# TRANSIENT MESH ADAPTATION USING CONFORMING AND NON CONFORMING MESH MODIFICATIONS*

Jean-François Remacle Xiangrong Li Nicolas Chevaugeon
Mark S. Shephard

*Scientific Computation Research Center,*
*Rensselaer Polytechnic Institute,*
*Troy, New York, USA.*
*Corresponding author: remacle@scorec.rpi.edu*

## ABSTRACT

In this paper, we present a method for performing isotropic and anisotropic adaptive computations. The discontinuous Galerkin method that is used for solving transient flow problem is briefly introduced. We show then a general scheme to compute high-order derivatives of discontinuous fields. The Hessian of the density is used for computing a correction indicator. We present three sample problems involving hundred of mesh refinements, both in 2D and 3D.

**Keywords: mesh adaptation, discontinuous Galerkin, anisotropy**

## 1. INTRODUCTION

Transient flow problems involving wave propagation are of great interest in computational fluid dynamics. An accurate tracking of features like moving shocks or fluid interfaces in an Eulerian fashion implies multiple mesh adaptations in order to follow complex features of the flow.

The discontinuous Galerkin method (DGM) is a good candidate for solving our problems of interest. The DGM can be regarded as an extension of finite volume methods to arbitrary orders of accuracy without the need to construct complex stencils for high-order reconstruction. In this paper, we outline the DGM we are using [1] and we introduce a general scheme for computing stable high-order derivatives of discontinuous fields. We will use the Hessian matrix as a correction indicator together with an innovative shock detector.

The aim of this is to be able to do computations involving thousands of mesh adaptations. We present two different methods for doing the mesh adaptation. The first one consists of non-conforming mesh modifications since the DGM does not impose inter-element continuity of the approximated fields. Therefore, do not need to construct complex constraints when we divide elements in a non-conforming way. The second mesh adaptation method is based on conformal mesh modifications: element splitting, edge swapping, edge collapsing and etc. This second adaptation scheme is *a priori* more risky for methods with conservation requirements because of the possible introduction of numerical diffusion during certain critical mesh modifications like edge swapping. Despite this, the second scheme has the advantage of being expandable to anisotropic mesh adaptation and easy to use with continuous finite element basis.

In sections §7. and §6., we present examples of 2D and 3D computations and show results on the advantage of anisotropic mesh refinement. For all examples, we have used the Algorithm Oriented Mesh Database. AOMD is available as open source at `http://www.scorec.rpi.edu/AOMD`. AOMD provides the operations that are necessary to do the mesh adaptation: local mesh modifications, introduction of solver callbacks, access to mesh entities and their classification [2].

*This work was supported by the ASCI Flash Center at the University of Chicago, under contract B341495, by the U.S. Army Research Office through grant DDAD19-01-0655 and the DOE SciDAC program through agreement DE-FC02-01-ER25460.

## 2. DISCONTINUOUS FINITE ELEMENTS FOR SOLVING CONSERVATION LAWS

Consider an open set $\Omega \subset \mathbb{R}^3$ whose boundary $\partial\Omega$ is Lipschitz continuous with a normal $\vec{n}$ that is defined everywhere. We seek to determine $\mathbf{u}(\Omega, t) : \mathbb{R}^3 \times \mathbb{R} \rightarrow \mathsf{L}^2(\Omega)^m = V(\Omega)$ as the solution of a *system of conservation laws*

$$\partial_t \mathbf{u} \; + \; \mathbf{div}\, \vec{\mathbf{F}}(\mathbf{u}) \; = \; \mathbf{r}. \tag{1}$$

Here $\mathbf{div} \; = (\mathrm{div}\,, \ldots, \mathrm{div}\,)$ is the vector valued divergence operator and

$$\vec{\mathbf{F}}(\mathbf{u}) = (\vec{F}_1(\mathbf{u}), \ldots, \vec{F}_m(\mathbf{u}))$$

is the flux vector with the $i$th component $\vec{F}_i(\mathbf{u}) \; : \; (\mathsf{H}^1(\Omega))^m \rightarrow \mathsf{H}(\mathrm{div}\,, \Omega)$. Function space $\mathsf{H}(\mathrm{div}\,, \Omega)$ consists of square integrable vector valued functions whose divergence is also square integrable i.e.,

$$\mathsf{H}(\mathrm{div}\,, \Omega) = \left\{ \; \vec{v} \mid \vec{v} \; \in \; \mathsf{L}^2(\Omega)^3 \; , \; \mathrm{div}\, \vec{v} \in \mathsf{L}^2(\Omega) \; \right\}.$$

With the aim of constructing a Galerkin form of (1), let $(\cdot, \cdot)_\Omega$ and $\langle \cdot, \cdot \rangle_{\partial\Omega}$ denote the standard $\mathsf{L}^2(\Omega)$ and $\mathsf{L}^2(\partial\Omega)$ scalar products respectively. Multiply equation (1) by a test function $\mathbf{w} \in V(\Omega)$, integrate over $\Omega$ and use the divergence theorem to obtain the following variational formulation

$$\begin{aligned}(\partial_t \mathbf{u}, \mathbf{w})_\Omega - (\vec{\mathbf{F}}(\mathbf{u}), \mathbf{grad}\, \mathbf{w})_\Omega + \langle \vec{\mathbf{F}}(\mathbf{u}) \cdot \vec{n}, \mathbf{w} \rangle_{\partial\Omega} \\ = (\mathbf{r}, \mathbf{w})_\Omega \; , \quad \forall \mathbf{w} \in V(\Omega).\end{aligned} \tag{2}$$

Finite element methods (FEMs) involve a double discretization. First, the physical domain $\Omega$ is discretized into a collection of $\mathcal{N}_e$ elements

$$\mathcal{T}_e = \bigcup_{e=1}^{\mathcal{N}_e} e \tag{3}$$

called a mesh. The continuous function space $V(\Omega)$ containing the solution of (2) is approximated on each element $e$ of the mesh to define a finite-dimensional space $V_e(\mathcal{T}_e)$. With discontinuous finite elements, $V_e$ is a "broken" function space that consists in the direct sum of elementary approximations $\mathbf{u}_e$ (we use here a polynomial basis $\mathbb{P}^q(e)$ of order $q$):

$$V_e(\mathcal{T}_e) = \{\mathbf{u} \mid \mathbf{u} \in \mathsf{L}^2(\Omega)^m, \mathbf{u}_e \in \mathbb{P}^q(e)^m = V_e(e)\}. \tag{4}$$

Because all approximation are disconnected, we can solve the conservation laws on each element to obtain

$$\begin{aligned}(\partial_t \mathbf{u}_e, \mathbf{w})_e - (\vec{\mathbf{F}}(\mathbf{u}_e), \mathbf{grad}\, \mathbf{w})_e + \langle \mathbf{F}_n, \mathbf{w} \rangle_{\partial e} \\ = (\mathbf{r}, \mathbf{w})_e \; , \quad \forall \mathbf{w} \in V_e(e).\end{aligned} \tag{5}$$

