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 finally present two 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 i.e., basically, as a 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 in non-conforming mesh modifications since the DGM does not impose inter-element continuity of the approximated fields. Therefore, we 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, vertex collapsing. 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 extendable to anisotropic mesh adaptation and easy to use with continuous finite element basis.

In section §7., we present two example computations. The first one is a four-contact Riemann problem. In this 2D problem, we will investigate how non-conforming mesh adaptation compares with isotropic conforming adaptation in terms of numerical diffusion. Then, we will solve the classical backward facing step both in 2D and 3D.

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 ith 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 Ω 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 Ω 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 ∂e. In this situation, a *numerical flux* $\mathbf{F}_n(\mathbf{u}_e, \mathbf{u}_{e_k})$ is usually used on each portion ∂_{e_k} of ∂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 ∂e_k and several operators are possible [3, 4]. It is usual to define the trace as the solution of a Riemann problem across ∂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 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.

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 [8], 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 C^0 finite elements, the computation of the gradient of the finite element solution is straightforward because of the 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

$$\mathrm{grad}\, u - \mathrm{grad}\, w = 0 \;\; \mathrm{in} \;\; \Omega \tag{7}$$

The fact that we have taken of the constant part out of V_e^0 allows (7) 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 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

$$\left((\operatorname{grad} u - \operatorname{grad} w), \operatorname{grad} w'\right)_e = 0 \ \ \forall w' \tag{8}$$

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 (8) by parts in order to have a term that reflects the jumps of u on ∂e.

$$\begin{aligned} &((w-u), \operatorname{div}(\operatorname{grad} w'))_e \qquad (9)\\ &+\sum_{k=1}^{n_e} \langle \{u\} - \{w\}, \operatorname{grad} w' \rangle_{\partial e_k} = 0 \quad \forall w'. \end{aligned}$$

Fields u and w on ∂_e are not defined directly. There we use the average fluxes

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

in evaluating (9). 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 [9]. A second integration by parts yields jumps of u and w across ∂e:

$$\begin{aligned} &(\operatorname{grad} u - \operatorname{grad} w), \operatorname{grad} w')_e \\ &+\sum_{k=1}^{n_e} \langle ([\![u]\!] - [\![w]\!]), \operatorname{grad} w' \rangle_{\partial e_k} = 0 \quad \forall w'. \end{aligned} \tag{10}$$

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 (10) is $u = w$. By choosing w in a space such that

$$([\![w]\!], \operatorname{grad} w')_e = 0 \quad \forall w', \tag{11}$$

i.e. choosing w so that there is no flux jump (in a weak sense) of the field through faces, equation (10) transforms to

$$B(w, w') = L(u, w') \ \ \forall w' \tag{12}$$

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 (11) requires further consideration: assumption (11) 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 (12) 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 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.

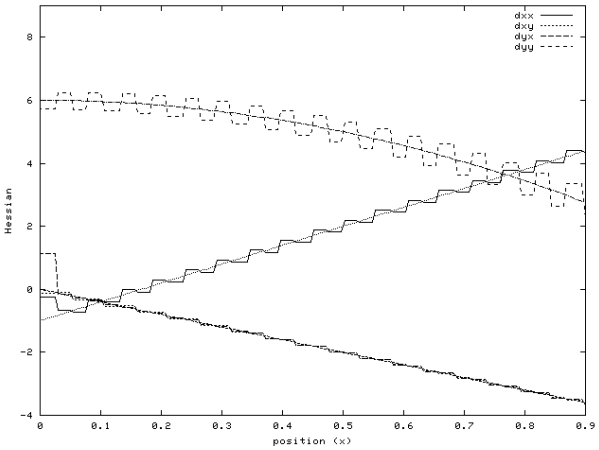

Figure 1: Numerical computation of the Hessian of a piecewise constant function, the figure shows the results on an triangular mesh (40 points per length).

