AIAA Paper 95-1657, in Proceedings 12th AIAA CFD Conference, June 19-22, San Diego USA, 1995

# ANISOTROPIC GRID REFINEMENT USING AN UNSTRUCTURED DISCONTINUOUS GALERKIN METHOD FOR THE THREE-DIMENSIONAL EULER EQUATIONS OF GAS DYNAMICS

J.J.W. van der Vegt*
National Aerospace Laboratory NLR
P.O. Box 90502, 1006BM Amsterdam, The Netherlands

## Abstract

A numerical method for the solution of the Euler equations of gas dynamics using anisotropic grid refinement of an unstructured, hexahedron type grid is presented. The discontinuous Galerkin finite element method is used to solve the Euler equations in three dimensions. Special attention is paid to the data structure and searching algorithms necessary for efficient calculation on highly irregular grids obtained with local grid refinement. The method is demonstrated with calculations of transonic flow on the ONERA M6 wing.

## Introduction

Accurate solutions of complicated three-dimensional flows frequently can only be obtained with reasonable efficiency using grid adaptation. Several types of grid adaptation are possible, the most important methods for compressible flow are local grid refinement ($h$-refinement) and methods which generate a completely new grid based on information about the flow solution ($r$-refinement). One of the main benefits of local grid refinement is that one does not have global constraints on the grid generation. In this paper a new grid adaptation method for the three-dimensional Euler equations of gas dynamics will be discussed. This work is part of a larger project aimed at obtaining time accurate solutions of viscous flows using Large Eddy Simulations. A basic element of the adaptation method is therefore that it can be easily extended to viscous flows.

The numerical method is a combination of local grid refinement of hexahedral cells with the Discontinuous Galerkin (DG) finite element method. The grid adaptation is done independently in all three directions to allow for maximum flexibility. Many important flow phenomena, such as shocks and shear layers, are locally pseudo two-dimensional and anisotropic grid refinement is more efficient in these cases than isotropic refinement.

Until now most of the unstructured algorithms for the Euler and Navier-Stokes equations use tetrahedral elements, for a review [9]. The use of hexahedral unstructured grids is a more recent development, e.g. [1]. Hexahedrons suffer less from loss of accuracy due to anisotropic refinement than tetrahedrons. Hexahedral cells are more accurate on highly stretched grids which are necessary for future applications to viscous flows. In order to deal with complicated geometries, elements such as prisms and tetrahedrons are used to deal efficiently with topological degeneracies. An additional benefit of hexahedrons is the fact that the initial coarse grid can be provided by standard multi-block grid generators which are widely available.

The data structure for $h$-refinement is considerably more complicated than for $r$-refinement. It was found that it is more efficient to replace the commonly used cell based octree data structure with a cell face based data structure. Especially when one does not want to impose restrictions on the number of neighboring cells. The description of this data structure is given special attention in this paper.

The use of highly irregular unstructured grids puts severe demands on the accuracy of the flow solver. One of the more difficult problems on this type of grids is the calculation of the flow gradient, required to make the scheme second order accurate and also for future calculations of viscous flows. The most popular method is to use of Gauss' identity to estimate the gradient, but this method requires a certain grid regularity to be accurate.

An alternative is presented by the Discontinuous Galerkin (DG) finite element method, as proposed by Cockburn and Shu, [5, 8, 7]. They proved that this method satisfies a maximum principle for multi-dimensional scalar hyperbolic conservation laws and is TVB stable. The DG method is a mixture of a finite volume and finite element method. In each cell the flow field is locally expanded in a polyno-

*Member AIAA, 

mial and equations for the polynomial coefficients are obtained.

The DG method therefore not only solves equations for the flow field, but also for the moments of the flow field. No information from neighboring cells is required to calculate the flow gradient and a completely local scheme is obtained which does not lose accuracy on highly irregular grids, such as those obtained with anisotropic grid refinement. Due to the local series expansion of the flow field the DG method does not have any difficulty with hanging nodes, which occur with local grid refinement. One does not have to invert a large mass matrix as is required with node based finite element methods.

The local nature of the discretization makes it easy to mix different types of elements, eg. hexahedrons, prisms and tetrahedrons. The discontinuous Galerkin method, together with Runge-Kutta time integration, is an excellent candidate for parallel computing due to it's local behavior. A disadvantage of the DG method is that it requires more memory because it is necessary to store several moments of the flow field. This does not have to be a limitation because grid adaptation will generally reduce the memory requirements significantly.

The DG method has until now primarily been used in two-dimensions. Cockburn and Shu applied the method on triangle based grids, [6], while Lin and Chin [12] and Bey and Oden [3] used quadrilateral cells. The extension of the DG method to three-dimensional flow will be discussed in this paper.

The outline of the paper is as follows. First, the discontinuous Galerkin method will be discussed for the three-dimensional Euler equations of gas dynamics. Next, the grid adaptation procedure will be discussed and an overview of the data structure and searching algorithms necessary for anisotropic grid refinement with hexahedral type cells will be given. Finally, the grid adaptation algorithm will be demonstrated with calculations of transonic flow on the ONERA M6 wing.

## Governing Equations

The Euler equations for inviscid gas dynamics in conservation form can be expressed as:

$$\frac{\partial}{\partial t}\mathbf{U}(\mathbf{x},t) + \frac{\partial}{\partial x_j}\mathbf{F}^j(\mathbf{U}) = 0, \qquad (1)$$

$(\mathbf{x},t) \in \Omega \times (0,T)$, with initial condition $\mathbf{U}(\mathbf{x},0) = \mathbf{U}_0(\mathbf{x})$, $\mathbf{x} \in \Omega$ and boundary condition $\mathbf{U}(\mathbf{x},t) = \mathcal{B}(\mathbf{U},\mathbf{U}_\infty)$, $(\mathbf{x},t) \in \partial\Omega \times (0,T)$; where B denotes the boundary operator and $\mathbf{U}_\infty$ the prescribed boundary data. Here $\Omega \in R^3$ is an open domain with boundary $\partial\Omega \subset \Omega$ and $t \in (0,T)$ represents time. The summation convention is used on repeated indices. The vectors with conserved flow variables $\mathbf{U} : \bar{\Omega} \rightarrow R^5$ and fluxes $\mathbf{F}^j$, $j = \{1,2,3\}$; $\mathbf{F}^j : R^5 \rightarrow R^5$, are defined as:

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u_i \\ \rho E \end{pmatrix}, \quad \mathbf{F}^j = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p\delta_{ij} \\ u_j(\rho E + p) \end{pmatrix},$$

where $\rho$, $p$ and $E$ denote the density, pressure and specific total energy and $u$, the velocity in the Cartesian coordinate directions $x_i$, $i = \{1,2,3\}$ and $\delta_{ij}$ the Kronecker delta symbol. This set of equations is completed with the equation of state: $p = (\gamma-1)\rho(E - \frac{1}{2}u_i u_i)$, with $\gamma$ the ratio of specific heats.