Now, a discontinuous basis implies that the normal trace $\mathbf{F}_n \; = \; \vec{\mathbf{F}}(\mathbf{u}) \cdot \vec{n}$ is not defined on $\partial e$. In this situation, a *numerical flux* $\mathbf{F}_n(\mathbf{u}_e, \mathbf{u}_{e_k})$ is usually used on each portion $\partial_{e_k}$ of $\partial e$ shared by element $e$ and neighboring element $e_k$. Here, $\mathbf{u}_e$ and $\mathbf{u}_{e_k}$ are the restrictions of solution $\mathbf{u}$, respectively, to element $e$ and element $e_k$. This numerical flux must be continuous, so $\vec{\mathbf{F}} \in \mathsf{H}(\mathrm{div}\,, \Omega)^m$, and be consistent, so $\mathbf{F}_n(\mathbf{u}, \mathbf{u}) = \vec{\mathbf{F}}(\mathbf{u}) \cdot \vec{n}$. With such a numerical flux, equation (5) becomes

$$\begin{aligned}(\partial_t \mathbf{u}_e, \mathbf{w})_e - (\vec{\mathbf{F}}(\mathbf{u}_e), \mathbf{grad}\, \mathbf{w})_e \\ + \sum_{k=1}^{n_e} \langle \mathbf{F}_n(\mathbf{u}_e, \mathbf{u}_{e_k}), \mathbf{w} \rangle_{\partial e_k} \\ = (\mathbf{r}, \mathbf{w})_e \; , \quad \forall \mathbf{w} \in V_e(e),\end{aligned} \tag{6}$$

where $n_e$ is the number of faces of element $e$. Only the normal traces have to be defined on $\partial e_k$ and several operators are possible [3, 4]. It is usual to define the trace as the solution of a Riemann problem across $\partial e_k$. Herein, we consider problems with strong shocks [4,5]. An exact Riemann solver is used to compute the numerical fluxes and a slope limiter [6] is used to produce monotonic solutions when polynomial degrees $q > 0$ are used.

The choice of a basis for $V_e(e)$ is an important issue in constructing an efficient method. Because the field is discontinuous, there is substantial freedom in the selection of the elemental basis. Here, we chose the $\mathsf{L}^2$-orthogonal basis described in [1] as a basis of $P(e)$:

$$P(e) = \{b_1, \ldots, b_k\}$$

where

$$(b_i, b_j)_e = \delta_{i,j}.$$

For the time discretization, we use the local time stepping procedure described in [7] that allows to use time steps more the 20 times bigger that the classical stability limit of explicit schemes.

We will present the results of some compressible inviscid flow problems involving the solution of the Euler equations [8] by a DG method. The three-dimensional Euler equations have the form (1) with

$$\mathbf{u} \;\; = \;\; \{\rho, \rho v_x, \rho v_y, \rho v_z, E\}^t \tag{7}$$

$$\begin{aligned}\vec{\mathbf{F}}(\mathbf{u}) \;\; = \;\; & \{\rho\vec{v}, \rho v_x \vec{v} + P\vec{e}_x, \\ & \rho v_y \vec{v} + P\vec{e}_y, \rho v_z \vec{v} + P\vec{e}_z, \\ & (\rho E + P)\vec{v}\}^t,\end{aligned} \tag{8}$$

$$\mathbf{r} \;\; = \;\; \mathbf{0}. \tag{9}$$

Here $\rho$ is the fluid density, $\vec{v}$ the velocity, $E$ the internal energy, $P$ the pressure and $\vec{e}_x$, $\vec{e}_y$ and $\vec{e}_z$ are the unit vectors in the $x$, $y$ and $z$ directions, respectively. An equation of state of the form $P = P(\rho, E)$ is also necessary to close the system. The DG method and the associated software [1] may be used for any equation of state which only enters the numerical method through the calculation of the numerical flux. Here, we have chosen the perfect gas equation of state

$$P \; = \; (\gamma - 1) \; \rho \; \left[ E - \frac{\|\vec{v}\|^2}{2} \right] \tag{10}$$

with the gas constant $\gamma = 1.4$.

## 3. COMPUTING HESSIAN'S OF DISCONTINUOUS SOLUTIONS

It is common in Computational Fluid Dynamics to use second order derivatives

$$H_{i,j}(u) = \frac{\partial^2 u}{\partial x_i \partial x_j}$$

of a flow variable $u$ in order to compute an error indicator. In [9], authors provide an *ad hoc* procedure for the computation of $H(u)$ for first order continuous finite elements solutions. Here, we provide a general approach which can be applied for higher order and/or discontinuous finite elements.

In case of classical $\mathsf{C}^0$ finite elements, the computation of the gradient of the finite element solution is straightforward because of the $\mathsf{C}^0$ continuity of the field. In case of discontinuous finite elements, gradients have a contribution due to inter-element jumps. In order to recover some control on the gradients, we compute them using a discontinuous Galerkin technique i.e. find $w \in V_e \backslash \mathbb{R} = V_e^0$ such that

$$\operatorname{grad} u - \operatorname{grad} w = 0 \ \text{ in } \ \Omega \tag{11}$$

The fact that we have taken of the constant part out of $V_e^0$ allows (11) to have a unique solution. Characterization of space $V_e^0(e)$ is done by choosing $V_e^0(e) = \{b_2, \ldots, b_k\}$, the constant part of a $\mathsf{L}^2$ orthogonal space being contained into its first function $b_1$. We consider the following formulation: find $w \in V_e^0(e)$ such that

$$\left( (\operatorname{grad} u - \operatorname{grad} w), \operatorname{grad} w' \right)_e = 0 \ \ \forall w' \tag{12}$$

This equation is solved in each element in order to recover a stable gradient $\operatorname{grad} w$. To stabilize $\operatorname{grad} w$, we will integrate (12) by parts in order to have a term that reflects the jumps of $u$ on $\partial e$.

$$\begin{aligned} &((w-u), \operatorname{div}(\operatorname{grad} w'))_e \qquad (13)\\ &+ \sum_{k=1}^{n_e} \langle \{u\} - \{w\}, \operatorname{grad} w' \rangle_{\partial e_k} = 0 \quad \forall w'. \end{aligned}$$

Fields $u$ and $w$ on $\partial_e$ are not defined directly. There we use the average fluxes

$$\{u\} = \vec{n}\, \frac{u_e + u_{e_k}}{2}.$$

in evaluating (13). This choice is not as "natural" as the one we made for the numerical flux in the hyperbolic case where this kind of Riemann problem has well known solution in terms of simple waves [8]. A second integration by parts yields jumps of $u$ and $w$ across $\partial e$:

$$\begin{aligned} &(\operatorname{grad} u - \operatorname{grad} w), \operatorname{grad} w')_e \\ &+ \sum_{k=1}^{n_e} \langle ([\![u]\!] - [\![w]\!]), \operatorname{grad} w' \rangle_{\partial e_k} = 0 \quad \forall w'. \end{aligned} \tag{14}$$

where

$$[\![u]\!] = \vec{n}\, \frac{u_e - u_{e_k}}{2}$$

is the half flux jump. Without any other assumption on the regularity of $w$, the solution of (14) is $u = w$. By choosing $w$ in a space such that

$$([\![w]\!], \operatorname{grad} w')_e = 0 \quad \forall w', \tag{15}$$

i.e. choosing $w$ so that there is no flux jump (in a weak sense) of the field through faces, equation (14) transforms to

$$B(w, w') = L(u, w') \ \ \forall w' \tag{16}$$

with

$$B(w, w') = (\operatorname{grad} w, \operatorname{grad} w')_e$$

and

$$L(u, w') = (\operatorname{grad} u, \operatorname{grad} w')_e + \sum_{k=1}^{n_e} \langle [\![u]\!], \operatorname{grad} w' \rangle_{\partial e_k}$$

The hypothesis in (15) requires further consideration: assumption (15) is similar to the one used for building the Crouzeix-Raviart finite element family [10]. Those elements generate stable gradients and that's what we are looking for.

