INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2000; **00**:1–6 *Prepared using* nmeauth.cls *[Version: 2002/09/18 v2.02]*

$\mathbf{P^1}$-conservative solution interpolation on unstructured triangular meshes

F. Alauzet[1,*] and M. Mehrenberger[2]

[1] *INRIA, Projet Gamma, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France.*
[2] *IRMA, Université Louis-Pasteur, UMR CNRS 7501, 7 rue René Descartes, 67084 Strasbourg, France.*

SUMMARY

This document presents an interpolation operator on unstructured triangular meshes that verifies the properties of mass conservation, $\mathbf{P^1}$-exactness (order 2) and maximum principle. This operator is important for the resolution of the conservation laws in CFD by means of mesh adaptation methods as the conservation properties is not verified throughout the computation. Indeed, the mass preservation can be crucial for the simulation accuracy. The conservation properties is achieved by local mesh intersection and quadrature formulae. Derivatives reconstruction are used to obtain an order 2 method. Algorithmically, our goal is to design a method which is robust and efficient. The robustness is mandatory to apply the operator to highly anisotropic meshes. The efficiency will permit the extension of the method to dimension three. Several numerical examples are presented to illustrate the efficiency of the approach.

KEY WORDS: Solution interpolation, conservative interpolation, localization algorithm, unstructured mesh, mesh adaptation, conservation laws

1. INTRODUCTION

Solution interpolation or solution transfer is an important stage for several applications in scientific computing. For instance, it is an essential component of the Arbitrary Lagrangian-Eulerian (ALE) methods. In such a context this stage is generally named remapping or rezoning. An accurate remapping has to verify some properties such as conservation, high order accuracy, bound preserving, ... Numerous works have addressed such remapping strategies, see for instance [1, 2, 3]. In these approaches, the mesh is considered with a fixed topology, *i.e.,* the number of vertices, elements and the connectivities remain unchanged. However, some of them have been extended to also handle meshes with changing topology as [4, 5].

Solution interpolation is also a key point in mesh adaptation for Eulerian simulations. Indeed, it links the mesh generation and the numerical flow solver, and it allows the simulation to be

*Correspondence to: INRIA, Projet Gamma, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France.

Received 21 January 2009
Revised 18 September 2009

restarted from the previous state. More precisely, after generating a new (possibly adapted) mesh, called *current mesh*, the aim is to recover the previous solution field defined on the old mesh, called *background mesh* on this new mesh to pursue the computation. This recurrent stage in adaptive simulations is crucial for time-dependent problems as the errors introduced by the interpolation procedure accumulate throughout the computations. The impact of such errors on the solution accuracy was pointed out in [6] where standard linear interpolation is applied.

In this paper, we consider the solution interpolation in the context of anisotropically adapted triangular meshes where the background and the current meshes are distinct, in the sense that the number of entities and the connectivities are completely different. Flows are modeled by the conservative compressible Euler equations and resolved by a second order finite volume scheme. Therefore, to obtain a consistent mesh adaptation loop, the proposed interpolation scheme must satisfy the following properties:

- mass conservation
- $\mathbf{P}^1$ exactness implying an order 2 for the method
- maximum principle.

Moreover, this method has to be algorithmically very robust as we deal with highly stretched elements and it has to be very efficient to be extensible to 3D. The word efficient signifies that it requires low memory storage and that the cpu time over cost with respect to the standard linear interpolation is acceptable.

The mass conservation property of the interpolation operator is achieved by local mesh intersection, *i.e.,* intersections are performed at the element level. The use of mesh intersection for conservative interpolation seems natural for unconnected meshes and has already been alluded in [7] or applied in [5] for order 1 reconstruction. The locality is primordial for the efficiency and the robustness. Once again for efficiency purposes, the proposed intersection algorithm is especially designed for simplicial meshes. Then, the idea is to compute the intersection between two simplexes and to mesh this intersection in order to use quadrature formulae to exactly compute the transfered mass.

The high-order accuracy is obtained by a solution gradient reconstruction from the discrete data and the use of Taylor formulae. This high-order interpolation can lead to loss of monotonicity. The maximum principle is then enforced by correcting the interpolated solution. Notice that much care has been taken while designing the localization algorithm as it is also critical for efficiency.

The proposed $\mathbf{P}^1$-conservative interpolation operator is suitable for solutions defined at elements or vertices.

The paper is outlined as follows. Section 2 introduces the main definitions and Section 3 presents the localization algorithm. The standard linear interpolation is recalled in Section 4. Then in Section 5, the proposed $\mathbf{P}^1$-conservative interpolation operator is described. First, the mesh intersection algorithm is given and at a second stage, $\mathbf{P}^1$-conservative reconstruction is discussed. In Section 6, we provide pseudo-conservative interpolation schemes based on high-order quadrature formulae or a Lagrangian approach. Finally, the efficiency of the proposed approach is emphasized on analytical examples in Section 7 and adaptive numerical simulations in Section 8. Some concluding remarks close the paper.

2. DEFINITIONS AND NOTATIONS

In this section, we provide the reader with notations, definitions and conventions used in the paper. Let us consider a bounded domain $\Omega \in \mathbb{R}^2$ and let $\partial\Omega$ be its boundary. We like to introduce a triangular mesh $\mathcal{H} = \bigcup K_i$ of domain Ω composed with triangles. A triangle in $\mathbb{R}^2$ is defined by the list of its vertices which are locally numbered in a convenient way. This list, enriched with some conventions, provides the complete definition of the related element, including the definition of its edges and neighbors, together with an orientation. Indeed, in our applications we strictly require an orientation of the elements of the mesh. In particular, the oriented local numbering of the triangle vertices enables us to compute its surface area while giving a sense to its sign. It also enables directional normals to be evaluated for each edge.

Formally speaking, the local numbering of vertices, edges and neighboring triangles is predefined in such a way that some properties are implicitly induced. This definition is only a convenient convention resulting in implicit properties. In the case of a triangle with vertices $[P_0, P_1, P_2]$ in this order, the first vertex having been chosen, the numbering of the others is deduced counter-clockwise, see Figure 1 (left). This orientation provides us with positive sign while computing the triangle surface area. Then, the topology can now be well defined thanks to the edges definition: $\vec{e}_0 = \overrightarrow{P_1P_2}$, $\vec{e}_1 = \overrightarrow{P_2P_0}$ and $\vec{e}_2 = \overrightarrow{P_0P_1}$. This numeration is such that the index of the edge is the index of the viewing vertex, *i.e.,* the opposite vertex. Regarding the neighboring triangles, we denote by K_i the neighbor viewing vertex P_i through edge $\vec{e}_i$, see Figure 1 (left).

In the rest of the paper all the indices in square bracket are given modulo 3 : $[i] = i \ mod(3)$. With all these notations, we now give some definitions utilized in all the paper algorithms.

Let $K = [P_0, P_1, P_2]$ be a triangle, its **signed (surface) area** A_K is given by:

$$A_K = \frac{1}{2}\begin{vmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix} = \frac{1}{2}\begin{vmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{vmatrix}.$$

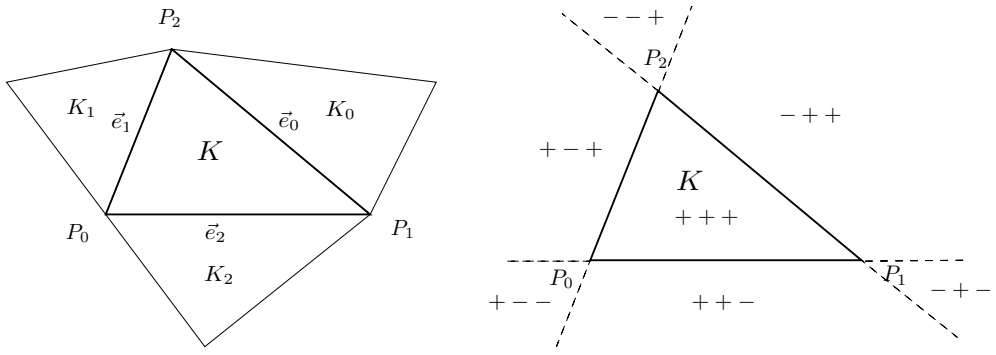

Figure 1. *Left, definition of a triangle K and its three neighbors K_i. Vertices indices are ordered counter-clockwise and the entities numeration is the same as the viewing vertices. Right, the seven regions defined by the signs of the three barycentric coordinates of a point P with respect to an element K.*

This area is **positive** if the triangle is numerated counter-clockwise which is our convention on the mesh orientation. The signed area is also given by one half of the z-component of $\overrightarrow{P_0P_1} \wedge \overrightarrow{P_0P_2}$.

Let P be a point, we denote by K^i the virtual triangle where vertex P_i is substituted by P. The signed areas A_{K^i}, for $i = 0 \dots 2$, are called the **barycentrics** of P. The three associated **barycentric coordinates** are given by:

$$\beta_i = \frac{A_{K^i}}{A_K} \quad \text{for} \quad i = 0 \dots 2 \,.$$

The sign of the three barycentric coordinates or barycentrics defines explicitly seven regions of the plane where point P can be located with respect to element K. The possible combinations are given in Figure 1 (right).

Finally, we introduce a definition of the distance of a point P with respect to an edge of a triangle $K = [P_0, P_1, P_2]$. The **signed distance**, also called **power**, of point P with respect to edge $\vec{e}_i = \overrightarrow{P_{[i+1]}\,P_{[i+2]}}$, for $i = 0 \dots 2$, is given by:

$$\mathcal{P}(P, \vec{e}_i) = \overrightarrow{P_{[i+1]}P} \,.\, \overrightarrow{N}_{e_i} = \overrightarrow{P_{[i+2]}P} \,.\, \overrightarrow{N}_{e_i} \,,$$

where $\overrightarrow{N}_{e_i}$ is the inward unit normal (for the element) of edge $\vec{e}_i$. Notice that the barycentrics and the powers are linked by the relation:

$$A_{K^i} = \frac{1}{2} \, ||\vec{e}_i|| \, \mathcal{P}(P, \vec{e}_i) \,.$$

From these relations, we deduce the coordinates of the orthogonal projection X of P on the line defined by edge $\vec{e}_i = \overrightarrow{P_{[i+1]}\,P_{[i+2]}}$:

$$X = P_{[i+1]} + \frac{\vec{e}_i \,.\, \overrightarrow{P_{[i+1]}P}}{\|\vec{e}_i\|^2} \, \vec{e}_i = P_{[i+2]} + \frac{\vec{e}_i \,.\, \overrightarrow{P_{[i+2]}P}}{\|\vec{e}_i\|^2} \, \vec{e}_i \,.$$

Finally, we recall some definitions relative to the interpolation schemes. Let u be a solution defined on a mesh $\mathcal{H}^1$ of a domain Ω. The **mass** of the solution over the mesh is simply $m = \int_{\mathcal{H}^1} u$. We deduce the notion of mass on an element K given by $m_K = \int_K u$.

