

ANISOTROPIC MESH ADAPTATION FOR TRANSIENT FLOWS SIMULATIONS

Pascal J. FREY^{†‡} and Frédéric ALAUZET[†]

[†] INRIA, Projet Gamma, Domaine de Voluceau, BP 105, 78153 Le Chesnay cedex, France

[‡] Laboratoire Jacques-Louis Lions, Université Pierre et Marie Curie, 75013 Paris, France

ABSTRACT

Unstructured mesh adaptation has already revealed very efficient for computing an accurate solution in a reasonable amount of time on current PC architectures. Two features are still missing in the adaptation scheme: (i) the creation of arbitrary anisotropic meshes and (ii) the capture of transient phenomena. Therefore, in this paper, we propose a global scheme suitable to compute steady-state as well as transient problems, based on anisotropic mesh adaptation. Several examples of numerical simulations in CFD are provided to emphasize the efficiency of the proposed approach.

Keywords: Mesh adaptation, anisotropy, metric, error estimate, CFD, transient problem

INTRODUCTION

Nowadays, in the context of numerical simulations, unstructured mesh adaptation is unanimously recognized as an efficient and clever method for improving the accuracy of the solution as well as for capturing the behavior of physical phenomena, even if it is not known *a priori*. Moreover, reducing the number of nodes (the number of degrees of freedom) allows to substantially reduce the CPU time. Hence, three-dimensional complex simulations are now commonly used in many engineering fields ! It becomes even possible to compute large unsteady simulations on current workstations (no parallel architecture required) almost on a daily basis. A missing piece on the simulation checkboard remained the anisotropic adaptation for unsteady phenomena. This paper attempts to bridge this gap.

Problem statement

As pointed out, mesh adaptation is now widely used in complex three-dimensional numerical sim-

ulations, especially in CFD. The aim of mesh adaptation is to control the generation of a new mesh in a computational scheme, such that the computational error (the approximation error) estimated on this mesh is bounded by a given threshold value. If we consider that the approximation error is bounded by the interpolation error, the problems turns to generate a new mesh on which the interpolation error is equidistributed.

Here, we attempt to glue various software pieces together, in order to build a fully automatic anisotropic mesh adaptation scheme suitable for unsteady problems. This challenging application is based on the definition of a geometric error estimate of the interpolation error, on the construction of a proper discrete metric tensor and on governed anisotropic surface and volume meshing algorithms. All these pieces have one feature in common: they all involve an anisotropic metric. Anisotropic mesh generators require a metric to be defined at the mesh vertices in order to create adapted meshes. The discrete metric is defined at the mesh vertices in order to account for the variations of (possibly all) the variables of the problem.

So, the error estimate has to relate the interpolation (or approximation) error to element sizes and directions, using a ... metric. It turns out that defining a suitable metric is the key to success in mesh adaptation.

Related work

Over the last few years, a rather large number of papers have been published on mesh adaptation for numerical simulations (see [14] for a survey). However, only a few number of papers have addressed the problem of constructing three-dimensional anisotropic meshes. Actually, most of these papers mainly suggested to modify the point insertion procedure, common to all meshing algorithms, to account for boundary layers (*i.e.*, close to the boundaries). Although some of these algorithms may be useful in such circumstances, all of them usually lack the kind of generality (or automation) required to capture, for instance, shocks waves in the regions away from the domain boundaries. Indeed, fully automatic tools are still crucially required in the context of mesh generation and error estimation. Once available, such tools would greatly facilitate the design and implementation of an automatic adaptation scheme, for any type of numerical simulation (the solver would be the only application dependent software piece).

Paper outline

In this paper, we briefly introduce an *a posteriori* geometric error estimate based on a discrete approximation of the second derivatives of the variables of the problem that will be used to define a suitable metric for mesh adaptation, Section 1. In Section 2, we recall the main stages of the mesh adaptation scheme, *i.e.*, surface and volume governed mesh generation. In Section 3, we present a new adaptation scheme based on a transient fixed point algorithm suitable for transient problems. Finally, in Section 4, three application examples of mesh adaptation are provided to illustrate the proposed approach, in the isotropic and anisotropic case.

1. METRIC RELATED ISSUES

In many engineering applications, it is often desirable to create adapted meshes presenting highly anisotropic features (stretched elements in arbitrary directions). This challenging problem can be (easily) resumed to that of constructing appropriate metric tensors in order to govern the mesh-

ing algorithms. The key idea is thus to define a convenient metric tensor (*via* a suitable error estimate) based on the discrete variables (solutions) of the problem. Such a discrete metric can be used to create *unit meshes* as will be seen in the next Section.

In this Section, we will first introduce the main principle of a geometric error estimate of the interpolation error. Then, we will explain how to construct a metric tensor for mesh adaptation.

1.1 A geometric error estimate

As the finite element solution u_i is not interpolant and as it is not possible to guarantee that u_i coincide with the exact solution u in at least one point of each element, it seems rather difficult to quantify the gap $e_i = u - u_i$. However, it is possible to use an indirect approach to measure this gap [1]. For elliptic problems, it has been proved (Céa's lemma, [12]) that the FE error is bounded by the interpolation error e_i : $\|e_i\| \leq C \|u - \Pi_h u_i\|$, where $\Pi_h u_i$ is the interpolation of u on the mesh \mathcal{H}_i , $\|\cdot\|$ is a norm of \mathbb{R}^3 and C a constant independent of the current mesh \mathcal{H}_i . We assume that this relation still holds in the class of problems described here. Actually, studies based on the interpolation error show (practically) that the link between the interpolation error and the approximation error is even stronger than the bound given by Céa's lemma. Hence, the interpolation error leads to a "good" error estimate [14]. The problem is to characterize the mesh on which the interpolation error is bounded by a given tolerance value or equidistributed.

It has been proved that the analysis of a "measure" of the interpolation error leads to define an anisotropic metric map which prescribes element sizes and directions [7]. To measure the interpolation error, we consider the discrete L_∞ norm of the error defined in tetrahedron K as:

$$\|u - \Pi_h u\|_{\infty, K} \leq c \max_{x \in K} \max_{\vec{e} \in E_K} \langle \vec{e}, |H_u(x)| \vec{e} \rangle, \quad (1)$$

where c is a constant dependent of the dimension, E_K is the set of element edges of K and $|H_u|$ is the absolute value of the Hessian of the variable u (symmetric definite positive tensor). As the Hessian matrix is symmetric, it can be decomposed as: $H_u = \mathcal{R} \Lambda \mathcal{R}^{-1}$ where \mathcal{R} is the eigenvector matrix and $\Lambda = \text{diag}(\lambda_i)$ is the eigenvalue matrix, the absolute value of the Hessian matrix is then

defined as follows:

$$|H_u| = \mathcal{R} |\Lambda| \mathcal{R}^{-1} \quad \text{with } |\Lambda| = \text{diag}(|\lambda_i|).$$

Notice that this error is related to the Hessian of the variable u and to the mesh edges, hence it provides directional thus anisotropic information. Controlling the mesh edges allows to control the interpolation error on the mesh elements.

