Element Method for High Speed Flows", publication AIAA 87-0558, 1987

9. J.G. KALLINDERIS et J.R. BARON
    "Adaptation Methods for a New Navier-Stokes Algorithm", publication AIAA 87-1167-CP, 1987

10. R. LOHNER et K.MORGAN
    "Improved Adaptive Refinement Strategies for Finite Element Aerodynamic Computations", publication AIAA 86-0499, 1986

11. C.L. JACKINS et S.L. TANIMOTO
    "Octrees and their use in the representation of Three Dimensional Objects", Comp. Graphics Image Process., Vol. 14, pp. 249-270, 1980

12. D.F. WATSON
    "Computing the n-dimensional Delaunay Tesselation with Application to Voronoï Polytopes",
    The computer Journal, Vol. 24, N°2, pp. 167-172, 1981

13. N.P. WEATHERHILL
    "The Generation of Unstructured Grids Using Dirichlet Tesselations", Princeton University, MAE Report, n°1715, 1985

14. T.J. BAKER
    "Three Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets", publication AIAA 87-1124, 1987

15. P.L. GEORGE, F. HECHT, et E. SALTEL
    "Constraint of the Boundary and Automatic Mesh Generation", Proceedings Second International Conference on Numerical Grid Generation in Computational Fluid Mechanics, pp. 589-597, 1988

16. J.C. CAVENDISH, D.A. FIELD et W.H. FREY
    "An approach to Automatic Three-Dimensionnal Finite Element Mesh Generation", Int. J. Num. Meth. Engng., Vol. 21, pp. 329-347, 1985

PLANCHE 1. VALEURS DES CONCAVITES SUR DES
CONFIGURATIONS COURANTES

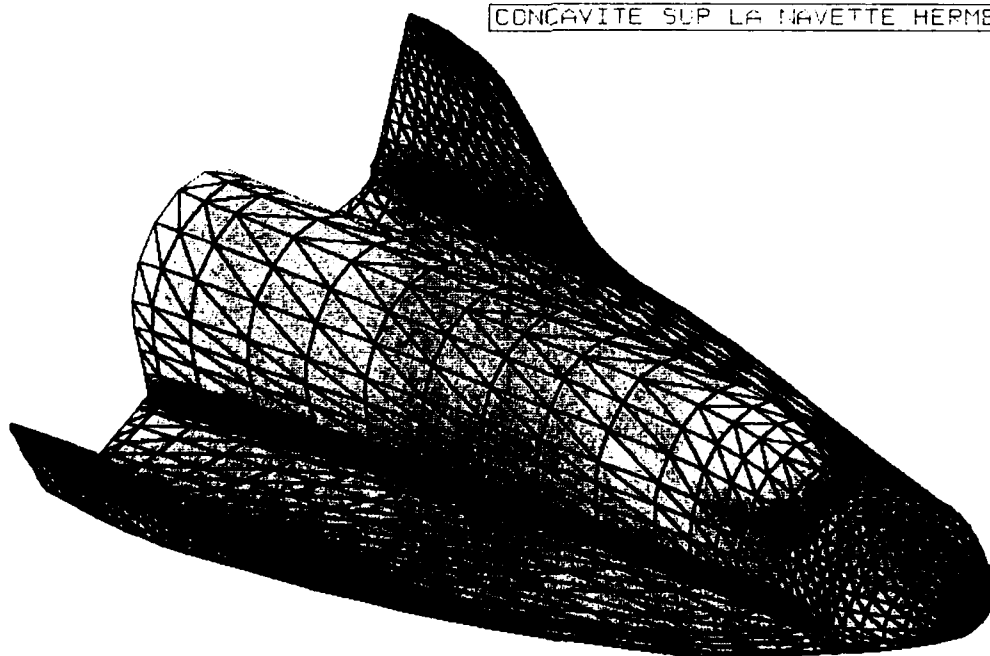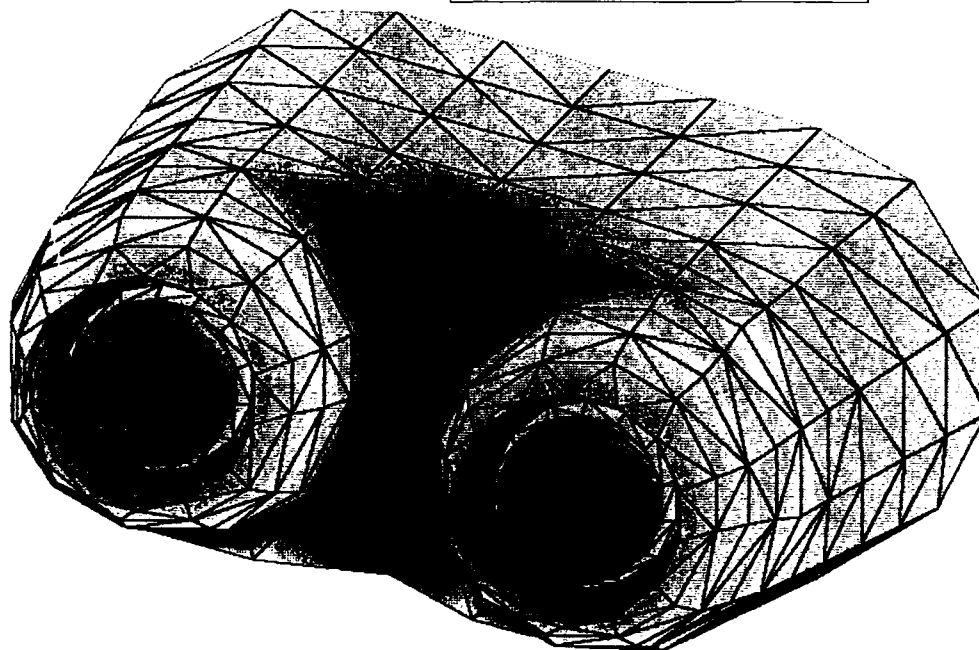CONCAVITE SUR UN FALCON
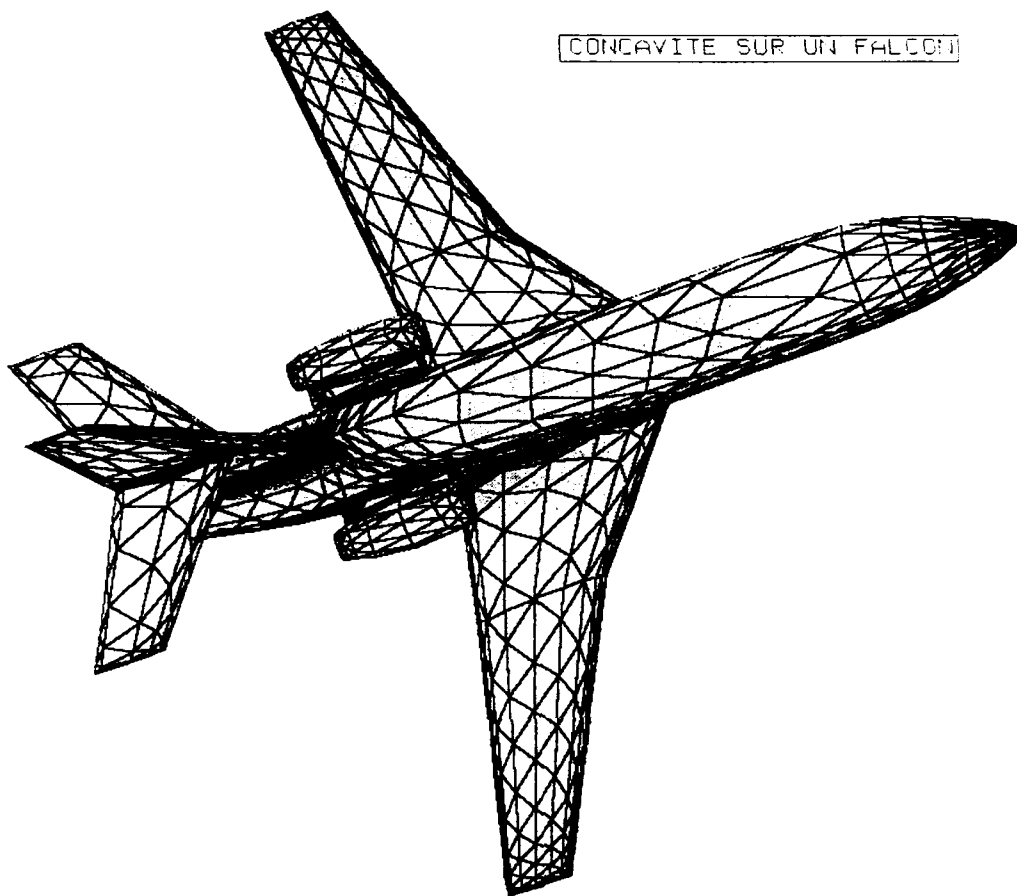
CONCAVITE SUR LA NAVETTE HERMES

PLANCHE 1. VALEURS DES CONCAVITES SUR DES
CONFIGURATIONS COURANTES

CONCAVITE SUR UNE TUYERE

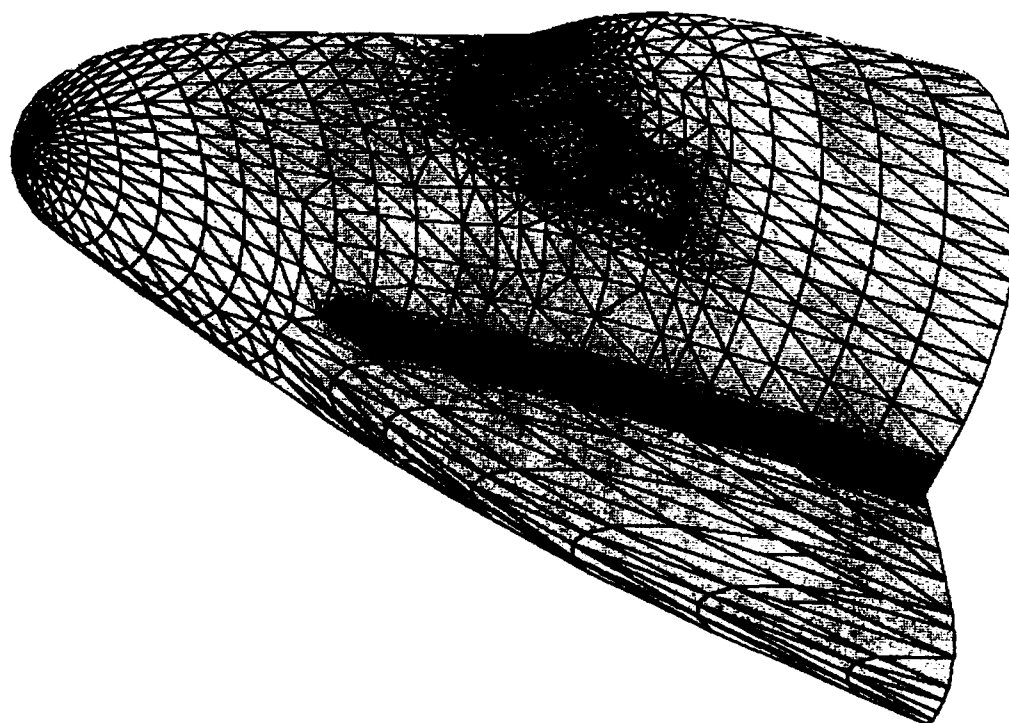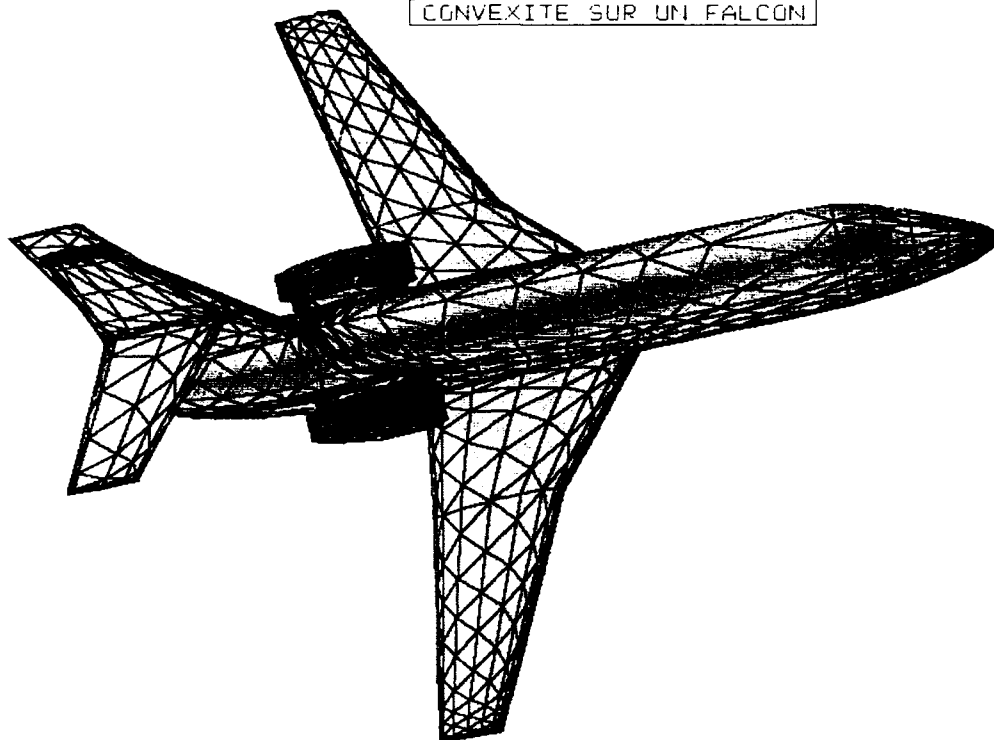CONCAVITE SUR LA CABINE DE L'HERMES

PLANCHE 1.   VALEURS DES CONVEXITES SUR DES
CONFIGURATIONS COURANTES

CONVEXITE SUR UN FALCON
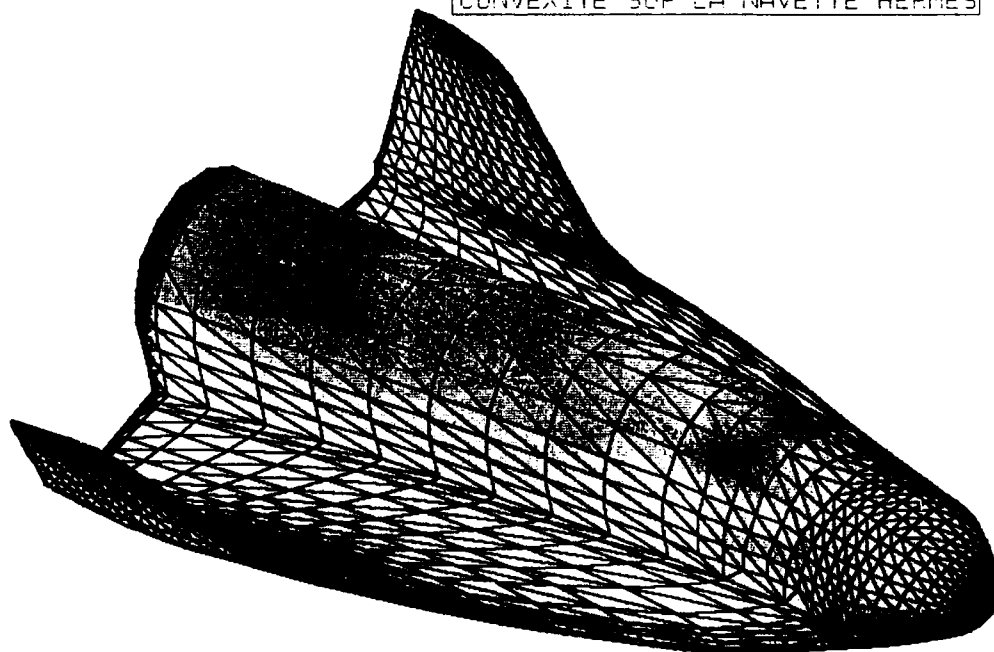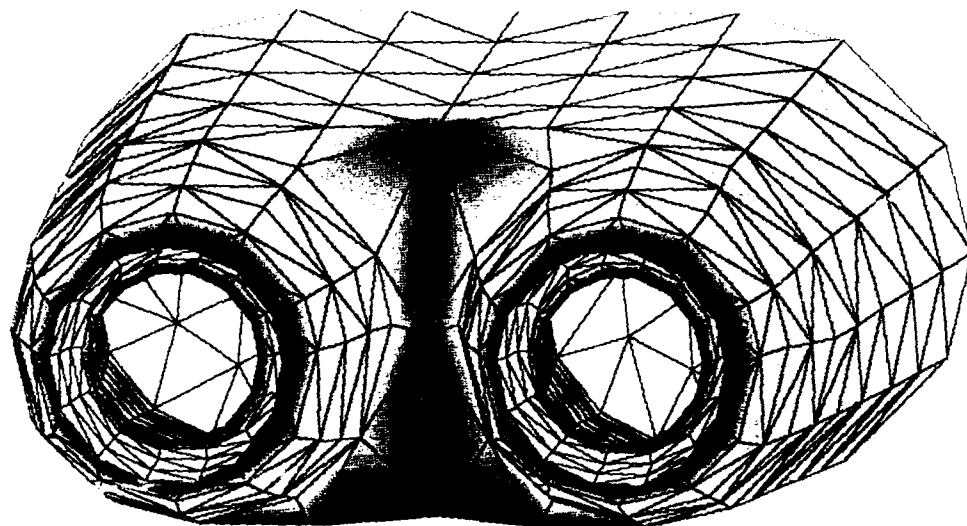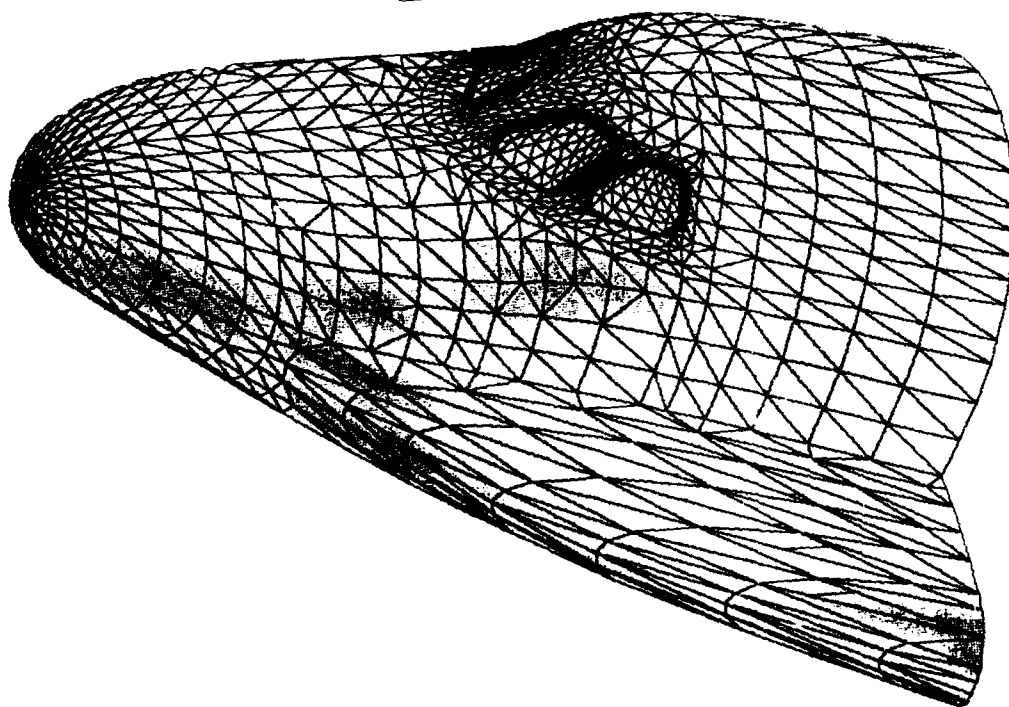
CONVEXITE SUR LA NAVETTE HERMES

PLANCHE 1. VALEURS DES CONVEXITES SUR DES
CONFIGURATIONS COURANTES

CONVEXITE SUR UNE TUYERE

CONVEXITE SUR LA CABINE DE L'HERMES

PLANCHE 2. QUELQUES FRONTS DE GENERATION D'UN MAILLAGE 2D.
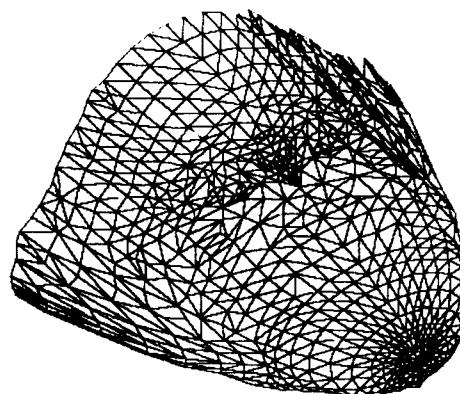
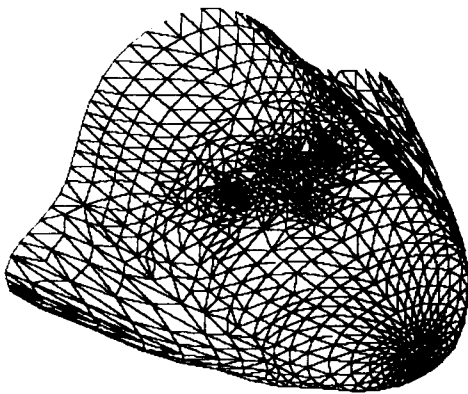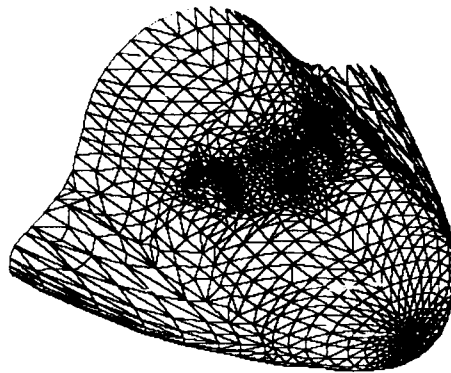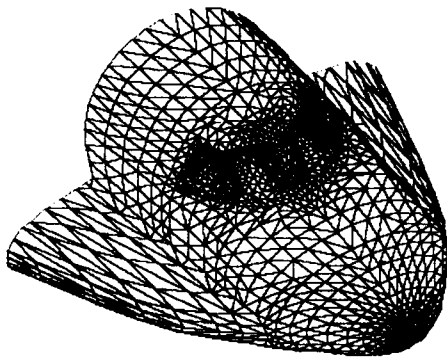PLANCHE 2. QUELQUES FRONTS DE GENERATION D'UN MAILLAGE 3D.
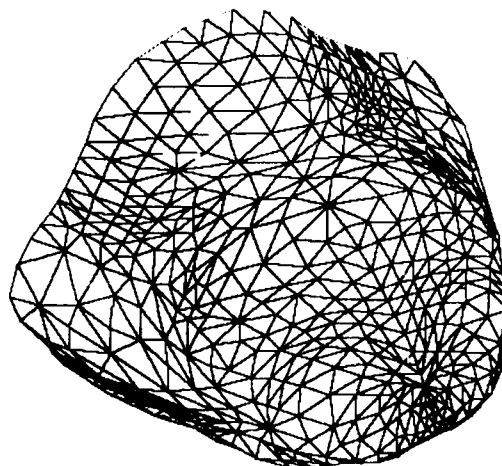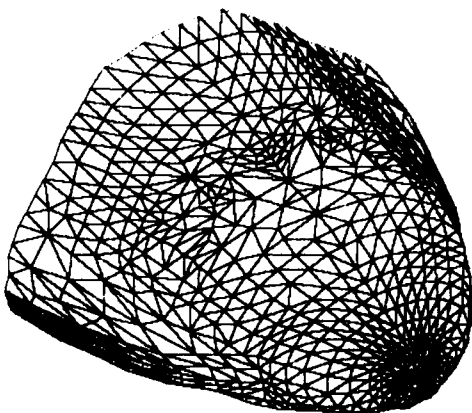
PLANCHE 3. RESULTATS DE CALCUL SUR L HERMES A L AIDE
D'UN CODE DE RESOLUTION DES EQUATIONS D'EULER

Valeurs des coefficients de pression

MACH=25.
INCI=30.
ZSOL=70000.

CALCUL EULER : Mach=1.5 — inci=10
coefficients de pression

Hermes

UNSTRUCTURED FINITE ELEMENT MESH GENERATION AND ADAPTIVE PROCEDURES FOR CFD

by

J. PERAIRE, K. MORGAN AND J. PEIRO

Department of Civil Engineering
University College
SWANSEA SA2 8PP
United Kingdom

SUMMARY

This paper describes a method for generating unstructured meshes of triangles or tetrahedra for computational domains of complex geometrical shape. To illustrate the power of the approach, it is applied to the solution of flows past several complete aircraft configurations. The advocated approach allows for the natural incorporation of mesh adaptivity and this is demonstrated for both inviscid and viscous computations in two and three dimensions.

1. INTRODUCTION

Over the past twenty years there has been a tremendous increase in the predictive capabilities offered by computational fluid dynamics to the aerodynamic design process. The numerical techniques which have been developed [1] allow for highly accurate computations to be performed for three dimensional flows over relatively simple shapes, such as wings. This success has led naturally to an increasing interest in the simulation of flows involving more complex geometries, such as complete aircraft configurations. The analyst responsible for such simulations is immediately faced with two major problems. The first is the problem of generating a mesh to cover the computational domain of interest and the second is the problem of developing a technique for mesh adaptation to improve the quality of the computed solution. Adaptive mesh methods will play an important role in the accurate simulation of flows past complex three dimensional configurations, since the number of mesh points involved is normally too large to contemplate the use of a uniform mesh subdivision.

One possible approach to overcome these difficulties is to allow the use of computational mesh which is completely unstructured. An early demonstration of the mesh generation capabilities of such methods was made by Bristeau et al [2], who computed the potential flow about a complete AMD/BA Falcon 50 configuration, and later by Jameson et al [3], who solved the Euler equations for the flow past a complete Boeing 747. More recent papers [4-6] show the current status of unstructured mesh generation techniques and indicate that the approach may now be routinely used to handle complex aerodynamic configurations. The decision to employ an unstructured mesh has the additional bonus that adaptive mesh techniques can then be implemented in a relatively straightforward manner. For the simulation of steady two dimensional Euler flows on triangular meshes, the improvements to solution quality which can be obtained by adaptive mesh enrichment methods was demonstrated by Löhner et al [7]. However, enrichment methods of this type are of limited usefulness for three dimensional analyses as the number of degrees of freedom can grow rapidly at each mesh adaptation. In this paper, we will demonstrate an adaptive mesh procedure which offers the possibility of enhancing the solution quality while allowing control over the increase in the total number of mesh points. The adaptivity is accomplished by complete regeneration of the mesh and an essential prerequisite will therefore be the development of an unstructured mesh generator with the ability to produce a mesh which agrees to certain externally prescribed requirements. The generator will be used to mesh the computational domain for the simulation of Euler flows about several complete aircraft configurations and sample solutions will be presented. The application of adaptivity will be demonstrated for viscous flows involving simple geometries, but complex flow features, in both two and three dimensions and also for the inviscid flow about a complete aircraft.

It should be mentioned that the decision to employ unstructured meshes for the solution of the compressible Euler or Navier-Stokes equations implies the availability of a suitable solver which can operate on such meshes. This subject is still in its infancy, in comparison with the massive investment which has taken place in the development of structured grid solvers. However, the accuracy of the solution produced on completely unstructured meshes can be expected to improve as the research effort in this area increases.

2. TWO DIMENSIONAL MESH GENERATION

We consider in this section the automatic generation of triangular meshes over arbitrarily shaped domains in the two dimensional plane. The algorithm to be presented will be capable of generating meshes which conform to an externally prescribed spatial distribution of element size. The ability to generate meshes which are locally stretched along prescribed directions will be included. The mesh generation algorithm used is a variant of the so called advancing front technique in which nodes and elements are created simultaneously [8,9].

The underlying basic concept in the advancing front technique is illustrated in figure 1. The boundary of the domain is discretised first. Nodal points are placed on the boundary curves in such a way that the distance between them is as close as possible to the desired mesh spacing. Contiguous nodes on the boundary curves are joined by straight line segments and assembled to form the initial generation front. At this stage the triangulation loop begins. A side from the front is chosen and a triangle is generated that will have this selected side as one edge. In generating this new triangle an interior node may be created or an existing node in the front may be chosen. After generating the new element the front is conveniently updated in such a way that it always contains the sides which are available to form a new triangle. The generation is completed when no sides are left in the front.

2.1 Boundary Representation

The boundary of the two dimensional domain is represented by closed loops of orientated piecewise cubic spline curves. For simply connected domains these boundary curves are orientated in a counter-clockwise sense while for multi-connected regions the exterior boundary curves are given a counter-clockwise orientation and all the interior boundary curves are orientated in a clockwise sense (figure 2). When these boundary curves are discretised [10,11], the boundary edges forming the initial front are orientated in the same fashion. Here the orientation of a boundary edge is defined by the order in which the two nodes of the edge are listed in the front. The ori-
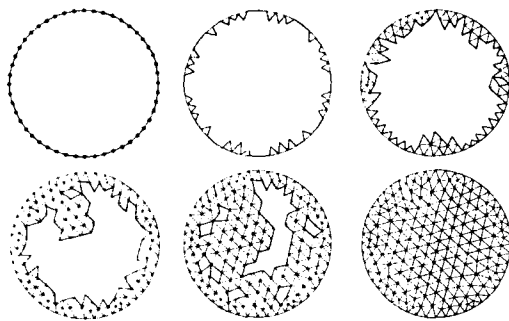
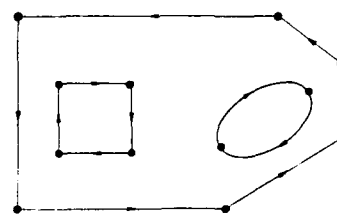**Figure** 1. Advancing front technique. Mesh generation procedure at different stages.
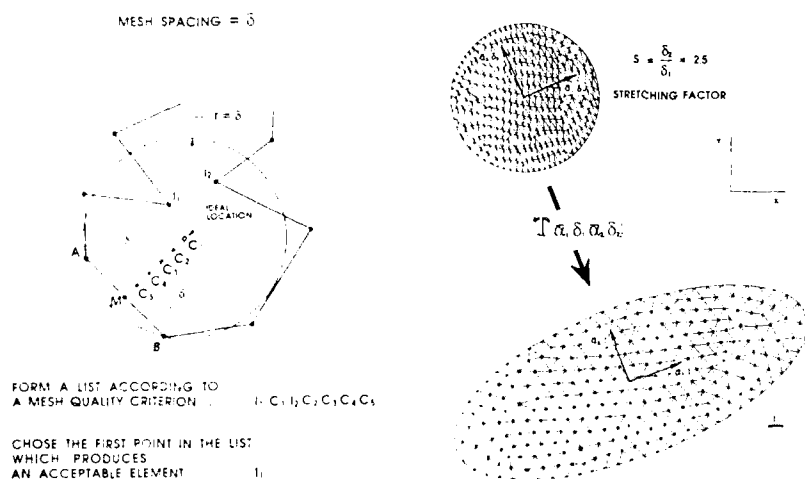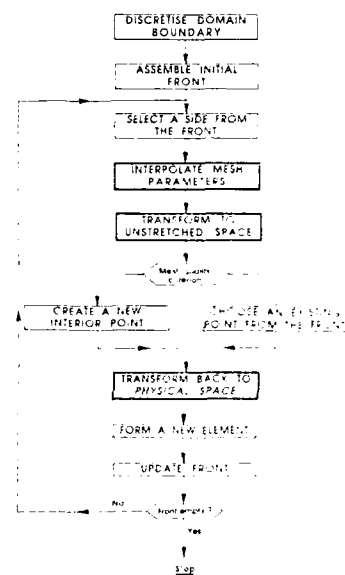
Figure 2. Orientation of the boundary.

MESH SPACING = δ

$$S = \frac{\delta_2}{\delta_1} = 25$$

STRETCHING FACTOR

$$T \alpha, \delta, \alpha_4 \delta_2$$

FORM A LIST ACCORDING TO
A MESH QUALITY CRITERION $l_1 C_1 l_2 C_2 C_3 C_4 C_5$

CHOSE THE FIRST POINT IN THE LIST
WHICH PRODUCES
AN ACCEPTABLE ELEMENT l_1

Figure 4. Transformation for stretching.

Figure 3. Generation of an element.

DISCRETISE DOMAIN BOUNDARY

ASSEMBLE INITIAL FRONT

SELECT A SIDE FROM THE FRONT

INTERPOLATE MESH PARAMETERS

TRANSFORM TO UNSTRETCHED SPACE

CREATE A NEW INTERIOR POINT

TRANSFORM BACK TO PHYSICAL SPACE

FORM A NEW ELEMENT

UPDATE FRONT

STOP

Figure 5. Mesh generation procedure.

Figure 7. Three dimensional mesh parameters.

Figure 6. Mesh generation examples in two dimensions.

MATHEMATICAL REPRESENTATION

3D DOMAIN

SURFACE COMPONENTS TENSOR PRODUCT OF CUBIC SPLINES

CURVE COMPONENTS CUBIC SPLINES

CORNER POINTS

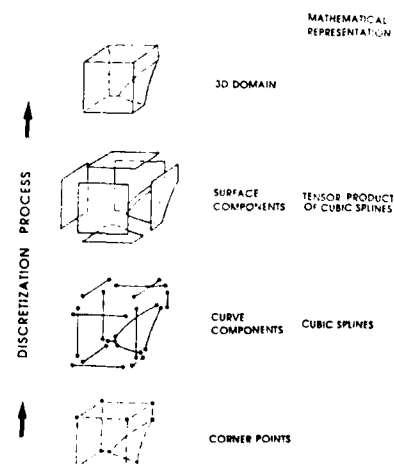DISCRETIZATION PROCESS

Figure 8. Mesh generation strategy in three dimensions.

entation of an edge is important as it identifies the area of the plane in which a valid triangle can be created using that edge as a base.

2.2 Domain Discretisation

The generation of a regular triangular element of size δ involves the following steps (figure 3):

(i)   Select an edge AB from the generation front.
(ii)   Using the orientation of the edge determine the position of point C_1, which lies at a distance δ from A and B.
(iii)   Determine all points in the front which lie inside a circle of radius δ and centre at C_1.
(iv)   Determine the positions of the equally spaced points C_2, C_3, C_4 and C_5 on the line joining C_1 and the midside point of AB.
(v)   Form a list containing all the points determined in step (iii) as well as points C_1, C_2, C_3, C_4 and C_5. Points in this list will be ordered according to their distance from the point C_1. For this ordering process the calculated distances associated with points C_1, C_2, C_3, C_4 and C_5 are incremented by an amount $\delta/2$.
(vi)   Create an element with nodes A, B and the first point in the list which satisfies the mesh consistency requirement, i.e. the two newly created edges do not cross any of the existing edges in the front.
(vii)   Update the front by removing the edge AB, and adding the appropriate number of new edges with the correct orientation.

2.3 Transformation for Stretching

The algorithm described above produces meshes in which the triangles will tend to be equilateral and of uniform size δ. In order to generate meshes in which the element size depends on the direction, the idea of a stretching transformation is introduced. Suppose that it is required to generate a mesh in which the elements will be approximately of size δ_1 in the direction α_1, and of size δ_2 in the direction α_2. By applying a simple linear transformation [10], the physical plane can be mapped into a parametric plane in which distances in the directions of α_1 and α_2 have been scaled by amounts δ_1 and δ_2 respectively. In the parametric plane the mesh generator described above is used to construct a regular mesh of element size equal to unity. The required mesh can be obtained by transforming the generated mesh back to the physical plane. The two vectors α_1 and α_2 and the scalar quantities δ_1 and δ_2 used to define the characteristics of the mesh are termed the mesh parameters. This process is illustrated in figure 4.

2.4 Variable Size and Stretching

For most practical applications it will be required to produce a mesh in which the parameters will vary from point to point in the domain. To specify this spatial variation of mesh parameters for the mesh generator a background mesh of linear triangles is employed. The mesh parameters are assumed to vary in a piecewise linear manner and this is accomplished by specifying the values of these parameters at each node of the background mesh. This background mesh must completely cover the domain which is to be discretised but there is no requirement for it to accurately follow the domain geometry. For most practical problems an initial mesh which exhibits a simple distribution of mesh size and stretching, can be produced with a background mesh which consists of only a few triangles. However, the generation of more sophisticated meshes, as for example meshes adapted to the flow solution, may require a substantially larger and more complex background mesh. A procedure to systematically produce background meshes of this type will be described later in the section on adaptivity.

When the spatial distribution of mesh parameters has been specified the desired mesh can be obtained by using the generation procedure illustrated in figure 5. In this figure the modifications to the previous procedure which are necessary to include the effects of variable size and stretching have been indicated by using bold boxes.

To illustrate this process, two meshes which have been generated over a rectangular region using a background mesh containing only two triangular elements are shown in figure 6.

3. THREE DIMENSIONAL MESH GENERATION

The mesh generation strategy proposed here for three dimensional domains is a direct extension of that presented above for two dimensions. The three dimensional space is discretised into tetrahedral elements and the characteristics of these elements are specified by the three dimensional mesh parameters, viz three mutually orthogonal directions α_1, α_2 and α_3 with corresponding sizes δ_1, δ_2 and δ_3 (figure 7). During the generation process, the local values of the mesh parameters are interpolated from a background mesh of linear tetrahedral elements. The boundary of the three dimensional domain is defined in terms of orientated surface components which intersect along curve components. The discretisation of the curve components is performed first and then each surface component is discretised into orientated triangular faces. The procedures employed for discretising both curves and surfaces make use of the techniques used in the two dimensional mesh generator described above. The collection of these triangular faces forms the initial generation front. The advancing front approach is used to discretise the domain into tetrahedral elements, with points and elements being created according to the distribution of the mesh parameters specified by the background mesh. The front is updated as each new tetrahedron is generated and the process terminates when the front is empty. This approach to the problem of discretising a general three dimensional domain is illustrated schematically in figure 8.

3.1 Complete Aircraft Configuration

In computational aerodynamics, a typical problem of current interest is the prediction of the inviscid flowfield about complete aircraft configurations. Figure 9(a) shows the computational domain which is adopted for the simulation of such a problem involving flow past a generic fighter with canard, cranked delta wing, vertical fin and engine inlet. The background mesh employed is illustrated in figure 9(b). The curve components, defined in terms of cubic splines and the discretisation of these components is displayed in figure 9(c). The individual surface components are described by patches of bi-cubic splines and the surface discretisation process is illustrated in figure 9(d). The triangulation of the components is performed by mapping each component in turn onto a two dimensional parameter plane and employing the two dimensional mesh generator [11]. The effect of the mapping on the mesh parameters must be correctly interpreted, so as to ensure that the generated mesh meets the specified mesh
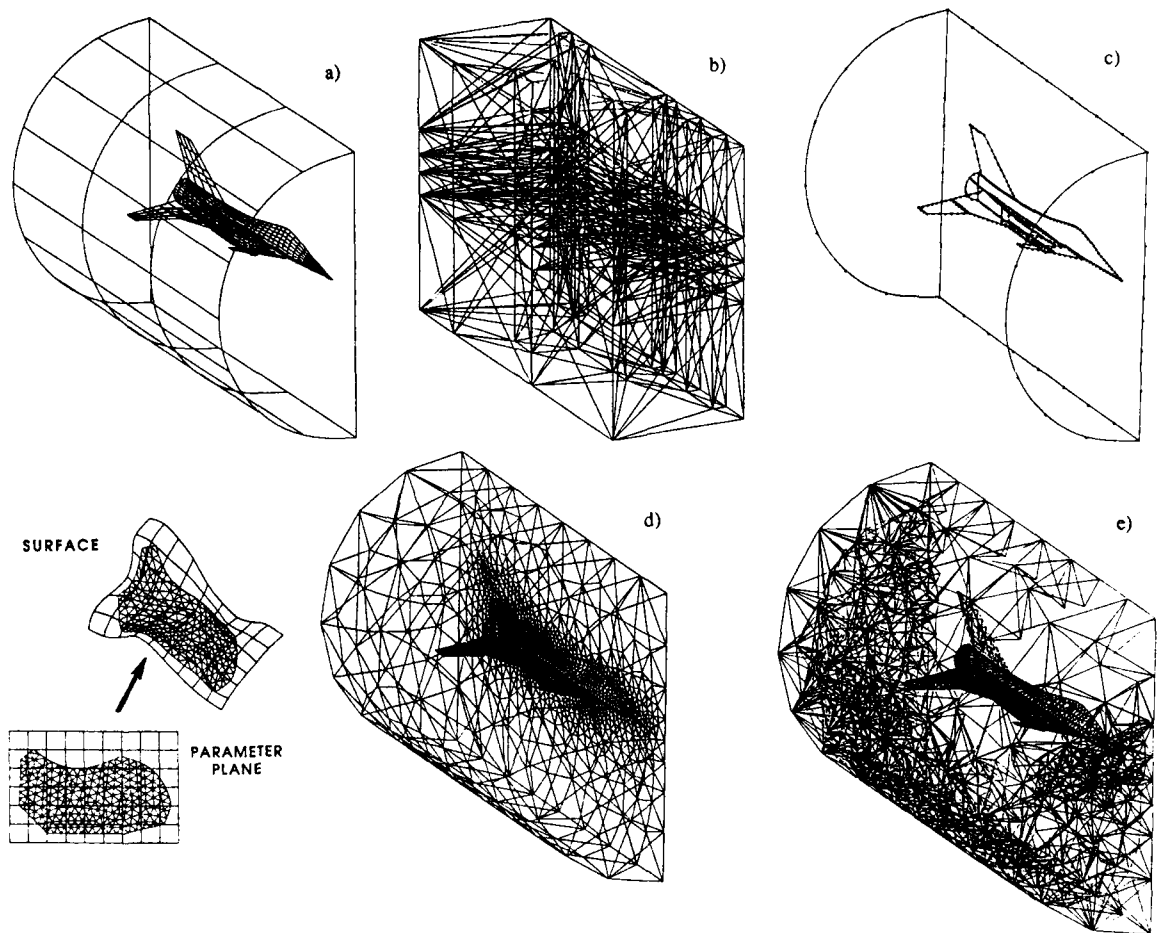
Figure 9. Mesh generation for a complete aircraft configuration. a) computational domain, b) background mesh, c) curve components representation and generated points, d) surface discretisation and e) partial view of the tetrahedral mesh.
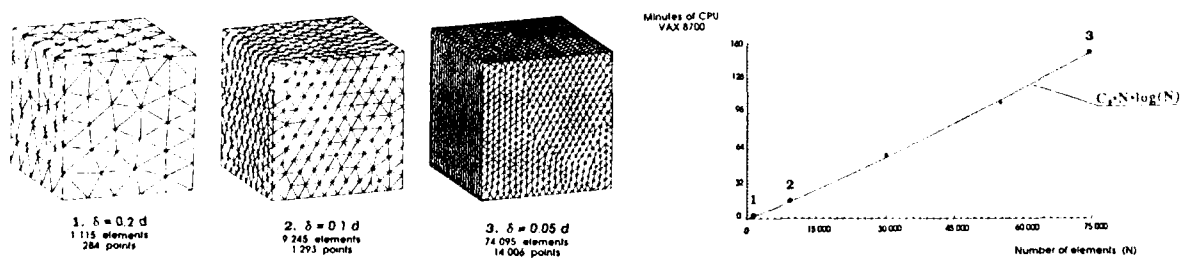

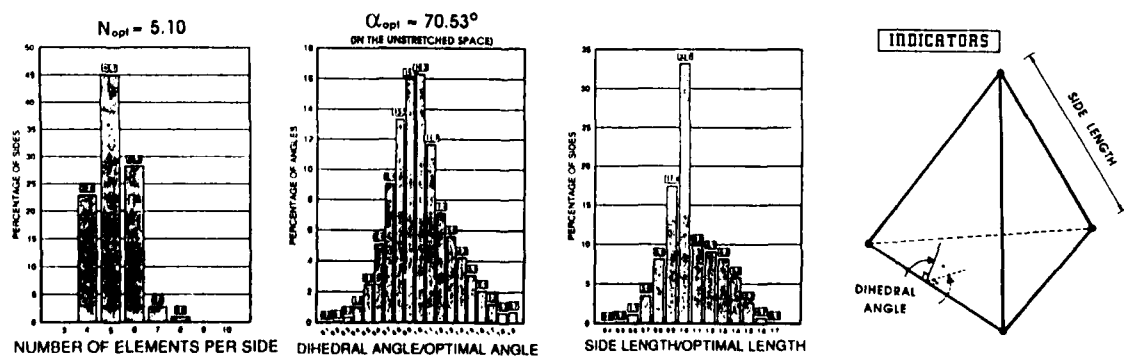**Figure 10.** Three dimensional mesh generation timings.

Figure 11. Mesh quality statistics.

requirements when mapped back onto the surface. The tetrahedra generation can now begin and an intermediate stage of the process is displayed in figure 9(c). This process involves the same algorithmic steps as the triangle generation procedure described for two dimensions. However, the consistency test which must be done on the newly generated elements is more complicated. In our implementation we ensure that no new edge intersects with any existing face in the front, and that no new face intersects with any existing edge in the front. Moreover, elements with excessively small angles are not accepted.

3.2 The Alternate Digital Tree (ADT) Data Structure

From the previous section it is apparent that a successful implementation of the three dimensional mesh generator will require the use of a data structure which enables certain sorting and searching operations to be performed efficiently. The main operations of this type which we have to perform are:

Insertion - Addition of new items to a list
Deletion - Removal of items from a list
Geometric searching - Identification of the elements from a list which are close in the physical space to a specified item in the list.
Geometric intersection - Identification of the elements from a list which intersect in the physical space with a specified item in the list.