An interpolation scheme is said to be **conservative** if it preserves the mass when transferring the solution field u from a mesh $\mathcal{H}^1$ to another $\mathcal{H}^2$. Formally speaking, if we denote by Πu the interpolated field on $\mathcal{H}^2$, then such scheme verifies

$$\int_{\mathcal{H}^1} u = \int_{\mathcal{H}^2} \Pi u \,.$$

A scheme is said to be $\mathbf{P}^k$**-exact** if it is exact for polynomial solutions of degree lower than or equal to k. Finally, a $\mathbf{P}^k$**-conservative** interpolation scheme is a scheme satisfying both properties.

3. LOCALIZATION ALGORITHM

The localization problem or research of point location consists in identifying the element of a simplicial mesh containing a given point. The localization of a given point in a mesh is a

frequent issue that arises in various situations. As regards interpolation methods, we initially have a mesh with a field, here the solution, that we call *background mesh*, denoted $\mathcal{H}^{back}$. We aim at transferring or interpolating the field onto another mesh called *current mesh* or *new mesh*, denoted $\mathcal{H}^{new}$. Therefore, the algorithm consists in finding which elements of the background mesh contain the vertices of the new mesh in order to apply an interpolation scheme.

Here, we consider the simplified problem where the background and the new meshes are discretizations of the same domain Ω. This problem has to be dealt with care in the case of simplicial meshes to handle difficult configurations. Indeed, background and current meshes can be non-convex and can contain holes. It is also possible that the overlapping of the current mesh does not coincide with the background mesh since their boundary discretization can differ. Consequently, some vertices of the current mesh can be outside of the background mesh and conversely. Moreover, efficient localization algorithms have to be implemented to avoid the naive quadratic scheme in $O(N_{ver}^{new} \times N_{tri}^{back})$ where N_{ver}^{new} is the number of vertices of $\mathcal{H}^{new}$ and N_{tri}^{back} the number of triangles of $\mathcal{H}^{back}$.

The localization can be solved efficiently by traversing the background mesh using its topology, *i.e.,* the neighboring elements of each element, thanks to a barycentric coordinates-based algorithm [8, 9]. More precisely, in two dimensions, let P be a vertex of the new mesh, $K = [P_0, P_1, P_2]$ a triangle of the background mesh. From the signs of the three barycentric coordinates $\{\beta_i\}_{i=0,2}$, three possible cases arise (see Figure 2):

- all barycentric coordinates are positive then vertex P is located inside element K
- one barycentric coordinate is negative then it indicates the direction for the next move. For instance, if barycentric β_i is negative then we move to neighboring element K_i sharing edge $\vec{e}_i$ with K. We say that P is viewed by edge $\vec{e}_i$
- two barycentric coordinates are negative then two neighboring triangles are possible for the next move. A random choice or a geometric one is used.

Starting from an initial element K_0 of the background mesh, we apply the previous test. According to the signs of the barycentric coordinates, we pass through the corresponding neighbor of K_0 and we repeat this process until the three barycentric coordinates are positive meaning that the visited triangle contains P. With this algorithm, we follow a path in the background mesh to locate vertex P as shown in Figure 3 (left). This algorithm complexity is in $O(n \times N_{ver}^{new})$ where n is the average number of visited triangles for each path.

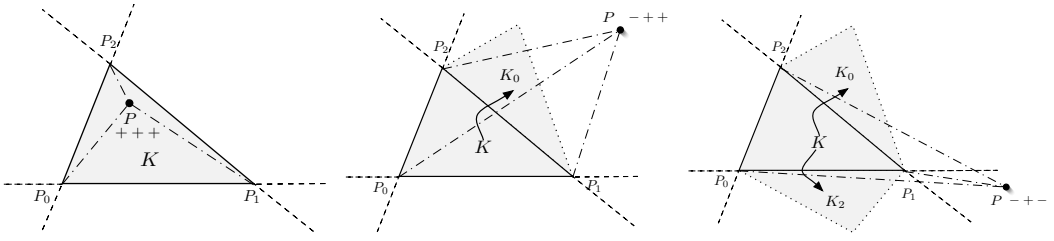

Figure 2. *Illustration of the three possible cases depending on the signs of the three barycentric coordinates of vertex P with respect to triangle K when moving inside the background mesh.*

However, cyclic or closed paths can occur. The element containing the vertex is missed and an infinite loop is obtained. In this case, the path leads us to an already tested element, as presented in Figure 3 (right). In this academic example, starting from K_0, triangles K_1, K_2, K_3, K_4 and K_5 are visited bringing us back to K_0. A color algorithm, to mark already visited elements, is used to avoid this problem allowing us to choose another direction when several choices reoccur. Another way to solve this problem is to consider a random choice when several possibilities occur.

Another difficulty arises when the path is stuck by the geometry of domain Ω. Starting element K_0 and vertex P are separated by a hole (Figure 4, left) or by a non-convex domain (Figure 4, right). The path demands to pass through the hole or the boundary to reach the element containing vertex P. A simple, but inefficient, way to remedy this problem is to make an exhaustive search, *i.e.,* for such a vertex all elements of the background mesh are tested. Besides, a more challenging solution is to follow a path on the boundary in order to bypass the obstacle.

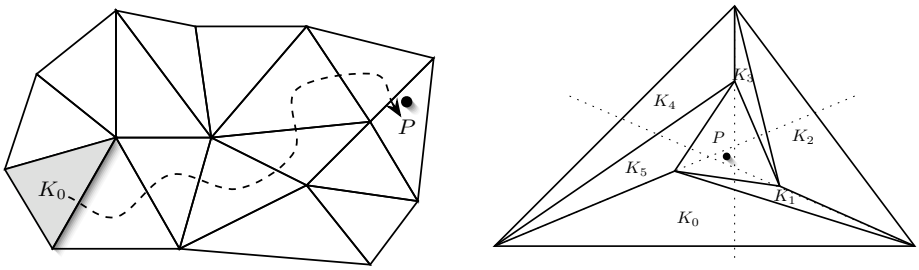

Figure 3. *Left, a possible path to locate the vertex P of the new mesh starting from the triangle K_0 of the background mesh. Right, cyclic path leading us to an already checked element. Starting from K_0, triangles K_1, K_2, K_3, K_4 and K_5 are visited bringing us back to K_0.*

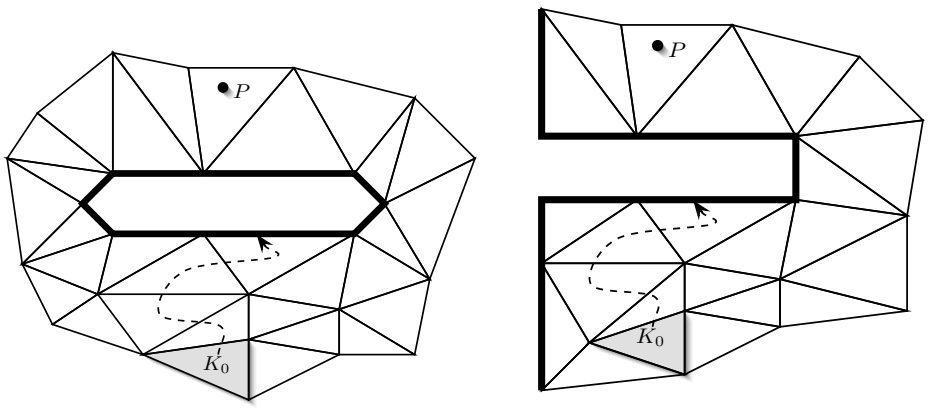

Figure 4. *Starting element K_0 and vertex P are separated by a hole (left) or the non-convex domain (right). The path demands to pass through the hole or the boundary to reach the element containing vertex P.*

Localization coupled with a grid structure. The previous algorithm can be very time consuming if a large number of elements (e.g. n is large) needs to be visited between triangle K_0 and the solution triangle. This can result in a large number of area computations. This major drawback leads us to consider a more local approach which aims at combining the algorithm with a grid structure (a tree-like structure can be considered). This facilitates and speeds up the localization process. A grid enclosing the mesh is constructed and, for each grid cell, one element of the background mesh located in it, if any, is recorded. Then, to locate a new vertex in the background mesh, the cell containing the vertex is first identified and then the localization scheme starts from the element associated with this cell. In this way, the number of visited triangles is reduced and the number of necessary computations decreases as well. In the case where we are stuck by the boundary, because of a hole or a non-convex domain, the grid structure helps us to bypass the obstacle. Indeed, elements associated with grid neighboring cells of the current one are considered as new initial guesses for the searching algorithm. The localization is restarted from one of these new elements.

Remark 3.1. *Note that the grid (or the tree-like structure) could be defined in various ways depending on the nature of the data set. In this respect, for a grid, the number of cells and thus the occupation of the cells are parameters that clearly affect the efficiency of the whole process.*

Localization using the topology of both meshes. We can even improve the locality of the localization scheme by using the topology of both meshes. Such scheme tends to minimize n the number of elements visited when locating new vertices. Instead of determining the location of the new vertices in their data (or storage) order, the idea is that once a vertex P has been located in a background element K, then we handle the set of vertices $\{Q_i\}_{i=1,m}$ of the ball of P, *i.e.,* the set of vertices that are connected to P by an edge. For the vertices $\{Q_i\}_{i=1,m}$, we set as starting element of the localization process the triangle K that contains P, see Figure 5. Consequently, the number of visited triangles is drastically reduced as in this algorithm the initial guess of the searching process is at the element (or connectivity) level. Moreover, with this approach, the scheme does not depend on any parameters.

Another advantage of this approach is that this scheme avoids the problem where the process is stuck by a hole or a non-convex boundary as vertices $\{Q_i\}_{i=1,m}$ are connected to P in the new mesh. This algorithm is also in $O(n \times N_{ver}^{new})$ where n is the average number of visited triangles and here n tends to be optimal. Indeed, in practice the number of visited triangles is on average less than 3. In fact n is of the order of the number of elements of the background mesh that are overlapped by an element of the current mesh.

Handling the "fork" problem. We assume that vertex P is inside the background discretized domain. When the path reaches a geometrical fork or a crossroads with multiple choices for the next move, the presented algorithm could make the wrong choice and ask to process in the wrong direction, see Figure 6. Then, we are no more able to locate vertex P as we are stuck by the boundary. Special treatment has to be considered to solve this problem. To this end, we require to have for each boundary edge:

- the list of its two neighboring boundary edges
- the unique triangle sharing this edge. This element will be used as initial element guess to localize P.