From a practical point of view, the right-hand side term of Equation (1) is not useful as it involves the maximum of the metric field $|H_u|$ that is usually not known. Nevertheless, it is possible to define a suitable metric tensor $\widetilde{\mathcal{M}}(K)$ such that the interpolation error on a mesh element is given by:

$$\varepsilon_K = c \max_{\vec{e} \in E_K} \langle \vec{e}, \widetilde{\mathcal{M}}(K) \vec{e} \rangle.$$

In the mesh adaptation context, the error tolerance ε that must be equidistributed over the mesh is fixed and the mesh element have to be characterized *via* this constraint. For a given mesh element K , we define the metric tensor $\mathcal{M}(K) = c\varepsilon^{-1} \widetilde{\mathcal{M}}(K)$, and all mesh edges must comply with the following equality:

$$\langle \vec{e}, \mathcal{M}(K) \vec{e} \rangle = 1. \quad (2)$$

A mesh for which all edges comply with this relationship is a so-called *unit mesh*.

In other words, to equidistribute the interpolation error over the mesh, we have modified the scalar product that lies under the notion of distance used in mesh generation algorithms, based on the local metric \mathcal{M} that replace the usual Euclidean metric. This tensor \mathcal{M} must still be defined more precisely.

The specificity of this error estimate is related to the following features:

- the analysis is not asymptotical, (h does not tend towards zero),
- it is based on the hessian of the solution,
- it is intrinsically anisotropic
- it does not depend of the nature of the operator (therefore it can be used for any type of equation).

Remark 1.1. This error estimate is called "geometric" as the solution on a mesh can be seen as a Cartesian surface and we attempt to define a geometric metric in order to control the gap to the surface.

1.2 Metric construction

Let us denote by h_{min} (resp. h_{max}) the minimal (resp. maximal) mesh element size. According to the previous section, we define the metric tensor as: $\mathcal{M} = \mathcal{R} \Lambda \mathcal{R}^{-1}$, with:

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

Introducing a minimal (resp. maximal) element size is a way of avoiding unrealistic (unpracticable) metrics. It also allows in a computational (explicit) scheme to control the time step.

The Equation (1) leads to a global upper bound of the interpolation error. However, in order to combine various variables, each of them having a different meaning or a different nature, it becomes necessary to introduce a relative bound on the interpolation error, to have dimensionless variables, as follows:

$$\frac{\|u - \Pi_h u\|_{\infty, K}}{\|u\|_{\infty, \Omega}} \leq c \max_{x \in K} \max_{\vec{e} \in E_K} \frac{\langle \vec{e}, |H_u(x)| \vec{e} \rangle}{\|u\|_{\infty, \Omega}}.$$

Then, all metrics can be combined together into a single metric tensor using a metric intersection scheme.

Moreover, in numerical simulations, solutions vary from several orders of magnitude (multi-scale phenomena, recirculations, 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. A local error estimation can overcome this problem. Following the previous idea, the error estimate is also normalized using the local value of the gradient norm of the variable u , weak phenomena can be captured even in presence of strong shocks. To this end, we introduce the following estimate:

$$\left\| \frac{u - \Pi_h u}{|u| + \bar{h} \|\nabla u\|_2} \right\|_{\infty, K} \leq c \max_{x \in K} \max_{\vec{e} \in E_K} \langle \vec{e}, \frac{|H_u(x)|}{|u| + \bar{h} \|\nabla u\|_2} \vec{e} \rangle,$$

where \bar{h} is the diameter of the element (its largest edge) in the background mesh (at the previous iteration).

1.3 Metric intersection

When several metrics are specified at the same vertex, a unique metric tensor must be defined taking into account all given metrics. To this end, a metric intersection procedure is used. Let \mathcal{M}_1 and \mathcal{M}_2 be two metric tensors given at a vertex P . The metric tensor $\mathcal{M}_{1\cap 2}$ corresponding to the intersection of \mathcal{M}_1 and \mathcal{M}_2 must be such that the interpolation error for each variable is bounded by the given tolerance value. To this end, we use the simultaneous reduction of the quadratic forms associated with the two metrics [1] (cf. Figure 1).

The two metric tensors are represented by the associated ellipsoids $\mathcal{E}_{\mathcal{M}_i}$. The ellipsoid $\mathcal{E}_{\mathcal{M}}$ of maximal volume included in the (geometric) intersection of these two ellipsoids defines the desired metric tensor. Formally speaking, let us consider

$$M_d = \{\mathcal{M} \mid \mathcal{M} \text{ metric tensor}\}$$

be the set of all metric tensors in \mathbb{R}^d and let us define the ellipsoid associated with the metric \mathcal{M} by:

$$\mathcal{E}_{\mathcal{M}} = \{M \mid \sqrt{{}^t\overrightarrow{PM} \mathcal{M} \overrightarrow{PM}} = 1\}.$$

Hence, the metric $\mathcal{M}_{1\cap 2}$ is defined by the ellipsoid $\mathcal{E}_{\mathcal{M}_{1\cap 2}} = \sup_{\mathcal{M}_i \in M_d} \mathcal{E}_{\mathcal{M}_i} \subset \mathcal{E}_{\mathcal{M}_1} \cap \mathcal{E}_{\mathcal{M}_2}$ or:

$$\sup_{\mathcal{M}_i \in M_d} \left\{ M \mid \sqrt{{}^t\overrightarrow{PM} \mathcal{M}_i \overrightarrow{PM}} = 1 \right\} \subset \mathcal{E}_{\mathcal{M}_1} \cap \mathcal{E}_{\mathcal{M}_2},$$

where the *sup* over the set of metrics represents the metric with the largest volume of its associated ellipsoid.

2. MESH ADAPTATION

As mentioned in the Introduction, the generation of an adapted mesh is based on the specification of a discrete anisotropic metric tensor at each mesh vertex of the current mesh. The aim is then to compute the edge lengths with respect to this metric. For the sake of simplicity, it is possible to define the metric tensor so as to prescribe a unit edge length. The standard Euclidean scalar product is

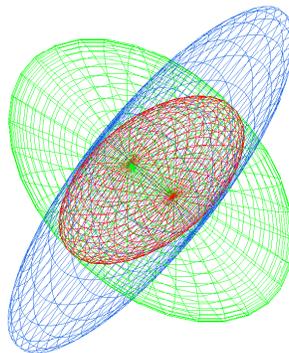


Figure 1. Metric intersection in three dimensions.

then modified using a proper metric tensor field. At each vertex, a different expression of the metric \mathcal{M} leads to a different expression of the scalar product. Let P be a vertex and let $\mathcal{M}(P)$ be the metric at P . The desired edge PX must have a length close to one w/r $\mathcal{M}(P)$:

$$l_{\mathcal{M}(P)}(\overrightarrow{PX}) = \sqrt{{}^t\overrightarrow{PX} \mathcal{M}(P) \overrightarrow{PX}} = 1.$$

As the metric varies in the domain (is not constant in an element), we need to consider the metrics at the edge endpoints as well as all intermediate metrics along the edge. To this end, we introduce the average length of PX as:

$$l_{\mathcal{M}}(\overrightarrow{PX}) = \int_0^1 \sqrt{{}^t\overrightarrow{PX} \mathcal{M}(P + t\overrightarrow{PX}) \overrightarrow{PX}} = 1. \quad (3)$$

The desired adapted mesh is then a *unit mesh*, i.e., a mesh such that for each edge $\vec{e} \in E_K$, $l_{\mathcal{M}}(\vec{e}) \approx 1$.

In our approach, the generation of adapted meshes is a two-steps process. At first the surface mesh is adapted using local modifications [16], then the volume mesh is adapted using a constrained Delaunay algorithm extended to the anisotropic case [19]. Notice that most of the vertices of the previous meshes are kept in order to reduce errors (as much as possible) when interpolating the solutions from one mesh to the other.

