CONTINUOUS MESH ADAPTATION MODELS FOR CFD

Bruno Koobus,[1,2] **Laurent Hascoët,**[1] **Frédéric Alauzet,**[3] **Adrien Loseille,**[3]
Youssef Mesri,[1] **Alain Dervieux**[1]

Abstract

Continuous mesh adaptation models can be derived from *a priori* error estimates. They are used for specifying a better mesh, and possibly the best mesh in some sense, by looking for the minimum of this error model. In the case of local error, this is done by exhibiting analytically the minimum. In the case of global approximation error, an Optimal Control problem is solved. Application to Computational Fluid Dynamics (Euler compressible flow) is considered.

Key Words: mesh adaptation, a priori errors, compressible flow

1 INTRODUCTION

With modern "controlled" mesh generators, is it relatively easy to build automatically a mesh that conforms with a precise prescription of mesh density and stretching. Complementarily, better error evaluation allows for better specification of the adapted mesh. Let us consider an optimal design problem:
(P): Find the design parameter γ_{opt} for which a functional j is minimum.
Symbol γ: holds for an aircraft shape parameter, and the functional is evaluated after the solution of a state system relying on a fluid model, Euler or Navier-Stokes.
In scientific computing, j is evaluated from a *discretized* CFD solution and thus suffers from the discretization error. We express this fact by introducing a mesh parameter $\mathcal{M}$ in the different ingredients of the Optimal Design problem:

$$\begin{aligned} j(\mathcal{M},\gamma) &= J(\mathcal{M},\gamma,U(\mathcal{M},\gamma)) \,, \\ \Psi_{State}(\mathcal{M},\gamma,U(\mathcal{M},\gamma)) &= 0 \,. \end{aligned} \qquad (1)$$

Beside considering the standard Optimal Design problem:
- minimize the functional with respect to shape γ,

our goal is to address the accuracy issue as follows:
- minimize the error on functional with respect to a mesh parameter, $\mathcal{M}$.

There are several ways in minimizing that error.

In the case where meshes are fine enough to be sure that asymptotic convergence applies to the computation, then the higher the theoretical order of convergence, the smallest the error. For example the usual evaluation of the functional can be greatly improved by introducing an adjoint-based correction term which will make the new functional evaluation a superconvergent one, with typically fourth-order convergence when the standard option is only second-order. We refer to [1].
However, this assumes that asymptotic convergence is also numerical, i.e. that it will be attained for the meshes used in practice. Otherwise higher-order terms maybe larger than low order.
A second standpoint, see [2], gives genuine estimates of the error on a given mesh after a computation on it. It also detects, thanks to the introduction of the same adjoint, the regions of the computational domain where a local error measure is larger than a given prescribed tolerance. The region of too large error must be refined and therefore this latter option is a also a mesh adaptation principle.
Another mesh adaptation approach consists in trying to find the best ideal mesh density minimizing a continuous model of the error, see [3], and [4].
An important advantage of mesh adaptation relies on the better convergence order for coarser meshes, when the solution is smooth enough, or a higher order even for discontinuous solutions. See for example [5].

1 INRIA Sophia-Antipolis, France
2 Université de Montpellier II, France
3 INRIA Rocquencourt, France

The subject of this paper is to examine how a continuous model can be combined with an adjoint in order to define adapted meshes for CFD models.

2 NUMERICAL MODEL

2.1 Governing equations

The Euler equations, which express the conservation of mass, momentum, and energy for the flow of inviscid, compressible fluids, may be written in the following integral conservation law form

$$\frac{\partial}{\partial t}\int_V U dV + \oint_S \mathbf{F}\cdot \mathbf{n}\, dS = 0 \qquad (2)$$

where U is the vector of conserved variables and $\mathbf{F}$ the flux of U across the bounding surface S with outward unit normal $\mathbf{n}$ of a any control volume V. The column vector U and flux vector $\mathbf{F}$ are given by

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix} \quad ; \quad \mathbf{F}(U) = \begin{pmatrix} \rho \mathbf{u} \\ \rho \mathbf{u} u + \mathbf{pi}_x \\ \rho \mathbf{u} v + \mathbf{pi}_y \\ \rho \mathbf{u} w + \mathbf{pi}_z \\ \rho \mathbf{u} H \end{pmatrix} \qquad (3)$$

Here ρ,p, and E represent the fluid density, thermodynamic pressure, and total energy per unit mass. u, v, and w are the Cartesian components of the velocity vector $\mathbf{u}$ and H is the total enthalpy given by $H = E + \frac{p}{\rho}$. If the fluid is assumed to be a thermally perfect ideal gas, then the closure equation linking the pressure p and the conserved quantities ρ and E is provided by the equation of state

$$p = \rho(\gamma - 1)[E - \frac{1}{2}(u^2 + v^2 + w^2)] \qquad (4)$$

in which γ stands for the ratio of specific heats at constant pressure and volume.

