AGARD-CP-578

AGARD-CP-578

AGARD

ADVISORY GROUP FOR AEROSPACE RESEARCH & DEVELOPMENT
7 RUE ANCELLE, 92200 NEUILLY-SUR-SEINE, FRANCE

AGARD CONFERENCE PROCEEDINGS 578

Progress and Challenges in CFD Methods and Algorithms

(Progrès réalisés et défis en méthodes et algorithmes CFD)

Papers presented and discussions recorded at the 77th Fluid Dynamics Panel Symposium held in Seville, Spain, 2-5 October 1995.

NORTH ATLANTIC TREATY ORGANIZATION

Published April 1996

Distribution and Availability on Back Cover

HEXAHEDRON BASED GRID ADAPTATION FOR FUTURE LARGE EDDY SIMULATION

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

SUMMARY

This paper discusses a new numerical method which enables the future application of Large Eddy Simulation to high Reynolds number aerodynamic flows. The new numerical method uses local grid refinement of hexahedral cells and the discontinuous Galerkin finite element method. This method offers maximum flexibility in grid adaptation and maintains accuracy on highly irregular grids. The method is demonstrated with calculations of inviscid transonic flow on a generic delta wing. The calculations are done on two parallel shared memory computers and the performance results are used to give estimates of the computing time and memory requirements for a Large Eddy Simulation of a clean wing on a NEC SX-4 supercomputer.

LIST OF SYMBOLS

b_K	external boundary face of element K
$\mathcal{B}$	boundary operator
$C^1[0,T]$	space of one time differentiable functions on the interval $[0,T]$
δ_{ij}	Kronecker delta symbol
E	specific total energy
e_K	face of polyhedron K
$\mathbf{F}^j(\mathbf{U})$	flux vector in Cartesian coordinate direction j
$\hat{\mathbf{F}}(\mathbf{U})$	inner product of $\mathbf{n}^T$ and $\mathcal{F}$
$\mathcal{F}(\mathbf{U})$	matrix with columns $\mathbf{F}^j$
F_K	mapping between elements $\hat{K}$ and K
γ	ratio of specific heats
$\cup_\alpha \Gamma_\alpha$	path in phase space between $\mathbf{U}_h^{int(K)}$ and $\mathbf{U}_h^{ext(K)}$
$\mathbf{h}(\mathbf{U}_h^{int(K)}, \mathbf{U}_h^{ext(K)})$	monotone Lipschitz flux
K	polyhedron element in $\mathcal{T}_h$
K'	neighboring elements of polyhedron K
$\hat{K}$	master element of polyhedron K
meas(K)	measure of polyhedron K
∂K	boundary of polyhedron K
M	maximum number of polynomial terms in expansion of $\mathbf{U}_h$
$[M_K]$	mass matrix of element K
N^+	set of positive natural numbers
$N(K)$	set of neighboring elements of K
$N^\xi(K)$	indices of neighboring elements of K in the ξ-direction
$\mathbf{n}$	unit outward normal vector
Ω	flow domain
$\partial\Omega$	boundary of Ω
p	pressure
$P^k(\hat{K})$	space of polynomial functions of degree $\leq k$ on $\hat{K}$
$P^k(K)$	space of functions whose images under F_K are functions in $P^k(\hat{K})$
$\mathbf{\Phi}_K$	limiter function defined on K
$\mathbf{\Phi}_K^i$	components of limiter function on K
$\hat{\phi}_j(\xi,\eta,\zeta)$	polynomial basis functions on $\hat{K}$
$\phi_j(\mathbf{x})$	basis functions on K
$\psi_i(\xi,\eta,\zeta)$	trilinear element shape functions
$\mathbf{R}_K$	residual in element K
$R_K^\xi, R_K^\eta, R_K^\zeta$	indicator functions for grid adaptation in ξ, η and ζ directions
R^n	Euclidian n-dimensional space
ρ	density
span	linear span
t	time
T	final time
$\mathcal{T}_h$	triangulation of Ω
$\mathbf{U}$	conservative flow variables
$\bar{\mathbf{U}}_K$	average of $\mathbf{U}$ in element K
$\mathbf{U}_w$	conservative flow variables specified at $\partial\Omega$
$\mathbf{U}_0$	initial conservative flow variables
$\mathbf{U}\|_K$	$\mathbf{U}$ restricted to element K
$\mathbf{U}_h$	numerical approximation of $\mathbf{U}$
$\mathbf{U}^{ext(K)}$	$\mathbf{U}$ at cell face taken as the limit from the exterior of K
$\mathbf{U}^{int(K)}$	$\mathbf{U}$ at cell face taken as the limit from the interior of K
$\mathbf{U}_K^{max}$	maximum $\mathbf{U}$ in K and it's neighboring cells
$\mathbf{U}_K^{min}$	minimum $\mathbf{U}$ in K and it's neighboring cells
U_{K*}^i	components of $\mathbf{U}_h$ at Gauss quadrature points in cell faces of $\hat{K}$
$\hat{\mathbf{U}}_m$	components of polynomial expansion of $\mathbf{U}$ in K
$\tilde{\mathbf{U}}_m$	limited components of polynomial expansion coefficients $\hat{\mathbf{U}}_m$ in K
$\tilde{\mathbf{U}}_h$	limited flow field $\mathbf{U}$ in each element
$\tilde{\mathbf{U}}_K$	vector with limited moments of flow field $\tilde{\mathbf{U}}_m$
u_i	Cartesian velocity components
$\mathbf{V}$	primitive flow variables
$\mathbf{V}_h^k(K)$	vectors with each component $p_i \in P^k(K)$
$\mathbf{W}_h$	vectors which belong to space $\mathbf{V}_h^k$
$\mathbf{x}$	position vector
x_j	components of position vector, $j = \{1,2,3\}$
$\mathbf{x}_K^i$	coordinates of corner points of element K
$\triangle\xi_K$	length of cell in local ξ-direction
ξ,η,ζ	local coordinates in element $\hat{K}$
$\forall$	for all
∇	nabla operator
$\subset$	subset
$\in$	element
$\circ$	composite mapping
$\otimes$	tensor product
^{T}	transposed

Paper presented at the AGARD FDP Symposium on "Progress and Challenges in CFD Methods and Algorithms" held in Seville, Spain, from 2-5 October 1995, and published in CP-578.

INTRODUCTION

Computational Fluid Dynamics (CFD) is used for increasingly complicated problems. Many advanced applications of CFD, such as Large Eddy Simulation (LES), can only be done with sophisticated grid adaptation algorithms and require significant computer resources. The aim of this paper is to demonstrate a new grid adaptation algorithm for future application to Large Eddy Simulation. With LES the filtered Navier-Stokes equations are solved which represent the part of the turbulent flow field that can be resolved on the grid. The turbulent length scales which can not be resolved have to be modeled with subgrid scale turbulence models. This approach is quite successful in most parts of the flow field, but as already mentioned by Chapman [3], fails in the near wall region which is critical for LES. Chapman proposed to use successively finer grids close to the wall to capture the viscous sublayer. This reduces the need to model the near wall region where the basic assumption of LES, namely the separation of the flow field in large and small scales, is not valid.

Despite the significant progress made in LES since Chapman's paper the proper solution of the near wall flow field is still one of the key elements preventing LES to be applied to more general problems in aerospace, Moin and Jimenez [10]. The use of successively finer grids can only be done efficiently with sophisticated grid adaptation techniques and requires a numerical scheme which is accurate on highly irregular grids. In this paper a new algorithm is presented, using a combination of local grid refinement and the discontinuous Galerkin (DG) finite element method. This method is capable of efficiently resolving local phenomena such as shear layers and shocks and has the potential to be applied to LES of wall bounded turbulent flows by properly resolving the near wall region. Hexahedron cells are used as basic elements because they suffer less from loss of accuracy due to successive refinements than the more commonly used tetrahedron cells and are more suited to viscous flows. This paper, however, will be limited to inviscid flow in order to demonstrate the basic algorithm.