2.1 Surface mesh adaptation

Given a discrete surface (a piecewise linear approximation of the domain boundaries) and a discrete

metric field, the aim is to generate an adapted mesh with respect to this metric. To this end, the approach we use consists in modifying iteratively the initial surface mesh so as to complete a unit mesh. Obviously, as the mesh is intended for FE computations, the mesh gradation is also a major concern. The ingredients to achieve this goal typically include mesh enrichment, mesh coarsening and local mesh optimization procedures. The local mesh modifications operators involved are: edge flipping, edge collapsing, edge splitting and node removal, node repositioning and degree relaxation.

As no CAD information is provided, an internal (at least) C^1 continuous geometric support is first constructed, using local quadrics (defined at the mesh vertices). Then, a geometric metric tensor \mathcal{G} is defined at the mesh vertices using this support, the local principal curvatures and directions are computed. This geometric metric \mathcal{G} has to be intersected with the computational metric \mathcal{M} so as to cope at best with these two requirements. In turn, this metric $\mathcal{G} \cap \mathcal{M}$ must be modified to account for the desired mesh gradation [10]. The resulting metric $\widetilde{\mathcal{M}}$ is used to govern all mesh modifications.

The surface mesh modification algorithm is pretty straightforward, edge lengths are computed with respect to the metric $\widetilde{\mathcal{M}}$ and edge too small are collapsed while edge too long are splitted into unit length segments. Edge flips and node repositioning operations are performed to improve the overall mesh quality (in terms of shape and size) [17].

2.2 Volume mesh adaptation

Once the surface mesh has been adapted, a unit volume mesh is generated with respect to the modified metric $\widetilde{\mathcal{M}}$. In our approach a constrained Delaunay procedure is used to build first an empty mesh (with no internal vertices). Then, based on an edge length analysis, internal nodes are added into the current mesh (most of them coming from the background mesh, at the previous iteration) using the *Delaunay kernel*, extended to the anisotropic case [19].

As pointed out, the classical distance evaluation is replaced using an evaluation related to the local anisotropic metric. Hence, let A and B be two points, the distance between A and B , $d(A, B)$ is now replaced by $l_{\mathcal{M}}(A, B)$ as defined by Equation (3). Then, O_K (the circumcenter of a tetra-

hedron) is computed as the solution of the system:

$$l_{\mathcal{M}}(O_K, P_i) = l_{\mathcal{M}}(O_K, P_j) \quad \forall i, j = 1, 4, i \neq j,$$

and r_K , the circumradius of K , is computed as:

$$r_K = l_{\mathcal{M}}(O_K, P_j).$$

Finally, the Delaunay measure:

$$\alpha_{\mathcal{M}}(P, K) = \frac{l_{\mathcal{M}}(O_K, P)}{l_{\mathcal{M}}(O_K, P_j)} < 1,$$

where $\alpha_{\mathcal{M}}(P, K)$ is used to define the *cavity* of K .

However, as we face a nonlinear system to compute O_K and as the metric is discrete, it is then not so easy to compute the desired length. Therefore, approximations are needed to return to the Euclidean context. To this end, we fix the metric and various approximations can be used.

2.3 Unsteady adaptation scheme

The classical adaptation algorithm is usually composed of four successive steps: (i) solution computation, (ii) error estimation and metric construction, (iii) governed mesh adaptation and (iv) interpolation of the solutions. However, this approach has been proved to be inadapted when dealing with transient (time-dependent) problems [2], mainly because of the impossibility to predict the behavior or the evolution of such phenomena. For stationary problems, mesh adaptation aims to find a fixed point for the couple mesh-solution. The objective is to converge towards both the stationary solution of the problem and the adapted mesh (that equidistribute the interpolation error). In such context, the mesh is always "late" with respect to the phenomenon and thus, if the number of adaptations is too small, the solution is diffused. Moreover, the adaptation introduce a quite large error due to the interpolation of the solution from one mesh (the background mesh) to the other, that impacts the time derivatives and thus introduce an error in phase. To overcome these problems, we have proposed an extension of the classical adaptation scheme, specifically intended for transient problems [2].

The transient fixed point algorithm is composed of two steps, the main adaptation loop and an internal loop in which the transient fixed point problem

is solved (Fig. 5). At each iteration of the main loop, we consider a time period $[t, t + \Delta t]$ in which the solution evolves. From the solution at time t , we compute the solution to time $t + \Delta t$, and the computation is iterated until the desired accuracy is obtained for the solution at $t + \Delta t$. Hence, the solution behavior is predicted in all regions of the domain where it evolves. A metric intersection procedure in time is introduced to adapt the mesh in these regions.

To generate an adapted mesh in the region where the solution advances in time, the metric must reflect this evolution. To this end, the metric must take into account the information given by the initial solution, the final solution and all intermediate solutions. Therefore, all metrics are intersected and the metric tensor at iteration i and the internal loop j is defined as:

$$\mathcal{M}_{i,j} = \bigcap_{k=1}^m \mathcal{M}_{i,j}^k,$$

where $\mathcal{M}_{i,j}^k$ is the k th intermediate metric given by the solution.

3. APPLICATION EXAMPLES

In this section, we present three examples of unstructured mesh adaptation intended to illustrate the various problems described previously. The first example aims at showing anisotropic surface and volume mesh adaptation for an analytical metric specification. The second example will focus on the error estimate for a steady-state Euler computation in two dimensions. Finally, the third example deals with a transient CFD problem on a complex geometry in three dimensions.

3.1 A "not-so-simple" analytical example

In this example, we will show the behavior of the anisotropic adaptation meshing procedure on an analytical function and, more precisely, the metric intersection algorithm. Let us consider the surface in \mathbb{R}^3 defined on $[-1, 1]^3$:

$$f_1(x, y, z) = \tanh((x + 1.3)^{20} (y - 0.3)^9 z),$$

the computational domain is a supertorus defined by the set of equations:

$$\begin{cases} x &= \cos^{n_1}(\theta) (r_0 + r_1 \cos^{n_2}(\phi)), \\ y &= \sin^{n_1}(\theta) (r_0 + r_1 \cos^{n_2}(\phi)), \\ z &= (r_0 + r_1) \sin^{n_2}(\phi), \end{cases}$$

where θ and ϕ vary in $[0, 2\pi]$ and $r_0 + r_1$ (resp. $r_0 - r_1$) represents the external (resp. internal) radius of the torus (here $n_1 = n_2 = 0.2$).

The specificity of this case is that it combines two different metric fields, the first one defined by the function f_1 , the second one related to the geometry of the domain (the intrinsic surface properties). Here, the surface meshing algorithm proceeds by first analysing the surface given an initial (crude) surface triangulation (Fig. 2) in order to construct the geometric metric tensor¹. The number of adaptations has been set *a priori* to 8, the desired error bound is $\varepsilon = 0.0084$. The Hessian of the variable (the surface curvature or the function) is computed a least-squares approximation [16]. Table 1 reports statistics about the initial triangulation and the sequence of adapted meshes, np , ne , nf representing the number of vertices, tetrahedra and triangles, respectively, $\tilde{\varepsilon}$ and ε_{max} represent the average and maximal error measured on the elements.

The distance to the hypersurface is computed on each tetrahedron K by considering the maximal value of the distances between the edge midpoints, the face midpoints and the barycenter of K . The average (resp. maximal) value of the error should be close to the desired error ε on the final adapted mesh, thus validating the proposed error estimate.