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 in shocks and other discontinuities is wrong,
- 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 correction indicator ϵ_e, that is able to track zones of high second order derivatives:

$$\epsilon_e = \frac{h_e^2 \, |H(\rho)|}{\bar{\rho}} \tag{13}$$

where $|H(\rho)|$ is the maximum eigenvalue of the Hessian of the density, $\bar{\rho}$ is the average value of ρ on e (the first coefficient of the expansion) and h_e the size of element e. If $\epsilon_e = \mathcal{O}(1)$, we apply a minimum size for the mesh. $\epsilon_e << 1$, we apply a maximum size.

In case of anisotropic mesh adaptation, (13) can be generalized to:

$$\epsilon_e = \frac{\Delta\vec{v} \, |H(\rho)| \, \Delta\vec{v}^T}{\bar{\rho}} \tag{14}$$

where $\Delta\vec{v}$ represents a vector associated with a mesh edge, and

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

is a positive definite symmetric matrix with R being the rotation matrix diagonalizing the Hessian and λ_1,λ_2 and λ_3 being the eigenvalues of the Hessian.

For a given correction indicator ϵ_e, a metric M can be constructed such that the desired length of edge associated with $\Delta\vec{v}$ is unity with respect to the metric. Thus:

$$M \overset{\text{def.}}{=} \frac{|H(\rho)|}{\bar{\rho} \, \epsilon_e}$$

The length of $\Delta\vec{v}$ with respect to M is then computed by:

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

with $L_M(\Delta\vec{v}) = 1$ if vector $\Delta\vec{v}$ is in desired length.

The metric M specifies the directional variation of desired edge 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. Consider a unit vector $\vec{e}$, the desired edge length along $\vec{e}$ specified by M is:

$$h(M, \vec{e}) = \frac{1}{\sqrt{\vec{e} \, M \, \vec{e}^T}} \tag{15}$$

Based on its definition, M can always be decomposed:

$$M = QQ^T$$

where

$$Q = \frac{1}{\sqrt{\bar{\rho} \, \epsilon_e}} \begin{bmatrix} \sqrt{\lambda_1} & 0 & 0 \\ 0 & \sqrt{\lambda_2} & 0 \\ 0 & 0 & \sqrt{\lambda_3} \end{bmatrix} R^T$$

therefore

$$L_M(\Delta\vec{v}) = (\Delta\vec{v}Q) \cdot (\Delta\vec{v}Q)^T$$

So Q defines a transformation from physical space to a rotated distorted space, where the geometry of desired length distribution is a unit sphere.

In general, because M varies over the domain, for two vector $\vec{a}$, $\vec{b}$ associated with the two vertices of edge M_i^1, its length is computed by:

$$L_M(\mathsf{M}_i^1) = \int_0^1 \sqrt{(\vec{b} - \vec{a}) \, M(t) \, (\vec{b} - \vec{a})^T} dt \tag{16}$$

where $M(t)$ is the metric defined at location $\vec{a} + t\vec{b}$.

5. MESH SIZE FIELD SMOOTHING

The mesh size field should define element transition between shocks and smooth regions such that, for a mesh adapted to this field, two neighboring elements have sizes that differs less than a prescribed factor. In case the variation of desired size over an mesh entity is large while the mesh entity is too small to allow a smooth mesh size transition, the larger desired mesh size should be reduced.

We use the technique described in [11] to define a smooth metric field by bounding the ratio of mesh edge lengths between two adjacent edges that match the field by a prescribed factor β. Let M_i^0 and M_j^0 denotes the vertices of edge M_k^1, $h(\mathsf{M}_i^0)$ and $h(\mathsf{M}_j^0)$ being the desired edge length along M_k^1 at these two vertices with $h(\mathsf{M}_i^0) > h(\mathsf{M}_j^0)$, we compute:

$$c(\mathsf{M}_k^1) = \left(\frac{h(\mathsf{M}_i^0)}{h(\mathsf{M}_j^0)} \right)^{\frac{1}{L_M(\mathsf{M}_k^1)}} \tag{17}$$

where $L_M(\mathsf{M}_k^1)$ is the length of M_k^1 with respect to the desired metric field, computed using (16) . If $c(\mathsf{M}_k^1) > \beta$, the metric attached to M_i^0 is reduced to make $c(\mathsf{M}_k^1) = \beta$.