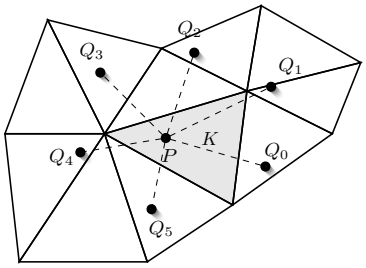

Figure 5. *Reducing the number of visited triangles in the localization scheme by using the topology of both meshes. Vertex P has been located in element K. Then, the set of vertices $\{Q_i\}_{i=1,m}$ connected to P uses element K as initial guess for the localization scheme.*

When the "fork" problem occurs, the algorithm is stuck in an element K_{stuck} for which P is seen by a boundary edge e. To localize vertex P, an iterative algorithm is considered. It consists in applying the localization process starting each time from a new triangle given by a boundary edge neighboring the current one to which we are stuck, this until vertex P is found. More precisely, at the first step, we consider as initial guess for the localization algorithm the triangles denoted K_{step1} associated with the two boundary edges neighbors of e, *i.e.,* the neighboring edges of order 1 of the current edge, see Figure 6 (right). If P is not found, then we consider the neighboring edges of order 2, the neighboring edges of the neighboring edges. We start from the elements denoted K_{step2}. If P is still not located, then we consider the next order of neighbors and so on until convergence of the algorithm. Notice that in two dimensions, at each new iteration, only two new edges are considered, since the other ones have already been checked. At worst all boundary edges are checked.

Figure 6, right, illustrates this algorithm. The localization process bring us to K_{stuck}. We apply the localization process starting from the two triangles denoted K_{step1} and we are still blocked. Then, we consider K_{step2}. If the algorithm still fails, we consider K_{step3} and P is localized.

Handling boundary problems. However, none of these algorithms are able to handle the case of vertices localized outside of the background discretized domain. Special treatment for this kind of vertices must be designed. We consider the simplified problem where the background and the new meshes are discretizations of the same domain Ω, when a vertex is outside of the background mesh its distance to the mesh is of the order of the boundary mesh tolerance (*i.e.,* the gap between the boundary discretization and the exact geometry). This will be used to set the tolerance ε to say if P is close or not to the boundary.

In this case, the aim is to find the "closest" triangle to vertex P which is the unique triangle associated with the "closest" boundary edge to vertex P. A boundary edge is considered as the closest boundary edge if two properties are verified :

(i) the power of vertex P with respect to the boundary edge e is lower than the given threshold ε:

$$\mathcal{P}(P, e) < \varepsilon,$$

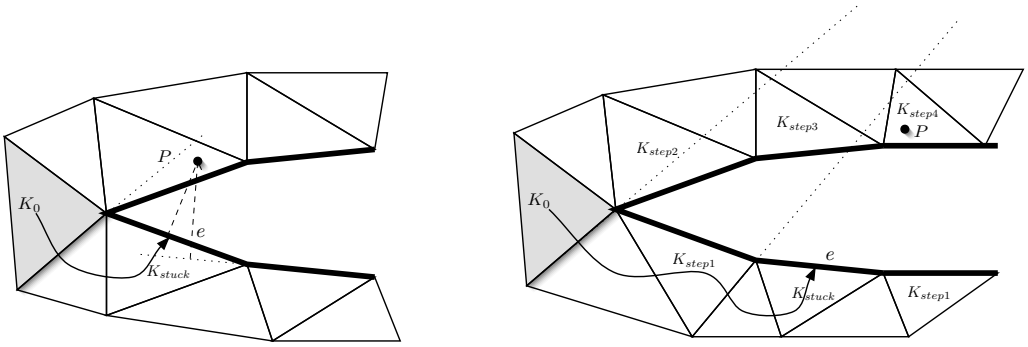

Figure 6. *Two illustrations of the "fork" problem when the algorithm has made the wrong choice. We are stuck in triangle K_{stuck} where P is viewed by boundary edge e. The figure on the right illustrates the process to remedy this problem by restarting the localization from new triangles. At first step, we choose triangles K_{step1} as initial guess. Then, K_{step2} if P has not been found. And so on.*

(*ii*) the orthogonal projection X of P on the line defined by e lies inside the edge $e = [Q_0, Q_1]$, mathematically speaking:

$$0 < \vec{e}.\overrightarrow{Q_0P} < \|\vec{e}\|^2.$$

The meaning of each condition is exemplified in Figure 7. The condition (*i*) is not verified on the left, whereas the condition (*ii*) is not fulfilled on the right.

Practically, we have no warranty that the localization scheme finds directly the correct boundary edge. If the scheme does not succeed then we apply the same algorithm as in the "fork" problem. The localization process restarts each time from a new triangle given by a boundary edge neighboring the current one where the process has been stuck, this until the "closest" triangle is found.

At worst, it is always possible to perform an exhaustive search. All boundary edges are checked and we select the closest one.

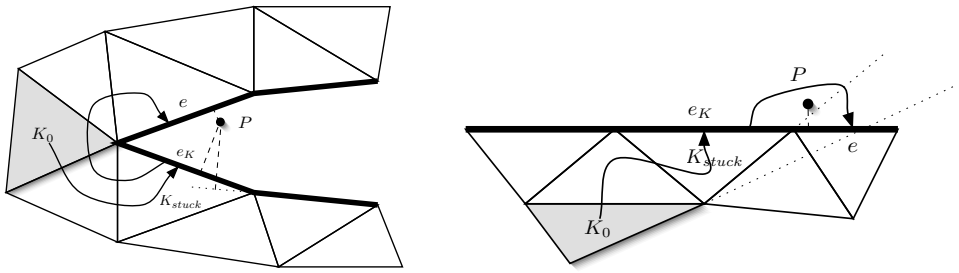

Figure 7. *Illustration of the possible cases when a vertex is outside of the domain. The localization algorithm ends in triangle K_{stuck}. The condition (i) (resp. (ii)) is not fulfilled for edge e_K on the left (resp. right) figure. We have to move to edge e to find the "closest" triangle.*

4. CLASSICAL LINEAR INTERPOLATION

In this section, we present the classical linear interpolation scheme which is not conservative. In this paper, the provided solution is considered to be piecewise linear by element. In the case of a nodal value representation of the solution, we get an implicit continuous piecewise linear solution by element.

Let P be a vertex of the current mesh, $K = [P_0, P_1, P_2]$ be a triangle of the background mesh containing P and β_i, for $i = 0, ..., 2$, be the barycentric coordinates of P with respect to K. We denote by $\mathbf{P}^k$ the set of polynomials of degree less or equal than k and by $\mathbf{P}_r$ the set of polynomials where the solution given by the interpolation scheme lies. The classical $\mathbf{P}^1$ interpolation scheme reads:

$$\Pi_1 u(P) = \sum_{i=0}^{2} \beta_i(P)\, u(P_i)\,.$$

where the interpolation operator has been denoted Π_1. This scheme is $\mathbf{P}^1$ exact, we have $\mathbf{P}_r = \mathbf{P}^1$ and it is thus of order 2. This scheme is monotone and satisfies the maximum principle. However, this scheme does not conserve the mass. Indeed, if an edge between two triangles is swapped then this interpolation keeps the solution at the triangles vertices unchanged whereas the mass of the solution has effectively changed.

Remark 4.1. *The interpolation operator* Π_1 *is independent of the mesh topology. Therefore, it can be applied to any points of the domain.*

5. $\mathbf{P}^1$-CONSERVATIVE INTERPOLATION

In this section, we present a $\mathbf{P}^1$-conservative interpolation scheme. The provided solution is considered to be piecewise linear by element. The piecewise representation can be continuous or discontinuous.

The idea of the conservative interpolation is to compute the mass of each element of the new mesh $\mathcal{H}^{new}$ knowing the mass of each element of the background mesh $\mathcal{H}^{back}$. To this end, a local mesh intersection algorithm is utilized. Then, in the case of vertex-centered solution, the solution is transferred accurately and conservatively to vertices using the mass of the elements of its ball. More precisely, the algorithm is decomposed in the following steps:

1. localize the vertices of $\mathcal{H}^{new}$ in $\mathcal{H}^{back}$
2. set mass for all $K^{back} \in \mathcal{H}^{back}$
3. for all $K^{new} \in \mathcal{H}^{new}$ compute its intersection with all $K_j^{back} \in \mathcal{H}^{back}$ that it overlaps
4. mesh the intersection polygon of each couple of elements (K^{new}, K_j^{back})
5. get the mass and the gradient mass on K^{new}. A piecewise discontinuous reconstruction is obtained
6. correct the reconstruction to enforce the maximum principle
7. set the solution values to vertices by averaging for nodal values solution

The mesh intersection procedure corresponding to step 3 and 4 is exposed in Section 5.1 and the conservative reconstruction, step 5 to 7, is described in Section 5.2.

5.1. Mesh intersection algorithm

The mesh intersection algorithm consists in intersecting each triangle of the current mesh with all the background mesh triangles that it overlaps and in meshing the intersection region. In the following, we first describe our generic intersection algorithm between any pair of triangles and how we discretize the polygon of intersection (Section 5.1.2). The core of this algorithm is the edge-edge intersection procedure (Section 5.1.1). Secondly, in the context of the conservative interpolation, the scheme to locate all background triangles that are overlapped by the current element and the way the intersections are handled are presented (Section 5.1.3).

5.1.1. Edge-edge intersection Let $e_P = [P_0P_1]$ and $e_Q = [Q_0Q_1]$ be two edges of the plane, cf. Figure 8. We denote by $\overrightarrow{N}_{e_P}$ and $\overrightarrow{N}_{e_Q}$ their counter-clockwise oriented unit normals, respectively. Assuming we are not in a degenerated case, *i.e.,* all powers are not zero, then there is intersection of the two edges if and only if:

$$\mathcal{P}(P_0, e_Q)\,\mathcal{P}(P_1, e_Q) < 0 \quad \text{and} \quad \mathcal{P}(Q_0, e_P)\,\mathcal{P}(Q_1, e_P) < 0\,.$$

Then, the intersection point of the two edges X is simply expressed in terms of powers by the relation:

$$X = P_0 + \frac{\|\overrightarrow{P_0X}\|}{\|\overrightarrow{P_0X}\| + \|\overrightarrow{P_1X}\|}\overrightarrow{P_0P_1} = P_0 + \frac{\mathcal{P}(P_0, e_Q)}{\mathcal{P}(P_0, e_Q) - \mathcal{P}(P_1, e_Q)}\overrightarrow{P_0P_1}\,,$$

or

$$X = P_0 + \frac{\mathcal{P}(P_0, e_Q)}{\overrightarrow{N}_{e_Q}.\overrightarrow{P_1P_0}}\overrightarrow{P_0P_1}\,. \tag{1}$$

There are three degenerated cases to handle with care:

- only one power is zero: for instance $\mathcal{P}(P_0, e_Q) = 0$, then there is intersection if and only if $\mathcal{P}(Q_0, e_P)\,\mathcal{P}(Q_1, e_P) < 0$ and $X = P_0$ (see Figure 8, right)
- two powers are zero, one for each edge: for instance $\mathcal{P}(P_0, e_Q)$ and $\mathcal{P}(Q_0, e_P)$, then there is intersection and $X = P_0 = Q_0$
- all powers are zero, then the two edges are aligned (see Figure 8, right). There is intersection if and only if the edges overlap each other. There are two sub-cases:
  - one intersection point that is the common point of the two edges
  - two intersection points that are the end-points of the edge included in the other one or one end-point of each edge in the other case, cf. Figure 8, right.