It	np	ne	nf	$\tilde{\varepsilon}$	ε_{max}
0	3,850	15,460	4,884	0.0438	1.0000
2	72,090	416,124	14,548	0.0064	0.5673
4	29,651	154,763	17,076	0.0047	0.0555
5	22,782	114,637	16,190	0.0051	0.0454
8	21,063	105,268	15,474	0.0051	0.0345

Table 1. Statistics about the adapted meshes at iterations 0 (initial), 2, 4, 5 and 8 (final) for the analytical example.

¹Needless to say that the analytical function is not used to get the first and second derivatives of the surface at the mesh vertices, discrete approximations are computed instead [16].

Remark. In this example, in the final adaptation, 99.6 % of the elements have an error below the given tolerance ε and the average error $\tilde{\varepsilon} = 0.0051$ is lesser than ε . Moreover, the maximal error ε_{max} is constantly decreasing over the iterations.

3.2 A 2D steady-state problem

This example related to a Euler computation at Mach 3 in a scramjet configuration is typical of numerical simulations in compressible fluids, involving highly anisotropic phenomena (shocks). The aim is to capture the behavior of the physical phenomenon and to emphasize the reduction of the number of degrees of freedom obtained thanks to the anisotropy [9]. The geometry of the computational domain is shown in Fig. 6.

Two series of mesh adaptation have been carried out on this example, isotropic and anisotropic, to be compared. However, the parameters were the same for both computations : 9 adaptations have been performed, each 400 time steps of the Euler solver. The density variable has been chosen to adapt the meshes with the following parameters: $\varepsilon = 0.02$, $h_{min} = 0.01$ m and $h_{max} = 2$ m.

The cartesian surface associated to the density presents a series of steps (shocks) in the domain. The error estimate analyses this surface in \mathbb{R}^3 and provide a 2D anisotropic metric, the desired unit mesh is thus a mesh adapted to this underlying surface. Table 2 reports statistics about the anisotropic and isotropic adapted meshes. Figure 6 compares the initial and adapted meshes (at iterations 2 and 9) with the corresponding isodensity distributions. The anisotropic final mesh contains more than 5 times less vertices than the final isotropic mesh. Accordingly, the anisotropic approach required 14mn (on a PC workstation) *vs.* 2h13mn for the isotropic approach, thus reducing the time by a impressive factor of 9. Moreover, by stretching the elements along the discontinuities, the numerical diffusion due to the Riemann solver has been significantly reduced (Fig. 8).

Remark 3.1. The same test case was already proposed a few years ago. However, the anisotropic mesh adaptation was giving nearly isotropic elements along the shock waves [9]. The result is now improved using the geometric error estimate described in Section 1.

3.3 A 3D transient CFD problem

Finally, the last example will demonstrate the efficiency of the transient adaptation procedure, on a

It	np_a	ne_a	np_i	ne_i
0	8,012	15,275	8,012	15,275
2	5,783	11,055	76,565	151,558
5	9,292	17,983	75,997	150,532
9	15,110	29,569	78,702	155,951

Table 2. Statistics about the initial and adapted meshes for the anisotropic and isotropic cases on the scramjet configuration.

isotropic example². The problem concerns a non linear wave propagation in a complex 3D geometry. This simulation can be seen as a generalisation of the Riemann problem (a shock tube) in higher dimension. More precisely, a Heavyside perturbation is introduced into a uniform field so as to simulate an explosion (a region of high pressure is introduced into the ambient atmosphere). The flow is approached using Euler equations in the conservative forms. A finite volume (for the flow computation) solver is used. Euler equations are solved using an explicit scheme, a four order Runge-Kutta is used for time integration.

In this example, the objective is to compute the solution at physical time $t = 0.1$ sec. The simulation is decomposed into 30 periods (adaptations). At each main iteration, 4 internal iterations are performed to solve the transient fixed point problem. The metric is defined based on the density variations. The adaptation parameters have been set to: $\varepsilon = 0.01$, $h_{min} = 0.3m.$, $h_{max} = 10m.$, for a computational domain size of $85m \times 85m \times 70m$. Table 3 reports statistics about the adapted (isotropic) meshes. Adapted meshes are presented in Figures 9 to 13.

It.	t	np	ne	nf
8	0.027	280,525	1,630,619	40,736
15	0.05	603,644	3,541,268	63,526
23	0.077	739,854	4,326,861	78,816
30	0.1	743,735	4,328,741	87,322

Table 3. Statistics about the adapted meshes for the transient problem.

²Anisotropy is expected soon in 3D.

4. CONCLUSIONS

In this paper, we have presented a global scheme for mesh adaptation in the context of numerical simulations. The proposed approach involves an anisotropic geometric error estimate, surface and volume mesh adaptation algorithms based on discrete metric tensors. It differs from the classical adaptation scheme by integrating a inner loop corresponding to a transient fixed point algorithm.

REFERENCES

- [1] F. ALAUZET ET P.J. FREY (2003), Estimateur d'erreur géométrique et métriques anisotropes pour l'adaptation de maillage. Partie I : aspects théoriques, RR-4759, INRIA Rocquencourt.
- [2] F.ALAUZET, P.J. FREY ET B. MOHAMMADI (2002), Adaptation de maillages pour des problèmes instationnaires, *C.R. Acad. Sci.*, Paris, t 336, Série I.
- [3] E.F. D'AZEVEDO AND B. SIMPSON (1991), On optimal triangular meshes for minimizing the gradient error, *Numerische Mathematik*, **59**(4), 321-348.
- [4] I. BABUŠKA AND W.C. RHEINBOLDT (1978), A posteriori error estimates for the finite element method, *Int. j. numer. methods eng.*, **12**, 1597-1615.
- [5] T.J. BAKER (1997), Mesh adaptation strategies for problems in fluid dynamics, *Finite Elements in Analysis and Design*, **25**(3-4), 243-273.
- [6] R.E. BANK (1997), Mesh smoothing using a posteriori estimates, *Siam J. numer. anal.*, **34**(3), 979-997.
- [7] M. BERZINS (1999), Mesh Quality : A Function of Geometry, Error Estimates or Both ?, *Engineering with Computers*, **15**, 236-247.
- [8] H. BOROUCAKI, F. HECHT, E. SALTEL AND P.L. GEORGE (1995), Reasonably efficient Delaunay based mesh generator in 3 dimensions, *Proc. 4th Int. Meshing Roundtable, Albuquerque, New Mexico*, 3-14.
- [9] H. BOROUCAKI, P.L. GEORGE, F. HECHT, P. LAUG, B. MOHAMMADI ET É. SALTEL, Maillieur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie II : applications, RR INRIA 27 60, déc.
- [10] H. BOROUCAKI, F. HECHT AND P.J. FREY (1998), Mesh gradation control, *Int. j. numer. methods eng.*, **43**(6), 1143-1165.
- [11] H. BOROUCAKI, D. CHAPELLE, P.L. GEORGE, P. LAUG ET P.J. FREY (2001), *Estimateurs d'erreur géométriques et adaptation de maillage*, in *Maillage et adaptation*, P.L. George ed., Série Mécanique et Ingénierie des Matériaux, Méthodes Numériques, Hermès Science, Paris.
- [12] P.G. CIARLET (1991), Basic Error Estimates for Elliptic Problems, in *Handbook of Numerical Analysis*, vol II, Finite Element methods (Part 1), P.G. Ciarlet and J.L. Lions Eds, North Holland, 17-352.
- [13] H.L. DECOUGNY AND M.S. SHEPHARD (1996), Surface Meshing Using Vertex Insertion, *Proc. 5th Int. Meshing Roundtable*, October 10-11, Pittsburgh, PA, 243-256.
- [14] M. FORTIN(2000), Estimation d'erreur a posteriori et adaptation de maillages, *Revue européenne des éléments finis*, **9**(4).
- [15] P.J. FREY AND P.L. GEORGE (2000), Mesh generation. application to finite elements, Hermès Science Publ., Paris, Oxford.
- [16] P.J. FREY(2000), About surface remeshing, In *Proc.of 9th Int. Meshing Roundtable*, New Orleans, LO, USA, 123-136.
- [17] P.J. FREY (2001), Yams : A fully Automatic Adaptive Isotropic Surface Remeshing Procedure, RT-0252, INRIA Rocquencourt.
- [18] P.L. GEORGE (1997), Improvement on Delaunay based 3D automatic mesh generator, *Finite Elements in Analysis and Design*, **25**(3-4), 297-317.
- [19] P.L. GEORGE (2002), *Gamanic3d*, Adaptive anisotropic tetrahedral mesh generator, Technical Report, INRIA.
- [20] R. LÖHNER (1989), Adaptive remeshing for transient problems, *Comput. Methods Appl. Mech. Engrg.*, **75**, 195-214.
- [21] R. LÖHNER (1996), Regriding Surface Triangulations, *Jour. of Comput. Phys.*, **126**, 1-10.
- [22] B. MOHAMMADI (1994), Fluid dynamics computation with NSC2KE - an User-Guide, Release 1.0, *RT INRIA 164*.