The smoothing procedure first loops over all edges of the mesh once. Whenever $c(\mathsf{M}_k^1) > \beta$, the metric attached to M_i^0 is reduced and all other edges connected to the vertex are appended into a dynamic list to be checked if they do not exist in the list yet. Then we continue to process the dynamic list until, for all edges in the mesh, $c(\mathsf{M}_k^1)$ is bounded by β.

6. 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 case 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

$$\mathcal{T}^{n+1} = \mathcal{T}^n + \mathcal{C} - \mathcal{C}'. \tag{18}$$

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 (18) in two steps:

$$\mathcal{T}' = \mathcal{T}^n + \mathcal{C}. \tag{19}$$

$$\mathcal{T}^{n+1} = \mathcal{T}' - \mathcal{C}'. \tag{20}$$

$\mathcal{T}'$ 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 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.

6.1 Conforming adaptation

The conforming mesh adaptation applies three kind of local modifications to match the smoothed mesh size field:

- element subdivision: first all edges that are too long are quickly determined and split at their centers in the transformed space with respect to the metric field, then all adjacent faces and regions of these edges are split using refinement templates [12]. Only 3 templates are needed to split triangles while a total of 42 templates are needed for tetrahedron splitting, with 4 of them have to introduce an additional vertex. When a triangle or a tetrahedron can be triangulated in multiple ways (see Figure 2 for example), we always select the one that produces the shortest long edge.
- edge collapsing: if an edge is too short, one of its vertex is removed from the mesh by collapsing it onto the vertex at the other end [12]. When several adjacent edges all need collapsing, edge collapsing is applied in a way that vertices are eliminated every other one and the shortest edge is collapsed first.
- edge/face swapping: after subdivision and collapsing, the mesh is enhanced by edge and face swapping's [13] to remove sliver triangles and tetrahedra in the transformed space with respect to the metric field. In the transformed space, a triangle (or tetrahedron) is a sliver if it has very small area (or volume) while all its bounding edges are not short.

The decision, whether an edge is too long or too short, is based on the length of the edge in the transformed space

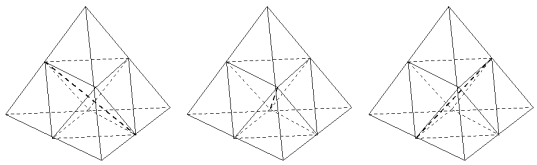

Figure 2: Three ways to split a tetrahedron into eight child tetrahedra.

with respect to the desired metric field, which is computed by (16) using one point quadrature. Consider mesh edge M_i^1, if $L_M(\mathsf{M}_i^1) > 1.33$, edge M_i^1 is marked for bisection while if $L_M(\mathsf{M}_i^1) < 0.67$, edge M_i^1 is considered to be too short, therefore, needs to be collapsed. If $0.67 < L_M(\mathsf{M}_i^1) < 1.33$, we keep edge M_i^1 since it can not match the desired size field better by splitting or collapsing.

Element subdivision, collapsing and swapping are performed successively until no edge need further refinement.

6.2 Non-conforming adaptation

Non-conforming adaptation consists in 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 tetrahedron, hexahedron 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.

7. EXAMPLES

We present the results of two compressible inviscid flow problems involving the solution of the Euler equations [9] by a DG method. The three-dimensional Euler equations have the form (1) with

$$\mathbf{u} = \{\rho, \rho v_x, \rho v_y, \rho v_z, E\}^t \tag{21}$$

$$\vec{\mathbf{F}}(\mathbf{u}) = \{\rho\vec{v}, \rho v_x\vec{v} + P\vec{e}_x, \rho v_y\vec{v} + P\vec{e}_y, \rho v_z\vec{v} + P\vec{e}_z, (\rho E + P)\vec{v}\}^t, \tag{22}$$

$$\mathbf{r} = \mathbf{0}. \tag{23}$$

Here ρ 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

$$P = (\gamma - 1)\,\rho \left[E - \frac{\|\vec{v}\|^2}{2} \right] \tag{24}$$

with the gas constant $\gamma = 1.4$.