Several edge-edge intersection cases are depicted in Figure 8.

5.1.2. Triangle-triangle intersection The triangle-triangle intersection procedure computes the intersection of two triangles and meshes the intersection region if it is not empty. Notice that if the intersection exists, the intersection region of two triangles is always a convex polygon given by the convex hull of the intersection points.

In the aim of applying the conservative interpolation to the three-dimensional case, it is crucial to propose a general intersection process that extends immediately to tetrahedron-tetrahedron intersection. Consequently, all the procedures assuming, for instance, that the

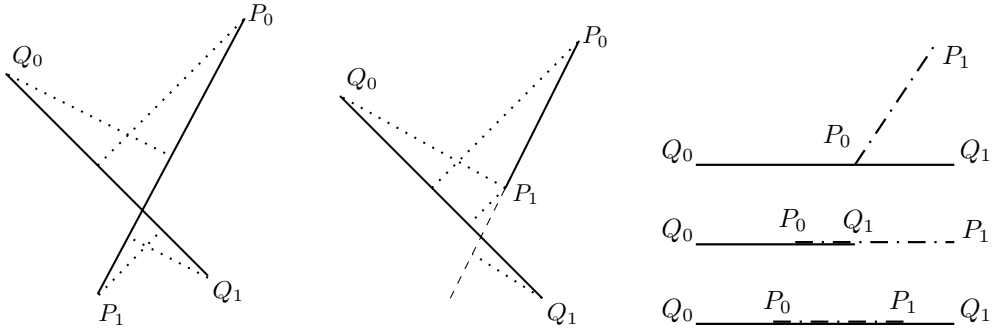

Figure 8. *Edge-edge intersection cases: an intersection (left), no intersection (middle) and several degenerated cases (right).*

intersection points are ordered and use this property to connect them are not considered. For instance, algorithms going through the element edges in the trigonometric order to compute the intersections are ignored. Indeed, such order does not exist in three dimensions.

One can also try to enumerate and classify all the possible configurations and design predefined meshes of the intersection region corresponding to each case. Each combination of vertices power signs describes explicitly an overlap configuration for any pair of triangles. In two dimensions, there are 18 possible cases. Nonetheless, this approach is too difficult to extend to 3D as the number of cases increases exponentially.

Therefore, the proposed approach must be generic and must not require any orders to be extensible to higher dimensions. To this end, a meshing technique, which extends to 3D, is proposed to obtain the mesh of the intersection region. It is a two steps process.

First, for the given pair of triangles, the list of the nine possible couple of edges, one for each triangle, is formed. The edge-edge intersection procedure is applied for each pair of edges. If an intersection occurs, then the intersection point is evaluated with Relation (1). These intersections result in a cloud of points. In degenerated cases, the number of cloud points is strictly less than 3. In non degenerated cases, the cloud of points contains between 3 and 6 points. In this case, the convex hull of this cloud of points forms a convex polygon representing the region of intersection of the triangles pair.

Second, this convex polygon is meshed by primarily constructing an oriented triangle with three points chosen randomly. Notice that the three points cannot be aligned by construction. Then, a new point is selected and the unique triangle edge which is viewing this point is looked for, *i.e.,* the only edge for which the barycentric coordinate is negative. A new oriented triangle is built by connecting the point to this edge. The process is iterated until all points are inserted by only checking edges forming the boundary of the current polygon (*i.e.,* edges which are not shared by two triangles). Notice that, at each iteration, as the current polygon is convex, there is only one boundary edge that views the selected point. A mesh of the polygon is then obtained. This process for five points is illustrated in Figure 9.

Remark 5.1. *This meshing procedure is just an application of the incremental Delaunay method to triangulate a cloud of points,* i.e., *the Delaunay kernel, in our particular case [7].*

Remark 5.2. *This method is extensible to three dimensions even if it is slightly more complicated, notably four points can be coplanar.*

No intersection criteria. When the triangles intersect, the proposed algorithm automatically detects it and meshes it whatever the configuration. But, three possible degenerated cases can occur during the process which are assimilated to *no intersection*:

- 0 intersecting point are found meaning an empty intersection,
- 1 intersecting point is found implying that the intersection of the triangles is a vertex,
- 2 intersecting points are found signifying that triangles intersect on a edge.

In such configurations, the algorithm has to make the distinction between the case where one triangle is included in the other one, meaning that the intersection is this triangle, and the case where the intersection is empty. We propose two criteria to discriminate these cases. Let $K_P = [P_0, P_1, P_2]$ and $K_Q = [Q_0, Q_1, Q_2]$ be two triangles. We denote by $e_j^P = [P_{[j+1]}\, P_{[j+2]}]$ and $e_j^Q = [Q_{[j+1]}\, Q_{[j+2]}]$ for $j = 0, .., 2$ the edges of K_P and K_Q, respectively.

Proposition 5.1. *K_P is* **included** *inside K_Q if and only if for all $P_i, i = 0, .., 2$ and for all $e_j^Q, j = 0, .., 2$, we have $\mathcal{P}(P_i, e_j^Q) \geq 0$. In other words, all powers of K_P are positives signifying that all the vertices of K_P are included inside K_Q.*

Proposition 5.2. *The intersection between K_P and K_Q is* **empty** *if and only if there exist e_j^P or e_j^Q such that for all Q_i or P_i we have $\mathcal{P}(Q_i, e_j^P) < 0$ or $\mathcal{P}(P_i, e_j^Q) < 0$. In other words, each triangle lies in a half-plane separated by the edge e_j^P or e_j^Q*

As regards the triangle-triangle intersection procedure, the algorithm computes the 9 possible edge-edge intersections. It requires the evaluation of the 18 possible vertices powers with respect to an edge. Actually, only their signs are needed. Consequently, the triangle inclusion case can be immediately checked. Then, if the intersection algorithm returns 0, 1 or 2 intersection point(s), it implies that no intersection occurs. Notwithstanding, Proposition 5.2 could be used to faster discriminate the non-intersection case.

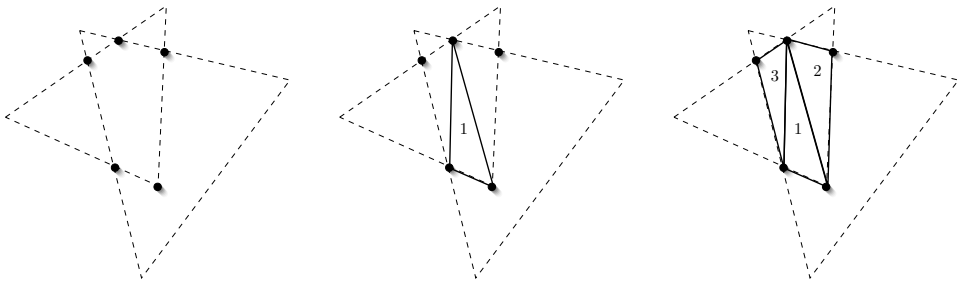

Figure 9. *Left, five intersection points have been computed for the triangle-triangle intersection. Middle, the first three points define the first triangle. Then right, the fourth point is connected to the vertices of the edge of triangle* 1 *viewing it. It creates triangle* 2*. Similarly, the triangle* 3 *is formed with the fifth point.*

5.1.3. Overlapped elements detection Now, in the context of the conservative interpolation, the method consists in computing for each triangle K^{new} of the current mesh $\mathcal{H}^{new}$ its intersection with all triangles K_j^{back} of the background mesh $\mathcal{H}^{back}$ that it overlaps. We present how this list of background elements is determined and the way the intersections are handled.

First of all, all vertices of the new mesh $\mathcal{H}^{new}$ are localized in the background mesh $\mathcal{H}^{back}$ using the algorithm presented in Section 3. Then, for each triangle K^{new} of $\mathcal{H}^{new}$, the initial list of background triangles overlapped is given by the elements containing the vertices of K^{new}. In degenerated cases where a vertex lies on an edge or on a background vertex, we add to the initial list both triangles sharing the edge or the ball of the background vertex, respectively.

Then, the intersections between K^{new} and the triangles of the initial list are computed. Then, new triangles are added to the list during the intersection procedure as follows:

- if an edge e_j of K^{back} is intersected by K^{new} (*i.e.,* it exists an edge of K^{new} for which the intersection occurs) then we add the neighbor K_j^{back} of K^{back} sharing the edge e_j to the list,
- if a vertex of K^{back} has all its powers positive with respect to the edges of K^{new} (*i.e.,* it lies inside K^{new}) then we add the ball of this vertex to the list.

With this simple procedure, all overlapped elements are automatically detected while computing intersections. The process is depicted in Figure 10. Overlapped elements are detected without any additional cost as powers and edge-edge intersections are already computed in the triangle-triangle intersection process.

As regards degenerated cases, when the edge-edge intersection results in an edge, then we still add the neighboring triangle. When it results in a vertex, we add the vertex ball to the list.

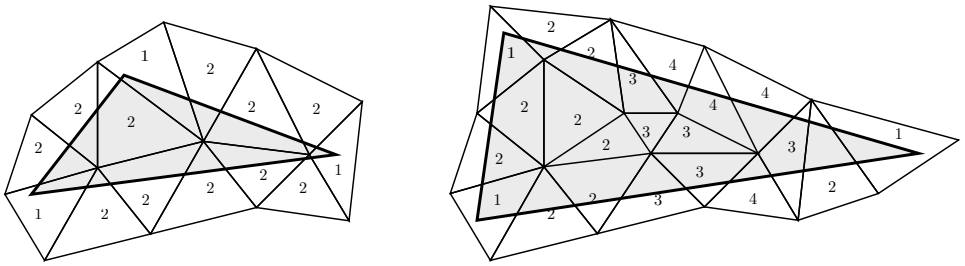

Figure 10. *Illustration of the overlapped elements detection process. First, the list of the background elements to which the vertices of K_{new} belong are detected: they are tagged* 1*. Then, the other elements are iteratively detected while computing intersections. The tag number represents the intersection step. Two steps are sufficient for the left case, whereas the right case requires four steps.*

5.2. $\mathbf{P}^1$*-conservative reconstruction*

In this section, we describe the $\mathbf{P}^1$-conservative solution reconstruction process. We consider a bounded domain Ω of $\mathbb{R}^2$ and two triangular meshes of this domain $\mathcal{H}^{back} = \bigcup K_i^{back}$

and $\mathcal{H}^{new} = \bigcup K_i^{new}$. For sake of simplicity, we first make the assumption that the discrete boundaries of both meshes are the same, *i.e.,* both meshes are discretization of the same polygonal domain Ω_h: $|\mathcal{H}^{back}| = |\mathcal{H}^{new}|$ where $|\mathcal{H}| = \int_{\Omega_h} dx$. The case of non-matching discrete boundaries is addressed in Section 5.2.3. For each mesh, a dual partition of the domain is defined by associating to each vertex of a mesh a control volume or cell (which is defined by some rules): $\mathcal{H}^{back} = \bigcup C_i^{back}$ and $\mathcal{H}^{new} = \bigcup C_i^{new}$. A $\mathbf{P}^1$ discrete solution field u is given on the background mesh $\mathcal{H}^{back}$.