The alternate digital tree [12] is a data structure which enables the above operations to be performed optimally. It is a generalisation of the binary tree search [13] and has the ability to deal with either nodes, edges, faces or elements as single items. It can be extended to any number of dimensions and only requires two additional memory addresses per item. An inconvenience associated with the ADT is that its use results in a scalar process with intensive indirect addressing and it is therefore not vectorisable. However, it offers interesting possibilities for parallelisation which are currently being explored.

In our implementation an ADT data structure is employed for the generation front and also for the background mesh. The computational performance of the method is illustrated in figure 10 which plots computer time against the number N of elements generated. It can be observed that a typical $N*\log(N)$ behaviour is attained. The three dimensional meshes shown in this paper have been generated on a VAX-8700 at an approximate rate of 30,000 tetrahedra per CPU hour.

3.3 Mesh Quality Assessment

Any discussion of mesh quality should be intimately related to the form of the solution which is to be represented on the mesh. Two factors need to be considered here:

1.- Determination of the characteristics of the optimal mesh for the problem at hand. This introduces the concept of adaptivity and this aspect is considered in the next section.

2.- Assessment of how well the generated mesh meets the requirements specified by the mesh parameters. This assessment can be made by examining the generated mesh and determining the statistical distribution of certain indicators. For example in figure 11 we have chosen as indicators the number of elements around an edge, the dihedral angle and the edge length. These indicators are compared with optimal values which are determined using the local values of the mesh parameters.

4. FLOW SOLVER

The solution of the Euler equations on arbitrary triangular and tetrahedral meshes is accomplished by using an explicit two step finite element method [5,14]. Stability in the vicinity of flow discontinuities is maintained by the application of an explicit artificial viscosity based on a pressure sensor. Boundary conditions are applied via the integral statement and are based upon the use of a linearised characteristic analysis. A highly vectorised form of the code has been produced [15]. The memory requirements for the three dimensional version of the code are 94 storage locations per node. On a single processor of a CRAY-XMP the time required per iteration and per node is approximately 70 microseconds.

For the solution of the Navier-Stokes equations, we employ a semi-structured mesh in the immediate vicinity of solid walls and a fully unstructured mesh elsewhere. The semi-structured mesh is constructed by expanding slightly the solid surface in the normal direction and dividing the region formed into a prescribed number of exponentially stretched layers of quadrilaterals in two dimensions or triangular prisms in three dimensions. An implicit solution scheme is used in the region where the mesh is semi-structured, with the mesh structure being utilised in an equation solution procedure based upon line relaxation. On the unstructured portion of the mesh the explicit two step algorithm is again applied [16].

5. ADAPTIVITY

An adaptive mesh approach is proposed in which each mesh adaptation is accomplished by complete regeneration of the mesh. The computed solution on the current mesh is used to provide local information for the 'optimum' distribution of the mesh parameters for the new mesh. The procedure is illustrated in the flowchart of figure 12. It can be observed that the essential feature of the process is the determination of the new distribution for the mesh parameters. This is accomplished by using an error indicator, which is produced here by applying the results of interpolation theory [17].

5.1 Error Indicator

We assume that we have computed a converged solution to the problem of interest on a certain grid and we intend to use this solution to predict the distribution of the error. Although we are dealing with a vector system of equations, the error indication is generally based upon a representative scalar 'key' variable, generally the density or the Mach number.
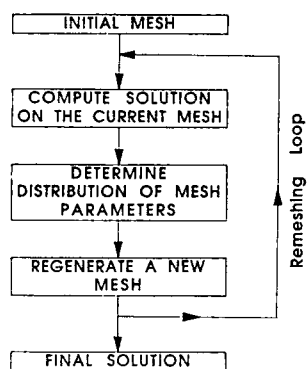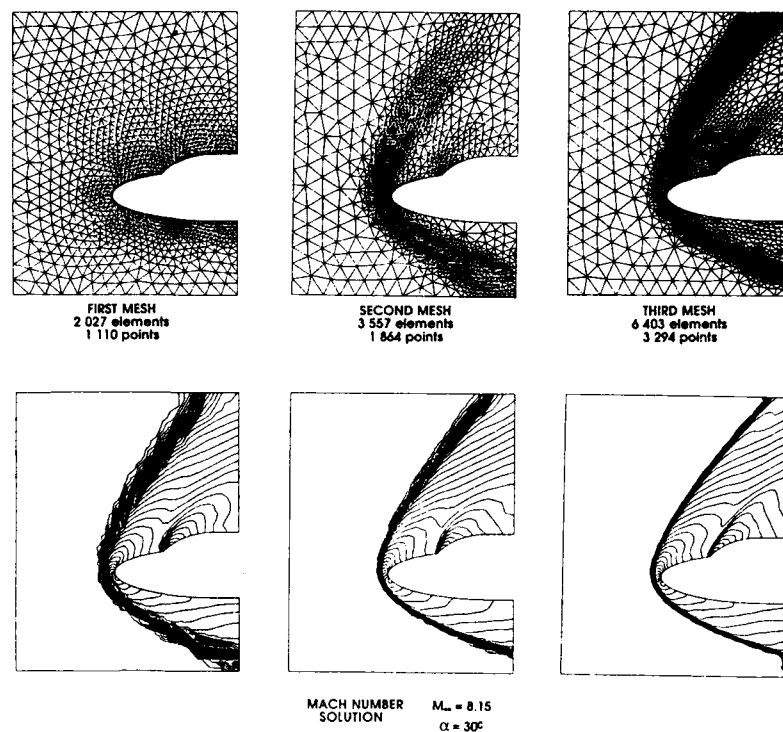
INITIAL MESH

COMPUTE SOLUTION
ON THE CURRENT MESH

DETERMINE
DISTRIBUTION OF MESH
PARAMETERS

REGENERATE A NEW
MESH

FINAL SOLUTION

Remeshing Loop

Figure 12. Adaptation procedure.

FIRST MESH
2 027 elements
1 110 points

SECOND MESH
3 557 elements
1 864 points

THIRD MESH
6 403 elements
3 294 points

MACH NUMBER
SOLUTION
$M_\infty = 8.15$
$\alpha = 30°$

Figure 13. Adaptive remeshing. Hypersonic flow over a double ellipse.

A   B   C   D

D
MESH

279 196 TETRAHEDRAL
ELEMENTS

53 066 NODAL POINTS

C

B

A

$C_L = 0.312$

-Cp

-Cp

-Cp

-Cp

X

X

X

X

SECTION A-A

SECTION B-B

SECTION C-C

SECTION D-D
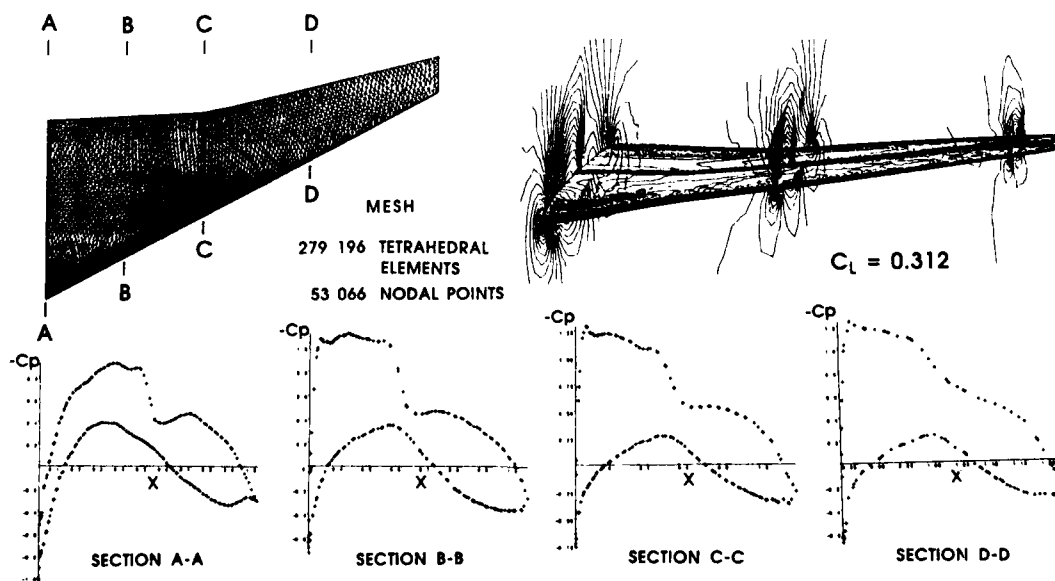
Figure 14. Transonic W4 wing.

GEOMETRY DEFINITION

57 INTERSECTION LINES
29 SURFACE COMPONENTS

SURFACE   26 060 TRIANGULAR FACES

DOMAIN   388 614 TETRAHEDRAL ELEMENTS

PRESSURE SOLUTION
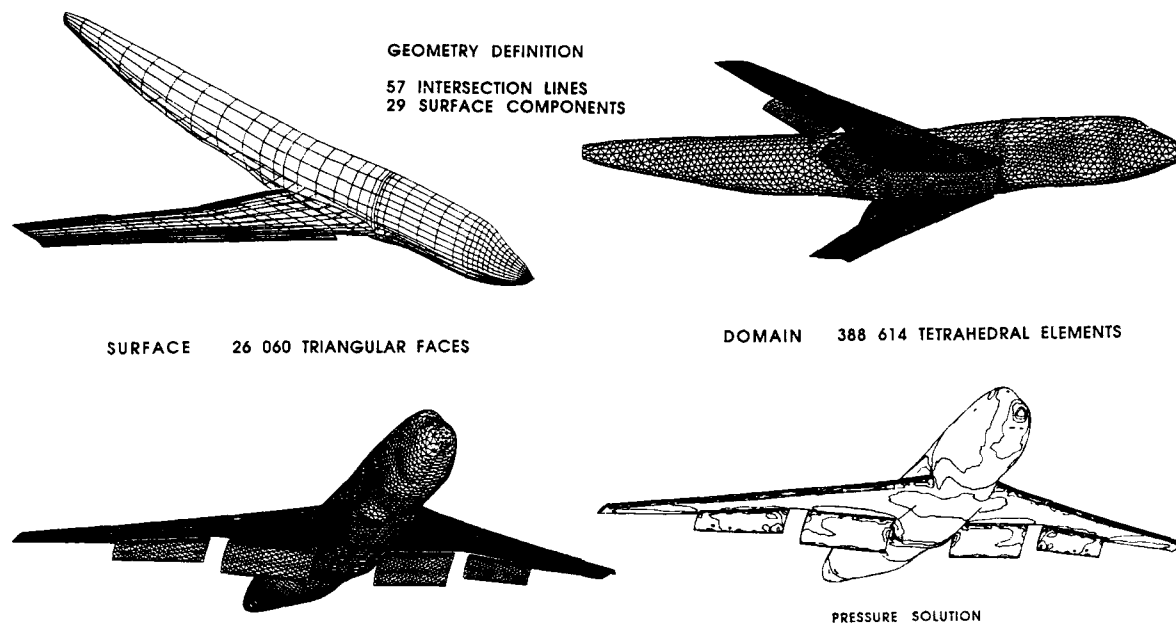
Figure 15.   Boeing 747 in landing configuration.

GEOMETRY DEFINITION

87 INTERSECTION LINES
37 SURFACE COMPONENTS

DOMAIN

903 282 TETRAHEDRAL ELEMENTS

SURFACE

61 468 TRIANGULAR FACES

PRESSURE COEFFICIENT

ENGINE DUCTS
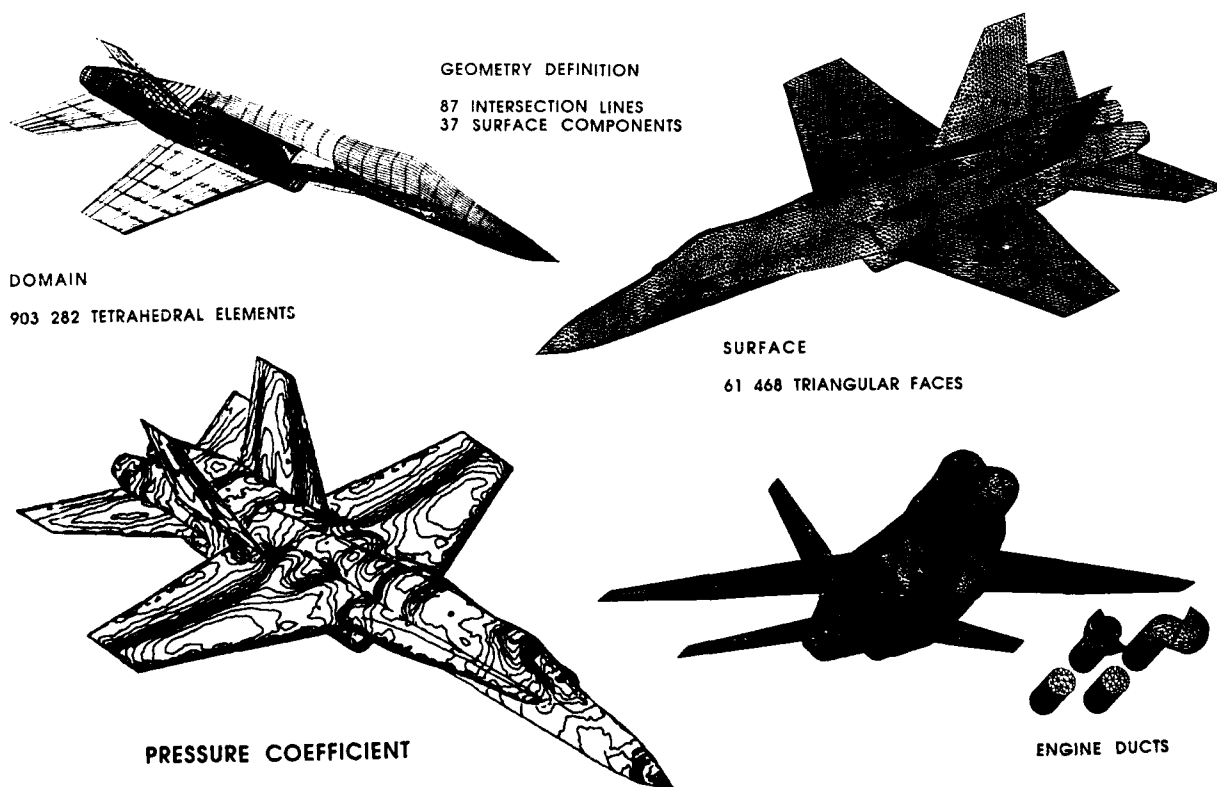
Figure 16.   F-18 fighter configuration.

The error indication procedure is first illustrated in one dimension. Having chosen a key variable θ, we assume that the computed nodal values are exact and then estimate the root mean square value of the local error as

$$E_e RMS = \frac{1}{11} h_e^2 \left| \frac{d^2\theta}{dx^2} \right| \tag{5.1}$$

The second derivative can be estimated, on a mesh of linear elements, by using the technique of variational recovery [14]. Applying the requirement of equi-distribution of the error [18], it follows that the size δ on the new mesh should be computed according to

$$\delta^2 \left| \frac{d^2\theta}{dx^2} \right| = \text{constant} \tag{5.2}$$

When these ideas are extended into two or three dimensions, we encounter a matrix m of second derivatives. The criterion of equi-distribution of the error leads to the requirement

$$\delta_\beta^2 \left| \frac{\partial^2\theta}{\partial\beta_i\partial\beta_j} \right| = \text{constant} \tag{5.3}$$

for the size δ_β in direction β. The local mesh parameters δ_1, δ_2 and δ_3, with corresponding directions α_1, α_2 and α_3, for the new mesh are now obtained by applying equation (5.3) in each of the principal directions of the matrix m in turn. The value adopted for the constant in equation (5.3) is proportional to the level of accuracy desired and hence governs the number of elements that will be present in the new mesh. It is apparent that in regions of uniform flow the computed values of the mesh size will be very large. Practical mesh generation constraints therefore require that the user should specify a maximum allowable value for the local size on the new mesh.

5.2 Adaptive Mesh Regeneration

With the values of the new mesh parameters computed at every node of the current mesh, the mesh generation process is employed again but now using the current mesh as the background mesh. The application of this approach is illustrated in figure 13, which shows the results obtained for the problem of two dimensional inviscid flow at a Mach number of 8.15 past a double ellipse configuration at 30^0 angle of attack.

6. EXAMPLES

In this section we present some examples which demonstrate the application of the techniques described above to the modelling of aerodynamical flows.

6.1 Transonic Wing

The transonic flow past a W4 wing is computed. The free stream Mach number is 0.778 and the angle of attack is 0.52^0. The outer boundary of the computational domain was taken at a distance of 15 root chords. A total of 279,196 tetrahedra is employed to discretise the solution domain. Although the ratio of the maximum to the minimum element size is of the order of several hundred, very little distortion is observed. Figure 14 shows the triangulation on the surface of the wing and the pressure coefficient distribution at various cross sections along the wing span after 2,000 iterations.

6.2 Boeing 747 in Landing Configuration

The flow past a landing Boeing 747 aircraft at a Mach number of 0.3 and 5^0 angle of attack is considered. Figure 15 shows the geometry definition employed which utilises 35 surface components together with the surface triangulation and the pressure distribution computed. The surface is represented by 26,060 triangular faces. The computational domain extends 30 chord lengths and is filled with 388,614 tetrahedral elements of varying size but without distortion. It is noted that the slats and flaps are modelled deployed with the trailing flaps detached from the main wing.

6.3 F-18 Fighter Configuration

The surface geometry of an F-18 configuration is defined in terms of 37 surface components and 87 line components. The body surface is discretised into 61,468 triangular faces and the computational domain is filled with 903,282 tetrahedral elements. A flow simulation was made at a Mach number of 0.9 and an angle of attack of 3^0. Engine inlet conditions took the form of a specified Mach number of 0.4 and a jet pressure ratio of 3 was assumed to determine the outlet conditions. The surface definition and the computed surface pressure contour distributions are shown in figure 16.

6.4 Shock Interference on Cylindrical Leading Edges

The prediction of the aerodynamic heating resulting from shock interaction problems on cylindrical leading edges is of great interest to the designers of hypersonic vehicles. Interactions of this type can lead to highly localised and intense pressures and heat transfer rates [19] which result in stress levels which are a significant hazard to load carrying structures. A two dimensional laminar Navier-Stokes simulation has been attempted using the adaptive remeshing procedure. The undisturbed free stream Mach number is 8.03 and the Reynolds number, based upon the cylinder radius is $1.71*10^5$. The fluid which has been turned by the shock generator enters the computational domain with a Mach number of 5.26. The mesh obtained after two adaptive iterations and the temperature contour distribution are shown in figure 17a. A structured mesh of 32 exponentially stretched layers has been used in the vicinity of the cylinder surface. Interest is now being directed towards a study of a similar problem in three dimensions where the cylindrical leading edge is swept in an attempt to determine whether or not the same behaviour can be expected. The experimental configuration and the domain chosen for the computational simulation
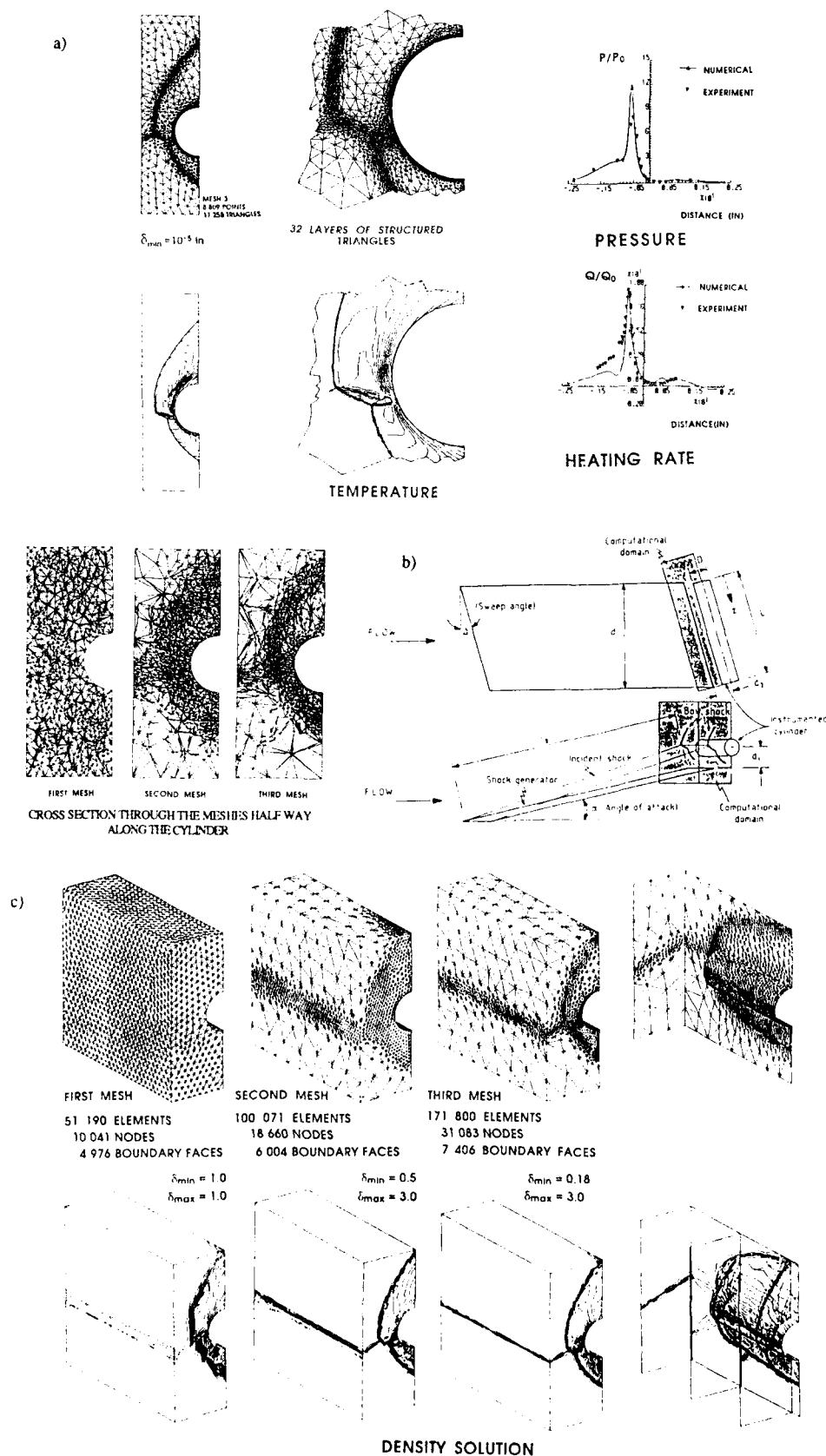
a)

$\delta_{min} = 10^{-5}$ in

32 LAYERS OF STRUCTURED TRIANGLES

PRESSURE

HEATING RATE

TEMPERATURE

CROSS SECTION THROUGH THE MESHES HALF WAY ALONG THE CYLINDER

FIRST MESH   SECOND MESH   THIRD MESH

b)

c)

FIRST MESH
51 190 ELEMENTS
10 041 NODES
4 976 BOUNDARY FACES
$\delta_{min} = 1.0$
$\delta_{max} = 1.0$

SECOND MESH
100 071 ELEMENTS
18 660 NODES
6 004 BOUNDARY FACES
$\delta_{min} = 0.5$
$\delta_{max} = 3.0$

THIRD MESH
171 800 ELEMENTS
31 083 NODES
7 406 BOUNDARY FACES
$\delta_{min} = 0.18$
$\delta_{max} = 3.0$

DENSITY SOLUTION

Figure 17. Shock-interaction on a cylindrical leading edge. a) mesh obtained after two adaptive iterations and the temperature contours, b) schematic of an experimental configuration in which the leading edge is swept, c) initial and adapted meshes with corresponding density contours for an inviscid simulation

FIRST MESH   SECOND MESH

10 STRUCTURED LAYERS
OF TRIANGULAR PRISMS

PRESSURE

PRESSURE

HEATING RATE
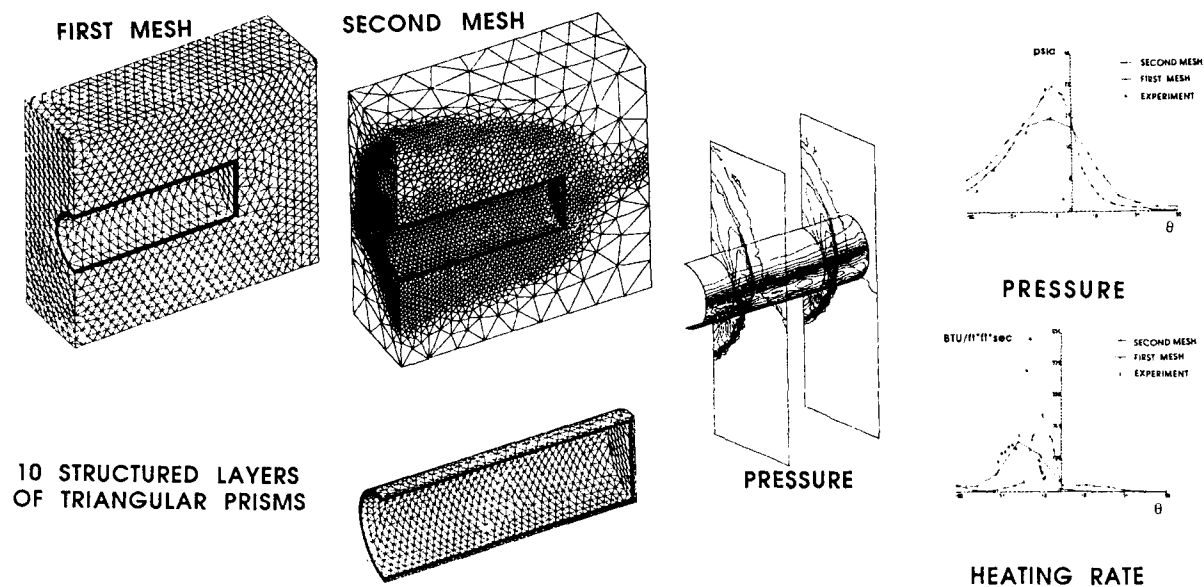
Figure 17 (continued)   Shock-interaction on a cylindrical leading edge. d) initial and adapted meshes with corresponding pressure (C_p) contours and a comparison between experimental and computed surface distributions of pressure and heating rate

FIRST MESH

SECOND MESH

FIRST MESH $\delta_{min} = 3.5$
153 044 ELEMENTS

PRESSURE SOLUTION

PRESSURE SOLUTION

SECOND MESH $\delta_{min} = 1.0$
140 250 ELEMENTS
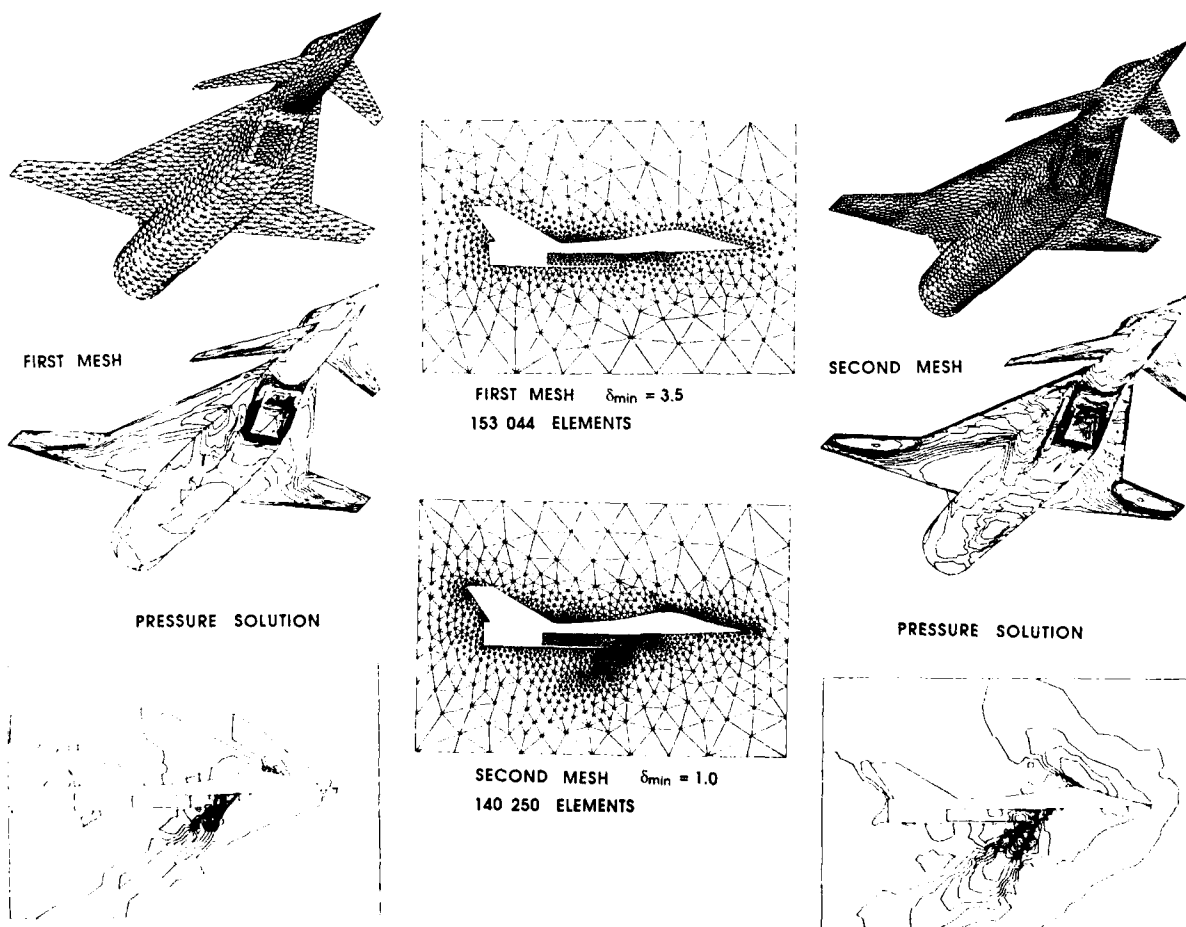
Figure 18.   Adaptive remeshing in three dimensions. Generic fighter configuration.

is shown diagrammatically in figure 17b. An inviscid analysis was performed first and the initial mesh used and the adapted meshes produced, with corresponding computed density contour distribution, are shown in figure 17c. A viscous simulation has been attempted using the initial mesh and adapted mesh shown in figure 17d. The distribution of the pressure (C_p) on selected planes is also shown as is the comparison between experimental and computed surface pressure and heat transfer distributions. In this case ten structured layers of triangular prisms are employed in the vicinity of the cylinder surface.

6.5 Generic Fighter Configuration

In this example the adaptive remeshing procedure is applied for first time to a full aircraft configuration. The geometry considered is that of the generic fighter model used for illustration in section 3 above and which has been studied previously using an algebraic structured mesh generation approach [20]. The flow conditions correspond to a supersonic Mach number of 2 and an angle of attack of 3.79^0 The computational domain considered includes a full simulation of the engine air intake. The engine inlet was modelled by prescribing an engine operational Mach number of 0.3. Supercritical flow conditions were prescribed at the engine outlet. The first mesh used consisted of 153,044 elements and the solution obtained for the pressure on the plane of symmetry and on the surface of the fighter is displayed in figure 18. In the remeshing procedure the density was chosen as the 'key' variable. The regenerated mesh consists of only 140,250 elements with the minimum element size being 3.5 times smaller than in the first mesh. This increase in resolution is apparent by examining the solution on the regenerated mesh which is also shown in figure 18. The surface of the fighter is represented in the regenerated mesh by 14,524 triangular faces, whereas only 8,256 triangular faces were used in the initial mesh.

7. CONCLUSIONS

We have presented an approach for generating unstructured meshes for computational domains of complex shape in both two and three dimensions. An essential feature of the advocated method is that it allows for the incorporation of a solution adaptive mesh procedure in a natural manner. Although the approach is powerful, in that it allows solutions for complete aircraft configurations to be obtained in a timescale of the order of two or three days, certain major problem areas still remain to be resolved. To improve the accuracy of the computed solutions, we need to be able to obtain more control on the quality of the generated mesh and to simultaneously develop solution algorithms which are less sensitive to the mesh quality. To improve the performance of the adaptivity algorithm we need to produce more sophisticated and reliable error indicators, which will enable the distribution of the new mesh parameters to be predicted with a high degree of confidence. While these problems remain, it is worth noting the rapid progress that has recently been made in the use of unstructured meshes for the simulation of aerodynamic flows. If this progress is maintained, it can be expected that unstructured mesh methods will play an increasingly important role in this area in the near future.

REFERENCES

1. A. Jameson, "Successes and challenges in computational aerodynamics", AIAA paper 87-1184, 1987.

2. M. O. Bristeau, O. Pironneau, R. Glowinski, J. Periaux, P. Perrier and G. Poirier, "On the numerical solution of non-linear problems in fluid dynamics by least squares and finite element methods II : Application to transonic flow simulations", Proceedings of the 3rd International Conference on Finite Elements in Nonlinear Mechanics, FENOMECH84, Stuttgart, edited by J. St. Doltsinis, North Holland, 363-394, 1985.

3. A. Jameson, T. J. Baker and N. P. Weatherill, "Calculation of inviscid transonic flow over a complete aircraft", AIAA Paper 86-0103, 1986.

4. T. J Baker, "Three dimensional mesh generation by triangulation of arbitrary point sets", AIAA paper 87-1124, 1987.

5. J. Peraire, J. Peiro, L. Formaggia, K. Morgan and O. C. Zienkiewicz, "Finite element Euler computations in three dimensions", Int. J. Num. Meth. Engng., 26, 2135-2159, 1988.

6. B. Stoufflet, J. Periaux, F. Fezoui and A. Dervieux, "Numerical simulation of 3D hypersonic Euler flows around space vehicles using adapted finite elements", AIAA Paper 87-0560, 1987.

7. R. Löhner, K. Morgan, J. Peraire and O. C. Zienkiewicz, "Finite element methods for high speed flows", AIAA Paper 85-1531-CP, 1985.

8. A. J. George, "Computer implementation of the finite element method", Ph. D. Thesis, Stanford University, STAN-CS-71-208, 1971.

9. J. Peraire, M. Vahdati, K. Morgan and O. C. Zienkiewicz, "Adaptive remeshing for compressible flow computations". J. Comp. Phys., 72, 449-466, 1987.

10. J. Peraire, J. Peiro, K. Morgan and O. C. Zienkiewicz, "Finite element mesh generation and adaptive procedures for CFD", Lecture presented at the GAMNI/SMAI Conference on Automated and Adaptive Mesh Generation, Grenoble, France, 1-2 October 1987.

11. J. Peiro, J. Peraire and K. Morgan, "The generation of triangular meshes on surfaces", Proceedings of the POLYMODEL XII Conference, Newcastle upon Tyne, May 23-24, 1989.

12. J. Bonet and J. Peraire, "An alternate digital tree algorithm for geometric searching and intersection problems", University College of Swansea Report C/R/619/88.

13. D. E. Knuth, " The art of computer programming, Volume 1 : Fundamental Algorithms", 2nd Edition, Addison Wesley Pub. Co., 1973.

14.   K. Morgan and J. Peraire. "Finite element methods for compressible flows" von Karman Institute for Fluid Dynamics, Lecture series 1987 - 04, 1987.

15.   L. Formaggia, J. Peraire, K. Morgan and J. Peiro, "Implementation of a 3D explicit Euler solver on a CRAY computer", Proceedings of the 4th International Symposium on Science and Engineering on CRAY Supercomputers, 45-65, Minneapolis, 1988.

16.   O. Hassan, K. Morgan and J. Peraire, "An adaptive implicit/explicit finite element scheme for compressible viscous high speed flows", AIAA Paper 89-0363, 1989.

17.   P. G. Ciarlet, "The finite element method for elliptic problems", North holland, 1978.

18.   J. T. Oden, "Grid optimisation and adaptive meshes for finite element methods", University of Texas at Austin Notes, 1983.

19.   A. R. Wieting, "Experimental study of shock wave interference heating on a cylindrical leading edge", NASA TM 100484, 1987.

20.   L.-E. Eriksson, R. E. Smith, M. R. Wiese and N. Farr, "Grid generation and inviscid flow computation about cranked-winged airplane geometries", AIAA Paper 87-1125, 1987.

ACKNOWLEDGEMENTS

GENERATION AND ADAPTATION OF 3-D UNSTRUCTURED GRIDS FOR TRANSIENT PROBLEMS

Rainald Löhner

School of Engineering and Applied Science
George Washington University
Washington, D.C. 20052, USA

SUMMARY

We describe grid generation and adaptive refinement techniques suitable for the simulation of strongly unsteady flows past geometrically complex bodies in 3-D. The grids are generated using the advancing front technique. Emphasis is placed not to generate elements that are too small, as this would severely increase the cost of simulations with explicit flow solvers. The grids are adapted to an evolving flowfield using simple h-refinement. A grid change is performed every 5-10 timesteps, and only one level of refinement/coarsening is allowed per mesh change.

1. INTRODUCTION

While the development of numerical algorithms to solve the Euler and Navier-Stokes equations has reached a considerable degree of maturity, the development of grid generation schemes has lagged behind. Currently, this area is the pacing item that precludes the full exploitation of CFD for engineering design and analysis. The methods proposed so far for the generation of unstructured grids may be grouped into two families:
  a) those that fill 'empty', i.e. not yet gridded space, and
  b) those that improve an existing grid, starting from a grid that only considers boundary points.
The first family of methods are the so-called advancing front algorithms [1-6], whereby empty space is filled by the introduction of elements. The second family of methods are the so-called generalized Voronoi algorithms [7-16], whereby an existing grid is improved by the introduction of points. Besides the choice of how to grid 3-D space, one must provide information as to how the element size, stretching and stretching directions, or equivalently the point distribution should vary in space. Two main families of methods may be identified:
  a) schemes that generate the point distributions before gridding the domain, and
  b) schemes that generate the point distributions while gridding the domain.
For the first family of methods, points have been generated from cartesian point distributions with or without embedding [1,10], random point distributions with or without embedding [2,7-9,11-13], and point distributions obtained from overlapping algebraic grids [14,15]. For the second family of methods, point distributions have been generated based on background grids [3-5], or according to some desired grid quality not yet achieved by the current grid [6,16]. Within an adaptive remeshing context [25-27], the background grid concept clearly offers the greatest flexibility. However, for the generation of a first mesh one may prefer one of the other schemes.

A second, separate issue that must be addressed is the technique used to define the domain to be gridded. More specifically: the definition of the surfaces surrounding the flow fields. Among the many techniques available, we mention: analytic definition (for simple geometries), Bezier-patches, and transfinite mappings. Our experience indicates that the input

of the surface-defining information typically takes a large portion of the total grid-generation time. It is man-hour intensive and difficult to automate. The advent of powerful graphics workstations, such as the IRIS-4D, has made it possible to considerably reduce the time required to input the information necessary for the generation of three-dimensional grids [17]. This is accomplished by providing the user with rapid and clear visualization of input and output. At the same time, error checking is provided in a natural way.

Finally, the third issue that must be addressed is the grid adaptation to the solution. Particularly for transient problems involving shock impact and reflection, the element size needed to achieve high accuracy changes with the position of the discontinuities. It would be extremely wasteful to use an overall fine grid. Therefore, we adapt the mesh to the solution as it evolves in time. We have found that classic h-refinement [20-24] works best for strongly unsteady flows, where a grid change is required every 5-10 timesteps. On the other hand, for slowly unsteady flows, we prefer adaptive remeshing [25-27].

2. THE GRID GENERATOR: ADVANCING FRONT WITH BACKGROUND GRID

The main algorithmic steps required to generate a grid using the advancing front method with point distributions imposed by a background grid are as follows:

F.1 Define the boundaries (surfaces) of the domain to be gridded. In the current approach, this is done with surface patches. At the same time, find the intersection lines of these surface patches.

F.2 Set up a background grid to define the spatial variation of the size, the stretching, and the stretching direction of the elements to be generated. The background grid consists of tetrahedrons. At the nodes define the desired element size, stretching and stretching direction. This background grid must completely cover the domain to be gridded.

F.3 Using the information stored on the background grid, form sides along the lines where surface patches intersect.

F.4 Using the information stored on the background grid, and the sides formed along the lines connecting surface patches, generate faces on each surface patch. The assembly of all these faces yields the initial front. At the same time, find the generation parameters (element size, stretching and stretching direction) for these faces from the background grid.

F.5 Select the next face to be deleted from the front; in order to avoid large elements crossing over regions of small elements, the face forming the smallest new element is selected as the next face to be deleted from the list of faces.

F.6 For the face to be deleted:

F.6.1 Select a 'best point' position for the introduction of a new point IPNEW.

F.6.2 Determine whether a point exists in the already generated grid that should be used in lieu of the new point. If there is such a point, set this point to IPNEW and continue searching (go to F.6.2).

F.6.3 Determine whether the element formed with the selected point IPNEW does not cross any given faces. If it does, select a new point as IPNEW and try again (go to F.6.3).

F.7 Add the new element, point, and faces to their respective lists.

F.8 Find the generation parameters for the new faces from the background grid.

F.9 Delete the known faces from the list of faces.

F.10 If there are any faces left in the front, go to F.5.

There are several interesting algorithmic aspects which should be mentioned:

a) Extensive use is made of optimal data structures to perform the search operations involved. In particular, we use heap-lists to find the next face to be deleted (step F.5), quad-trees to find the closest given points to a new point (step F.6.2), and linked lists to find the faces adjacent to a given point (step F.6.3). We combine quad-trees and linked lists to find for any given location the values of generation parameters from the background grid (steps F.3,F.4 and F.8). The algorithmic complexity of the overall algorithm should be of $O(N \log N)$. In practice, we find it to be closer to $O(N)$, as we continuously delete domain points from the lists, and the subroutine-calls require some overhead.

b) The checking of face-intersections is a non-trivial problem in 3-D. It also requires a large amount of CPU-time due to the algorithmic complexity involved. In order to make this process faster, a layered face-checking approach was implemented [5]. This resulted in a significant reduction of CPU time.

c) The current version of the 3-D grid generator code runs at about 450-500 tetrahedra/second on a CRAY-2S. Thus, it takes about 40 minutes to generate a grid of one million tetrahedra. Such a grid is deemed adequate for most standard Euler calculations.

d) In order to speed up the grid generation process further, we use global h-refinement. This is accomplished by subdividing each tetrahedron into eight smaller ones. As this operation is totally vectorizable, it poses no overhead for the grid generator. Grid generation times are thus reduced by a factor of eight, making it possible to generate a grid of one million tetrahedra in approximately 5 minutes.

e) Most of the current shock-interaction flow codes advance the solution in time using explicit schemes. Therefore, it is important not to create even a single element that is too small. To this end, we have developed post-processing routines that delete from the mesh all elements that are too small.

3. A VORONOI ALGORITHM WITH BACKGROUND GRID

In certain applications (e.g. simple finite-volume Maxwell-solvers), one may desire to guarantee that the generated grid represents a Delauney triangulation. We can also use the background-grid concept to obtain point-distributions in conjunction with Delauney triangulations. All that is required is to modify the algorithm described above as follows:

F.5 Select the next element to introduce a point; it makes good sense to introduce a point for the element that shows the highest discrepancy between the actual and the desired element size and shape.

F.6 For the element where a point is to be introduced:

F.6.1 Introduce the new point IPNEW.

F.6.2 Determine the tetrahedra that have to be broken up in order to recover a Delauney triangulation.

F.8 Find the desired element size and shape for the newly formed tetrahedra from the background grid.

4. ADAPTIVE REFINEMENT

Besides their ability to discretize accurately complex geometries, a second very attractive feature of unstructured grids is the ease with which adaptive refinement can be incorporated into them. The addition of further degrees of freedom does not destroy any previous structure. Thus, the flow solver requires no further modification when operating on an adapted grid. For many practical problems, the regions that need to be refined are extremely small as compared to the overall domain. Therefore, the savings in storage and CPU-requirements typically range between 10-100 as compared to an overall fine mesh [22,23]. Our experience indicates that for the majority of the daily production-type runs, adaptive refinement makes the difference between being or not being able to run the problems to an acceptable accuracy in a reasonable

time [23]. Without it, we would be forced to use much coarser grids, with lower accuracy, for the same expense.

Any adaptive refinement scheme is composed of three main ingredients. These are
1) an optimal-mesh criterion,
2) an error indicator, and
3) a method to refine and coarsen the mesh.

They give answers to the questions
1) how should the optimal mesh be ?,
2) where is refinement/ coarsening required ?, and
3) how should the refinement/ coarsening be accomplished ?

Many variants of each of these subtopics have been explored and shown to be useful for a certain class of problems [18,19]. Here, we seek a method that is efficient and reliable for transient compressible flow problems. This leads us to the following design criteria for the error indicator:
a) The error indicator should be fast.
b) The error indicator should be dimensionless, so that several 'key variables' can be monitored at the same time.
c) The error indicator should be bounded, so that no further user intervention becomes necessary as the solution evolves.
d) The error indicator should not only mark the regions with strong shocks to be refined, but also weak shocks, contact discontinuities and other 'weak features' in the flow.

For the refinement method, the design criteria are as follows:
e) The method should be conservative, i.e. a mesh change should not result in the production or loss of mass, momentum or energy.
f) The method should not produce elements that are too small, as this would reduce too severely the allowable timestep of the explicit flow solvers employed.
g) The method should be fast. In particular, it should lend itself to some degree of parallelism.
h) The method should not involve a major storage overhead.