## Discontinuous Galerkin Approximation

The discontinuous Galerkin approximation of the Euler equations is defined by the following steps:

Suppose the open domain $\Omega$ is a polyhedron and denote by $\mathcal{T}_h$ a triangulation of $\Omega$ into a disjunct set of polyhedrons $K_j$, $j \in N^+$, such that $\cup K_j = \Omega$. Each polyhedron $K$ has $n$ faces $e_K^i$, $i \in N^+$ with $\cup_i e_K^i = \partial K \subset K$. Each face $e_K^i$ can connect to multiple faces $e_{K'}^j$. The faces $e_K^i$ are split into subfaces $s_{KK'}^{ij} = e_K^i \cap e_{K'}^j$. The faces $s_{KK'}^{ij}$ therefore always connect to two neighboring cells in $\Omega$. This greatly facilitates the update of the cell face fluxes. The boundary faces $e_K^i \subset \partial\Omega$ are denoted $b_K^i$. As basic elements hexahedrons ($n = 6$) are used, but in order to deal with topologically degenerated cases, hexahedrons with degenerated edges, such as prisms and tetrahedrons, are also used when necessary.

Each of the elements $K \in \mathcal{T}_h$ is subparametrically mapped into the cubic master element $\hat{K} = [-1,1] \times [-1,1] \times [-1,1]$, with local coordinates $-1 \leq \xi,\eta,\zeta \leq 1$, using the standard linear finite element shape functions:

$$F_K : \mathbf{x}(\xi,\eta,\zeta) = \sum_{i=1}^{8} \mathbf{x}_K^i \psi_i(\xi,\eta,\zeta),$$

with $\psi_i(\xi,\eta,\zeta)$ trilinear element shape functions and $\mathbf{x}_K^i$ the coordinates of the corner points of the hexahedron $K$.

Define $P^k(\hat{K})$ as the space of polynomial functions of degree $\leq k$ on the master element $\hat{K}$: $P^k(\hat{K}) = span\{\hat{\phi}_j(\xi,\eta,\zeta), j = 0,\cdots,M\}$. Define $P^k(K)$ as the space of functions whose images under $F_K$ are

functions in $P^k(\hat{K})$: $P^k(K) = span\{\phi_j(\mathbf{x}) = \hat{\phi}_j \circ F_K^{-1}, j = 0, \cdots, M\}$.

Define $\mathbf{V}_h^k(K) = \{\mathbf{P} \in R^5 : \text{with each component } p_i \in P^k(K)\}$, then $\mathbf{U}(\mathbf{x},t)\mid_K$ can be approximated by $\mathbf{U}_h(\mathbf{x},t) \in \mathbf{V}_h^k(K) \times C^1[0,T]$ as:

$$\mathbf{U}_h(\mathbf{x},t) = \sum_{m=0}^{M} \hat{\mathbf{U}}_m(t)\phi_m(\mathbf{x}) \tag{2}$$

A major difference with standard node based Galerkin finite element methods is that the expansion of $\mathbf{U}$ is local in each element, without any continuity across element boundaries. This has as important benefit that hanging nodes, which frequently appear after $h$-refinement, do not give any complications.

A weak formulation of the Euler equations is obtained by multiplying equation (1) with $\mathbf{W}_h \in \mathbf{V}_h^k(K)$, integrating over the element $K$, and replacing the exact solution $\mathbf{U}$ with its approximation $\mathbf{U}_h \in \mathbf{V}_h^k(K) \times C^1[0,T]$:

Find $\mathbf{U}_h \in \mathbf{V}_h^k(K) \times C^1[0,T]$, such that $\mathbf{U}_h(\mathbf{x},0) = \mathbf{U}_0(\mathbf{x})\mid_K \in \mathbf{V}_h^k(K)$, and for $\forall \mathbf{W}_h \in \mathbf{V}_h^k(K)$:

$$\begin{aligned}&\frac{\partial}{\partial t}\int_K \mathbf{W}_h^T(\mathbf{x})\mathbf{U}_h(\mathbf{x},t)d\Omega = -\sum_i \int_{e_K^i} \mathbf{W}_h^T(\mathbf{x})\mathbf{n}^T(\mathbf{x})\\ &\mathcal{F}(\mathbf{U}_h)dS - \sum_i \int_{b_K^i} \mathbf{W}_h^T(\mathbf{x})\mathbf{n}^T(\mathbf{x})\mathcal{F}(\mathbf{B}(\mathbf{U}_h,\mathbf{U}_w))dS\\ &+\int_K \nabla\mathbf{W}_h^T(\mathbf{x})\mathcal{F}(\mathbf{U}_h)d\Omega,\end{aligned} \tag{3}$$

with $\mathcal{F} = \mathbf{F}^j$, $j = \{1,2,3\}$ and $\mathbf{n}$ the unit outward normal vector at the faces $e_K^i$ and $b_K^i$. In this paper $k$ is restricted to one.

Due to the fact that the polynomial basis functions $P^k(\hat{K})$ are discontinuous across element boundaries it is necessary to replace the flux with a monotone flux, $\mathbf{h}(\mathbf{U}_h^{int(K)}, \mathbf{U}_h^{ext(K)})$, which is consistent, $\mathbf{h}(\mathbf{U},\mathbf{U}) = \mathbf{n}^T\mathcal{F}(\mathbf{U}) \equiv \hat{\mathbf{F}}(\mathbf{U})$, [7]. Here $\mathbf{U}^{int(K)}$ and $\mathbf{U}^{ext(K)}$ denote the value of $\mathbf{U}$ at $\partial K$ taken as the limit from the interior and exterior of $K$. The use of the monotone Lipschitz flux $\mathbf{h}$ introduces upwinding into the Galerkin method by solving the (approximate) Riemann problem given by $(\mathbf{U}_h^{int(K)}, \mathbf{U}_h^{ext(K)})$. Suitable fluxes are those from Godunov, Roe, Lax-Friedrichs and Osher. In this paper the Osher approximate Riemann solver [14] is used, because of it's good shock capturing capabilities, and the possibility to easily modify the Riemann problem to account for boundary conditions. An important additional reason for the use of the Osher scheme is that it gives an exact solution for a steady contact discontinuity, and therefore it has a very low numerical dissipation in boundary layers, [18], which is important for future extension of the algorithm to the Navier-Stokes equations. The Osher approximate Riemann solver is defined as:

$$\begin{aligned}\mathbf{h}_{s_{KK'}^{ij}}(\mathbf{U}_h^{int(K)}, \mathbf{U}_h^{ext(K)}) = \; &\tfrac{1}{2}(\hat{\mathbf{F}}_{e_K^i} + \hat{\mathbf{F}}_{e_{K'}^j} -\\ &\textstyle\sum_\alpha \int_{\Gamma_\alpha} |\partial\hat{\mathbf{F}}|d\Gamma),\end{aligned}$$

where $\cup_\alpha\Gamma_\alpha$ is a path in phase space between $\mathbf{U}_h^{int(K)}$ and $\mathbf{U}_h^{ext(K)}$. Details of the calculation of this path integral in multi-dimensions can be found in [14]. At the boundary surface the path $\Gamma_\alpha$ must be modified to account for boundary conditions. In this way a Riemann initial-boundary value problem is solved instead of an initial value problem, [14], and a completely unified and consistent treatment of the flux calculations is obtained, both at interior and exterior faces.

A key element of any numerical method for hyperbolic conservation laws is the prevention of oscillations around discontinuities. Cockburn et al. [7] presented a discontinuous Galerkin local projection method for multi-dimensional scalar conservation laws, which is second order accurate and satisfies a maximum principle when combined with a TVD Runge-Kutta time integration method [16]. Cockburn et al. [7] used triangular elements and the extension to quadrilaterals is presented in [3]. The extension to the Euler equations is usually done with a local characteristic decomposition, but in multiple dimensions this decomposition is only approximate and it can not be proven that the limiter satisfies a maximum principle. Therefore a slightly different approach is followed and the multi-dimensional limiter proposed by Barth and Jesperson [2] is used directly on the conservative variables. This limiter saves the considerable expense of computing the local characteristic decomposition.

Define for each component $\bar{U}_K^i$ of the cell average $\mathbf{U}_K = \frac{1}{meas(K)}\int_K \mathbf{U}_h(\mathbf{x})d\Omega$:

$$\begin{aligned}U_{K\,min}^i &= \min_{\forall K' \in N(K)}(\bar{U}_K^i, \bar{U}_{K'}^i)\\ U_{K\,max}^i &= \max_{\forall K' \in N(K)}(\bar{U}_K^i, \bar{U}_{K'}^i),\end{aligned}$$

with $N(K)$ the set of neighboring cells which satisfy $s_{KK'}^{ij} \subset e_{K'}^j$. In order to maintain monotonicity the approximate flow field $\mathbf{U}_h$ must satisfy $\mathbf{U}_h(\mathbf{x}) \in [U_K^{min}, U_K^{max}]$, $\forall \mathbf{x} \in K$, which is accom-

plished with the limiter function $\Phi_K$ defined as:

$$\Phi_K^i = \begin{cases} \min\left(1, \frac{U_{K,\max}^i - \bar{U}_K^i}{U_{K^*}^i - \bar{U}_K^i}\right) & \text{if} \quad U_{K^*}^i - \bar{U}_K^i > 0 \\ \min\left(1, \frac{U_{K,\min}^i - \bar{U}_K^i}{U_{K^*}^i - \bar{U}_K^i}\right) & \text{if} \quad U_{K^*}^i - \bar{U}_K^i < 0 \\ 1 & \text{if} \quad U_{K^*}^i - \bar{U}_K^i = 0 \end{cases}$$

Here $U_{K^*}^i$ are the components of $\mathbf{U}_h$ at the Gauss quadrature points in $K$, used to evaluate the integrals in equation (3). The limiter $\Phi_K$ is applied independently to each component of the flow field: $\tilde{U}_m^i = \Phi_K^i \tilde{U}_m^i$, $m = \{1, 2, 3\}$. This is slightly less robust then using $\Phi_K = \min_i \Phi_K^i$, but gives significantly less numerical dissipation. The coefficients $\tilde{U}_m, m = \{1, 2, 3\}$ in equation (2) represent the gradient of the flow field with respect to the local coordinates in $\hat{K}$. This modification of the local gradient would violate conservation of $\mathbf{U}$ in $K$, which can be corrected by modifying the coefficient $\tilde{U}_0$:

$$\tilde{U}_0^i = \bar{U}_0^i + \frac{1 - \Phi_K^i}{meas(K)} \sum_{m=1}^{3} \tilde{U}_j^i \int_K \phi_m(\mathbf{x}) d\Omega$$

This relation is obtained from the condition $\frac{1}{meas(K)} \int_K \tilde{\mathbf{U}}_h(\mathbf{x}) d\Omega = \bar{\mathbf{U}}_K$. The limited flow field in cell $K$ is then equal to:

$$\tilde{\mathbf{U}}_h(\mathbf{x}, t) = \sum_{m=0}^{3} \tilde{\mathbf{U}}_m(t) \phi_m(\mathbf{x}).$$

Unfortunately, the direct application of this limiter can seriously hamper convergence to steady state. In order to alleviate this problem the modifications proposed by Venkatakrishnan, which result in a smoother limiter, are used with reasonable success. For more details see [19].

The equations for $\tilde{\mathbf{U}}_m(t)$ are now obtained after transformation of the integrals over $K$ and $\partial K$ in equation (3) into integrals over the master element $\hat{K}$. The first integrals $\int \mathbf{W}_h^T \mathbf{U}_h d\Omega$, are calculated analytically, which requires quite some algebra, whereas the other integrals are calculated with Gauss quadrature rules. Cockburn et al. [7] proved that if the quadrature rules for the surface integrals in equation (3) are exact for polynomials of degree $(2k + 1)$ and exact for polynomials of degree $2k$ for the volume integrals then the spatial accuracy of the DG method is $k + 1$. In order to preserve uniform flow it is necessary to use quadrature rules which are exact for polynomials of order 3. For $k = 1$ the surface integrals are calculated with four point Gauss quadrature rules. They require, however, special attention to prevent that they are unnecessarily expensive. A direct application of a four point Gauss quadrature rule would require four calculations of the Osher integral $\sum_\alpha \int_{\Gamma_\alpha} |\partial \tilde{\mathbf{F}}| d\Gamma$, which is the most expensive part of the flux calculation. This number can be reduced to one integration using the fact that $\mathbf{h}$ is a Lipschitz flux and $|\mathbf{U}_h^{ini(K)} - \mathbf{U}_h^{ext(K)}| = O(h^{k+1})$ in the smooth part of the flow. The following approximation can then be obtained:

$$\int_{e_K^i} \mathbf{W}_h^T \left( \int_{\Gamma_\alpha} |\partial \tilde{\mathbf{F}}| d\Gamma \right) dS \cong \int_{\Gamma_{\alpha_0}} |\partial \tilde{\mathbf{F}}| d\Gamma \int_{e_K^i} \mathbf{W}_h^T dS$$

where the path $\Gamma_{\alpha_0}$ is the path in phase space between the left and right state at the cell face center. In shocks $|\mathbf{U}^{ini(K)} - \mathbf{U}^{ext(K)}|$ is not small, but in these regions the scheme is first order accurate anyway. With this modification the integration of the fluxes becomes approximately equally expensive as for upwind finite volume schemes using an (approximate) Riemann solver.