2.2 Spatial discretization

The Euler system is solved by means of a vertex-centered Mixed-Element-Volume approximation on unstructured meshes, using a Roe Riemann solver combine with a MUSCL reconstruction with Van Albada type limiters in order to obtain a space accuracy of order two. An explicit time stepping algorithm is used and a three-stage Runge-Kutta scheme for pseudo-time integration is applied.

3 ANALYSIS OF THE ERROR

Let us simplify the approximation error by considering the first term of its Taylor expansion. The error minimization problem writes as an Optimal Control problem:

$$\begin{aligned} \bar{\mathrm{j}}(\mathcal{M}) &= \bar{\mathrm{J}}(U^{exact} - U(\mathcal{M})) = \\ &\quad |J_{exact}(U^{exact})'.(U^{exact} - U(\mathcal{M}))|^2 \quad (5) \\ \overline{\mathcal{M}} &= \mathrm{ArgMin}\ \bar{\mathrm{j}}(\mathcal{M}) \\ \text{with } & \Psi_{ERROR}(\mathcal{M}, \gamma, U^{exact} - U) = 0. \end{aligned}$$

3.1 Mesh asymptotics

This section restricts to isotropic meshes for the sake of simplicity. We consider a continuous positive function d defined on the computational domain. To d corresponds the class of isotropic meshes which have d as local mesh density.

$$\mathcal{M} = d \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

From d we derive a set of proportional densities:

$$d_h = h^{-1}\, d$$

for any positive small enough real number h. In the sequel, making h tend to zero will mean that we restrict to meshes with densities proportional to the same model d, and we examine how in those conditions the numerical error behaves.

3.2 A variational model for Euler equations

The proposed analysis is performed in this paper for pure Galerkin formulation, which means that we do not consider the effect of Godunov-type stabilization. W is a non-scalar field on the computational domain Ω and satisfy, for any non-scalar test function ϕ in the functional space $H^1(\Omega)$.

$$\int_\Omega \phi \nabla . \mathcal{F}(W) d\Omega - \int_{\partial\Omega} \phi \overline{\mathcal{F}}(W).n d\partial\Omega = 0.$$

In the corresponding discretization, the test function is taken into a V_h included in V:

$$\int_\Omega \phi_h \nabla . \mathcal{F}_h(W_h) d\Omega - \int_{\partial\Omega} \phi_h \overline{\mathcal{F}_h}(W_h).n d\partial\Omega = 0$$

where $\mathcal{F}_h(W)$ is the interpolate of $\mathcal{F}$, i.e. $\mathcal{F}_h(W_h) = \Pi_h \mathcal{F}(W_h)$, and same for $\overline{\mathcal{F}_h}(W_h)$.
Replacing ϕ by ϕ_h in the continuous system and choosing then $\phi_h = \Pi_h \phi$ in both systems we get:

$$\begin{aligned} &\int_\Omega \phi_h \nabla \mathcal{F}_h(W_h) d\Omega - \int_{\partial\Omega} \phi_h \overline{\mathcal{F}_h}(W_h).n d\partial\Omega = \\ &\qquad \int_\Omega \phi_h \nabla \mathcal{F}(W) d\Omega - \int_{\partial\Omega} \phi_h \overline{\mathcal{F}}(W).n d\partial\Omega \end{aligned}$$

In the above systems, the boundary data are involved inside the $\overline{\mathcal{F}}(W)$ and $\overline{\mathcal{F}_h}(W)$ boundary terms. For the sake of simplicity, we assume that these terms can be split in W-dependant terms, denoted respectively by $\overline{\mathcal{F}}^{out}(W)$ and $\overline{\mathcal{F}_h}^{out}(W_h)$, and constant terms, denoted $\overline{\mathcal{F}}^{in}$ and $\overline{\mathcal{F}_h}^{in}$. Therefore:

$$\int_\Omega \phi_h \nabla.\mathcal{F}_h(W_h)d\Omega - \int_{\partial\Omega} \phi_h \overline{\mathcal{F}_h}^{out}(W_h).nd\partial\Omega = \int_\Omega \phi_h \nabla.\mathcal{F}(W)d\Omega - \int_{\partial\Omega} \phi_h \overline{\mathcal{F}}^{out}(W).nd\partial\Omega$$

3.3 Error estimate

We assume now that both W and ϕ are several time continuously differentiable. In order to estimate the error, we introduce on both sides the same expression with interpolations:

$$\int_\Omega \phi_h \nabla.(\mathcal{F}_h(W_h) - \Pi_h\mathcal{F}(W))d\Omega - \int_{\partial\Omega} \phi_h(\overline{\mathcal{F}_h}^{out}(W_h) - \Pi_h\overline{\mathcal{F}}^{out}(W)).nd\partial\Omega = \int_\Omega \phi_h \nabla.(\mathcal{F}(W) - \Pi_h\mathcal{F}(W))d\Omega - \int_{\partial\Omega} \phi_h(\overline{\mathcal{F}}^{out}(W) - \Pi_h\overline{\mathcal{F}}^{out}(W)).nd\partial\Omega.$$

The left-hand side will be inverted and the right-hand side will be expanded to get the error estimate.