4.1 The Error Indicator

An error indicator that meets the design criteria a)-d) was proposed in [22]. In general terms, it is of the form

$$error = \frac{h^2 \, |second \ derivatives|}{h \, |first \ derivatives| + \epsilon \, |mean \ value|} \ .$$

By dividing the second derivatives by the absolute value of the first derivatives the error indicator becomes bounded, dimensionless, and the 'eating up' effect of strong shocks is avoided. The terms following ϵ are added as a 'noise' filter in order not to refine 'wiggles' or 'ripples' which may appear due to loss of monotonicity. The value for ϵ thus depends on the algorithm chosen to solve the PDEs describing the physical process at hand. The multidimensional form of this error indicator is given by

$$E^I = \sqrt{\frac{\sum_{k,l}(\int_\Omega N^I_{,k} N^J_{,l} d\Omega \cdot U_J)^2}{\sum_{k,l}(\int_\Omega |N^I_{,k}| \left[|N^J_{,l} U_J| + \epsilon \left(|N^J_{,l}||U_J|\right)\right] d\Omega)^2}}, \tag{1}$$

where N^I denotes the shape-function of node I. After having determined the values of the error indicators in the elements, all elements lying above a preset threshold value CTORE are refined, while all elements lying below a preset threshold value CTODE are coarsened.

4.2 Adaptive Refinement Method

Extensive experience in 2-D indicates that the only two refinement methods that are truly general and efficient for the class of problems considered here are h-refinement [20-24] and remeshing [25-27]. However, for strongly unsteady problems, where a new grid is required every 5-10 timesteps, local h-refinement seems to be preferable. Several reasons can be given for this choice. Firstly, h-refinement is easy to implement and maintain. Secondly, h-refinement is very well suited to vector- and parallel processors. This is of particular importance in the present context, where a mesh change is performed every 5-10 timesteps. Thirdly, conservation presents no problem for h-refinement.

In order to obtain an algorithm that is as simple and fast as possible, we limit the number of refinement/ coarsening levels per mesh change to one. Moreover, we only allow refinement of a tetrahedron into two (along a side), four (along a face) or eight new tetrahedra. We call these tetrahedra 1:2, 1:4 and 1:8 tetrahedra or refinement cases respectively. At the same time, a 1:2 or 1:4 tetrahedron can only be refined further to a 1:4 tetrahedron, or by first going back to a 1:8 tetrahedron with subsequent further refinement of the 8 sub-elements. We call these the 2:4, 2:8+ and 4:8+ refinement cases. The refinement cases are summarized in Figure 1. This restrictive set of refinement rules avoids ill-deformed elements, and considerably simplifies the grid logic. An interesting phenomenon that does not appear in 2-D is the apparently free choice of the inner diagonal for the 1:8 refinement case. As shown in Figure 2, we can place the inner four elements around the inner diagonals 5-10, 6-8, or 7-9. In the present case, the shortest inner diagonal was chosen. This choice produces the smallest amount of distorted tetrahedra in the refined grid. When coarsening, we again only allow a limited number of cases that are compatible with the refinement. Thus, the coarsening cases become 8:4, 8:2, 8:1, 4:2, 4:1, 2:1. These coarsening cases are summarized in Figure 3.

4.3 Algorithmic Implementation

One complete grid change requires algorithmically the following five steps:

1) Construction of the missing grid information needed for a mesh change.
2) Identification of the elements to be refined.
3) Identification of the elements to be deleted.
4) Refinement of the grid where needed.
5) Coarsening of the grid where needed.

4.3.1 Construction of Missing Grid Information

The missing information consists of the sides of the mesh and the sides adjoining each element. The sides are dynamically stored in two arrays, one containing the two points each side connects and the other one (a pointer-array) containing the lowest side-number reaching out of a point. The formation of these two arrays is accomplished in three main loops over the elements, which are partially vectorizable. After having formed these two side-arrays, a further loop over the elements is performed, identifying which sides belong to each element.

4.3.2 Identification of Elements to be Refined

The aim of this sub-step is to determine on which sides further gridpoints need to be introduced. To this end, we first determine - using the modified error indicator given by eqn.(1) and the prescribed refinement tolerance CTORE - those elements that require further refinement. Thereafter, if desired, protective layers of elements are to be added ahead of the feature to be refined. After having identified the elements to be refined, we delete from the list of elements to be refined those elements which are already too small (if a minimum allowed element size has been given), or have already been refined too many times (if a maximum allowed number of refinement levels has been prescribed).

With the side/element information obtained in sub-step 4.3.1, we can now determine a first set of sides on which new gridpoints need to be introduced. This set of sides is still preliminary, as we only allow certain types of refinement. Therefore, special logic is incorporated for the 1:2 and 1:4 tetrahedra in order to obtain the allowable refinement cases shown in Figure 1. We then perform as many loops as required (usually two or three) over the elements, adding further sides whenever an unallowed refinement case appears. This then yields the final set of sides on which new gridpoints are introduced.

4.3.3 Identification of Elements to be Deleted

As before, we start by determining - using the modified error indicator given by eqn.(1) and the prescribed deletion tolerance CTODE - those elements that should be coarsened. Thereafter, only the parent elements to be coarsened are considered further. This set of elements is still preliminary, as we only allow certain types of coarsening. Therefore, special logic is incorporated in order to obtain the allowable coarsening cases shown in Figure 3. While the refinement logic yielded a set of sides where new gridpoints are to be introduced, the coarsening logic yields a set of points to be deleted. We call this set of points the 'total deletion points'.

4.3.4 Refinement of the Grid Where Needed

The introduction of further points and elements is performed in two independent steps, which in principle could be performed in parallel.

To add further points, the sides marked for refinement in sub-step 4.3.2 are grouped together. For each of these sides a new grid-point will be introduced. The interpolation of the coordinates and unknowns is then performed using the side/point information obtained in sub-step 4.3.1. These new coordinates and unknowns are added to their respective arrays. In the same way new boundary conditions are introduced where required.

In order to add further elements, the sides marked for refinement are labelled with their new gridpoint-number. Thereafter, the element/side information obtained in sub-step 4.3.1 above is employed to add the new elements. The elements to be refined are grouped together according to the refinement cases shown in Figure 1. Each case is treated in block fashion in a separate subroutine. Perhaps the major coding breakthrough was the reduction of the many possible refinement cases to only six. In order to accomplish this, some information for the 2:8+ and the 4:8+ cases is stored ahead in scratch arrays. After these elements have been refined according to the 2:8 and 4:8 cases, their sons are screened for further refinement using this information. All sons that require further refinement are then grouped together as 1:2 or 1:4 cases, and processed in turn.

4.3.5 Coarsening of the Grid Where Needed

The deletion of points and elements is again performed in two independent steps, which in principle could be performed in parallel.

The points to be deleted having been marked in sub-step 4.3.3 above, all that remains to be done is to fill up the voids in the coordinate-, unknown- and boundary condition-arrays by renumbering points and boundary conditions.

The deletion of elements is again performed blockwise, by grouping together all elements corresponding to the coarsening cases shown in Figure 3. Thereafter, the elements are also renumbered (in order to fill up the gaps left by the deleted elements), and the point-renumbering is taken into consideration within the connectivity-arrays.

5. NUMERICAL EXAMPLES

5.1 Spherical Blast-Wave: The problem statement, as well as the solutions obtained are shown in Figure 4. An octant of a cube in the lower left hand corner was given a density of 10.0 and a pressure of 40.0, while the rest of the computational region was filled with density 1.0 and pressure 1.0. Because all grid points inside radius 5.1 were disturbed and all gridpoints outside were not, the surface of the cylinder on the finite element grid is not completely circular. The number of refinement levels allowed in this case was NREMX=2 which would correspond to a regular grid of 6*32*32*32=196,608 elements. This case was run to test the symmetry or 'circularity' of the numerical solution. Figure 4a shows the initial grid, and the solution at time T=0.0. Figure 4b shows the solution at time T=4.4. At this time, the mesh has increased to NPOIN=2,894 points and NELEM=14,112 elements. The 2-D equivalent of this case shows no implosion effect, whereas in the present case at later times we observe a negative radial velocity (pointing towards the origin). Figure 4c shows the solution at time T=7.3. The mesh now consists of NPOIN=5,068 points and NELEM=26,006 elements.

5.2 Shock-Locomotive Interaction: The problem statement, as well as the solution at time T=5.0 are shown in Figures 5(a-d). A weak shock ($M_s = 1.4$) interacts with a diesel-electric locomotive head-on. The grid consisted of NELEM=261,865 elements and NPOIN=47,730 points. Figures 5a,b show two views of the surface mesh from different angles. As this is a shock-object interaction case run without adaptive refinement, a fairly uniform grid was employed. Figures 5c,d show two views of the surface pressure at time $T = 1.4$. The main shock has reflected from the lower front portion of the locomotive, and is about to reflect from the top front portion. The total run-time required on the CRAY-2 for this run was approximately 2 hours. The overall turnaround time was of the order of two days: about 3 hours to sketch, input and edit the surface definition, 1 hour for background grid input and distance parameter distribution trials, five minutes to generate the mesh on the CRAY-2, 2 hours to run the calculation. The rest of the time was spent waiting in queues, transmitting files back and forth between the workstations and the CRAY-2, and plotting.

5.3 Compression Corner: The problem statement, as well as the solutions obtained are shown in Figure 6. Supersonic flow at $M_\infty = 3.0$ interacts with two wedges that are positioned perpendicular to each other. Each wedge by itself would produce an attached shock at the leading edge. The two shocks produced by the wedges interact with each other, producing a significant overpressure. The number of refinement levels allowed in this case was NREMX=2 which would correspond to a regular grid of 64*11,197=716,608 elements. Figure 6a shows the initial grid, and the corresponding steady-state solution. Observe that the interaction region of the shocks is decribed poorly due to the lack of grid resolution. This first mesh was then refined once, resulting in a grid of NELEM=37,823 elements and NPOIN=7,312 points. After converging the solution on this mesh to steady-state, a second refinement was invoked. Figure 6b shows this grid after the second refinement, and the corresponding steady-state solution. This final mesh contains NELEM=191,030 elements and NPOIN=35,007 points. Observe how the interaction region of the shocks becomes more and more defined as the grid is refined further.

6. CONCLUSIONS

Fast, general, user-friendly grid generation and adaptive refinement represent major ingredients in any design and analysis capability. While we have witnessed major developments in both areas, one can identify the following shortcomings at the present time:

For the grid generators:

- A more direct link to the CAD data bases to reduce data conversion times.
- Improved interactive tools to reduce input times.
- Better ways to input background grids or other element-size defining information.

- More control over the resulting tetrahedral elements, in particular shape.
- Better smoothing schemes for tetrahedral meshes.
- Improved display capabilities for unstructured grids.

For adaptive refinement methods:

- Error indicators for viscous flows.
- Refinement methods for transient, viscous flows with separation.
- Combinations of h-refinement and remeshing for strongly unsteady flows with moving bodies.

7. REFERENCES

[1] S.H. Lo - A New Mesh Generation Scheme for Arbitrary Planar Domains; *Int. J. Num. Meth. Eng.* 21, 1403-1426 (1985).

[2] N. van Phai - Automatic Mesh Generation with Tetrahedron Elements; *Int. J. Num. Meth. Eng.* 18, 237-289 (1982).

[3] J. Peraire, J. Peiro, L. Formaggia, K. Morgan and O.C. Zienkiewicz - Finite Element Euler Computations in Three Dimensions; AIAA-88-0032 (1988).

[4] R. Löhner - Some Useful Data Structures for the Generation of Unstructured Grids; *Comm. Appl. Num. Meth.* 4, 123-135 (1988).

[5] R. Löhner and P. Parikh - Three-Dimensional Grid Generation by the Advancing Front Method; *Int. J. Num. Meth. Fluids* 8, 1135-1149 (1988).

[6] F. Huet - Generation de Maillage Automatique dans les Configurations Tridimensionelles Complexes utilization d'une Methode de 'Front'; Paper presented at the 64th AGARD Fluid Dynamics Meeting, Loen, Norway, May 1989.

[7] A. Bowyer - Computing Dirichlet Tesselations; *The Computer Journal* 24,2, 162-167 (1981).

[8] D.F. Watson - Computing the N-Dimensional Delaunay Tesselation with Application to Voronoi Polytopes; *The Computer Journal* 24,2, 167-172 (1981).

[9] M. Tanemura, T. Ogawa and N. Ogita - A New Algorithm for Three- Dimensional Voronoi Tesselation; *J. Comp. Phys.* 51, 191-207 (1983).

[10] M.A. Yerry and M.S. Shephard - Automatic Three-Dimensional Mesh Generation by the Modified-Octree Technique; *Int. J. Num. Meth. Eng.* 20, 1965-1990 (1984).

[11] D.N. Shenton and Z.J. Cendes - Three-Dimensional Finite Element Mesh Generation Using Delaunay Tesselation; *IEEE Trans. on Magnetics*, MAG-21, 2535-2538 (1985).

[12] J.L. Coulomb, Y. du Terrail and G. Meunier - FLUX3D, a Finite Element Package for Magnetic Computation; *IEEE Trans. on Magnetics*, MAG-21, 2499-2502 (1985).

[13] J.C. Cavendish, D.A. Field and W.H. Frey - An Approach to Automatic Three-Dimensional Finite Element Mesh Generation; *Int. J. Num. Meth. Eng.* 21, 329-347 (1985).

[14] T.J. Baker - Three Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets; AIAA-87-1124-CP (1987).

[15] D. Mavriplis - Adaptive Mesh Generation for Viscous Flows Using Dalauney Traingulation; pp. 611-620 in Numerical Grid Generation in Computational Fluid Dynamics (Sengupta et al. eds.), Pineridge Press (1988).

[16] D.G. Holmes and D.D. Snyder - The Generation of Unstructured Triangular Meshes Using Delaunay Triangulation; pp. 643-652 in Numerical Grid Generation in Computational Fluid Dynamics (Sengupta et al. eds.), Pineridge Press (1988).

[17] R. Löhner, P. Parikh and C. Gumbert - Interactive Generation of Unstructured Grids for Three Dimensional Problems; pp. 687-698 in Numerical Grid Generation in Computational Fluid Dynamics (Sengupta et al. eds.), Pineridge Press (1988).

[18] I. Babuska, J. Chandra and J.E. Flaherty (eds.) - *Adaptive Computational Methods for Partial Differential Equations*; SIAM Philadelphia (1983).

[19] I. Babuska et.al. (eds.) - *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*; J. Wiley and Sons (1986).

[20] F. Angrand, V. Billey, A. Dervieux, J. Periaux, C. Pouletty and B. Stoufflet - 2-D and 3-D Euler Flow Calculations with a Second-Order Accurate Galerkin Finite Element Method; AIAA-85-1706 (1985).

[21] R. Shapiro and E.M. Murman - Adaptive Finite Element Methods for the Euler Equations; AIAA-88-0034 (1988).

[22] R. Löhner - An Adaptive Finite Element Scheme for Transient Problems in CFD; *Comp. Meth. Appl. Mech. Eng.* 61, 323-338 (1987).

[23] J.D. Baum and R. Löhner - Numerical Simulation of Shock-Elevated Box Interaction Using an Adaptive Finite Element Shock Capturing Scheme; AIAA-89-0653 (1989).

[24] R. Löhner - Adaptive H-Refinement on 3-D Unstructured Grids for Transient Problems; AIAA-89-0365 (1989).

[25] J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz - Adaptive Remeshing for Compressible Flow Computations; *J. Comp. Phys.* 72, 449-466 (1987).

[26] O. Hassan, K. Morgan and J. Peraire - An Adaptive Implicit/Explicit Finite Element Scheme for Compressible Viscous High Speed Flows; AIAA-89-0363 (1989).

[27] R. Löhner - An Adaptive Finite Element Solver for Transient Problems with Moving Bodies; *Comp. Struct.* 30, 303-317 (1988).

[28] P. Parikh, R. Löhner, C. Gumbert and S. Prizadeh - Numerical Solutions on a PATHFINDER and other configurations Using Unstructured Grids and a Finite Element Solver; AIAA-89-0362 (1989).

ACKNOWLEDGEMENTS
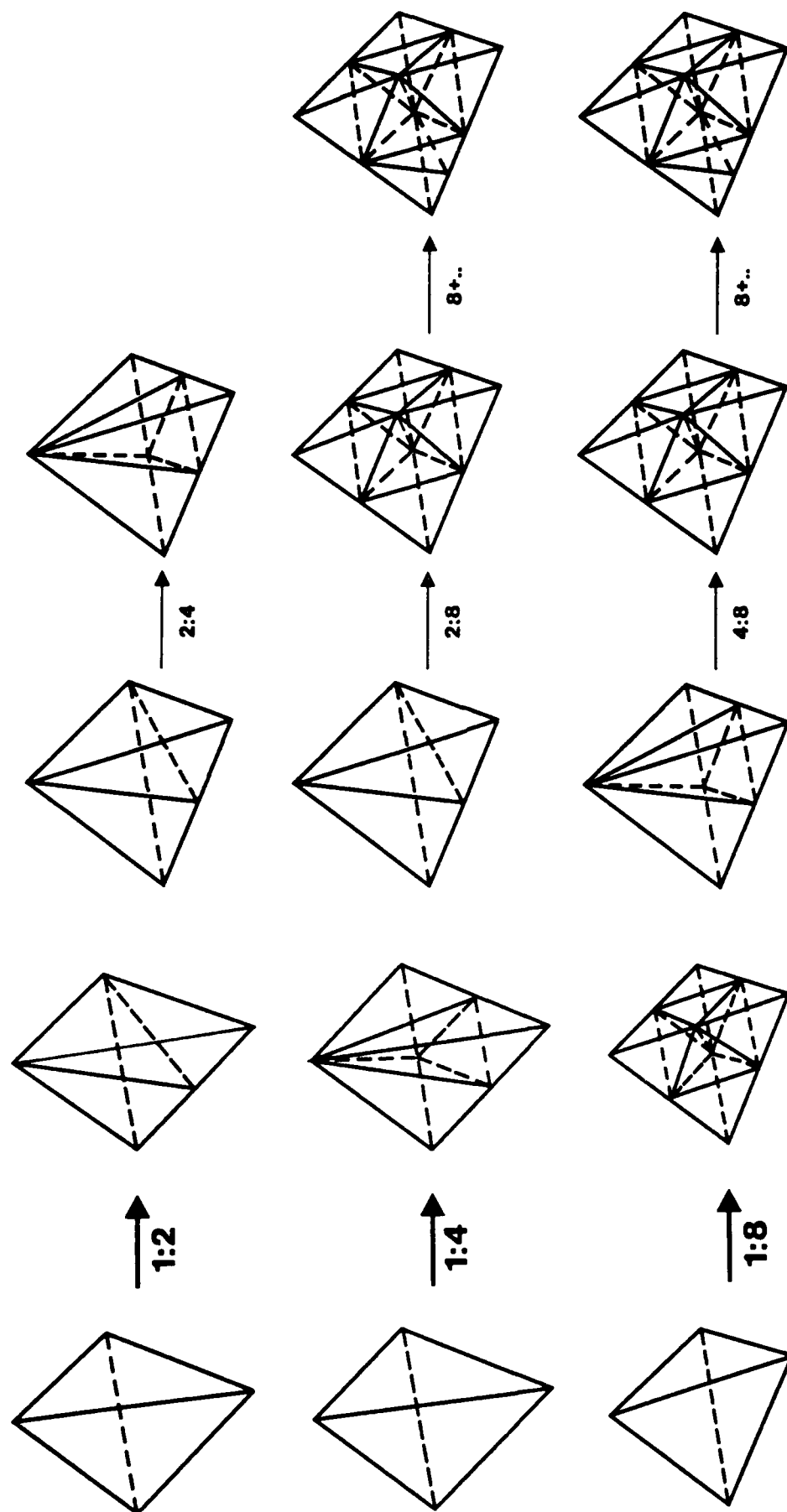
Figure 1: Refinement Cases
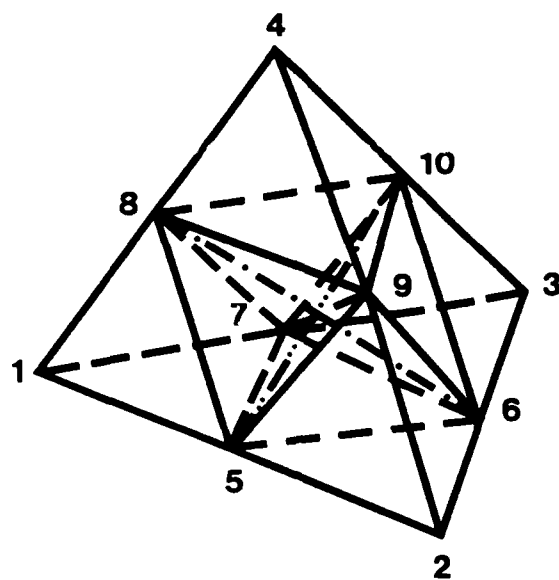
Figure 2: Possible Choices for 'Inner Diagonals'
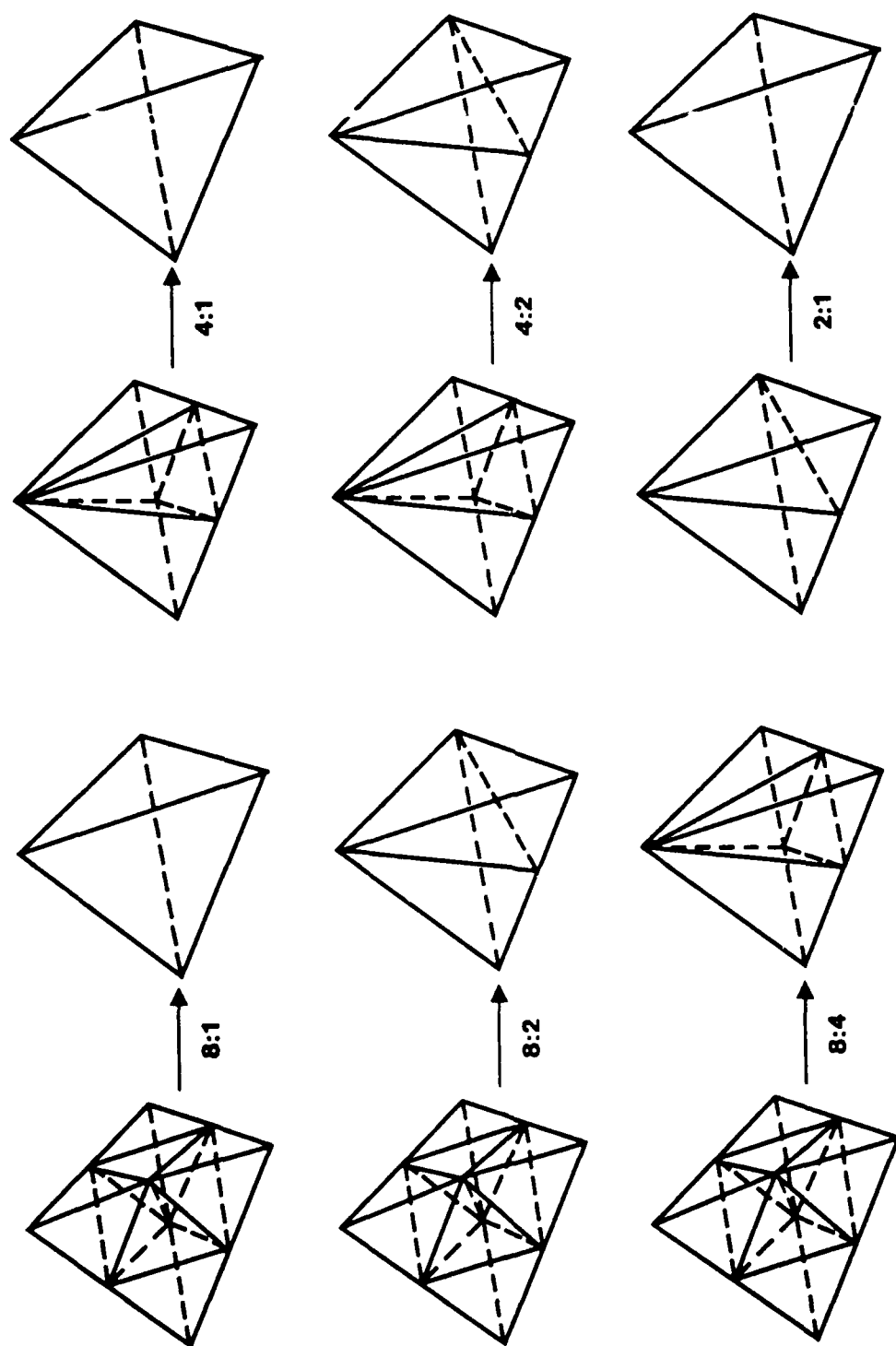
Figure 3: De-Refinement Cases
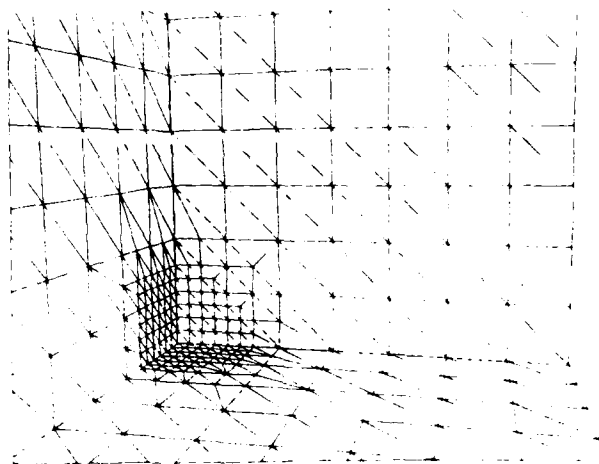
Figure 4a: Spherical Blast Wave. Solution at time T=0.0.
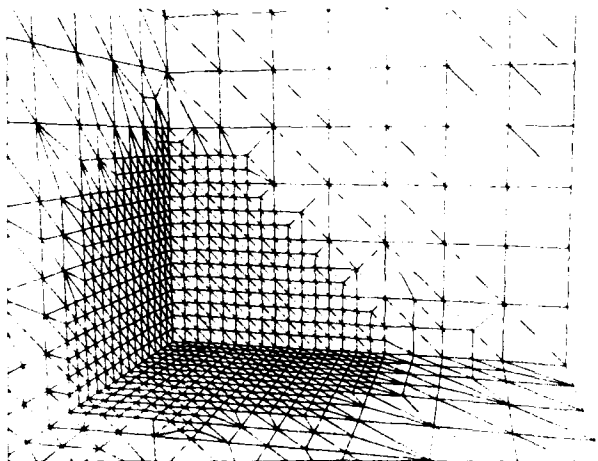
Pressure on the surface.

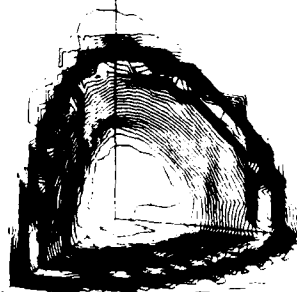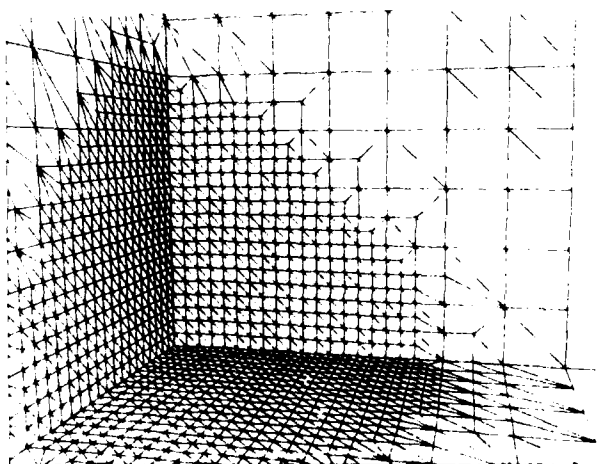Figure 4b: Spherical Blast Wave. Solution at time T=4.4.

Pressure on the surface.

Figure 4c: Spherical Blast Wave. Solution at time T=7.8.
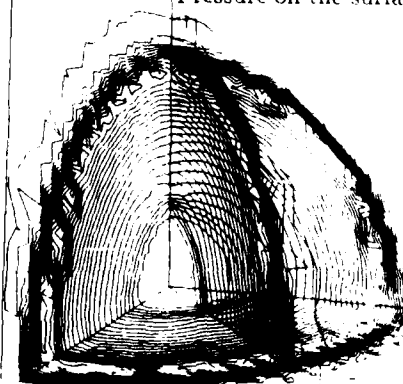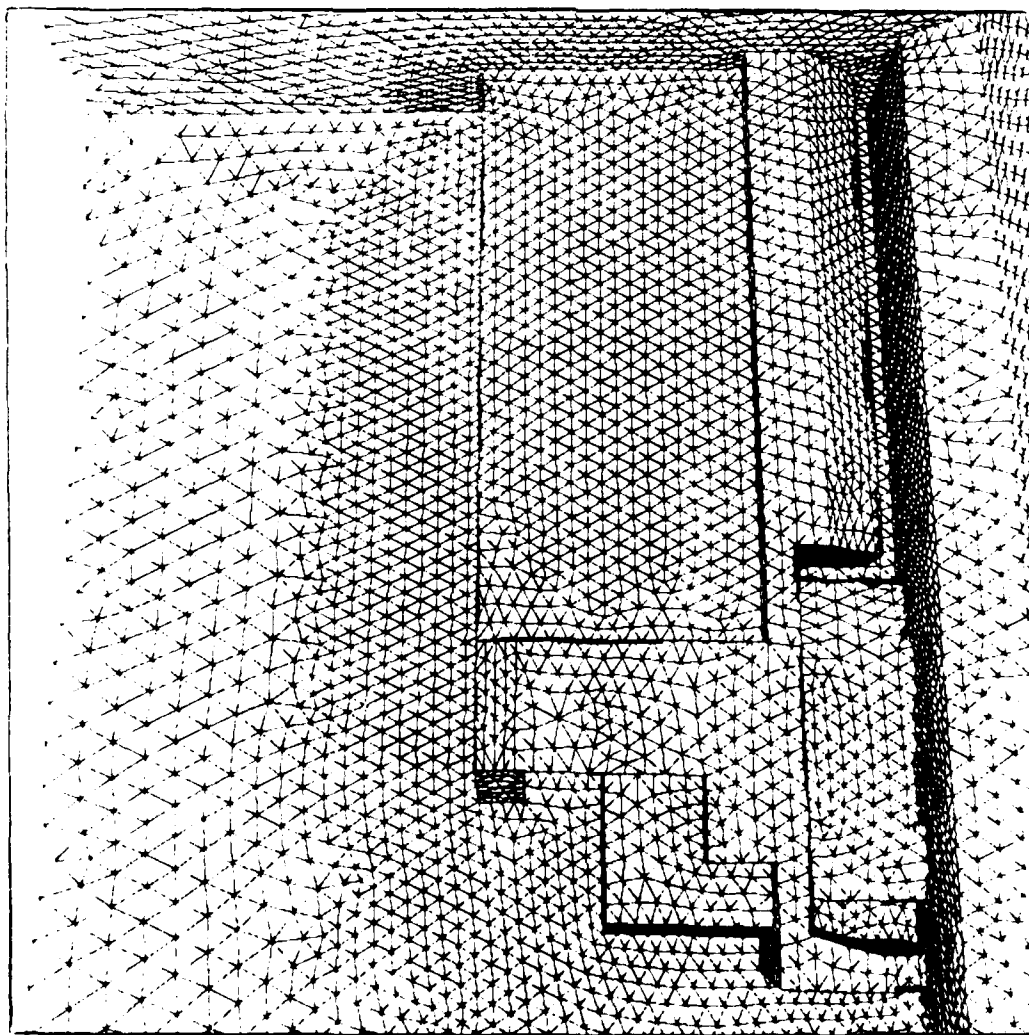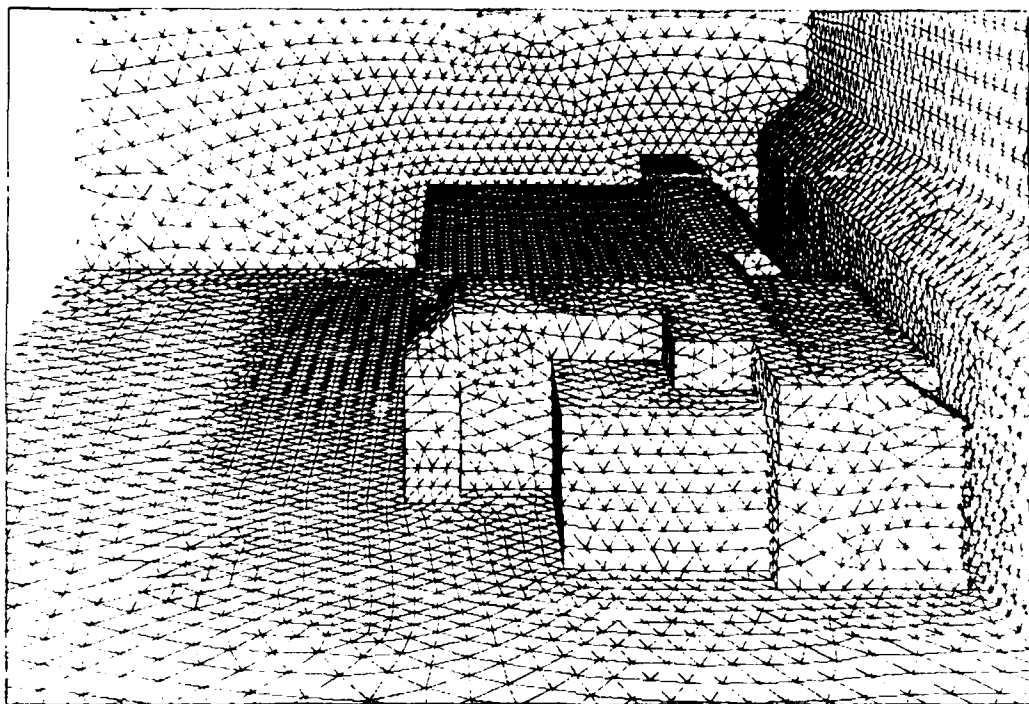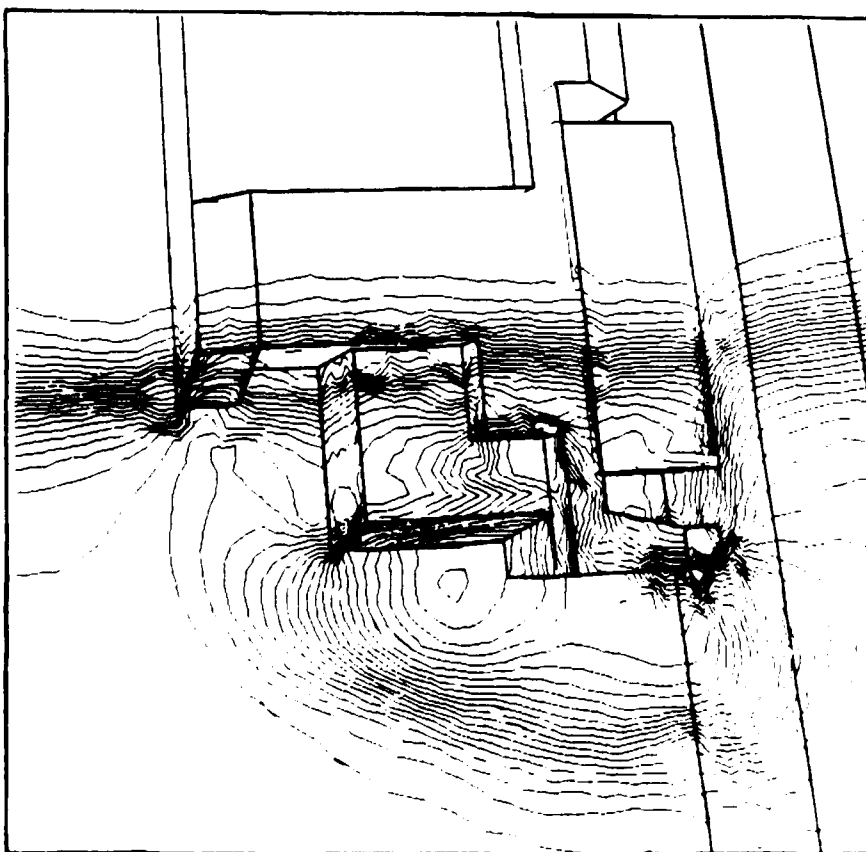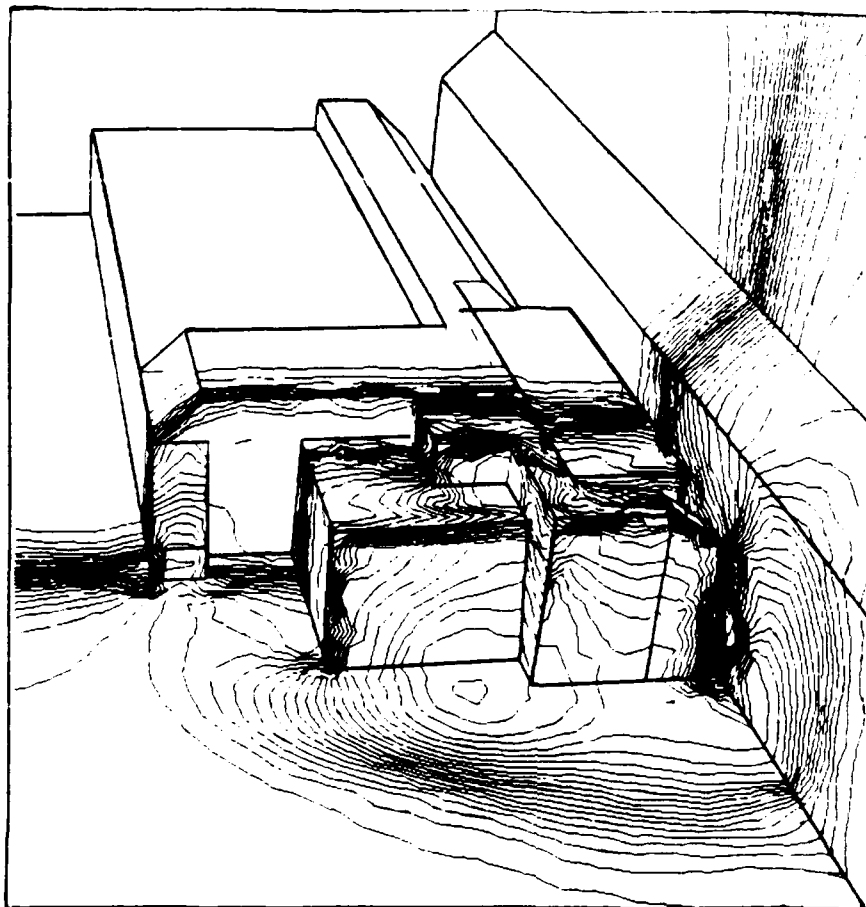
Pressure on the surface and a mid-plane

b) Surface Mesh

a) Surface Mesh

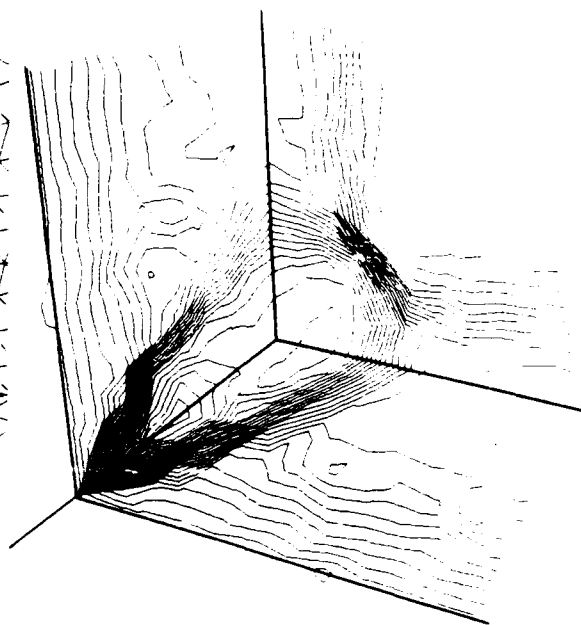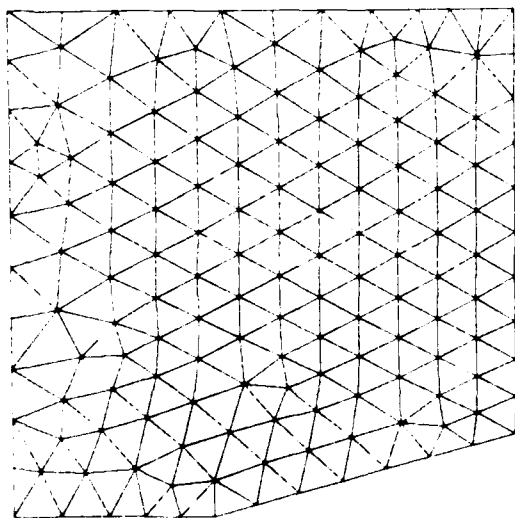Figure 5: Shock-Locomotive Interaction

d) Surface Pressure

c) Surface Pressure

Figure 5: Shock-Locomotive Interaction
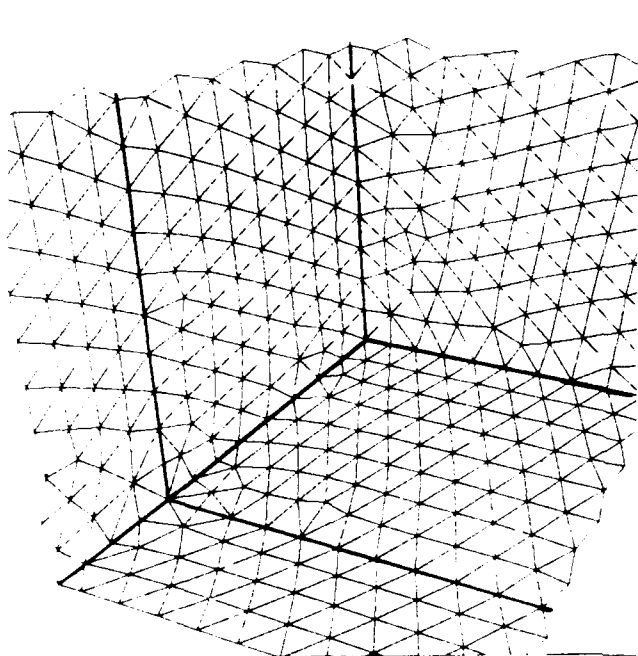
Surface mesh at the corner.

Pressure at the corner.

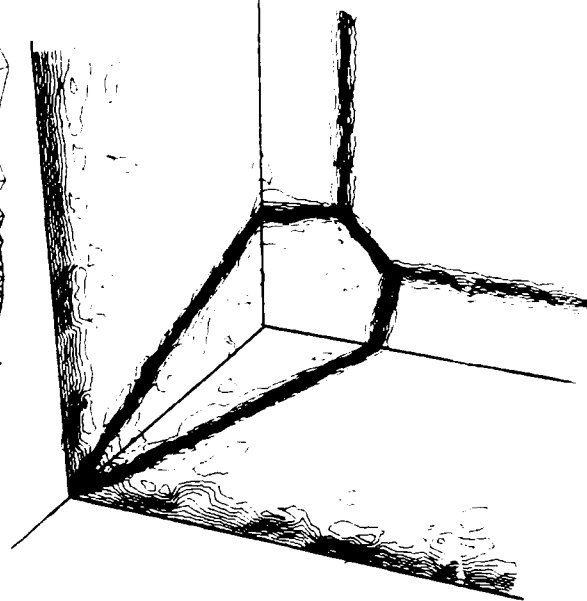Surface mesh for top plane ($y=2.5$).
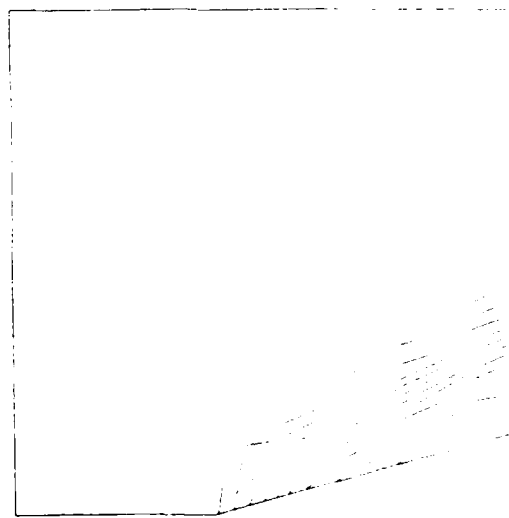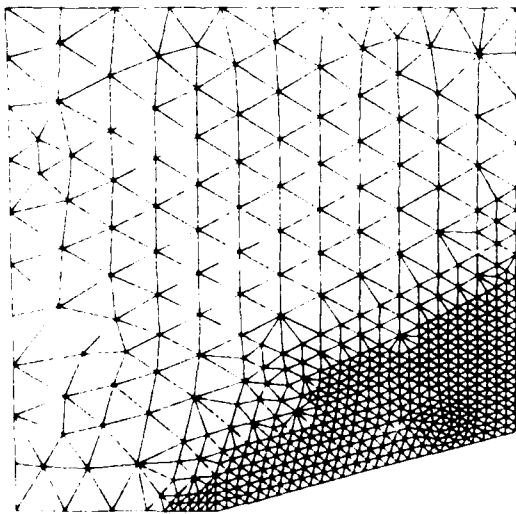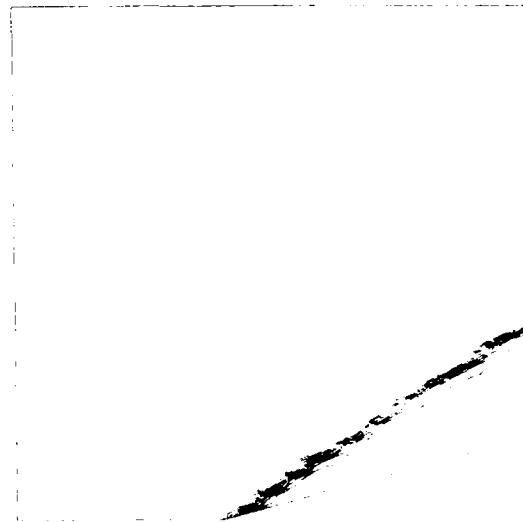

Pressure for top plane ($y=2.5$).

