INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 25, 475–491 (1997)

ANISOTROPIC UNSTRUCTURED MESH ADAPTION FOR FLOW SIMULATIONS

M. J. CASTRO-DÍAZ,[1] F. HECHT,[2] B. MOHAMMADI[2]* AND O. PIRONNEAU[2]

[1]*University of Malaga, Malaga, Spain*
[2]*INRIA and University of Paris 6, Paris, France*

SUMMARY

Three new ideas for anisotropic adaption of unstructured triangular grids are presented, with particular emphasis on fluid flow computations. © 1997 by John Wiley & Sons, Ltd.

Int. J. Numer. Meth. Fluids, **25**: 475–491 (1997).

No. of Figures: 12. No. of Tables: 0. No. of References: 15.

KEY WORDS: mesh; anisotropic; adaptation; Delaunay; metric; CFD

1. INTRODUCTION

From finite element theory it is well known that for isotropic problems, meshes with equilateral triangles are more suitable. However, the notion of equilaterality involves lengths through scalar products in a given metric. Therefore anisotropic meshes might be seen as isotropic with respect to a different metric. Now, if we define this metric by an *a posteriori* error estimation, we can adapt the mesh to follow the solution.

Of course, an unstructured grid environment is the natural framework for the introduction of general adaptivity and anisotropy concepts.[1–3] However, our experience shows that in adaption procedures based on local metric changes, three major difficulties remain.

1. As the metric is defined from the interpolation error of some quantity, the extension to systems of PDEs is not clear, especially when phenomena of different types interact (e.g. in shock–boundary layer interactions).
2. Boundary layers are not correctly resolved and wall coefficients (especially the friction coefficient) obtained over adapted meshes are useless. This is due to the fact that the distance from the first-layer points to the wall is not uniform. This is one of the major weakness of anisotropic adaption in CFD.
3. Multiscale phenomena are hard to capture by adaption (e.g. small eddies in a turbulent flow).

In this paper we present three ideas to remove the above difficulties.

*Correspondence to: B. Mohammadi, INRIA-MENUSIN, Domaine de Voluceau, F-78153 Le Chesnay, France.

CCC 0271–2091/97/040475–17 $17.50 *Received July 1996*

The first idea is the intersection of the different metrics obtained for the different variables of a PDE, which in our case are the conservation variables. This also removes the difficulty we had until now in choosing the 'right' variable for building the metric from its interpolation error. Indeed, we used to choose e.g. the pressure or the density for Euler computations and e.g. the local Mach number or the entropy for viscous computations. Of course, this is a dead-end, as none of these choices could give satisfaction because one variable cannot encapsulate all the physics of the system.

The second idea is motivated by two requirements coming in fact from structured meshes, which are known to be more suitable near walls: (i) we want the meshes to be as orthogonal as possible near the body; (ii) we want each node layer to be at a uniform distance from the wall. The idea consists of a modification of our local metric near the wall. To enforce the orthogonality of the mesh in this area, we change the eigenvectors of our metric. The second requirement is satisfied by giving the user the possibility of prescribing the normal size of the elements along the wall and propagating this through the domain by a relaxation procedure to obtain a quasi-orthogonal mesh in near-wall regions with each node layer at a uniform distance from the wall.

The third idea helps the capture of multiscale phenomena by introducing a relative rather than a global definition of the metric. This also removes the dimensional incompatibilities between the metrics coming from different variables when we use the first idea.

These ideas can be extended to three-dimensional cases. In fact, to generate 2D meshes, we have used our 3D surface grid generator after setting z to zero. The generation procedure is fast (an average of 300,000 triangles per minute on a 10 Mflops workstation).

Our adaption loop can be defined as follows.

```
Start from an initial mesh.
Adaptation loop.
(a) Solve the PDE (in this case the fluid dynamics system).
(b) Build a metric from the multivariable solution obtained in (a).
(c) Build an equilateral mesh with respect to the metric of (b) using a Delaunay-type mesh
    generator.
End of loop.
```

We will discuss in particular point (b) where we introduce the idea presented above. We also give a brief description of the fluid solver and the mesh generator.

2. NAVIER–STOKES SOLVER

We use the NSC2KE fluid solver for these computations. More details can be found in Reference 4. A finite volume Galerkin formulation of the Navier–Stokes equations in conservation form has been considered. A four-stage Runge–Kutta scheme is used for time integration. The Roe Riemann solver[5] has been used for the Euler part together with a MUSCL-type reconstruction and van Albada limiters[6] for second-order accuracy. The P^1 finite element has been used for the viscous part of the operators. A Steger–Warming[7] flux-splitting procedure has been used at the inflow and outflow boundaries, while no-penetration or no-slip boundary conditions are applied to solid walls depending on the nature of the flow. Turbulent modelling is done using the classical k–ε model[8] with special wall laws enabling separated and unsteady flow computation.[9] This solver is in free access (anonymous ftp piranha.inria.fr under pub/).

3. MESH GENERATION

We give a brief description of our Delaunay-type mesh generator; more details can be found in References 10 and 11. For mesh adaption, two strategies are possible: local optimization of the mesh or global regeneration at each adaption.[1,2] In this work we use the first approach. Both approach have advantages and disadvantages from the implementation point of view.

Once the metric is given (see below for the metric definition), the mesh generation procedure is applied. This algorithm uses five different local tools: *edge suppression, vertex suppression, vertex addition, edge swapping* and *vertex reallocation* (*barycentring step*). We need three different grids: one defining the domain geometry (*'CAD mesh'*, $\mathscr{T}_G$) which is fixed during the mesh adaption loop, one that contains the control space information (i.e. the metric tensor definition) denoted $\mathscr{T}_0$ and the adapted mesh $\mathscr{T}_1$. We can say that this algorithm solves the following problem: *find 'an optimal mesh' $\mathscr{T}_1$ adapted from $\mathscr{T}_0$ with respect to the criterion imposed by the metric tensor $\mathscr{M}$ and compatible with $\mathscr{T}_G$.*

Thus we suppose that the meshes $\mathscr{T}_G$ and $\mathscr{T}_0$ are given together with the metric tensor $\mathscr{M}$ obtained from a finite element solution over $\mathscr{T}_0$. Initially we have $\mathscr{T}_1 = \mathscr{T}_0$, but $\mathscr{T}_G$ can of course be different from $\mathscr{T}_0$.

Different steps can be distinguished in our mesh adaption procedure.

Data structure creation and verification

Prior to grid adaption it is necessary to create some auxiliary data: mesh connectivity arrays, boundary and inter-subdomain (geometrical and physical)* edges with their associated tangent vectors, fixed point localization, etc. A topological verification step together with a triangle orientation is made in order to guarantee that the given meshes are correct and oriented.

Geometry reconstruction

The generalized Farin algorithm[10] is used in order to define a 'G^1' Bézier surface over $\mathscr{T}_G$.