For second order derivatives, we proceed exactly the same way. If $u$ is the variable we intend to compute second order derivatives, we do it in 2 steps:

- First step, the gradients:

$$B(w, w') = L(u, w')$$

- Second step: the Hessian:

$$B(h_j, w') = L\left( \frac{\partial w}{\partial x_j}, w' \right) \ \ , j = 1, 2, 3$$

where $\operatorname{grad} h_j$ is the $j^{th}$ row of the Hessian matrix. The interest of this technique is that it allows us to build any higher order derivative of a field by a general procedure. Even if the number of elements that are involved to compute a higher-order derivative is increased at each step, the procedure (16) remains the same. Gradients are constructed using face neighbor elements. The Hessian is also constructed using gradients at neighbor elements but those were constructed using their own neighbors. Discrete versions of forms $L$ and $B$ can be re-used so that the procedure is fast and easy to code. As an example, let us consider the following function

$$f(x, y) = x^3 + x^2 y^2 + 3y^3$$

whose Hessian is easy to compute. We have $\mathsf{L}^2$-projected this function on an unstructured triangular mesh and have used $q = 0$ for the finite element approximation. Results of numerical computation of the Hessian components $H_{ij}$ on a line $0 < x < 1$ , $y = 0.5$ are shown on Figure 1.

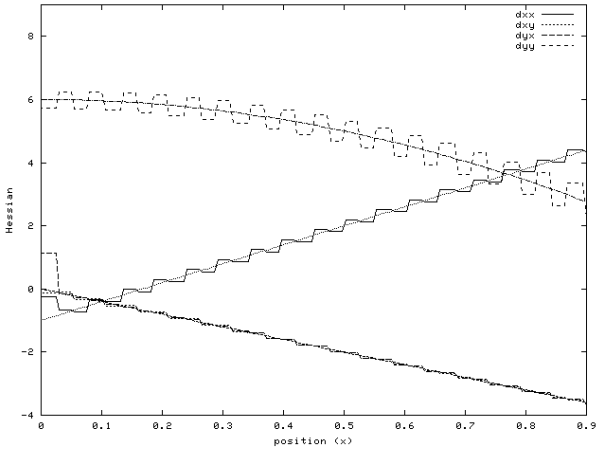


**Figure 1: Numerical computation of the Hessian of a piecewise constant function, the figure shows the results on an triangular mesh (40 points per length).**

## 4. A CORRECTION INDICATOR AND MESH SIZE FIELD

Using Hessians for tracking shocks may be a pragmatic technique but it cannot be considered as an error estimator:

- the hypothesis of smooth convergence (interpolation theory) of finite element solution cannot be used in shocks and other discontinuities,
- higher order solutions may have large second order derivatives which are perfectly resolved.

In order to track shocks and other complex features of transient flows, we build a smoothness indicator. Then, we will use the Hessian to compute the direction of shocks and other discontinuities.

### *4.1 Smoothness indicator*

Let us consider 2 elements $e$ and $e_k$ sharing the common face $\partial e_k$. The numerical solution $\mathbf{u}$ is in general discontinuous along inter-element boundaries. We denote by $\mathbf{u}_e$ and $\mathbf{u}_{e_k}$ value of $\mathbf{u}$ on $e$ and $e_k$ respectively. Since the method is of order $q$, we expect $\|\mathbf{u}_e - \bar{\mathbf{u}}\|_{\mathsf{L}^2} = \mathcal{O}(h_e^{q+1})$ with $h_e$ being the size of element $e$ and $\bar{\mathbf{u}}$ being the exact solution of (1). In the DGM, there exists a quantity that exhibits faster convergence than $\mathsf{L}^2$ norm of the error: the error is super-convergent in average at outflow boundaries [11]. If we assume that $\partial e_k$ is an inflow boundary for $e_k$ (outflow for $e$), we have that

$$I_e = \int_{\partial e_k} |\mathbf{u}_e - \mathbf{u}_{e_k}| \, dl = \mathcal{O}(h_e^{2q+1}) \tag{17}$$

Result (17) is only valid in smooth regions. In rough regions, the super-convergence does not apply and the accuracy of the numerical solution is of order $\mathcal{O}(1)$ i.e. amplitude of jumps of the solution across inter-element boundaries are of the same order of magnitude as the solution itself. Then, in rough regions, $I_e = \mathcal{O}(h_e)$. Using those results, we build the following smoothness indicator:

$$\epsilon_e = \frac{I_e}{\int\limits_{\partial e_k} \mathbf{u}_e \, dl} \tag{18}$$

We distinguish 2 cases. In presence of a discontinuity, we have that $\epsilon_e = \mathcal{O}(1)$. Practically, we decide that, whenever $\epsilon_e > 0.1$, the element is considered to be crossed by a discontinuity. In smooth regions, $\epsilon_e = \mathcal{O}(h_e^{2q})$ i.e. this value converges to 0 with adaptation. It is then possible, using $\epsilon_e$, to detect regions where the numerical scheme is unable to capture strongly varying features.

Using the smoothness indicator (18), we are able to build a size field. Our goal is, classically, to obtain an uniform distribution of $\epsilon_e = \epsilon^*$ for all elements. In any element $e$ of a smooth region, we compute the desired size $h_e^*$ of elements of the optimum mesh in the region delimited by $e$ like

$$h_e^* = h_e \left( \frac{\epsilon}{\epsilon^*} \right)^{2q}.$$

In rough regions, convergence rate is lost and we can not define an appropriate mesh size based on the smooth convergence rate. Alternatively, we can decide *a priori* the resolution of discontinuities we want to achieve to reasonable resolve and isolate the jumps. If the smoothness indicator $\epsilon_e > 0.1$, we chose $h_e^* = h_s$ where $h_s$ is a give small desired mesh size.

In order to illustrate the power of $\epsilon_e$, we consider the following example. We consider a square domain of size $1 \times 1$ centered at $x = 0$ and $y = 0$. The problem is initially divided into four quadrants. Quadrant 1 is the upper right, 2 the upper left, 3 the lower left and 4 the lower right. All boundary conditions are transmitting (we copy the interior data perpendicular to the boundary). We initialize each quadrant with the quantities given in Table 4.1. In this problem four

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $\rho$ | 1.0 | 2.0 | 1.0 | 3.0 |
| $v_x$ | 0.75 | 0.75 | -0.75 | -0.75 |
| $v_y$ | -0.5 | 0.5 | 0.5 | -0.5 |
| $P$ | 1.0 | 1.0 | 1.0 | 1.0 |

**Table 1: Initial conditions for the four-contact Riemann problem.**

contact discontinuities are rotating around the center of the square creating vortex sheets. This example has been chosen to prove the efficiency of the smoothness indicator. We used $\epsilon_e$ for both refinement and smoothness indications. When a discontinuity is detected, i.e. $\epsilon_e > 0.1$, the limiter described in [6] is used to remove oscillations from the solutions. Everywhere else, i.e. in smooth regions, the DGM scheme is kept intact with its full accuracy. Figure 2 shows solution for