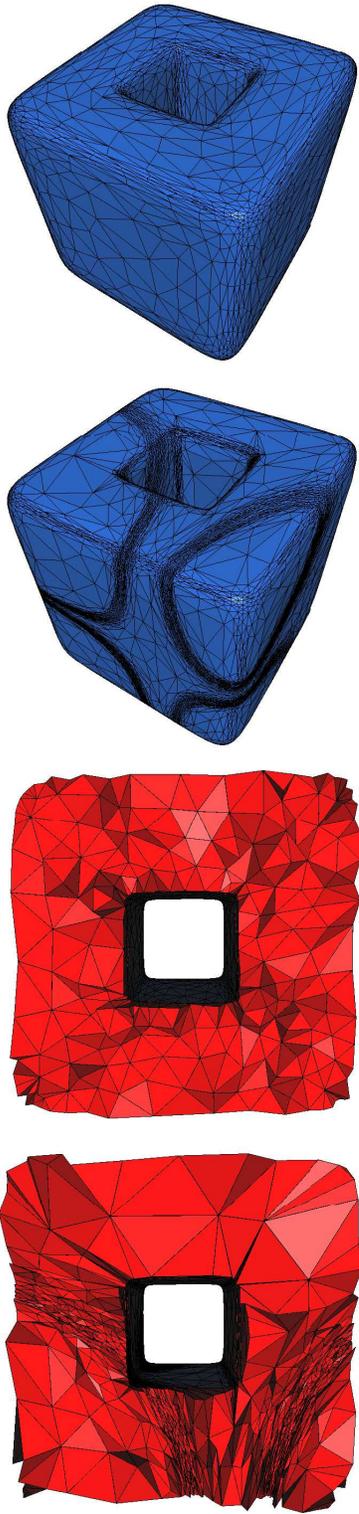


Figure 2. Initial and final adapted surface meshes for the analytical metric. Cut through the volume to show anisotropic tetrahedra within this domain.

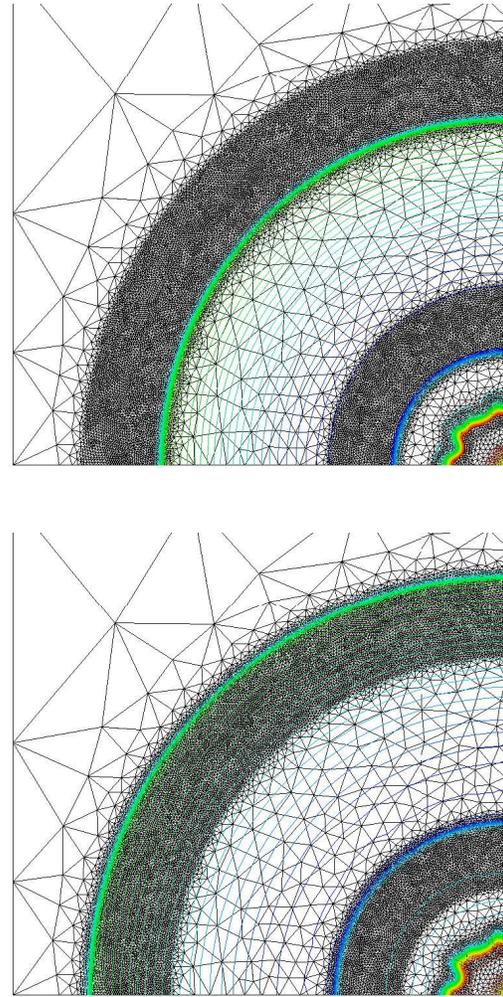


Figure 3. Impact of the metric intersection scheme (initial and final solution during a period).

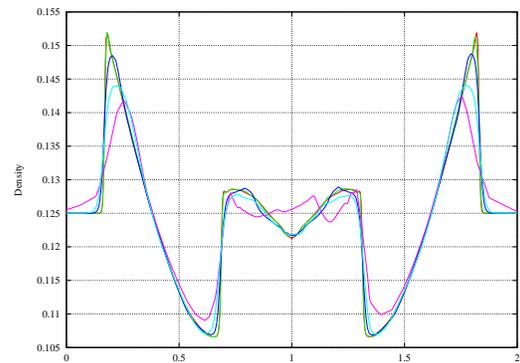


Figure 4. Comparison of the adapted solution with a reference solution (uniform mesh) in red. Green: 6 metric intersection, blue: 2 metric intersection, magenta: only the initial solution, cyan: only the final solution.

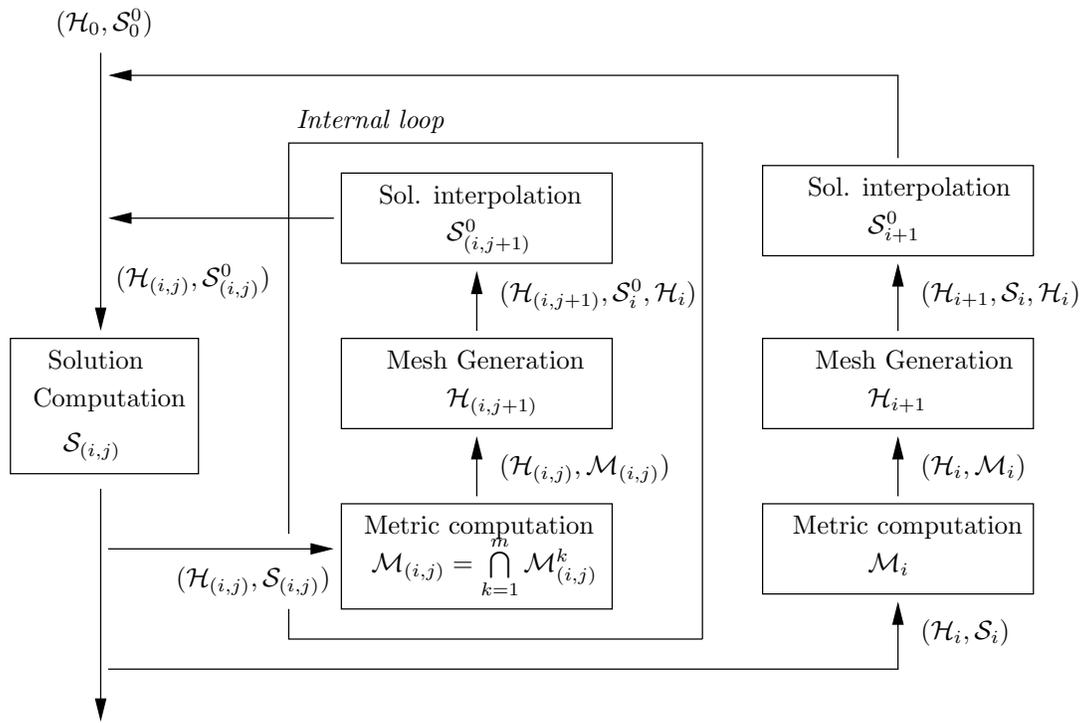


Figure 5. Global overview of the modified mesh adaptation scheme for transient simulations.