Initial regularization

Depending on user choices, an initial optimization step (edge swapping) is made in order to improve the initial mesh quality. Numerical experiences show that the final result is better if this initial optimization is made.

Grid adaption

The triangles of $\mathscr{T}_1$ are ordered in a particular data structure called a *double dynamic list* (*DDL*). Three double lists are considered (vertices, edges and triangles). Suppose that the first $i-1$ edges a_j, $j=1,\dots,i-1$, are treated; then the edge a_i is taken. The procedure that we propose is the following one.

1. Let d_i be the length of a_i computed with the metric tensor $\mathscr{M}$. We have three possibilities.
   (a) If $d_i > l_{\max}$ ($l_{\max} \approx 1{\cdot}4L_{\mathrm{ref}}$), then a_i must be cut into two edges using the add-vertex procedure. The new mesh elements (vertices, edges and triangles) are added at the end of

*The first ones correspond to intersections of regular surfaces and the second ones correspond to intersections of user-defined subdomains.

their corresponding DDL. Now a_i has changed and we again compute its length, d_i. If $d_i > l_{\max}$, then we repeat the same process until its associated length $d_i \leqslant l_{\max}$.

(b) If $d_i < l_{\min}$ ($l_{\min} \approx 0{\cdot}6 L_{\text{ref}}$), then a_i is suppressed, leading to the creation of new edges. We check whether their lengths are bigger than $l_{\max}$, in which case the previous process is applied, or smaller than $l_{\min}$, in which case this process is repeated.

(c) If $l_{\min} \leqslant d_i \leqslant l_{\max}$, a_i is kept.

2. If we have done a local mesh modification, a local optimization step must be considered. In this case, edge swapping is applied to the new or modified edges.

This process is repeated over all the edges.

Final optimization

Finally a global optimization process is considered. All non-prescribed interior vertices with less than four elements connected to them are suppressed and a barycentring step is applied: all non-fixed vertices are moved to the centre of gravity of their neighbours with respect to the metric $\mathscr{M}$.

Thus a final mesh $\mathscr{T}_1$ adapted from $\mathscr{T}_0$ with respect to the metric tensor $\mathscr{M}$ is obtained and it satisfies the geometrical and topological constraints. After each adaption step an initial solution is obtained over the new mesh by interpolation of the solution on the previous mesh. A good interpolation is crucial for unsteady computations.

Remarks on mesh generation

1. The remeshing algorithm always works locally. This strategy allows us to remesh 3D surfaces with only a local parametrization.
2. Being based on a Delaunay algorithm, this technique completely avoids the possibility of any overlapping in the mesh from the moment that the metric is well defined (i.e. definite positive).

4. METRIC COMPUTATIONS

During automatic mesh generation it is necessary to have as much information as possible on the nature and local behaviour of the solution. These indications constitute the *control space* which governs the grid generation. As we are interested in information on the element shapes and sizes, we need to convert somehow what we know about the solution to something having the dimension of a length and containing directional information. This might be done by giving at each point $x \in \Omega \subset \mathbb{R}^2$ three parameters (six in $\mathbb{R}^3$). This kind of control is equivalent to an isotropic control with a change in the metric tensor all over the domain.*

It is easy to prove that if K_0 is a non-degenerate triangle, then there exists a unique metric where K_0 is equilateral with unity edge lengths. Thus, in order to obtain triangles with a given stretching and size over a subdomain, it is sufficient to construct an isotropic mesh using the metric tensor $\mathscr{M}$ with an associated refinement function h equal to unity everywhere. $\mathscr{M}$ is given at every point $x \in \Omega \subset \mathbb{R}^2$ by

$$\mathscr{M}(x) = \mathscr{R}(x) \begin{pmatrix} \gamma_1(x) & 0 \\ 0 & \gamma_2(x) \end{pmatrix} \mathscr{R}(x)^{-1}, \tag{1}$$

where $\gamma_1(x) > 0$ and $\gamma_2(x) > 0$ are the $\mathscr{M}(x)$ eigenvalues and $\mathscr{R}(x)$ is the rotation matrix of angle $\alpha(x)$ that maps the $\mathbb{R}^2$ canonical basis over the $\mathscr{M}(x)$ unit eigenvectors.

*For the sake of simplicity we will consider in this section that $\Omega \subset \mathbb{R}^2$. We obtain the same results if Ω defines a surface or if $\Omega \subset \mathbb{R}^3$. In the last case, 'triangles' are replaced by 'tetrahedran'.

Elementary differential geometry says that the length of a parametric curve $\Gamma(t)$, where $t \in [0, 1l$, in the new metric is defined by

$$L(\Gamma) = \int_0^1 \sqrt{[\Gamma'(t)^{\mathrm{T}} \mathcal{M}(\Gamma(t))\Gamma'(t)]}\mathrm{d}t. \tag{2}$$

We propose the following discretization of the metric tensor $\mathcal{M}$. Let $\mathcal{M}$ be given at vertices $\{\mathcal{M}(x_i),\ i = 1, \ldots, n_{\mathrm{v}}\}$; then over each mesh triangle K_0 the $\mathcal{M}$ coefficients are linearly interpolated and a continuous discretization of the metric tensor $\mathcal{M}$ is obtained.

Let $\Gamma = [x_0, x_1]$ be a segment, $\Gamma \subset K_0$. Computing its length using equation (2), we obtain

$$L(\Gamma) = \int_0^1 \sqrt{[(\Gamma')^{\mathrm{T}} \mathcal{M}(\Gamma(t))\Gamma']}\mathrm{d}t \tag{3}$$

$$= \int_0^1 \sqrt{[l_0^2 + t(l_1^2 - l_0^2]}\mathrm{d}t \tag{4}$$

$$= \frac{2}{3}\frac{l_0^2 + l_0 l_1 + l_1^2}{l_0 + l_1}, \tag{5}$$

where $l_i = \sqrt{[(\Gamma')^{\mathrm{T}} \mathcal{M}(x_i)\Gamma']}$, $i = 0, 1$.

We now discuss how the metric tensor $\mathcal{M}$ is defined in order to satisfy an adaption criterion. Suppose that we only work with one variable denoted η. We are going to determine the metric tensor in order to equilibrate the interpolation error.