The discontinuous Galerkin method with Runge-Kutta time integration (RKDG) was originally proposed by Cockburn and Shu [4, 6, 5] for hyperbolic conservation laws. They proved that the RKDG method is TVB stable and satisfies a maximum principle for multi-dimensional scalar hyperbolic conservation laws. This work was mainly theoretical and limited to one and two-dimensional flow fields. The extension to three dimensions was recently presented by van der Vegt [14]. The discontinuous Galerkin method uses a local polynomial expansion in each cell which results in a discontinuity at each cell face. This discontinuity can be represented as a Riemann problem which provides a natural way to introduce upwinding into a finite element method. The DG method can therefore be considered as a mixture of an upwind finite volume method and a finite element method.

A key feature of the DG method is that also equations for the moments of the flow field are solved. In this way a completely local higher order accurate spatial discretization can be obtained without the need to use neighboring cells in the discretization. An alternative to obtain the flow field gradients is to use Gauss' identity, but this method requires grid regularity to be accurate. The use of the moment equations is extremely useful in combination with local grid refinement because no problems with hanging nodes occur and the scheme maintains it's accuracy on highly irregular grids, which generally occur after several grid refinement steps. In this paper the spatial accuracy is limited to second order and the moments represent the flow field gradients. A disadvantage of using the moment equations is that more memory is needed to store the additional moments of the flow field. For future LES applications in wall bounded flows these disadvantages are, however, more than compensated by the increased computational efficiency of the adapted grid.

The DG method makes it easy to mix different types of elements. As basic elements hexahedrons are used, but whenever necessary due to topological degeneracies, prisms, tetrahedrons and other degenerated hexahedrons are used. The initial coarse grid is obtained from a multi-block structured grid, generated with the NLR ENFLOW system. This grid is transformed into an unstructured grid using a face-based data structure, van der Vegt [14]. This data structure is more suited to anisotropic local grid refinement than the commonly used octree data structure. Anisotropic grid refinement is important because many flow phenomena are locally pseudo two-dimensional, eg. shocks and shear layers, and can not be efficiently captured with isotropic grid refinement.

The DG method combined with the face based data structure is extremely local in nature and makes it a good candidate for parallel computing. Parallel computers offer the possibility to overcome the physical limits on single processor speed, but require a significant effort to optimize numerical schemes and coding. LES requires significant computer resources and the performance of the DG method on two different types of parallel shared memory computers, namely a two processor NEC SX-3 and a four processor SGI Power Challenge, will be discussed in this paper. The choice for parallel shared memory computers is made initially to limit the effort in modifying codes.

The outline of the paper is as follows. After a brief description of the governing equations, the DG method will be discussed followed by a description of the grid adaptation algorithm. The algorithm will be demonstrated on the flow field around a generic delta wing. Next, several aspects of using parallel shared memory computers will be discussed and performance results will be presented. These data will be used to give an estimate of the computational complexity of a LES of a clean wing. The papers finishes with concluding remarks.

GOVERNING EQUATIONS

The Euler equations for inviscid gas dynamics in conservation form can be expressed in the flow domain Ω as:

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

Here $\mathbf{x}$ and t represent the coordinate vector, with components $x_i, i = \{1,2,3\}$, in the Cartesian directions, and time, respectively. The Euler equations are supplemented with initial condition $\mathbf{U}(\mathbf{x},0) = \mathbf{U}_0(\mathbf{x})$ and boundary condition $\mathbf{U}(\mathbf{x},t)|_{\partial\Omega} = \mathcal{B}(\mathbf{U},\mathbf{U}_w)$; where B denotes the boundary operator and $\mathbf{U}_w$ the prescribed boundary data. The vectors with conserved flow variables $\mathbf{U}$ and fluxes $\mathbf{F}^j$, $j = \{1,2,3\}$, 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 ρ, p and E denote the density, pressure and specific total energy and u_i the velocity in the Cartesian coordinate directions x_i, $i = \{1,2,3\}$ and δ_{ij} the Kronecker delta symbol. The summation convention is used on repeated indices. This set of equations is completed with the equation of state: $p = (\gamma - 1)\rho(E - \frac{1}{2}u_i u_i)$, with γ the ratio of specific heats.

DISCONTINUOUS GALERKIN APPROXIMATION

The flow domain Ω, which is assumed to be a polyhedron, is covered with a triangulation $\mathcal{T}_h = \{K\}$ of hexahedrons, which are related to the master element $\hat{K}$ through the mapping F_K:

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

with $\psi_i(\xi,\eta,\zeta)$ the standard linear finite element shape functions and $\mathbf{x}_K^i$ the coordinates of the vertices of the hexahedron K.

Define on the master element $\hat{K} = [-1,1]^3$ the space of polynomials: $P^k(\hat{K}) = \text{span}\{\hat{\phi}_j(\xi,\eta,\zeta), j = 0,\cdots,M\}$ and the related space $P^k(K)$ as the space of functions whose images under F_K are functions in $P^k(\hat{K})$: $P^k(K) = \text{span}\{\phi_j(\mathbf{x}) = \hat{\phi}_j \circ F_K^{-1}, j = 0,\cdots,M\}$. In this paper $k = 1$, which yields a second order accurate spatial discretization with polynomials $\hat{\phi} \in \{1,\xi,\eta,\zeta\}$ with $M = 3$.

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

$$\mathbf{U}_h(\mathbf{x},t) = \sum_{m=0}^{3} \hat{\mathbf{U}}_m(t)\phi_m(\mathbf{x}). \tag{1}$$

The expansion of $\mathbf{U}$ is local in each element and there is no continuity across element boundaries, which is a major difference with node based Galerkin finite element methods. The element based expansion has as important benefit that hanging nodes, which frequently appear after local grid refinement, do not give any complications. Degenerated hexahedrons, such as prisms and tetrahedrons, which are necessary to deal with topological degeneracies in the grid, are allowed without further complications because the degenerated surfaces do not contribute to the flux balance.

The discontinuous Galerkin finite element formulation of the Euler equations is given by:

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

$$\begin{aligned} \frac{\partial}{\partial t}\int_K \mathbf{W}_h^T(\mathbf{x})\mathbf{U}_h(\mathbf{x},t)d\Omega = \\ - \int_{e_K} \mathbf{W}_h^T(\mathbf{x})\left(\mathbf{n}^T(\mathbf{x})\mathcal{F}(\mathbf{U}_h)\right)dS \\ - \int_{b_K} \mathbf{W}_h^T(\mathbf{x})\left(\mathbf{n}^T(\mathbf{x})\mathcal{F}(\mathbf{B}(\mathbf{U}_h,\mathbf{U}_w))\right)dS \\ + \int_K \nabla\mathbf{W}_h^T(\mathbf{x})\mathcal{F}(\mathbf{U}_h)d\Omega, \end{aligned} \tag{2}$$

with $\mathcal{F} = \mathbf{F}^j$, $j = \{1,2,3\}$, and $e_K \subset \partial K \backslash \partial\Omega$ and $b_K \subset \partial K \cap \partial\Omega$ the faces of element K in the interior and at the boundary of the domain Ω, respectively. The vector $\mathbf{n}^T$ represents the transposed unit outward normal vector at ∂K.

The flux at the faces e_K, namely $\mathbf{n}^T\mathcal{F}(\mathbf{U}) \equiv \hat{\mathbf{F}}(\mathbf{U})$, is not clearly defined, because the flow field $\mathbf{U}_h$ is discontinuous at the cell faces. The flux is therefore replaced with a monotone flux function $\mathbf{h}(\mathbf{U}_h^{int(K)}, \mathbf{U}_h^{ext(K)})$, which is consistent, $\mathbf{h}(\mathbf{U},\mathbf{U}) = \hat{\mathbf{F}}(\mathbf{U})$. Here $\mathbf{U}^{int(K)}$ and $\mathbf{U}^{ext(K)}$ denote the value of $\mathbf{U}$ at ∂K taken as the limit from the interior and exterior of K. More details can be found in Cockburn et al. [5]. 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 [11] 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, [13], which is important for future extension of the algorithm to the Navier-Stokes equations. The Osher approximate Riemann solver is defined as:

$$\begin{aligned} \mathbf{h}(\mathbf{U}_h^{int(K)}, \mathbf{U}_h^{ext(K)}) = \frac{1}{2}(\hat{\mathbf{F}}(\mathbf{U}_h^{int(K)}) + \hat{\mathbf{F}}(\mathbf{U}_h^{ext(K)}) - \\ \sum_\alpha \int_{\Gamma_\alpha} |\partial\hat{\mathbf{F}}| d\Gamma), \end{aligned}$$

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 [11]. At the boundary surface the path Γ_α 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, [11], and a completely unified and consistent treatment of the flux calculations is obtained, both at interior and exterior faces.