Figure 6a: Compression Corner. Original mesh and solution.
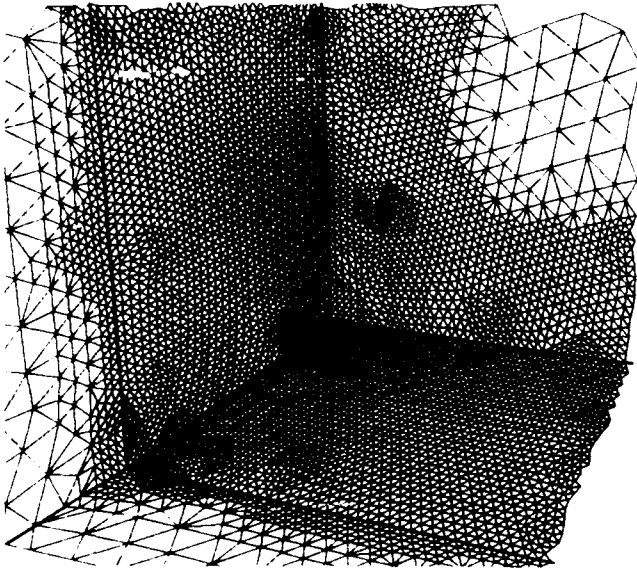
Surface mesh at the corner.

Pressure at the corner.

Surface mesh for top plane (y=2.5).

Pressure for top plane (y=2.5).

Figure 6b: Compression Corner. Mesh and solution after 2 refinements.

UNSTRUCTURED MESH GENERATION BY A
GENERALIZED DELAUNAY ALGORITHM

by
Timothy J. Baker
Department of Mechanical and Aerospace Engineering
Princeton University
Princeton, NJ 08544
USA

SUMMARY

A method for generating tetrahedral meshes will be described. The algorithm is based on the Delaunay triangulation and can treat objects of essentially arbitrary complexity. In order to preserve the surface triangulation of solid objects, it is necessary to override the Delaunay property and redefine the triangulation when points are introduced close to solid boundaries. Details of the generalized algorithm are presented and an efficient implementation of the triangulation method is described.

1. INTRODUCTION

Tetrahedral meshes offer an attractive approach to the problem of discretizing the space around complex three dimensional shapes. In principle, it is always possible to connect a collection of mesh points to provide a surface conforming mesh of tetrahedra around an object of essentially arbitrary complexity. However, the problem of finding an efficient and reliable algorithm to achieve this task is far from trivial.

There are at least three methods that have been used to generate tetrahedral meshes. Shephard et al. [1] have developed a mesh generator which first derives an octree representation of the object and surrounding space and then cuts each of the octree cubes into tetrahedra. The main difficulty in this approach is the problem of finding a consistent and unambiguous way of treating the surface of the solid object. The moving front technique has been successfully developed three dimensions by Peraire et al. [2] and Löhner [3]. The main difficulty here appears to be the need to track the front as it develops and folds over on itself. The third method that has been exploited by Cavendish et al. [4] and Perronnet [5] in structural problems, Cendes et al. [6] in electromagnetic problems, Weatherill [7], Baker [8,9], Holmes et al. [10] and George et al. [11] for aerodynamic problems, is the use of the Delaunay triangulation. The difficulty with this method lies in the requirement to establish the correct triangulation on any boundary surface and then preserve the surface triangulation while introducing the remaining flow field points. This can only be achieved by overriding the Delaunay algorithm to prevent connections which would break through the surface triangulation. George et al. [11] allow the Delaunay triangulation of the flow field points to proceed unchecked and then reconfigure the tetrahedra near the surface in order to reconstitute the surface triangulation. The approach described here uses a generalized version of the Delaunay algorithm which restricts the connections made near the body surface to ensure that the surface triangulation remains intact at all times.

The next section defines the Delaunay triangulation and outlines Bowyer's algorithm [12]. This is followed by a discussion of the generalized algorithm for overriding the Delaunay triangulation near fixed boundaries. Section 4 presents an efficient implementation of the algorithm based on the use of an octree data structure for storing the points that are to be triangulated. The next section discusses tetrahedron quality and the construction of a mesh containing mostly well shaped tetrahedra. The final section describes the application of this triangulation procedure to generate tetrahedral meshes around complete aircraft.

2. DELAUNAY TRIANGULATION AND BOWYER'S ALGORITHM

Definition 1

Given a set V_n of n points in k-space, the Delaunay triangulation is the unique triangulation of V_n such that no point $P_i \in V_n$ lies inside the circumsphere of any k-simplex.

A particularly straightforward method for generating the Delaunay triangulation is Bowyer's algorithm [12] which can readily be applied to any number of dimensions. It is an incremental algorithm which directly exploits the above characterization of the Delaunay triangulation as follows:

Let T_n be the Delaunay triangulation of the set of n points $V_n = \{P_i \mid i=1,\ldots,n\}$. For any simplex $S \in T_n$, let R_s be the circumradius and Q_s be the circumcenter. Now introduce a new point P_{n+1} inside the convex hull of V_n and define

$$B = \{S \mid S \in T_n, \ d(P_{n+1}, Q_s) < R_s\}$$

where $d(P,Q)$ is the Euclidean distance between points P and Q. Now B is non-empty since P_{n+1} is inside the convex hull of V_n and hence inside some simplex $S' \in T_n$, from which it follows that $S' \in B$. The region C formed when B is removed from T is simply connected, contains P_{n+1} (since P_{n+1} is inside $S' \in B$) and P_{n+1} is visible from all points on the boundary of C. It is therefore possible to generate a triangulation of the set of points $V_{n+1} = V_n \cup \{P_{n+1}\}$ by connecting P_{n+1} to all points on the boundary of C. Furthermore this triangulation is precisely the Delaunay triangulation T_{n+1}. A proof that Bowyer's algorithm is a valid procedure for generating the Delaunay triangulation is presented in references [9,17].

Definition 2

Let S be a simplex with circumcenter Q_S and circumradius R_S. A comparison of the distance $d(P,Q_S)$ between a given point P and Q_S will be referred to as the Delaunay test of P for simplex S. If $d(P,Q_S) < R_S$ then we say that P has failed the Delaunay test for simplex S and that simplex S has been broken by the point P.

Definition 3

Let T be the Delaunay triangulation of a point set V and consider a point $P \notin V$. Let $B=\{S|S \varepsilon T, \; d(P,Q_S) < R_S\}$. Thus B is the set of simplexes for which P fails the Delaunay test. Then the region C created by removing B from T is called a cavity.

Definition 4

Two points A and B of a graph are visible from one another if it is possible to join A to B without intersecting an edge.

To implement Bowyer's algorithm in three dimensions we start with a super tetrahedron, or super cube partitioned into five tetrahedra, which contains all the other points. The remaining points, which comprise the mesh to be triangulated, are now introduced one at a time and Bowyer's algorithm is applied to create the Delaunay triangulation after each point insertion.

It is necessary to maintain two lists, each of length four, for each tetrahedron in the existing structure. One list holds the forming points of the tetrahedron, the other holds the addresses of the four neighboring tetrahedra which have a common face. The second list, which provides information about the contiguities between the tetrahedra, is not strictly necessary for the implementation of the algorithm. However, it allows one to find all broken tetrahedra in a cavity by means of a tree search, once one broken tetrahedron has been found. Without this contiguity information, the algorithm would be hopelessly inefficient. It is also convenient to store the radius of the circumsphere and the coordinates of the circumcenter for each tetrahedron.

The remaining step in Bowyer's algorithm is the requirement to update the data structure. Tetrahedra belonging to the set B are deleted from the lists and new tetrahedra, obtained by connecting the new point to all triangular faces of the cavity boundary, are added. Finally it is necessary to determine the contiguities that exist among the new tetrahedra and also between the new tetrahedra and the old tetrahedra which have faces on the cavity boundary.

The only floating point operations required in this algorithm occur in the Delaunay test for each tetrahedron that is examined when searching for those tetrahedra which make up the cavity. Owing to the finite precision arithmetic that is used, the Delaunay test will make an ambiguous decision if the new point falls on the circumsphere of a tetrahedron. It is therefore necessary to use high precision arithmetic. Moreover it is particularly important, when forming the set B of broken tetrahedra, to exclude from B any tetrahedron whose circumsphere does not strictly contain the new point. We therefore introduce a tolerance $\varepsilon > 0$ and include in B only those tetrahedra S for which $d(P,Q_S) < R_S - \varepsilon$, where ε is chosen sufficiently large to ensure strict inclusion.

3. GENERALIZED ALGORITHM

In order to generate an unstructured mesh around a solid object, it proves convenient to allow the triangulation of the space inside as well as outside boundary surfaces. The triangulation therefore starts by introducing the farfield points and the object points. The tetrahedra which make up the interior triangulation of the object are identified and flagged. The remaining points, which correspond to the flowfield points, are then introduced.

When a flowfield point is introduced very close to the object surface it may fall inside the circumspheres of one or possibly several interior tetrahedra. The procedure that was adopted in reference [9] would reject any such points in order to prevent reconnections which would penetrate the object surface. This approach has now been replaced by a generalized Delaunay algorithm which allows a partial retriangulation of the cavity, leaving intact any tetrahedra which form the solid object and are therefore regarded as fixed tetrahedra. Thus, if the cavity created by the introduction of a new point contains one or more of the fixed tetrahedra, reconnections are restricted to the part of the cavity that does not contain any of the fixed interior tetrahedra.

For planar triangulations, this procedure always leads to a valid triangulation [17]. This follows from the observation that the removal of one or more triangles from a planar cavity partitions the cavity into disjoint regions. It can then be shown that P is visible from all vertices on the boundary of restricted cavity in which P lies. In three dimensions the removal of a tetrahedron, from a tetrahedral complex which forms the Delaunay cavity, does not necessarily divide the cavity into disjoint regions. There is therefore no guarantee that point P is visible from the boundary of the restricted cavity, and it follows that in three space the reconnection of P to the boundary of a restricted cavity will not necessarily produce a valid triangulation.

However, we may reasonably expect that in many cases all vertices of the restricted cavity in three space, will be visible from the new point P. If we can detect the cases when P is not visible from every vertex, we can reject those points and generate a triangulation of three space with the remaining points. Let $\{S_i\}_{i=1,n}$ be the set of tetrahedra which make up the restricted cavity. Now let $\{\hat{S}_i\}_{i=1,m}$ be the tetrahedra that are obtained in the retriangulation by joining the new point P to each of the triangular faces on the boundary of the cavity. The cavity volume is given by

$$\sum_{i=1}^{n} vol(S_i)$$

where $vol(S_i)$ is the volume of tetrahedron S_i. If P is visible from all vertices of the cavity so that a proper retriangulation is formed, then the volume is also given by

$$\sum_{j=1}^{m} vol(\hat{S}_j)$$

If P is not visible from all vertices of the cavity then one or more of the $\hat{S}_j$ will intersect non-cavity tetrahedra. In this case

$$\sum_{i=1}^{n} vol(S_i) < \sum_{j=1}^{m} vol(\hat{S}_j)$$

It is important to note that the volume of each tetrahedron is found as a by-product in the computation of the circumcenter coordinates. The application of this particular test is therefore computationally inexpensive.

If point P is found to be not visible from all vertices of the cavity then the cavity size is further reduced by excluding more tetrahedra and re-testing to determine whether P is now visible from the vertices of the reduced cavity. This procedure is facilitated by the ordering of the cavity tetrahedra. We first find the tetrahedron that contains the point P. From this tetrahedron we search outward looking at its neighbors and retaining those tetrahedra which belong to the cavity. This forms the second layer; the third layer consists of the neighbors of the second layer which belong to the cavity and have not already appeared in the list. This process continues until the tree search terminates and we have found all cavity tetrahedra.

If P is not visible from all cavity vertices we remove the outermost layer and re-test for visibility. If we continue removing layers we must eventually reach a stage where P is visible from all vertices of the reduced cavity, since the first layer consists of the tetrahedron containing P and the vertices of this tetrahedron are certainly visible from P.

4. ALGORITHM EFFICIENCY

When a new point is inserted, a search is made through the list of tetrahedra to find the first tetrahedron that fails the Delaunay test. The remaining tetrahedra which make up the cavity can be found by a tree search. After these tetrahedra have been removed, the points on the boundary of the cavity are connected to the new point P and the new tetrahedra thus formed are added to the data structure.

The time required to triangulate N points will be given by

$$T = \sum_{k}^{N} (T_k + T'_k)$$

Here, T_k is the time taken to search for the first tetrahedron broken by the introduction of the kth point into the triangulation of k-1 points. T'_k is the time taken to find all remaining tetrahedra in the cavity and construct the new triangulation. The time T'_k will be proportional to the number of tetrahedra in the cavity. If the points are inserted in a widely distributed manner corresponding to a coarse sprinkling followed by a finer distribution [9], the cavity size and hence time T'_k should be roughly independent of k. The majority of points are flowfield points which are introduced first as a coarse lattice for the farfield followed by a finer lattice of midfield points and so on. The time T'_k can therefore be regarded as O(1).

The time complexity of the algorithm is therefore dominated by the search time T_k. In general, the list of tetrahedra will be randomly ordered and, in the worst case, T_k will be O(k) leading to an overall time complexity for the triangulation that is $O(N^2)$. In reference [9] it is shown that this can be improved to $O(N^{4/3})$ by starting the search with the most recently created tetrahedra. However, this requires the points to be inserted in a series of forward and backward sweeps which presumes a well defined ordering.

It is therefore necessary to introduce a data structure which allows an efficient search for the first tetrahedron which fails the Delaunay test irrespective of the point ordering. To achieve this, an octree structure has been exploited to store the points that have previously been inserted. Octree (in two dimensions quadtree) data structures have been used in a variety of contexts [1,13,14] More recently, Löhner [3,15] has adopted this data structure to produce efficient search procedures for his unstructured mesh generator based on the moving front technique.

A quadtree structure for a planar point set is illustrated in Figure 1. The terminal quads are those which contain no more than four points. When a fifth point falls inside a quad, this quad is divided into four daughter quads and each of the five points is re-assigned to one of the daughters. Each quad is associated with a list whose first four entries hold the addresses of the points it contains in the case of a terminal quad, and the addresses of the daughter quads for a quad which is not terminal. In Figure 1, quads 1 and 5 are parent quads and therefore hold the addresses of their daughters. The remaining quads are terminal and therefore have the addresses of their points, indicated by the letters A through K. In this way, a tree structure is built up which makes it possible to determine quickly the location of any point in the data structure. The list associated with each quad has three further entries which hold such information as the address of the present quad's parent, its position in the parent quad and the number of points it contains. In three dimensions, the quad is replaced by a cube which
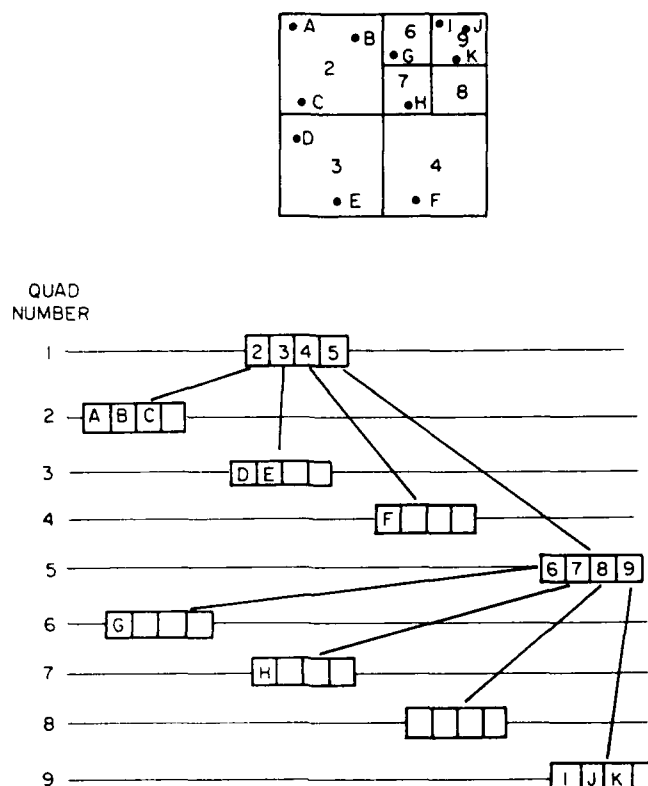
Fig. 1   Quadtree Data Structure

is split into eight octants.   In this case, the list associated with each octree has length eleven of which the first eight entries contain the addresses either of its daughters or, for a terminal octree, the addresses of the points it contains.

In the Delaunay algorithm, the octree data structure is exploited to find the point nearest to a newly introduced point. With each previously introduced point one associates a tetrahedron which has this point as a vertex. The search for the first broken tetrahedron thus starts with the tetrahedron associated with the point nearest to the new point and proceeds to examine all neighboring tetrahedra which have this nearest point as a vertex. The octree search to find the nearest point is accomplished in $O(\log k)$ time.   The remaining search to find a broken tetrahedron takes $O(1)$ time. In this way, it is possible to find the first broken tetrahedron in a time T_k which is $O(\log k)$. It follows that the overall time complexity of the algorithm $O(N \log N)$.

5.   QUALITY OF TETRAHEDRAL ELEMENTS

It is well known, from finite element theory, that badly shaped elements can lead to inaccurate and unstable approximations [18]. It is therefore important to observe geometric constraints on the shape of element that is allowed. For example, a planar triangulation is usually required to satisfy the minimum angle condition, although this has been shown to be too restrictive and can be replaced by a condition which limits the maximum allowable angle [19]. The minimum angle condition, however, can be reformulated as a requirement that the in-circle should not be too small, and this condition can be readily extended to higher dimensions.

In two dimensions we associate with each triangular element the linear shape functions $\phi_i(x,y)$, $i = 1,2,3$ such that ϕ_i takes the value one at the vertex P_i and zero at the other two vertices. A function $u(x,y)$ which takes the value $u_i = u(x_i,y_i)$ at the nodes $i = 1,2,3$ can be approximated by the linear interpolant

$$\hat{u}(x,y) = \sum_{i=1}^{3} u_i \phi_i(x,y)$$

If we define the interpolation error $\varepsilon = u - \hat{u}$, it can be shown [18] that for u sufficiently smooth the maximum error satisfies

$$|\varepsilon| \leq C_0 H^2 \max_{|m|=2} |D^m u|$$

where $D^m = \dfrac{\partial^{|m|}}{\partial x^\alpha \partial y^\beta}$, $|m| = \alpha + \beta$, and H is the maximum edge length. A corresponding estimate of

the form

$$\left| D\varepsilon \right| \leq C_1 H \max_{|m|=2} \left| D^m u \right|$$

is obtained for the derivative of the interpolant provided the derivatives of the shape functions satisfy

$$\max \left| D\phi_i \right| \leq \frac{C}{H}, \quad i = 1,2,3$$

This is known as the uniformity condition and for a planar triangulation is satisfied provided that all angles in the triangulation exceed some lower bound as $H \to 0$ [18].

Let h_i be the length of the edge opposite vertex P_i, let $H = \max \{h_1, h_2, h_3\}$ and r be the radius of the inscribed circle. It can be shown [20] that the uniformity condition implies that

$$\frac{H}{r} = 0(1)$$

Now let R be the circumradius of the triangular element and let $h = \min \{h_1, h_2, h_3\}$. We define three geometric parameters which allow us to characterize well-shaped and badly-shaped triangles:-

$$\sigma = \frac{H}{r}, \quad \omega = \frac{R}{H}, \quad \tau = \frac{H}{h}$$

The uniformity condition implies that a triangle is badly shaped if $\sigma \gg 1$. There are two distinct types of bad triangle for which $\sigma \gg 1$. Either $\omega = 0(1)$ and $\tau \gg 1$ which corresponds to an acute angled triangle (the angle opposite side h is very small); or otherwise $\omega \gg 1$ which occurs if the triangle is obtuse and the angle opposite edge H is close to 180 degrees.

We now postulate that the points are distributed so that the distance between neighboring points changes slowly, and that within any small region the point distribution is almost uniform. Since the circumcircles of the Delaunay triangles can contain no other points, it follows from the uniform point distribution hypothesis that $\omega = 0(1)$ thus precluding the formation of obtuse triangles. The same conditions also imply that all triangles will have $\tau = 0(1)$. Thus the combination of a smooth point distribution and the Delaunay triangulation ensures that only well-shaped triangles are formed.

In three dimensions, it can be shown [20] that the uniformity condition again requires $\sigma = 0(1)$. If we now consider the degenerate tetrahedra with $\sigma \gg 1$, we find three distinct types:-

<u>Type (i)</u>	$\tau \gg 1, \quad \omega = 0(1)$
<u>Type (ii)</u>	$\omega \gg 1$
<u>Type (iii)</u>	$\omega = 0(1)$

Type (i) corresponds to the highly acute triangle in the planar case; type (ii) occurs if one or more of the tetrahedral faces is highly obtuse. Type (iii) is known as a sliver [4] and is formed by four almost coplanar points.

The requirement that the points be smoothly distributed excludes the possibility of forming type (i) and type (ii) tetrahedra. However, type (iii) tetrahedra may arise and must be detected and eliminated. This can be achieved in a straightforward manner by inserting a new point inside the circumsphere of the sliver and allowing the Delaunay algorithm to create a triangulation containing the new point.

Since the sliver is formed from four nearly coplanar points, we can determine a direction that is approximately normal to the plane on which the sliver lies by taking the vector product of a pair of opposite edges (ie. edges which do not have a common vertex). There are three pairs of opposite edges and it is possible that one or two of the pairs are composed of two nearly parallel edges. We therefore calculate the vector product for the three pairs and take the product with the maximum magnitude. Suppose this

product is $\underline{a} \times \underline{b}$ where $\underline{a}$ and $\underline{b}$ are opposite edges. The vector $\underline{p} = \dfrac{a \times b}{|a \times b|}$ is a unit vector normal to $\underline{a}$ and $\underline{b}$ and hence approximately normal to the plane on which the sliver lies. Let $\underline{x}_c$ be the circumcenter for the sliver and define a new point by

$$\underline{x} = \underline{x}_c + \theta \, \underline{p}$$

where $|\theta| < 1$. This restriction on θ is necessary to ensure that the point lies inside the sliver's circumsphere thus producing a retriangulation in which the sliver is eliminated. In most cases we also find that the circumcenter $\underline{x}_c$ will lie near to the sliver and so θ should not be too close to zero.

In practice relatively few slivers occur and the procedure described here succeeds in removing these bad tetrahedra. The resulting mesh is therefore composed of good quality tetrahedra which all satisfy the uniformity condition.

The distribution of points should reflect the need to resolve rapid flow variations near the aircraft surface which requires a high density of points in that region. On the other hand, the requirement that the points be smoothly distributed prohibits any rapid change in point density.

The points outside the body are organized into a collection of several sets, arranged in order of increasing density or refinement. The first set is composed of a very coarse lattice extending about twenty body lengths away from the body in all directions. The next two sets are made up of points forming denser lattices throughout the same region. The fourth set extends
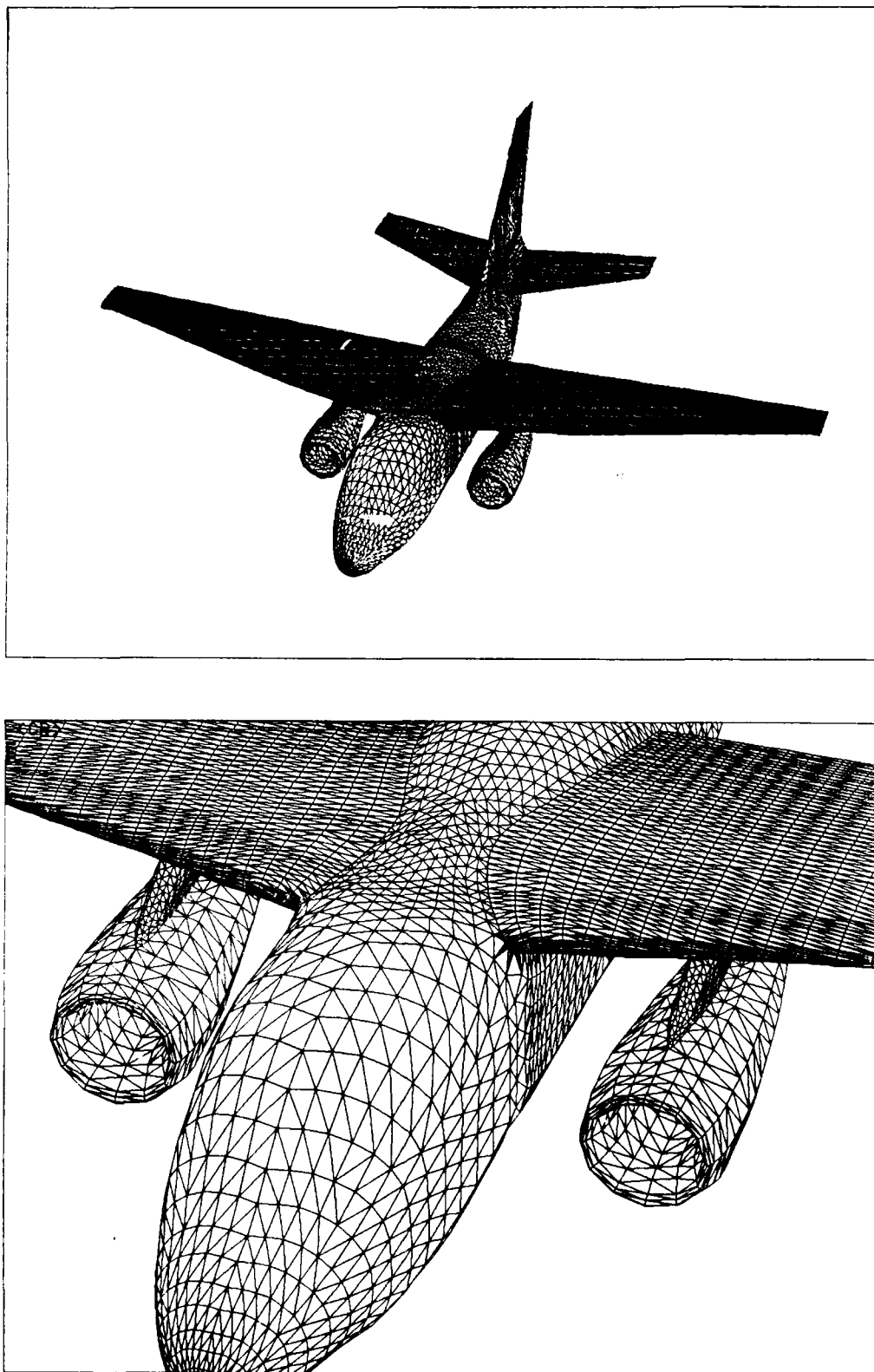
Fig. 2   Surface Triangulation on the Lockheed S3A

about ten body lengths away from the body in all directions with the spacing between adjacent points one half that of the underlying lattice from the third set. The remaining sets follow this trend of covering a region closer to the body and halving the spacing between points until the point spacing is about one percent of the body length. The final set comprises a shell determined by placing a few points along a normal to each surface point.

In this way, it is possible to generate well-shaped tetrahedra throughout most of the flowfield. Difficulties occur near the body surface, however, since the surface points are not smoothly distributed. This is particularly apparent in areas of high curvature such as wing leading edges. The points must be closely spaced to accommodate the high curvature at a wing leading edge but need not be closely spaced in the spanwise direction. In this case, needle-like type (i) tetrahedra for which $r \gg 1$ must inevitably be formed in the vicinity of the body surface.

The generalized Delaunay algorithm which permits points to be triangulated close to the body surface will also allow type (ii) tetrahedra to be formed. It is therefore necessary to monitor the tetrahedra which are generated and reject any point whose insertion would result in a tetrahedron with a large circumradius. In effect, this procedure disallows any tetrahedron whose minimum altitude is much less than its maximum edge length.

The resulting mesh is mainly composed of well-shaped, nearly regular tetrahedra. Near the body surface, however, the quality of the tetrahedra depends on the distribution of surface points. In particular, where regions of high curvature dictate a surface distribution of unevenly spaced points, we can expect needle-like tetrahedra to be formed. The extent to which the quality of the flow solution is adversely affected by the presence of thin tetrahedra requires further investigation.

6. MESHES FOR COMPLETE AIRCRAFT

The constrained Delaunay algorithm provides a robust method for generating tetrahedral meshes around and inside complex three dimensional objects. A mesh generator based on the principles described above has been linked to an Euler flow solver [8,16] in order to calculate inviscid transonic flow over complete aircraft.

The mesh points which define the aircraft surface, together with the set of automatically generated internal points, are triangulated first. At this stage, it is necessary to determine which tetrahedra comprise the aircraft structure. One now has the option of stopping the procedure and examining the surface triangulation, or allowing the triangulation of all remaining points to be carried out. The triangulation rate is about 5000 points per minute on a Cray 2. Thus a typical mesh containing about 200,000 points for the space around half the aircraft will be triangulated in approximately 40 minutes. The number of surface points typically ranges from about 5,000 to 10,000 depending on the complexity of the aircraft, with an equal number of internal points. It follows that the time required to triangulate all the aircraft points (surface points plus internal points) varies between two and four minutes on a Cray 2. The examples presented in Figures 2,3 and 4 show the surface triangulation over three different aircraft and thus demonstrate the scope and versatility of this method. Figure 2 show the Lockheed S3A with approximately 5000 points on half the aircraft surface. The flowfield region that is triangulated includes the duct inside the nacelles, thus allowing a calculation of the flow through the nacelles as well as the flow around the rest of the aircraft. A more complex shape is the McDonnell-Douglas F15 shown in Figure 3. This aircraft is defined by about 8000 surface points for half the aircraft. Again the complete mesh includes the region inside the duct running from the intake through the aircraft to the nozzle at the rear. Although difficult to discern from Figure 3, the engine intake is separated from the fuselage thus correctly modeling the intake bleed. Note also the presence of the snag on the horizontal tail where there is a deliberate discontinuity in the leading edge.

Figures 4 show the surface triangulation around the Boeing 747-200. This example has two engine nacelles on each half of the aircraft and each nacelle has both an outer cowl and an inner primary. The detail in this region is apparent from Figure 4, and the complete triangulation includes the space through and around the entire nacelle assembly. Flow calculations have been run assuming flow-through conditions. There would be no difficulty, however, in modifying the flow solver to accommodate a powered nacelle boundary condition.

In order to generate a mesh around a new aircraft geometry one needs a detailed surface definition. Typically this would be supplied in the form of a series of patches or sections for each component of the aircraft. It is important to display the surface triangulation on a graphics terminal in order to decide whether the distribution of surface points is adequate and whether the surface triangulation is satisfactory. It is, in fact, usually necessary to interpolate some extra sections of points in various critical regions in order to obtain sufficient resolution of every geometric detail. When one is satisfied with the surface definition, the triangulation of the remaining flowfield points proceeds completely automatically.

Mesh generation of complete aircraft flowfields is thus accomplished with minimal user intervention, and as the examples indicate, it is possible to treat aircraft shapes of essentially arbitrary complexity.
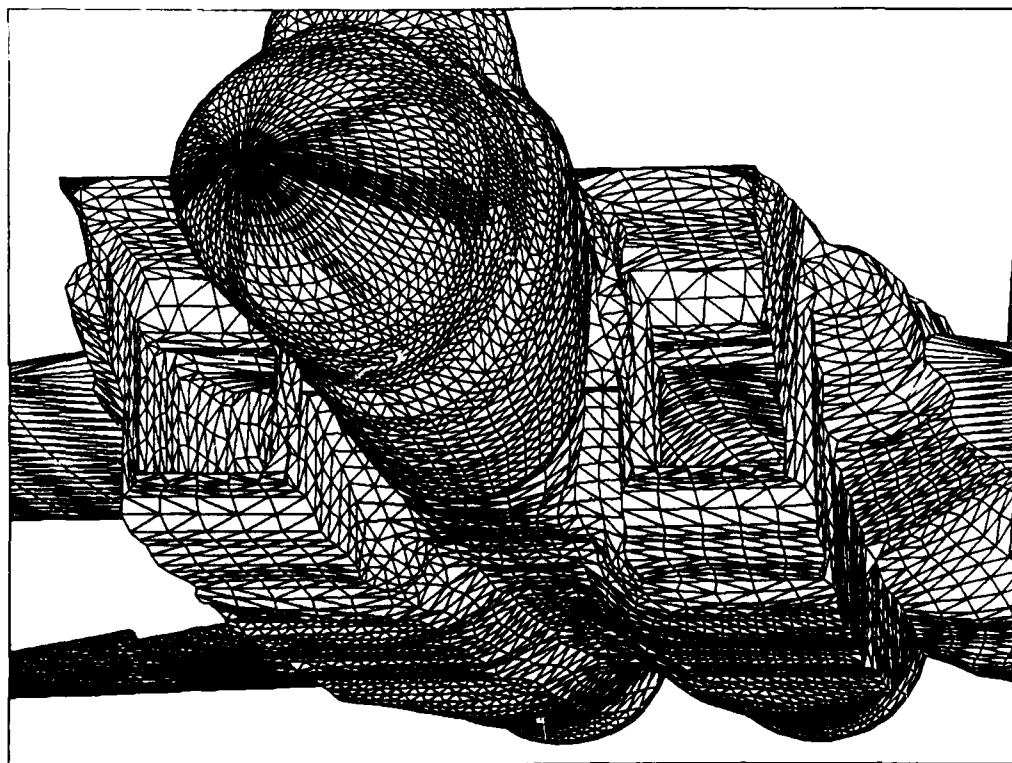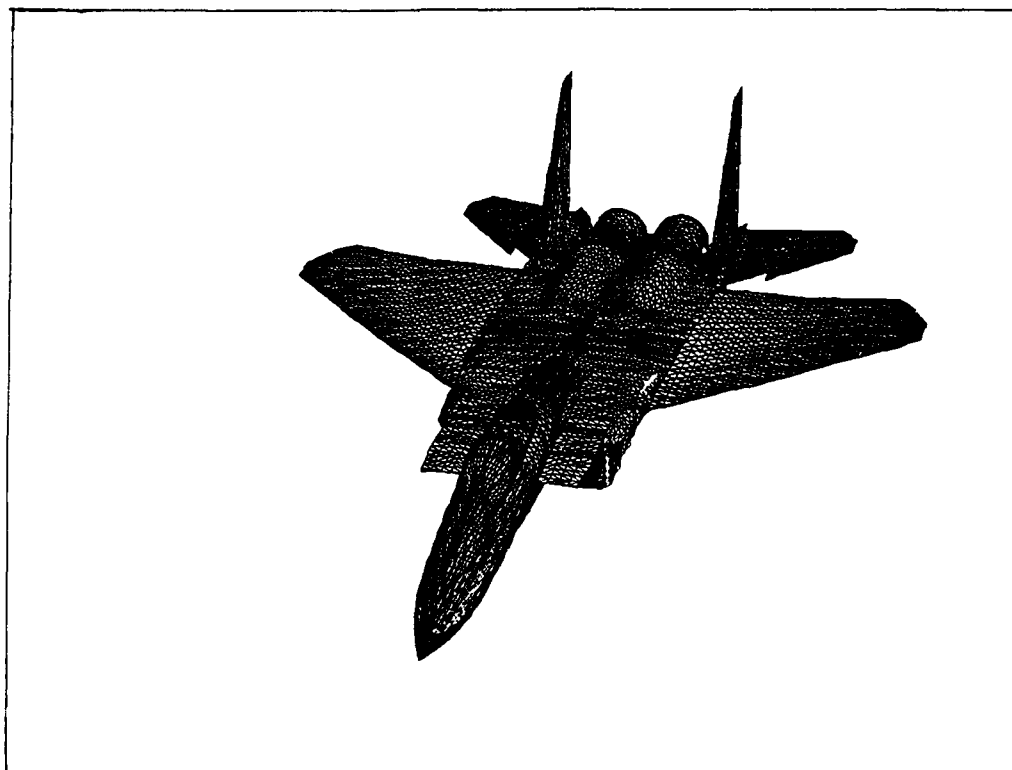
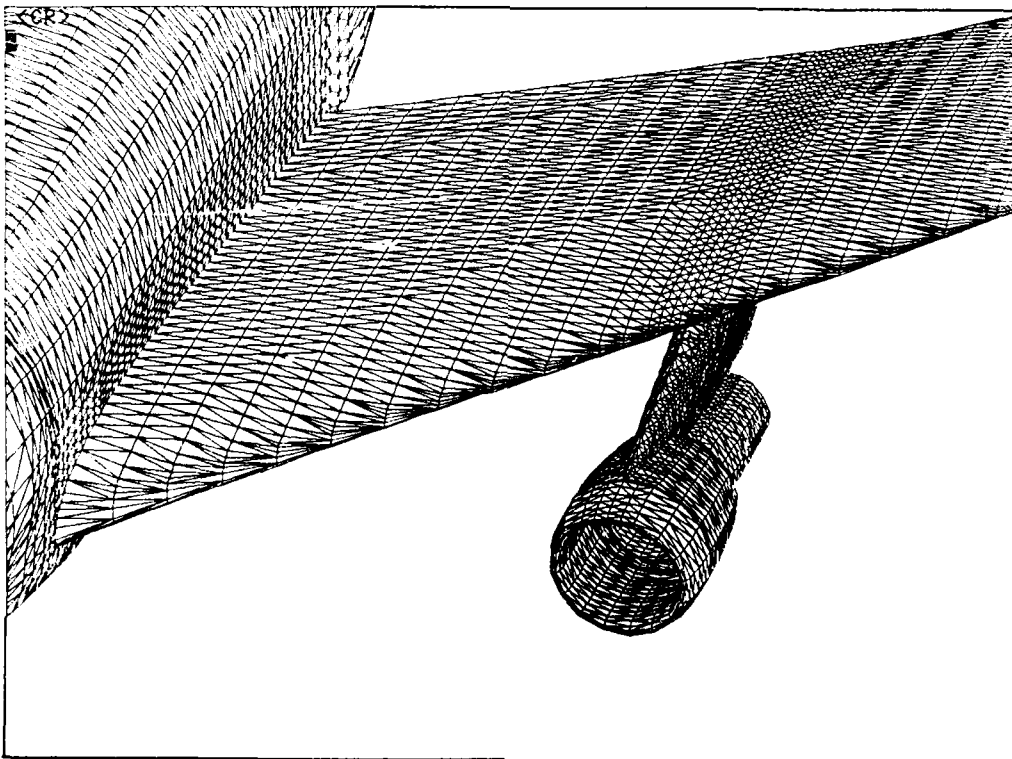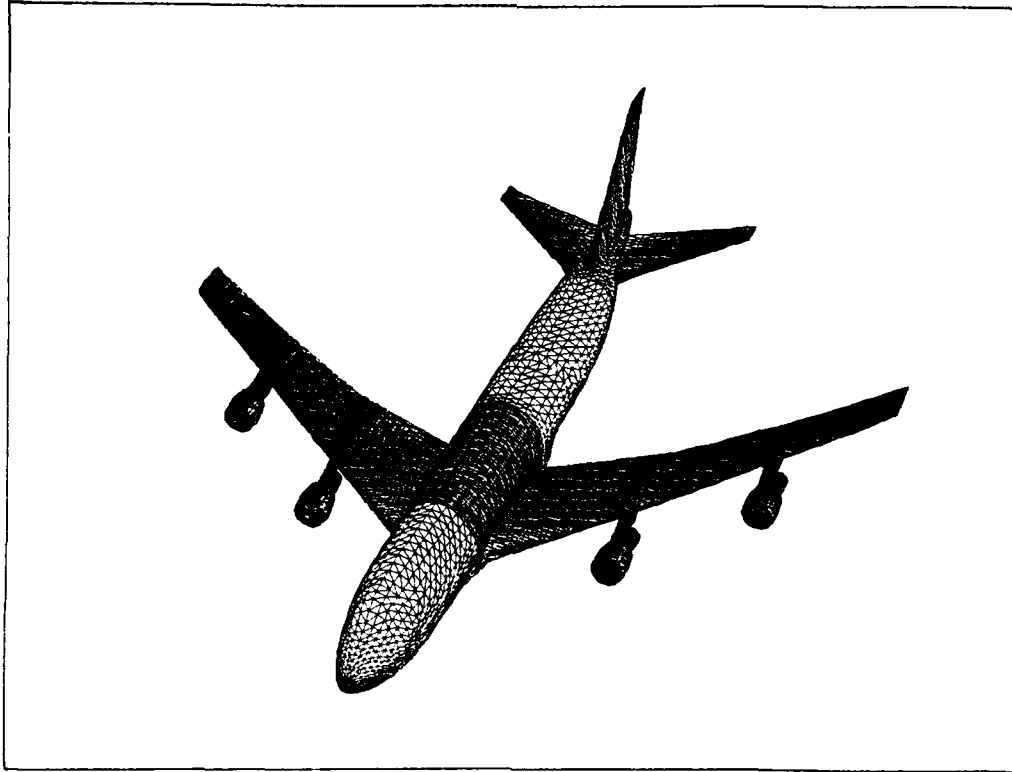Fig. 3   Surface Triangulation on the McDonnell Douglas F-15

Fig. 4   Surface Triangulation on the Boeing 747-200

REFERENCES

1. M.S. Shephard, F. Guerinoni, J.E. Flaherty, R.A. Ludwig and P.L. Baehmann, "Finite Octree Mesh Generation for Automated Adaptive Three-Dimensional Flow Analysis", Proceedings Second International Conference on Numerical Grid Generation in Computational Fluid Mechanics, Miami, pp 709-718, December 1988

2. J. Peraire, J. Peiro, L. Formaggia, K. Morgan and O.C. Zienkiewicz, "Finite Element Euler Computations in Three Dimensions, AIAA 26th Aerospace Sciences Meeting, Reno, AIAA Paper 88-0032, January 1988

3. R. Löhner, "Generation of Three-Dimensional Unstructured Grids by the Advancing - Front Method", AIAA 26th Aerospace Sciences Meeting, Reno, AIAA Paper 88-0515, January 1988

4. J.C. Cavendish, D.A. Field and W.H. Frey, "An Approach to Automatic Three-Dimensional Finite Element Mesh Generation", Int. J. Numer. Meth. Eng., Vol. 21, pp 329-347, 1985

5. A. Perronnet, "A Generator of Tetrahedral Finite Elements For Multi-Material Objects or Fluids", Proceedings Second International Conference on Numerical Grid Generation in Computational Fluid Mechanics, Miami, pp. 719-728, December 1988

6. Z.J. Cendes and S. H. Shahnasser, "Magnetic Field Computations using Delaunay Triangulation and Complementary Finite Element Methods", IEEE Trans. Magnetics, Vol. MAG-19, No. 6, 1983

7. N.P. Weatherill, "A method for Generating Irregular Computational Grids in Multiply Connected Planar Domains", Int. J. Num. Meth. in Fluids, Vol. 8, pp. 181-197, 1988

8. A. Jameson, T.J. Baker and N.P. Weatherill, "Calculation of Inviscid Transonic Flow over a complete Aircraft", AIAA 24th Aerospace Sciences Meeting, Reno, AIAA Paper 86-0103, January 1986

9. T.J. Baker, "Three Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets", AIAA 8th Computational Fluid Dynamics Conference, AIAA Paper 87-1124-CP, Hawaii, June 1987

10. D.G. Holmes and S.H. Lamson, "Adaptive Triangular Meshes for Compressible Flow Solutions", Proceedings First International Conference on Numerical Grid Generation in Computational Fluid Dynamics, Landshut FRG, pp 413-424, July 1986

11. P.L. George, F. Hecht and E. Saltel, "Constraint of the Boundary and Automatic Mesh Generation", Proceedings Second International Conference on Numerical Grid Generation in Computational Fluid Mechanics, Miami, pp 589-597, December 1988

12. A. Bowyer, "Computing Dirichlet Tessellations", The Computer Journal, Vol. 24, No. 2, pp. 162-166, 1981

13. C.L. Jackins and S.L. Tanimoto, "Octrees and their use in the Representation of Three-Dimensional Objects", Comp. Graphics Image Process., Vol. 14, pp. 249-270, 1980

14. D. Meagher, "Geometric Modeling using Octree Encoding", Comp. Graphics Image Process., Vol 19, pp. 129-147, 1982

15. R. Löhner, "Some Useful Data Structures for the Generation of Un-structured Grids, Comm. Appl. Num. Meth., Vol. 4, pp 123-135, 1988

16. A. Jameson and T.J. Baker, "Improvements to the Aircraft Euler Method", AIAA 25th Aerospace Sciences Meeting, Reno, AIAA Paper 87-0452, January 1987

17. T.J. Baker, "Automatic Mesh Generation for Complex Three Dimensional Regions using a Constrained Delaunay Triangulation", to appear in Engineering with Computers.