Now, we have to define a projection operator Π_1^c from $\mathcal{H}^{back}$ to $\mathcal{H}^{new}$ with the following properties:

- Π_1^c is conservative: $\displaystyle\int_{\mathcal{H}^{back}} u = \int_{\mathcal{H}^{new}} \Pi_1^c u$
- Π_1^c is $\mathbf{P}^1$ exact: if u is affine then $\Pi_1^c u = u$.

In the following, we define the projection operator for solutions defined at elements and solutions defined at vertices.

5.2.1. Solution defined at elements In this case, the solution is piecewise linear by elements and can be discontinuous. We have for each background triangle K^{back}:

- the mass $m_{K^{back}} = \int_{K^{back}} u = |K^{back}|\, u(G)$, where G is the barycenter of the triangle
- the constant gradient $\nabla u_{K^{back}}$.

For each triangle K^{new} of the current mesh, we compute the intersection with all triangles of the background mesh $\{K_j^{back}\}$ it overlaps as described in the previous section. Each couple of triangles K^{new} and K_j^{back} provides a simplicial mesh of their intersection region denoted $\mathcal{T}_j = K^{new} \cap K_j^{back}$. The integrals $\displaystyle\int_{\mathcal{T}_j} u$ and $\displaystyle\int_{\mathcal{T}_j} \nabla u$ are then computed exactly using Gauss quadrature formulae. Consequently, we obtain for each triangle of the current mesh the mass and the gradient:

$$m_{K^{new}} = \int_{K_{new}} \Pi_1^c u = \sum_j \int_{\mathcal{T}_j} u \quad \text{and} \quad (\nabla\, \Pi_1^c u)_{|K^{new}} = \frac{\sum_j \int_{\mathcal{T}_j} \nabla u}{|K^{new}|}\,.$$

This reconstruction is conservative and $\mathbf{P}^1$ exact. It gives a $\mathbf{P}^1$ by element discontinuous solution. A specific treatment of the reconstruction is carried out to verify the maximum principle.

Verifying the maximum principle. Let K be a triangle of the new mesh. In the following, for clarity we denote by u_K the $\mathbf{P}^1$-conservative interpolated solution $\Pi_1^c u$ on K. The value at the barycenter and the gradient of the interpolated solution on K are given by:

$$u_K(G_K) = \frac{1}{|K|}\int_K u \quad \text{and} \quad \nabla u_K = \frac{1}{|K|}\int_K \nabla u_K\,.$$

Consequently, for each vertex P_i of the new mesh, we get a value of the solution using Taylor expansion for each element K of its ball:

$$u_K(P_i) = u_K(G_K) + \nabla u_K \cdot \overrightarrow{G_K P_i}\,. \tag{2}$$

A correction is applied to the linear representation of the solution on each element in order to verify the maximum principle. To this end, let $\mathcal{K}$ be the set of elements of the background mesh that K overlaps, see Figure 10, and let $\mathcal{Q}$ be the set of vertices of $\mathcal{K}$:

$$\mathcal{K} = \{K_j^{back} \,|\, K \cap K_j^{back} \neq \emptyset\} \quad \text{and} \quad \mathcal{Q} = \{Q_j \,|\, Q_j \in K^{back} \text{ such that } K^{back} \in \mathcal{K}\}\,.$$

Then, for each vertex P_i of each element K of the new mesh, the nodal value $u_K(P_i)$ verify the maximum principle if:

$$u_{\min} = \min_{Q\in\mathcal{Q}} u(Q) \leq u_K(P_i) \leq \max_{Q\in\mathcal{Q}} u(Q) = u_{\max}\,.$$

Notice that $u_K(G_K)$ satisfies the maximum principle. If a vertex does not verify the maximum principle on an element K then the gradient value of this element is corrected. Two approaches are proposed. One is based on the notion of limiter used in numerical schemes and the other one results from a minimization problem.

A first correction with limiters (I). For each element K, the limited vertex reconstruction is given by:

$$u_K^{\mathrm{I}}(P_i) = u_K(G_K) + \Phi_K \nabla u_K \cdot \overrightarrow{G_K P_i} = u_K(G_K) + \Phi_K(u_K(P_i) - u_K(G_K))\,,$$

where $\Phi_K \in [0,1]$ is defined by $\Phi_K = \min_{i=0,1,2} \phi_K(P_i)$, with

$$\phi_K(P_i) = \begin{cases} \min\left(1, \dfrac{u_{\max} - u_K(G_K)}{u_K(P_i) - u_K(G_K)}\right) & \text{if} \quad u_K(P_i) - u_K(G_K) > 0 \\ \min\left(1, \dfrac{u_{\min} - u_K(G_K)}{u_K(P_i) - u_K(G_K)}\right) & \text{if} \quad u_K(P_i) - u_K(G_K) < 0 \\ 1 & \text{if} \quad u_K(P_i) - u_K(G_K) = 0\,. \end{cases}$$

Another correction (II). We first reorder the indices such that $u_K(P_0) \leq u_K(P_1) \leq u_K(P_2)$. We then set

$$\begin{aligned} u_K^M(P_2) &= \min(u_K(P_2), u_{\max}) \\ u_K^M(P_1) &= \min(u_K(P_1) + \frac{1}{2}\max(0, u_K(P_2) - u_{\max}), u_{\max}) \\ u_K^M(P_0) &= 3u_K(G) - u_K^M(P_1) - u_K^M(P_2), \end{aligned}$$

and

$$\begin{aligned} \tilde{u}_K(P_0) &= \max(u_K^M(P_0), u_{\min}) \\ \tilde{u}_K(P_1) &= \max(u_K^M(P_1) - \frac{1}{2}\max(0, u_{\min} - u_K^M(P_0)), u_{\min}) \\ \tilde{u}_K(P_2) &= 3u_K(G) - \tilde{u}_K(P_0) - \tilde{u}_K(P_1). \end{aligned}$$

These new nodal values $\tilde{u}_K(P_i)$ define the corrected linear representation of the solution on K. For any points X included in K, its solution value is then given by:

$$u_K^{\mathrm{II}}(X) = \sum_{i=0}^{2} \beta_i(X)\tilde{u}_K(P_i)\,,$$

where $\beta_i(X)$ are the barycentric coordinates of X with respect to K. The final interpolated solution verifies all required properties:

Proposition 5.3. *Let $S \in \{I,II\}$. The reconstruction u_K^S satisfies the maximum principle, is linear preserving and is conservative. Moreover, we have:*

$$u_K(P_0) \leq u_K(P_1) \leq u_K(P_2) \quad \Longrightarrow \quad u_K^S(P_0) \leq u_K^S(P_1) \leq u_K^S(P_2)$$

and

$$u_{\min} \leq u_K(P_i) \leq u_{\max} \ \text{ for } \ i = 0,1,2 \quad \Longrightarrow \quad u_K^S(P_i) = u_K(P_i) \ \text{ for } \ i = 0,1,2.$$

Notice that the reconstruction II comes from a minimization problem. Indeed, we have

Proposition 5.4. *Suppose that $u_K(P_0) \leq u_K(P_1) \leq u_K(P_2)$ and that $u_{\max} < u_K(P_2)$. Then, we have*

$$\sum_{i=0}^{2} |u_K(P_i) - u_K^M(P_i)|^2 \leq \sum_{i=0}^{2} |u_K(P_i) - v_K(P_i)|^2$$

for all the linear reconstructions v_K satisfying $v_K(P_2) = u_{\max}$, $v_K(P_i) \leq u_{\max}$ for $i = 0,1$ and $\int_K v_K = \int_K u_K$.

The proofs of these propositions are given in the Appendix.

5.2.2. Solution defined at vertices When the solution is given at vertices of the background mesh, *i.e.,* nodal values are provided, it defines implicitly a piecewise linear continuous representation of the solution at the elements. Therefore, the $\mathbf{P}^1$-conservative interpolation defined in the previous section can be applied. A piecewise linear solution by elements, which is generally discontinuous, is then obtained on the new mesh. Now, one more stage is required to retrieve a solution at vertices of the new mesh. This solution transferred from elements to vertices must preserve the properties of the interpolation scheme.

The solution is simply re-distributed to each vertex P of the new mesh by averaging:

$$u^S(P) = \frac{\sum_{K_i^{new} \ni P} |K_i^{new}| u_{K_i^{new}}^S(P)}{\sum_{K_i^{new} \ni P} |K_i|},$$

where $S \in$ {I,II} depends on the chosen reconstruction and u is the interpolated solution on the new mesh. Notice that after re-distribution to vertices the interpolated solution still satisfies the maximum principle, is linear preserving and is conservative.

Remark 5.3. *As the mass of the solution is linked to the topology of the mesh, the $\mathbf{P}^1$-conservative interpolation operator Π_1^c depends on the mesh topology on which it is applied. In consequence, it cannot be applied to interpolate solution at any points of a given domain.*

5.2.3. Non-matching discrete boundaries Let Ω be a bounded domain of $\mathbb{R}^2$ and $\mathcal{H}^{back}$ and $\mathcal{H}^{new}$ two meshes of Ω. We consider the case where the discrete boundaries of $\mathcal{H}^{back}$ and $\mathcal{H}^{new}$ do not match. In other words, $\mathcal{H}^{back}$ and $\mathcal{H}^{new}$ are meshes of two different polygonal domains Ω_h^{back} and Ω_h^{new} the boundary of which differs: $\Gamma_h^{back} \neq \Gamma_h^{new}$. Therefore, the volume of each mesh differs: $|\mathcal{H}_{back}| \neq |\mathcal{H}_{new}|$.

In the conservative interpolation, the non-matching discrete boundaries are handled differently depending on the solution behavior. We consider the following specific cases:

1. if the solution is constant or linear, we require to keep a constant or linear solution to satisfy the $\mathbf{P}^1$-exactness property. If an uniform field is given as initial set, we want to return an uniform solution set. In this case, the conservation principle is violated.
2. else, if the solution does not vary linearly, then we verify the conservation principle by modifying the solution.

6. PSEUDO-CONSERVATIVE INTERPOLATION

In this section, two pseudo-conservative interpolation schemes are proposed for readers who want an intermediate interpolation scheme between the classical (Section 4) and the conservative (Section 5) ones. These interpolations are just given as possible alternative approaches but they are not compared to other approaches in the numerical examples sections.