The first order accurate discontinuous Galerkin method with an (approximate) Riemann solver yields monotone results, but second and higher order discretizations need a slope limiter to prevent numerical oscillations around discontinuities and in regions with steep gradients. Cockburn et al. [5] derived a local projection limiter on **B**-triangulations for multi-dimensional scalar conservation laws, which gives a second order accurate scheme and satisfies a maximum principle when combined with a TVD Runge-Kutta time integration method [12]. The extension to quadrilaterals is presented by Bey and Oden [2], but turned out to be very dissipative.

In this paper a different approach is followed. The second order discontinuous Galerkin method strongly resembles a MUSCL upwind scheme, with as main difference the procedure to determine the flow gradient. In the DG-method the gradient is determined by solving equations for the moments $\hat{\mathbf{U}}_m$, $m = \{1,2,3\}$, whereas the MUSCL scheme determines the

gradient using data from surrounding cells. The same limiting procedure can, however, be followed. In this paper the multi-dimensional limiter from Barth and Jesperson [1], with the modifications proposed by Venkatakrishnan [15], is used. The limiter from Barth and Jespersen has as benefit that it is a truly multi-dimensional limiter and yields a positive scheme.

The limiter from Barth and Jespersen can, however, seriously degrade convergence to steady state. This was analysed by Venkatakrishnan [15] and the two main causes for this phenomenon are the non-smoothness of the limiter, which uses min- and max-functions, and the fact that the limiter is active in smooth parts of the flow, eg. in the far field.

The limiter according to Venkatakrishnan [15] is directly applied to the conservative variables, which saves the considerable expense of computing the local characteristic decomposition.

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

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

with $N(K)$ the set of neighboring cells which connect to cell K. In order to maintain monotonicity the approximate flow field $\mathbf{U}_h$ must satisfy $\mathbf{U}_h(\mathbf{x}) \in [\mathbf{U}^{min}_K, \mathbf{U}^{max}_K], \forall \mathbf{x} \in K$, which is accomplished with the limiter function $\mathbf{\Phi}_K$ defined as:

$$\mathbf{\Phi}^i_K = \begin{cases} \phi_L\left(\frac{U^i_{K\,\max} - \bar{U}^i_K}{U^i_{K^*} - \bar{U}^i_K}\right) & \text{if} \quad U^i_{K^*} - \bar{U}^i_K > 0 \\ \phi_L\left(\frac{U^i_{K\,\min} - \bar{U}^i_K}{U^i_{K^*} - \bar{U}^i_K}\right) & \text{if} \quad U^i_{K^*} - \bar{U}^i_K < 0 \\ 1 & \text{if} \quad U^i_{K^*} - \bar{U}^i_K = 0 \end{cases}$$

Here $U^i_{K^*}$ are the components of $\mathbf{U}_h$ at the Gauss quadrature points in $\hat{K}$, used to evaluate the integrals in equation (2). The function $\phi_L(y)$ replaces $\min(1, y)$ in the original Barth and Jesperson limiter and is defined as:

$$\phi_L(y) = \frac{y^2 + 2y}{y^2 + y + 2}$$

Defining $\triangle = U^i_{K^*} - \bar{U}^i_K$, $\triangle_+ = U^i_{K\,\max} - \bar{U}^i_K$ and $\triangle_- = U^i_{K\,\min} - \bar{U}^i_K$ and replacing $\triangle^2_\pm$ with $\triangle^2_\pm + \epsilon^2$ a smoother limiter is obtained:

$$\mathbf{\Phi}^i_K = \begin{cases} \frac{\triangle^2_+ + \epsilon^2_K + 2\triangle\triangle_+}{\triangle^2_+ + \epsilon^2_K + 2\triangle^2 + \triangle\triangle_+} & \text{if} \quad \triangle > 0 \\ \frac{\triangle^2_- + \epsilon^2_K + 2\triangle\triangle_-}{\triangle^2_- + \epsilon^2_K + 2\triangle^2 + \triangle\triangle_-} & \text{if} \quad \triangle < 0 \\ 1 & \text{if} \quad \triangle = 0 \end{cases}$$

The coefficient ϵ_K is set equal to $\epsilon_K = (C \triangle s_K)^3$, with $\triangle s_K$ the minimum distance between the cell face centers of two opposite faces of element K. The constant C determines the balance between limiting and no limiting and thereby influences the convergence to steady state. If $C = 0$ the original Barth and Jespersen limiter is obtained. In this paper $C = 1$ is used.

The limiter $\mathbf{\Phi}_K$ is applied independently to each component of the flow field: $\tilde{U}^i_m = \mathbf{\Phi}^i_K \hat{U}^i_m$, $m = \{1, 2, 3\}$. This is slightly less robust then using $\mathbf{\Phi}_K = \min_i \mathbf{\Phi}^i_K$, but gives significantly less numerical dissipation. The coefficients $\hat{\mathbf{U}}_m$, $m = \{1, 2, 3\}$ in equation (1) 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 $\hat{\mathbf{U}}_0$:

$$\tilde{U}^i_0 = \hat{U}^i_0 + \frac{1 - \mathbf{\Phi}^i_K}{\text{meas}(K)} \sum_{m=1}^{3} \hat{U}^i_m \int_K \phi_m(\mathbf{x}) d\Omega$$

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

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

The final discontinuous Galerkin finite element discretization is now obtained by evaluating the integrals over the element K and it's boundary ∂K in equation (2). This is done using the transformation F_K, between K and the master element $\hat{K}$. The integrals $\int \mathbf{W}^T_h \mathbf{U}_h d\Omega$, are calculated analytically, which requires quite some algebra, whereas the other integrals are calculated with Gauss quadrature rules. Cockburn et al. [5] proved that if the quadrature rules for the surface integrals in equation (2) 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. The volume integrals require six point Gauss quadrature rules.

The use of four and six point Gauss quadrature rules is, however, unnecessarily expensive. The number of flux calculations in the approximation of the surface integrals can be reduced from four to one using the following approximation, which is second order accurate in the mean:

$$\begin{aligned} \int_{\partial\Omega} \phi_n(\mathbf{x}) \mathbf{n}^T \mathcal{F}(\mathbf{U}) d\Omega &= \int_{\partial\hat{\Omega}} \hat{\phi}_n(\mathbf{x}) \mathbf{n}^T \mathcal{F}(\mathbf{U}) J_e d\hat{\Omega} \\ &\cong \mathcal{F}(\mathbf{U})|_c \int_{\partial\hat{\Omega}} \hat{\phi}_n(\mathbf{x}) \mathbf{n}^T J_e d\hat{\Omega} \end{aligned}$$

with $\mathcal{F}(\mathbf{U})|_c$ calculated at the cell face center and J_e the Jacobian of the transformation of the cell face $\partial\Omega$ to $\partial\hat{\Omega}$ on the master element $\hat{K}$. The integrals $\int_{\partial\hat{\Omega}} \hat{\phi}_n(\mathbf{x}) \mathbf{n}^T J_e d\hat{\Omega}$ are precalculated with four point Gauss quadrature rules, which are exact using elements defined with linear shape functions, and therefore free stream consistency is preserved with this approximation. A similar approximation can be made for the volume integral $\int_K \nabla \mathbf{W}^T_h(\mathbf{x}) \mathcal{F}(\mathbf{U}_h) d\Omega$, with $\mathcal{F}(\mathbf{U})$ calculated in the center of $\hat{K}$ and the geometrical part of the volume integral precalculated with a six point Gauss quadrature rule. This formulation requires about four times less computing time than using the more accurate evaluation of the flux integrals and yields similar results. The discretization using four and six Gauss quadrature points for the surface and volume integrals yields, however, a slightly more robust scheme on coarse grids. This is mainly due to the fact that the cross-coupling terms in the moment equations are retained in this case.