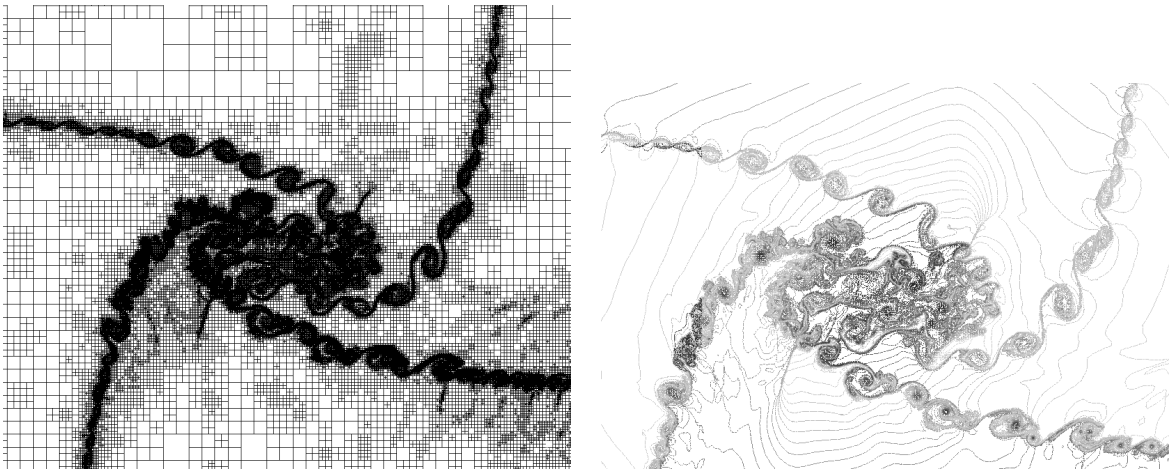

**Figure 2: Refined mesh and density contours at $t = 0.3$ for the four-contact problem. Views show a zoom of the central area for the mostly refined computation.**

a effective mesh of $2560 \times 2560$ grid points which would require more that a hundred millions of degrees of freedom to have the same accuracy on a uniform mesh. The actual mesh at $t = 0.3$ that is shown on the left side of Figure 2 has four million degrees of freedom after having performed 300 mesh adaptations (one adaptation every $0.001$ seconds). At small times $t = 0.1$, contact discontinuities are stable on their major parts. After that, Kelvin-Helmholtz's instabilities grow starting at the center of the square and reaching the whole interface. The center of the domain is now filled by a turbulent mixing zone. The cascade to small scales is not moderated by viscosity effects because we use the Euler's equations. For that reason, the more refinement we will allow, the smaller features we will get.

### *4.2 Anisotropic metric*

In this paper, the rationale is that a primary goal of anisotropic mesh refinement is to efficiently capture discontinuities. In this case, there is one only direction for anisotropy: the one orthogonal to discontinuities. All the rest of the field, apart of iscontinuities, is then supposed to be isotropic.

The anisotropy of the mesh is favorable in 2 ways:

- reducing the number of elements for the same accuracy,
- aligning the element faces with discontinuities which has the effect of reducing the numerical dissipation of the Riemann solver

In order to build an anisotropic metric field, we compute the Hessian of one of the variables, the fluid density $\rho$ e.g:

$$H(\rho) = R \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} R^T$$

with $R$ being the rotation matrix diagonalizing $H$ and $\lambda_1$,$\lambda_2$ and $\lambda_3$ being the eigenvalues of the $H$.

We assume here that $|\lambda_1| > |\lambda_2|$ and $|\lambda_2| > |\lambda_3|$. We build the metric $M$ for element $e$ like

$$M = R \begin{bmatrix} 1/h_s^2 & 0 & 0 \\ 0 & |\lambda_1/\lambda_2|/h_s^2 & 0 \\ 0 & 0 & |\lambda_1/\lambda_3|/h_s^2 \end{bmatrix} R^T$$

Metric $M$ has two properties:

- Specify an ellipsoidal directional variation of desired edge length in physical space;
- Define a transformation to a space where the desired anisotropic mesh is isotropic and normalized.

Let $\Delta\vec{v}$ represent the vector associated with a mesh edge. The length of $\Delta\vec{v}$ with respect to $M$ is then computed by:

$$L_M(\Delta\vec{v}) = \sqrt{\Delta\vec{v}\, M\, \Delta\vec{v}^T}$$

with $L_M(\Delta\vec{v}) = 1$ if vector $\Delta\vec{v}$ is of the desired length. Since, geometrically, $\Delta\vec{v}\, M\, \Delta\vec{v}^T = 1$ describes an ellipsoid in physical space, the desired length variation with respect to $M$ follows an ellipsoidal distribution.

Due to symmetry and positive definiteness, metric $M$ can always be decomposed:

$$M = Q\, Q^T$$

where

$$Q = R \begin{bmatrix} 1/h_s & 0 & 0 \\ 0 & \sqrt{|\lambda_1/\lambda_2|}/h_s & 0 \\ 0 & 0 & \sqrt{|\lambda_1/\lambda_3|}/h_s \end{bmatrix}$$

which is the representation of a linear transformation from physical space to a rotated distorted space where desired mesh is normalized and isotropic.

In general, $M$ varies over the domain. The length and volume with respect to the metric field can be computed by:

$$L_M(\mathsf{M}_i^1) = \int_0^1 \sqrt{\Delta\vec{v}\, Q(t)\, Q(t)^T\, \Delta\vec{v}^T} dt \tag{19}$$

$$V_M(\mathsf{M}_j^3) = \int_{\mathsf{M}_j^3} |Q(x,y,z)|\, dV \tag{20}$$

where $\Delta\vec{v}$ is the vector associated with edge $\mathsf{M}_i^1$ in physical space, and $Q$ represents the specified transformation matrix over the considered mesh entity.

## 5. MESH ADAPTATION

We want to perform adaptive computations with the aim of tracking transient features including sharp fronts. In this paper, we want to compare two kind of mesh adaptation procedures: conforming and non-conforming. Mesh adaptation is performed in both cases in terms of local mesh modifications. One cavity triangulation $\mathcal{C}$, i.e. a set of mesh entities that form a connected volume, is replaced by another cavity triangulation $\mathcal{C}'$ with the same closure. Formally, we write

$$\mathcal{T}^{n+1} = \mathcal{T}^n + \mathcal{C}' - \mathcal{C}. \tag{21}$$

where $\mathcal{T}^n$ denotes the mesh before the local mesh modification and $\mathcal{T}^{n+1}$ is the mesh after the local mesh modification. In the kernel of each mesh modification, we ensure that there is a place when both cavity triangulations are present by doing (21) in two steps:

$$\mathcal{T}' = \mathcal{T}^n + \mathcal{C}'. \tag{22}$$

$$\mathcal{T}^{n+1} = \mathcal{T}' - \mathcal{C}. \tag{23}$$

$\mathcal{T}'$ represents the mesh that is topologically incorrect but both cavity triangulations $\mathcal{C}$ and $\mathcal{C}'$ are present so that we can insert a callback to the solver. Here, we call back the DGM solver that executes a $\mathsf{L}^2$ projection of the the solution from $\mathcal{C}$ to $\mathcal{C}'$ *without losing conservation* i.e. with doing the projection so that mass, momentum's and energy are conserved during the process in the local region covered by the cavity.