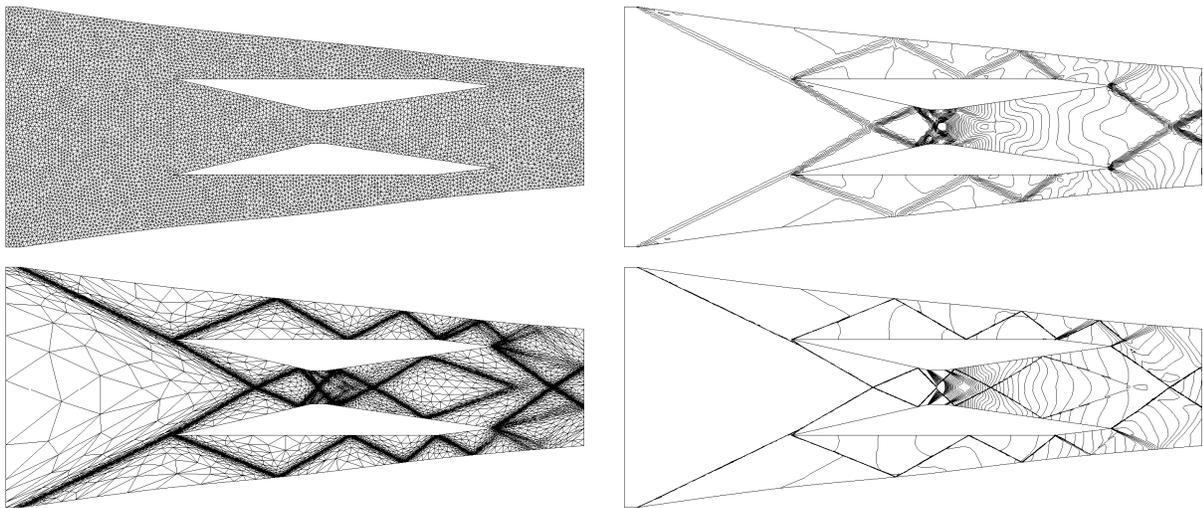


Figure 6. Initial and adapted meshes at the iterations 2 and 9 and corresponding isodensity distributions for the scramjet configuration.

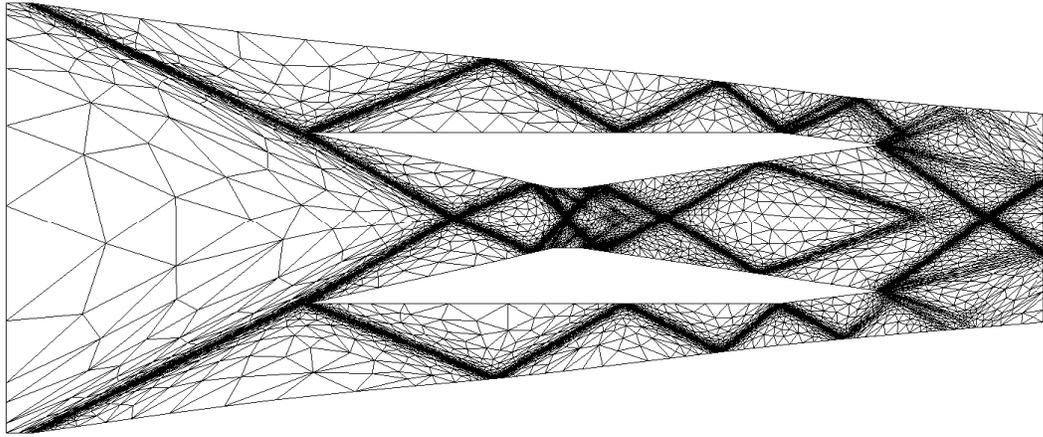


Figure 7. *Final anisotropic mesh for the scramjet configuration.*

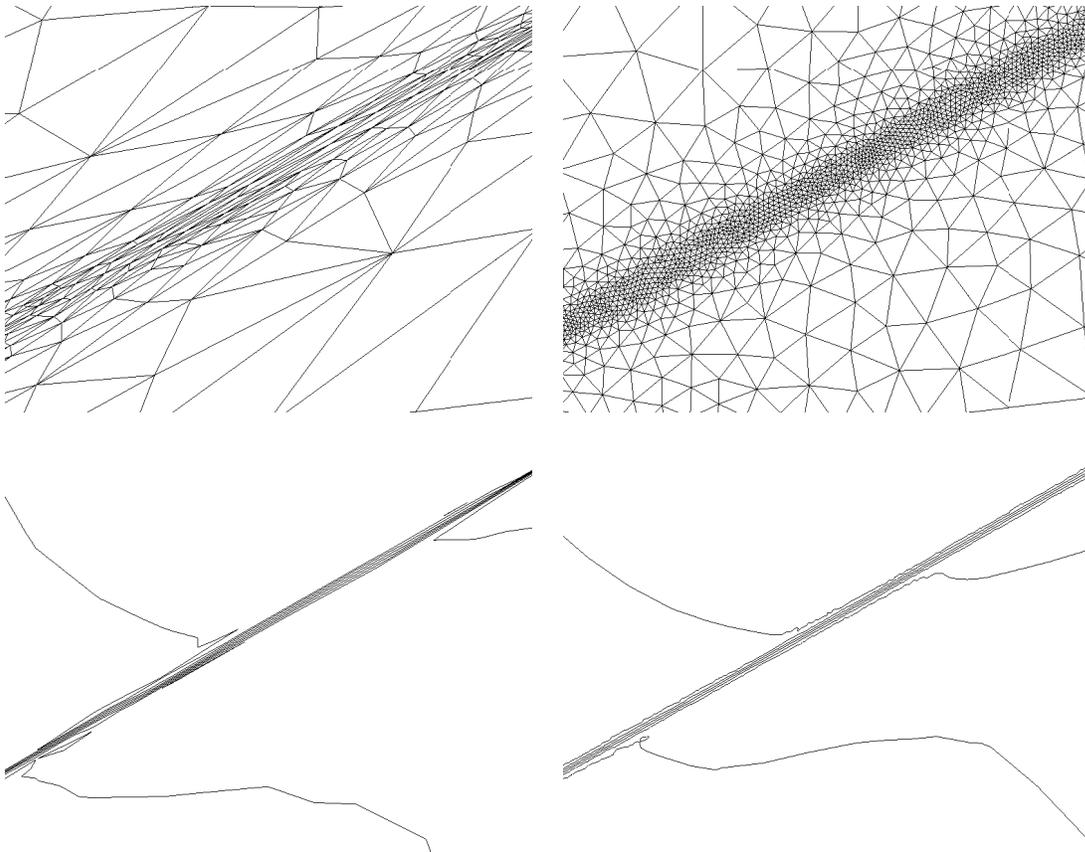


Figure 8. *Anisotropic vs. isotropic mesh comparison for the final meshes (local zoom).*