7.1 A problem with four contact discontinuities

We consider a square domain of size 1×1 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 7.1. In this problem four contact discontinu-

	1	2	3	4
ρ	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.

ities are rotating around the center of the square creating a layered instability. Both isotropic and anisotropic features are present in the flow so that it is an interesting example. Figure 3 show solution for a effective mesh of 2560×2560 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 Figure 3 has four million degrees of freedom after having performed 300 mesh adaptations. 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.

Figure 4 shows the same computation with a smaller refinement, using an initial unstructured triangular mesh and doing non-conforming mesh refinement. We do 150 mesh refinements and, at the final stage of the computation, the problem has $\mathcal{O}(100,000)$ degrees of freedom.

Figure 4 shows results using isotropic conforming and non-conforming mesh refinement.

Because the flexibility of non conforming mesh refinement is not as good as the one of conforming mesh refinement for respecting a size field, we have tried to generate meshes that are as similar as possible. We have chosen $\beta = 2$ for the conforming smoothing in order to mimic the one-to-two behavior of non-conforming mesh refinement. The number of degrees of freedom vs. time curve is plotted in Figure 5. Both non-conforming and conforming mesh refinement generate roughly the same complexity, with of the order of 20% more degrees of freedom for the conforming refinement.

On figures 4, we see that the effect of multiple edge swapping is present but limited. On the left horizontal contact discontinuity for example, wee see that non-conforming refinement is producing smaller features. It is also true near the center of the square. Some smaller features are generated by the non-conforming mesh, especially at $t = 0.3$ but the impact of those thousands of edge swappings is much less severe than we expected (at each refinement step, we do $\mathcal{O}(500)$ edge swappings).

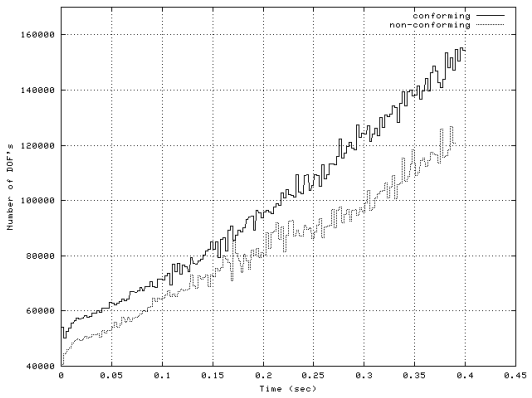

Figure 5: Number of degrees of freedom vs. time for the four-contact problem.

7.2 Backward Facing step

Consider the parallel Mach 3 flow of a gas in a channel where a step is impulsively inserted [4,14]. The channel is of length 3, height 1 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 anisotropic conforming and isotropic non-conforming mesh refinement techniques while, as well as for the previous problem, trying to use the same amount of degrees of freedom. For both runs, we start from the same initial unstructured triangular mesh with an uniform mesh size of 0.1. Figure 6 show mesh evolution during time.

The advantage of anisotropic mesh refinement is obvious when we look at Figure 7. 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: shocks are then re-