3.4 Interpolation errors

We recall that $\Pi_h\mathcal{F}(W) = \mathcal{F}_h(W)$. The left-hand side writes:

$$LHS = \int_\Omega \phi_h \nabla.(\mathcal{F}_h(W_h) - \mathcal{F}_h(W))d\Omega - \int_{\partial\Omega} \phi_h(\overline{\mathcal{F}_h}^{out}(W_h) - \overline{\mathcal{F}}_h^{out}(W)).nd\partial\Omega.$$

We linearize it as follows:

$$LHS = \int_\Omega \phi_h \nabla.(\Pi_h\frac{\partial\mathcal{F}}{\partial W}(W_h - W))d\Omega - \int_{\partial\Omega} \phi_h(\Pi_h\frac{\partial\overline{\mathcal{F}_h}^{out}}{\partial W}(W_h - W)).nd\partial\Omega.$$

Where the derivatives $\frac{\partial\mathcal{F}}{\partial W}$ and $\frac{\partial\overline{\mathcal{F}_h}^{out}}{\partial W}$ are evaluated from vertex values of W. We denote this in short:

$$LHS = \mathcal{A}_h(W)(W_h - \Pi_h W)$$

We assume that the corresponding linearized operator, which is the Jacobian $\mathcal{A}_h(W)$ of the discretized Euler system is invertible. This means that the implicit error $W_h - \Pi_h W$ is obtained as the unique solution of:

$$W_h - \Pi_h W = (\mathcal{A}_h(W))^{-1}\ RHS\ .$$

In the right-hand side:

$$RHS = \int_\Omega \phi_h \nabla.(\mathcal{F}(W) - \Pi_h\mathcal{F}(W))d\Omega - \int_{\partial\Omega} \phi_h(\overline{\mathcal{F}}^{out}(W) - \Pi_h\overline{\mathcal{F}}^{out}(W)).nd\partial\Omega$$

we recall that $\phi_h = \Pi_h\phi$ and we add and subtract a ϕ term:

$$RHS = RHS_1 + RHS_2$$

with:

$$RHS_1 = \int_\Omega (\Pi_h\phi - \phi)\nabla.(\mathcal{F}(W) - \Pi_h\mathcal{F}(W))d\Omega - \int_{\partial\Omega} (\Pi_h\phi - \phi)(\overline{\mathcal{F}}^{out}(W) - \Pi_h\overline{\mathcal{F}}^{out}(W)).nd\partial\Omega$$

Assuming smoothness of ϕ and $\mathcal{F}(W)$, we deduce that on Ω, interpolation errors are of order two and their gradients are of order one, same on boundary, and RHS_1 is thus of order three.

$$RHS_1 \leq \text{const.}h^3$$

The second term writes:

$$RHS_2 = \int_\Omega \phi\nabla.(\mathcal{F}(W) - \Pi_h\mathcal{F}(W))d\Omega - \int_{\partial\Omega} \phi(\overline{\mathcal{F}}^{out}(W) - \Pi_h\overline{\mathcal{F}}^{out}(W)).nd\partial\Omega$$

and we transform it as follows:

$$RHS_2 = -\int_\Omega (\nabla\phi).(\mathcal{F}(W) - \Pi_h\mathcal{F}(W))d\Omega + \int_{\partial\Omega} \phi(\mathcal{F}(W) - \Pi_h\mathcal{F}(W)).nd\partial\Omega - \int_{\partial\Omega} \phi(\overline{\mathcal{F}}^{out}(W) - \Pi_h\overline{\mathcal{F}}^{out}(W)).nd\partial\Omega$$

In RHS_2 we can apply the same asymptotic extension as in the elliptic case studied in [6]. The expression of RHS_2 is in fact very good news. Indeed, due to the smoothness assumptions for ϕ and W, L^2 estimates for interpolation error on volume and on boundary apply, so that this term appears as a second-order one:

$$RHS_2 \leq \text{const.}h^2.$$

Further, using the same techniques as in [6], this terms can be extended as follows:

$$RHS_2 \;=\; h^2\;(G(W,d),\phi)\;+\;R$$

where the last parenthesis is to be understood as a distribution one. The term R is of higher order:

$$R \;=\; o(h^2).$$

3.5 Provisional conclusion

The above study shows that the implicit error $W_h - \Pi_h W$ is a linear function of the interpolation error $W - \Pi_h W$.
A first option consists in reducing the interpolation error. This option is studied in Sections 4 and 5.
In Section 6 we shall define how to minimize with the implicit error by introducing an adjoint.

4. INTERPOLATION ERROR REDUCTION

The above analysis motivates the application of strategies relying on the minimization of L^2 and L^∞ models for interpolation error. This is an attracting option since anisotropic meshes are easily specified via the so-called anisotropic metric parametrization. This kind of strategy is already rather popular, we recall its main features. See [7, 10] for more details.