Assume that a first solution has been computed over a given mesh and let η be continuous piecewise linear interpolated. Then[12,13] the interpolation error depends on the Hessian matrix of η:

$$\mathcal{E} = |\eta - \Pi_h \eta|_0 \leqslant c_0 h^2 |\mathcal{H}(\eta)|_0, \tag{6}$$

where $\pi_{\mathrm{Ih}}\eta$ is the P^1 interpolation of η,

$$\mathcal{H} = \begin{pmatrix} \partial^2\eta/\partial x^2 & \partial^2\eta/\partial x\partial y \\ \partial^2\eta/\partial x\partial y & \partial^2\eta/\partial y^2 \end{pmatrix} = \mathcal{R}\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\mathcal{R}^{-1} \tag{7}$$

and the metric tensor $\mathcal{M}$ is defined by

$$\mathcal{M} = \mathcal{R}\begin{pmatrix} |\lambda_1| & 0 \\ 0 & |\lambda_2| \end{pmatrix}\mathcal{R}^{-1}. \tag{8}$$

The error over a mesh edge a_i can be computed as

$$E_i \approx |a_i^{\mathrm{T}} H a_i| \geqslant \sqrt{[c_0 a_i^{\mathrm{T}} M a_i]}. \tag{9}$$

Now the mesh is equilateral and of edge length $\sqrt{c_0}$ in the metric defined by $\mathcal{M}$.

We can associate an ellipse $\mathcal{E}_{\mathcal{M}}$ with the metric tensor $\mathcal{M}$ having axes d_1 and d_2 (eigenvectors of $\mathcal{M}$) of lengths $1/\lambda_1$ and $1/\lambda_2$ respectively.

Remarks on metric computations

1. In the metric definition we have to introduce the maximum and minimum edge lengths in the mesh to avoid unrealistic metrics. This is not really a restriction as usually we have a good idea of what these quantities should be. More precisely, the eigenvalues of the metric are limited as follows:

$$\tilde{\lambda}_{1,2} = \min\left(\max\left(|\lambda_{1,2}|, \frac{1}{h_{\max}^2}\right), \frac{1}{h_{\min}^2}\right),$$

with $h_{\min}$ and $h_{\max}$ being the minimal and maximal edge lengths allowed in the mesh.

2. The key point in the metric definition is the second derivatives. However, in our application the solver is a P^1 finite element solver. Therefore a weak formulation (by Green's formula) has to be used to compute the Hessians.

4.1. Extension to systems

Suppose now that several variables $\eta_1, \eta_2, \ldots, \eta_r$ are given. The problem becomes: *find the metric so that the maximum interpolation error is minimized for all the variables*. It is clear from the geometrical identification ellipse–metric that the solution to the previous minimization problem is to find the biggest ellipse contained in the intersection of all the ellipses $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_r$ corresponding to the metrics $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_r$ computed from the variables $\eta_1, \eta_2, \ldots, \eta_r$. It is not easy in general to find the optimal solution of this problem. However, the following algorithm seems to be suitable enough (see Figure 1). Suppose that only two variables η_1 and η_2 are provided. We find an approximation of the optimal intersection ellipse by the following procedure.

1. If the two ellipses $\mathcal{E}_1$ and $\mathcal{E}_2$ do not intersect (this happens when one ellipse is contained the other), the one with smallest area is taken as intersection and the suitable metric is the associated one.
2. Otherwise, let λ_i^j and v_i^j, $i, j = 1, 2$, be the eigenvalues and eigenvectors respectively of $\mathcal{M}_j$, $j = 1, 2$. The intersection metric $\hat{\mathcal{M}}$ is defined by

$$\hat{\mathcal{M}} = \frac{\hat{\mathcal{M}}_1 + \hat{\mathcal{M}}_2}{2}, \tag{10}$$

where $\hat{\mathcal{M}}_1$ (resp. $\hat{\mathcal{M}}_2$) has the same eigenvectors as $\mathcal{M}_1$, (v_1^1, v_2^1) and eigenvalues

$$\tilde{\lambda}_i^1 = \max(\lambda_i^1, v_{i\mathrm{T}}^1 \quad \mathcal{M}_2 v_i^1), \qquad i = 1, 2. \tag{11}$$

Now, if n variables are given, the final intersection metric is computed as follows.

3. $\hat{\mathcal{M}} =$ intersection $(\mathcal{M}_1, \mathcal{M}_2)$.
4. For $i = 3, \ldots, n$, $\hat{\mathcal{M}} =$ intersection $(\hat{\mathcal{M}}, \mathcal{M}_i)$.

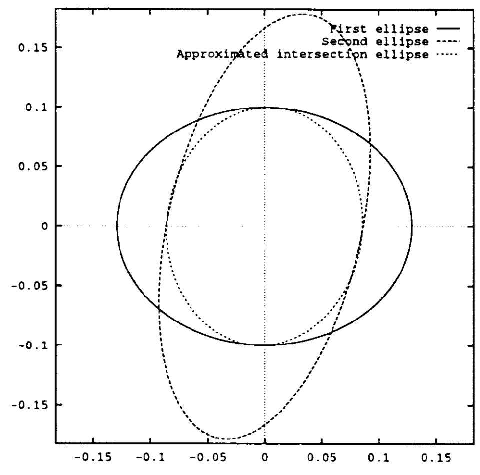


Figure 1. Approximated optimal ellipse of two-ellipse intersection

Here a dimensional problem appears owing to the fact that the different metrics are based on different variables having different dimensions. We will see that a relative rather than the global error estimation (6) permits us to avoid this problem.

4.2. Boundary layer improvement

As stated previously, metric evaluation is done by Green's formula with homogeneous Neumann boundary condition on all boundaries. This is quite logical and means that elements do not change in size and shape near boundaries. However, it also means that information on normal mesh sizes is not available at the walls and as a result the sizes are generally overestimated. Our experience shows that this is not suitable for viscous computation in boundary layers. This is why structured meshes are usually more suitable for these regions.

We have tried to give a new boundary condition for the metric on the walls to improve the mesh definition in the near-wall area by approaching structured meshes. More precisely, along the wall the previous metric $\mathscr{M}(x)$ is replaced by a new metric

$$\hat{\mathscr{M}}(x) - T\Lambda T^{-1},$$

where

$$\Lambda = \mathrm{diag}\left(\frac{1}{h_n^2}, \lambda_\tau\right), \qquad T = (\vec{n}(x),\ \vec{\tau}(x)).$$

In other words, along solid walls the eigenvectors are now the unit normal and tangent vectors at the wall, with $1/h_n^2$ and λ_τ the corresponding eigenvalues. The refinement along the wall can now be monitored through h_n. More precisely, in the adapted mesh the distance from the wall of the first-layer nodes will be h_n. This quantity is given by the user and depends on the Reynolds number. On the other hand, λ_τ is computed by taking the maximum of the two projection lengths on $\vec{\tau}(x)$ of the $\mathscr{M}$ unit eigenvectors times the corresponding eigenvalues.

It is necessary to keep the information coming from the original metric on λ_τ to be able to recognize situations such as shock–boundary layer interactions.

A relaxation procedure can now be applied to propagate this orthogonality property further in the mesh. In this paper this has been done for only two node layers. Figure 8 (see Section 5.2) shows the distance of the points belonging to these layers from the wall for a supersonic viscous flow over an aerofoil. To give an idea of what we get without this boundary condition on the metric, the corresponding result with only the homogeneous Neumann boundary condition for the metric along the wall has also been shown in the same figure.