### 5.1 Conforming adaptation

The overall procedure for conforming mesh adaptation consists of:

- Properly mark edges to be split in terms of the mesh size field and refine the mesh using refinement templates;
- Collapse short mesh edges with respect to the mesh size field to coarsen or fix up the mesh;
- Eliminate sliver elements with respect to the mesh size field to improve connectivity;
- Repeat above steps until the mesh size field is satisfied or no more improvement possible.

We consider the mesh satisfying the mesh size field if: (i) all its edge lengths in the transformed space fall into interval $[0.6, 1.4]$, and (ii) no sliver element exists in the transformed space.

The reason for using interval $[0.6, 1.4]$ is to ensure that the two new edges from a bisection will not be short edges so that oscillation between refining and coarsening will be prevented, and unit length is in the middle of the interval since the desired edge length is normalized to one in the transformed space. Note that the interval to be $[0.6, 1.4]$ is not unique. Intervals other than $[0.6, 1.4]$ can also be used as long as they will not cause oscillation.

Note that we do not consider here vertex repositioning in our mesh modification procedure. We intend to perform mesh adaptation in the context of transient computations. We think that vertex repositioning could potentially degrade the solution by introducing numerical diffusion during the solution transfer. This will be investigated in forthcoming work.

#### 5.1.1 Refinement

We refine the mesh to desired size level in several iterations. The refinement algorithm can be described as follows:

1. Find $L_{max}$, the maximum edge length of the mesh in the transformed space;
2. Mark all mesh edges longer than $max(1.4, \alpha L_{max})$ in the transformed space as edges to be split, where $\alpha$ is a given factor between $0.5 \sim 1.0$;
3. Split all marked edges at their middle points in the transformed space;
4. Split all adjacent faces and regions of these marked edges using refinement templates with proper diagonal edges created;
5. Eliminate short edges possibly created in refinement;
6. Repeat above steps until $L_{max} < 1.4$.

To maintain mesh quality during refinement, the algorithm only marks a set of long mesh edges in the transformed space in each iteration, then splits these edges at their middle points in the transformed space, as well as all adjacent faces and regions of these edges using refinement templates [12]. Since refinement templates may create short edges in case the initial mesh quality is poor with respect to the mesh size field (see figure 5 for an example), a following short edge collapsing step is needed before the next refinement iteration.

When applying refinement templates, we consider all possible surface triangulation options (42 options) of splitting a tetrahedron so that no additional edge is over refined. Of these options, four requires creating an interior vertex since we can not triangulate the prism polyhedron as illustrated in Figure 3(a), and several have the freedom to select a diagonal edge to create (refer to Figure 4). To maintain the mesh quality of the refined mesh, it is critical to create the diagonal edges such that the introduction of interior vertex is prevented as many as possible, and in case multiple diagonal options, the shortest diagonal edge in the transformed space is always created.

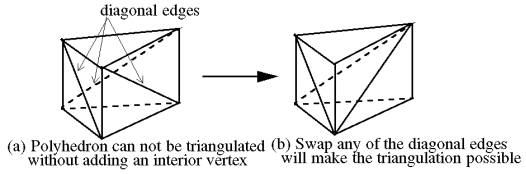


**Figure 3: Select the diagonal edge to avoid inserting interior vertices.**

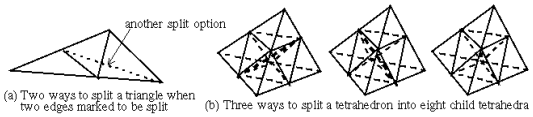


**Figure 4: Multiple options to create diagonal edges.**

Figure 5 gives a 2D example to demonstrate the refinement algorithm. In this example, it takes three iterations to refine the two initial triangles to desired size level. In the first iteration, only two edges (indicated by circles) are split, which creates a short edge since one of the initial triangle has poor quality. Collapsing the short edge is needed, and it can be seen that the combination of split and collapse not only maintains but improves the mesh quality. The second iteration only split the longest edge since it is much longer than others. The third iteration splits eight long edges that are in close length.

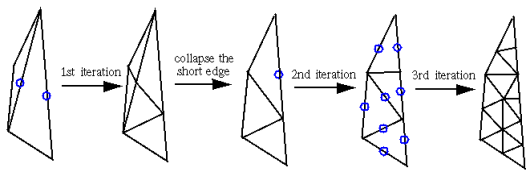


**Figure 5: 2D example of the refinement algorithm.**

### 5.1.2 Collapsing short edges

We consider an edge as short if its length in the transformed space is less than 0.6. Short edges may exist in the given initial mesh, or created during the application of refinement templates and swap operations.

Two local mesh modification operators are used to eliminate short edges:

- edge collapsing;
- compound operator [13] (the chain of multiple simple mesh modifications).

The basic operation is edge collapsing, which removes either vertex of the short edge from the mesh by collapsing it onto the vertex at the other end [12]. Compound operations are investigated only if the edge collapsing is invalid (negative volume) or not acceptable. The compound operation first collapse the short edge, then apply one or several swap (or collapse) operations to eliminate all tetrahedra that become unacceptable due to the initial collapsing operation. The swap (or collapse) operation(s) is identified by analyzing these unacceptable tetrahedra. Collapse operation is first attempted in case the invalid/unacceptable tetrahedron has a short edge. Swap operation is attempted in case the invalid/unacceptable tetrahedron is a sliver. For instance in the example given in figure 6, collapsing $\mathsf{M}_i^0$ to $\mathsf{M}_j^0$ is not possible since $\mathsf{M}_k^3$ becomes flat. However, it is easy to determine the modification that swaps edge $\mathsf{M}_l^1$ to eliminate flat element $\mathsf{M}_k^3$. The compound operator that collapses $\mathsf{M}_i^0$ to $\mathsf{M}_j^0$, then swaps $\mathsf{M}_l^1$ allows vertex $\mathsf{M}_i^0$ to be removed from the existing mesh.

To prevent the possible oscillation among refining, collapsing short edges and eliminating slivers, we consider any edge collapsing and compound operation unacceptable if it creates much longer mesh edge or decrease local mesh quality too much in the transformed space.

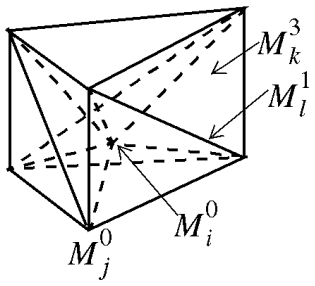


**Figure 6: eliminating $\mathsf{M}_i^0$ by compound operation.**

### 5.1.3 Eliminating sliver elements

Sliver element is characterized by very small volume without short bounding edges in the transformed space. Since the existing of slivers can dramatically reduce time step and degrade results, it is critical to eliminate them.

As indicated in figure 7, we distinguish two types of slivers:

**Type I** two opposite edges are almost intersected.

**Type II** one vertex is close to the centroid of its opposite face;

For type I, the key mesh entity in determining the local mesh modification to eliminate the sliver is a pair of mesh edges (as indicated by circles); For type II, the key mesh entity is the vertex indicated by a black bullet and the face opposite to the vertex.