The idea is to use a high-order local interpolation to compute an accurate mass on each element of the new mesh. We denote by Π_k^{pc} the k-order pseudo-conservative interpolation operator. More precisely, for each element K of the new mesh, a linear interpolation is performed for its center of gravity to get the solution and thus an initial guess of its mass:

$$m_K^1 = \int_K \Pi_1^{pc} u = |K| \, \Pi_1 u(G) \, ,$$

where $\Pi_1 u(G)$ is the linearly interpolated solution at point G. This defines the first order local pseudo conservative operator Π_1^{pc}. Then, the order of interpolation is locally increased on this element to compute the element mass until a given threshold ε:

$$\text{reach order } \; k \; \text{ such that } \; \frac{|m_K^k - m_K^{k-1}|}{|K|} = \frac{|\int_K (\Pi_k^{pc} u - \Pi_{k-1}^{pc} u)|}{|K|} < \varepsilon \, .$$

Finally, the approximated element mass is $m_K^k = \int_K \Pi_k^{pc} u \approx m_K$. This approach is not conservative but the amount of error is controlled by the threshold. It gives a tolerance.

We now specify how high-order interpolations are defined. The first one is based on Gauss quadrature formulae and the second on Lagrange polynomials.

Gauss quadrature approach. The local projection order is increased by considering higher-order Gauss quadrature formulae. Such quadratures rules for triangles are given in Table I from order 1 to 5. The approximated mass at order k is then given by:

$$m_K \approx m_K^k = \int_K \Pi_k^{pc} u(x) dx = \sum_{i=1}^{n_{gp}} \omega_i \Pi_1 u(G_i) \, ,$$

where n_{pg} is the number of Gauss points, $G_i = \sum_{j=1}^{3} \beta_j P_j$ the Gauss points and ω_i their weights. Solutions at the Gauss points $\Pi_1 u(G_i)$ are obtained by linear interpolation after localization of points G_i in the background mesh.

Lagrange polynomials approach. For a k-order interpolation, the current triangle of new mesh is split into 4^{k-1} sub-triangles in a Lagrangian fashion. Iso-barycenters of each sub-triangle are localized. Then, a linear interpolation is performed to get the solution at these

centers of gravity. The approximated mass at order k is given by:

$$m_K \approx \tilde{m}_K^k = \int_K \Pi_k^{pc}(x)dx = \sum_{i=1}^{4^{k-1}} \frac{|K|}{4^{k-1}} \Pi_1 u(G_i)\,.$$

The weakness of this approach is to have 64 points to localize and interpolate at order 4 and even more after. Consequently, it becomes rapidly time consuming.

7. ACCURACY AND CONVERGENCE STUDY ON ANALYTICAL FUNCTIONS

In this section, the behavior of the $\mathbf{P}^1$-conservative interpolation is analyzed on four analytical functions defined on the domain $[-1, 1]$ which are representative of several physical phenomena encountered in computational fluid dynamics (CFD). The $\mathbf{P}^1$-conservative interpolation is compared to the linear interpolation, in particular the mass conservation and the convergence order of the schemes are studied.

To perform this analysis, two meshes $\mathcal{H}_1^1$ and $\mathcal{H}_1^2$, composed respectively of 631 and 611 vertices, are considered. These meshes are completely different and unconnected. They are depicted in Figure 11. In order to study the methods convergence order, each mesh spans a series of embedded meshes denoted $(\mathcal{H}_i^1)_{i=1\ldots5}$ and $(\mathcal{H}_i^2)_{i=1\ldots5}$. At each time, the mesh $\mathcal{H}_{i+1}^j$

$\mathcal{O}_{pg}$	n_{pg}	β_j	multiplicity	weight ω_i
1	1	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	1	$\lvert K\rvert$
2	3	$(\frac{1}{6}, \frac{1}{6}, \frac{2}{3})$	3	$\frac{1}{3}\lvert K\rvert$
3	4	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	1	$-\frac{9}{16}\lvert K\rvert$
		$(\frac{1}{5}, \frac{1}{5}, \frac{3}{5})$	3	$\frac{25}{48}\lvert K\rvert$
4	6	$(a_i, a_i, 1-2\,a_i)$ for $i = 1, 2$	3	
		$a_1 = 0.445948490915965$		$0.223381589678010\,\lvert K\rvert$
		$a_2 = 0.091576213509771$		$0.109951743655322\,\lvert K\rvert$
5	7	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	1	$\frac{9}{40}\lvert K\rvert$
		$(a_i, a_i, 1-2\,a_i)$ for $i = 1, 2$	3	
		$a_1 = \frac{6-\sqrt{15}}{21}$		$\frac{155-\sqrt{15}}{1200}\lvert K\rvert$
		$a_2 = \frac{6+\sqrt{15}}{21}$		$\frac{155+\sqrt{15}}{1200}\lvert K\rvert$

Table I. *Gauss quadrature formulae for triangles from [10]. $\mathcal{O}_{pg}$ is the quadrature order, n_{pg} is the number of Gauss points, β_i are the barycentric coordinates that define the Gauss points and ω_i their associated weights.*

Step	# vertices $\mathcal{H}_i^1$	# vertices $\mathcal{H}_i^2$
1	631	611
2	2,444	2,366
3	9,619	9,311
4	38,165	36,941
5	152,041	147,161

Table II. *Mesh sizes of the series of embedded mesh* $(\mathcal{H}_i^1)_{i=1\ldots5}$ *and* $(\mathcal{H}_i^2)_{i=1\ldots5}$.

is deduced from $\mathcal{H}_i^j$ by splitting each triangle into four triangles in a Lagrangian fashion, *i.e.,* in a *isoparametric* way. These series of meshes are summarized in Table II.

For each case, the function is applied on $\mathcal{H}_i^1$ providing a solution field u_i^1. This solution field is transfered from $\mathcal{H}_i^1$ to $\mathcal{H}_i^2$, we get Πu_i^2. This solution transfer is called transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2$. The error is computed by comparing the interpolated solution Πu_i^2 with the function applied on $\mathcal{H}_i^2$, *i.e.,* u_i^2, in $\mathbf{L}^2$-norm:

$$\varepsilon_i^2 = \frac{\int_{\mathcal{H}_i^2}(u_i^2 - \Pi u_i^2)^2}{\int_{\mathcal{H}_i^2}(u_i^2)^2},$$

where the integrals are computed using a 5-order Gaussian quadrature. The series of errors enable a convergence study.

We have also analyzed the error when the solution field is re-interpolated back to $\mathcal{H}_i^1$. That is to say, the function is applied on $\mathcal{H}_i^1$ giving u_i^1, then it is interpolated on $\mathcal{H}_i^2$ giving Πu_i^2 and finally Πu_i^2 is interpolated from $\mathcal{H}_i^2$ to $\mathcal{H}_i^1$ and we obtain Πu_i^1. The error ϵ_i is obtained by computing the gap in $\mathbf{L}^2$-norm between u_i^1 and Πu_i^1 on $\mathcal{H}_i^1$. This double solution transfer is called transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2 \rightarrow \mathcal{H}_i^1$.

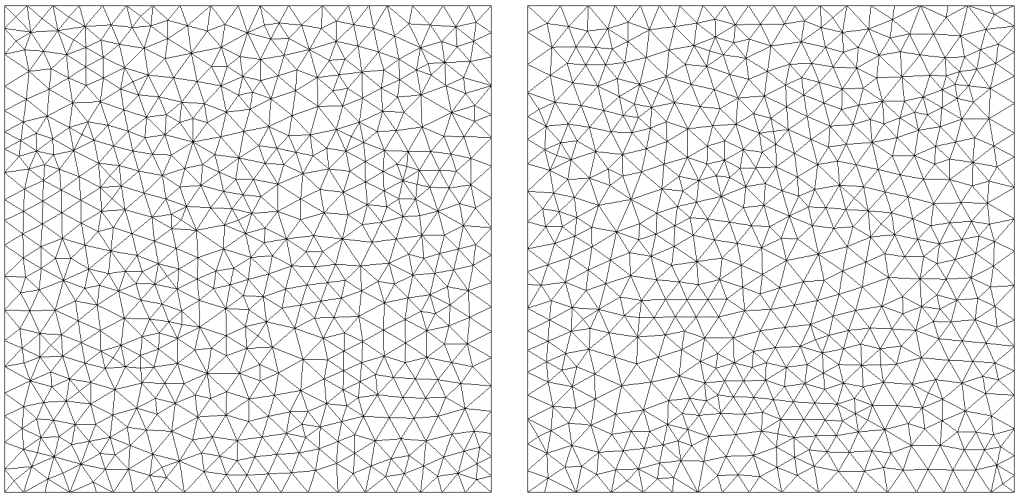

Figure 11. *Uniform unstructured triangular meshes used to compare interpolation schemes. Left,* $\mathcal{H}_1^1$ *containing* 631 *vertices, and right,* $\mathcal{H}_1^2$ *containing* 611 *vertices.*

For each analytical function, a figure is given providing:

- top left, a three-dimensionnal representation of the function
- top right, the mass variation for the solution transfer $\mathcal{H}_i^1 \to \mathcal{H}_i^2$
- bottom left, the error ε_i for the solution transfer $\mathcal{H}_i^1 \to \mathcal{H}_i^2$
- and bottom right, the error ϵ_i for the solution transfer $\mathcal{H}_i^1 \to \mathcal{H}_i^2 \to \mathcal{H}_i^1$.

The linear interpolation scheme is represented by the red lines and the $\mathbf{P}^1$-conservative interpolation is represented by the blue lines.

A Gaussian function. The first analytical function is a gaussian given by:

$$f_1(x, y) = \exp(-30\,(x^2 + y^2))\,.$$

This function is smooth and is representative of the vortices encountered in CFD, Figure 12.