4.3. Multiscale phenomena

As we have stated, our experience shows that the previous approach is not satisfactory for capturing multiscale phenomena. For instance, when several eddies with variable energy are present, the mesh adaption has difficulty in capturing the weaker ones, especially if there are shocks involved in the flow. We notice that equation (6) leads to a global error while we would like to have a relative one. Indeed, for the backward-facing step for instance, the mean flow velocities at the inflow and in the main and secondary recirculations differ by several orders of magnitude (10^4–10^5). Therefore, when using a global criterion to evaluate the metric, the secondary recirculation is difficult to capture.

We propose the following estimation which takes into account not only the dimension of the variables but also their magnitude:

$$\mathscr{E} = \left| \frac{\eta - \Pi_h \eta}{\max(|\Pi_h \eta|, \varepsilon} \right|_0 \leqslant c_0 h^2 \left| \frac{D^2 \eta}{\max(|\Pi_h \eta|, \varepsilon} \right|_0, \tag{12}$$

where we have introduced the local value of the variable in the norm. The parameter ε is a cut-off to avoid numerical difficulties and also to define the difference between the orders of magnitude of the smallest and largest scales we try to capture. Indeed, when a phenomenon falls below ε, it will not be captured. This can also be seen as looking for a more precise estimation in regions where the variable is small (i.e. a variable c_0 in (6)). Another important consequence of this estimation is that it removes the dimensional problems when intersecting metrics coming from different quantities.

5. NUMERICAL EXAMPLES

In this section we describe four configurations of compressible inviscid and viscous flows. For all these cases the initial meshes have been generated using EMC^2.[14]

For the first two steady cases we compare the normalized residual evolution of the adapted computation with a direct computation where we have started on the last adapted mesh from uniform solution using the same time integration procedure. The residual is based on the norm of the right-hand side of the equations and not on the time-derivative terms. In this way the time step size does not influence the convergence history. This gives an idea of the cost of similar computations on a large uniform grid. In fact, for the same resolution we would need many more grid points on a uniform mesh, but this just reinforces our conclusions. For both cases the convergence is accelerated by the adaption technique. In CPU terms, because in the adaption loop most of the work is done at coarser levels, the computation cost is reduced by at least a factor of 20.

For the steady cases the stop criterion for the adaption loop has been the mesh independence of the results (especially wall coefficients).

These techniques have been validated on several other configurations such as multielement aerofoils. These computations can be found in Reference 15.

5.1. Supersonic scramjet inlet

The first case consists of an internal supersonic flow at Mach 3 in a scramjet inlet. Despite the configuration being symmetric, the whole domain has been computed to see whether the solution remains symmetric or not. This case is devoted to an evaluation of our multivariable strategy for the metric definition. The second modification is not used here as it is an inviscid flow.

Five automatic adaptions have been applied. We show the meshes and the corresponding iso-Mach contours from the initial to the last adaption step in Figures 2 and 3. We can see how precise the resolution of the discontinuities is for this flow at step 5. It is interesting to notice that there is no oscillation in the solution and that the fluid solver seems to nicely accept such meshes. Moreover, we see that the solution stays perfectly symmetric even on a clearly non-symmetric mesh.

The initial mesh has around 8000 triangles and the final mesh around 40,000. This gives an idea of the number of elements we might need if we want an equivalent resolution on structured grids. The convergence histories are shown in Figures 4 and 5 for the adapted and direct computations versus the number of explicit iterations and CPU time.

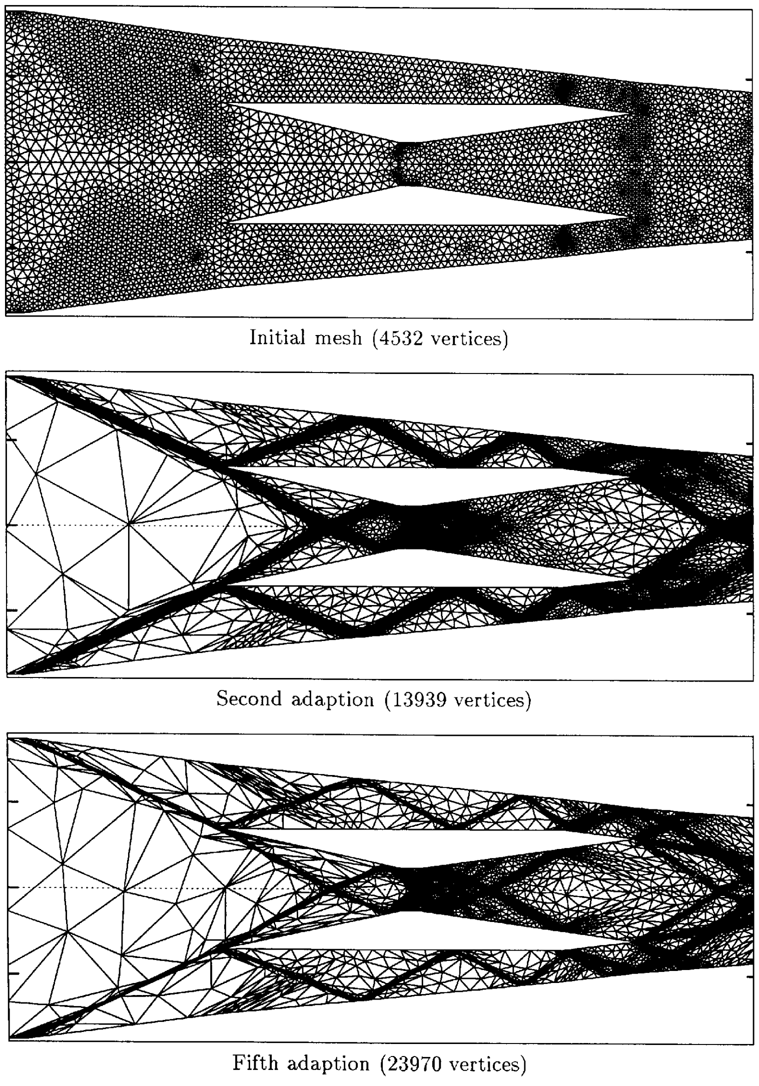

Initial mesh (4532 vertices)
Second adaption (13939 vertices)
Fifth adaption (23970 vertices)

Figure 2. Evolution of mesh (initial, steps 2 and 5) for a supersonic scramjet inlet

5.2. *Supersonic viscous flow over an NACA 0012 aerofoil*

A Mach 2 supersonic flow at Reynolds number 10,000 is studied over an NACA 0012 aerofoil. In this configuration the two new ideas we propose here are used. The quality of the wall coefficients (especially the friction coefficient) is studied too. In particular, we want to see how much they depend on the mesh.