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 (2). The equations for $\frac{\partial}{\partial t}\tilde{\mathbf{U}}_K$ are integrated in time using the third order TVD Runge-Kutta scheme from Shu [12]. 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 use of increasingly finer grids in LES in the near wall region, as proposed by Chapman [3], and in other regions with strong shear layers or shocks can be most efficiently done using local grid refinement. The grid is locally enriched by subdividing cells, independently in each of the three local grid directions, ξ, η or ζ, of $\hat{K}$. This anisotropic grid refinement is more efficient in capturing local flow phenomena than isotropic refinement, because many flow features are frequently pseudo two-dimensional. A coarse initial grid is used, which is generated with a multi-block structured grid generator, and transferred into an unstructured hexahedron grid. If necessary degenerated hexahedrons, such as prisms and tetrahedrons, are allowed to deal with topological degeneracies. After calculating the flow field, the grid cells are split in the local ξ-direction if:

$$\frac{R_K^\xi}{\max_{\forall K \in \mathcal{T}_h} R_K^\xi} > \text{tolerance}$$

with the sensor function R_K^ξ for the cell K defined as:

$$R_K^\xi = \max_{i \in \{1,\cdots,5\}, \forall K' \in N^\xi(K)} (V_K^i - V_{K'}^i)^2 \triangle \xi_K^2 \qquad (3)$$

Here $\triangle \xi_K$ is the length of cell K in the local ξ-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 ξ-direction. Equivalent expressions are used for the η and ζ directions. This sensor is based on an equidistribution principle, see for instance Marchant et al. [9]. An important advantage of this sensor is that it prevents regions with discontinuities from constantly dominating the local grid refinement. After several refinements the relative contribution of regions with discontinuities reduces, because $\triangle \xi_K$ in equation 3 becomes progressively smaller.

DATA STRUCTURE

The discontinuous Galerkin method with local grid refinement of hexahedrons requires a significantly different data structure than the frequently used edge based data structure. The edge based data structure is very efficient for unstructured vertex based schemes using tetrahedrons. The discontinuous Galerkin method is a cell based algorithm and the primary calculations are the evaluation of fluxes through cell faces. This can be done efficiently using a face based data structure. A face based data structure also has as important benefit that there are no limitations on the number of cells which can connect to one cell face and is crucial for local grid refinement. The alternative would be an octree data structure, but this data structure does not combine well with anisotropic grid refinement. In van der Vegt [14] an algorithm is presented to determine all face to cell connections efficiently. The main element in this algorithm is that cell faces are split into smaller subfaces until each face connects only to one cell on each side. There are no limits on the number of neighboring cells and using advanced searching algorithms a very efficient scheme is obtained, which can establish all face to cell connections in $O(N\log_2(N))$ operations with N the number of faces. The fluxes are calculated in one loop over all the faces, which can be fully vectorized using a coloring scheme. The face based data structure does not put any limitations on the number of neighboring cells, but if the number of cells connecting to one face becomes too large then the number of colors significantly increases. This reduces the efficiency on vector and parallel computers and will be a topic of future research. In the grid adaptation process cells are added and deleted which is done efficiently using AVL-trees, for more details see van der Vegt [14]

Adaptation Step	Cells	Grid Points	Faces
0	19152	20790	59594
1	33094	38277	132038
2	49088	63357	203400
3	73091	104435	307783
4	124030	197424	538109
5	211578	357752	933616
6	322708	592441	1447763

Table 1: *Number of cells, grid points and faces after each adaptation step*

DISCUSSION AND RESULTS

The grid adaptation algorithm has been tested on the flow around a generic delta wing. The geometry is a cropped-delta wing with a 65-degree sweep angle and a sharp leading edge. A constant airfoil section in the streamwise direction is used (modified NACA 64A005 profile; straight line aft of 75% chord) with 5% relative thickness, no twist and camber. More information about the geometry and experimental results can be found in Elsenaar et al. [7]. A transonic flow test case is used with angle of attack $\alpha = 20°$ and free stream Mach number $M_\infty = 0.85$. The initial grid consisted of 19152 cells and 20790 grid points. The grid is adapted six times, independently in all three directions and the final grid consists of 322708 cells and 592441 grid points, see Table 1. During each adaptation step approximately 15 % of the cells is deleted, after which the number of cells is increased between 70 % and 90%. The removal of grid cells is important, because initially on the coarse grid the refinement sensor is less accurate and some unnecessary refinement takes place. Local time stepping is used and significantly improves convergence to steady state, see Figure 1. The sharp peaks in the convergence plot are caused by the grid adaptation, except for the first peak, which results from freezing the slope limiter after 750 time steps to improve converge. Freezing of the slope limiter is not necessary after grid adaptation.

Figure 3 shows the pressure field and grid lines on the leeward side of the delta wing. The flow field is dominated by a strong primary vortex which starts at the apex and moves downstream under an angle of 20 degrees with the streamwise direction. Vorticity is generated at the sharp leading edge in a thin vortex sheet and rolls-up into the primary vortex. The velocity under this vortex, just above the upper surface, becomes very large and a strong shock develops between the primary vortex and

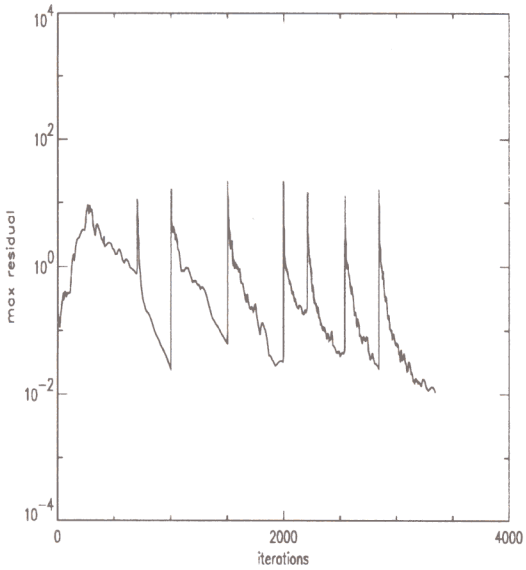

Figure 1: *Maximum residual in flow field.*

the upper surface, see also Figures 4 and 6. The benefits of anisotropic grid refinement are very clear in Figure 3, where the grid is strongly adapted along the primary vortex in the first 85% of the delta wing, where the flow field is approximately conical. At a chord length of 85%, where the sharp leading edge connects to the tip, the primary vortex and related shock have a sharp kink, see Figures 3 and 6. Two shocks develop in the primary vortex. One normal to the leading edge and connected to the kink in the shock structure under the primary vortex and another one from the same location on the leading edge and connected more upstream to the shock under the primary vortex. A similar shock structure, although slightly more downstream, was observed by Hoeijmakers et al. [8] using a much finer structured grid. This shock structure has a strong influence on the primary vortex, which completely blows up behind it, see Figure 5, and is very well captured by the grid adaptation. Also visible in Figure 5 is that the grid is adapted to the trailing edge vortex. The primary vortex significantly grows after 85% chord and merges with the tip vortex, see Figure 4. Also visible is the start of roll-up of the wake, which develops into a mushroom type vortex structure. In addition to the shock structures in and around the primary vortex there is also a shock starting at about 75% downstream at the center line and connected to the trailing edge at approximately mid span. A better view of this shock can be obtained in Figure 6 which gives a perspective view of the delta wing and the grid and flow field at approximately 70% chord. Figure 5 clearly shows the strong primary vortex and the shock between the vortex and body. Also visible is the significant refinement in this region and the vortex layer starting at the sharp leading edge.