4.1 Mesh adaptation iteration

For stationary problems, the mesh adaptation scheme aims at finding a fixed point for the mesh-solution couple. In other words, the goal is to converge towards the stationary solution of the problem and similarly towards the corresponding invariant adapted mesh.

At each stage, a numerical solution is computed on the current mesh and has to be analyzed by means of an error estimate. The mesh adaptation is based on the edge length computation with respect to a discrete anisotropic metric specified at the mesh vertices. This metric is defined *via* a geometric error estimate that translates the solution variations into elements sizes and directions. Next, an adapted mesh is generated with respect to this metric. Finally, the solution is interpolated linearly on the new mesh. This procedure is repeated until the convergence of the solution and of the mesh is achieved.

4.2 Metric computation

We will focus here on the construction of the metric tensor based on the interpolation error in L^2 or L^∞ norm. On a particular element of the mesh, both norms are evaluated in a similar way using a geometric error estimate. They are then defined at every mesh vertex. For CFD simulations, we propose a specific error estimate normalization.

A geometric error estimate. As for the elliptic problems, we shall assume here that controlling the interpolation error allows us to control the approximation error Hence, we deliberately based our anisotropic geometric error estimate on the interpolation error. The error estimate aims at defining a discrete metric field that prescribes size and stretching requirements for the mesh adaptation procedure. Consequently, in an adapted mesh the interpolation error is equidistributed in all directions. More precisely, for each mesh element K, the anisotropic error interpolation bound involves the second derivatives of the variable u:

$$\| u - \Pi_h u \|_{\infty,K} \leq c_d \max_{x\in K} \max_{e\in E_K} \langle \overrightarrow{e}\ ,\ |H_u(x)|\ \overrightarrow{e}\ \rangle = \varepsilon_K\,, \tag{6}$$

where c_d is a constant related to the dimension, E_K is the set of edges of K and $|H_u| = \mathcal{R}|\Lambda|\mathcal{R}^{-1}$ is the absolute value of the Hessian of the variable u ($\mathcal{R}$ being the matrix of eigenvectors and $|\Lambda| = diag(|\lambda_i|)$ being the absolute value of the matrix of eigenvalues).

Metric construction. A discrete metric approximation which uses the mesh vertices as support is considered. Let us denote by h_{min} (resp. h_{max}) the minimal (resp. maximal) mesh element size and ε the desired interpolation error. Then, according to Relation (6), we define at each mesh vertex the anisotropic metric tensor $\mathcal{M}$ as:

$$\mathcal{M} = \mathcal{R}\,\bar{\Lambda}\,\mathcal{R}^{-1},\ \text{ where } \bar{\Lambda} = diag(\bar{\lambda}_i)$$

$$\text{and } \bar{\lambda}_i = \min\left(\max\left(\frac{c\,|\lambda_i|}{\varepsilon}, \frac{1}{h_{max}^2}\right), \frac{1}{h_{min}^2}\right).$$

Introducing a minimal (resp. maximal) element size is a practical way to avoid unrealistic metrics. It also allows us to control the time stepping in the computational scheme. In other words, in view of equidistributing the interpolation error over the mesh, we have modified the scalar product that underlies the notion of distance used in mesh generation algorithms (where the local metric $\mathcal{M}$ replaces the usual Euclidean metric).

Error estimate in CFD. Physical phenomena can involve large scale variations (multi-scale phenomena, recirculation, weak and strong shocks, etc.). It is thus difficult to capture the weakest phenomena *via* mesh adaptation, and even harder to do it when, for instance in CFD, shocks are located in the flow. Capturing such weak phenomena is crucial for obtaining an accurate solution by taking into account all phenomena interactions in the main flow area.

A local error estimation can overcome this problem [8]. Relation (6) is normalized using the local absolute value of the variable u :

$$\left\| \frac{u - \Pi_h u}{|u|_e} \right\|_{\infty,K} \leq c \max_{x \in K} \max_{\overrightarrow{e} \in \mathcal{E}_x} \langle \overrightarrow{e}\ , \frac{|H_u(x)|}{|u(x)|_e}\ \overrightarrow{e}\ \rangle\ , \tag{7}$$

where $|u|_e = max(|u|, e\,\|u\|_{\infty,\Omega})$ with $e \ll 1$ a constant. The term $e\,\|u\|_{\infty,\Omega}$ introduces a cut off to avoid numerical problems.

However, in the context of anisotropic mesh adaptation for compressible flow, capturing weak phenomena by means of Relation (7) leads to mesh isotropically strong shocks. This is due to the discretization of the solution that introduces "virtual" oscillations in the parallel direction of the shock. These oscillations have a magnitude of the same order of weak phenomena. To preserve the anisotropy, we propose to filter these oscillations with the local gradient of the solution. To this end, we suggest the following error estimate:

$$\left\| \frac{u - \Pi_h u}{\gamma |u|_e + (1-\gamma)\,\overline{h}\ \|\nabla u\|_2} \right\|_{\infty,K}$$
$$\leq$$
$$c \max_{x \in K} \max_{\overrightarrow{e} \in \mathcal{E}_x} \langle \overrightarrow{e}\ , \frac{|H_u(x)|}{\gamma |u(x)|_e + (1-\gamma)\,\overline{h}\ \|\nabla u(x)\|_2}\ \overrightarrow{e}\ \rangle\ , \tag{8}$$

where $\overline{h}$ is the diameter (*i.e.*, the length of its largest edge) of element K and γ is a parameter belongs to $[0,1]$ that will be considered close to zero if strong shocks are involved in the flow.