For each element $K$ a system of ordinary differential equations is now obtained:

$$[M_K] \frac{\partial}{\partial t} \tilde{\mathbf{U}}_K = \mathbf{R}_K$$

with $\tilde{\mathbf{U}}_K$ a vector with the moments of the flow field in each element, $\tilde{\mathbf{U}}_m, m = \{0, \cdots, 3\}$ and $\mathbf{R}_K$ the right-hand side of equation (3). The equations for $\frac{\partial}{\partial t} \mathbf{U}_K$ are integrated in time using the TVD Runge-Kutta scheme from Shu [16]. For steady state calculations convergence is accelerated using local time stepping.

A significant difference with node based FEM is that the mass matrix $[M_K]$ is uncoupled for each element $K$ and can be easily inverted.

## Directional Grid Adaptation

The grid adaptation procedure is based on subdividing cells independently in each of their three local grid directions, $\xi$, $\eta$ or $\zeta$. A coarse initial grid is used, which is generated with a multi-block structured grid generator. This initial structured multi-block grid is transferred into an unstructured hexahedron grid, and degenerated hexahedrons, such as prisms and tetrahedrons, are used when topological degeneracies make this necessary. This grid is called root grid. The root grid can also be generated directly, without first using a block-structured grid, but this is not part of the present paper. After calculating the flow field, the grid cells are split in the local $\xi$-direction if:

$$\frac{R_K^\xi}{\max_{\forall K \in \Gamma_h} R_K^\xi} > \text{tolerance} \qquad (4)$$

with the sensor function $R_K^\xi$ for the cell $K$ defined as:

$$R_K^\xi = \max_{i\in\{1,\cdots,5\},\forall K'\in N^\xi(K)} (V_K^i - V_{K'}^i)^2 \triangle\xi_K^2 \quad (5)$$

Here $\triangle\xi_K$ is the length of the cell $K$ in the local $\xi$-direction. $\mathbf{V} = (\rho, u, v, w, p)^T$ the vector with primitive variables and $N^\xi(K)$ the indices of the neighboring cells of cell $K$ in the $\xi$-direction. Equivalent expressions are used for the $\eta$ and $\zeta$ directions. This sensor is based on the equidistribution principle, see for instance Marchant et al. [13] or Hagmeijer [10]. The main advantage of this sensor is that it prevents discontinuities, such as shocks, from dominating the refinement sensor, because at some point their cell length becomes so small that other flow features will start to become important.

Each cell is now adapted independently in all three directions, by dividing the cells which meet the adaptation criterion into two new cells.

## Data Structure

The success of an unstructured grid adaptation algorithm strongly depends on the efficiency of the data structure. The data structure for $h$-type grid adaptation is more complicated than for $r$-type adaptation, because one cell can be connected to multiple neighboring cells. An important criterion in the design of the data structure is that no searching is required in the calculation of the flow field. All the necessary searching to update the data structure is done during the adaptation step. This greatly enhances the efficiency of the code, because all the basic operations then can be vectorized and parallelized using a proper coloring and domain decomposition scheme. Until now, most of the applications with hexahedron type cells where restricted to two-dimensional flows, where generally a quadtree data structure is used. In three dimensions this becomes an octree data structure. An octree data structure is, however, more suited for isotropic cell refinement, where each cell has eight children, but is inefficient for anisotropic grid refinement.

An efficient data structure for the DG method is obtained using the cell faces instead of cells as the basic element. This has several major advantages. The primary loop in a DG method is the calculation of the fluxes, which can be done without any searching using a face based data structure. A second benefit of a face based data structure versus a cell based data structure is that each subface $s_{KK'}^{ij}$ can have only two neighboring cells, whereas each cell can have unlimited neighbors. A loop over cell faces can therefore be done without searching. The face based data structure has some resemblance with the edge based data structure commonly used with vertex based unstructured algorithms using tetrahedrons.

### Grid Structure

Each element $K$ can be subparametrically mapped into its master element $\hat{K}$. The faces and vertices of element $\hat{K}$ are numbered uniquely and the topology of each element $K$ is defined by the coordinates of the vertices and the mapping $F_K$. The following arrays are used to define the grid structure: Array $ICG(icell, n), (n = 1, \cdots, 8)$ to store the addresses of the vertices of the cells and array $IcTree(icell, n), (n = 1, \cdots, 4)$ to store the cell connectivity. The first element of $IcTree$ is the address of the parent cell and the second and third element are the addresses of the first and second child. For efficiency reasons the type of refinement ($\xi, \eta$ or $\zeta$) is also stored.

Due to the dynamic behavior of the grid, points are added and deleted, it is important to store the grid points efficiently. This is done using an AVL-tree, for a detailed description of AVL trees see [11] and [20]. The array $IG\_AVL$ contains the addresses of the $x$, $y$ and $z$-coordinates of the grid points. The AVL-tree uses the same key as proposed in [17], viz. $(x_1, y_1, z_1) < (x_2, y_2, z_2)$ if $x_1 < x_2$, or if $x_1 = x_2$ and $y_1 < y_2$ or if $x_1 = x_2$ and $y_1 = y_2$ and $z_1 < z_2$.

Together with vectors for the $x$, $y$ and $z$-coordinates of the grid points this information is sufficient to describe the grid. The use of an AVL-tree is very efficient. When a cell is divided it is possible to find in $O(log_2(N))$ steps if a grid point already exists in the tree or must be added. Both insertion and deletion of an element in the AVL tree can be done in $O(log_2(N))$ operations, with $N$ the number of grid points.

### Establishing Face to Cell Connectivity

The most difficult part of $h$-type grid adaptation on an unstructured hexahedral mesh is to establish the face to cell connectivity $s_{KK'}^{ij}$. It is impractical to try to determine in advance the large number of possible connections, even if only a limited number of neighboring cells is allowed. The following algorithm can find all possible connections:

At the root grid level all cell connections are known, because they can be obtained from the original unadapted grid. At this level there is no local grid refinement.