PARALLELIZATION

The above described algorithm has been implemented in the program Hexadap, which is parallelized on shared memory machines, namely:

- A two processor NEC SX-3/22 with a peak performance of 2×2.75 GFlop/s, a main memory unit (MMU) of 1 GByte and 4 GByte Extended Memory Unit (XMU) of which 1.2 GByte can be efficiently used to store run-time data,

	Small	Medium	Large	Long
vector length	8000	2000	1000	120,000
iterations	100	100	300	100
adaptations	0	1	1	0

Table 2: *Problem sizes*

- A four processor SGI Power Challenge with a peak performance of 4×350 MFlop/s, main memory of 256 MByte and 16 KByte primary and 4 MByte secondary cache.

The parallelization uses microtasking, adding parallelization compiler directives, for both machines, and macrotasking, explicitly assigning tasks to different processors. (Implementation on the SGI Power Challenge is done with the CONCURRENT CALL assertion). The advantage of microtasking is that the code remains portable. The advantage of macrotaking is that large tasks can be assigned, even if the tasks have no do-loop structure, and memory can be used more efficiently.

The above described algorithm consists of two parts, namely grid adaptation and flow computation. The grid adaptation part, which consists predominantly of scalar operations, requires a domain decomposition for parallelization and is not considered in this paper. The flow computation has as most important component the calculation of cell face fluxes and consists of loops over the cell faces. The result is added to the residual in the two cells connected to each cell face. The loops use indirect addressing and in order to vectorize these loops a coloring scheme has been applied.

The initial flow field and the flow field after three and six adaptations is used to test the parallel performance of the flow solution algorithm, see Table 1. These cases are denoted Small, Medium and Large. The average vector length and number of iterations are presented in Table 2, which shows that the average vector length decreases with problem size. This is caused by the increasing number of colors after grid adaptation. A reduction in the number of colors is possible by limiting the number of neighboring cells connected to each cell face. In order to investigate the dependence of the performance results on the vector length a special case, labeled Long, is also tested, see Table 2.

NEC SX-3

The two computationally most intensive parts of the flow solution algorithm are the routines Limit and Flux. Limit applies a slope limiter to ensure monotonicity and Flux computes the fluxes through cell faces. The suffixes 1G and 4G in Tables 3 and 4 refer to the number of Gauss quadrature points used in the evaluation of the flux integrals. The two routines constitute 90% of the total computing time. They have roughly the same structure: a nested loop, first over all colors and then over all faces of one color.

MFlop rates on a single processor NEC SX-3 are reported in Table 3. The rates are based on flop counts and elapsed times. The decrease in overall performance for the Medium and Large problems is caused by the larger number of colors after grid adaptation which results in a reduced vector length. The case Long does not suffer from this reduction in performance. Also indicated in Table 2 is if the grid is adapted.

	Flux	Limit	Total
Small(4G)	474	392	426
Medium(1G)	406	258	232
Large(1G)	371	241	265
Long(4G)	484	445	452
Long(1G)	463	318	314

Table 3: *Mega flop rates on single processor NEC SX-3 (based on elapsed times)*

Two parallelization strategies have been tested. The first strategy executes the loops over the colors in parallel and vectorizes the inner loop over the faces. Part of the inner loop over the faces consists of an update of the residuals at the cell centers. Within one color all faces connect to cells with different cell addresses, but this is not assured between different colors, causing a data dependency. Hence, in the above parallelization strategy, the residual updates have to be performed in a critical section, where only one processor is active at a time. The second strategy divides the loop over the faces within one color over the available processors. The main problem with this approach is that sufficient vector length should remain after loop division.

The MFlop rates and speedup results are presented in Table 4. The timings and speedups are influenced by the use of the external memory unit XMU of the SX-3. The XMU allows for fast access to data which cannot be placed in core memory. Sequentially, the use of the XMU instead of core memory hardly decreases performance. During parallel execution, however, locks applied during I/O seriously deterioriate the performance. If we compensate for the time spent during I/O to the XMU speedups increase, the corrected speedups are labeled Corr in Table 4. The MFlop rates in Table 4 are based on the corrected speedups.

The results for the first parallelization strategy, namely parallel execution of loops over the colors, are obtained using micro tasking and are labeled 'C' in Table 4. The speedups are with respect to elapsed times. It is clear from the results that the efficiency of the parallelization is rather low. This has two reasons. First, the critical section consumes 20% of the computing time, and second, the parallel system overhead is about 10%. This large sequential part limits the maximum attainable speedup on more processors to 5.

The second parallelization strategy, namely parallel execution of loops over the faces within one color, does not suffer from a critical section. At first the code was parallelized using micro-tasking. The program structure is such that the flux computation is split into many different loops in different functional subroutines. Therefore the computational load per loop is low, less than 1.5 msec. It turned out that this load is too low to be efficient on the NEC SX-3: the parallel overhead was as large as, or even larger than the parallel gain and no speedup was obtained.

Using macrotasking the parallel overhead could be reduced significantly. Instead of parallelizing each loop separately, the work is divided into two tasks in the subroutines Flux and Limit, each task doing the same job as the subroutines, but on only half the loop. This not only reduced the parallel system overhead, but also reduced memory use. In microtasking local data is copied

		Flux	Corr	Limit	Total	Corr	MFlop/s
SX-3	C	1.5	1.6	1.6	1.4	1.5	624
Small(4G)	F	1.6	1.6	1.7	1.3	1.3	566
SX-3	C	1.5	1.8	1.2	1.5	1.6	364
Medium(1G)	F	1.3	1.6	1.4	1.5	1.6	376
SX-3	C	1.4	1.8	1.2	1.2	1.3	356
Large(1G)	F	1.1	1.4	1.2	1.1	1.2	322
SX-3	C	1.5	1.5	1.4	1.3	1.4	614
Long(4G)	F	1.7	1.7	1.6	1.5	1.6	701
SX-3	C	1.5	1.8	1.3	1.3	1.4	440
Long(1G)	F	1.6	1.9	1.6	1.5	1.6	495
SGI	LL	2.9	-	3.3	2.1	-	85
Small(4G)	F	3.7	-	2.9	2.3	-	94
SGI	LL	1.5	-	2.0	1.5	-	37
Medium(1G)	F	2.9	-	2.3	2.0	-	51

Table 4: *Speedups relative to single processor performance (based on elapsed times); SX-3 two processors; SGI four processors; C: parallel loop over colors (microtasking); F: parallel loop over faces within one color (macrotasking); LL: Low level microtasking*

for each processor, in macrotasking the local data can be defined per task, and thus approximately halved with respect to the sequential program. Memory use for the medium sized problem is 498 MByte for the sequential program, 540 MByte for the microtasked program and 515 MByte for the macrotasked program. Speedups for the macrotasked program are presented in Table 4 and labeled 'F'.

The decrease in parallel performance with increased problem size can be attributed to the reduced vector length. This is clearly demonstrated by the results of test case Long, which has an average vector length of 120000 in the loops over the cell faces. This problem reaches the highest parallel performance, with a speed-up of 1.9 in routine Flux. Another factor which significantly reduces the performance of the flow solution algorithm on a NEC SX-3 computer is the limited memory bandwidth. This is especially important for the large number of indirectly addressed loops and a main reason for the big gap between sustained and peak performance. The memory bandwidth limitations are the most evident in Limit, where the ratio between computations and load/stores is rather low.

SGI Power Challenge

The SGI Power Challenge has scalar processors and therefore no problems with data dependencies within a processor. The code was therefore parallelized using the second parallelization strategy, namely parallel execution of the loops over the cell faces. Only the Small and Medium problems were tested, since the other problems did not fit in memory.

Two implementations are made, one by parallelizing each loop separately (low-level), and one using the same macrotasking structure as described in the previous section. Parallelization is straightforward using the parallel code of the SX-3. Directives are changed to SGI directives. The macrotasking is accomplished using the CONCURRENT CALL assertion.