The adaption loop length is six for this case. The convergence histories for the adaption loop and a direct computation are shown in Figure 7 versus the number of explicit iterations and CPU time. As expected, for viscous computations the convergence acceleration of the adaption strategy is more important.

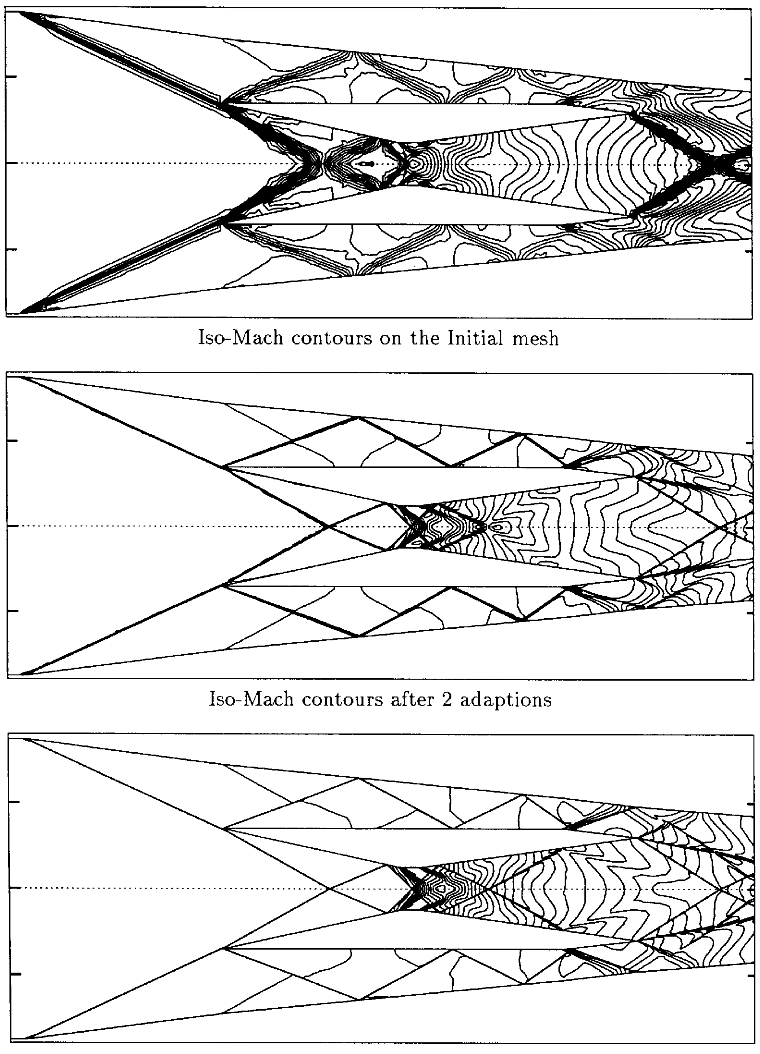


Figure 3. Evolution of iso-Mach contours for an inviscid Mach 3 scramjet inlet

We can also see that the perturbation due to the adaption on the residual is more important than in the last case. This comes from the fact that the flow is viscous and that we have our new boundary condition for the metric along the wall.

The meshes we obtain during the adaption loop and the corresponding iso-Mach contours are shown in Figure 6. We can see that the multivariable intersection procedure works well for the identification of shock, boundary layer and wake regions; even the weak shocks at the trailing edge have been detected.

Figure 8 shows the normal distance of the first and second node layers to the wall. We can see first that for the first layer this distance is exactly the same for all the nodes, which means that our boundary condition on the metric works well, and second that the relaxation procedure we have applied to propagate this property to the next layers also works quite well. For these computations we

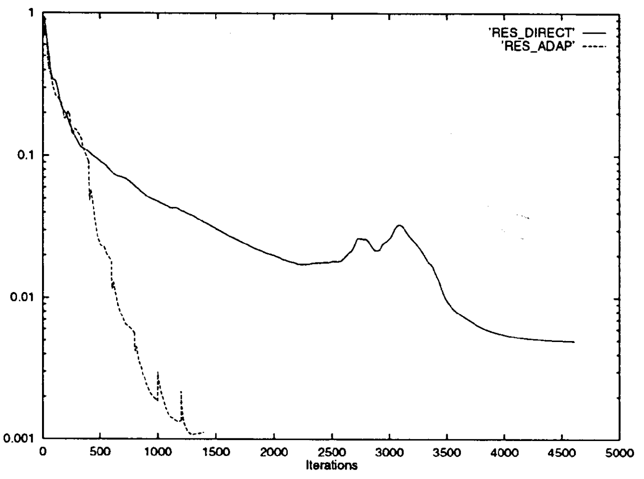


Figure 4. Convergence history versus number of explicit RK4 iterations (adaption versus direct)

have just propagated the property of being at the same distance from the wall only to the first two layers, but the same procedure can be applied to the other layers. In order to show what we obtain without this boundary condition on the metric, we have also shown the corresponding distances to the wall for the same layers after switching off our correction (i.e. the homogeneous Neumann boundary condition for the metric). This explains the poor quality of the gradients evaluated at the wall until now on adapted meshes and the fact that nice friction coefficients were hard to obtain.

Figure 9 shows the pressure and friction coefficients obtained on the initial mesh, on the mesh at step 5 and on the last one. The results for steps 5 and 6 are identical.

5.3. Turbulent flow over a backward-facing step

This is the classical backward-facing step (ratio of two between step and channel heights) at $Re_{\infty/H} = 44{,}000$ and inflow Mach number 0·1. The aim here is to show the difference between the

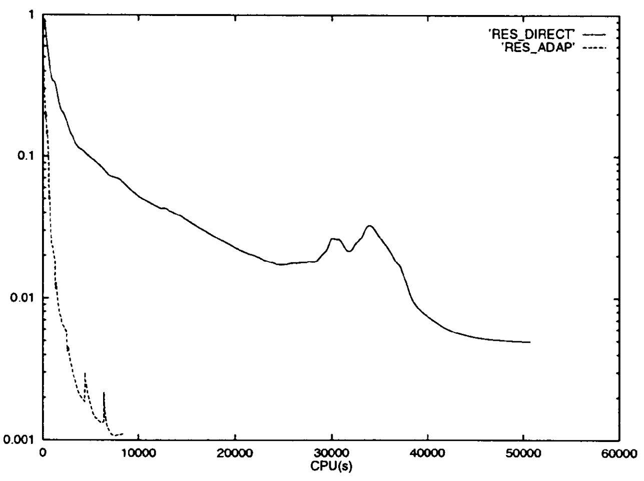


Figure 5. Convergence history versus CPU time (adaption versus direct)