For all root cell faces the addresses and face indices of the two cells which connect to this cell face are stored in the array $IfTree$. Next, the tree $IcTree$ is traversed. For each cell face which is the connection between the two children cells $K'$ and $K''$, ($s^{ij}_{K'K''} = e^i_{K'} \wedge s^{ij}_{K'K''} = e^j_{K''}$) the addresses and face indices of the left and right children are also stored in array $IfTree$. The set of these faces and the root cell faces are called elementary faces.

To find the remaining face to cell connections each elementary face is mapped to $[0,1] \times [0,1]$. Then for each side of the elementary face the tree $IcTree$ is traversed to find the local $(s,t)$ coordinates of the four corners and center of the cell faces of the children cells which connect to the elementary face. This can be done easily using the type of refinement, $\xi$, $\eta$ or $\zeta$, and the face index of the elementary cell. If necessary the local coordinate system $(s',t')$ of cell face $e^j_{K'}$ is transformed to the $(s,t)$ coordinate system of cell face $e^i_k$.

The coordinates of the corner points and cell face centers at both sides of the elementary face are stored in arrays $FaceKeyL$ and $FaceKeyR$. For both sides of the cell face also the addresses of the children are stored in separate binary trees $IfTreeL$ and $IfTreeR$, using the cell face center as key. This part of the algorithm has some similarity to that proposed in [17] to find hanging nodes in a node based finite element method. Their problem is a point search problem, but the determination of the face to cell connectivity is a geometric searching problem and the alternating digital tree algorithm is used, [4].

First for all the cells on the left side of the cell face, the tree $IfTreeR$ is traversed to find the cell face at the opposite side which has the same corner points or is completely contained in the left cell face. This can be done in $O(\log_2(N))$ operations. The same is done for all the cells at the right cell face. In order to efficiently eliminate face to cell connections which occur twice, it is necessary to store the new face to cell connections in a binary tree.

After this search most face to cell connections are found, but depending on the refinement strategy it is possible that one cell face connects at both sides to more than one cell. If this happens it's face to cell connection is not established in the previous search and the cell faces for which no connection could be found must be split into two faces on one of the sides of the elementary face. By cyclically splitting the cell faces for which no connection could be found in $s$ and $t$ directions and restarting the search finally all connections will be found. It is easy to test if all cell to face connections are found because their area should add up to one on both sides. After the search is completed all connections which are found are added to the tree $IfTree$.

The alternative to subdividion of cell faces would to further subdivide cells, but this can easily generate new faces which connect to more than one cell. This does not occur with subdivision of cell faces and the searching algorithm will finish in a finite time. The only complication of using subfaces is that they have be accounted for in the flux calculation, because now the face $e^i_k$ is subdivided into several faces instead of one. With this algorithm all face to cell connections can be found and the algorithm can be completely parallelized, because the determination of the subdivision of each elementary face is completely independent from one another.

The update of the cell face fluxes can now be done easily in one loop over the cell faces, without any difficulty caused by hanging nodes. This algorithm can be completely vectorized and parallelized using a proper coloring and domain decomposition scheme.

## Discussion and Results

The grid adaptation algorithm has been tested on the flow around the ONERA M6 wing, [21]. The ONERA M6 wing has a trapezoidal planform with 30° leading edge sweep, and a taper ratio of 0.56. The wing sections are based on the symmetrical ONERA-D profile with 5% thickness/chord ratio. The wing tip is rounded by rotating the tip section around its symmetry axis. The freestream Mach number is 0.84 and angle of attack is 3.06°.

The grid adaptation was started by first calculating a steady solution on the initial grid, which consisted of 8192 cells and 9441 grid points. The grid was subsequently adapted seven times, independently in all three directions and the final grid consisted of 195200 cells and 345210 grid points. See Table 1 for more details. Figure 1. shows the convergence history of the maximum pointwise residual. The spikes indicate the various instances when the grid was adapted.

Local time stepping was used to accelerate convergence. Initially problems were experienced in estimating the local time step accurately on the highly irregular grids which occur after grid adaptation. An efficient algorithm for local time stepping was obtained by modifying the procedure discussed by Barth [9] to the Osher approximate Riemann solver. This algorithm gives an accurate prediction of the local time step and the necessary information about local wave speeds can be obtained directly from the Osher scheme. All calculations were done with a

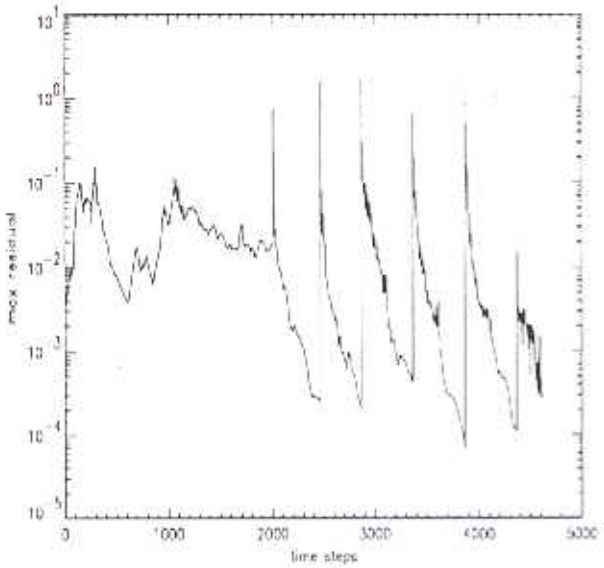


Figure 1: Maximum pointwise residual in flow field

local CFL number of 0.7.

It can be seen that convergence on the initial coarse grid is rather slow. This is caused by the slope limiter which is activated in the far field, for an analysis of this problem see [19]. After grid adaptation convergence improves significantly and approximately 500 time steps were sufficient to obtain at least three orders of magnitude reduction in residual.

The use of the sensor functions $R_K$, equations (4) and (5), which approximate the gradient of the primitive flow variables in all three directions was quite effective in capturing the relevant flow features. Generally the most dominant feature is the stagnation region, especially on the initial coarse grid, but shocks and shear layers were captured reasonably well after refinement. An important feature of this sensor function is that it is weighted with the local grid distance, which prevents one aspect of the flow to constantly dominate the adaptation process. This is strongly influenced by the power of $\Delta\xi_K$ in equation 5

Figures 2 till 9 show the flow field at the upper surface after the various adaptation steps. The initial grid is very coarse and only shows a shock at about 55% chord. The first three adaptation steps significantly improve the resolution of the aft shock. After four adaptation steps the forward shock develops and the solution on the final grid clearly shows the lambda-shock structure. Figure 9 shows that the two shocks merge at 87% span and separate at approximately 94% span. The shock structure compares well with the results obtained by Rausch et al. [15]. The forward shock is weaker than the aft shock and takes longer to be captured by the grid adaptation process. During each adaptation step, except the first, 15% of the cells with the smallest indicator function were deleted, after which the total number of cells was increased with 70% through refinement of cells with the largest indicator function. The adaptation process was completely automatic, no user interference was necessary.