Results of speedups and MFlop rates are presented in Table 4. The low-level parallelization is labeled 'LL' and the macrotask-

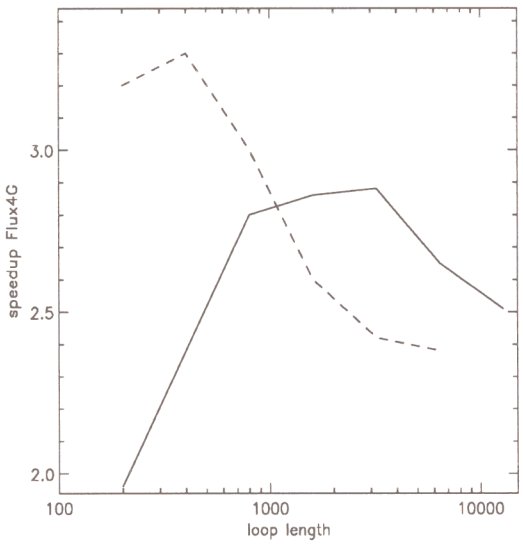

Figure 2: *Cache dependency of speedups on the SGI Power Challenge in routine Flux(4G) (— Small - - - - Medium)*

ing results are labeled 'F'. Since the SGI has no XMU there is no correction for the speedups: the entire program is run in core memory. The speedups for macrotasking are better than for the low level parallelization. The performance in MFlops of the SGI four processor Power Challenge, as listed in Table 4, is between 10% and 17% of the two processor SX-3 performance and not sufficient for large scale computing. The percentage of peak performance is between 3% and 7% on the SGI Power Challenge and between 6% and 13% on the two processor NEC SX-3.

Results of the SGI Power Challenge are rather sensitive to cache misses. A parameter in the flow solution algorithm determines the number of cell faces in the flux calculation processed at one time. Varying this parameter changes the amount of the data being processed, and can be used to optimize the cache use of the program. Significant differences can occur, and the optimal value of the parameter depends on the problem at hand. (see Figure 2). The speedups of Table 4 are computed using the optimal timing results.

Estimate of the computing time for a LES of a clean wing on a NEC SX-4/16 computer

The parallel performance on the NLR NEC SX-3/22 has been used to estimate the problem size of a Large Eddy Simulation of a clean wing on a 16 processor NEC SX-4, which will be delivered to NLR in 1996. The NLR NEC SX-4/16 is expected to have a peak performance of 32 Gflop/s, a main memory of 4 GByte and 8 GByte XMU. With respect to the SX-3/22 its architecture is more suited for indirect addressing and a single processor speedup of 2 is expected for programs using indirect addressing.

The size of the LES is primarily determined by the available memory. Let N be the number of grid points, and n the number of flow variables. For a Large Eddy Simulation with a one-equation turbulence model we have $n = 6$. The memory use of Hexadap is $8(12n + 40)N + 2 \cdot 10^8$ Byte. With an available memory of 8 Gbyte and 8 bytes per variable the maximum number of grid points $N = 9 \cdot 10^6$. Using the estimates given by Chapman [3], this number allows for a LES with sublayer resolution around a clean wing at a Reynolds number of approximately 10^6.

The computing time for one time step is estimated from the relation:

$$t_{cpu} = \frac{n \cdot N}{S_A \cdot S_C} \cdot \left(\frac{f_S}{r_S} + \frac{1}{S_{16}} \left(\frac{1.3 \cdot 1.1 \cdot f_F}{r_F} + \frac{f_L}{r_L} + \frac{f_R}{r_R} \right) \right)$$

with: S_A a factor to account for grid adaptation, $S_A = 0.9$, S_C the single processor speedup of the NEC SX-4 compared to the SX-3, $S_C = 2$. The suffixes S, F, L and R refer to the following parts of the algorithm: S, serial part, F, subroutine Flux(1G), L, subroutine Limit, and R the remaining part of the flow solution algorithm which is parallelizable. The variables f_* denote flop counts in the respective parts of the algorithm to advance one flow variable one time step in one grid point. The measured values are: $f_S = 90$, $f_F = 1570$, $f_L = 880$ and $f_R = 180$. The variables r_* denote the measured flop rates in the respective parts of the algorithm and are equal to: $r_S = r_R = 350 \cdot 10^6$, $r_F = 463 \cdot 10^6$ and $r_L = 350 \cdot 10^6$ flop/s. The flop count in routine Flux is increased with 10% for the viscous contribution and 30% for a one-equation subgrid model using the Germano approach. The parallel speedup, denoted by S_{16} on a 16 processor NEC SX-4 is estimated as twelve. The computing time required to advance one time step on a grid with $9 \cdot 10^6$ grid points is then approximately 28 seconds.

The time scale of the smallest eddies in the flow field will be approximately 100 times larger than the CFL limit for an explicit scheme. The CFL time step limitation can be removed with an implicit, time accurate temporal discretization using multigrid acceleration. With these assumptions a Large Eddy Simulation of a clean wing at a Reynolds number 10^6 on a mesh with $9 \cdot 10^6$ grid points which evolves 6500 time steps, which should be sufficient to obtain a reasonable statistical sample, would require 50 hours on a 16 processor NEC SX-4.

Conclusions of the parallelization

Provided that the vector length is sufficient, the most efficient parallelization strategy for the present flow solution algorithm is a high level parallelization of loops over faces of one color using macrotasking. Macrotasking reduces parallel system overhead and memory use. Correcting for the XMU a maximum speedup of 1.9 is reached on a two processor SX-3.

There are three causes for the not perfect overall performance on the NEC SX-3:

- I/O between Main Memory and XMU in parallel processing takes significantly more time,
- Vector length decreases, and hence single processor speed,
- Parallel system overhead.

Concerning the latter cause, the balance between the two processors is, when corrected for the I/O between MMU and XMU, as predicted by the size of the parallel part of the algorithm. Hence, the computational load is well balanced, and the remaining performance loss can only be explained by parallel system overhead. Since the NEC SX-3 is not primarily suited for par-

allel use, the relatively high parallel system overhead is not too surprising. It is expected that the NEC SX-4 has significantly less overhead.

Low-level do-loop parallelization on the NEC SX-3 turns out to be only sufficient for loops with a computational load greater than 1.5 msec.

The parallel efficiency on the SGI Power Challenge is similar, the percentage of peak performance is relatively low, even compared with the NEC SX-3. Moreover, the cache sensitivity makes the optimization problem dependent.

The present parallelization on the NEC SX-3 will not be sufficiently efficient on the 16 processor SX-4. The parallel execution of the loops over the cell faces is inefficient since the loop length will be too short to be divided over 16 processors. This problem can be solved by limiting the number of neighboring cells connected to one cell face to at most four, which significantly reduces the number of colors and thereby increases vector length. The parallel execution of the loop over the colors contains a sequential part of 20%, and hence has a maximum speedup of 5. This sequential part can be eliminated using a domain decomposition of the grid, which also has as main benefit that the grid adaptation part can be executed in parallel.

CONCLUDING REMARKS

The discontinuous Galerkin finite element method with local grid enrichment has been demonstrated on the three-dimensional, inviscid flow field around a delta wing at transonic speed. The use of anisotropic grid refinement of hexahedron type cells is effective in capturing the shock structure and primary vortex on the leeward side of the delta wing. The discontinuous Galerkin method works well on highly irregular grids and is therefore a good candidate for Large Eddy Simulations, because it offers the opportunity to capture viscous sublayers with successively finer grids through local grid refinement. An estimate of the required computational resources for such a simulation is presented. The use of a face based data structure works well in combination with local grid refinement and allows efficient vectorization and parallelization of the code. On the NEC SX-3 the possible speedup through parallelization strongly depends on the vector length. A maximum speed-up of 1.9 on the two processor NEC SX-3 is obtained when sufficient vector length was available. A good parallel performance, with a speed-up of 3.7, is obtained on the four processor SGI Power Challenge, but the results are sensitive to cache misses.