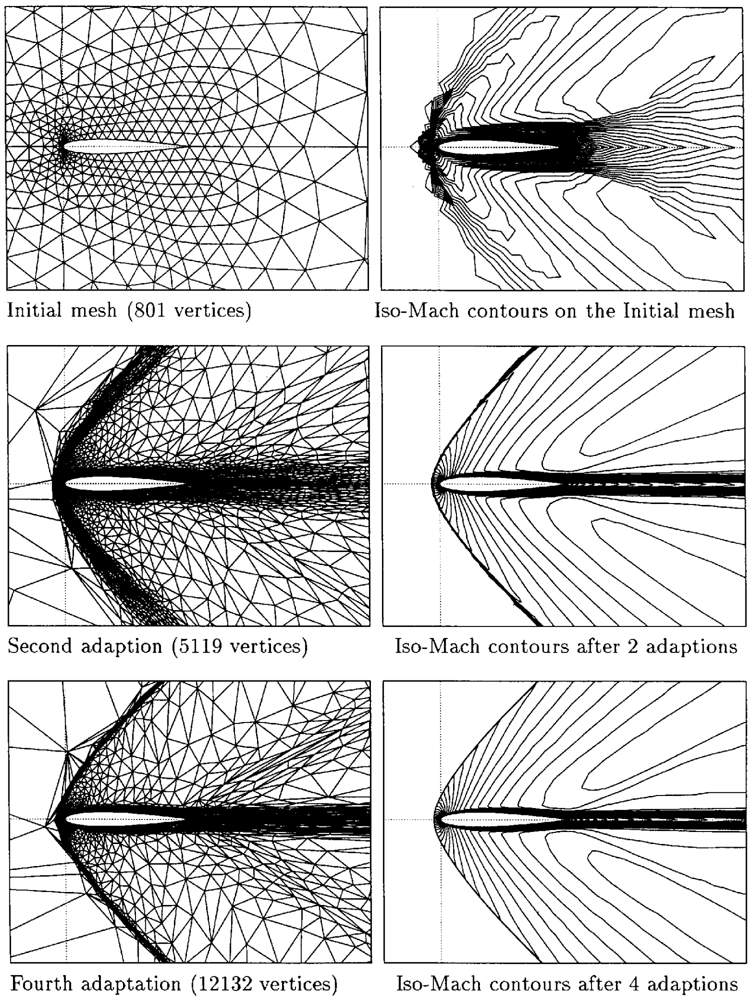
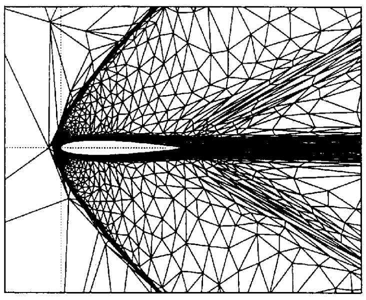
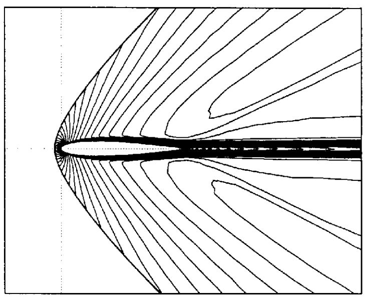

Initial mesh (801 vertices)
Iso-Mach contours on the Initial mesh
Second adaption (5119 vertices)
Iso-Mach contours after 2 adaptions
Fourth adaptation (12132 vertices)
Iso-Mach contours after 4 adaptions

Figure 6. Evolution of mesh and Mach contours (initial, steps 2 and 4) for supersonic viscous flow over an aerofoil

relative and global error estimates for the capture of multiscale phenomena. The flow has been computed on a fine regular mesh and the solution is shown in Figure 10. The predicted separation length is seven. We capture surprisingly well the secondary recirculation with our wall laws. This is an important point, as it is usually stated that this is an impossible task with eddy viscosity models and wall laws and that we need more sophisticated models for this. The secondary recirculation has a length of about $0{\cdot}2H$ and a height of about $0{\cdot}35H$.

From this mesh and solution we do one adaption using the global and relative criteria with the same $c_0\mathscr{E} = 1 \times 10^{-2}$. The obtained meshes are shown in Figure 11. We can see that when using the global estimation, the secondary bubble area (although it exists) is not detected by the adaption. The relative estimation, however, clearly captures the bubble. The relative estimation also leads to a much finer mesh in regions of interest. For the global criterion to have a similar refinement, we need to ask for a global error several orders of magnitude less, but this will lead to a uniform refinement


The final mesh at step 6, 13202 vertices, 25901 triangles


The final iso-mach contours

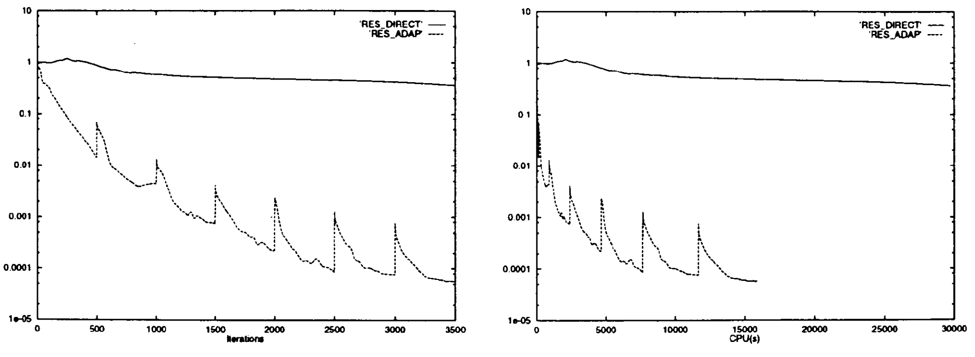


Figure 7. Convergence history (adaption versus direct)

everywhere. Therefore, if the global criterion is not able to detect a phenomenon which is already present, there is no hope for it to predict it at all.

5.4. *Unsteady flow around a cylinder*

This is a subsonic flow at Mach 0·4 and Reynolds number 80. Our aim here is to see how the mesh adaption procedure follows an unsteady phenomenon.