Figure 10 shows the final adapted grid which clearly shows the lambda shock structure. The mesh adapts to regions with large flow activity and significantly improves resolution in the shock regions and around the tip. Due to the fact that the calculation was started on a coarse grid it proved very important to be able to both add and delete cells, because initially the grid is primarily refined in the stagnation and rear shock regions which tend to become overresolved in the initial adaptation stages. It will be more efficient to start on a finer initial grid, but this would require multigrid convergence acceleration to obtain a steady initial solution. Further improvements in sensor functions will also contribute to improved efficiency in the grid adaptation process.

The calculations were done on a NEC SX-3 2/2 computer at NLR and required approximately 3 hours of CPU time and 60 Mwords of main memory. The program runs approximately at a speed of 350 Mflops on a single processor.

## Concluding Remarks

The extension of the discontinuous Galerkin method using hexahedron type cells to three dimensional inviscid, compressible flow has been successfully demonstrated. It was shown that the DG method can be combined with anisotropic grid adaptation which significantly improved the accuracy. A new algorithm to establish face to cell connectivity was presented which works well with $h$-refinement of hexahedron cells and the DG method. Steady transonic flow results of the ONERA M6 wing were presented which demonstrated the efficiency of the adaptation algorithm in capturing the lambda shock wave. The DG method is a very local scheme and works well on highly irregular grids. The combination of the DG method with grid adaptation looks promissing for extension to viscous flows, especially Large Eddy Simulation which will require very sophisticated grid adaptation methods.

| Adaptation Step | Cells | Grid Points |
|---|---|---|
| 0 | 8192 | 9441 |
| 1 | 14156 | 17647 |
| 2 | 21010 | 29601 |
| 3 | 31416 | 48744 |
| 4 | 47565 | 80716 |
| 5 | 72092 | 128202 |
| 6 | 109881 | 205948 |
| 7 | 195200 | 345210 |

Table 1. Number of grid points and cells after each adaptation step

## References


[1] Aftosmis, M.J. Upwind method for simulation of viscous flow on adaptively refined meshes. *AIAA J.*, 32:268–277, 1994.

[2] Barth, T.J. and Jespersen, D.C. The design and application of upwind schemes on unstructured meshes. AIAA Paper 89-0366, 1989.

[3] Bey, K.S. and Oden, J.T. A Runge-Kutta discontinuous finite element method for high speed flows. AIAA Paper 91-1575-CP, 1991.

[4] Bonet, J. and Peraire, J. An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems. *Int. J. for Num. Meth. in Eng.*, 31:1–17, 1991.

[5] Cockburn, B and Shu, C.W. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II. General framework. *Math. Comp.*, 52:411–435, 1989.

[6] Cockburn, B. and Shu, C.W. The $P^1$-RKDG method for two-dimensional Euler equations of gas dynamics. Technical Report 91-32, ICASE, 1991.

[7] Cockburn, B., Hou, S. and Shu, C.W. The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case. *Math. Comp.*, 54:545–581, 1990.

[8] Cockburn, B., Lin, S.Y. and Shu, C.W. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One-dimensional systems. *JCP*, 84:90–113, 1989.

[9] Deconinck, H. and Barth, T. Special course on unstructured grid methods for advection dominated flows. AGARD Report 787, 1992.

[10] Hagmeijer, R. Grid adaptation based on modified anisotropic diffusion equations formulated in the parametric domain. *JCP*, 115:169–183, 1994.

[11] Knuth, D.E. *The Art of Computer Programming, Volume 3, Sorting and Searching.* Addison-Wesley, 1973.

[12] Lin, S.Y. and Chin, Y.S. Discontinuous Galerkin finite element method for Euler and Navier-Stokes equations. *AIAA J.*, 31:2016–2026, 1993.

[13] Marchant, M.J. and Weatherhill, N.P. Adaptivity techniches for compressible inviscid flows. *Comp. Meth. in Appl. Mech. and Eng.*, 106:83–106, 1993.

[14] Osher, S. and Chakravarthy, S. Upwind schemes and boundary conditions with applications to Euler equations in general geometries. *JCP*, 50:447–481, 1983.

[15] Rausch, R.D., Batina, J.T. and Yang, H.T.Y. Spatial adaptation procedures on tetrahedral meshes for unsteady aerodynamic flow calculations. AIAA Paper 93-0670, 1993.

[16] Shu, C.W. and Osher, S. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *JCP*, 77:439–471, 1988.

[17] Tan, Z. and Varghese, P.L. Directionally adaptive finite element method for multidimensional Euler and Navier-Stokes equations. AIAA Paper 93-3320-CP, 1993.

[18] Van Der Vegt, J.J.W. Higher-order accurate Osher schemes with application to compressible boundary layer stability. AIAA Paper 93-3051, 1993.

[19] Venkatakrishnan, V. On the accuracy of limiters and convergence to steady state solutions. AIAA Paper 93-0880, 1993.

[20] Wirth, N. *Algorithms & Data Structures.* Prentice-Hall, 1986.

[21] Yoshihara, H. and Sacher, P. Test cases for inviscid flow field methods. AGARD Advisory Report 211, 1985.