From the present results it is estimated that for future LES applications in wall bounded flows, the gain from the increased computational efficiency obtained from highly adapted grids more than compensates the increased number of operations and memory use. A LES of a clean wing at a Reynolds number of 10^6 will become feasible on a 16 processor NEC SX-4 in a turnaround time of one weekend. Significant further developments, such as the addition of the viscous contribution and implicit time-accurate temporal discretization using multigrid acceleration (in progress), will, however, be needed to reach this goal.

REFERENCES

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

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

[3] Chapman, D.R. Computational aerodynamics development and outlook. AIAA Paper 79-0129, 1979.

[4] 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.

[5] 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.

[6] 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.

[7] Elsenaar, A., Hjelmberg, L., Bütefisch, K.A. and Bannink, W.J. The international vortex flow experiment. AGARD Symposium on Validation of Computational Fluid Dynamics, Lisbon, AGARD CP 437, 1987, also AGARD Advisory Report 303, 1994.

[8] Hoeijmakers, H.W.M., Jacobs, J.M.J.W. and Van Den Berg, J.I. Numerical simulation of vortical flow over a delta wing at subsonic and transonic speed. Presented at 17th ICAS Congress, 1990, Stockholm,Sweden, 1990.

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

[10] Moin, P. and Jimenez, J. Large eddy simulation of complex turbulent flows. AIAA Paper 93-3099, 1993.

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

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

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

[14] Van Der Vegt, J.J.W. Anisotropic grid refinement using an unstructured discontinuous Galerkin method for the three-dimensional Euler equations of gas dynamics. AIAA Paper 95-1657, 1995.

[15] Venkatakrishnan, V. Convergence to steady state solutions of the Euler equations on unstructured grids with limiters. *JCP*, 118:120–130, 1995.

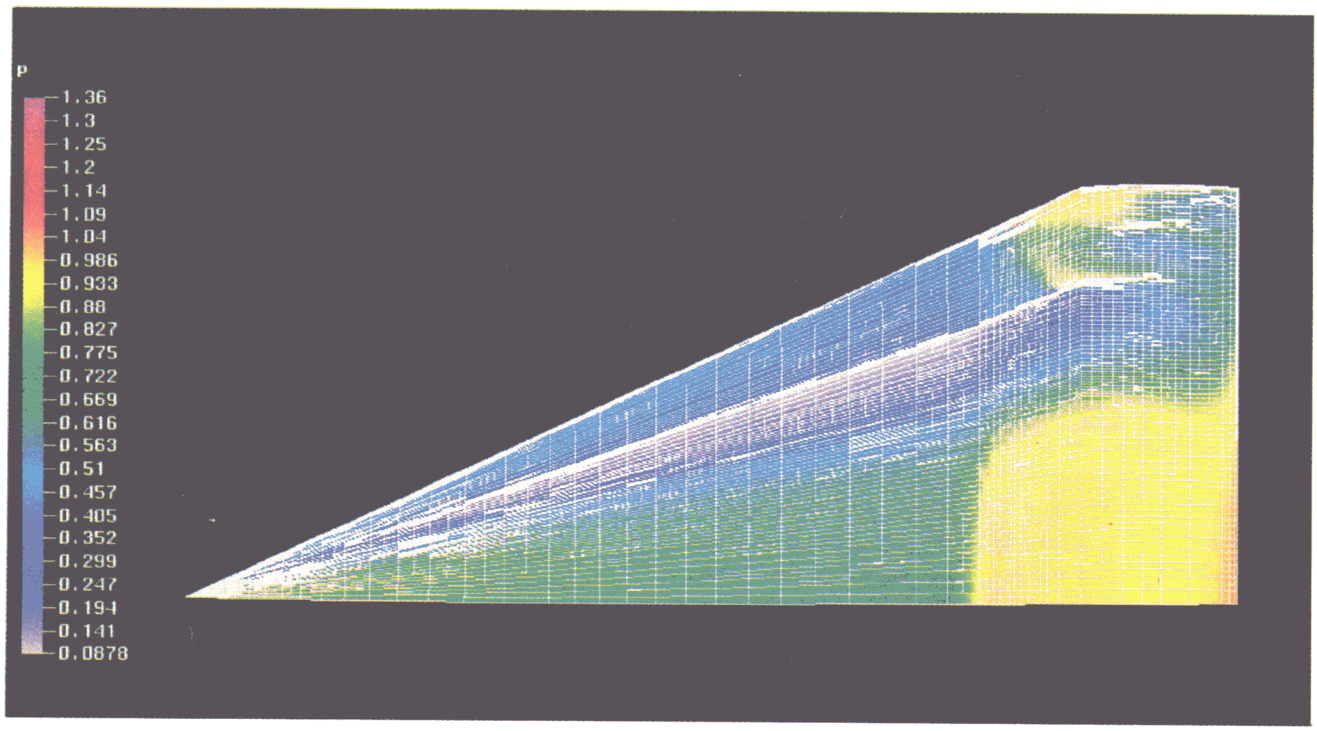

Figure 3. *Pressure field and adapted grid on upper surface of delta wing.* ($M_\infty = 0.85, \alpha = 20°$)

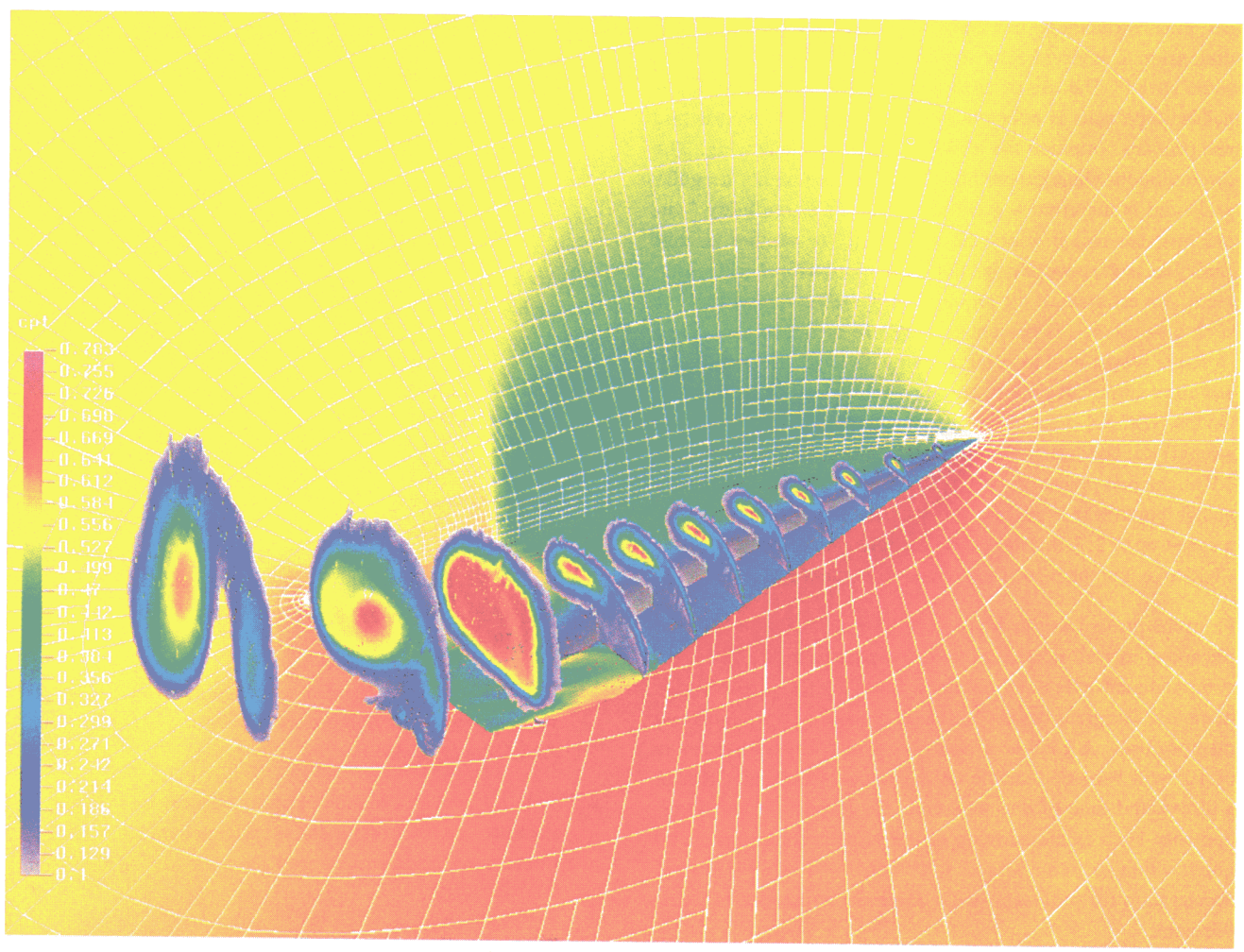

Figure 4. *Vortex structure on leeward side of delta wing, visualized as total pressure loss.* ($M_\infty = 0.85, \alpha = 20°$)