18. G. Strang and G. J. Fix, "An Analysis of the Finite Element Method", pub. Prentice Hall, 1973

19. I. Babuška and A.K. Aziz, "On the Angle Condition in the Finite Element Method", SIAM J. Numer. Anal., Vol. 13, No. 2, pp 214-226, 1976

20. T.J. Baker, "Element Quality in Tetrahedral Meshes", Proceedings Seventh International Conference on Finite Element Methods in Flow Problems, Huntsville, pp. 1018-1024, April 1989

APPLICATION OF THE TRANAIR RECTANGULAR GRID APPROACH TO THE AERODYNAMIC ANALYSIS OF COMPLEX CONFIGURATIONS*

by

Forrester T.Johnson and Satish S.Samant†
Michael B.Bieterman, Robin G.Melvin and David P.Young‡
John E.Bussoletti§ and Mike D.Madson¶
Boeing Commercial Airplanes
PO Box 3707
Seattle, WA 98124-2207
United States

ABSTRACT

In this paper we describe a numerical method which uses a rectangular grid to solve the nonlinear full potential equation about complex configurations. The grid is locally refined to resolve high velocity gradients arising from leading edge expansions or shock waves. The grid penetrates the boundary (described by networks of quadrilateral panels) and is generated automatically. Discrete operators are constructed using the finite element method. The system of nonlinear discrete equations is solved iteratively using a Krylov subspace method preconditioned by an exterior Poisson solver and a direct sparse solver. The primary emphasis of this work is to provide design engineers with an aerodynamic analysis tool (the TRANAIR code) which is accurate, reliable, economical, and flexible to use. Computational results for many interesting configurations are presented.

INTRODUCTION

Many aerodynamic designs are geometrically complex. Commercial aircraft, for example, have nacelles, stabilizers, slats, flaps, and ailerons, in addition to wings and fuselages. The geometry of typical military aircraft can be even more complex. The flow about these configurations may include shock waves, engine power effects, and very strong leading edge expansions. To aid in the design of such complex configurations the engineering community needs computational tools which can solve the associated boundary value problems on unbounded domains with many types of boundary conditions. If such tools are to be used effectively in a design project, the underlying numerical methods have to be accurate, reliable, efficient, and flexible

In three space dimensions, panel methods (boundary integral methods) have provided the desired degree of efficiency and geometry flexibility [1]. For the last twenty years full aircraft configurations have been routinely analyzed using panel methods. Panel method users have taken for granted the ability to add, move, or delete components at will, readily select and change boundary condition types, and obtain accurate solutions at reasonable cost in reasonable time. The primary drawback of panel methods is their limitation to linear subsonic or supersonic flows.

Unfortunately, many current aircraft fly at high speeds where a substantial portion of the flow is transonic. Transonic flow exhibits nonlinear behavior and requires computations in the volume exterior to the configuration surface. Many methods in aerodynamics have used body fitted structured or block structured grids that allow each grid cell to be treated in a similar fashion with minimal storage [2]. This, however, introduces the subsidiary problem of generating a body conforming grid. A second approach that allows arbitrary grid refinement is to use tetrahedral cells [3]. This approach also requires the generation of a body conforming mesh. A third approach uses rectangular grids with special operators near the boundary [4, 5]. We have chosen to use the rectangular grid approach for the reasons that are outlined below.

First, rectangular grids are easy to generate and use. One can start with a suitable uniform global grid and locally refine the rectangular cells in a hierarchical manner. Local refinement may be controlled externally by the user or adapted to the solution. Oct-tree data structures can be used to efficiently store and extract all the information related to the grid [6, 7]. Second, the rectangular grid approach lends itself to efficient use of the computing resources. Since the local refinement is hierarchi-

cal, all cells away from the boundary are geometrically similar, and the discrete operators for all these cells are identical up to a multiplication factor and require practically no storage. On the other hand, the operators for irregularly shaped cells near the boundary, which can be constructed using the finite element method, must be stored. However, this is not a major problem since the number of such cells is usually small compared to the total number of cells in the grid. It is also possible to take advantage of the rectangular grid in designing efficient algorithms that calculate residuals, the finite element stiffness matrix, etc.. Third, the rectangular cells are numerically far less ill-conditioned and allow the application of powerful preconditioners to ensure convergence. They allow an implicit extension of the finite computational region to infinity through the use of an exterior Green's function. This greatly reduces the number of cells needed to solve a given problem. And, finally, the requirement to generate a body fitted grid is eliminated since rectangular grids are independent of the boundary description. The engineering users of the computational tool have to provide only the boundary description to the code which significantly enhances the usability of the code.

We have developed and implemented a numerical method based on the rectangular grid approach in a computer code called TRANAIR. It solves the full potential equation with regions of differing total pressure, temperature, and swirl. TRANAIR has been developed to replace panel codes as a workhorse aerodynamics analysis tool for full configuration design. In order to accomplish this goal a wide variety of numerical algorithms have been combined.

In the following sections we will discuss these algorithms with particular emphasis on the grid. We will then discuss a number of applications of TRANAIR to aerodynamic analyses of complex configurations. This is followed by concluding remarks.

DESCRIPTION OF THE METHOD

In this section we describe the numerical method. First, we define the continuous problem to be solved. Then we describe the grid and boundary specification, and finally, the construction of the discrete operator system and its iterative solution. More details on discretization and the solution techniques may be found elsewhere [8].

Problem Definition

We solve the full potential equation given by

$$\mathcal{L}(\Phi) \equiv \vec{\nabla} \cdot \rho \vec{\nabla} \Phi = 0 \tag{1}$$

where the density and pressure are given by isentropic formulas

$$\rho = \rho_\infty \left[1 + \frac{\gamma - 1}{2} M_\infty^2 (1 - \frac{q^2}{q_\infty^2}) \right]^{\frac{1}{\gamma - 1}} \qquad p = p_\infty \left[1 + \frac{\gamma - 1}{2} M_\infty^2 (1 - \frac{q^2}{q_\infty^2}) \right]^{\frac{2}{\gamma - 1}} \tag{2}$$

Here, Φ is the total velocity potential to be determined, $q = \| \vec{\nabla} \Phi \|_2$ is the speed, M is the Mach number, γ is the ratio of specific heats, and subscript ∞ denotes a value at a far upstream location.

The far field condition is that the perturbation potential $\phi = \Phi - \Phi_\infty = \mathcal{O}(\frac{1}{R})$. The normal flow boundary condition is $\rho \frac{\partial \Phi}{\partial n} = h$ where h is zero for impermeability or may be specified on other surfaces, such as engine inlets. We may impose the Dirichlet condition, $\Phi = g$, on an engine exhaust surface, where tangential flow can be prohibited by specifying g to be a constant. The boundary conditions on wakes represent conservation of mass and normal momentum across the wake, .i.e., $\hat{n} \cdot \Delta(\rho \vec{\nabla} \Phi) = 0$, and $\Delta p = 0$ where Δ represents the jump across the wake surface.

The boundary value problem described above can be cast in a variational form for the purpose of applying the finite element method of solution. The principle we use is a generalization of the Bateman principle [9]. Specifically, the functional required to be stationary is given by

$$J = \int_{flow} p \, dV + \int_{inlets} h\Phi dS - \int_{wakes} \alpha(\rho \frac{\partial \Phi}{\partial n})(\Delta \Phi - \mu) + \frac{\alpha \rho}{2\Delta l}(\Delta \Phi - \mu)^2 dS + \int_{exits} \rho \frac{\partial \Phi}{\partial n}(\Phi - g) - \frac{\rho}{2\Delta l}(\Phi - g)^2 dS \tag{3}$$

where α denotes the average across the surface, and the wake strength μ is determined from $\Delta p = 0$.

A modification of the above formulation allows the simulation of flows involving regions of differing total temperature and pressure. The potential flow exists in each separate region as long as total

temperature and pressure are constant in each region. Hence to model such regions only pressure and density in those regions have to be redefined in the following way:

$$\rho = \rho_\infty \frac{r_p}{r_T}\left[1 + \frac{\gamma-1}{2}M_\infty^2(1 - \frac{q^2}{q_\infty^2 r_T})\right]^{\frac{1}{\gamma-1}} \qquad p = p_\infty r_p\left[1 + \frac{\gamma-1}{2}M_\infty^2(1 - \frac{q^2}{q_\infty^2 r_T})\right]^{\frac{\gamma}{\gamma-1}} \qquad (4)$$

Here, r_p and r_T are the ratios of the total pressure and total temperature in the region to their corresponding freestream values.

Surface Geometry and Grid Specification

The boundary surfaces are prescribed via networks of panels. This input format allows relatively easy specification of complicated boundaries. In TRANAIR, it is identical to the input format of a panel method code that is extensively used for linear flow analysis [1].

The surface networks are embedded in a rectangular grid that is confined to a finite rectangular computational region. The restriction of the grid to a compact region is justified through a source transformation. The argument goes as follows. Suppose that the partial differential operator $\mathcal{L}$ is well approximated by a constant coefficient differential operator $\mathcal{T}$ in the far field. Then the original differential equation $\mathcal{L}\Phi = 0$ is equivalent to

$$Q + (\mathcal{L} - \mathcal{T})\mathcal{G} * Q = 0. \qquad (5)$$

where $\mathcal{G}$ is the Green's function for $\mathcal{T}$ such that $\mathcal{T}(\mathcal{G}*Q) = Q$ for all Q and $\Phi = \mathcal{G}*Q$ satisfies the proper far field condition. If we use the sources Q as unknowns then we can restrict our computation to that region where Q is nonzero. We note that for most problems of interest Q approaches zero at ∞ much faster than Φ approaches Φ_∞. Hence it is possible to find a finite computational region outside which the approximation $Q \cong 0$ holds very well. For a wing in transonic flow the computational region needs to extend only one or two chord lengths away. The same argument holds for the discrete problem. For the full potential equation the Prandtl-Glauert operator may be taken as $\mathcal{T}$. Construction of a discrete Green's function for the Prandtl Glauert operator that satisfies the proper far field boundary condition is facilitated by the use of rectangular grids [5, 10, 11].

For a given rectangular computational region the process of constructing the volume grid is automatic and is described below. First, the rectangular finite computational region (which extends far enough to include all boundaries and regions where the flow is expected to be nonlinear) is refined into a uniform grid called the global grid. The global grid is independent of the boundary surfaces and its density has to be sufficient so that the discrete Green's function is able to resolve the far field adequately.

The global grid is refined, where necessary, in a hierarchical manner, i.e., selected grid boxes are cut into eight equal geometrically similar boxes. This process is repeated to give a grid with any desired local resolution. The final refinement pattern which is optimally suited to obtain a solution is rarely known a priori. Hence certain guidance is required to perform refinement. A good working strategy is one in which such guidance is provided initially by the user based on his knowledge of the problem and later derived from the solution itself as it evolves iteratively. In other words, a good guess to the initial grid is obtained by using criteria provided by the user (see below) and that grid is subsequently refined (or derefined) depending on the magnitude of the solution errors. (Currently the grids are obtained based on user provided inputs while we are implementing solution adaptivity into the code.)

The initial refinement can be controlled by the user by specifying two criteria. The first criterion is based on the length scale of the surface panels used to describe the boundary. Every box that is sufficiently close to a panel is refined if some weighted length scale associated with the panel is smaller than the length scale associated with that box. This criterion is effective in providing denser local grid near certain parts of the boundary if it is known that the solution has strong gradients there. The second criterion allows refinement away from boundary surfaces. Special "regions of interest or disinterest" can be prescribed each with desired minimum and maximum refinement levels. All the boxes in a special region are refined recursively until the minimum level is reached. Further refinement depends on the first criterion. This criterion is useful, for example, for problems in which shock waves exist. The special regions are hexahedral (very easy to input) and provide a fair amount of flexibility in generating off-surface refinement.

To adapt the refinement to the solution, an error indicator can be calculated for every box in the grid. This indicator can be adjusted based on the user provided information (regions of interest or disinterest) and used in either refining or derefining the grid. We note that we are currently implementing adaptive refinement in the code. We expect that this will significantly increase the reliability and efficiency of the code.

To take advantage of the hierarchical nature of the refinement and geometric similarity of the boxes we have developed a compact data structure based on an oct-tree [6, 7]. The oct-tree data structure we use is based on boxes. At the beginning of the oct-tree array certain overhead information that describes the size of the grid is stored. This is followed by two blocks of data, each as long as the size of the global grid, describing its refinement. Then follow the branch data elements of the tree which describe the hierarchical refinement and constitute the bulk of the the oct-tree. A typical branch data element in an oct-tree is illustrated in Figure 1. The first word in a branch data element points to the father box, the next eight words point to the refinement branches of the sons, if any, and the last word contains an accumulation index specifying the number of nodes encountered up to that point in the oct-tree. A null pointer (value zero) in a son entry of any branch data element represents an unrefined box.

Even though the oct-tree data structure described above is able to describe any collection of hier-
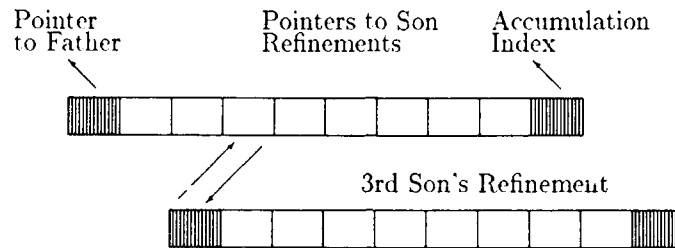
Figure 1: Oct-tree Data Structure Element.

archically refined grid boxes, it is convenient to restrict the refinement pattern. We require that no two face or edge neighbors in a "legal" refined grid differ by more than one level (see Figure 2). This rule prevents pathologically large stencils (see Figure 2), that otherwise might occur under certain circumstances, but allows refinement down to an arbitrary level within one adjacent coarse grid box.

We have extended the oct-tree data structure to accommodate nodal and boundary information. The index of a box is assigned to the node at its lower-left-near corner. In order to ensure that all nodes at refinement interfaces are accounted for, we perform pseudo-refinement (see Figure 2). Boxes added by the pseudo-refinement are used only to identify nodes and do not introduce extra degrees of freedom. This allows us to keep track of nodes as well as boxes using the same oct-tree with only a modest increase in storage. Any unrefined box cut by a boundary is identified with a negative number (equal to its index in a list of such boxes) placed in the son entry of a branch data element. This is a convenient and compact way of accounting for the presence of the boundary.

The data structure allows efficient extraction of a variety of information, such as the location of nodes and box centroids, box size, box refinement level, node indices, box adjacency and identity of boxes intersecting the boundary. The location of a node or a centroid of an unrefined box can be calculated by climbing up to the ancestor in the global grid and then using the global box information.
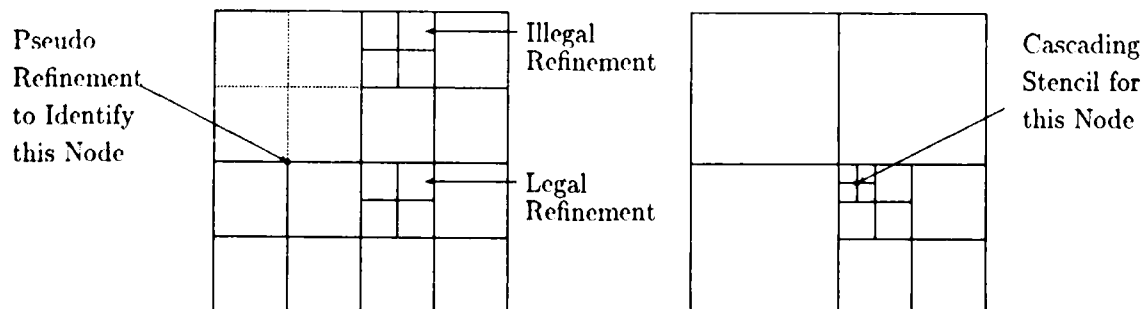

Figure 2: Some Issues in Hierarchical Refinement.

Adjacency (box-to-box connectivity) information can be obtained by starting at a box and climbing up the oct-tree to the root of the branch that includes that box and climbing down a complementary path to the neighboring box. Asymptotically this data structure requires storage equal to 10/7 the number of unrefined boxes for the box information. However, when the nodal information is added, the factor is closer to two for those cases where a substantial number of boxes have neighbor of different refinement level.

Discretization

The finite element method is used to discretize the partial differential operator. On each box a standard piecewise trilinear solution, parameterized by eight unknowns located at the corners, is defined. see Figure 3a. Element stiffness matrices are generated by taking variations of the functional J in Equation (3) with respect to each degree of freedom. For a standard box not cut by a boundary we may write the variation as

$$\delta J = -\int_{flow} \rho \vec{\nabla} \Phi \cdot \vec{\nabla} \delta \Phi dV \tag{6}$$

In evaluating the integral the density is taken as a constant over the box with the value of density evaluated at the centroid. Since the boxes are obtained by hierarchical refinement, every standard (off-boundary) box has the same element stiffness matrix up to a constant factor that depends only on the refinement level of the box and the density at the centroid of that box. Only one stiffness matrix must be stored for all the boxes not cut by the boundary. This can be exploited to reduce the storage requirements drastically.

a) Standard Box   b) Boundary Box
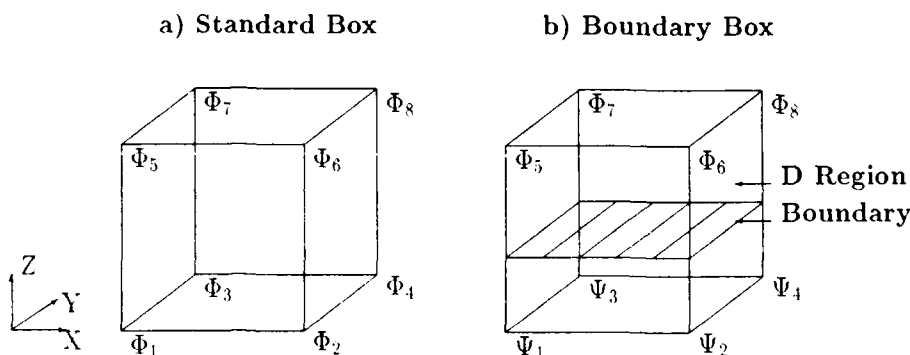
Figure 3: Box Finite Element with Eight Corner Unknowns.

For the boxes cut by the boundary the element stiffness matrices are derived from Equation (3) which includes appropriate surface integral terms. The domain of volume integration, referred to as a D region. is any connected flow region in the box, see Figure 3b. The domain for the surface integrals is the part of the boundary cut by the box. The trilinear approximation function for boxes cut by the boundary is defined in the same manner as described above except that the corner unknowns on the other side of a boundary surface can be viewed as extrapolated values, denoted in Figure 3b by Ψ. For complex geometries, the D regions can take any shape, and the integration involved in computing the stiffness matrix appears to be a formidable task at first sight. But the integrands are polynomials so that Gauss' and Stokes' theorems and simple one dimensional integration formulas can be used to systematically reduce these integrals to point evaluations at vertices of the bounding surfaces. It is possible to have more than one D region and consequently more than one approximation function in a given rectangular box and more than one unknown at some of its nodes. Each D region has a distinct element stiffness matrix which must be stored. However, these regions represent typically only 10 to 20% of the total regions needed to give an accurate solution of the boundary value problem. Hence, the storage required is acceptable.

In order to maintain conservation of mass across the grid interfaces one must maintain the continuity of basis functions used in the finite element method. This is achieved through introduction of pseudo-unknowns which lie on the edges and faces of boxes with finer neighbors. The value of a given pseudo-unknown is forced to be the average of its parents located at the endpoints of the edge or the corners of the face on the coarser neighbor.

Standard first order upwinding of the density is used to produce the artificial dissipation required when supersonic flow is present [12]:

$$\tilde{\rho} = \rho - \mu \ \Delta l \ \hat{v} \cdot \vec{\nabla}_{-}\rho \quad \text{where} \quad \mu = 1 - \mathrm{M}_c^2/\mathrm{M}^2. \tag{7}$$

Here μ is the switching function, M_c is the cutoff Mach number, Δl is an elemental length associated with the cell, $\hat{v}$ is the unit velocity vector, and the density gradient is upwinded.

Solution Techniques

Since the discrete system thus generated is generally large, nonsymmetric, nonlinear and often poorly conditioned, we have chosen the Krylov subspace method GMRES [13] as the basic iterative solver. GMRES is only effective, however, when it is preconditioned appropriately to produce a favorable distribution of eigenvalues. Denoting the solution unknown symbolically by χ let the discrete system to be solved be

$$L(\chi) = f \tag{8}$$

Standard left preconditioning is the application of GMRES to $N^{-1}L(\chi) = N^{-1}f$ where N^{-1} is the preconditioner. Standard right preconditioning is the application of GMRES to $LT^{-1}(Q) = f$ where $Q = T\chi$.

We have found that a combination of right and left preconditioning works best for the types of problems being considered here. We use a combination

$$\nu T N^{-1} L T^{-1} Q = \nu T N^{-1} f \tag{9}$$

where ν is a cutoff imposed at the outer boundary of the computational region. Here T^{-1} is the convolution with the Green's function for the discrete Prandtl-Glauert operator defined on the global grid. For other unknowns, this operator is simply the identity. This preconditioner is extremely effective for subsonic regions and ensures that the far field boundary condition is satisfied. The left preconditioner N^{-1} is chosen to be an approximate inverse of the global stiffness matrix restricted to a reduced set containing the unknowns near the boundary and in the supersonic regions. We obtain N^{-1} by a direct sparse incomplete factorization of N. To make direct decomposition feasible for large problems we introduce a dynamic drop tolerance and nested dissection ordering for the unknowns. The small elements (below a specified drop tolerance) in the decomposition are dropped as they are generated. This has a cascading effect and dramatically reduces fill [14]. A grid based nested dissection ordering reduces the fill during elimination.

To handle nonlinearities we embed GMRES in an approximate Newton method with GMRES driving a linearized operator obtained by finite differencing the nonlinear operator. A Newton method is rarely globally convergent if the initial solution is taken to be $\varphi = 0$, which is usually not a good approximation to the solution. Therefore, a Newton method generally must be damped, especially for large transonic problems.

We damp the Newton method by controlling the amount and the extent of the region of artificial dissipation. Higher dissipation (higher values of μ in Equation (7)) is applied over a larger region (lower values of M_c in Equation (7)) during the initial Newton steps so that supersonic zones and shock positions are located fairly early in the process, even though the shocks are quite smeared. After a few Newton steps, both the amount of dissipation and the extent over which it is applied are reduced to appropriate levels.

An alternate method of aiding the Newton iteration is to use a sequence of coarse to fine grids. The solution on a coarse grid is interpolated to provide the initial guess for the next finer grid. This procedure has the same effect as using large artificial dissipation (higher value of Δl in Equation (7)) initially and decreasing it in the subsequent grids. The shocks are smeared in the coarser grids but are closer to their actual location.

RESULTS

In this section we present some results obtained by applying the TRANAIR code to many practical problems of interest. The code runs on the Cray X-MP or the Cray Y-MP machines with an SSD but all the results shown here are obtained on the Cray X-MP. We present results for the ONERA M6

wing, the F16 fighter configuration with and without tanks and missiles, and several transport aircraft configurations with nacelles, struts, tip feathers, and installed powered nacelles. These cases have been analyzed with up to 320,000 grid boxes and involve sparse matrix decompositions of matrices with up to 200,000 unknowns. The solutions, converged to 6-8 digits accuracy, are typically obtained in run times of about 1-2 hours.

A standard aerodynamic test case is the ONERA M6 wing at $M_\infty = 0.84$ and $\alpha = 3.06°$. The TRANAIR results are compared to those obtained using FLO28 [2]. Dense grids (TRANAIR grid with 311,000 boxes and FLO28 grid with 364,000 cells) were used in both codes to capture the oblique (supersonic to supersonic) shock accurately. In Figure 4, we show two cuts through the TRANAIR grid. In Figure 5 we compare surface pressures at four span stations. The TRANAIR solution agrees quite well with the FLO28 solution using first order dissipation. In this problem, both codes used grid sequencing and obtained comparable accuracy (seven digits) at comparable cost (approximately an hour and a half).

Cut at 90 % Span   Waterline Cut
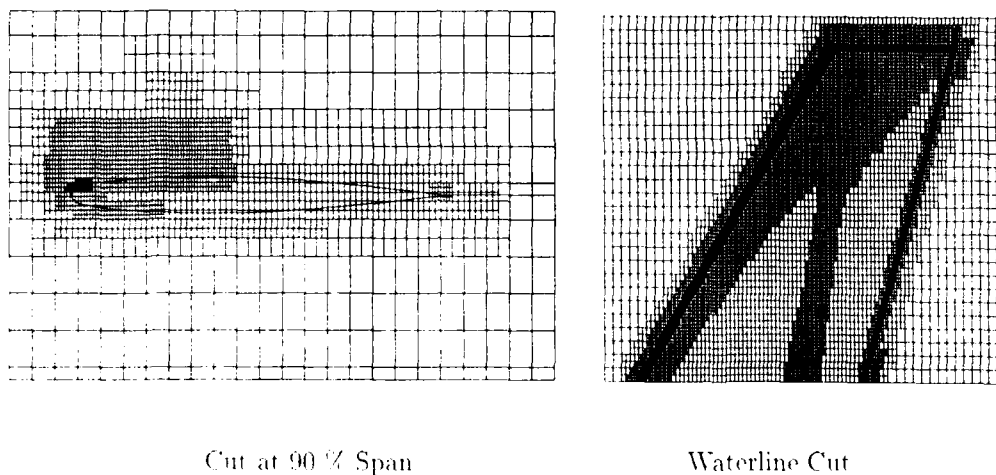
Figure 4: Two Cuts Through Grid for ONERA M6 Wing. $M_\infty = 0.84, \alpha = 3.06°$.

Next we present the results for the F16 fighter configuration, shown in Figure 6 with and without the tank and missile. The F16 without the tank and missile was analyzed at $M_\infty = 0.9$ and $\alpha = 4.0°$. The computational grid had 189,000 boxes. A comparison of computed surface pressure with wind tunnel data at two wing stations is shown in Figure 7. The agreement between computed results and test data is good. For the F16 with tank and missile, Figure 8 shows two plane cuts through the computational grid, which has about 216,900 boxes. Figure 9 compares computed surface pressure (at $M_\infty = 0.9$ and $\alpha = 4.0°$ just inboard and just outboard of the tank strut with those for the F16 without tanks and missiles. The qualitative effect of the tank is to speed up the flow underwing and is predicted well.

The next case is a 747-200 transport configuration with wing, body, struts and nacelles. The geometry description includes about 23,000 panels (see Figure 10) and is much bigger than which can be handled by panel codes. The flow analysis was performed at $\alpha = 2.7°$ and $M_\infty = 0.8$. This is approximately the largest freestream Mach number at which an inviscid solver can obtain reasonable results without boundary layer coupling. The grid used for this problem consisted of approximately 219,000 boxes. Figure 11 shows a cut through the grid. Figure 12 compares computed pressure with wind tunnel pressure data at four span stations of the wing. Overall, one sees very good agreement with experiment. Most of the differences are seen in the upper surface pressures and are attributable to viscous effects not currently modeled in TRANAIR. On the lower surface one can clearly see (in Figure 12) the effect of the outboard nacelle at the 69% span station. Near the leading edge of the wing at the same span station the TRANAIR solution indicates a supersonic to supersonic shock, also evident in the experimental data.

The next case we present is a transport aircraft with tip feathers. The paneling for the configuration is shown in Figure 13a. This configuration is a good example of where the length scales associated with different components are vastly different. Typical grid sections are shown in Figure 13b. We note
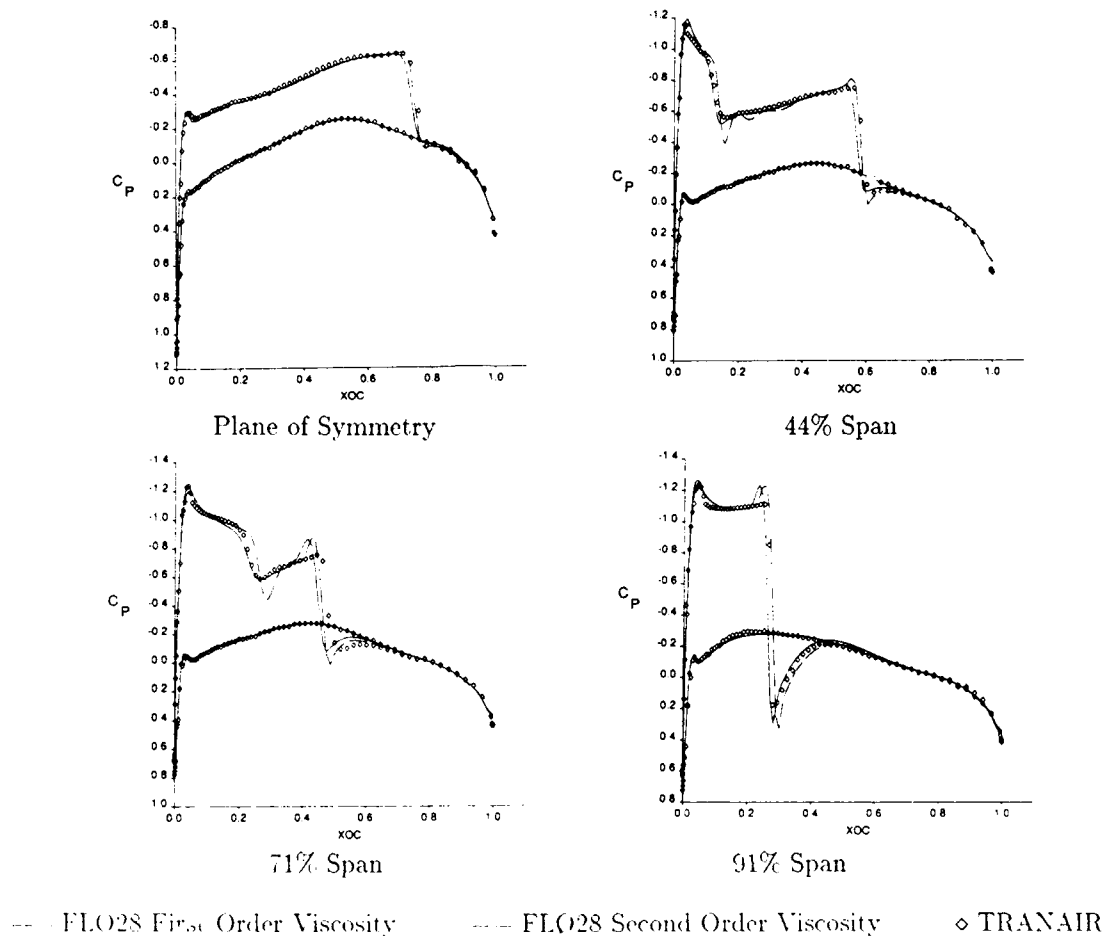
Plane of Symmetry   44% Span

71% Span   91% Span

—— FLO28 First Order Viscosity   — — FLO28 Second Order Viscosity   ◇ TRANAIR

Figure 5: Comparison of Surface Pressure at Four Span Stations on ONERA M6 Wing. $M = 0.84, \alpha = 3.06°$.
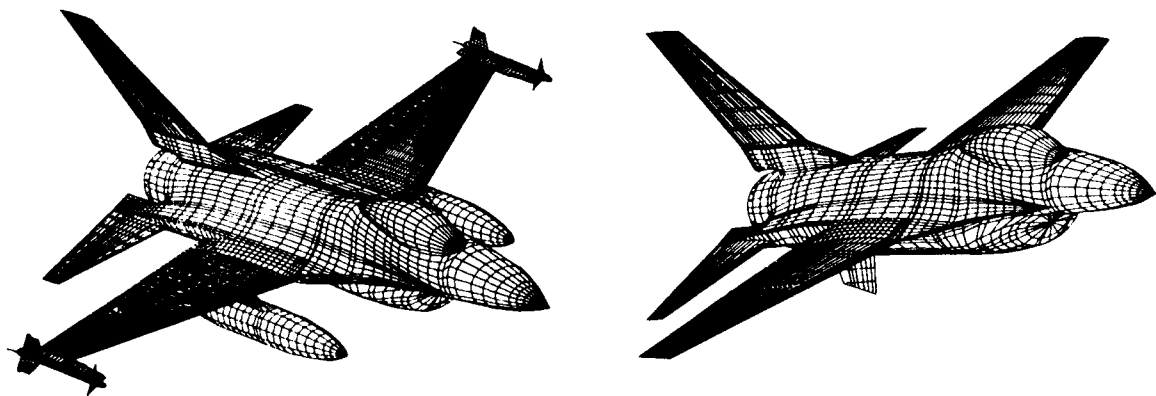
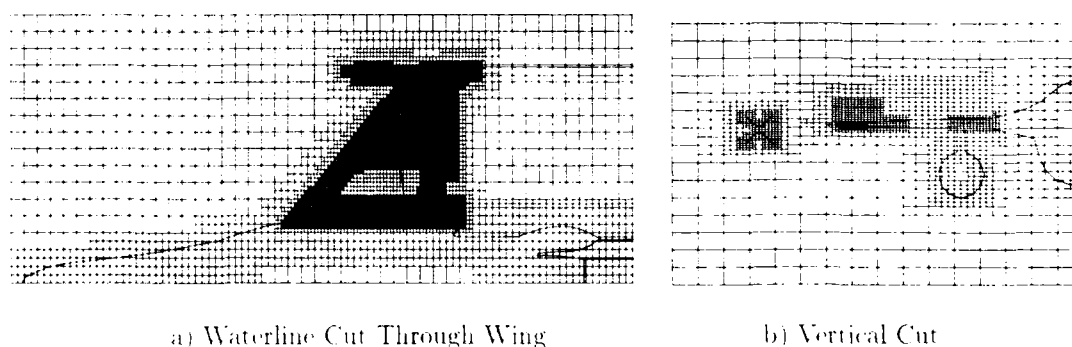Figure 6: F16 Aircraft Configuration with and without Tank/Missile.

that the finest grid box is seven levels below the biggest box (with volume ratio of approximately 2 million). The flow solution obtained at $M_\infty = 0.8$ and $\alpha = 1.6°$ is shown in Figure 13c. At this Mach number and angle of attack the shock on the forward feather is too far aft which can be attributed to lack of boundary layer modeling in TRANAIR.

Finally, we present analysis of a transport aircraft with installed powered nacelles. The plumes behind the nacelle are simulated as regions of different total pressure and temperature. In Figure 14a and 14b, we show the paneling for the configuration and a typical section of the grid with about 230,000 boxes. In Figure 14c and 14d we compare the pressure computed at an underwing station and inboard strut station with and without power (flight idle (ram) and cruise conditions). The effect

Pressure at 45% Span   Pressure at 71% Span

o Wind Tunnel Test Data   TRANAIR
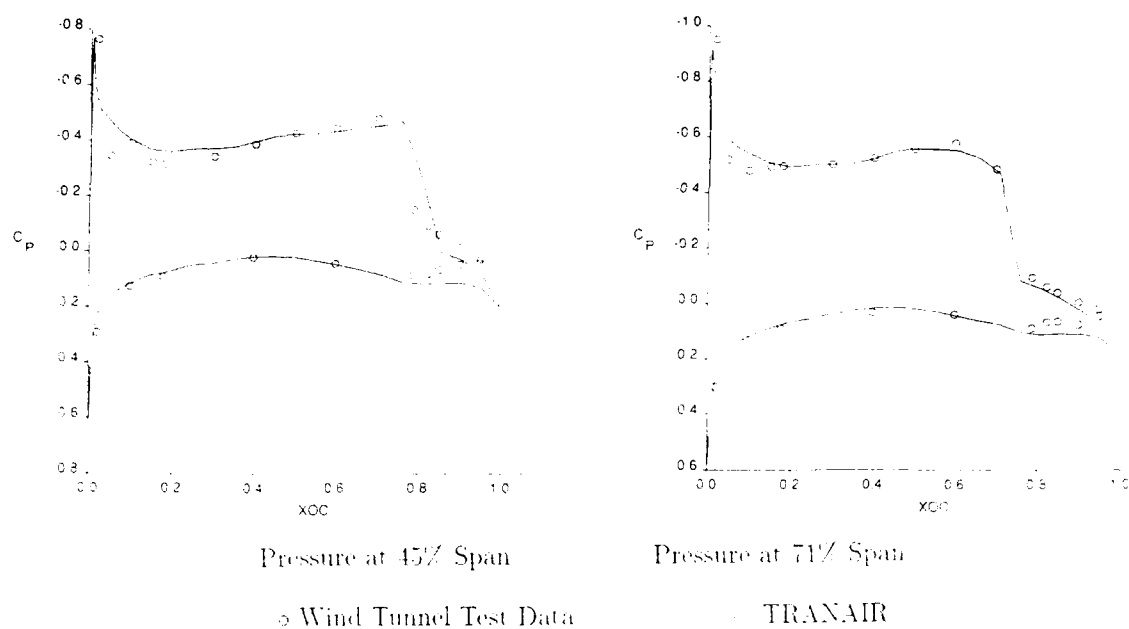
Figure 7: Wing Pressures for the F16, $M_\infty = 0.9, \alpha = 4.0°$.

a) Waterline Cut Through Wing   b) Vertical Cut

Figure 8: Cuts Through the Grid for the F16 With Tanks and Missiles, $M_\infty = 0.9, \alpha = 4.0°$.

Inboard Side of Strut   Outboard Side of Strut

---- F16 Without Tanks and Missiles   F16 With Tanks and Missiles
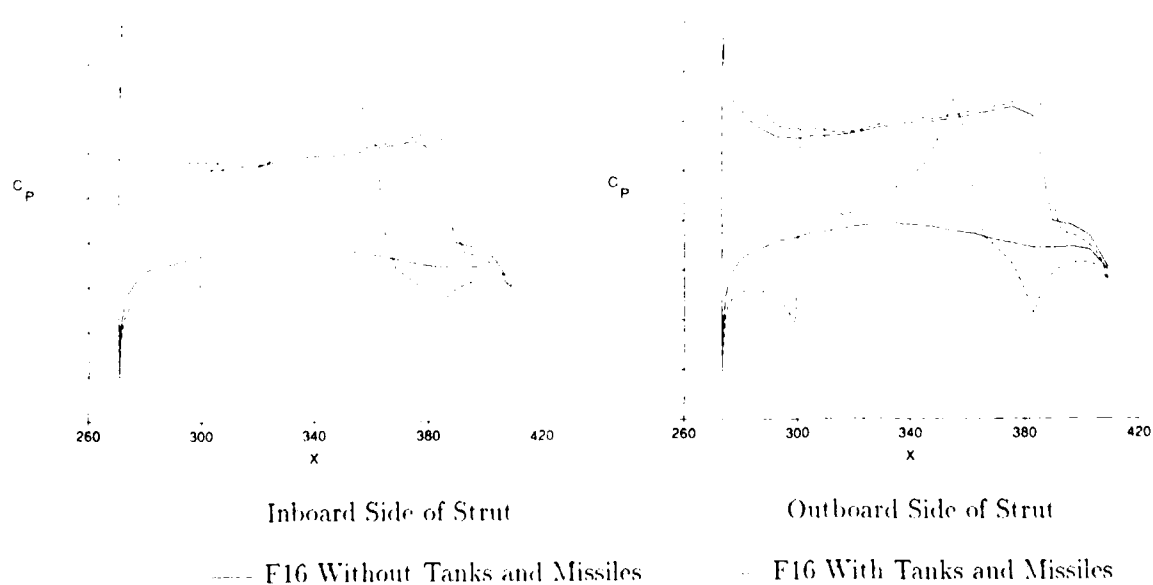
Figure 9: Computed Wing Pressures for the F16 with Tanks and Missiles, $M_\infty = 0.9, \alpha = 4.0°$.
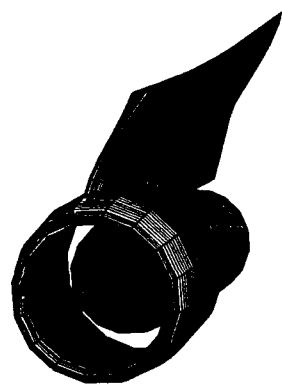
a) Inboard Nacelle and Strut
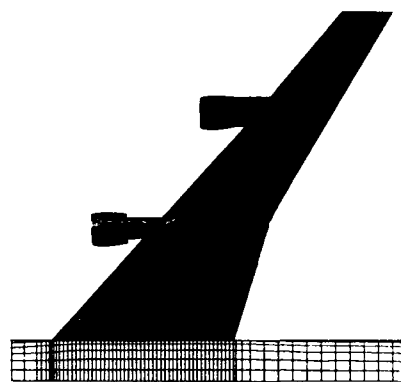
b) Top View of Wing and Body

Figure 10: 747-200 Transport Configuration.
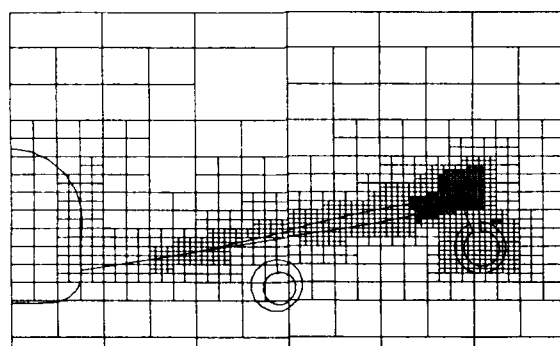
Constant x Cut Behind Inboard Nacelle

Figure 11: A Cut Through TRANAIR Grid for 747-200 Case.

26% Span

60% Span

69% Span
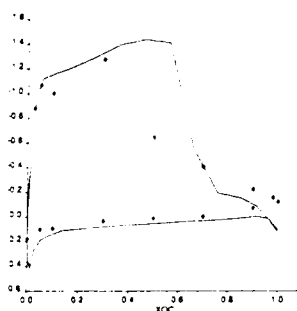
o   Wind Tunnel Test Data
—   TRANAIR

81% Span
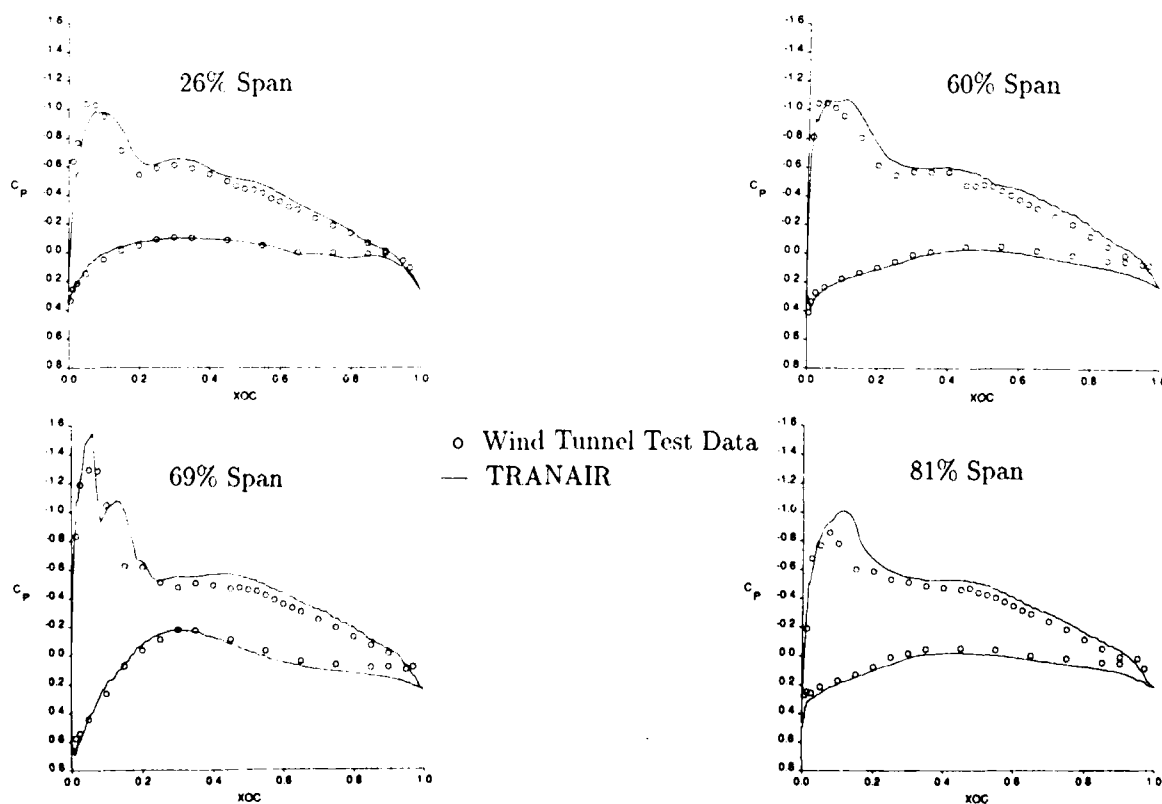
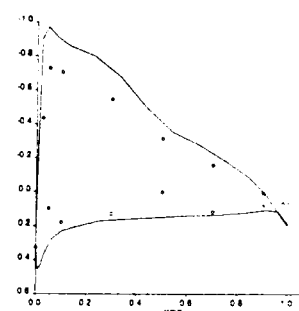Figure 12: Wing Pressures for 747-200, $M_\infty = 0.8$, $\alpha = 2.7°$.
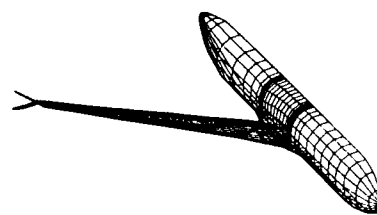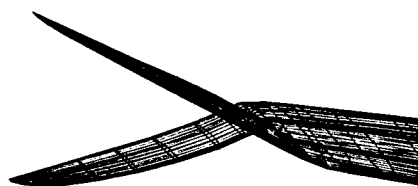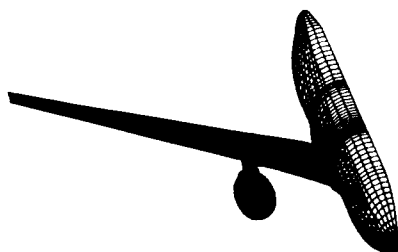
a) Configuration Paneling
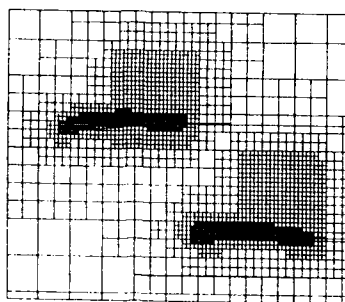
b) Sections in the Grid

—— TRANAIR

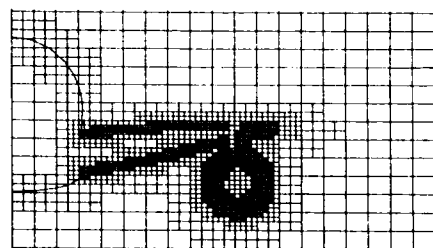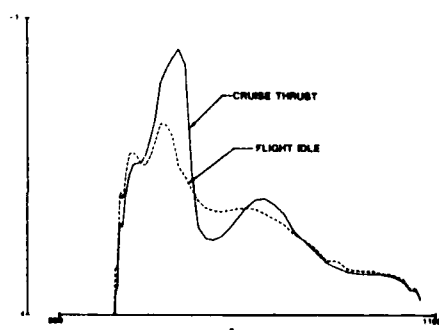◊ Wind Tunnel Test Data

c) Section Pressures

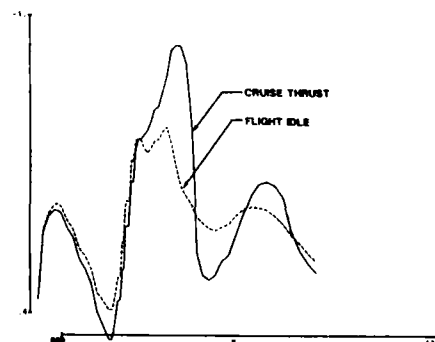Figure 13: TRANAIR Analysis of a Transport with Tip Feathers.

a) Surface Paneling

b) Section of TRANAIR Grid

c) Underwing Station

d) Inboard Strut Station

Figure 14: TRANAIR Analysis of a Transport with Wing/Body/Nacelle/Strut and Power.

of power on local flow is obvious. This case demonstrates the capability of the TRANAIR code to handle power effects, a capability usually associated with an Euler formulation.