4.3 Anisotropic mesh adaptation

In our approach, the adaptation of the current mesh is based on the specification of a discrete anisotropic metric tensor at each vertex. For these purposes, the standard Euclidean scalar product is modified according to a proper metric tensor field $\mathcal{M}$. The aim is then to generate a mesh such that all edges have a length of (or close to) one in the prescribed metric and such that all elements are almost regular. Such a mesh is called a *unit mesh*. Let P be a vertex and let $\mathcal{M}(P)$ be the metric at P, the length of the edge PX with respect to $\mathcal{M}(P)$ is defined as:

$$l_{\mathcal{M}(P)}(PX) = \langle \overrightarrow{PX}\ , \overrightarrow{PX} \rangle^{\frac{1}{2}}_{\mathcal{M}(P)} = \sqrt{{}^t\overrightarrow{PX}\ \mathcal{M}(P)\ \overrightarrow{PX}}\ .$$

As the metric is not uniform over the domain, we need to consider the metrics at the edge endpoints as well as all intermediate metrics along the edge. To achieve this, we assume that an edge PX has a local parametrization $PX = P + t\overrightarrow{PX}$ and we introduce its average length as:

$$l_{\mathcal{M}}(\overrightarrow{PX}) = \int_0^1 \sqrt{{}^t\overrightarrow{PX}\ \mathcal{M}(P + t\overrightarrow{PX})\ \overrightarrow{PX}}\ dt\ . \tag{9}$$

Assuming that the metric is normalized, the desired adapted mesh is then a *unit mesh*, as all edges must have a length close to one.

Here, we consider the generation of adapted meshes in three dimensions as a two-steps process. At first the surface mesh is adapted using local modifications [9], then the volume mesh is adapted using a constrained Delaunay algorithm [11]. Notice that, during the point insertion phase of the volume mesh generation, most of the vertices of the previous mesh are reused for cpu concerns. This could reduce interpolation errors but in practice kept vertices are mostly in non critical area.

4.4 Solution interpolation

Solution interpolation is a key point in the mesh adaptation algorithm. The aim is to recover the solution field after generating a new adapted mesh. As we have a discrete solution field, we need an interpolation scheme to transfer this information from the current mesh to the newly adapted mesh.

During the interpolation stage, two problems have to be taken into account. First, locating the new vertices in the background mesh by identifying the elements containing them. This can be solved by moving inside the (oriented) mesh by using its topology thanks to a barycentric coordinates based algorithm [10]. Once the localization has been solved, an interpolation scheme is used to extract the information from the solution field. In our case, as the solution is considered piecewise linear by elements (because the solution is defined only at the mesh vertices) we use a classical P_1 interpolation scheme.

5 SOME NUMERICAL RESULTS

To illustrate the efficiency of the proposed approach, we will now present two application examples of three-dimensional CFD simulations.

5.1 Supersonic business jet

The first example concerns the simulation of supersonic flow for a future business jet at Mach 1.8 with an angle of incidence of 3 degrees at an altitude of 15, 200 meters. The design and conception of this future business plane (Dassault Aviation) has led to investigate the control of the sonic boom phenomenon. Beside classical studies aimed at reducing the drag and at increasing the lift of the airplane, the shape optimization process for supersonic civil aircrafts includes another component: the need to reduce the noise at

the ground level. In our case, the goal was to analyze the impact on the sonic boom of the optimization of the front part of the airplane geometry. Numerically, this study combine a good near field flow computation (including the computation of the aircraft signature: pressure fields) and a good sonic boom prediction (far field propagation). In the preliminary stage of this project, mesh adaptation already proved to be a very efficient tool.

The variable used to adapt the mesh is the Mach number. The mesh has been adapted 9 times, every 250 time steps. Figures 1 and 2 present the final adapted mesh with the corresponding Mach number distribution for this simulation. In this Figure, the Mach cones are clearly identified in front of the fuselage on the adapted mesh. The initial mesh contains 41,137 vertices, 20,588 boundary triangles and 216,916 tetrahedra. The final mesh (iteration 9) contains 798,756 vertices, 38,492 boundary triangles and 4,714,162 tetrahedra. The CPU times required to compute the whole simulation is 44 hours on a 600MHz workstation with 1Gb of memory. Notice that the meshing time represents only 2% of the solver CPU times.