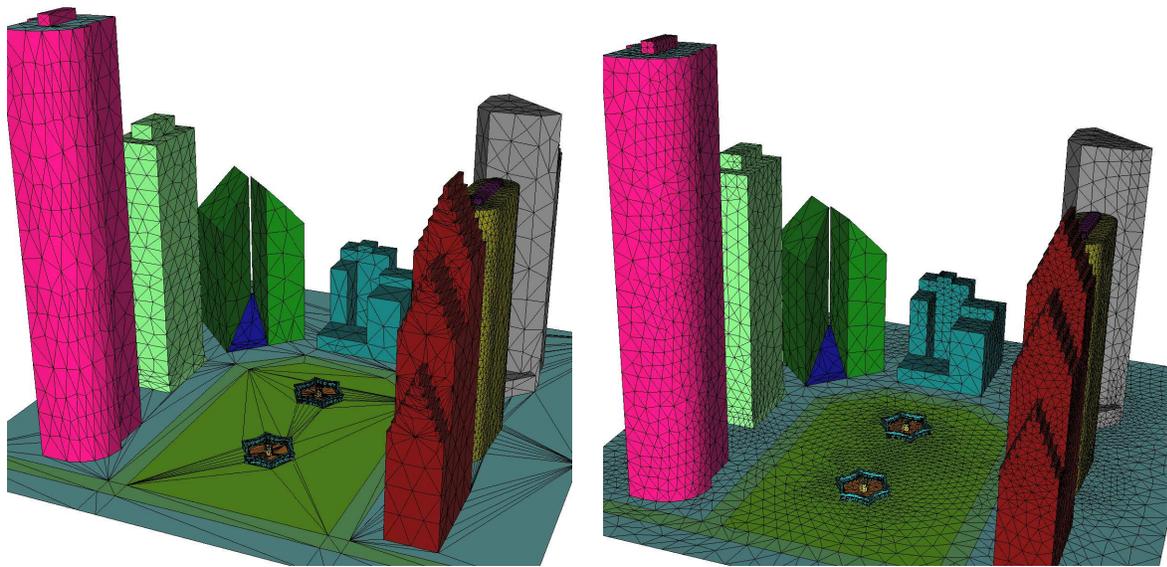


Figure 9. *Geometric surface mesh (left-hand side) and initial computational surface mesh (right-hand side).*

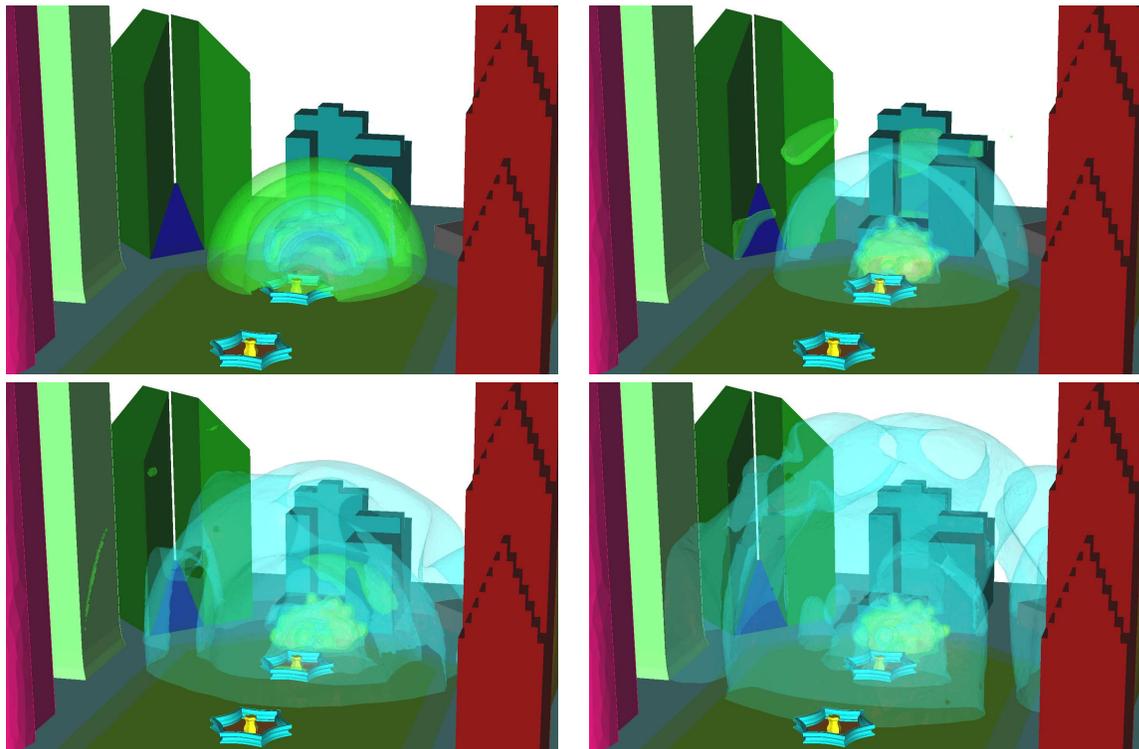


Figure 10. *Isodensity surfaces at times $t = 0.027$ sec, $t = 0.05$ sec, $t = 0.077$ sec and $t = 0.1$ sec.*

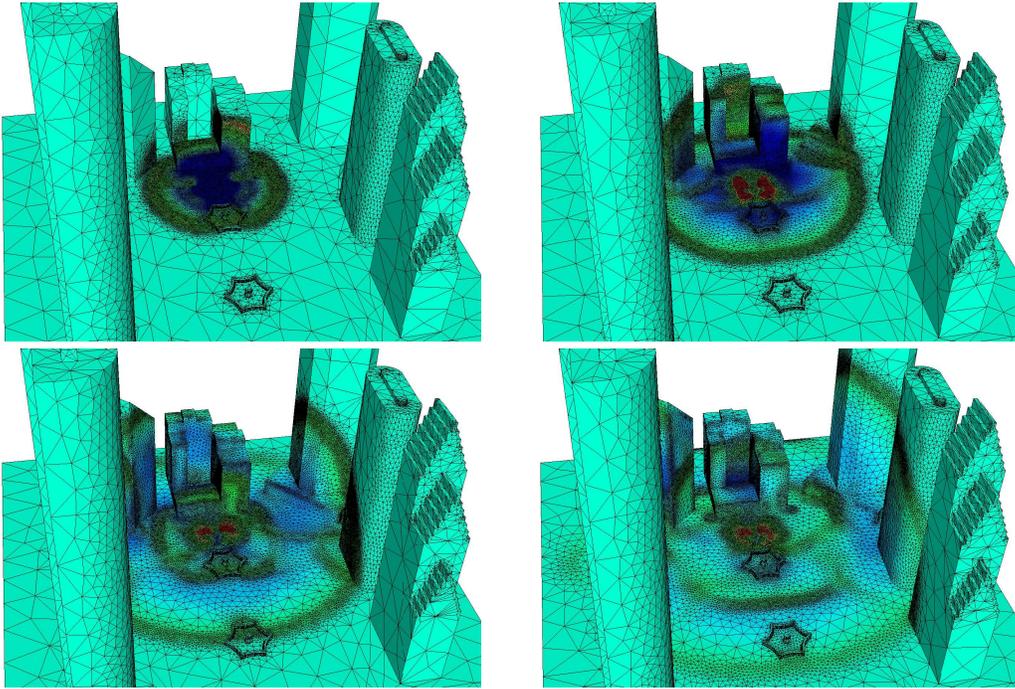


Figure 11. *Isotropic adapted surface meshes and isodensity distributions at times $t = 0.027$ sec, $t = 0.05$ sec, $t = 0.077$ sec and $t = 0.1$ sec.*