Concluding Remarks

The method described in this paper is fully implemented for aerodynamics applications in a computer program called TRANAIR. The results shown in this paper demonstrate the flexibility of TRANAIR to handle complex geometries in transonic flow. With the ability to account for regions with different total pressure and total temperature the code is capable of handling engine power effects accurately. Reliability, generality, efficiency and usability of TRANAIR approach those of panel codes even though the flow now contains shocks and other nonlinear effects.

We are currently implementing adaptive grid refinement which will further enhance the efficiency, usability and reliability of this code. We plan to add a coupled boundary layer simulation to this code in the near future. In addition, we are also investigating a wake capturing scheme which has potential to capture sharp vortex sheets.

References

[1] F. T. Johnson, "A General Panel Method for the Analysis and Design of Arbitrary Configurations in Subsonic Flows", NASA CR-3079, (1979).

[2] A. Jameson, "Transonic Flow Calculations," Princeton University Department of Mechanical and Aerospace Engineering report 1651, 1983.

[3] R. Löhner, K. Morgan, and O. C. Zienkiewicz, "Adaptive Grid Refinement for the Compressible Euler Equations", Chapter 15 in **Accuracy Estimates and Adaptive Refinements in Finite Element Computations**, edited by I. Babuška, O. C. Zienkiewicz, J. Gago, and E. R. de A. Oliveira (John Wiley and Sons, New York, 1986), p. 281.

[4] B. Wedan and J. C. South, "A Method for Solving the Transonic Full Potential Equation for General Configurations," AIAA Paper 83-1889, 1983.

[5] P. E. Rubbert, J. E. Bussoletti, F. T. Johnson, K. W. Sidwell, W. S. Rowe, S. S. Samant, G. SenGupta, W. H. Weatherill, R. H. Burkhart, B. L. Everson, D. P. Young, and A. C. Woo, "A New Approach to the Solution of Boundary Value Problems Involving Complex Configurations," in **Computational Mechanics - Advances and Trends**, edited by Ahmed K. Noor (The American Society of Mechanical Engineers, New York, 1986), p. 49.

[6] H. Samet, "The Quadtree and Related Hierarchical Data-Structures", **Computing Surveys 16**, 1984.

[7] S. S. Samant, J. E. Bussoletti, F. T. Johnson, R. G. Melvin, and D. P. Young, " Transonic Analysis of Arbitrary Configurations using Locally Refined Grids", in **Proceedings of the 11th International Conference on Numerical Method in Fluid Dynamics**, (1988).

[8] D. P. Young, R. G. Melvin, M. B. Bieterman, F. T. Johnson, Satish S. Samant, and J. E. Bussoletti, "A Locally Refined Rectangular Grid Finite Element Method," Boeing SCA Report 108, 1989.

[9] H. Bateman, "Irrotational Motion of a Compressible Inviscid Fluid," **Proceedings of the National Academy of Sciences 16**, 816 (1930).

[10] O. Buneman, "Analytic Inversion of the Five-Point Poisson Operator," **J. Comput. Phys. 8**, 500 (1971).

[11] "TRANAIR Computer Code (Theory Document)", NASA Contractor's Report, 1987.

[12] M. Hafez, E. M. Murman, and J. C. South, "Artificial Compressibility Methods for Numerical Solutions of Transonic Full Potential Equations," AIAA Paper 78-1148, 1978.

[13] Y. Saad and M. H. Schultz, "GMRES: A Generalized Minimal Residual Method for Solving Nonsymmetric Linear Systems," Yale University Department of Computer Science Research Report YALEU/DCS/RR-254, 1983; **SIAM J. Sci. Stat. Comput. 7**, 856 (1986).

[14] D. P. Young, R. G. Melvin, F. T. Johnson, J. E. Bussoletti, L. B. Wigton, and S. S. Samant, "Application of Sparse Matrix Solvers as Effective Preconditioners", Boeing Computer Services Engineering and Scientific Services Technical Report ETA-TR-99, 1988

THE SOLUTION OF SCATTERING AND RADIATION PROBLEMS FOR COMPLEX CONFIGURATIONS BASED ON 3-D GRID AND PATCH MODELS

V. Stein

Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V.
Institut für Hochfrequenztechnik
8031 Oberpfaffenhofen, FRG

SUMMARY

The increasing interest in predicting the scattering and radiation characteristics of objects with a complicated structure has stimulated the development of several theories. A rigorous treatment of the electrodynamic problem requires the solution of a boundary value problem based on Maxwell's differential equations or on the equivalent integral equations. The application of rigorous methods for objects whose dimensions are large compared to the wavelength is limited by the required computer memory and execution time. Therefore, methods which solve the boundary value problem approximately come into consideration. Each type of solution method involves a typical model either of the surface or the volume of the structure and it's surroundings. So, geometric models consisting of canonical shapes, wire-grids, surface patches, ard volume cells are described and the requirements of the specific solution methods are discussed. In some cases estimations for the necessary modeling accuracy are given. Methods which are based on geometrical-optics principles require models where the surface parts which are illuminated by the incident wave and the surface parts which are hidden can be separated for each aspect angle. Such a procedure is discussed as well as the procedure to treat double reflections. Some computational examples for radiation and scattering processes are given and comparisons with measurements are made.

1. INTRODUCTION

In electrodynamics there is an increasing demand in predicting the radiated field of antennas installed on complicated structures or the scattered field of radar objects. Therefore, the extension of known methods and the development of new methods is stimulated to describe the interaction process between the electromagnetic wave and the structure under test. With the following figures some typical problems of electrodynamics are illustrated.

Fig. 1.1 shows a periscope within a sea surface. One is interested in the polarisation dependent radar cross-section of the object in this specific surroundings. The radar cross section is a far-field quantity, which means that it is to be determined at a distance which exceeds $2D^2/\lambda$. The object with dimension D is large against the wave length λ of the incident electromagnetic wave.
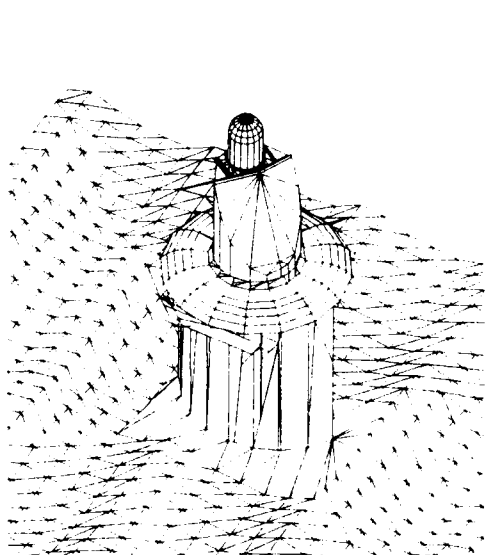

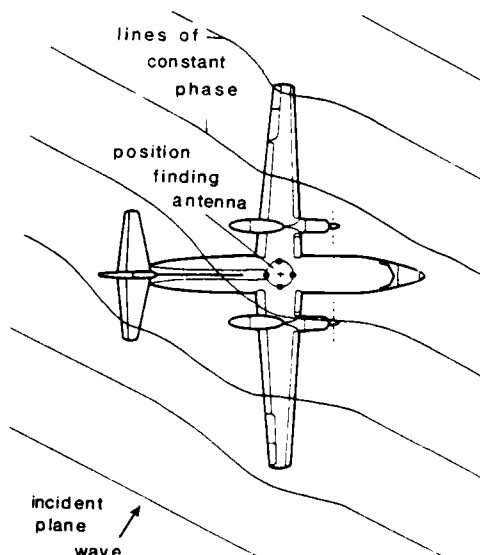Fig. 1.1 Periscope within a moving sea surface.

Fig. 1.2 Distortion of a plane wave due to the interaction between the incident electromagnetic wave and the structure.

Fig. 1.2 shows an airplane which is equipped for position finding purposes. Since the phase front of the incoming plane wave is distorted by the airplane structure, the question arises, where an appropriate place on the airplane for the installation of the direction finding antenna can be found. The magnitude of the distortion of the wave front in dependence from frequency, polarization and angle of incidence is of interest. The solution of this problem requires the computation of the scattered near-field, since the dimensions of the object are in the order of the wavelength and the position finding antenna is installed at a distance above the surface which amounts to a fraction of a wavelength.

Fig. 1.3 shows a reflector antenna for a satellite earth station. The radiation forming parts are the hornparabola, the subreflector and the main reflector. The spherical wave which propagates in the horn is reflected by the parabolic reflector of the horn towards the subreflector and transformed into a plane wave. Since the subreflector also is parabolic the wave is again transformed into a spherical wave. A

third transformation by the parabolic main reflector finally generates the radiated plane wave. It takes place if the focus of the main reflector coincides with the phase center of the spherical wave going out from the subreflector. The computation of the antenna characteristics is a typical far-field problem since the dimensions of such an antenna are much greater than the wavelength.
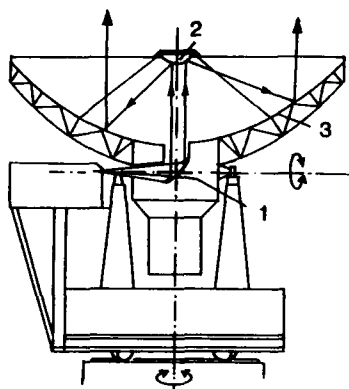
Fig. 1.3 Principle of a satellite earth station antenna (cassegrain type), 1 hornparabola, 2 subreflector, 3 main reflector.
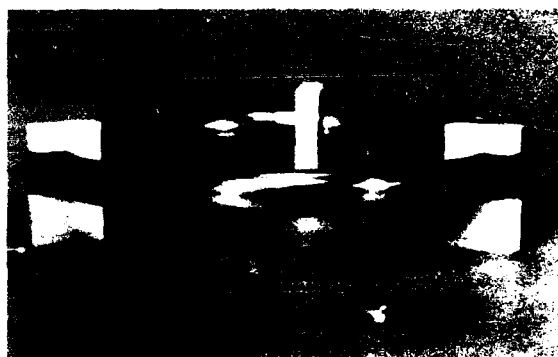
Fig. 1.4 Torus antenna, capacitive coupled to the inner conductor of a coaxial line.

Fig. 1.4 finally shows a torus antenna mounted above a circular plate and excited by a small flat plate. Such a device is a very compact radiator since the circumference of the torus antenna is in the order of a wave length. It finds use as a primary radiator in a central fed reflector. Usually one is not only interested in the far-field characteristics of such an antenna, but also in the input admittance which is a near-field quantity.

In the past only approximate theoretical or experimental methods were available for prediction purposes. While approximate methods can fail in giving sufficient accurate predictions of the interaction process, experimental methods require a high effort especially if changes of the structure must be carried out for optimization purposes. In these cases it is desirable to reduce the number of experimental studies to a few final measurements, which can be defined by the preceding theoretical analysis. This requires from the theory to develop and validate methods which are able to describe the interaction of electromagnetic waves with a complicated structure with an accuracy, which is sufficient for practical applications.

The progress in computer techniques permits in an increasing manner the use of theories, which because of their high numerical effort could not be considered in the past. There is a great variety of methods which in principle come into question. Each method has its specific advantages and drawbacks depending from the type of the problem. Among these methods there are as well as heuristic methods like the physical optics method, and the geometrical theory of diffraction as rigorous methods, which solve Maxwell's differential equations directly or undertake the solution of the equivalent integral equations. The choice of geometric models is influenced by the chosen solution method. Within the context of this paper it is not possible to describe the electrodynamic methods in detail. More informations may be obtained by the cited references. A rough classification of the theories can be given as follows.

In the radar case the distance between the source of the incident wave and the object is much greater than the dimensions of the object. The antenna case is characterized by the fact that the source of the incident wave is either within the radiating structure itself (e.g. a feeding gap of a dipole antenna) or in the immediate neighbourhood of the scattering object (e.g. an antenna on an airplane). From the standpoint of a rigorous solution of Maxwell's equations there is no difference between a radiation and a scattering problem. In the following the more general term scattering problem will be used for both types of problems.

If one is interested in the description of near-field characteristics, e.g. the scattered field in the immediate neighbourhood of the structure or the field at the surface of the structure itself, only rigorous methods can be used. If the dimension of the structure, however, exceeds the order of several wavelengths (high frequency case), rigorous methods will fail in practice because of the required high computer effort. Fortunately the main contributions to the scattered field in this case originate from parts of the structure which are illuminated from the incident wave or which give rise to double or multiple reflections. These phenomena may be treated by geometrical optics. Wedges in the structure give rise to diffraction processes which can be described by an asymptotic evaluation of the rigorous theories for large distances and large dimensions of the structure. This concept leads to the development of heuristic methods for the high frequency case. So a very important criterium for the choice of the solution methods is the dimension of the object referred to the wavelength, that is the parameter D/λ.

The solution of Maxwell's equations for the electric field $\vec{E}$ and the magnetic field $\vec{H}$ can be carried out for the volume which surrounds the scatterer. For this purpose it is necessary not only to model the surface of the scatterer but also the surrounding volume. This leads to one type of geometric models. Maxwell's differential equations can be transduced in equivalent integral equations over the tangential fields at the surface of the scatterer. This leads to a class of geometric models which have to represent only the surface of the scatterer. Following the publications in electrodynamics the use of surface models is more common than the use of volume models. In the subsequent sections several geometric models are discussed which are typical for the individual electrodynamic solution procedures. Models which are suited for solutions with the integral equation or the physical optics method are emphasized.

2. MODELING WITH CANONICAL SHAPES

The structure is considered to be an ensemble of components, each of which can be geometrically approximated by a simple shape. A coarse model of an airplane for example may be established by a cylinder, flat plates, section of a sphere, see Fig. 2.1. For a certain class of canonical shapes, in particular wedges and smooth surfaces, theories are available to determine the scattered field for arbitrary aspect angles, from which the radar cross-section can be evaluated. Several problems will arise. The first problem consists in combining the contributions of the individual shapes by proper phasing to the total field or cross-section. A further problem arises by the fact that in dependence from the aspect angle one canonical substructure may hide completely or partially the other one. A third problem concerns the electromagnetic interaction of one canonical shape with the other.
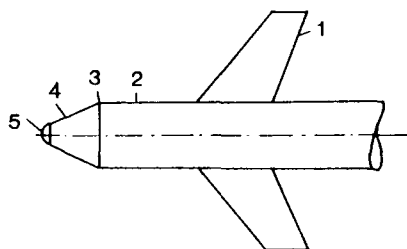
Fig. 2.1
Model of an airplane with simple shapes.

1 flat plate

2 cylinder

3 wedge

4 conus

5 sphere

A procedure which overcomes the first problem consists in combining the contribution of the several canonical shapes by random phase [1]. This is based on the assumption that the different phases of the canonical structures are randomly distributed; then upon averaging over the phases one obtains for the radar cross-section σ_t of the total object, which is composed of N canonical shapes with cross-sections σ_j, the expression

$$(2.1) \qquad \sigma_t = \sum_{j=1}^{N} \sigma_j .$$

In connection with this method of approach one can estimate the amount of probable deviation from the average cross-section σ_t by employing the root mean square spread. This measure of the probable variation in cross-section due to relative-phase effects leads to the following bounds: $\sigma_t \pm s$, where

$$(2.2) \qquad s^2 = (\sum_{j=1}^{N} \sigma_j)^2 - \sum_{j=1}^{N} \sigma_j^2 .$$

The random phase method is designed to give estimates of the amount by which the cross-section might deviate from the average value because of phase effects. If one is interested in finding an order-of-magnitude estimate of the cross-section as a function of aspect angle the random-phase procedure would be adequate. Since only cross-sections play a role one also can use experimental data for such shapes for which no theoretical values are available. A further advantage of this method concerns the computer effort since the evaluation of the radar cross-section for canonical shapes can be done analytically. The values of the individual shapes can also be calculated in advance for discrete elevation and azimuth angles and together with the experimental data for the non-canonical shapes properly arranged in a data file. For arbitrary aspect angles the cross-sections can be evaluated by interpolation schemes.

If one is interested not only to estimate the average value and the amount by which the cross-section deviates from the average value but also to predict the peaks and nulls of the radar cross-section in dependence from frequency, polarization and position of the observer point one has to combine the scattered field of the canonical shapes with their relative phases referred to a common reference point. This makes the use of experimental data more difficult.

At this time the most capable method in electrodynamics which is based on canonical shapes and solves the relative phase problem is the geometrical theory of diffraction (GTD) [12]. The GTD is a ray optical technique and is, therefore, bound to such applications where the object's dimensions are much greater than a wavelength. The GTD makes use from the rigorous solutions for canonical shapes in such a way that a diffraction coefficient D is evaluated which connects the diffracted field with the incident field. This diffraction coefficient plays a similar role as the well-known reflection coefficient R in optics. The most important canonical shapes of the GTD are shown together with typical ray paths in Fig. 2.2. For the wedge and the smooth surface the diffraction coefficients are known in case of perfectly conducting bodies while the reflection coefficient can be determined also for nonperfectly conducting multilayered panels, see Sec. 7.1.

The rays which undergo reflection and diffraction at the structure and reach the observer point are to be determined from geometrical and differential geometrical considerations. This ray tracing procedure is illustrated at hand of Fig. 2.3. A source (slot antenna) is installed at the top of the fuselage of an airplane. For a given observer direction the following rays are depicted:

- direct ray,
- reflected ray, the reflection occurs at the surface of the left wing,
- wedge diffracted ray, the diffraction occurs at the trailing edge of the left wing,
- surface diffracted ray (creeping wave) which encircles the fuselage and is then radiated toward the observer point.

The contribution of the several rays are summed up with correct amplitude and phase. If multiply reflected and diffracted rays are taken into account the complexity increases considerably. The ray tracing

for arbitrary positions of source and observer is one of the main problems of this technique. Nevertheless, the GTD besides the physical optics method is the most promising approach for the solution of high frequency problems.
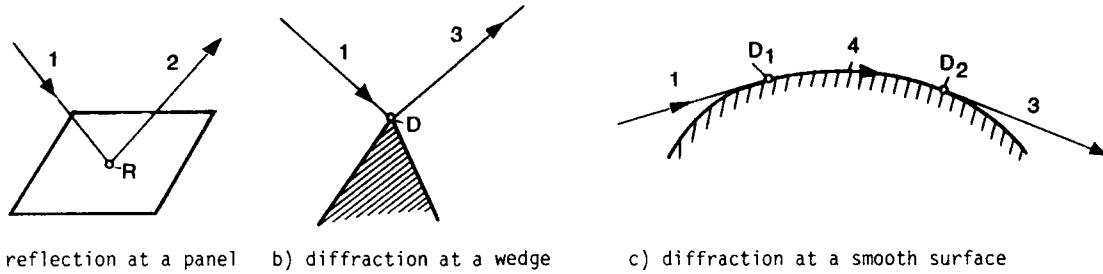

a) reflection at a panel   b) diffraction at a wedge   c) diffraction at a smooth surface

Fig. 2.2 Canonical shapes and interaction processes. 1 incident ray, 2 reflected ray, 3 diffracted ray, 4 creeping wave.
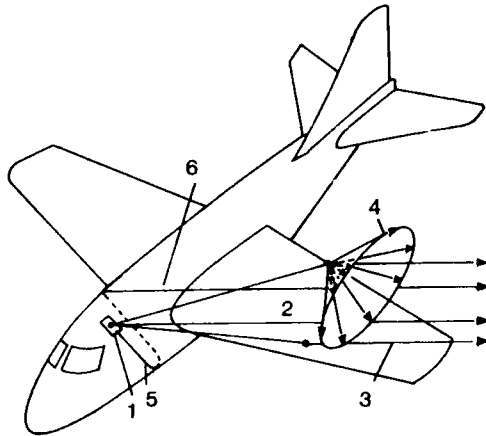

Fig. 2.3
Ray paths.
1 slot antenna
2 direct ray
3 reflected ray
4 cone of diffracted rays
5 geodesic path
6 surface diffracted ray

3. MODELS USED IN THE INTEGRAL EQUATION APPROACH

3.1 REMARKS TO THE ELECTRODYNAMIC THEORY

The electrodynamic processes which are expressed by the differential equations of Maxwell under satisfying the boundary conditions can be formulated by equivalent integral equations (IE) for the fields tangential to the surface of the scatterer. For a perfectly conducting scatterer one obtains the following integral equations which each independently from the other can be applied to solve for the surface tangential field. One of the integral equations is denoted as the electric field integral equation (EFIE) and given by [12, 14]

$$(3.1) \qquad \vec{n}(\vec{r}) \times \vec{E}_e(\vec{r}) = -\frac{1}{4\pi j \omega \varepsilon} \vec{n}(\vec{r}) \times \oint_F ((-1-jkR+k^2R^2) \vec{J}_F(\vec{r}') + (3+3jkR-k^2R^2)(\vec{J}_F(\vec{r}')\cdot\vec{e}_R)\cdot\vec{e}_R) \frac{e^{-jkR}}{R^3} \, df' \, ,$$

the other is denoted as the magnetic field integral equation (MFIE) and expressed by

$$(3.2) \qquad \vec{J}_F(\vec{r}) = 2\vec{n}(\vec{r}) \times \vec{H}_e(\vec{r}) + \frac{1}{2\pi} \vec{n}(\vec{r}) \times \oint_F (1+jkR)(\vec{J}_F(\vec{r}') \times \vec{e}_R) \frac{e^{-jkR}}{R^2} \, df' \, .$$

The geometric magnitudes $\vec{r}$, $\vec{r}'$, $\vec{R}$, $\vec{e}_R$, $\vec{n}$, F, df' are illustrated in Fig. 3.1. The magnitudes ω, ε, k are explained in Sec. 7.1. $\vec{E}_e$ resp. $\vec{H}_e$ is the incident electric resp. magnetic field at the observer point which is situated on the surface of the scatterer. In radar problems $\vec{E}_e$, $\vec{H}_e$ is supported by a plane wave which comes from infinity. In the antenna case $\vec{E}_e$, $\vec{H}_e$ is the outgoing field of a feeding gap (cylinder antenna) or of an aperture (horn antenna). The unknown $\vec{J}_F$ is the electric surface current which is identical with the tangential component $\vec{n}(\vec{r}')\times\vec{H}(\vec{r}')$ of the total magnetic field which for its part is the sum of the incident field $\vec{H}_e$ and the scattered field $\vec{H}_s$.
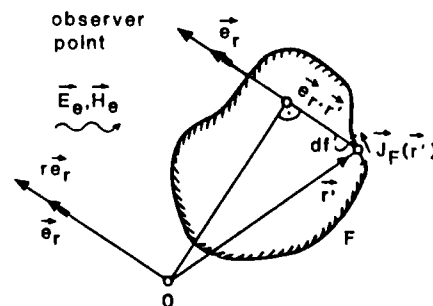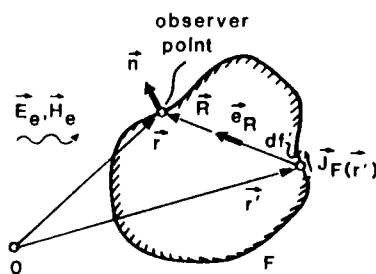

Fig. 3.1 Geometry for the evaluation of the EFIE, MFIE.   Fig. 3.2 Far-field geometry.

The EFIE is an integral equation of the first kind, the MFIE an integral equation of the second kind. In principle both integral equations may be used independently from another to determine the surface current. However, in practice it turns out that there are significant differences depending from the geometry of the scatterer. The MFIE has advantages if the structure consists of smooth surfaces (e.g. an airplane), whose radius of curvature are large compared to the wavelength. The first term on the right-hand side represents the physical optics current distribution, see Sec. 4. The contribution of the integral can be considered as a correction term to the physical optics theory. The MFIE cannot be used for structures which are very thin compared to the wavelength like thin cylinders or thin shells. In these cases the EFIE must be used. The use of the EFIE is also recommended if the scatterer shows a lot of structural details (e.g. a tank or a helicopter).

The integral equations can be transduced to linear systems of equations by the method of moments. For this purpose a set of known basis functions (expansion functions) with unknown complex coefficients are introduced according which the surface current is expanded. With a second set of known testing functions (weighting functions) both sides of the equations are multiplied and then integrated over the region of the structure. These steps result in a linear system of equations for the unknown coefficients of the basis functions. The right-hand side contains the known incident field depending from aspect angle and polarization. The matrix of the system of equations, frequently denoted as impedance matrix, depends only from the frequency and from the geometry of the structure.

The computer effort of the integral equation method is due to the calculation of the matrix elements and to the matrix inversion. The calculation of the matrix elements requires in general two integrations, one for the evaluation of the original surface integral over the basis functions, the other for the construction of the moments with the testing functions. Both integrals in general must be evaluated numerically. Very often the testing functions were chosen to be Dirac functions. In this case the value of the second integral is represented by the integrand itself and one speaks from the point-matching method. The method of Galerkin is characterized by testing functions which are identical with the basis functions.

An important feature of the integral equation approach consists in the fact, that besides the modeling of the structure no more geometric problems must be solved if the direction of the incident wave varies. This is due to the magnitude R in the field equations which represents the geometric distance between the integration point and the observer point on the surface. This is also true if there exists some part of the structure between these points.

If the matrix of the system of equations is inverted directly the surface currents can be calculated for each aspect angle and polarization of the incident wave by merely multiplying the right-hand column vector with the inverse. This solution method should, therefore, be preferred if for only a few frequencies a lot of changes of the incident field are foreseen.

If the surface current is known a further integration has to be carried out in order to compute the scattered far-field, from which the scattering matrix T and the polarization dependent radar cross-sections may be derived. The expressions for the far-field are given by:

$$(3.3) \qquad \vec{E}_s(\vec{r}) = \frac{j\omega\mu}{4\pi} \frac{e^{-jkr}}{r} \int_F (\vec{e}_r \times (\vec{e}_r \times \vec{J}_F(\vec{r}'))) e^{jk\vec{e}_r \cdot \vec{r}'} \, df' \ ,$$

The far-field geometry is represented in Fig. 3.2.

The components of the backscattered field can be related to the components of the incident field by a scattering matrix [T] in the following manner:

$$(3.4) \qquad \begin{bmatrix} E_{sx} \\ E_{sy} \end{bmatrix} = [T] \begin{bmatrix} E_{ex} \\ E_{ey} \end{bmatrix}, \qquad \text{with} \qquad (3.5) \qquad [T] = \frac{1}{\sqrt{4\pi r^2}} \begin{bmatrix} t_{xx} & t_{xy} \\ t_{yx} & t_{yy} \end{bmatrix} .$$

For this representation it has been assumed that the z-axis of a cartesian coordinate system with origin in the neighbourhood of the object is directed towards the radar observer. The elements t_{ij} of the scattering matrix are given by

$$(3.6) \qquad t_{ij} = \sqrt{4\pi r^2} \frac{E_{si}}{E_{ej}} \ , \qquad r \to \infty \ , \qquad i = x,y, \qquad j = x,y \ .$$

The polarization depend cross-sections then are given by

$$(3.7) \qquad \sigma_{ij} = t_{ij} \, t_{ij}^* \ .$$

An analytical evaluation of the scattering matrix is not possible in general. Since the integral equation method determines the current distribution at the surface of the structure a 3-D model of only the surface is required. There are two principal ways to establish surface models. One way consists in modeling a solid surface body with a grid of wires, the so-called wire-grid model. The other common approach breaks the surface up into patches or cells each having a continuous metallic surface. Both models are discussed in the following two sections in more detail.

3.2 WIRE-GRID MODEL

In the field of antennas there are a variety of structures which consist of wires like a corner reflector antenna, see Fig. 3.3 or a Yagi-Uda-antenna, see Fig. 3.4. For both antennas the wire technique is used to generate specific antenna characteristics.
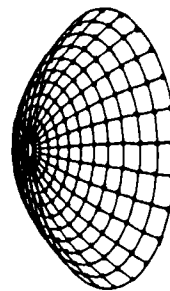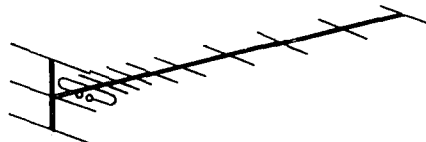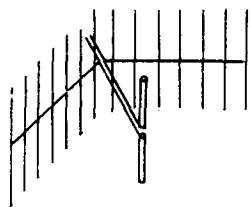
Fig. 3.3 Corner reflector antenna.   Fig. 3.4 Yagi-Uda array.   Fig. 3.5 Radar antenna reflector.

Wire-grid meshes also find many uses in applications where the effect of a solid conducting surface is required but the weight and/or wind resistance of the latter must be avoided. They may be used, for example, to fabricate radar antenna reflectors, see Fig. 3.5, and as shields to screen sensitive equipment from stray fields [8].

If any of the three structures would be modeled by a wire-grid, the model would perfectly agree with the original structure. The substitution of an arbitrary solid surface by a wire-grid model depends upon the fact, that as the mesh size becomes smaller relative to the shortest wavelength of concern, the mesh supports a surface current distribution which approaches that on the continuous surface. The current is only an approximation to the actual current, however, and as such it can be expected to reasonably predict the far-fields but possibly not the near-fields. This is due to the fact that the grid supports an evanescent reactive field on both sides of its surface. An actual continuous surface is not capable of supporting such a field [18].

If thin (compared to the wavelength) wires are chosen to construct a wire-grid the current essentially has only an axial component. In this case the EFIE simplifies considerably. The MFIE, however, will fail in the thin-wire approximation. The thin-wire EFIE is given by

$$(3.8) \qquad \vec{s} \cdot \vec{E}_e(s) = - \frac{1}{4\pi j \omega \varepsilon} \vec{s} \cdot \int_L \left((-1-jkR+k^2R^2) \, \vec{s}' + (3+3jkR-k^2R^2)(\vec{s}' \cdot \vec{e}_R)\vec{e}_R \right) I(s') \frac{e^{-jkR}}{R^3} \, ds' \, ,$$

where $\vec{s}$ is the unit tangent vector of the wire at the observer point and $\vec{s}'$ is the unit tangent vector of the wire at the integration point. Fig. 3.6 shows two wires i and j from a wire-grid for which the interaction after discretization of Eq. (3.8) is computed. j is the wire with the integration point and i the wire with the observer point. For more details, see [14].

[10] was apparently the first report on the application of the thin-wire EFIE to the analyses of wire-grid models for circular disks and spheres. Satisfactory agreement was demonstrated between the wire-grid results and independent analytical or experimental back-scatter cross-section data presented as a function of frequency.

Since the number of the unknown of the linear system of equations is identical with the number of wire segments an estimation of the minimum mesh width is of great interest. This is primarily dependent on the choice of basis and test functions.

A good estimation of the mesh width is obtained, if a source is positioned within a structure, which is modeled by a wire-grid with variable mesh width. The field in the exterior of the structure, which should be zero, is then computed in dependence on the mesh width.
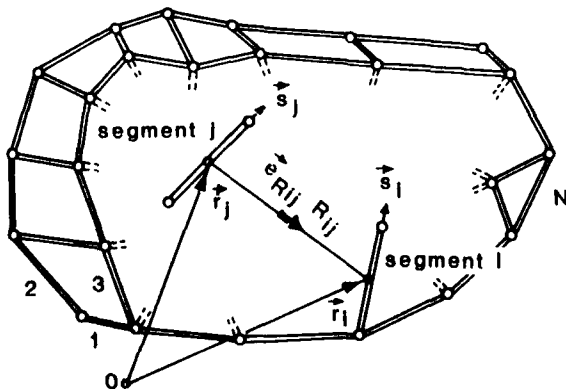

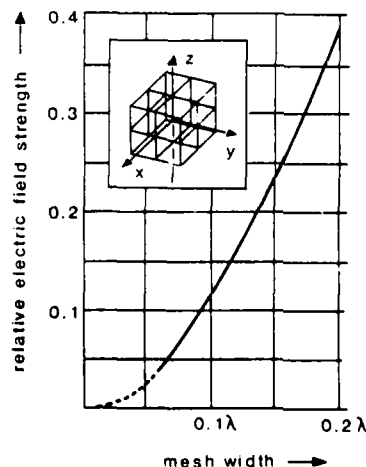Fig. 3.6 Geometric situation of two wires i and j of the ensemble of N wires.

Fig. 3.7 Influence of the mesh width.

For this theoretical experiment a cube with an edge length of 0.4 λ has been chosen. In the center of the cube a point source was positioned. The diameter of the wire was 0.003 λ, so that the thin-wire approximation of the EFIE holds. The basis functions are chosen to be pulses, the boundary conditions were satisfied in the center of the wire (Dirac functions as testing functions).

Fig. 3.7 shows the electric field in the exterior in dependence from the mesh width. A mesh width of 0.057 λ corresponds to about 600 wire segments. One can assume, that a mesh width of about 0.1 λ in modeling a structure should be sufficient for far-field computations. This means, that about 200 wire segments are necessary to model a surface with a size of λ × λ. Since per segment only one current coefficient has to be determined the dimension of the linear system of equations is given by

(3.9a) $N \approx 200 \, F/\lambda^2$,   F = surface of the scatterer.

Shorter wire segments, 0.05 λ or less, may be needed in modeling critical regions of a structure.

More sophisticated studies show, that the accuracy with which a wire-grid model simulates an actual surface depends on the computer code (i.e., expansion and weighting functions) used, the radius of the wire segments used, as well as the grid size. For example, with pulse basis functions it has been found that a grid spacing of about 0.1 λ to 0.2 λ yields good results. With the piecewise sinusoidal Galerkin method, it has been found that the grid size should not exceed λ/4 and that a suitable wire radius is a = w/25 where w denotes the width or length (whichever is greater) of the apertures [18]. A grid size of λ/4 would lead to the following expression for the dimension of the linear system of equations:

(3.9b) $N \approx 32 \, F/\lambda^2$.

From the above remarks concerning the substitution of a solid surface by a wire-grid system one concludes that the far-field properties of the structures such as the radar cross-section or the antenna radiation pattern can be predicted with sufficient accuracy. This is demonstrated at hand of the computed far-field pattern of an antenna system installed on a helicopter of the type BO 105.

Fig. 3.8 presents the details of the actual structure with the position of the two λ/4-monopoles operating in the VHF-band. Fig. 3.9 shows the wire-grid model. The antenna on the right-hand side of the flight direction was driven, the other was terminated. The in flight-measurement of the radiation pattern in the horizontal plane for a frequency of 117.6 MHz is presented in Fig. 3.10 by the dashed line. The flight direction is defined by $\varphi = 0°$.

It could be shown that the immediate neighbourhood of the antennas, that is the shape of the top side and the drive for the blades, must carefully be modeled while the farer parts of the helicopter, especially the lower part could be approximated only roughly or even completely neglected. The actual thickness of the shaft of the drive was taken into account. In principle a monopole array, consisting of a driven and two parasitic excited monopoles, a thin and thick one, over a finite plane, was analyzed. The theoretical results are illustrated in Fig. 3.10 by the solid line. For more details see [12].

Such studies, while illustrating the applicability of wire-grid meshes as models for solid surfaces in terms of their far-field electromagnetic behavior are not entirely convincing as to the use of wire-grid models to determine near-field quantities such as current distributions [8]. Preliminary studies in this regard to compare the results obtained with independent theoretical results or with experimental results are not yet conclusive. Such comparisons should do much to more clearly define areas of applications and limitations of wire-grid models. A special problem seems to be the stability of numerical results, that is the independence on the solution from the number of wires.

This is demonstrated at hand of a simple dipole-antenna of length 0.45 λ and diameter of 0.014 λ. Fig. 3.11 shows the input admittance (proportional to the current at the source point) which is a near-field quantity in dependence from the number of wires which are used to model this dipole [24]. The upper diagram represents the real part (conductance), the lower diagram the imaginary part (susceptance) for two feeding models. The dashed line is based on a voltage-source model, the solid line on a frill-current model. One realizes that the conductance for both source models tends to a finite value in dependence from the number of wires. The susceptance, however, does not reach a stable value in both cases. This behavior may be due to either the source models or the wire approximation of the dipole or to the specific combination of basis and testing functions. Further studies are in progress.
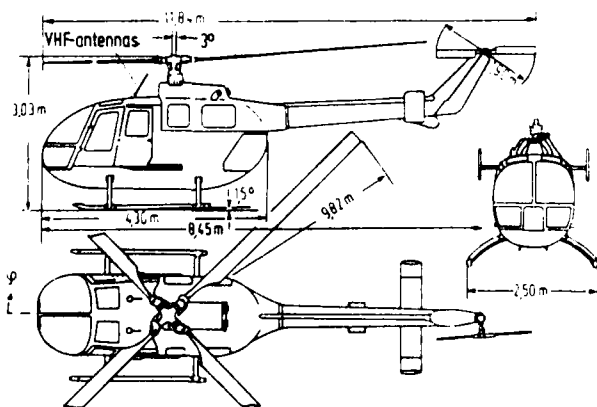

Fig. 3.8
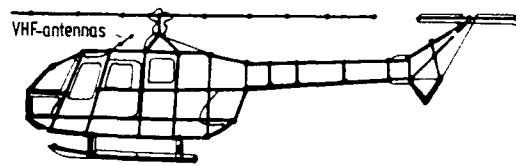Position of the VHF-antennas on the helicopter.
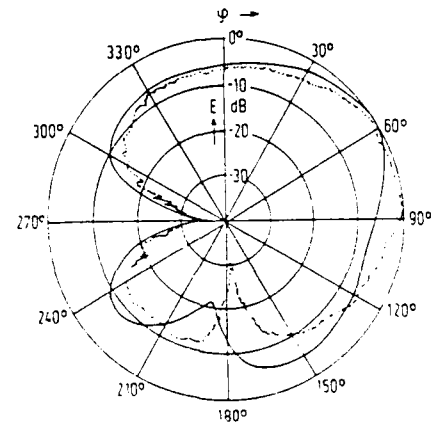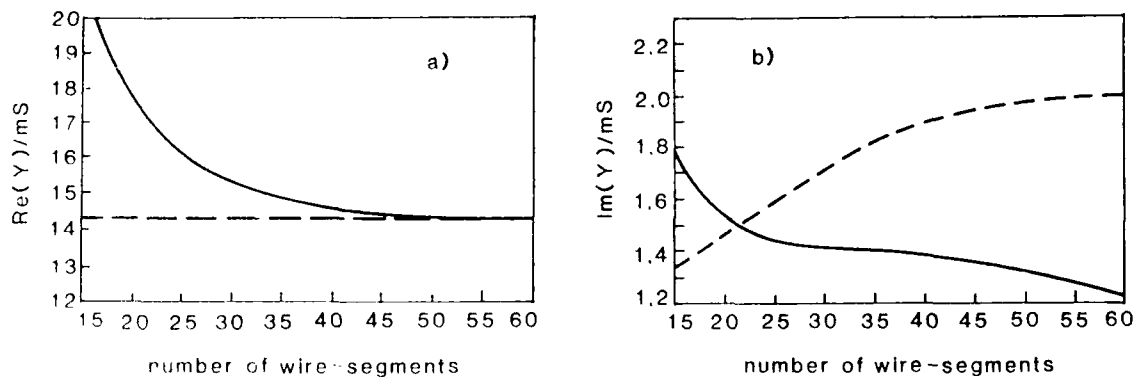
VHF-antennas

Fig. 3.9 Wire-grid model of the helicopter.

Fig. 3.10 Comparison between in flight-
measurements (dashed line) and results of
the IEM (solid line), frequency 117.6 MHz.

a)

b)

number of wire-segments

number of wire-segments

Fig. 3.11 Input admittance Y of a dipole antenna in dependence from the number of wire-segments;
parameter: — current-source, --- voltage-source. a) conductance, b) susceptance.

3.3 SURFACE PATCH MODEL

The continuous surface model by patches is an alternative approach for the modeling of 3-D complex
structures. It is mainly used for smooth surfaces. Mostly the patches are chosen to be flat plates (panels)
of triangular or rectangular shape. In principle both integral equations may be used. Following the
publications the MFIE is more frequently used than the EFIE. This is due to the integrand which is more
simple in the case of the MFIE. The use of pulse functions as basis functions predominates.

The surface of the smooth body is modeled by N^* surfaces patches with individual sizes ΔF. Fig. 3.12
shows such a model, where two surface patches q and p are chosen to demonstrate the geometric parameters
required in the discretized form of the EFIE or MFIE. Patch q contains the integration point, patch p the
observation point. In contrast to the wires of the grid model two current components have to be determined
in order to describe the direction of the surface current. For this purpose a local coordinate system
$(\hat{e}_\xi, \hat{e}_\eta, \hat{n} = \hat{e}_\xi \times \hat{e}_\eta)$ is introduced for each panel.
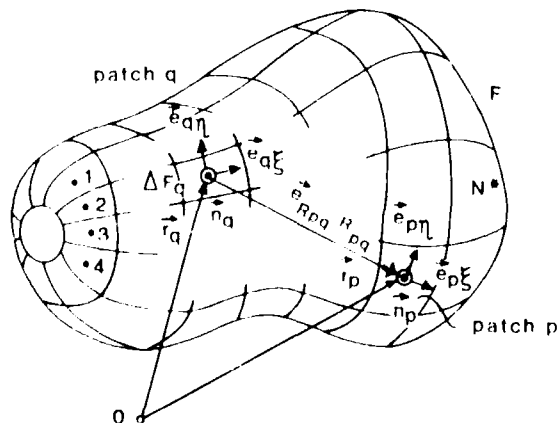
patch q

patch p

Fig. 3.12
Geometric situation of two patches p and q of
the ensemble of N^* patches used for a continuous
surface model of a structure.

The experience shows that far-field problems will be solved with sufficient accuracy if one models a
surface of one square wavelengths by at least 20 to 30 patches. This means that the edge length of a sur-
face patch should have an amount of less than about $\lambda/5$ (maximum size of about 0.04 square wavelengths).
This estimation is based on pulse functions as basis functions and Dirac-functions as testing functions.
The dimension N (number of complex unknowns) of the linear system of equations to be solved is determined
as follows: a patch size of about $\lambda/5$ results in 25 patches for one square wavelength of the scatterer's

surface. For each patch 2 complex current coefficients have to be determined. Therefore, the dimension of the complex linear system of equations can be estimated by

(3.10) $N = 2N^{*} \approx 50 \, F/\lambda^{2}$.

An estimation of the computer time τ for matrix inversion in dependence on the surface area F of a general 3-D structure is presented in Table 3.1. In the second column of the table the edge length a/λ of an equivalent cube as a representative for a general 3-D structure is given and in the third column the number N of the complex unknowns which have to be determined are listed. It is assumed that an economic programming under introducing of block structures has been achieved. From this table one can realize that the limit of application of the integral equation method with respect to the structure's dimensions is drawn by the required computer effort.

F/λ^2	a/λ	N	τ/s
10	1,29	500	< 50
20	1,82	1000	50
30	2,23	1500	180
40	2,58	2000	350
50	2,89	2500	700
60	3,16	3000	1200

Table 3.1
Matrix inversion time (computer Cray-1) in dependence on the surface area F of a general 3-D structure.

The figure N is in the same order of magnitude as it is in the wire-grid model. In the case of near-field considerations much finer modeling has to be chosen. Since the surface patch model in principle should be suited to compute near-field quantities one has applied the surface patch model for the calculation of the lines of constant phases in the extreme near-field of an airplane (type Do 228). For the solution of this problem the MFIE was chosen with pulse functions as basis functions and Dirac-functions as testing functions. Fig. 3.13 shows a surface patch model of 280 patches for half of the airplane. The patches are quadrangles and triangles.