5.2 Anisotropic ONERA M6 Wing

The second example concerns a classic numerical simulation of transonic air flow around the ONERA M6 wing. A Euler solution is computed for Mach number equal to 0.8395 with an angle of attack of 3.06 degrees. This transonic simulation case features a well-known lambda-shock. The initial mesh is a relatively coarse mesh containing 7,815 vertices, 5,848 boundary triangles and 37,922 tetrahedra. The variable used to adapt the mesh is the Mach number. The mesh has been adapted 9 times, every 250 time steps. Figure 3 (resp. 4) shows the adaptation in the isotropic (resp. anisotropic) case. The final isotropic mesh (iteration 9) contains 231,113 vertices and 1,316,631 tetrahedra and the final anisotropic mesh contains 23,516 vertices and 132,676 tetrahedra. In this example, the maximal aspect ratio of the anisotropic elements is about 10. Nevertheless, the anisotropic metric leads to a dramatic reduction of the number of degrees of freedom, roughly one order less than in the isotropic case, for the same error level. The CPU times required to generate the final surface (resp. volume) mesh is 31 (resp. 132) seconds, and to compute the Euler solution over 250 time steps is 2,782 seconds in the isotropic case, on a Pentium 4 3Mhz machine. Similarly, the CPU times required to generate the surface (resp. volume) mesh is 3 (resp. 25) seconds, and to compute the Euler solution over 250 time steps is 318 seconds in the anisotropic case.

6 TOWARDS IMPLICIT ERROR REDUCTION

The analysis proposed in Sec. 3 of the implicit approximation error Y leads us to modelize it as the solution of the following system:

$$\frac{\partial \Psi}{\partial W}(W,d)Y = G(W,d),$$

where W represents the flow variables the accuracy of which we optimize. At the moment when we optimize the mesh parameter (density or metric), W is frozen. Symbol d stands for the mesh local density to be ajusted for minimizing the error functional $\overline{j}$.

The optimality system for the optimal mesh problem writes:

$$\begin{aligned}
\frac{\partial \Psi}{\partial W}(W,d)Y - G(W,d) &= 0 \\
\left(\frac{\partial \Psi}{\partial W}(W,d)\right)^* \Pi - \frac{\partial \overline{J}}{\partial Y}(Y,d) &= 0 \\
\overline{j}' = -\left(\frac{\partial (\frac{\partial \Psi}{\partial W}Y - G)}{\partial d}(W,d)\right)^* \Pi + \frac{\partial \overline{J}}{\partial d}(Y,d) &= 0
\end{aligned}$$

In [6, 12], we show that for an elliptic linear model, the assembly of the residual of the optimality equations can be obtained by Automatic Differentiation [14] of procedures from the initial solver for Y. Specifically, this uses the reverse mode of AD, which is available in AD tools such as TAPENADE [13]. To briefly summarize AD, consider a routine that computes an output array v from an input array u,

$$u \mapsto v = \Phi(u)$$

the *tangent mode* of AD produces a routine computing the directional derivative:

$$u, \dot{u} \mapsto \frac{\partial \Phi}{\partial u}(u)\ \dot{u}\ .$$

The *reverse mode* of AD produces a routine computing from u and any array $\overline{v}$ of the same dimension as v the following product which has the dimension of u:

$$u, \overline{v} \mapsto \left(\frac{\partial \Phi}{\partial u}(u)\right)^*\ \overline{v}\ .$$

For example, we can apply this process to the resolution for Π. We first notice that the assembly for the two terms of the equation for Π in the optimal mesh system both result from reverse differentiation of the existing procedures for Ψ and $\overline{J}$. Furthermore, the solution algorithm is the same for any adjoint state

corresponding to the present state equation. This allows us for even more code re-use.
Let us now focus on the third equation of the optimality system. It requires to assemble in particular:

$$\left(\frac{\partial}{\partial d}\left(\frac{\partial \Psi}{\partial W}Y\right)\right)^{*}\Pi$$

We can see that this term can be naturally assembled in two steps, "reverse-on-tangent fashion". First apply tangent mode AD to Ψ with respect to W, setting W to Y. Then apply reverse mode AD to the resulting program, with respect to d, and setting the dual direction to Π. This term is also equal to:

$$\frac{\partial}{\partial W}\left(\frac{\partial \Psi}{\partial d}^{*}\Pi\right)Y$$

which can be assembled in two steps, "tangent-on-reverse fashion". First apply reverse mode AD to Ψ with respect to d, setting Ψ to Π. Then apply tangent mode AD to the resulting program, with respect to W, setting the tangent direction to Y.
The two methods are equivalent and produce codes that perform the same operations, although in a slightly different order. This is not the same situation as the computation of Hessian matrices, for which it has been observed that the tangent-on-reverse approach yields a more efficient program. In our case, reverse-on-tangent even allows us to re-use an existing piece of code.

7 CONCLUSION

Our propositions for mesh adaptation concern rather sophisticated algorithms involving flow solver, local error evaluation, sensitivity analysis by Automated Differentiation, adjoint and gradient based optimization, controlled generation of next mesh, interpolation of solution till a convergence of a global loop is attained. Our experience on this way showed that when well justified, sophisticated algorithm do not bring only robustness, but also efficiency. We expect to demonstate this further in future steps.