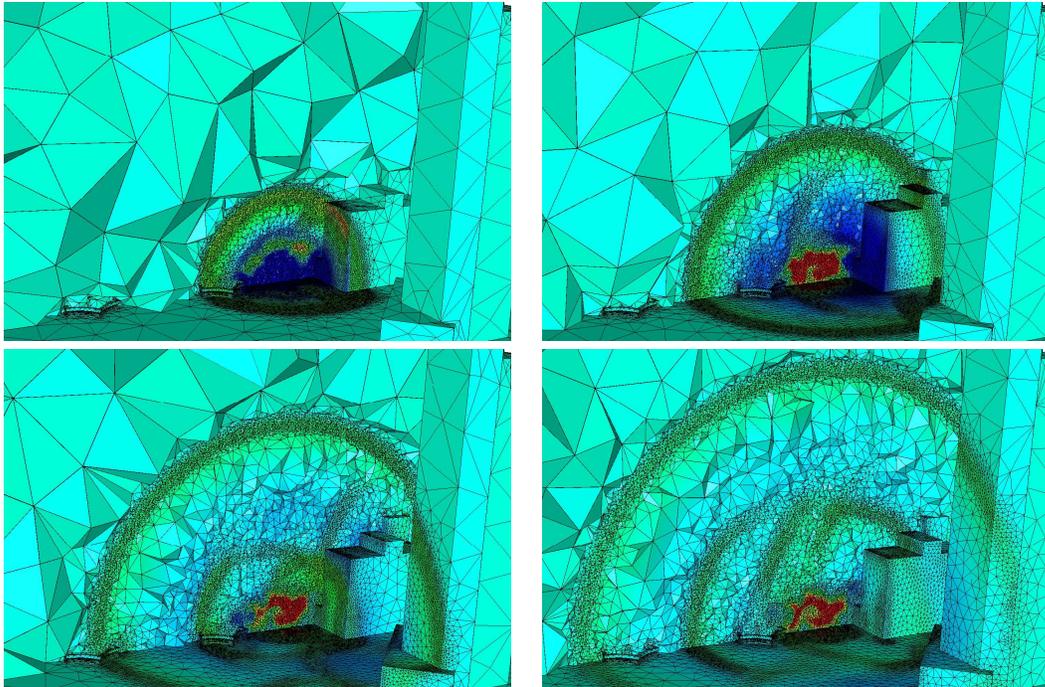


Figure 12. *Isotropic adapted volume meshes and cutting plane through the domain showing the isodensity distributions at times $t = 0.027$ sec, $t = 0.05$ sec, $t = 0.077$ sec and $t = 0.1$ sec.*

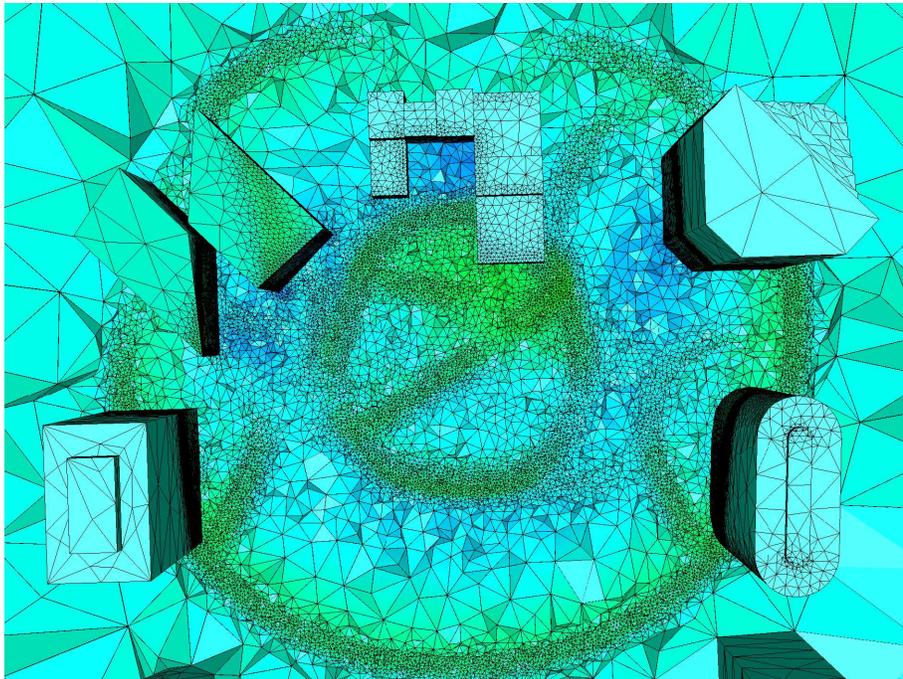
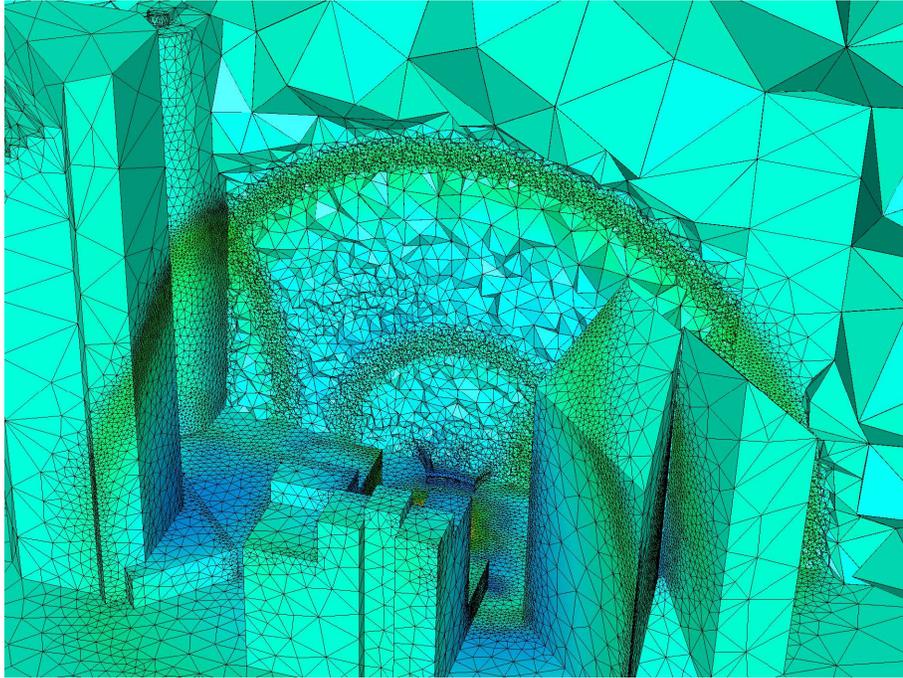


Figure 13. *Two cutting planes through the adapted volume mesh at time $t = 0.077$ sec and corresponding isodensity distributions.*