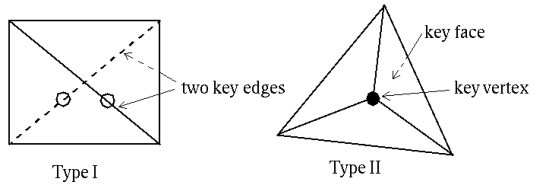


**Figure 7: Two types of slivers and associated key entities.**

We use three local mesh modification operators to eliminate slivers in the priority order as listed. The next operator is attempted only if all previous operators are not acceptable.

1. Delete the sliver and reclassify new boundary mesh entities in case the sliver is next to model boundary and it is valid to remove it;
2. Edge swapping. In particular, for a sliver of type I, we check swapping either of the two key mesh edges; For a sliver of type II, we check swapping any edge that bounds the key mesh face. If more than one swap operations are possible, the one that leads to better result mesh quality is selected;
3. The chain of split(s) and collapse operation. Specifically, for a type I sliver, it first splits both key mesh edges, then collapse the new interior edge; For a type II sliver, it first splits the key mesh face, then collapse the new interior edge.

On Figure 8, we have adapted a mesh based on an analytical metric field. The metric field represents the intersection of two spheres.

### *5.2 Non-conforming adaptation*

Non-conforming adaptation consists of splitting elements independently, leading to the creation of hanging nodes. One refinement template is needed which simplifies greatly the process. The non conforming adaptation is easily applicable to hybrid grids composed of a mixture of tetrahedra, hexahedra and prisms. The different levels of the mesh are stored in memory so that the coarsening procedure consists simply in retrieving upper level meshes locally. This procedure is fast and simple. One major advantage of the method is that projections between cavities $\mathcal{C}$ and $\mathcal{C}'$ can be done without introducing any loss of precision:

- in case of cell splitting, the projection is an identity operator so that no loss of accuracy is observed,
- in case of cell unsplitting, the error on $\mathcal{C}$ is small so that loss of accuracy is small.

As a smoothing procedure, we have decided to allow only one level refinement between two neighboring cells.

## 6. A TWO-DIMENSIONAL EXAMPLE

Consider the parallel Mach 3 flow of a gas in a channel where a step is impulsively inserted. In order to generate a more complex flow, we also insert a quadrilateral object into the channel. The channel is of length 3, height 2 and the step is situated at $x = 0.6$ and of height $0.2$. The initial conditions are $P = 1$, $\rho = 1$ and $\vec{v} = (M_s\sqrt{\gamma}, 0)$.

We have solved this problem using the anisotropic conforming mesh refinement technique, starting from an initial unstructured triangular mesh with an uniform mesh size of $0.1$. The final time of the computation was $1.2$ sec and the mesh was refined every $0.01$ second, which makes a total of $120$ mesh refinements. The parameter $h_s$ was chosen as $h_s = 0.005$.

Figure 9 show mesh evolution during time together with density profiles. The main advantage of the anisotropic mesh refinement technique is the ability to align elements with shocks, allowing the numerical scheme to produce a minimum of numerical diffusion. Figure 9 shows highly well captured shocks. On Figure 10, we see zooms of mesh refinement at shocks intersections.

## 7. A THREE-DIMENSIONAL EXAMPLE

References [1, 14, 15] and the example of Figure 2 demonstrate that non-conforming refinement is an effective procedure to track complex flow features like shocks or vortex sheets when used in combination with discontinuous spatial discretizations. The non-conforming strategy is fast (less than $5\%$ of the total computation time) and does not introduce any loss of accuracy, even when it is used hundreds of times like in the example of Figure 2. In fact, we will not expect conforming isotropic mesh refinement to compete with non-conforming refinement for such problems. However, the anisotropic mesh refinement can provide improvement to the non-conforming technique because of its directional control. However, we must consider that, in the conforming case, the refinement procedure takes approximatively $30\%$ of the total computational time in the current explicit code.

As a first attempt to demonstrate the efficiency of our transient anisotropic refinement strategy, we have computed a

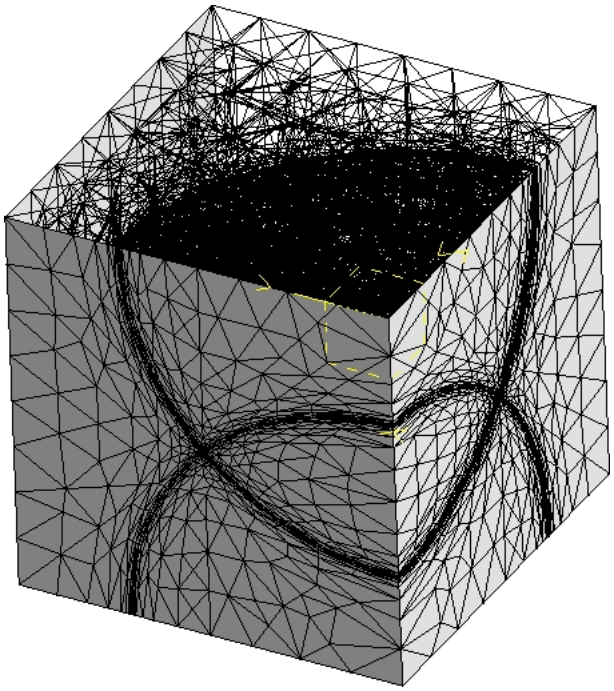

**Figure 8: Anisotropic adapted mesh based on an analytical metric field.**

simple problem: a traveling planar shock moving at Mach 10. Boundary conditions are set to those corresponding to the exact motion of a Mach 10 shock. Physical parameters for the gas ahead of the shock are $P_1 = 1$ and $\rho_1 = 1.4$. The Rankine-Hugoniot relations

$$v_s = M_s\sqrt{\gamma P_1/\rho_1} = 10,$$

$$P_2/P_1 = (2\gamma M_s^2 - (\gamma - 1))/(\gamma + 1),$$

$$\rho_2/\rho_1 = (\gamma + 1)M_s^2/((\gamma - 1)M_s^2 + 2),$$

and

$$\rho_1 v_s = \rho_2(v_s - v_2)$$

are used to compute post shock conditions. The shock is propagating in a rectangular box. We have used a value of $h_s = 0.05$ while the initial mesh was uniform with a mesh size of $0.4$ i.e. a reduction of a factor of $8$ to the mesh in the shock. In the non-conforming case, we use $3$ levels of refinement which also corresponds to a reduction of $8$ of the element sizes in the shocks.

Figure 12 and 11 show meshes and density contours for both non-conforming and anisotropic conforming refinement techniques. Some important improvements have been achieved by using anisotropic mesh refinement:

- The number of elements is significantly smaller in the anisotropic case;
- The shock is much better captured in the anisotropic case, even if the size of the elements on the direction normal to the shock is similar. On Figure 11, we see that the pre and post-shock noises are largely reduced in comparison with the non-conforming case. This is due to the fact that those element faces that are aligned with the shock will introduce minimal numerical dissipation in the Riemann solver (the real Riemann problem is not 1D but 3D);
- C.P.U. time for computation is, in this case, $\mathcal{O}(5)$ times smaller in the anisotropic case, even with the refinement procedure being more expensive. Our local time stepping procedure [7] has allowed us to use time steps $\mathcal{O}(20)$ times bigger that the CFL limit in both cases. It has taken one hour of C.P.U. on a $1.4$ GHz Intel Pentium 4 to reach $t = 0.5$, involving $100$ anisotropic mesh adaptations;
- In general, it seems that it is possible to use numerous mesh adaptations while predicting correctly the position of shocks, even using conforming mesh modifications including edge swapping's.