The mass variation with the classical linear interpolation converges toward zero. It has a variation lower than 10^{-5} for meshes with more than $35,000$ vertices. Conversely, the mass

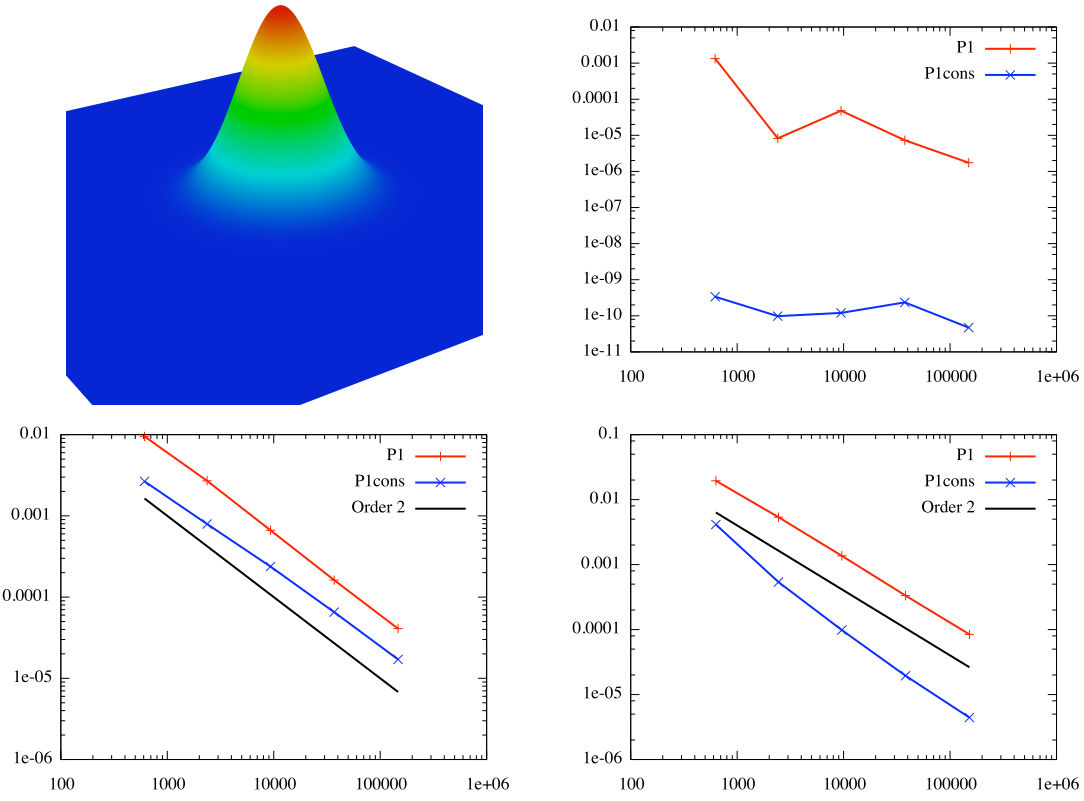

Figure 12. *Gaussian analytical function f_1. Top left, a three-dimensionnal representation of the function. Top right, the mass variation for the transfer $\mathcal{H}_i^1 \to \mathcal{H}_i^2$. Bottom left, the error ε_i for the transfer $\mathcal{H}_i^1 \to \mathcal{H}_i^2$. Bottom right, the error ϵ_i for the transfer $\mathcal{H}_i^1 \to \mathcal{H}_i^2 \to \mathcal{H}_i^1$.*

variation with the $\mathbf{P}^1$-conservative interpolation is of the order of the numerical zero ($\approx 10^{-10}$) for all interpolation steps.

As regards the accuracy and the convergence order, both interpolation scheme are converging at order 2 for solution transfers $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2$ and $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2 \rightarrow \mathcal{H}_i^1$. This fits to the theory. We notice that the $\mathbf{P}^1$-conservative interpolation is more accurate than the linear one in both cases. The difference in accuracy varies between 2 and 3 for the solution transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2$ and between 3 and 12 for the transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2 \rightarrow \mathcal{H}_i^1$.

A continuous sinusoidal shock. This analytical function represents a continuous model of a shock which can be assimilated to the numerical capture of a shock with a dissipative flow solver, *i.e.*, the solver captures the shock on several mesh elements. This smooth function is given by:

$$f_2(x, y) = \tanh(100\,(y + 0.3\,\sin(-2\,x)))\,.$$

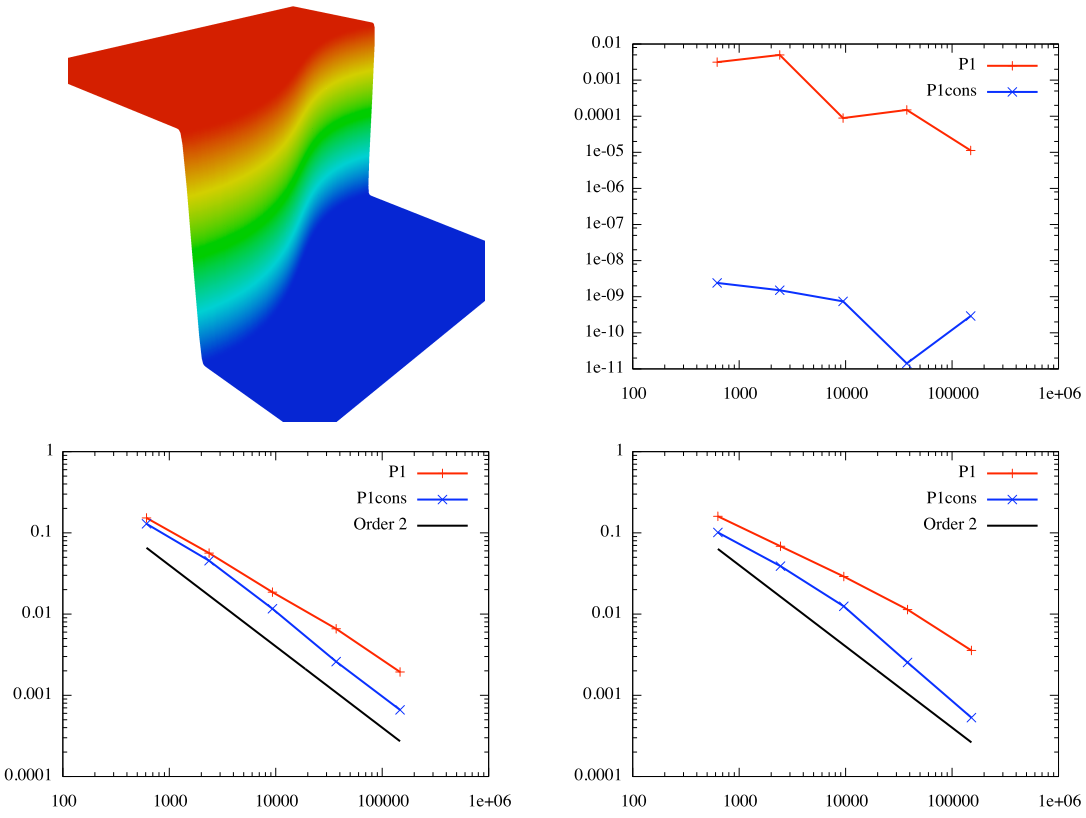

Figure 13. *Continuous sinusoidal shock analytical function f_2. Top left, a three-dimensionnal representation of the function. Top right, the mass variation for the transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2$. Bottom left, the error ε_i for the transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2$. Bottom right, the error ϵ_i for the transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2 \rightarrow \mathcal{H}_i^1$.*

It contains two quasi-constant regions that are separated by a sinusoidal region in which a strong gradient variation occurs continuously.

As previously, the mass variation with the $\mathbf{P}^1$-conservative interpolation is of the order of the numerical zero ($\approx 10^{-9}$) for all interpolation steps. For the linear interpolation the mass variation decreases when mesh accuracy increases. The mass variation is almost 10^{-5} for meshes with more than one hundred thousand vertices. Notice that for coarser meshes (the first two steps) the mass variation is between 0.3 and 0.6% for just one solution transfer.

The $\mathbf{P}^1$-conservative interpolation achieves an order 2 of convergence for solution transfers $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2$ and $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2 \rightarrow \mathcal{H}_i^1$ whereas the linear interpolation has a convergence order less than 2. The convergence order is almost 1.6 in both cases. Actually, meshes are not fine enough to reach the asymptotical mesh convergence order for the linear interpolation. As regards the accuracy, the $\mathbf{P}^1$-conservative interpolation is more accurate than the linear one in all cases and the difference increases while meshes are refined.

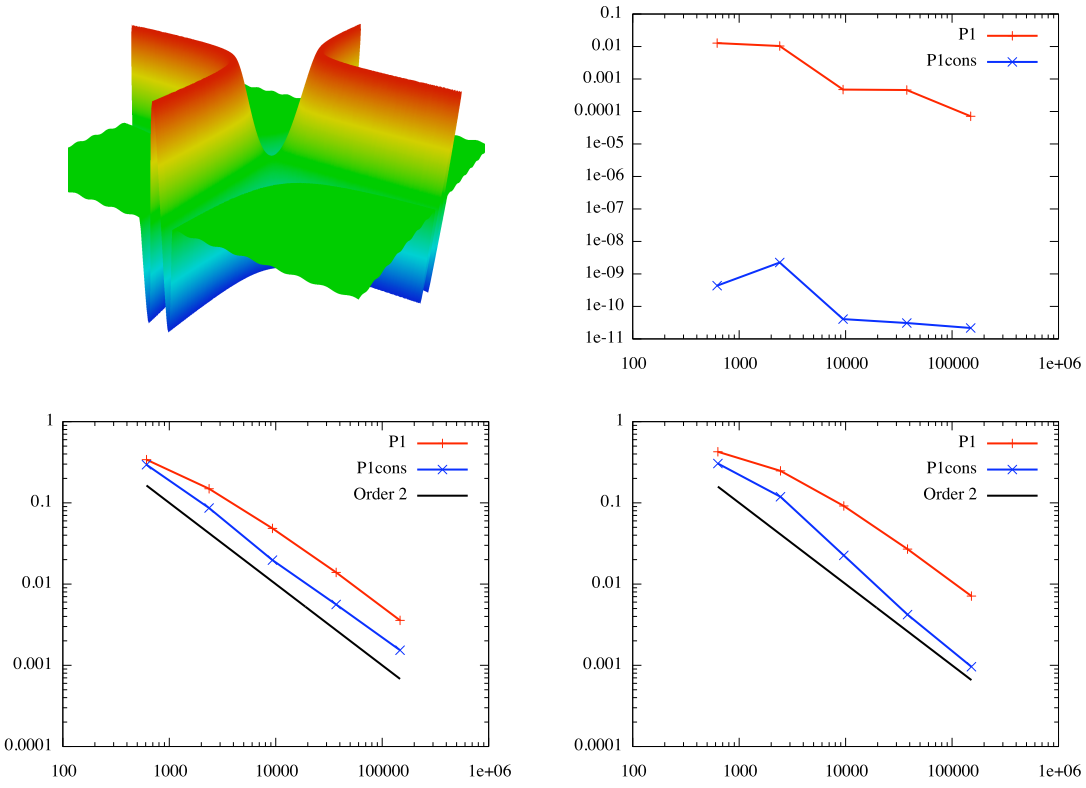

Figure 14. *Multi-scales smooth analytical function f_3. Top left, a three-dimensionnal representation of the function. Top right, the mass variation for the transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2$. Bottom left, the error ε_i for the transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2$. Bottom right, the error ϵ_i for the transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2 \rightarrow \mathcal{H}_i^1$.*

A multi-scales smooth function. This function presents smooth sinusoidal variations but at different scales. There are two order of magnitudes between small and large scales variations. This function reads:

$$f_3(x,y) = \begin{cases} 0.01\,\sin(50\,x\,y) & \text{if} \quad x\,y \leq \dfrac{-\pi}{50} \\ \sin(50\,x\,y) & \text{if} \quad \dfrac{-\pi}{50} < x\,y \leq \dfrac{2\,\pi}{50} \\ 0.01\,\sin(50\,x\,y) & \text{if} \quad \dfrac{2\,\pi}{50} < x\,y \end{cases} .$$

The mass variation with the $\mathbf{P}^1$-conservative interpolation is still of the order of the numerical zero ($\approx 10^{-10}$) for all interpolation steps. For the linear interpolation, the mass variation is large from 1% for step 1 to 0.01% for step 5 for only one solution transfer. However, the mass variation still converges toward zero while the mesh size converges toward zero.