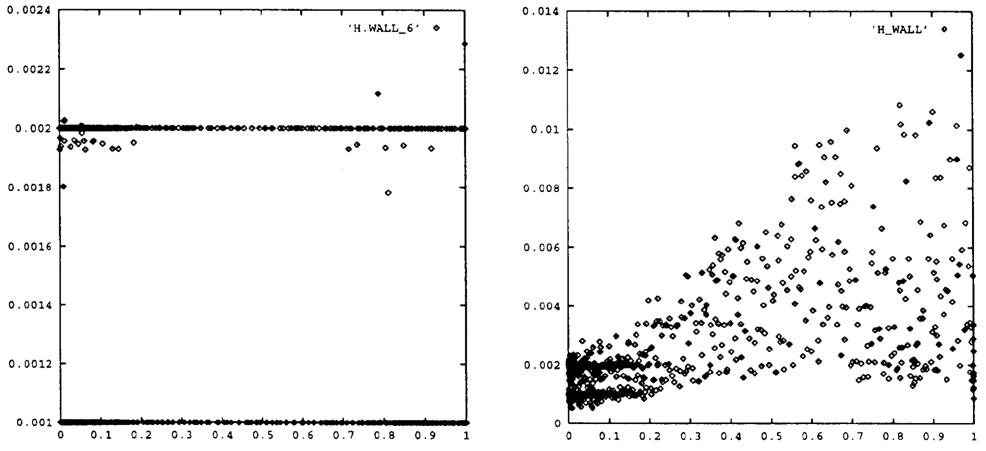


Figure 8. Wall distance with (left) and without (right) Dirichlet boundary condition

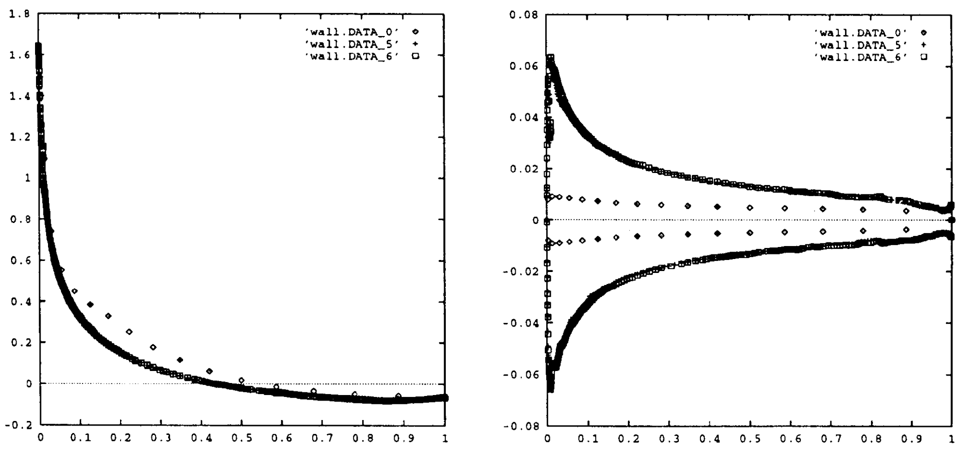


Figure 9. C_p (left) and C_f (right) at steps 0 (initial), 5 and 6. Influence of Dirichlet boundary condition for metric on wall quantities

We first do a computation on an initial mesh with about 6000 triangles until the unsteady flow is established. The adimensional period of the movement being around seven, mesh adaption has been carried out eight times in a period.

There is a major difficulty in unsteady computations compared with steady cases. Indeed, as we saw in the previous convergence history, at each adaption step the residual was perturbed. This is due

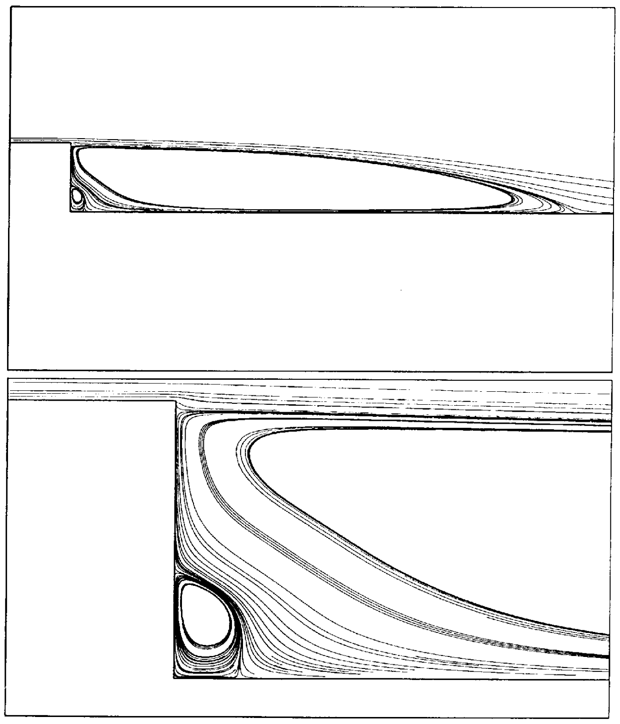


Figure 10. Backward-facing step: particle tracking for computation using relative error estimation

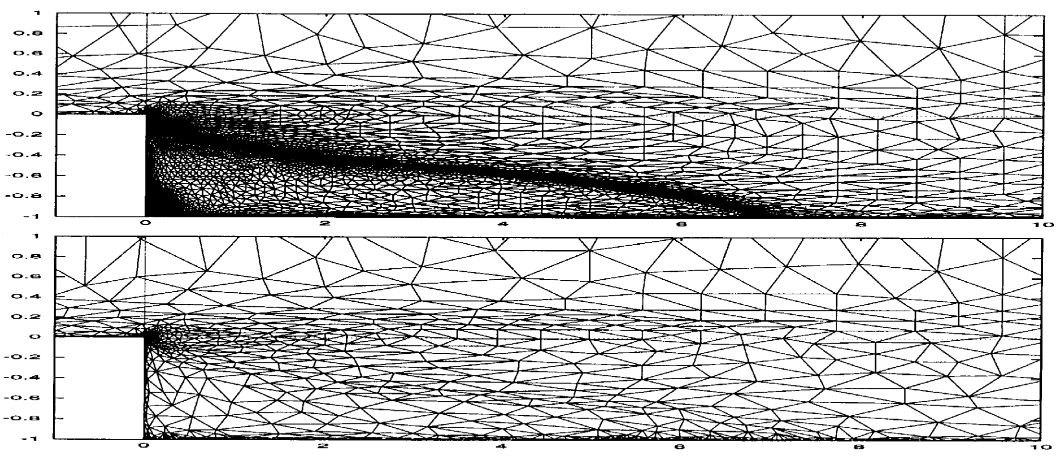


Figure 11. Backward-facing step: partial view of meshes obtained with relative (top) and global (bottom) criteria. The main and secondary recirculations are correctly identified using relative error estimation

to the fact that the linear interpolation of the solution on the new mesh cannot be compatible with the non-linear nature of the solution and the equations need time to recover their level of resolution. Basically, we need either a relaxation procedure after each adaption for the solution to be coherent with the previous resolution level or an interpolation consistent with the equations we solve. In this work, nothing has been done for this purpose, so the time evolution of quantities such as the drag and lift coefficients presents the same behaviour as the convergence history in steady computations. However, we present the mesh evolution and the corresponding iso-Mach contours in Figure 12. We can see that the mesh adaption procedure follows the unsteadiness of the flow, while the number of nodes remains almost constant.