The lines of constant phases, represented in Fig. 3.14, are calculated for a wavelength of $\lambda = 6.02\text{m}$ and a distance of $\lambda/30$ from the upper part of the fuselage. The plane wave is incident towards the nose of the airplane under an elevation angle of 30° over the horizontal plane. The polarization vector is vertical to this plane. The increment between the lines of constant phases has an amount of 10°.
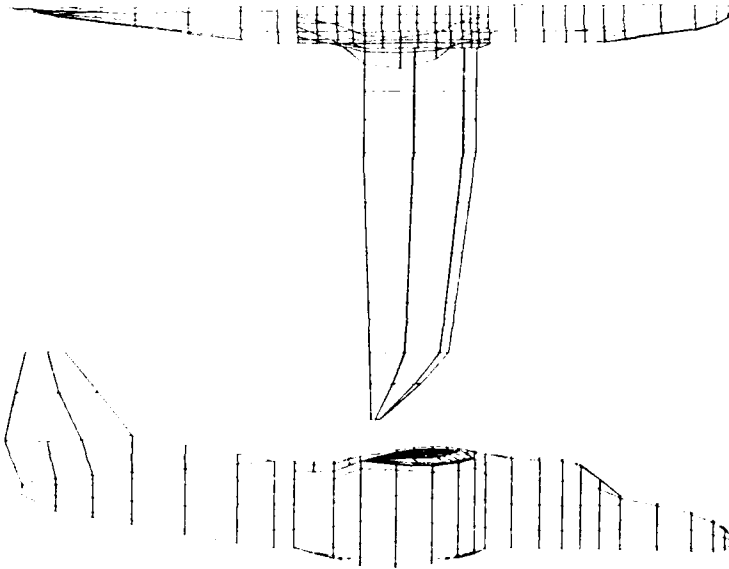
Fig. 3.13
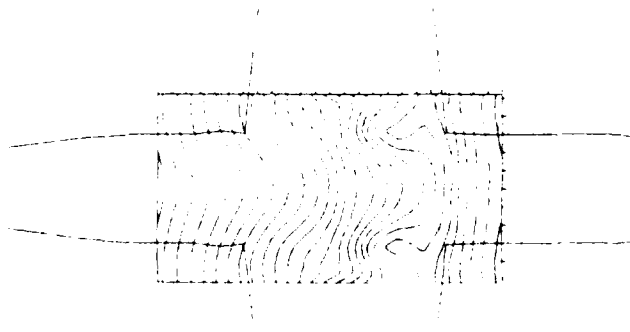Surface patch model of an airplane (type Do 228).

Fig. 3.14
Lines of constant phases in the near-field of an airplane.

Since direct experimental validations require a high effort the method was verified at hand of a cube with edge length a = λ. One side of the cube was subdivided in 5 × 5 quadratic panels for measurement purposes, see Fig. 3.15. In the middle of each panel a hole was bored by which probes for the surface fields could be positioned from the interior of the cube. For computational purposes the number of the panels was varied until a stable result was attained.

The comparison between measurement (circle) and theory (solid line) is drawn in Fig. 3.16. Fig. 3.16a represents the amplitude and Fig. 3.16b the phase of the surface current on the front of the cube.

The component of the current distribution is chosen to be parallel to the vector of the field incident vertically on the front. Amplitudes and phases are relative values. Parameter of the curves is the row of holes defined by the normalized z-coordinate.
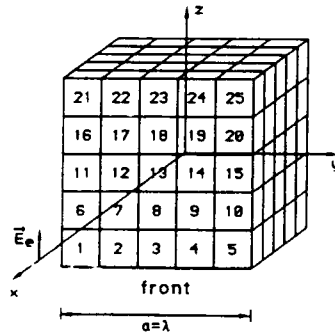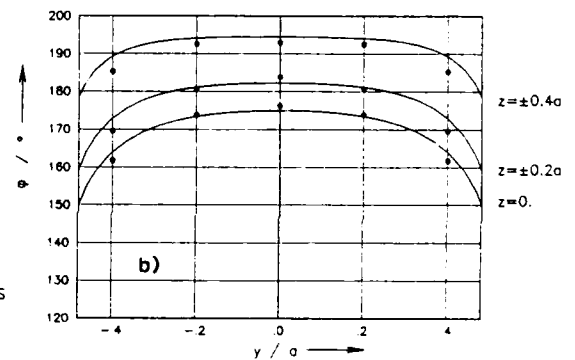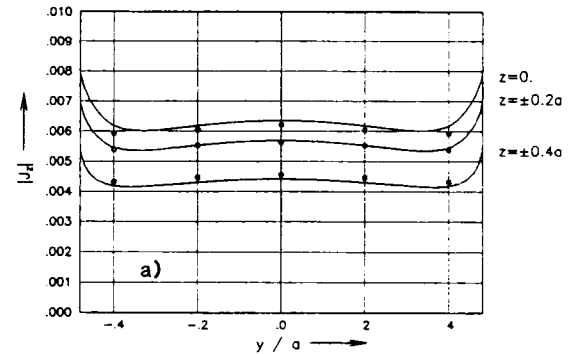
Fig. 3.15 Panel model of a cube.

Fig. 3.16 Experimental and theoretical results for the current distribution on the front of the cube. a) amplitude, b) phase.

4. MODELS USED IN THE PHYSICAL OPTICS APPROACH

4.1 REMARKS TO THE ELECTRODYNAMIC THEORY

The physical optics (PO) method is based on the diffraction theory of Kirchhoff, who described the diffraction phenomena of light (scalar problem) by approximating the boundary conditions at the surface of the scatterer with the aid of optics principles. The idea of Kirchhoff has been extended to the electrodynamic vectorial problem for perfectly conducting bodies [3, 4, 11]. The application and extension of the PO-method for complicated structures including nonperfect conductivity and double reflections is reported in the references [5 - 7, 15].

If the scatterer is perfectly conducting the following boundary conditions according the idea of Kirchhoff are valid for the shadow region of the structure:

(4.1) $\vec{n} \times \vec{E} = 0$,

(4.2) $\vec{n} \times \vec{H} = 0$,

while for the illuminated region the boundary conditions are given by

(4.3) $\vec{n} \times \vec{E} = 0$,

(4.4) $\vec{n} \times \vec{H} = \vec{n} \times \vec{H}_e + \vec{n} \times \vec{H}_r = 2\vec{n} \times \vec{H}_e$,

where $\vec{H}_r$ is the vector of the reflected magnetic field.

The boundary conditions for the electric field are exact, while the boundary conditions for the magnetic field would only be exact if the scatterer would consist of an infinitely extended plane, where $\vec{n} \times \vec{H}_r = \vec{n} \times \vec{H}_e$ in the illuminated region and $\vec{n} \times \vec{H} = 0$ in the shadow region holds. However, it can be assumed that the above given boundary conditions would approximate very well the actual ones, if the scatterer is modeled by panels which are large compared to the wavelength. So a modeling of the structure by flat surface patches (panels) is a natural consequence of the formulation of the boundary conditions.

From Eq. (4.4) follows that the surface current is given by

(4.5) $\vec{J}_F = 2\vec{n} \times \vec{H}_e$

which, therefore, is identical to the first term on the right-hand side of the rigorous integral equation, see Eq. (3.2). The evaluation of the far-field integral, Eq. (3.3), can be done analytically and leads to the following expression for the backscattering matrix of a single panel

(4.6) $[T_1] = - \dfrac{jke^{-jkr}}{2\pi r} \displaystyle\int_{F_p} e^{2jkz'} \, dx' \, dy' \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$

The scattering matrix of the total object then is the sum of the scattering matrices of the individual panels. If a panel has straight edges the phase integral can be solved analytically, so that the calculation of the scattered field requires only a minor computer effort.

If the scatterer is nonperfectly conducting one cannot formulate any exact boundary condition in the PO-sense. In the shadow region the boundary conditions again are approximated by Eqs. (4.1) and (4.2) while in the illuminated region

(4.7) $\vec{n} \times \vec{E} = \vec{n} \times \vec{E}_e + \vec{n} \times \vec{E}_r$,

(4.8) $\vec{n} \times \vec{H} = \vec{n} \times \vec{H}_e + \vec{n} \times \vec{H}_r$

are assumed. $\vec{E}_r$ is the vector of the reflected electric field. Again these boundary conditions would be exact, if the scatterer could be represented by an infinitely extended nonperfectly conducting plane. Since the reflected field can be calculated from the incident field by multiplication with the reflection coefficients of Fresnel the scattered field again can be evaluated analytically from the integral representation for the far-field

(4.9) $\vec{E}_s(\vec{r}) = \dfrac{j\omega\mu}{4\pi} \dfrac{e^{-jkr}}{r} \displaystyle\int_F (\vec{e}_r \times (\vec{e}_r \times \vec{J}_F(\vec{r}')) + \sqrt{\tfrac{\varepsilon}{\mu}} (\vec{e}_r \times \vec{K}_F(\vec{r}'))) \, e^{jk\vec{e}_r \cdot \vec{r}'} \, df'$,

where the magnetic surface current is given by

(4.10) $\vec{K}_F = -\vec{n} \times \vec{E}$.

One receives for the scattering matrix of a single panel

(4.11) $[T_2] = - \dfrac{jke^{-jkr}}{2\pi r \, (1-n_z^2)} \displaystyle\int_{F_p} e^{2jkz'} \, dx' \, dy' \begin{bmatrix} R_H \, n_x^2 - R_E \, n_y^2 & (R_H + R_E) \, n_x \, n_y \\ (R_H + R_E) n_x \, n_y & R_H \, n_y^2 - R_E \, n_x^2 \end{bmatrix}.$

n_x, n_y, n_z are the components of the normal unit vector of the panel. R_H resp. R_E are the reflection coefficients at the surface of a multilayered panel for the case that the incident magnetic field resp. electric field is directed parallel to the surface of the panel, see Sec. 7.1 and [9].

The ansatz of PO implies that the current distribution will be constant in the amplitude over the surface of a panel and varies proportional to the phase of the incident field. Since this result differs from the result of more rigorous solutions it is necessary to estimate the deviations. For this purpose a strip with a width of $a = \lambda$ is considered, the edges of which are directed along the z-axis, which is vertical to the plane of the Fig. 4.1. The vector of the incident electric field, which hits the strip plane vertically is also directed along the z-axis, that is parallel to the edges of the plane. The current distribution over the strip computed with the IE-method [23] is represented by the solid line, whereas the constant amplitude of the PO-current is indicated by the dashed line.
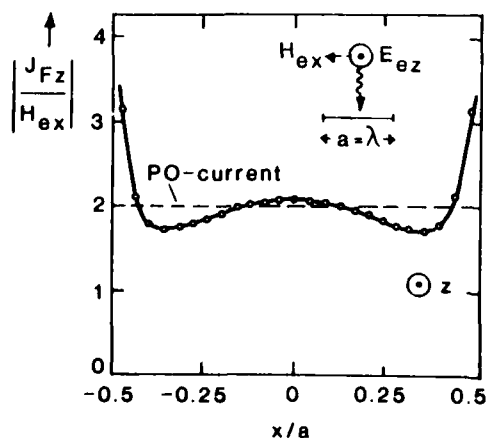

Fig. 4.1
Comparison of the current distributions calculated by the IE- and the PO-method.

One can conclude that for an isolated panel the magnitude of the differences between the IE- and PO-result depends on the distance from the edge. In the middle of the panel the deviations are minimal while in approaching the edges the differences increase considerably. With increasing panel size referred to the wave length, the panel area in which edge current effects play a role are much smaller than the remaining panel area where the PO-current dominates. Therefore, the PO-field will represent the actual field with increasing accuracy if the panel size increases. If, however, a structure whose dimensions are large compared to the wavelength is modeled by a series of connected panels, the size of the panels can be as small as the admissible deviation between true and modeled surface, see Sec. 4.2.

The need to improve the PO-field for small panels gives rise to the development of an edge correction term. The evaluation of an edge correction term for panels with perfectly conducting faces is based on the rigorous solution of an infinitely extended wedge. This solution, however, is very complex so that an asymptotic representation (kr >> 1) of the result is preferred, which leads to analytical expressions for the total scattered field consisting of the PO-field and an edge diffracted field. Since a correction term is required the PO-field in its asymptotic form must be removed from the total scattered field. Following this idea which is the basis of the physical theory of diffraction (PTD), one receives the backscattering matrix for the edge of a perfectly conducting wedge as follows [16, 17]:

$$(4.12) \quad [T_3] = - \frac{e^{-jkr}}{r} L \frac{\sin(kL \cos\beta)}{kL \cos\beta} \frac{e^{j2kr_{Mz}}}{1-t_z^2} \begin{bmatrix} D_e t_x^2 + D_m t_y^2 - D_{em} t_x t_y & (D_e - D_m) t_x t_y + D_{em} t_x^2 \\ (D_e - D_m) t_x t_y - D_{em} t_y^2 & D_m t_x^2 + D_e t_y^2 + D_{em} t_x t_y \end{bmatrix},$$

with L = length of the edge, r_{Mz} = z-coordinate of the edge mid-point, t_x, t_y, t_z = components of the unit tangent vector of the edge, β = angle between the direction of incidence and the edge.

The coefficients D_e, D_m, D_{em} (β, ψ, n) describe the difference between the asymptotic rigorous solution and the asymptotic PO-solution and depend besides β from the outer wedge angle $n\pi$ and the angle ψ between the plane of incidence and a face of the wedge, see Sec. 7.2. It is emphasized that the rigorous asymptotic solution is carried out for an infinite long wedge, while actually the results for a finite edge length are needed. That is corner effects are not taken into account. This can cause the unsymmetry of the above given matrix. However, the effects of an edge with finite length is comparable to the basis idea of PO which uses a reflection coefficient of an infinite plane to estimate the effects of a finite panel. A similar solution for nonperfectly conducting wedges is missing at this time.

The matrixes $[T_1]$, $[T_2]$ and $[T_3]$ represent the most important analytical tool of the PO. The illuminated elements of the surface patch model can be classified as follows: perfectly conducting panels, nonperfectly conducting panels, perfectly conducting doubly reflecting panels, nonperfectly conducting doubly reflecting panels, perfectly conducting edges. The doubly reflecting panels can be treated like the directly reflecting panels using the method discussed in Sec. 4.4. Their polarization characteristics can be described by a further scattering matrix $[T_4]$ which is not represented here, but can be evaluated from the polarization vector given in [7]. An independent summation of all matrixes $[T_1]$, $[T_2]$, $[T_3]$ and $[T_4]$ can be achieved. The sum of the individual summations then represents the scattering matrix of the total object.

The evaluation of the matrix elements and the summation of the matrices can be done in a very economic way. Thus the computer effort for the electrodynamic calculations (about 0.01 s per panel and aspect angle) is much less than that of the IE-method. Another important difference concerns the size of the panels. The IE-method requires panel dimensions in the order of $\lambda/5$ even if a cube is modeled for example. The panels of the PO-method may be as large as the actual surface can be modeled with sufficient accuracy, see the following section. The cube, therefore, can be modeled accurately with only 6 panels. This fact is also suited to reduce the computer time considerably. However, in contrary to the IE-method the geometric problems are not at all solved with the creation of a surface model. According the geometric optics idea a decision must be made whether a panel is illuminated, partially illuminated or hidden. The same is true for all edges. This decision must be made whenever the observer point changes. A similar geometric problem is connected with double and multiple reflections. It can be concluded that the ray-tracing problem inherent to the GTD, becomes also relevant with a PO-method of increasing complexity.

While the principle of PO is known since a long time only a limited experience exists in applying the method for complex structures. This requires an extensive comparison of the theoretical results with experimental results which has be done with good success for a variety of metallic objects [5 - 7, 15]. An estimation of the edge diffraction theory will be available in short. Tests to compare theory and experiment for nonperfectly conducting bodies are in progress [16, 17].

4.2 SIZE OF THE PANELS

In modeling a structure by panels the question arises according to which criterium the size of the panels has to be determined. On the one hand one would like to choose the panel size as large as possible in order to safe computer time. On the other hand the admissible deviation between the true surface and the model surface is subject to the required accuracy of the electromagnetic magnitudes. A series of tests has shown that the deviation between the true surface and the model surface should not exceed a value of about

$$(4.13) \quad \Delta \approx \lambda/16, \quad \text{see Fig. 4.2.}$$

This criterium is well known from antenna measuring technique. If the admissible phase error over the aperture with diameter D of the antenna under test is assumed to be 22.5^0 ($\lambda/16$), then the far-field distance R must be chosen in such a way that $R > 2D^2/\lambda$. The true far-field pattern for the distance $R \to \infty$ then will differ only in a negligible amount from the measured one.

In order to estimate the errors which are generated by the differences between an actual surface and a panel model of it the following test was arranged. The test object consists of a cone, a cylinder and a half-sphere and is manufactured twice: one configuration with smooth surfaces and the other by modeling the smooth surfaces by panels, see Fig. 4.3.
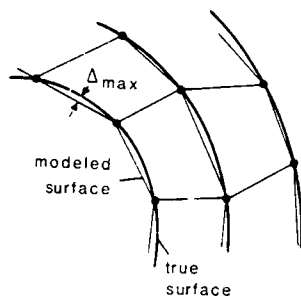
Fig. 4.2 Admissible deviation between the true and the modeled surface: $\Delta_{max} \approx \lambda/16$.

Fig. 4.3 Test object modeled by panels.

The geometric differences between these two objects do not exceed a value of $\lambda/16$. For both objects the radar cross-section has been determined by experiment. The results are shown in Fig. 4.4a for the smooth object and in Fig. 4.4b for the paneled object. At an aspect angle of 90° the object is seen from broadside, at 180° the half-sphere is seen.
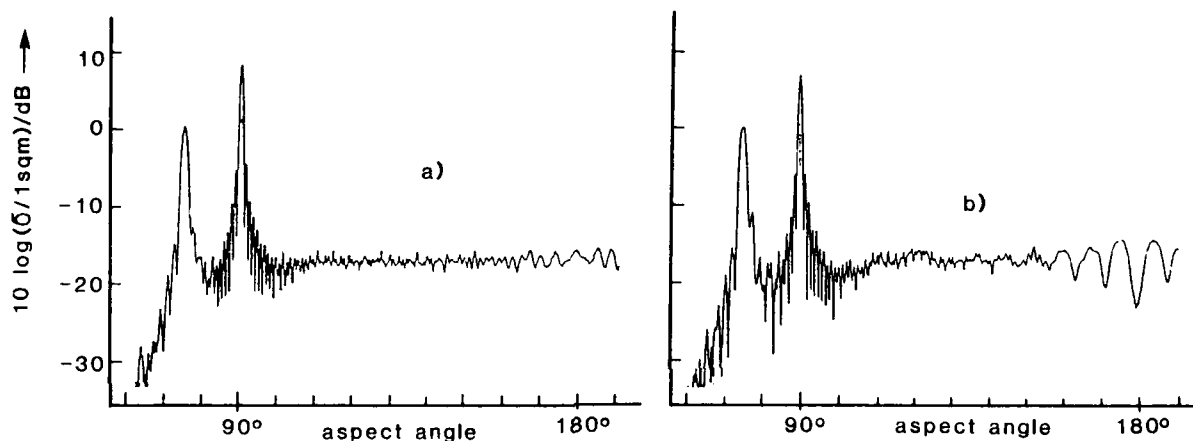

Fig. 4.4 Comparison of the measured radar cross-section for a smooth and a paneled object. a) smooth object, b) paneled object. Length: 2450 mm, diameter: 440 mm, frequency: 15.5 GHz, polarization: hh.

Significant differences only occur at aspect angles near 180° where the half-sphere becomes visible. The radar cross-section level in this region, however, has a level of only -16 dB (≈ 0.025 m^2) compared to the peak level of about 8.5 dB (≈ 7 m^2). Under practical viewpoints this difference can be ignored. If, however, one is interested in higher accuracies the panel model has to be refined.

4.3 HIDDEN SURFACE PROCEDURE

In contrary to the models of the rigorous methods, Sec. 3 and Sec. 5, the surface model of the PO-me-thod is not invariant against the position of the observer point. According the optics basis idea of PO for each point of observation an elimination of the hidden panels must be done. If panels are partially hidden they must be broken into new panels, some of which again are completely hidden. The hidden surface procedure [13] consists of several steps which are explained at hand of the following series of figures. Fig. 4.5 shows the original situation: a rectangular box modeled by rectangular panels with a triangle in front of it. Fig. 4.6 shows the situation after removal of all surfaces with normal vectors including more than 90° with the observer direction.
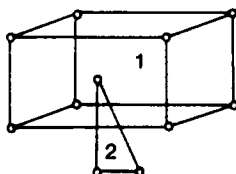

Fig. 4.5
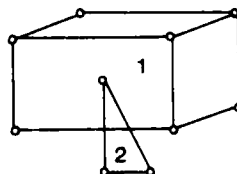Original situation:
box and triangle in
front of it.

Fig. 4.6
Removal of those surfaces,
whose unit normal vector has
no component in the direction
of the point of observation.

The next step consists in a comparison of the remaining panels two by two. For each pair the upper one i.e. the one closest to the observer has to be found. For this purpose the edges of the respective two pa-nels are projected in the xy-plane. The intersection points of the projected edges are determined. For each original edge then the spatial points belonging to the two-dimensional intersection points are calculated. From the differences in the z-values of the spatial points the upper panel can be determined. In principle it is sufficient to compare the differences in the z-coordinate of one intersection point. In Fig. 4.7 the discussed procedure is illustrated at hand of the triangle and the front-face of the box.
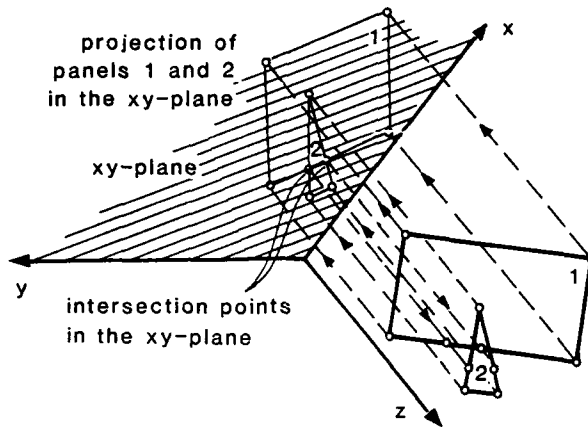
Fig. 4.7
Determination of the upper one of two panels.

If no intersection points can be found the following two situations may occur: one projected panel lies in the interior of the other, see Fig. 4.8, or both panels are wholly apart, see Fig. 4.9.
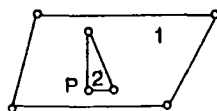

Fig. 4.8
No intersection
point: panel 2 lies
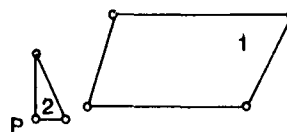in the interior
of panel 1.

Fig. 4.9
No intersection point:
the panels don't hide each
other.

For the decision which situation is in question each corner point P of the second panel is inserted in the equations ($\vec{x} = \vec{x}_{P0} + \lambda \vec{x}_{P2} + \mu \vec{x}_{P4}$) for two identical planes which are generated by the vectors of the adjacent edges of panel 1. This is illustrated by Fig. 4.10. If all of the parameters of the plane equations have positive values the edge point P and therewith the total projected panel 2 lies in the interior of the projected panel 1. Again from the difference of the z-coordinates of the projected corner points one can decide which of the panels is the upper one.
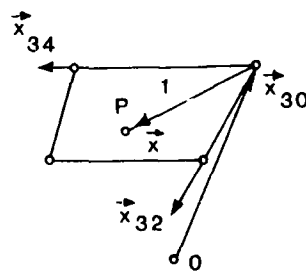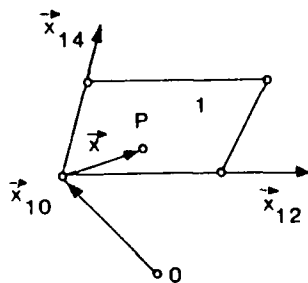
Fig. 4.10
Test if a corner point P of panel 2 is
in the interior of panel 1.
a) plane defined by $\vec{x}_{10}$, $\vec{x}_{12}$, $\vec{x}_{14}$
b) plane defined by $\vec{x}_{30}$, $\vec{x}_{32}$, $\vec{x}_{34}$

The last step consists in breaking up the visible part of the lower panel in new panels which are visible and those which are hidden by the upper panel. This is illustrated by Fig. 4.11.
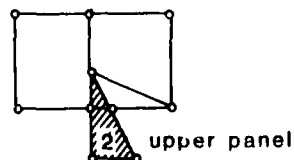
Fig. 4.11
Splitting up of the lower panel into new panels.

Fig. 4.12 shows the original box with the triangle in front of it after the discussed method has been applied to all panels. Fig. 4.13 just shows the remaining visible panels of the box.

Fig. 4.12
Box and triangle
after application
of the hidden sur-
face method.

Fig. 4.13
Remaining panels of the
box.

Since all the panels of a structure must be compared two by two for each angle of observation one

tries to minimize the number of comparisons by a pre-processing of the panel data. One efficient method is based on the construction of a minimum rectangle in the projection plane. This is illustrated at hand of Fig. 4.14.

Fig. 4.14
Construction of minimum rectangles.

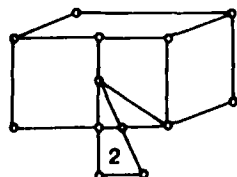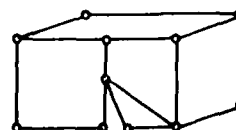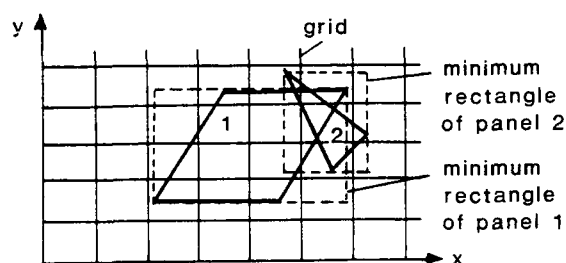A grid is generated which is parallel to the axis of the xy-plane. For each panel the minimum rectangle which also is parallel to the axis is determined and all the elements of the grid which completely or partially are hidden by the minimum rectangles are computed and associated to the individual panels. Only those panels of the structure must be compared which are associated to the same grid element.

4.4 DOUBLY REFLECTING PANELS

Double reflection between two panels A and B occurs, see Fig. 4.15, if

1. both panels are visible from the radar transmitter/receiver,

2. the unit normal vectors $\vec{n}_a$ and $\vec{n}_b$ of the panels are vertical to each other, which is expressed by the formula

(4.14) $\arccos(\vec{n}_a \cdot \vec{n}_b) = \frac{\pi}{2} \pm \epsilon$,

and

3. the line of intersection of the panel planes is vertical to the z axis:

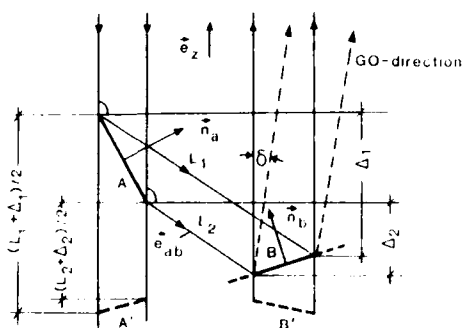(4.15) $\arccos((\vec{n}_a \times \vec{n}_b) \cdot \vec{e}_z) = \frac{\pi}{2} \pm \epsilon$.

Fig. 4.15 Geometry for double reflection.
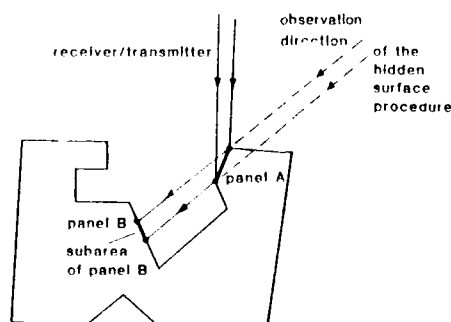
Fig. 4.16 Determination of the illuminated subarea.

For double reflections in the GO sense the angle ϵ is set to zero. ϵ takes into account that the PO field of panel B propagates not only in the GO direction. A search procedure has been developed which specifies all those panels of the structure which meet the above conditions. The appropriate value of the angle ϵ is the subject of current investigations.

Fig. 4.16 shows two doubly reflecting panels A and B where only a subarea of panel B is illuminated by the reflected rays emanating from panel A. The subarea of panel B is constructed by a further application of the previous explained hidden surface procedure, where the observation direction now is coincident with the direction of the reflections from panel A. The part of panel B which is shadowed by panel A is the desired subarea.

The backscattering matrix of each panel is described directly by Eq. (4.11). Additionally, each panel generates a reflected field at the surface of the other panel, which is scattered toward the receiver. This means that two doubly reflecting panels produce two backscattered fields and two field contributions due to double reflections. In Fig. 4.15 only the path from the transmitter via panel A and B back to the receiver is illustrated. The construction of the backscattered field is outlined as follows.

The field incident from the transmitter at panel A has to be decomposed into components parallel and vertical to the plane of incidence and multiplied with the appropriate reflection coefficients. From these components the reflected field of panel A and therewith the incident field at panel B is constructed, taking into account the appropriate path length. The same procedure is repeated at panel B to receive the reflected GO-field. From the incident and the reflected field the total field at the surface of panel B can

be constructed and Eq. (4.9) can be applied to compute the scattered field and therewith the scattering matrix for the direction of the receiver.

For double reflection the phase integral in Eq. (4.9) can be treated in the same manner as in the case of a single panel by introducing the so-called virtual panels. A virtual panel is constructed in such a way that the path length transmitter → doubly reflecting panels → receiver is the same as the path length transmitter → virtual panel → receiver. In Fig. 4.15 the construction of the virtual panel A' is illustrated.

The theory has been tested at hand of a cube with additional surfaces which give rise to shadowing and double reflection effects, see Fig. 4.17. The comparison between theoretical and experimental results for the radar cross-section at a frequency of 16.66 GHz (λ = 18 mm) is presented in Fig. 4.18 for vertical polarization and for aspect angles ranging from -45° to 135°. Within this range the interference of the two doubly reflecting parts of the structure takes place.

Comparing the measured with the computed results, one observes a rather good agreement down to levels of about -30 dB. The peaks at 0° and 90° are due to the GO reflections, the peaks between arise from the interference of the PO fields from the two doubly reflecting areas. The theoretical and experimental determined number of these peaks agree well.
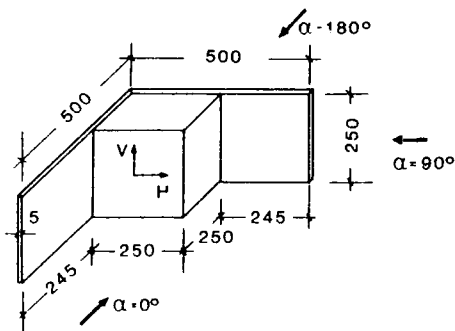

Fig. 4.17
Cube with additional shadowing surfaces, dimensions in mm.
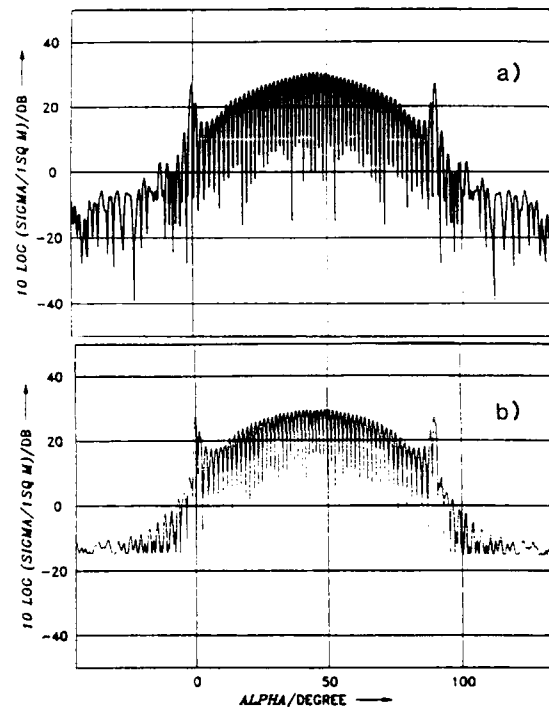
Fig. 4.18
Radar cross-section of a cube with additional shadowing surfaces.
a) theoretical results,
b) experimental results.

5. MODELS USED FOR THE TIME-DEPENDENT SOLUTION OF MAXWELL'S EQUATIONS

In the following two rather new methods are discussed which solve Maxwell's time-dependent curl equations numerically. These methods seem to be useful for studying propagation of an electromagnetic wave into a volume of space containing an arbitrary-shaped dielectric or conducting body. By time-stepping or repeatedly implementing a finite-difference analog of the curl equations at each cell of the corresponding space lattice, the incident wave is tracked as it first propagates to the structure and then interacts with it via penetration and diffraction. Wave tracking is completed when the desired late-time or sinusoidal steady-state behavior is observed at each lattice cell. In contrary to the IE- or PO-method not only the surface of the scatterer but also the surrounding volume must be modeled.

A basic problem with any finite difference solution of Maxwell's equations is the treatment of the field vector components at the lattice truncation. Because of limited computer storage, the lattice must terminate close to the scatterer. Proper truncaction of the lattice requires that any outgoing wave disappears at the lattice boundary without reflections during the continuous time stepping of the algorithm.

In the first reported method [19 ÷ 21] Maxwell's equations

$$(5.1) \qquad \frac{\partial \vec{E}}{\partial t} = \frac{1}{\varepsilon} \nabla \times \vec{H} \, ,$$

$$(5.2) \qquad \frac{\partial \vec{H}}{\partial t} = - \frac{1}{\mu} \nabla \times \vec{E}$$

are solved directly

The scatterer is enclosed in a rectangular volume, see Fig. 5.1. The various details of the structure are modeled with a measuring resolution of one unit cell. Time-stepping is accomplished by an explicit fi-

nite-difference procedure. For a Cartesian cubic-cell space lattice, this procedure involves positioning the components of $\vec{E}$ and $\vec{H}$ about a unit cell of the lattice as shown in Fig. 5.1 and evaluating $\vec{E}$ and $\vec{H}$ at alternate half-time steps [22]. In this manner, centered difference expressions can be used for both the space and time derivatives to attain seccᵒᵈ order accuracy in the space and time increments. This leads to a system of 6 finite-difference equatioₙs.

For example [19], the x-component of Eq. (5.2), written as

$$(5.3) \qquad \frac{\partial H_x}{\partial t} = \frac{1}{\mu} \left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} \right)$$

is implemented as the following time-stepping relation for H_x:

$$(5.4) \qquad H_x^{n+1/2}(i, j+1/2, k+1/2) = H_x^{n-1/2}(i, j+1/2, k+1/2) + \frac{\delta t}{\mu(i, j+1/2, k+1/2)\,\delta} \times$$

$$\times \left(E_y^n(i, j+1/2, k+1) - E_y^n(i, j+1/2, k) + E_z^n(i, j, k+1/2) - E_z^n(i, j+1, k+1/2) \right).$$

The space-time functional notation $F^n(i,j,k) = F(i\delta, j\delta, k\delta, n\delta t)$ is used, where $\delta = \delta x = \delta y = \delta z$ is the space increment, δt is the time increment, and i, j, k and n are integers.
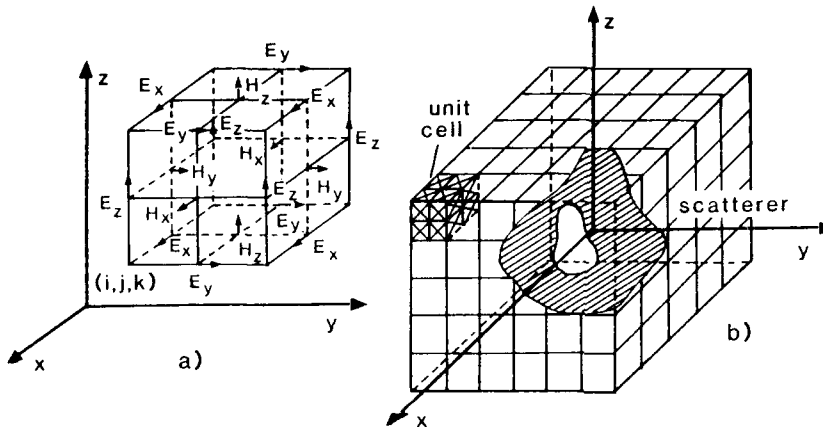

Fig. 5.1
Geometry of a scatterer and lattice arrangement (b), Positions of the field components about a unit cell (a) [19].

With the system of finite-difference equations the new value of a field vector component at any lattice point depends only on its previous value and on the previous values of the components of the other field vector at adjacent points. Therefore, at any given time step the computation of a field vector may proceed one point at a time.

The second time-domain approach [2] to be discussed is based on an integral form of Maxwell's equations. Integration of Eq. (5.1) resp. Eq. (5.2) over a volume V fixed in space with surface F yields

$$(5.5) \qquad \frac{d}{dt} \int_V \vec{E}\, dv = -\frac{1}{\varepsilon} \int_F \vec{H} \cdot d\vec{f},$$

$$(5.6) \qquad \frac{d}{dt} \int_V \vec{H}\, dv = \frac{1}{\mu} \int_F \vec{E} \cdot d\vec{f}.$$

The computational domain around the structure is finite and is descretized by a grid aligned to the structure's surface. The grid consists of curved coordinate surfaces i = const., j = const. and k = const., the volume elements $V(i,j,k)$ are general hexahedra (Fig. 5.2a) with surface vectors $\vec{f}_i(i,j,k)$, $\vec{f}_i(i+1,j,k)$, $\vec{f}_j(i,j,k)$, ... $\vec{f}_k(i,j,k+1)$. According to Eqs. (5.5) and (5.6) the vectors $\vec{H}(i,j,k)$, $\vec{E}(i,j,k)$ of the left-hand side are centered within $V(i,j,k)$ and the vectors $\vec{H}_i(i,j,k)$, $\vec{E}_i(i,j,k)$, ... are centered within related faces (Fig. 5.2b).

The vectors are volume-averaged resp. face-averaged field quantities. Eq. (5.6) for example has the following centered difference expression for the space derivatives:

$$(5.7) \qquad \frac{d}{dt} \vec{H}(i,j,k) = \frac{1/\mu}{V(i,j,k)} \left((\vec{E}_i \times \vec{f}_i)_{i+1,j,k} + (\vec{E}_i \times \vec{f}_i)_{i,j,k} + (\vec{E}_j \times \vec{f}_j)_{i,j+1,k} + (\vec{E}_j \times \vec{f}_j)_{i,j,k} + \right.$$

$$\left. + (\vec{E}_k \times \vec{f}_k)_{i,j,k+1} + (\vec{E}_k \times \vec{f}_k)_{i,j,k} \right).$$

No difference expression for the time derivative is used. The surface vectors $\vec{H}_i$, $\vec{E}_i$, ... are computed from the volume vectors $\vec{H}$, $\vec{E}$ which are known from the preceding time step by linear interpolation in the index space (i,j,k). A similar expression holds for $d\vec{E}(i,j,k)/dt$. Thus Eqs. (5.5) and (5.6) are replaced by a set of ordinary differential equations with respect to time t. These equations are integrated by a Runge/Kutta-type procedure with appropriate step-size control.
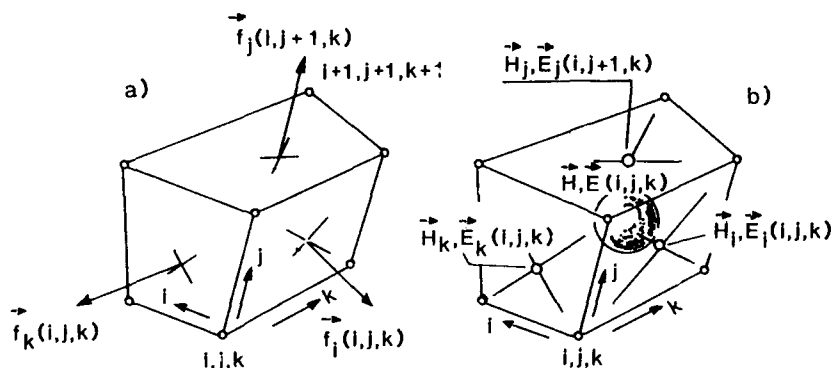
Fig. 5.2
Volume element V(i,j,k) and
location of vectors [2].

Since in both approaches no matrix equations have to be solved, it is possible to split the grid into subdomains. In addition both computer codes are suited for adaption to parallel-processing and vector-array-processing computers. The required memory increases for both methods only linearly with N, the total number of field components to be determined. The value of N is a function of the normalized electric volume of a scatterer, e.g. $N \sim (D/\lambda)^3$ where D is a characteristic dimension of the scatterer. With L, the number of required time steps, the execution time is estimated to be proportional to $L \times (D/\lambda)^3$.

For the IE-method applied to perfectly conducting scatterers the number of unknowns varies with the surface of the scatterer, see Eqs. (3.9) and (3.10), that is $N \sim (D/\lambda)^2$. The required memory for matrix inversion varies with $(D/\lambda)^4$. The estimation of the execution time depends on the chosen method for the solution of linear system of equations. For iterative methods the execution time varies with $(D/\lambda)^4$ which is little more than the requirements of the methods discussed in this Section. For direct solution methods however, the execution time varies with $(D/\lambda)^6$ which is significantly greater as for the time-domain approach. A direct inversion, however, has the advantage that is must be done only once per frequency, since the matrix is independent from the incident field. The procedures discussed here must be repeated like iterative matrix solution methods for each variation of the angle of observation.

Fig. 5.3a shows a hollow square cylinder as test structure with length $L = 9\lambda$, side length of the square $a = 3\lambda$, wall thickness $s = \lambda/10$. This test object has been published in [20]. Fig. 5.3b shows the backscatter cross-section in dependence from the aspect angle. The solid lines represent experimental results, the circles represent theoretical results of the first method discussed in this section and the stars represent results of the second method [2].
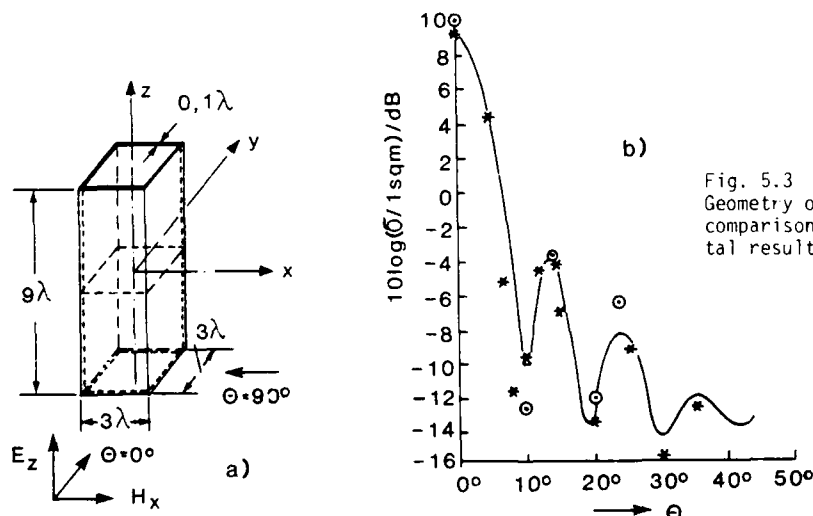
Fig. 5.3
Geometry of hollow cylinder a),
comparison of theoretical and experimental results b).

A lattice cell size of approximately $1/11\lambda$ was selected for the first time-domain approach. Each cylinder wall was formed by 96 x 32 x 1 cells, and the overall lattice size was 112 x 48 x 48 cells. 661 time steps were used, equivalent to 31 cycles of the incident field. For the application of the second approach a grid with 85 x 17 x 73 cells was used, see Fig. 5.4.
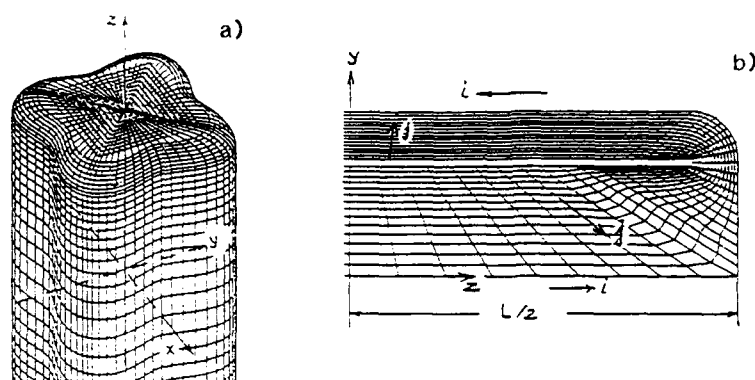
Fig. 5.4
Grid for the hollow cylinder.
a) Outer grid surface,
b) Grid in a plane section
   k = const.

6. CONCLUSION AND REMARKS FOR FURTHER WORK

In this paper the basic ideas of a variety of methods in electromagnetics are outlined in short. The appropriate geometric model of each method is discussed in more detail. The requirements with respect to the type of the model and with respect to the accuracy are established. For some models and methods an estimation of the computer effort is given. For the following problems comparison between theory and experiment has been drawn: radiation characteristic of an antenna installed on a helicopter, current distribution of a cube, radar cross-section of a structure with doubly reflecting surfaces, radar cross-section of a hollow cylinder. Much more test objects are presented in the references cited.

Despite the fact that the number of validation tests increases in the publications there are some essentials gaps in estimating the overall effectiveness of the methods for practical applications. In applying any of the methods for example the following problems are not satisfyingly clarified: incorporation of dielectric media into the model, examination of 3-D structures with empty and loaded cavities and of structures in the resonance region, investigation of near-field properties. Clear statements concerning the stability of results against the type of model (e.g. wire-grid or surface patch model) and the refinement of the model are missing. In the IE-method only a few results are available which are obtained by the independent use of either the EFIE or the MFIE for the same structure. The PO-method has to be extended to treat multiple reflections, creeping waves, bistatic and quasi near-field problems, and must be validated for these cases. For the two time-domain approaches discussed more results concerning cavity problems, internal resonances and nonperfectly conducting bodies should be available since these techniques seem to be advantageous in these special cases. Further studies should also deal with demonstration of accuracy for coarse lattice sizes.