## 8. CONCLUSIONS

This paper has presented a new procedure to build high-order derivatives of discontinuous fields. Then, we have shown how to construct an error indicator based on super-convergence of the DGM at downwind faces of elements. Combined with the Hessian, we have developed a method to compute a metric field that serves as input to the mesh

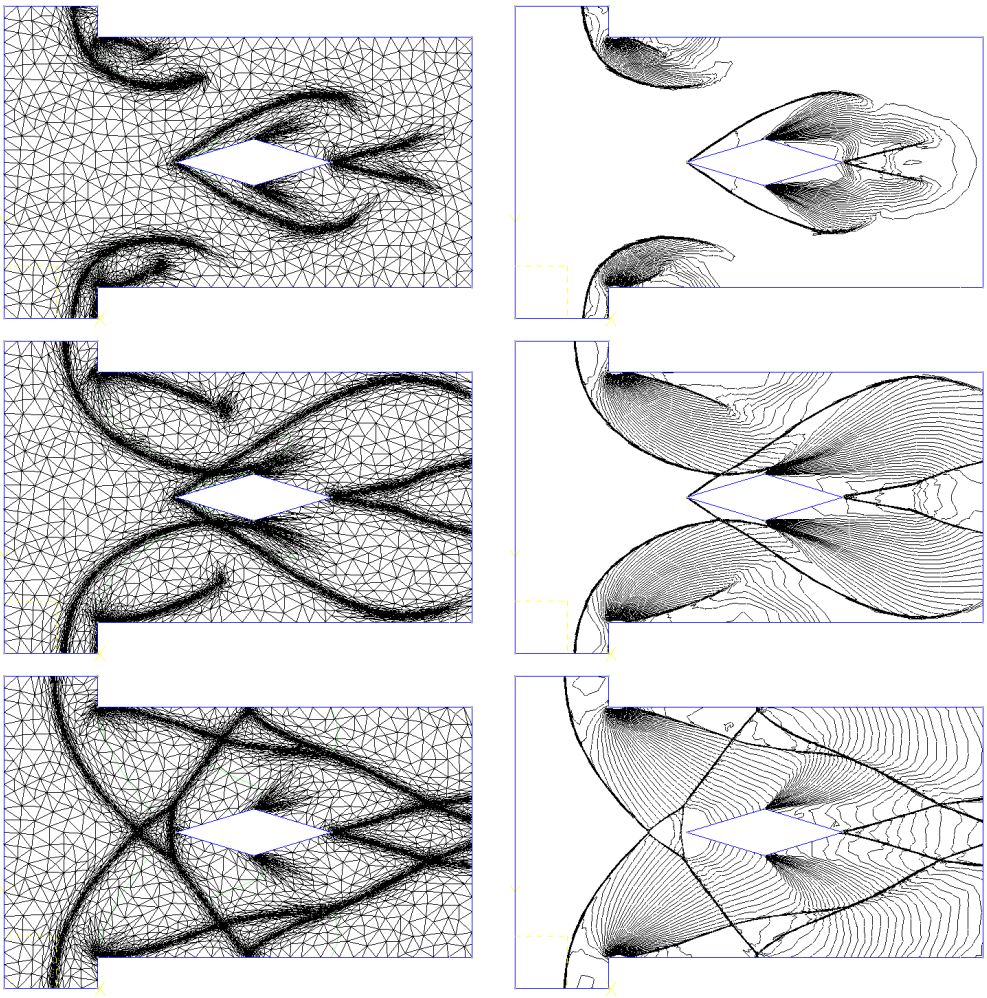

**Figure 9: Mesh (left) and density contours (log scale) of the 2D backward facing step at times $t = 0.2$, $t = 0.4$ and $t = 1.2$.**

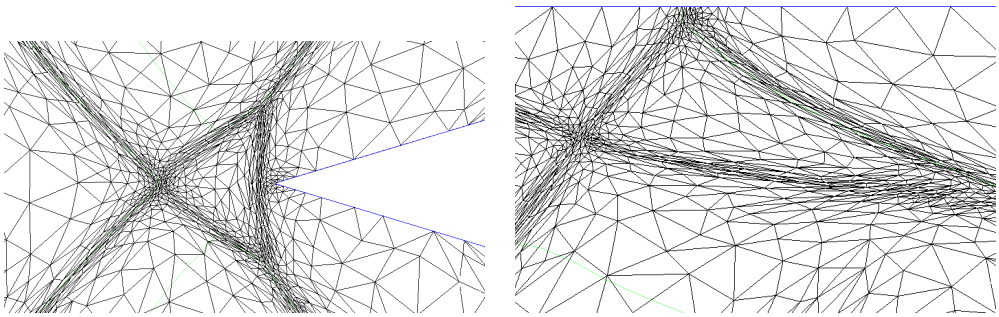

**Figure 10: Zooms of the mesh at $t = 1.2$.**

adaptation procedure. Finally, we have then applied non-conforming and conforming mesh refinement to transient compressible flow problems and have found that anisotropic mesh refinement is advantageous to track discontinuous features of flows like shocks: it produces less diffusion and sharper shocks for the same mesh resolution through discontinuities, while needing less computer resources.