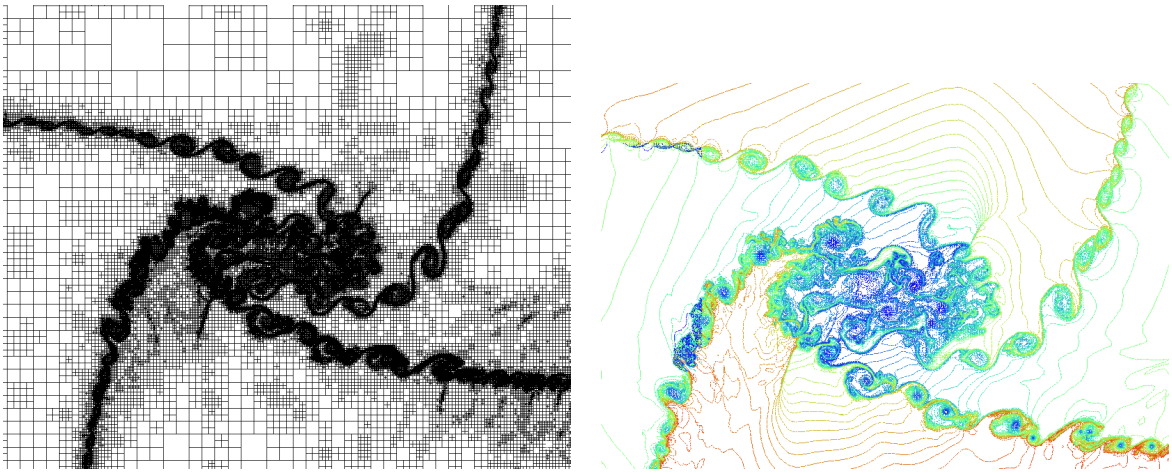

Figure 3: Refined mesh and density contours at $t = 0.3$ for the four-contact problem. Views show a zoom of the central area for the mostly refined computation.

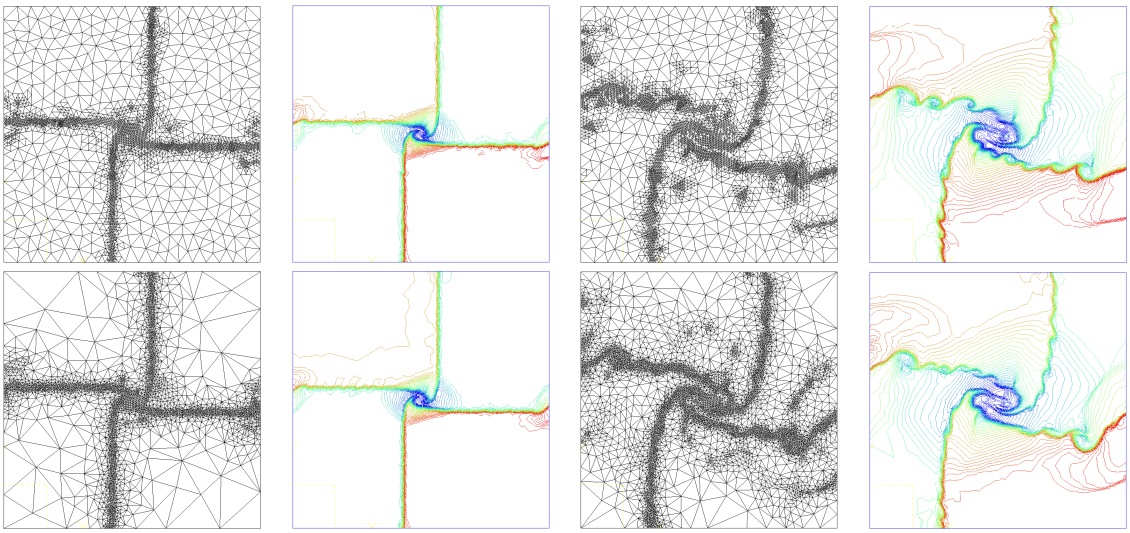

Figure 4: Refined mesh and density contours at $t = 0.1$ and $t = 0.3$ for the four contact problem. Non-conforming mesh refinement results are shown on the top while conforming ones are shown on the bottom.