A further gap which should be closed is due to the classification of the methods in those which are suited to treat electrically small structures (rigorous methods) and those which are advantageous in treating electrically large structures (approximate methods). Studies seem to be useful to estimate the structure's dimensions where one type of solution method can be substituted by the other one without significant loss in accuracy for the results. If for example the PO-method could be used instead of the IE-method one could solve the problem much more economically.

Finally the study of the efficiency of the different approaches at the same test objects (an example has been given in the paper) could lead to recommendations of high practical interest. The test objects could be selected with increasing complexity to study effects of edge and corner diffraction, creeping waves, cavity and resonance phenomena, perfectly and nonperfectly conducting bodies.

7. APPENDIX

7.1 REFLECTION COEFFICIENTS

The reflection coefficients at the surface of a panel consisting of N layers, see Fig. 7.1, are given by

$$(7.1) \qquad R_H = \frac{Z^{(1)} \cos_i^{(1)} - Z_{iH}^{(2)} \cos_i^{(2)}}{Z^{(1)} \cos_i^{(1)} + Z_{iH}^{(2)} \cos_i^{(2)}}, \qquad (7.2) \qquad R_E = \frac{Z_{iE}^{(2)} \cos_i^{(1)} - Z^{(1)} \cos_i^{(2)}}{Z_{iE}^{(2)} \cos_i^{(1)} + Z^{(1)} \cos_i^{(2)}}$$

where $Z_{iH}^{(2)}$, $Z_{iE}^{(2)}$ are computed by the following recurrence formulas:

$$(7.3) \qquad Z_{iH}^{(n)} = \frac{Z_{iH}^{(n+1)} \cos_i^{(n+1)} + j Z^{(n)} \cos_i^{(n)} \tan(c^{(n)} d^{(n)})}{Z^{(n)} \cos_i^{(n)} + j Z_{iH}^{(n+1)} \cos_i^{(n+1)} \tan(c^{(n)} d^{(n)})} Z^{(n)},$$

$$(7.4) \qquad Z_{iE}^{(n)} = \frac{Z_{iE}^{(n+1)} \cos_i^{(n)} + j Z^{(n)} \cos_i^{(n+1)} \tan(c^{(n)} d^{(n)})}{Z^{(n)} \cos_i^{(n+1)} + j Z_{iE}^{(n+1)} \cos_i^{(n)} \tan(c^{(n)} d^{(n)})} Z^{(n)},$$

$n = 2,3, \ldots, N-1$, where $Z_{iH}^{(n)}$, $Z_{iE}^{(n)}$ are input impedances of the nth layer for H- or E-polarization, respectively. For $n = N$, Eqs. (7.3) and (7.4) must be replaced by

$$(7.5) \qquad Z_{iH}^{(N)} = Z_{iE}^{(N)} = Z^{(N)}.$$

Fig. 7.1
Geometry of a multilayered panel.

The impedance of the nth layer is given by

$$Z^{(n)} = \sqrt{\frac{\mu_0}{\varepsilon_0}} \sqrt{\frac{\mu_r^{(n)}}{\varepsilon_r^{(n)}}} , \text{ where } \varepsilon_r^{(n)} = \varepsilon_r'^{(n)} - j\varepsilon_r''^{(n)}$$

is the complex relative permittivity (dielectric constant) of the nth layer.

$\mu_r^{(n)} = \mu_r'^{(n)} - j\mu_r''^{(n)}$ is the complex relative permeability, and $c^{(n)} = k^{(n)} \cos\alpha^{(n)}$,

with $k^{(n)} = \omega \sqrt{\varepsilon_0 \varepsilon_r^{(n)} \mu_0 \mu_r^{(n)}}$ being the wavenumber of the nth layer and $d^{(n)}$ the thickness of the nth layer. $\omega = 2\pi f$ = angular frequency.

The value $\cos\alpha^{(n)}$ can be derived from the equation

$$\cos\alpha^{(n)} = \sqrt{1 - \frac{\mu_r^{(1)} \varepsilon_r^{(1)}}{\mu_r^{(n)} \varepsilon_r^{(n)}} \sin^2 \alpha^{(1)}} .$$

The last and the first layer are assumed to be of semi-infinite thickness. For more details see [7, 9].

7.2 PTD EDGE DIFFRACTION COEFFICIENTS

The coefficients of the backscattering matrix, see Eq. (4.12), are given by

(7.6) $D_\nu = D_{\nu 1} - D_{\nu 1}^{po} + D_{\nu 2} - D_{\nu 2}^{po}$, $\nu = e, m, em$

(7.7) $D_{e1} = \frac{\sin\frac{\psi_e}{n}}{n} \frac{1}{\cos\frac{\pi-\alpha_1}{n} - \cos\frac{\psi_e}{n}}$,   (7.8) $D_{e1}^{po} = - U(\pi-\psi_e) \frac{\sin\psi_e}{\mu_1 + \cos\psi_e}$,

(7.9) $D_{m1} = \frac{\sin\frac{\psi_e}{n} \sin\frac{\pi-\alpha_1}{n}}{n \sin\alpha_1} \frac{1}{\cos\frac{\pi-\alpha_1}{n} - \cos\frac{\psi_e}{n}}$,   (7.10) $D_{m1}^{po} = -U(\pi-\psi_e) \frac{\sin\psi_e}{\mu_1 + \cos\psi_e}$,

(7.11) $D_{em1} = \frac{1}{n} \cos\psi_e(\mu_1 + \cos\psi_e) \frac{\sin\frac{\pi-\alpha_1}{n}}{\sin\alpha_1} \frac{1}{\cos\frac{\pi-\alpha_1}{n} - \cos\frac{\psi_e}{n}}$,   (7.12) $D_{em1}^{po} = 0$,

(7.13) $\mu_1 = \cos\psi_e - \frac{2}{\tan^2\psi_e}$,

(7.14) $\alpha_1 = \arccos\mu_1$,

(7.15) $U(x) = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}$ for $\begin{cases} x > 0 , \\ x < 0 . \end{cases}$

Fig. 7.2
Geometry of the bistatic wedge diffraction. In the backscattering case $\psi_s = \psi_e$, $\beta_s = \pi - \beta_e$.

The coefficients D_{v2} are obtained from the coefficients D_{v1} by the following substitutions:

$$\psi_e \to n\pi - \psi_e \ , \quad \beta_e \to \pi - \beta_e \ , \quad \mu_1 \to \mu_2 \ , \quad \alpha_1 \to \alpha_2 \ .$$

Fore more details, see [16, 17].

8. REFERENCES

[1] J.W. Crispin and K.M. Siegel, "Methods of Radar Cross-Section Analysis", New York: Academic Press, 1968, Chapter 9.

[2] Grashof, J., "Finite-Volume Method for the Time-Dependent Maxwell Equations and Prediction of the Scattering from Perfectly Conducting Bodies", 3rd International IGTE Symposium, "Numerical Field Calculation in Electrical Engineering", Graz, Austria, 1988.

[3] H. Hönl, A.W. Maue and K. Westpfahl, "Theorie der Beugung", in "Handbuch der Physik", S. Flügge, Hrsg., Berlin, Göttingen, Heidelberg: Springer-Verlag, 1961, S. 218-219, 236, 238-245, 268-289.

[4] G.F. Koch, "Die verschiedenen Ansätze des Kirchhoffschen Prinzips und ihre Anwendung auf die Beugungs-diagramme bei elektromagnetischen Wellen", A.E.Ü., Band 14, 1960, Heft 2, S. 77-98 und Heft 3, S. 132-153.

[5] D. Klement, J. Preißner und V. Stein, "Computation of the Scattering Matrix of Radar Targets: Concept of the Method and First Results", AGARD Conference Print No. 364, 1984, pp. 20-1 to 20-23.

[6] D. Klement, "Scattering by Complicated Structures: Calculation of the Radar Properties of Metallic Targets by Means of Physical Optics", ESA-TT 946, Translation of DFVLR-FB 85-22, 1985.

[7] D. Klement, J. Preißner and V. Stein, "Special Problems in Applying the Physical Optics Method for Backscatter Computations of Complicated Objects", IEEE Trans. Antennas Propagation, 1988, Vol. 36, No. 2, pp. 228-237.

[8] E.K. Miller and F.J. Deadrick, "Some Computational Aspects of Thin-Wire Modeling" in "Numerical and Asymptotic Techniques in Electromagnetics", R. Mittra, ed., Berlin, Heidelberg, New York: Springer Verlag, 1975, p. 118.

[9] J. Preißner, "Die Reflexionseigenschaften mehrschichtiger Medien und ihre Eignung als Radarabsorber", to be published in the NTZ-Archiv.

[10] J.H. Richmond, "A Wire-Grid Model for Scattering by Conducting Bodies", IEEE Trans. Antennas Propagation, AP-14, 782 (1966).

[11] A. Rubinowicz, "Die Beugungswelle in der Kirchhoffschen Theorie der Beugung", Berlin, Heidelberg, New York: Springer-Verlag, Warszawa: Polnischer Verlag der Wissenschaften, 1966, S. 1-58.

[12] A. Schroth and V. Stein, "Moderne numerische Verfahren zur Lösung von Antennen- und Streuproblemen", München, Wien: R. Oldenbourg Verlag, 1985.

[13] R. Schwemmer, "Ein Algorithmus zur Ermittlung der sichtbaren Teile von Strukturen, die durch Drei-ecke und Vierecke modelliert sind", DFVLR-Mitt. 88-03.

[14] V. Stein, "Numerical Modeling: Integral Equation Method", AGARD Lecture Series No. 152 on "Theore-tical Aspects of Target Classification", 1987, pp. 10-1 to 10-20.

[15] V. Stein, "Physical Optics Method: Prediction of Radar Signatures", AGARD Lecture Series No. 152 on "Theoretical Aspects of Target Classification", 1987, pp. 5-1 to 5-17.

[16] V. Stein, "Beziehungen zwischen bekannten Theorien zur Behandlung der Beugung am Keil im Hochfre-quenzfall", report in preparation.

[17] V. Stein, "Die Rückstreumatrix einer endlich langen Kante", report in preparation.

[18] W.L. Stutzmann and G.A. Thiele, "Antenna Theory and Design", New York, Chichester, Brisbane, Toronto: John Wiley and Sons, 1981, pp. 356-370.

[19] A. Taflove, "Radar Cross Section of General Three-Dimensional Scatterers", IEEE Trans. Electromagne-tic Compatibility, Vol. EMC-25, No. 4, Nov. 1983, pp. 433-440.

[20] A. Taflove, K. Umashankar and T.G. Jurgens, "Validation of FD-TD Modeling of the Radar Cross Section of Three-Dimensional Structures Spanning Up to Nine Wavelength", IEEE Trans. Antennas Propagation, Vol. AP-33, No. 6, June 1985, pp. 662-666.

[21] K. Umashankar and A. Taflove, "A Novel Method to Analyze Electromagnetic Scattering of Complex Ob-jects", IEEE Trans. Electromagnetic Compatibility, Vol. EMC-24, No. 4, Nov. 1982, pp. 397-405.

[22] K.S. Yee, "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media", IEEE Trans. Antennas and Propagation, Vol. AP-14, pp. 302-307, May 1966.

[23] D.R. Wilton and S. Govind, "Incorporation of Edge Conditions in Moment Method Solutions", IEEE Trans. Antennas Propagation, Vol. AP-25, No. 5, Nov. 1977, pp. 845-850.

[24] M. Zink, "Bestimmung charakteristischer Strahlungseigenschaften einer Torusantenne", DFVLR-FB 87-46,

9. ACKNOWLEDGEMENT

SPECIALIST'S MEETING ON "APPLICATIONS OF MESH GENERATION TO
COMPLEX 3-D CONFIGURATIONS
24-25 May 1989 - NORWAY

Mr. W. Schmidt

Before we are closing our specialists' meeting, we all will have the pleasure to listen to our evaluator,
Dr. Steger.

Dr. Steger

Like Preston Henne, I want to start out by thanking the Panel for inviting me here to give the
evaluation. Unlike Preston Henne, I won't take it upon myself to give out gold stars or little soggy
raspberries -- I am not even sure I know what a raspberry is, I guess it is an English expression -- I
don't want to find out either. The way I have approached this is that naturally I was hoping that every
author would send me a copy of their paper two weeks in advance. Of course, most of the papers didn't
show up until a couple of days ago. What I ended up doing finally was to sit down, look at the meeting
announcement and the theme, and make up my own ideas of where I think things stand based on what the theme
of the meeting is. Like any good NASA engineer, I proceeded to make viewgraphs. These viewgraphs reflect
my thoughts as of last week. Then in the last couple days I have also been pencilling in comments based
on what I have heard here (now italicized on the viewgraphs). So I just want to go through things in that
way.

*Viewgraph 1

THEME

...efficient difference schemes ...have evolved for simulating the flow
about relatively simple configurations. However, (for complex configu-
rations) ... it is not clear that routine CFD solution procedures have
evolved, especially for high Reynolds number viscous flow...

...focus on the time- and cost-consuming operations of geometry dis-
cretization and generation of meshes for complex 3-D configurations...

...survey the capabilities of the CFD community for gridding complex
3-D configurations.

...provide insight as to the present state of grid generation... to
assess whether this presents a long term stumbling block to routine use
of CFD...

* * *

If you look at the announcement there were several themes listed. The idea of the announcement was that
quite a bit of progress has been made for simple configurations, but it wasn't at all clear that CFD had
reached the point that you could routinely solve flows about complex configurations especially for high
Reynolds number viscous flow. The organizers of the meeting broke things up based on several ideas. One
is that there should be a focus on the time and cost consuming tasks, of geometry discretization, setting
up the grids, what have you. A lot of the papers here have addressed that particular thought. Another
idea was to survey the capabilities of the community for doing complex configurations. Then I think the
most important is this down here. Assess whether or not we have gotten far enough to decide whether or
not grid generation represents a stumbling block to routine, and I underline the word routine here, use of
CFD. These concepts set the theme.

*Viewgraph 2

GRID CONSTRAINTS (structured or unstructured)

- sufficiently dense for accuracy, not so dense to be impractical
- smooth grid variation enhances accuracy
- body conforming grids simplify BC application, viscous layer resolution
- skewness, discontinuities, and aspect ratio effect algorithm stiffness and
accuracy
- organized data enhances machine efficiency and reduces memory
- grid choice should not lead to overly complex computer codes
- methodology must be compatible with current computers -*useful now*
- gridding readily applicable to complex configurations - *manhour costs
and tradeoffs*
- ease of generation

* * *

To some people grid generation is an extremely easy thing to do. It is just putting a mesh around the
configuration. What makes this difficult and why are there so many different approaches? One thing that
came out of this meeting was that there really aren't so many different approaches. I think what we have
mainly seen are multi-block procedures and unstructured procedures. But there have been one or two odd
papers, such as the last one from Boeing, where some rather unusual approaches have been taken.

There are a lot of constraints on the grid. The grid has to be dense enough that you get accuracy but you cannot afford, on current computers, to make the grid so dense it takes hours and hours of computational time.

There is a tendancy to say that certain finite-volume procedures can give you a good solution on a grid that is changing rapidly. Nevertheless, it is probably true that smooth grid variation enhances accuracy. These are some of the constraints that we have to live with. I think that everybody recognizes that body conforming grids simplify the application of boundary conditions and if you want to simulate viscous flow at a high Reynolds number, a body conforming grid is almost a necessary constraint.

There have been several papers that have addressed the problems of skewness, aspect ratio, discontinuities, how that effects algorithm stiffness and accuracy. That is just another problem that has to be weighed into the overall situation.

This is an argument that I use quite a bit -- I am typically a structured grid person -- the way that organized data enhances machine efficiency and reduces memory requirements. The unstructured people would say, yes, but, if you have a good unstructured grid that adapts well, you can get by with so many fewer grid points that it more than makes up for not having organized data. That is another argument, but we really would, if possible, like organized data.

A grid choice, and I think we would all agree to this, should not lead to a computer code that is unduly complex. Fluid mechanics has not evolved to the state that we can simply turn things on, that we know all the physics and trust the codes. Most of us, when we do viscous flow calculations are in there modelling, trying to use physical intuitions to improve things. As a consequence most codes are constantly in change. You don't want codes to be so complex that you can't still make changes.

If we are going to be useful now, and I think all of us want to be useful to the designer now, the methodology that we employ has to be compatible with current computers. At the same time we don't want to be building something that is going to be obsolete 5 years from now.

Gridding has to be applicable, what these two dots here really say is that we have to make trade offs. We can't expend hundreds of manhours building up a grid to get a solution. We have to reduce that cost. Finally, whatever we do there are certain valid physical approximations we can use. We should be able to incorporate those into our procedures.

Viewgraph 3

AGARDograph No. 309 - Status 1986-1987

Assessment

Composite Grids:

- considerable capability exists, but set up time may measure in weeks (unless similar topology).

- pacing items are subgridding, generating interface boundaries, and surface gridding + +

Unstructured grids:

- high Reynolds number viscous flow simulations not carried out

- 3D inviscid results for complex configurations not extensively validated

* * *

Those are the constraints that we are living with, and that is why this area is hard - and why there are a lot of things going on. Joe Thompson and I about two years ago put together an AGARDograph for which many of you wrote articles. Our contribution was to collect those articles, put a summary article out in front, and make an assessment of where things stood at that time. We collected all these things together in 1986-1987, so we are talking about two years ago. At that time, solutions were coming out of composite grids for fairly complex configurations. The assessment we made was that considerable capability existed, it was mainly of the multi-block type scheme, and that is still true I think. But the set-up time could measure in weeks unless you were doing a topology that was extremely similar to what you had just done before. So as a result of that, the pacing items at that time seemed to be sub-gridding, generating interface boundaries, surface griddings - the geometry-type things. I put two little pluses here because I think those problems have really been addressed by a lot of the authors here and a great deal of improvement has occurred. Our AGARDograph really did not include any unstructured grid codes, probably because of the bias of the people that collected articles for that AGARDograph. But also, because at that time there weren't many three-dimensional solutions coming out on unstructured grids which had been validated, and I don't think there were any (high Reynolds number) viscous flow solutions on an unstructured grid. So we kind of ignored unstructured grids, but in truth, the composite grids weren't much better off at that time.

Viewgraph 4

AGARDograph No. 309 - Status 1986-1987

Recommendations:

- Surface gridding, automation and quality, in considerable need for improvement. More emphasis on development of CAD tools suited to CFD needs.

- Critical need for graphical interaction tools (workstations) for setting up surface grids, zonal boundaries, checking grid results...

* * *

In the AGARDograph we had made some recommendations that a lot more effort should be put on surface gridding, automation of that procedure and more emphasis on quality of grids. The idea was that a lot of this could be done with graphics -- workstations, CAD-CAM devices, and all of this should be automated as much as possible. It was obvious that those kind of things were going to happen because the machinery was becoming available, and again, I think that we have seen a lot of progress in that area at this meeting, although those are areas where we can continue to make a lot of progress.

*Viewgraph 5

PROGRESS

- solutions obtained about complex geometries (fighters, transports, space shuttle, turbomachinary...)
- more effective use of workstations
- progress with composite structured and unstructured grids
- team efforts organized
- grids and solutions for 3D configurations with moving components

* * *

Thinking about some of the progress that has occurred since the AGARDograph, I just made these notes, and I will go through this quickly. Since preparing the AGARDograph in mid-87, thinking about other meetings that I have attended, as well as this one, it is clear that a lot of progress had been made in treating complex geometries. We have seen solutions of fighters, transport aircraft, space shuttle, turbo machinery, what have you. Workstations are really coming into being. Several papers have appeared (unfortunately none of them were invited to this conference) on combining structured grids with unstructured grids. Weatherwill for one, Nakahashi from Japan for another -- but I suppose we are not going to invite Japanese to an AGARD conference. But these look very good and combined structured and unstructured grids is a very promising way to go. Moreover, (perhaps this is from my viewpoint being a researcher at NASA where we usually work pretty much as individual researchers) in this overall area the complexity of the problem has caused a lot of people to come together and work together in teams. That was very impressive. We have also seen some solutions for moving components, store separation and things like that.

*Viewgraph 6

RECOMMENDATIONS FOR NEAR TERM

- more effective use of workstations -*good progress here*
- more use of composite structured and unstructured grids
- use of 'hairy' grids -*new ideas*
- gain more experience with unsteady flows -*and viscous flows*
-*if it doesn't extend to viscous flow, don't do it*
- continue to improve grid generation algorithms -*and adaptive schemes*
-*validation needed, our responsibility, no edicts from above*
-*improve grid smoothness*

* * *

Sitting back in California I made up some recommendations. Since coming here I have tried to change some of these. The first recommendation was to make more effective use of workstations, but as I have pointed out several times, I think that we are doing a good job in trying to do that now.

I really think this is the wave of the future, if you like, the use of composite structured grids and unstructured grids combined. In my way of thinking the structured grids are still the most efficient way of getting solutions when they are applicable. It has been pointed out many many times that when you get really complicated geometries the structured grids break down, which is why we go to a composite structured grid, the multi-block grids, but the unstructured grids seem to be more powerful and they seem to be able to adapt to solution areas better. The smart, safe thing to say is let's combine the two in one algorithm because I don't think that this is very difficult to do. The composite structured grids already have interface-type-logic arrays, pointers, what have you, that are very similar to what you use for unstructured grids. It is not very hard to adapt the flow algorithms to structured-unstructured grids. I think that this is one way to go, to get the efficiency and generality in one code.

Use of 'hairy' grids. This is a term I simply made up before coming, and what I mean is this. I still want to think of viscous flow. What we want to go to is high Reynolds number viscous flow simulations, and I think a good way to resolve a viscous flow is to have a boundary layer approximation near the surface. Grid rays that leave the body normal to the surface allow you to come up with good discretization techniques for picking up the viscous terms in the boundary layers, and also allow you to use implicit operators very easily in that near wall vicinity. I think when you get to fine viscous near-wall grids you will want to use ray-like or hair-like grids. By 'hairy' grids, I also mean that we still need to think of new ideas. If we summarize what is going on here, there are really two main camps, the multi-block camp and the unstructured camp. We saw a couple of other techniques that are a bit different, and I would like to see those developed and continue, but we need new ideas.

Most of the calculations have been for steady flow. I think we need to gain more experience with unsteady flow, although certainly we saw some beautiful unsteady solutions this afternoon. We really have to work on viscous flow more. There has been very little done on viscous flow at this conference. I still take the viewpoint that if the methods you are using won't extend to the Navier-Stokes equations, it is probably not a good idea to use them. I will probably get some flak on that, we will see.

This area used to be the main topic of grid generation — how do we create algorithms to generate grids. At least for the structured grids, there doesn't seem to be much activity here, rather the work seems to be focussed on how do we interface block grids together. For unstructured grids there is still a lot of activity in this area - grid generation and grid adaptiveness.

This is a conference on grid generation, but not enough authors in my opinion spent time trying to show that the grids that they have been devising really are adequate for getting good solutions. We still have this problem of doing validation — and I think that validation is our responsibility. I don't want management or someone from above saying that you can't publish a paper unless you 'validate' the results, because I think that we want the flexibility to work on the problems at hand. At the same time we really have to try to show that the solutions that we are getting right now on these grids are meaningful solutions. So we have that responsibility.

A lot of the grids shown here were somewhat discontinuous, disjoint. It is true that some of the numerical schemes can handle non-smooth grids, but I think we should put more emphasis on (and we actually had a couple of papers talk about) improving grid smoothness, skewness, what have you.

Viewgraph 7

SUMMARY

• Grid generation remains a pacing item, but solutions about complex configurations are being obtained. -*not yet routinely*

• Continued effort needed to make solutions easier and less costly to obtain.

• Like geometry definition, grid generation is a continuing problem that needs resources and creativity.

-*management: this is still the driver, more important, for example, than turbulence modeling!*

* * *

In summary, I would say grid generation remains a pacing item, we are getting solutions, but not routinely. I think that we have had a few people claim that we are getting close to routine, and I won't dispute some of those claims. We still need to work in this area to make the solutions easier and less costly to obtain. We still need a lot of creativity, as this is a problem that is with us for a long time. The last item I have added because I think it may be part of the job of the evaluator to tell some of the management, and a lot of it is here, that this area is still really the driver in computational fluid dynamics. Solution accuracy right now depends more on our ability to resolve a geometry accurately, supplying good grid resolution, and getting good results quickly. These are much more critical than something like turbulence modelling, as far as getting an overall good result. I think the emphasis (in CFD) still has to be in this area. That may be controversial, and may get some discussion going.

Mr. W. Schmidt

We now will start with our Round Table Discussion and we would like to ask many people to either comment directly to Dr. Steger's comments or to raise any of the other questions that have been left open during the last two days, or any new ideas that showed up.

Dr. P. Kutler, NASA Ames

Grid generation is very important. It is one area that I think we know a lot about. We can generate grid insensitive solutions a lot easier than we can probably generate turbulence model insensitive solutions. Therefore my emphasis I think as far as pacing elements go and the discipline of CFD would be that we need more research in the area of turbulence models because there is a lot unknown about turbulence. I do think that although we can't routinely generate grids quickly, we can still very easily refine those grids until we get solutions that are insensitive to the grid, and therefore, if there are errors in the solutions, a good reason why there may be errors is because of the turbulence model.

The second comment regards the amount of time it takes to generate grids. I know that there is some controversy especially from the unstructured people, or at least Tony Jameson from the conference we had at NASA Ames back in March, about the amount of time it takes to set up a grid to do a computational solution. I tend to believe that it is not routine and does take on the order of weeks to do, but I have heard counter views from Tony Jameson. I guess I would like to find out what the real story on this is; can we do grids in a matter of hours on complicated configurations or does it really take a matter of weeks?

T. Baker, Princeton University

I was going to make a few comments and maybe I might make those after answering the question as best I can. If one is using an unstructured method such as the one I presented, how quickly can one generate a mesh around a complete aircraft starting from the geometric definition? I assume that the definition of the aircraft's surface is given as a well-defined set of patches such as one might have for a panel method. Now one can allow the triangulation to go ahead for just the aircraft's surface points and then display that triangulation. In many cases when you do that, you will find that the quality of the mesh on the surface is not as good as you would like, and it is necessary to interpolate extra points, extra sections in various regions in order to get sufficient resolution in critical regions such as near the

wing body junction or around pylons or nacelles. It is an interactive procedure where you sit at the workstation and repeat the triangulation. Typically, starting with a new configuration that might take a couple of days to do. When you are happy with the resolution of the surface mesh, the rest is entirely automatic. The introduction of flow field points proceeds in an entirely automated way and is just a matter of allowing the Cray computer to crunch away with it for 30 minutes or whatever, and the computer will produce a mesh which you can then use to carry out a flowfield solution.

I would like to make a couple of comments on your summing up on the issue of unstructured versus structured, one of the disadvantages of unstructured meshes which I didn't admit to, and which was not brought out by my antagonists is the fact that the flow solvers for unstructured meshes are inevitably less efficient because of the indirect addressing that is involved. It is much harder to vectorize unstructured flowsolvers. Although you can vectorize them to some extent, you don't vectorize as well. Typically, an unstructured flow solver will be two to three times slower than the corresponding stuctured flow solver.

Now I think that there are clearly cases, let us say of a wing alone, where there is absolutely no difficulty in producing a structured mesh. I think that for such an example there would be no point in using an unstructured mesh. There are other situations, a complete aircraft I would dare suggest, where an unstructured mesh would be better. But I agree with you that in the future one will see a composite approach. What we have seen so far, I believe, with composite approaches is the use of a structured mesh around the surface and unstructured meshes outside that. I think that is completely the opposite way to which it should be done. The structured part of the mesh should cover the major part of the flow field and it is only in the interior region close to the aircraft's surface, where it is difficult to mesh, that one should go unstructured. I think that is the kind of composite mesh you need, and I think you will then reap the advantages of structured as well as unstructured meshes.

The final comment I want to make addresses this issue of mesh smoothness which everybody has talked about. Really a great deal has been said about it from a kind of empirical or heuristic point of view. Very little work, as far as I see, has been done to try to address these issues theoretically. In other words, what one would like to see is some kind of theoretical prediction or measure of how important or how bad mesh stretching is, mesh skewness is, point distribution is, and again that will depend on the type of solver we use. It is true that some discretization schemes are more robust. Nevertheless, I think that theoretical estimates which allow you to say yes, this mesh and this flow solver are perfectly adequate, and this mesh and this flow solver are not, are needed in order to take some of the uncertainty out of this business.

Dr. P. Kutler, NASA Ames

There is one other element that you have to include when you are looking at mesh quality and that is the fluid physics that occurs, because that is where you are going to be getting your gradients. I think Peter Eisemann can talk a little bit more about mesh quality. We talk a lot about grid quality, but I don't think that we have derived any criteria on how to judge whether or not we have a good grid other than by looking at it and saying it is good.

Mr. W. Schmidt

I could also make a proposal to look into the quality of meshes. You just build a windtunnel model representing the surface by the mesh with all its patches, you test it and then you see if that is adequate or not.

Mr. R. Bradley, General Dynamics

I would like to say that Joe's comments are right on. I agree with them 100%. I do think that one of the most critical issues that we face as users is in the grid area. Turbulence is important, but I think that there is probably as much uncertainty in grids as there is in turbulence. At the present time I don't have the computer power to keep reducing grid sizes until I get insensitivity. I am not sure I know how far that is either, by the way. My basic feeling and my experience tells me that I really need to understand and know better how to use the grids and how to gain confidence in them. I also agree with Wolfgang that the best way to determine the credibility of your grid is with an experiment and I think the validation issue that was missing in the conference is the key issue for grids as well as for algorithms and turbulence models.

Mr. J. Steger

I would like to agree with Dick Bradley. I don't think that any of our solutions are anywhere grid-independent at this point. I know from my own problems that the geometry is more important than anything else as far as getting the right first order effects for a solution.

Mr. W.J. McCroskey, Ames Research Center

I think that there are several problems that will be more important in the future than perhaps some of the ones we have seen today. I would be interested in the participants' comments on the relative merits of structured and unstructured or maybe a composite of the two on these problems. They are the following: First - capturing low level outgoing waves, like perhaps acoustic waves, when you need details in the flow field away from the body. Second - bodies in relative motion, which includes a store separation that has been described here, turbomachinery with rows in relative motion and the rotor and body of helicopter configurations. Third - concentrated vortices that move through important parts of the flow field that may not be immediately adjacent to the body. Those are problems that haven't seen much attention here, but I am curious about what the future holds vis-a-vis structured, unstructured, or combination thereof.

Clive Albone, RAE, Farnborough

Having listened to what Paul Kutler said about grid insensitivity and some other people have spoken about it, I would like to be able to agree with him. If he were right that there are plenty of examples of grid insensitivity, then clearly turbulence modelling would take over as the main pacing item. However, when I look at the papers that we have seen in the last two days, there are very few examples where people have shown results of convergence under grid refinement by systematically doubling. I showed one example for an extremely simple case where I ran four meshes (and I have good meshes on that simple example), but there was still quite a lot of grid dependency. So if indeed there are plenty examples of grid independence around, I would like to see some of them. Why didn't we see any at this conference?

Mr. A. Roshko, Cal Tech

I don't know anything about grid generation, but I wonder after listening to the discussion whether anybody ever tries to calculate flow over a circular cylinder or a sphere or something like that, where, I presume, the grid generation should be no problem at all.

Mr. J. Steger

It is a hard problem, because you have to go back to the problem that Jim McCroskey brought up. How do you resolve the vortices and everything else that influences the flow field?

Mr. J. Slooff, NLR, Amsterdam

One of the aspects that I feel might become important in the future is that if you look at grid generation in terms of what sort of problem it is, it is actually I think an optimization type of problem because one has to compromise the various requirements in terms of grid complexity and grid quality and that sort of thing. I wouldn't be surprised that in the future we will have some kind of combination of optimization algorithms with grid generation approaches and in that respect variational approaches to the mesh generation problem are also a very interesting option, I think. In my personal opinion we had quite a good example of that in terms of the work of Mr. Jacquotte. I was much impressed by that and it is my feeling that we will see a lot more of that sort of optimization approach in the future. I also have a question for the unstructured people and that is do we really have a measure of the quality of the grids in the sense of what Mr. Jacquotte was using.

Mr. P. Eiseman, Columbia University

I feel that there is a measure of quality for both structured and unstructured, but I think that the comment just made on grid quality was a rather narrow perspective on grid quality. It is just a few of the quality features of a grid. Certainly, for example, orthogonality and smoothness are important. Also you might consider the rate of growth of the grid cells as being important. Furthermore, it depends on how much information you have. For example, you can inject the physics as Paul has mentioned or you can inject the numerical methods you are dealing with. Taking available information into consideration, you can now concentrate on the way in which you view the results. It is not necessarily a total integral type of thing that you have minimized as you would in say a variational type of technique. There are a good number of those as well. Variational techniques are good, but a measure of quality is a broader issue and not quite as narrow as indicated.

Mr. T. Baker, Princeton University

I suppose Dr. Slooff threw out a challenge about the measure of mesh quality in unstructured meshes. I believe that in the paper of Jacquottes which I was very impressed with, he was showing that he can achieve orthogonality of his mesh; (3-D as well as 2-D), and also, to some degree, any mesh aspect ratio he specified. With an unstructured mesh, as I alluded to in my talk, one can certainly base measures of the element quality on such geometric characteristics as minimum edge length, maximum edge length, in-radius, circum radius.

It is possible to use measures based on the ratio of those characteristics to decide what the quality of the particular individual elements is, and if necessary one can restructure the mesh to ensure that the quality of all the elements is good. In that sense I think one does have some control and knowledge about the quality of unstructured meshes. However, that is only dealing with the individual elements. As far as the mesh connectivity is concerned, how many elements are incident on one edge for example, at the moment one doesn't have a great deal of control over such matters, although that will depend to some extent on the point distribution. So I think that with unstructured meshes one does have some degree of control over the quality of the individual elements but further work is needed to ensure overall mesh quality.

Mr. R. Lohner,

I would like to comment on the question that Jim McCroskey had. In particular the two example problems that intrigued me most. These are a) moving bodies (say a store separation) and b) turbine blades. You see a lot of runs which are done with structured grids. What you see particularly in the blade to blade calculations is that when the wake of the first blade enters the grid belonging to the second blade, the wake is simply lost. One can put a nice C mesh on the one blade. However, when you go into the other region, as the blades move one against each other, you lose totally the wake because the grid density on one side does not correspond to the grid density on the other side. Either you take a completely fine grid everywhere, which of course in 3-D is impossible, or you lose it. There are enough papers (e.g., at least at the last Reno meeting there were two) which show that very clearly. I think that all of these time dependent problems will only be tackled with adaptive refinement. As far as I can see the best way to adaptively refine a mesh is with an unstructured mesh. It is just natural. It is so easy. That is basically my view of the future.

Mr. P. Eiseman

That is a very convincing argument. I might add a few comments here. We have seen for example the Chimera scheme which is currently a structured type of approach which does handle mesh movement. But looking a little further into the future at least for block structures and that technology; we don't have it now, but I think that the next field that people will address and that will be done is what I would call an automatic topology generator. If we had such a generator what would then happen is that we would be able to very efficiently remesh, have all the advantages of structured grids, and be able to have motion like for example, store separations where the body comes out and tumbles. So I think that if this were like the horse races, I would say that yes the unstructured and chimera schemes are ahead of the game, but that might not be for very long.

Mr. Van Ingen, Delft

I think that we should come back to the issue of what is the worst problem, turbulence modelling or the mesh generation, because two managers have spoken up and the result is about 50/50; and I think that is very dangerous. I think, what is important is, that you look at the rate of progress in both fields and then you must come to the conclusion, as far as I can see it anyway, that the rate of progress in grid generation is much better than in turbulence modelling. Even if it is 50/50 at this moment in time, it will not be in a few years from now. It is very dangerous to claim that turbulence modelling is not so important at this moment.

Mr. J. Steger

I don't think that I claimed that it wasn't important. I think that what I was trying to claim is that the big payoff is obtained by putting the emphasis in this particular area. I would also argue that it is difficult and maybe even useless to try to do turbulence modelling if your turbulence model is actually trying to compensate for errors in the grid. You have to get those other errors out of there first to know how well your turbulence model is doing.

Mr. W. Schmidt

In fact, you need a very good grid to do turbulence modelling.

Mr. J. Steger

It may be possible that turbulence modelling will never work. So you try to put your money in an area where you know you are going to get some results.

Mr. R. Graves, NASA Headquarters

I am also one of these managers who has to decide between these areas at times. I would like to make a quick comment on two observations. One is the turbulence modelling problem or turbulence in general in my view is the last grand challenge in fluid mechanics and therefore is an intellectually stimulating challenge and one in which a lot of researchers like to work on in terms of understanding the physics, because the physics are still unknown. Whereas grid generation, and I tend to agree with some of the comments which have just been made, is more of an engineering problem, one of once we define what grid quality is, it becomes an optimization problem. It is a problem on which a lot of progress has been made and will continue to be made on. From that standpoint when I look at balancing research, that is research between turbulence, turbulence modelling, and grid generation, I think for the future we have been putting our efforts in turbulence and turbulence modelling because that is where the critical problems still are and I think that the grid optimization problem will be solved in the near future whereas we may not solve the turbulence problem for a number of years.

Mr. W. Schmidt

This might be a good final word on this 50/50 business. From my point of view, just checking through all the papers, I have found actually one area missing somewhat and this is a more indepth look into the flexibility as far as configurational changes is concerned. For instance, if you start designing or developing a new aircraft you start with a wing body and add a nacelle and you add a tail and you add control surfaces and so forth. As it stands now almost everybody as far as I see from the papers will start from scratch everytime he is doing a new calculation rather than using the existing mesh and adding something or just changing something. The same is true if you think of the design problems that we had in the first days. If you really make an attempt to apply a design method on a new wing body configuration, you start redesigning the wing with an inverse method that is a finite difference, a finite volume, or a finite element type, you will definitely end up with new meshes. Does that imply that you do new mesh generations every cycle, every time that you get a modification of the shape of what is going on. I was really missing this aspect in the whole meeting this time. I think that we should really look into this for future improvements.

We now have reached our time limit and we should stop this meeting. On behalf of the Committee of this Specialists Meeting, I would like to thank all authors that gave their very nice presentations, also all those observers that came over here and gave contributions by making comments. I think that we have had two very interesting days and got presentations on all topics we had asked for in our call for papers.

My last Thank You is to Joe Steger and Paul Kutler, both of you being so kind to step in the two gaps we had. You gave us some very interesting information along with our theme. Thank you all very much, and I would like to hand this over now for the final remarks from the Panel Chairman.

Mr. D. Peckham

Thank you Wolfgang. Now it is time to bring our 64th meeting of the Fluid Dynamics Panel to a close. I hope you found the two Specialists' Meetings this week both informative and stimulating and that you will return to your computer terminals ready to implement more efficient mesh generation methods, be prepared to run your programs in an inverse manner, and apply optimization techniques to them. On your behalf I would like to thank the Program Committees for both of these Specialists' Meetings, and in particular the chairmen, Professor Slooff and Dr. Schmidt and the members of the Committee who acted as Session Chairmen, thank you very much for your efforts. Next I wish to thank our Norwegian hosts for everything they have done to ensure the success of our meeting here in Loen. In particular our thanks go to Mrs. Inge Hoff, Major of the Stryn commune who gave the welcoming address on Monday and the slide show, and in particular our two Norwegian panel members who have had so much to do in organizing this meeting, Professor Ytrehus and Professor Norstrud. On Tuesday evening we had a very interesting video presentation on Surface Effect Ships from the Illstein group, and I ask you to join me in thanking Dr. Thurder of the group for this very interesting presentation and very enjoyable reception which followed. Once everything is in place for a meeting, the smooth running depends very much on the staff and our Panel Executive Member, Michael Fischer and his secretary Anne-Marie Rivault, together with the technicians Mr. Colin and Mr. Koolen who have operated the projection and audio equipment so efficiently during the week. I invite you to join me in thanking them all. Last but not least, at the back of the room throughout the week our team of interpreters have had to work very hard and often at very great speed. I ask you to thank them, that is Mrs. Beck-Hertz, Mrs. Lamon and Miss Mazaud.

Thank you very much, Ladies and Gentlemen, that concludes our meeting.

REPORT DOCUMENTATION PAGE

1. Recipient's Reference	2. Originator's Reference	3. Further Reference	4. Security Classification of Document
	AGARD-CP-464	ISBN 92-835-0551-4	UNCLASSIFIED

5. Originator
Advisory Group for Aerospace Research and Development
North Atlantic Treaty Organization
7 rue Ancelle, 92200 Neuilly sur Seine, France

6. Title
APPLICATIONS OF MESH GENERATION TO COMPLEX
3-D CONFIGURATIONS

7. Presented
and discussions held at the Specialists' Meeting of the Fluid Dynamics Panel
in Loen, Norway, 24th—25th May 1989.

8. Author(s)/Editor(s)	9. Date
Various	March 1990

10. Author's/Editor's Address	11. Pages
Various	308

12. Distribution Statement
This document is distributed in accordance with AGARD
policies and regulations, which are outlined on the
Outside Back Covers of all AGARD publications.

13. Keywords/Descriptors

Grids (coordinates)	Three dimensional flow
Aerodynamic configurations	Fluid dynamics
Computation	

14. Abstract

The AGARD Fluid Dynamics Panel sponsored this Symposium to provide a survey of the capabilities of the CFD community for griding complex 3-D configurations. The intent was to provide some insight to the present state of grid generation for complex configurations to help assess whether this task presents a long-term stumbling block to the routine use of CFD in aerodynamic applications.

To this end, the meeting was structured in five sessions: General Surveys, Algebraic Grid Generation, Block Structured Meshes, Multiblock-Adaptive Meshes and Unstructured Meshes. Twenty-two papers from these sessions amply demonstrated that the viability of a numerical solution depends directly on the quality of the mesh and surface representation as measured by its spacing and resolution. Of particular interest was the mesh generation for complex configurations, such as advanced fighter or transport aircraft, missiles and space vehicles, where complex geometries and/or complex flowfields have to be analysed.

Results from this meeting indicate that geometry discretization and generation of meshes for complex 3-D configurations in aerospace will continue to be time- and cost-consuming operations for some time to come.

AGARD-CP-464
Grids (coordinates)
Aerodynamic configurations
Computation
Three dimensional flow
Fluid dynamics

AGARD Conference Proceedings No.464
Advisory Group for Aerospace Research and
Development. NATO
APPLICATIONS OF MESH GENERATION TO
COMPLEX 3-D CONFIGURATIONS
Published March 1990
308 pages

The AGARD Fluid Dynamics Panel sponsored this Symposium to provide a survey of the capabilities of the CFD community for griding complex 3-D configurations. The intent was to provide some insight to the present state of grid generation for complex configurations to help assess whether this task presents a long-term stumbling block to the routine use of CFD in aerodynamic applications.

P.T.O.

AGARD-CP-464
Grids (coordinates)
Aerodynamic configurations
Computation
Three dimensional flow
Fluid dynamics

AGARD Conference Proceedings No.464
Advisory Group for Aerospace Research and
Development. NATO
APPLICATIONS OF MESH GENERATION TO
COMPLEX 3-D CONFIGURATIONS
Published March 1990
308 pages

The AGARD Fluid Dynamics Panel sponsored this Symposium to provide a survey of the capabilities of the CFD community for griding complex 3-D configurations. The intent was to provide some insight to the present state of grid generation for complex configurations to help assess whether this task presents a long-term stumbling block to the routine use of CFD in aerodynamic applications.

P.T.O.

AGARD-CP-464
Grids (coordinates)
Aerodynamic configurations
Computation
Three dimensional flow
Fluid dynamics

AGARD Conference Proceedings No.464
Advisory Group for Aerospace Research and
Development. NATO
APPLICATIONS OF MESH GENERATION TO
COMPLEX 3-D CONFIGURATIONS
Published March 1990
308 pages

The AGARD Fluid Dynamics Panel sponsored this Symposium to provide a survey of the capabilities of the CFD community for griding complex 3-D configurations. The intent was to provide some insight to the present state of grid generation for complex configurations to help assess whether this task presents a long-term stumbling block to the routine use of CFD in aerodynamic applications.

P.T.O.

AGARD-CP-464
Grids (coordinates)
Aerodynamic configurations
Computation
Three dimensional flow
Fluid dynamics

AGARD Conference Proceedings No.464
Advisory Group for Aerospace Research and
Development. NATO
APPLICATIONS OF MESH GENERATION TO
COMPLEX 3-D CONFIGURATIONS
Published March 1990
308 pages

The AGARD Fluid Dynamics Panel sponsored this Symposium to provide a survey of the capabilities of the CFD community for griding complex 3-D configurations. The intent was to provide some insight to the present state of grid generation for complex configurations to help assess whether this task presents a long-term stumbling block to the routine use of CFD in aerodynamic applications.

P.T.O.

To this end, the meeting was structured in five sessions: General Surveys, Algebraic Grid Generation, Block Structured Meshes, Multiblock-Adaptive Meshes and Unstructured Meshes. Twenty-two papers from these sessions amply demonstrated that the viability of a numerical solution depends directly on the quality of the mesh and surface representation as measured by its spacing and resolution. Of particular interest was the mesh generation for complex configurations, such as advanced fighter or transport aircraft, missiles and space vehicles, where complex geometries and/or complex flowfields have to be analysed.

Results from this meeting indicate that geometry discretization and generation of meshes for complex 3-D configurations in aerospace will continue to be time- and cost-consuming operations for some time to come.