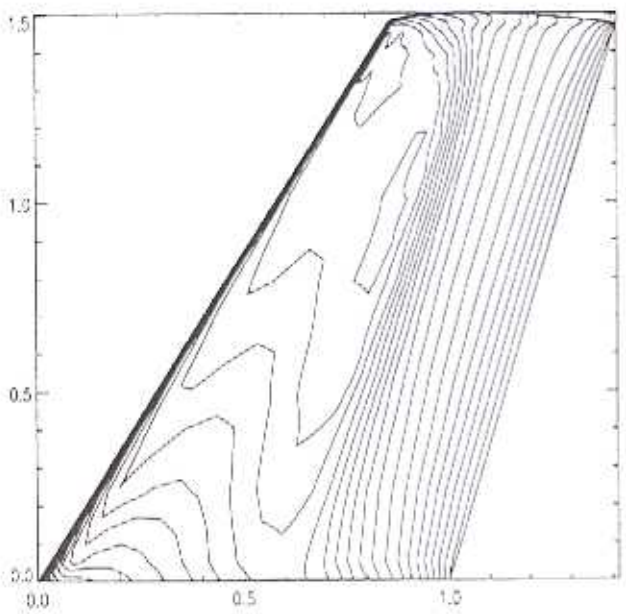


Figure 2: Pressure field on ONERA M6 wing at coarse grid

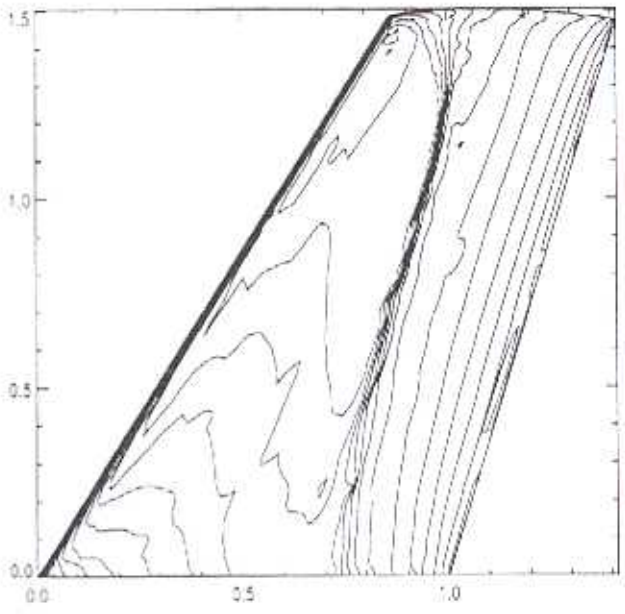


Figure 4: Pressure field on ONERA M6 wing after two adaptation steps

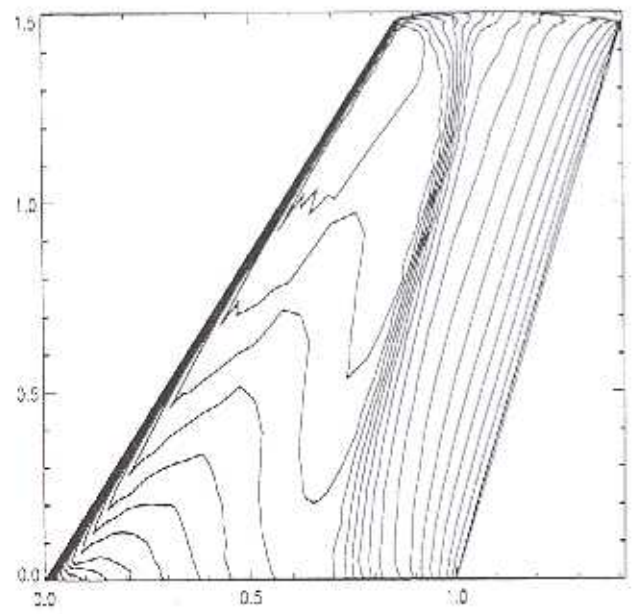


Figure 3: Pressure field on ONERA M6 wing after one adaptation step

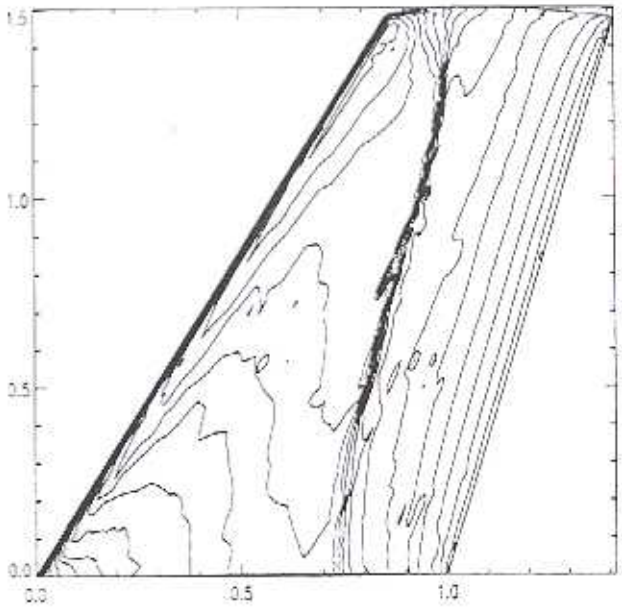


Figure 5: Pressure field on ONERA M6 wing after three adaptation steps

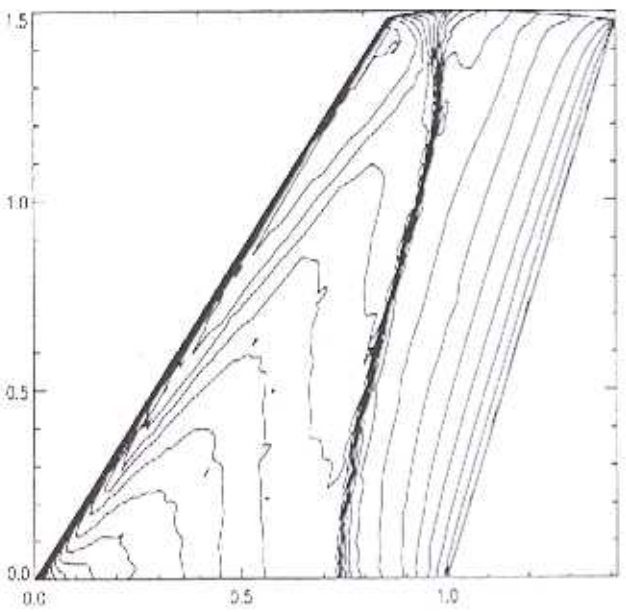