The mesh adaptation procedure that we have developed is

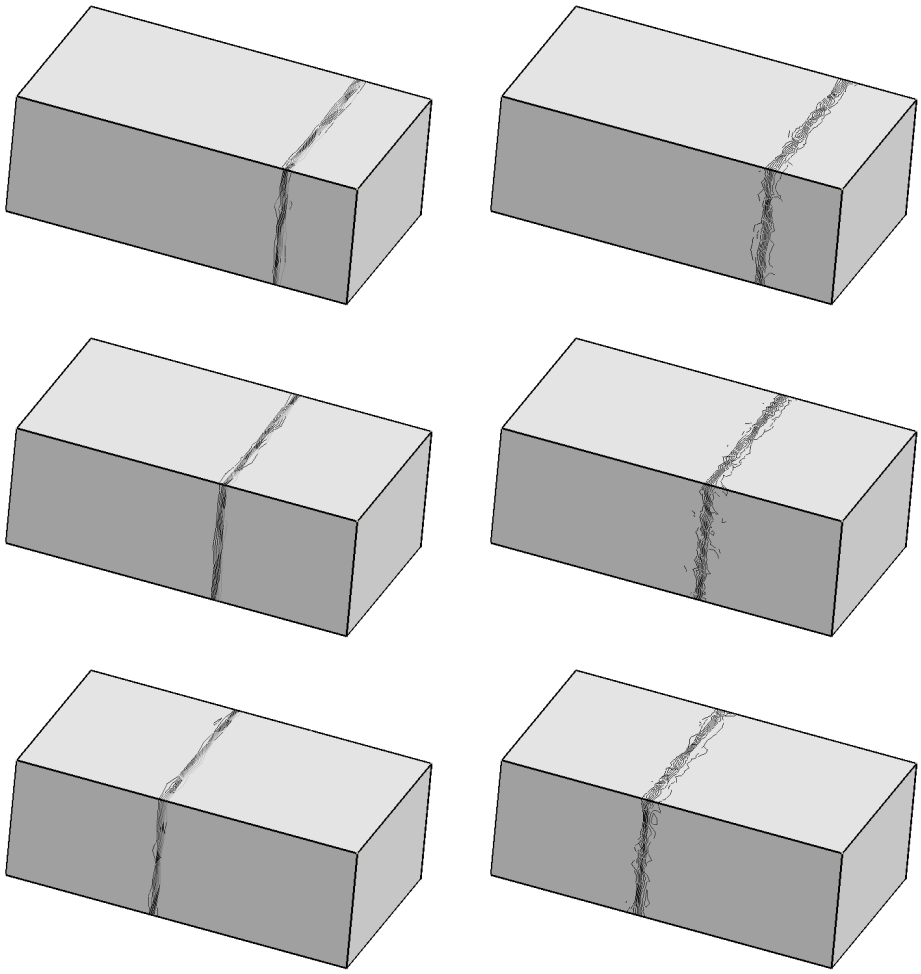

**Figure 11: Density contours after $20$ (top), $40$ (middle) and $60$ (bottom) refinements. The problem has $\mathcal{O}(150,000)$ degrees of freedom for the anisotropic case (left) and $\mathcal{O}(450,000)$ degrees of freedom for the non-conforming case (right).**

able to produce highly anisotropic meshes when the adaptation is driven by an analytic metric field (see Figure 8 for example). Our numerical method, the DGM, is able to handle large number of refinements while preserving conservativity, for both non-conforming and conforming adaptations (see Figures 2 and 11 for example). The next point which need further investigations is the application of our method on real 3D problems with complex geometries.

## References


[1] Remacle J.F., Flaherty J.E., Shephard M.S. "An Adaptive Discontinuous Galerkin Technique with an Orthogonal Basis Applied to Compressible Flow Problems." *SIAM Journal on Scientific Computing, in press*, 2002

[2] Remacle J.F., Shephard M.S. "An Algorithm Oriented Mesh Database." *International Journal for Numerical Methods in Engineering*, 2002. Accepted

[3] Van Leer B. "Flux vector splitting for the Euler equations." Tech. rep., ICASE Report, NASA Langley Research Center, 1995

[4] Woodward P., Colella P. "The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks." *Journal of Computational Physics*, vol. 54, 115–173, 1984

[5] Colella P., Glaz H.M. "Efficient Solution Algorithms for the Riemann Problem for Real Gases." *Journal of Computational Physics*, vol. 59, 264–289, 1985

[6] Biswas R., Devine K.D., Flaherty J.E. "Parallel Adaptive Finite Element Method for Conservation Laws." *Applied Numerical Mathematics*, vol. 14, 255–283, 1984

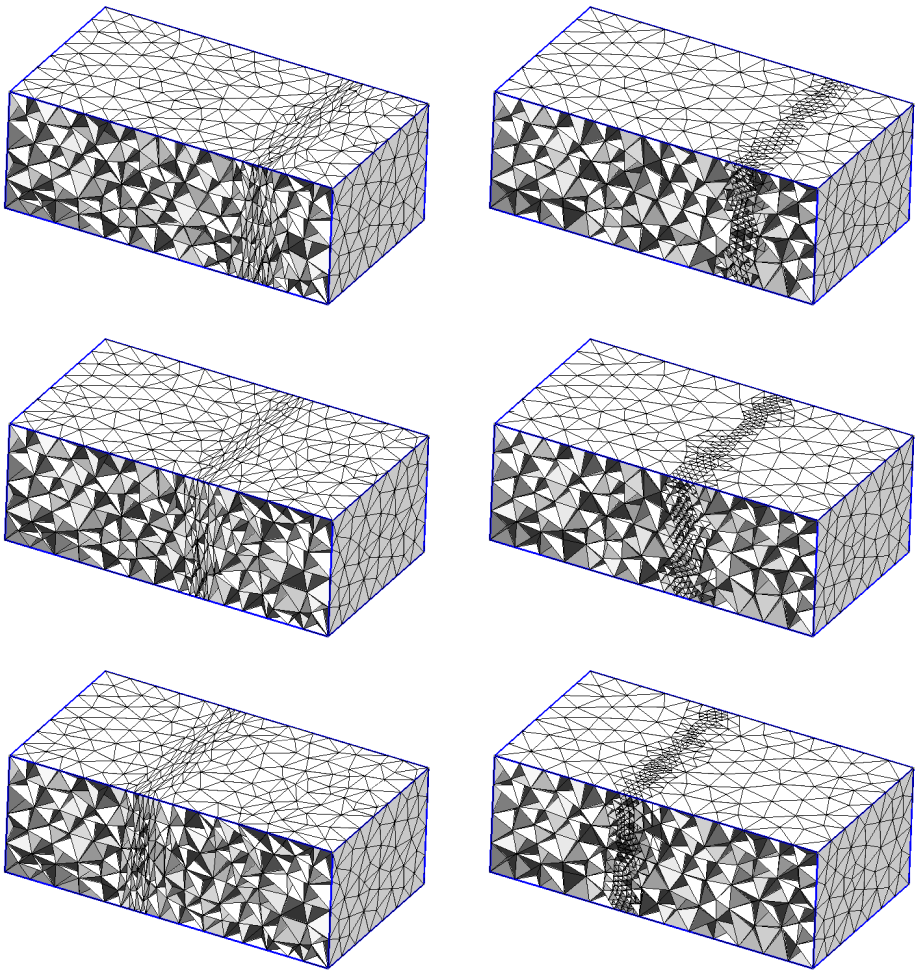

**Figure 12: Meshes after $20$ (top), $40$ (middle) and $60$ (bottom) anisotropic (left) and non-conforming(right) refinements. The triangular faces classified on the front model face have been hidden in order to see interior mesh faces.**


[7] Remacle J.F., Pinchedez K., Flaherty J.E., Shephard M.S. "An efficient local time stepping-Discontinuous Galerkin scheme for adaptive transient computations." *Computer Methods in Applied Mechanics and Engineering*, 2002. Accepted

[8] LeVeque R. *Numerical Methods for Conservation Laws*. Birkhäuser-Verlag, 1992

[9] George P.L., Hecht F. "Non isotropic grids." J. Thompson, B.K. Soni, N.P. Weatherill, editors, *Handbook of Grid Generation*. CRC Press, 1999

[10] Braess D. *Finite Elements : Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge Univ. Pr., 1997

[11] Krivodonova L., Flaherty J.E. "Error Estimation for Discontinuous Galerkin Solutions of Two-Dimensional Hyperbolic Problems." *Advances in Computational Mathematics*, 2002. Submitted

[12] de Cougny H.L., Shephard M.S. "Parallel refinement and coarsening of tetrahedral meshes." *International Journal for Numerical Methods in Engineering*, vol. 46, no. 7, 1101–1125, 1999

[13] Li X., Shephard M.S., Beall M.W. "Accounting for curved domains in mesh adaptation." *International Journal for Numerical Methods in Engineering*, 2002. Accepted

[14] Remacle J.F., Klaas O., Flaherty J.E., Shephard M.S. "A Parallel Algorithm Oriented Mesh Database." *to appear in Engineering With Computers*, 2002

[15] Remacle J.F., Flaherty J.E., Shephard M.S. "Parallel Algorithm Oriented Mesh Database." *Tenth International Meshing Roundtable*. 2001