REFERENCES

[1] M. Giles and E. Suli. Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality. *Acta Numerica*, pages 145–236, 2002.

[2] R. Becker and R. Rannacher. A feed-back approach to error control in finite element methods: basic analysis and examples. *East-West J. Numer. Math.*, 4:237–264, 1996.

[3] I. Babuška and T. Strouboulis. *The Finite Element Method and its reliability*. Oxford Scientific Publications, New York, 2001.

[4] F. Courty, D. Leservoisier, P.-L. George, and A. Dervieux. Continuous metrics and mesh optimization. *Applied Numerical Mathematics*, 2004. http://www-sop.inria.fr/tropics/Alain.Dervieux/ApplNum04.pdf to appear.

[5] A. Dervieux, D. Leservoisier, P.-L. George, Y. Coudiere, About theoretical and practical impact of mesh adaptations on approximation of functions and of solution of PDE, *Int. J. Numer. Meth. Fluids*, 43, 507-516, 2003

[6] F. Courty, T. Roy, B. Koobus, M. Vázquez, A. Dervieux Error analysis for P1-exact schemes, Finite Element for Flow Problems April 4-6,2005, Swansea, Wales, UK, extended paper submitted for I.J. Num. Methods in Fluids

[7] P.J. Frey and F. Alauzet, Anisotropic mesh adaptation for CFD computations *Comput. Methods Appl. Mech. Engrg.*, 194, 5068-5082 (2005).

[8] M.J. Castro-Diaz, F. Hecht, B. Mohammadi and O. Pironneau, Anisotropic Unstructured Mesh Adaptation for Flow Simulations. *Int. J. Numer. Meth. Fluids*, 25, 475-491 (1997).

[9] P.J. Frey, About surface remeshing, in *Proc. of 9th Int. Meshing Rountable*, New Orleans, LO, USA, 123-136, (2002).

[10] P.J. Frey and P.L. George, *Mesh generation. Application to finite elements*. Hermès Science Publ., Paris, Oxford (2000).

[11] P.L. George, Tet meshing: construction, optimization and adaptation. in *8th International Meshing Roundtable*, South Lake Tahoe, CA, USA, (1999).

[12] A. Dervieux, L. Hascoet, M. Vázquez and B. Koobus. Optimization loops for shape and error control, *Proc. of Post-SAROD Workshop*, Bangalore, 2005.

[13] L. Hascoet and V. Pascual. Tapenade 2.1 user's guide. Technical Report 0300, INRIA, 2004.

[14] A. Griewank. *Evaluating Derivatives: principles and techniques of Algorithmic Differentiation*, SIAM, 2000.

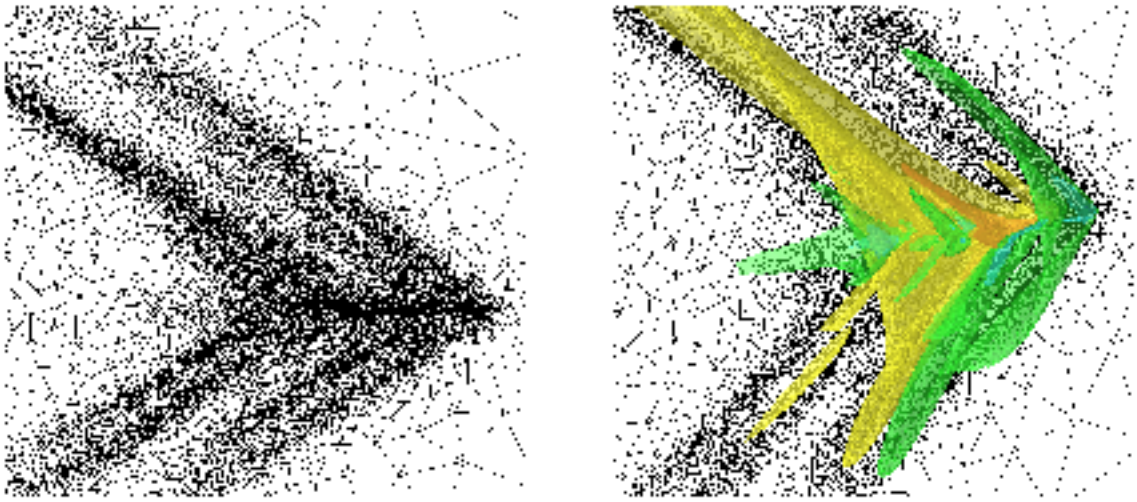

Fig.1: *Supersonic business jet test case: isotropic surface mesh at iteration 9 of the adaptation scheme. Right, isosurface of Mach number for the final solution. Mach cones are clearly identified*