Figure 6: Pressure field on ONERA M6 wing after four adaptation steps

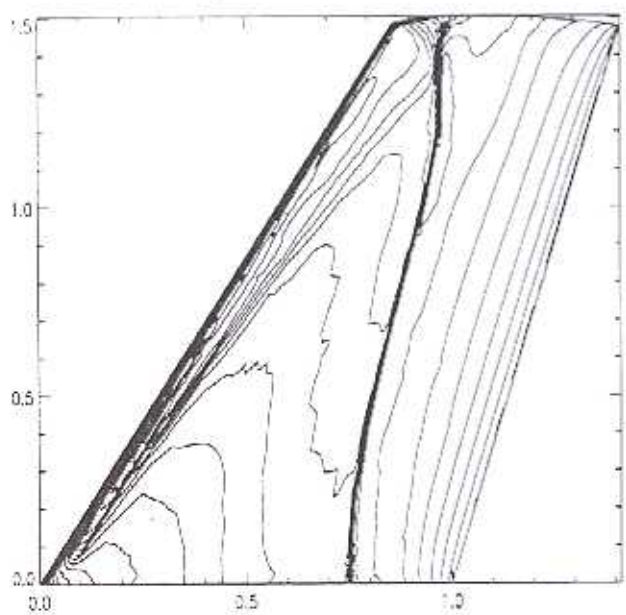


Figure 7: Pressure field on ONERA M6 wing after five adaptation steps

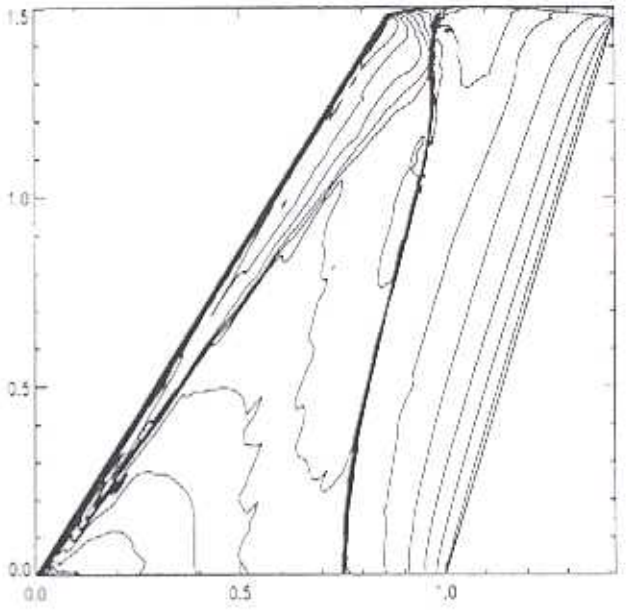


Figure 8: Pressure field on ONERA M6 wing after six adaptation steps

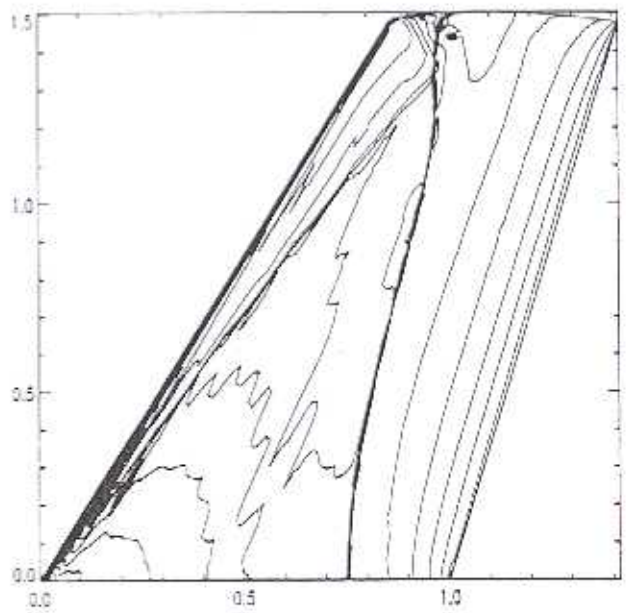


Figure 9: Pressure field on ONERA M6 wing after seven adaptation steps

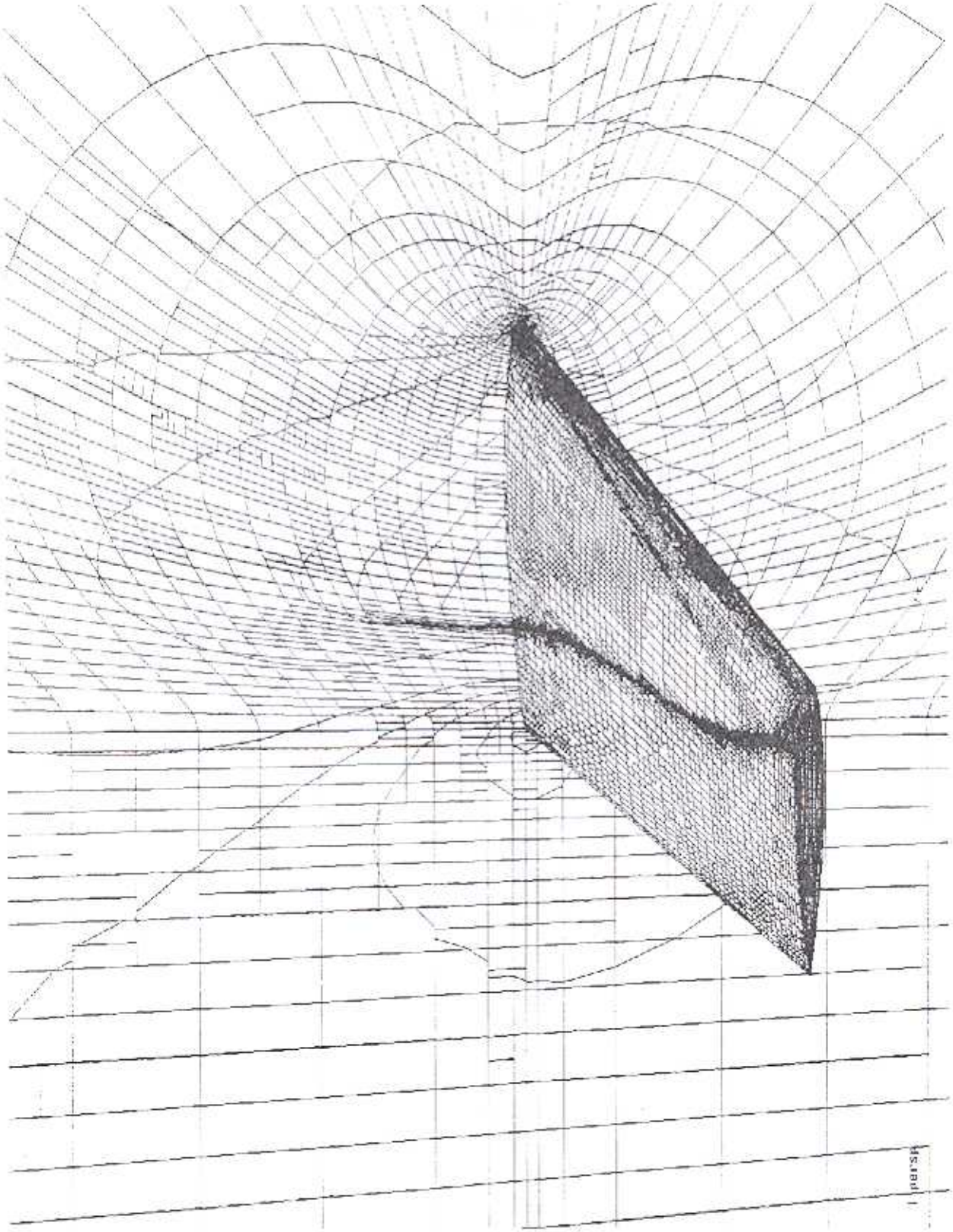

Figure 10: Final adapted grid on ONERA M6 wing