5.5. *CPU and Memory considerations*

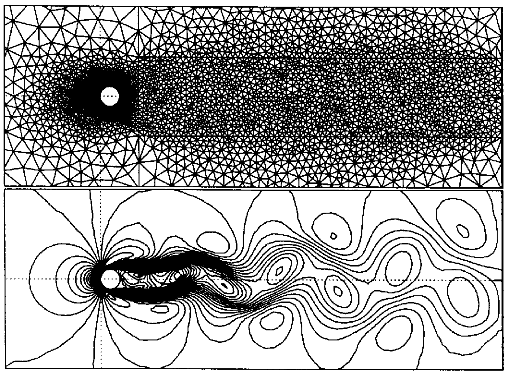
As NSC2KE is an explicit code, it does not require too much memory. The global memory required is about (1500 times the number of nodes) bytes. The speed is about 10^{-4} per node for iteration on a 1 Mflop computer for second-order viscous computations using the hybrid upwind scheme.

The metric computation and mesh generation modules are in C++ and have been optimized for memory requirement. The average speed for the generation of the adapted mesh is 500 triangles per second on a 1 Mflop computer.

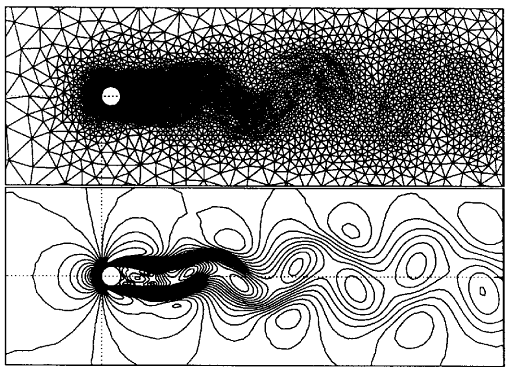
6. CONCLUDING REMARKS

Anisotropic mesh adaption has been applied with success to compressible viscous flows for a wide range of Reynolds and Mach numbers. In particular, three new ideas have been described to remove the basic difficulties with adapted meshes for flow computations involving multiple physical interactions, boundary layers and multiscale phenomena. It has been shown that these modifications greatly improve the quality of the adapted meshes for these situations.

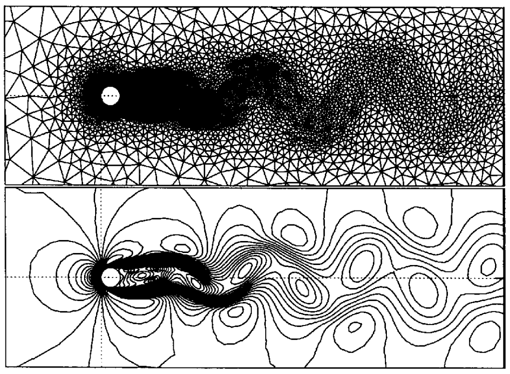
These ideas are simple to take into account in any adaption procedure because they involve only local and simple modifications of codes. Therefore they can be easily applied by other users. Similarly, their extension to 3D configurations is straightforward.



Initial mesh (3724 vertices)



1st adaption (5263 vertices)



5th adaption (5437 vertices)

Figure 12. Mesh and iso-Mach evolutions for one period

We also saw that our upwind solver did not particularly suffer from the fact that the meshes had poor quality in the Euclidean sense. We could also expect difficulties in the evaluation of second derivatives on such meshes. However, we saw that in weak form the obtained Hessian, despite not being perfect, identifies reasonably correctly the regions of interest.

Another important conclusion is that for steady computations, convergence of the explicit flow solver is improved when using mesh adaption. Indeed, as we saw, only a few thousand iterations were necessary for convergence of both the Euler and Navier–Stokes cases on quite large meshes with only a few of them on the finer levels. We recover here some kind of multigrid behaviour, but with a set of independent refined levels where only interpolations (coarse to fine) are used and not projections (fine to coarse). Therefore the global CPU requirement is significantly reduced compared with a direct computation on a uniform fine mesh.

Developments are under way for the extension of these techniques to 3D applications. This includes four steps:

(a) the metric evaluation as in 2D, obtained by a Laplacian assembly (done)
(b) the intersection of the solution metric with the geometrical metric of the 3D surface representing the boundary (done)
(c) the adaption of the corresponding 3D surface mesh (done)
(d) existing 3D Delaunay mesh generator[1] modification to take into account a local metric defined on a background mesh (in progress).

REFERENCES

1. P. L. George, *Automatic Mesh Generation, Application to Finite Element Methods*, Wiley, Chichester, 1991.
2. P. L. George, F. Hecht and E. Saltel, 'Automatic mesh generation with specified boundary', *Comput. Methods Appl. Mech. Eng.*, **92**, 269–288 (1991).
3. M. G. Vallet, *Génération de Maillages Éléments Finis Anisotropes et Adaptatifs*, Université Paris VI, Paris, 1992.
4. B. Mohammadi, 'CFD with NSC2KE: an user-Guide', *Tech. Rep. INRIA Tech. Rep. RT-0164*, 1994.
5. P. L. Roe, 'Approximate Riemann solvers, parameters vectors and difference schemes', *J. Comput. Phys.*, **43**, (1981).
6. G. D. van Albada and B. van Leer, 'Flux vector splitting and Runge–Kutta methods for the Euler equations', *ICASE Rep. 84–27*, 1984.
7. J. Steger and R. F. Warming, 'Flux vector splitting for the inviscid gas dynamic with applications to finite-difference methods', *J. Comput. Phys.*, **40**, 263–293 (1982).
8. B. E. Launder and D. B. Spalding, *Mathematical Models of Turbulence*, Academic, New York, 1972.
9. B. Mohammadi and O. Pironneau, *Analysis of the K–Epsilon Turbulence Model*, Wiley, Chichester/Masson, Paris, 1994.
10. M. J. Castro Díaz, 'Mesh refinement over surfaces', *INRIA, Res. Rep. RR-2462*, 1994.
11. M. J. Castro Díaz and F. Hecht, 'Anisotropic surface mesh generation', *INRIA Res. Rep. RR-2672*, 1995.
12. E. F. d'Azevedo and R. B. Simpson, 'On optimal interpolation triangle incidences', *SIAM J. Sci. Stat. Comput.*, **10**, 1063–1075 (1989).
13. E. F. d'Azevedo and R. B. Simpson, 'On optimal triangular meshes for minimizing the gradient error', *Numer. Math.*, **59**, 321–348 (1991).
14. F. Hecht and E. Saltel, 'EMC^2 un logiciel d'edition de maillages et de contours bidimensionnels', *INRIA, RT-0118*, 1990.
15. M. J. Castro Díaz, F. Hecht and B. Mohammadi, 'New progress in anisotropic mesh adaption for inviscid and viscous flow simulations', *INRIA Res. Rep. RR-2671*, 1995.