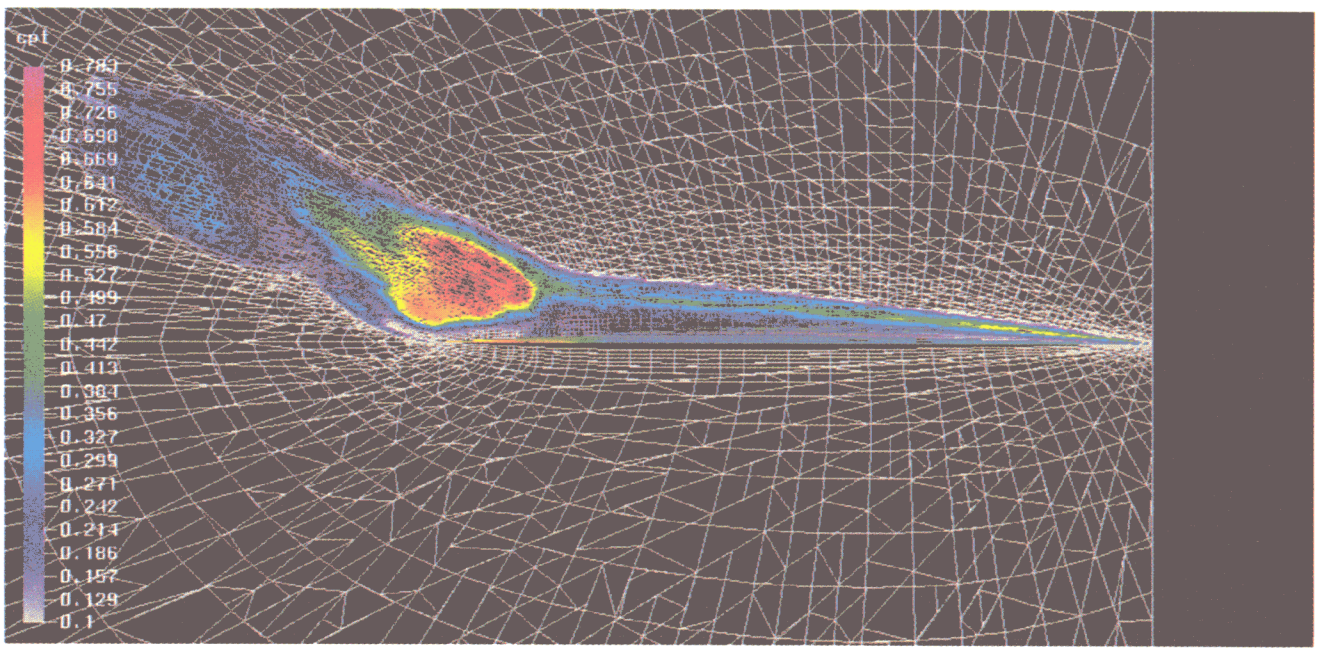

Figure 5. *Total pressure loss and adapted grid in cross-section through primary vortex core.* ($M_\infty = 0.85, \alpha = 20°$)

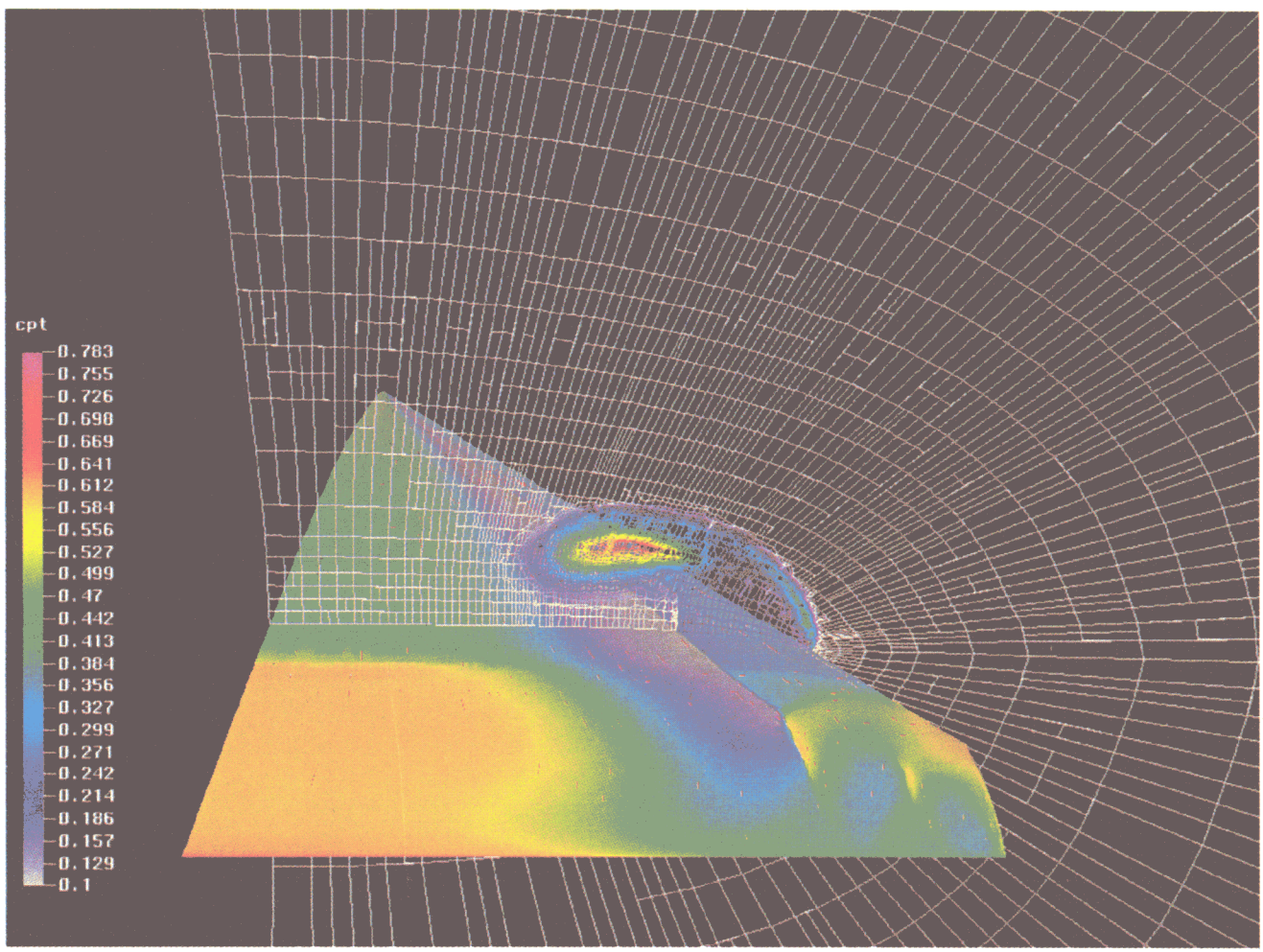

Figure 6. *Total pressure loss and adapted grid in cross-section at 70% chord.* ($M_\infty = 0.85, \alpha = 20°$)