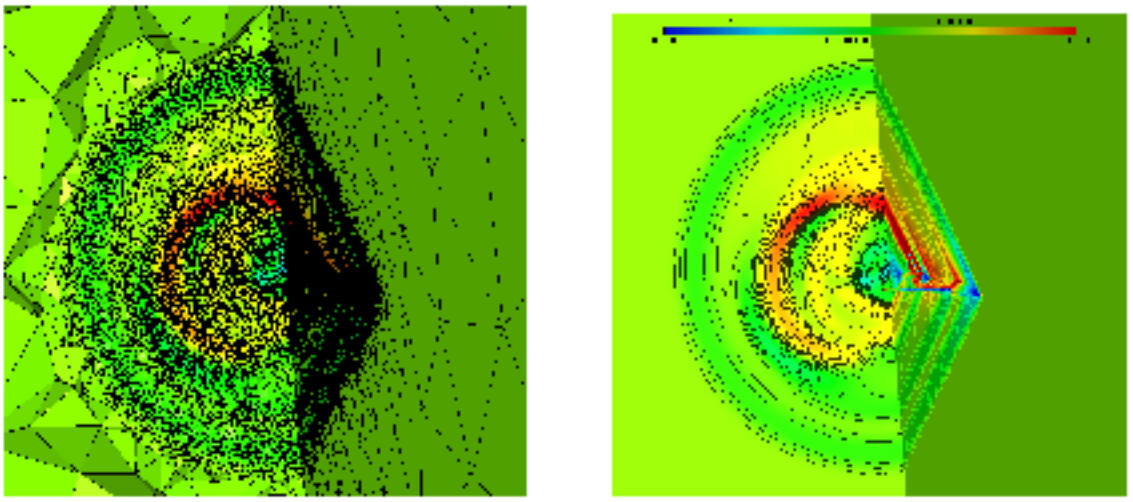

Fig.2: *Supersonic business jet test case: cut through the isotropic volume mesh at iteration 9 of the adaptation scheme. Right, isoline of Mach number in the cut plane.*

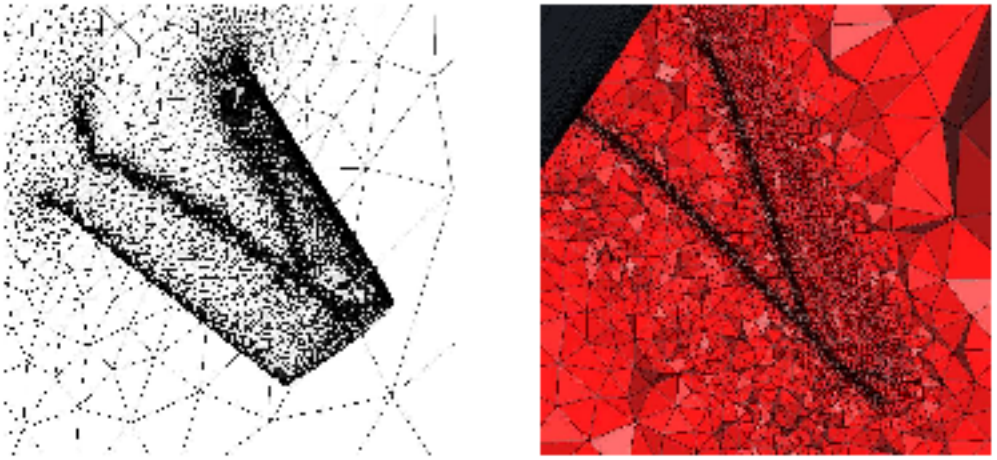

Fig.3: *Onera M6 wing test case: isotropic surface and cut through the volume mesh at iteration 9 of the adaptation scheme.*

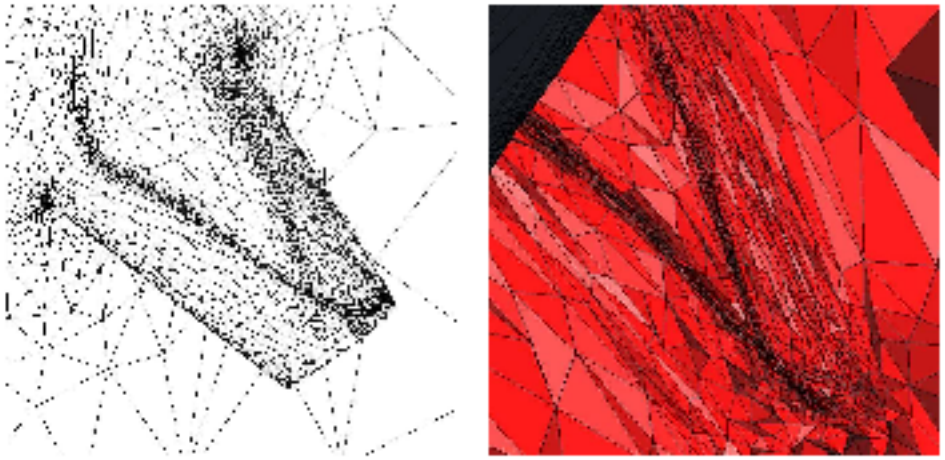

Fig.4: *Onera M6 wing test case: anisotropic surface and cut through the volume mesh at iteration 9 of the adaptation scheme.*