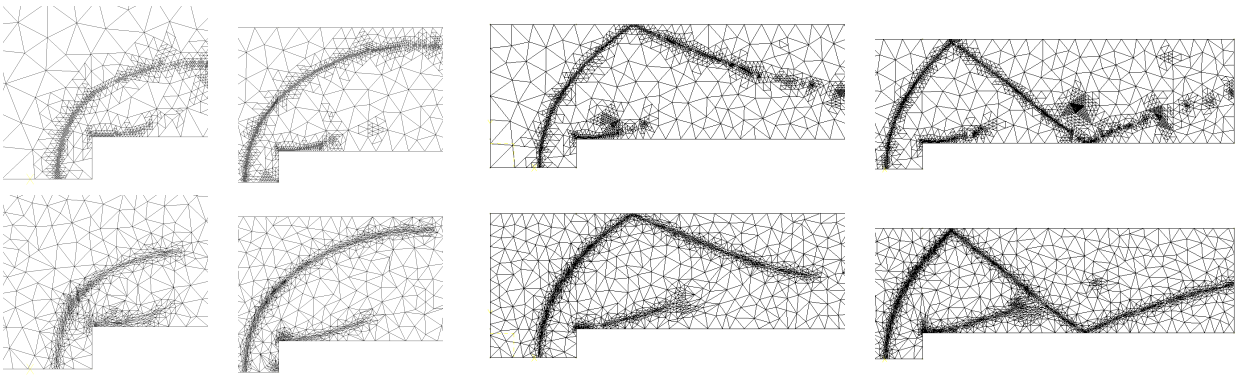

Figure 6: Mesh of the 2D backward facing step at times (from left to right) $t = 0.2$, $t = 0.4$, $t = 0.75$ and $t = 1.35$.

solved in one only element. Our anisotropic scheme did not do a great job for capturing the lefter shock. This is principally dued to our Hessian calculation. On early stages, the Hessian is calculated on a very distorted which usually lead to very bad accuracy on higher order derivatives. Then, the mesh is never enough regular in order to produce accurate second order derivatives. We belive that, for such transient problems, we should not compute the metric field based on the previous mesh but, for example, based on an octree or any other regular structure. We will develop this technique in another paper. Apart from the lefter shock, all other features are much better resolved using the anisotropic technique.

Finally, we show on Figure 8 some results of non-conforming mesh refinement for the backward facing step problem at different times. The moving shock is well captured thanks to mesh adaptation in time. One mesh adaptation was performed every 0.01 seconds so that a total of 230 mesh adaptations were necessary to reach 2.3 seconds of computation. The total number of degrees of freedom was $2.1\ 10^5$ at $t = 0.0$, $2.1\ 10^6$ at $t = 0.75$ and $2.7\ 10^6$ at $t = 2.0$.

8. CONCLUSIONS

Two new ideas have been proposed in this paper. First, we have shown how to build high-order derivatives of discontinuous fields. We have then applied non-conforming and conforming mesh refinement to transient compressible flow problems and have found that non-conforming adaptation was giving slightly better results when isotropic refinement is used, essentially because non-conforming adaptation is harmless in terms of solution transfer.

We have then shown that anisotropic mesh refinement was advantageous in comparison to non-conformin technique, producing less diffusion and sharper shocks for the same computational effort. It was also clear for us that computing Hessians using very distorted meshes was not satisfactory. Further work will take advantage of regular structures like octrees to compute the metric field.

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." *accepted in the International Journal for Numerical Methods in Engineering*, 2002

[3] Leer B.V. "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

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

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

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

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

[11] Borouchaki H., Hecht F., Frey P.J. "Mesh gradation control." *International Journal for Numerical Methods in Engineering*, vol. 43, no. 6, 1143–1165, 1998

[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] de l'Isle E.B., George P.L. "Optimization of tetrahedral meshes." I. Babuska, J. E.Flaherty, W.D.H. J. E.Hopcroft, J. E.Oliger, T. Tezduyar, editors, *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, 75, pp. 97–128. Springer-Verlag, 1993

[14] Cockburn B., Shu C. "TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for the Conservation Laws II: General Framework." *Mathematics of Computations*, vol. 52, 411–435, 1989

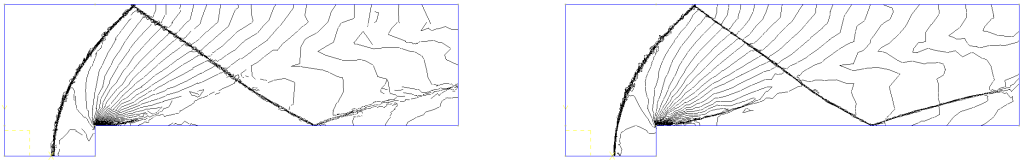

Figure 7: Density contours at $t = 1.1$ using isotropic (left) and anisotropic (right) mesh refinement methods.

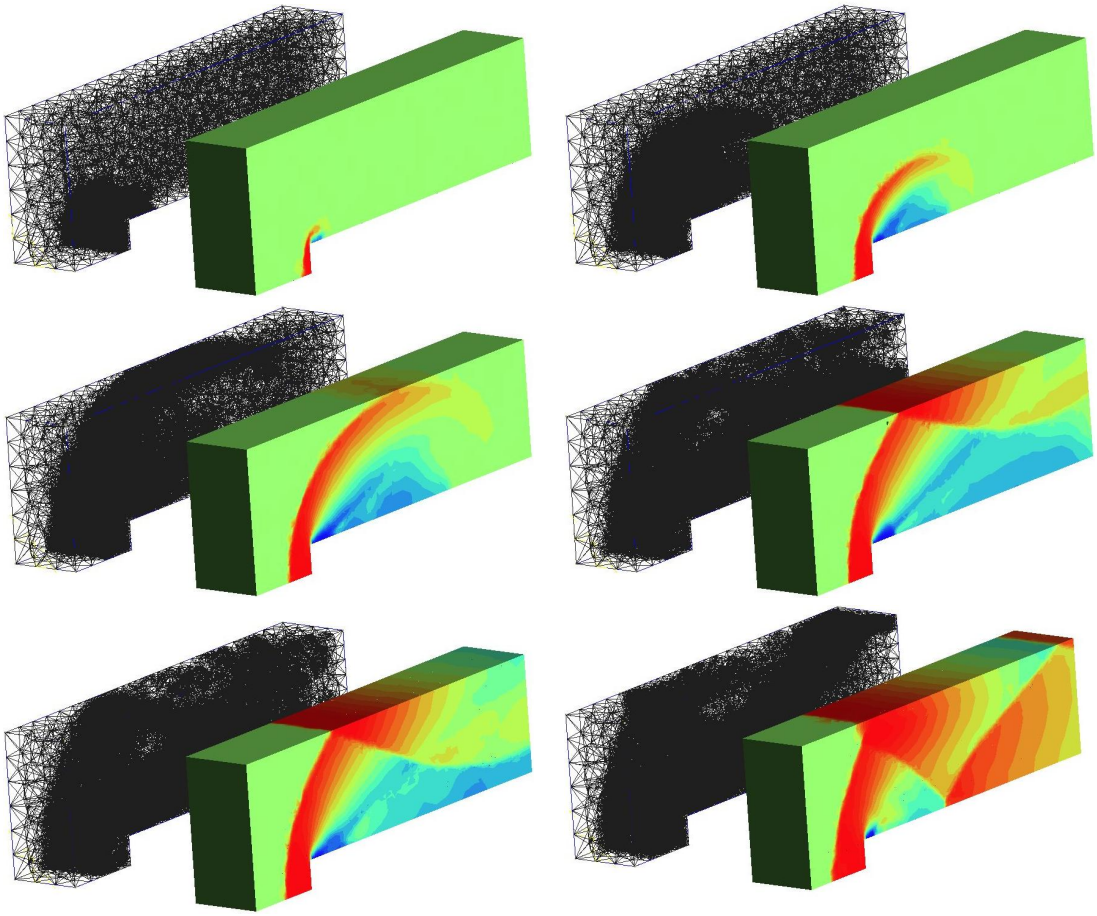

Figure 8: Backward facing step at times $t = 0.07$, $t = 0.27$, $t = 0.47$, $t = 0.75$, $t = 0.87$ and $t = 2.30$. Figures shows the adaptive mesh and density contours.