For this smooth case, the two approaches reach an order 2 of convergence for solution transfers $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2$ and $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2 \rightarrow \mathcal{H}_i^1$. Concerning the accuracy, the conclusions are the

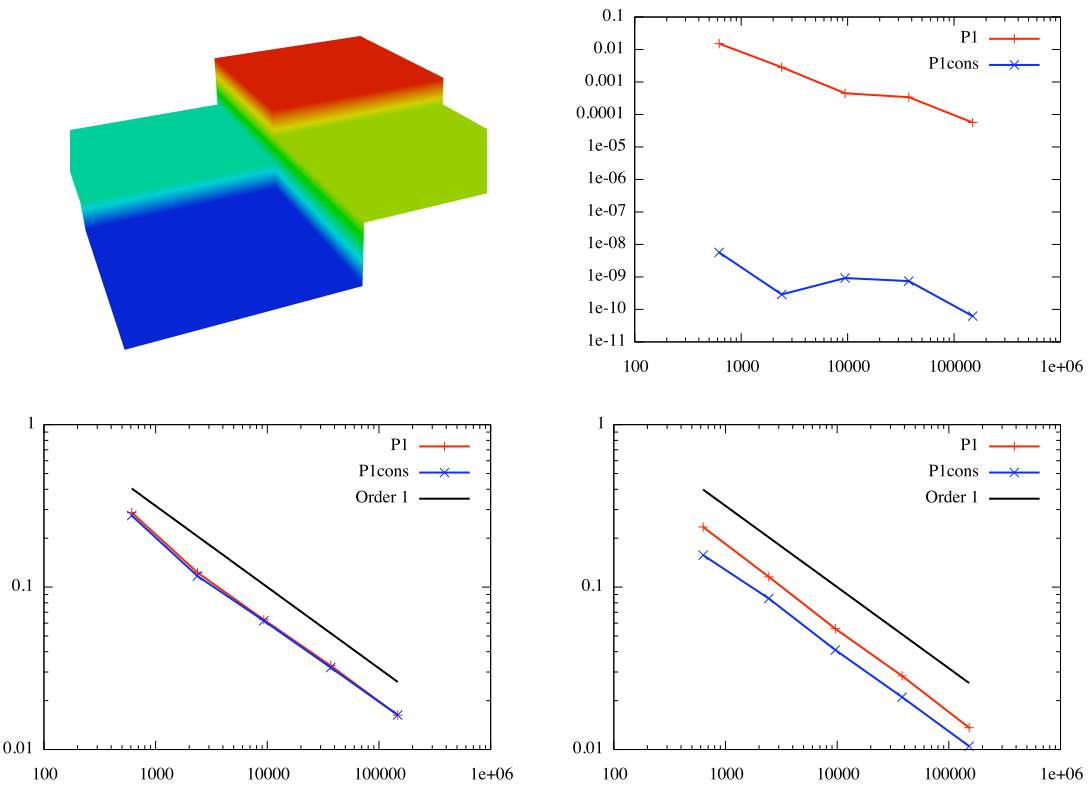

Figure 15. *Discontinuous analytical function f_4. Top left, a three-dimensionnal representation of the function. Top right, the mass variation for the transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2$. Bottom left, the error ε_i for the transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2$. Bottom right, the error ϵ_i for the transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2 \rightarrow \mathcal{H}_i^1$.*

same as in the previous cases. The $\mathbf{P}^1$-conservative interpolation achieves better accuracy than the standard linear interpolation and even more for the solution transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2 \rightarrow \mathcal{H}_i^1$ where the error is reduced by an order of magnitude for the finest meshes (step 4 and 5).

A discontinuous function. The final function is discontinuous and represents four steps:

$$f_4(x,y) = \begin{cases} 1 & \text{if} \quad x \geq 0 \quad \text{and} \quad y \geq 0 \\ 2 & \text{if} \quad x \geq 0 \quad \text{and} \quad y < 0 \\ 3 & \text{if} \quad x < 0 \quad \text{and} \quad y \geq 0 \\ 4 & \text{if} \quad x < 0 \quad \text{and} \quad y < 0 \end{cases} .$$

The solution is constant in four squared regions and it is discontinuous at the interface of each region.

The mass variation with the $\mathbf{P}^1$-conservative interpolation, in this case too, is of the order of the numerical zero for all the interpolation steps. For the linear interpolation, the mass variation is large from 1% for step 1 to 0.01% for step 5 for only one solution transfer. However, it still converges toward zero while the size approaches zero.

Even if the mass is preserved, while transferring the solution from a mesh to a finer one, *i.e.,* the solution transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2$, the same accuracy is obtained for both approaches. Nevertheless, for the solution transfer $\mathcal{H}_i^1 \rightarrow \mathcal{H}_i^2 \rightarrow \mathcal{H}_i^1$ the $\mathbf{P}^1$-conservative interpolation performs better than the classical linear approach.

Conclusions. For all those analytical cases, while preserving the mass, the $\mathbf{P}^1$-conservative interpolation achieves better accuracy than the classical linear interpolation and for some cases it converges at a faster rate.

8. APPLICATION TO MESH ADAPTATION

Mesh adaptation provides a way of controlling the accuracy of the numerical solution by modifying the domain discretization according to size and directional constraints. It is well known that mesh adaptation captures accurately physical phenomena in the computational domain while reducing significantly the cpu time, see [6, 11, 12].

8.1. Unsteady mesh adaptation scheme

Anisotropic mesh adaptation is a non-linear problem. Therefore, an iterative procedure is required to solve this problem. For stationary simulations, an adaptive computation is carried out *via* a mesh adaptation loop inside which an algorithmic convergence of the mesh-solution couple is sought.

At each stage, a numerical solution is computed on the current mesh with a flow solver and has to be analyzed by means of an error estimate. The considered error estimate aims at minimizing the global interpolation error in norm $\mathbf{L}^p$, thus it is independent of the problem at hand. From the multi-scales metric theory in [13] and [12], an analytical expression of the optimal metric is exhibited in two dimensions that minimizes the interpolation error in norm

$\mathbf{L}^p$:

$$\mathcal{M}_{\mathbf{L}^p} = D_{\mathbf{L}^p} \left(\det |H_u|\right)^{\frac{-1}{2p+2}} \mathcal{R}_u^{-1} |\Lambda| \mathcal{R}_u \quad \text{with} \quad D_{\mathbf{L}^p} = 2\,\varepsilon^{-1} \left(\int_\Omega (\det |H_u|)^{\frac{p}{2p+2}} \right)^{\frac{1}{p}}, \qquad (3)$$

where ε is the prescribed error threshold. This anisotropic metric is a function of the Hessian of the solution which is reconstructed from the numerical solution by a double $\mathbf{L}^2$ projection. This metric will replace the Euclidean one modifying the scalar product that underlies the notion of distance used in mesh generation algorithms. Next, an adapted mesh is generated with respect to this metric where the aim is to generate a mesh such that all its edges have a length of (or close to) one in the prescribed metric and such that all its elements are almost regular. Such a mesh is called a *unit mesh*. Here, the mesh is adapted by local mesh modifications of the previous mesh (the mesh is not regenerated) using classical mesh operations: vertex insertion, edge and face swap, collapse and node displacement [8]. Finally, the solution is interpolated on the new mesh using one of the interpolation schemes presented in this paper. This procedure is repeated until the convergence of the solution and of the mesh is achieved. This algorithm is represented by the external loop of the mesh adaptation scheme in Figure 16.

To solve the non-linear problem of mesh adaptation for unsteady simulation, a new algorithm generalizing the mesh adaptation scheme coupled with a metric intersection in time procedure has been proposed in [6]. This procedure, based on the resolution of a transient fixed point problem for the couple mesh-solution at each iteration of the mesh adaptation loop, predicts the solution evolution in the computational domain. Knowing then the solution evolution during a short period of time, the mesh is suitably adapted in all the regions where the solution progresses so as to preserve its accuracy.

This iterative algorithm consists of two steps: the main adaptation loop and an internal loop in which the transient fixed point problem is solved. At each iteration of the main adaptation loop, we consider a time period $[t, t + \Delta t]$ in which the solution evolves. During this period, we try to algorithmically converge to the solution at $t + \Delta t$ and to the associated adapted mesh. In other words, from the solution at time t, we compute the solution to time $t + \Delta t$, and the computation is iterated *via* the internal loop until the desired accuracy is obtained for the solution at $t + \Delta t$. Similarly, we algorithmically converge toward the corresponding invariant mesh adapted to this period $[t, t+\Delta t]$ throughout a sequence of consecutively adapted meshes. The solution behavior is thus predicted in all the regions of the domain where the solution evolves. To take into account the solution progression, a metric is defined by means of an intersection procedure in time. More precisely, metrics associated to several solutions throughout the time period $[t, t+\Delta t]$ are evaluated and intersected into a unique one. Then, a new mesh is generated according to this metric field. Finally, the initial solution of this period is interpolated and the computation is resumed. This scheme, illustrated in Figure 16, controls the spatial and the time error during the whole computation.

8.1.1. Flow solver In all the examples, the flow is modeled by the conservative Euler equations. Assuming that the gas is perfect, inviscid and that there is no thermal diffusion,

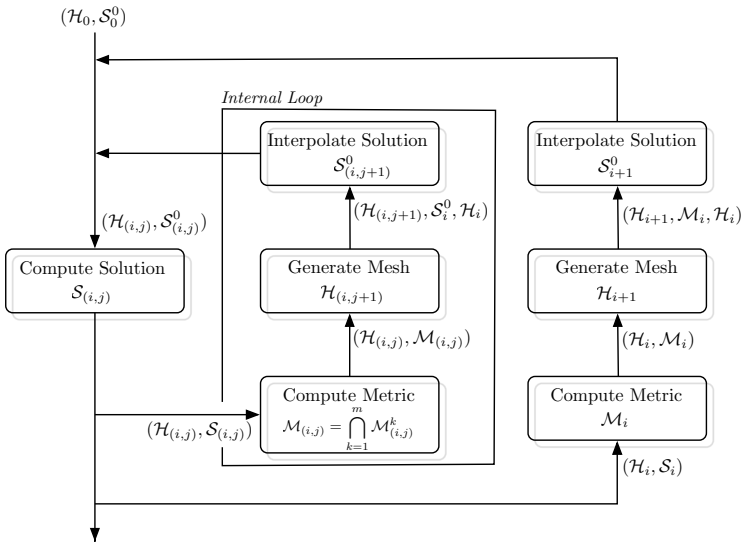

Figure 16. *Unsteady mesh adaptation scheme.*

the Euler equations for mass, momentum and energy conservation read:

$$\left\{ \begin{array}{rcl} \dfrac{\partial \rho}{\partial t} + \nabla.(\rho \vec{U}) & = & \mathcal{S}_\rho \,, \\ \dfrac{\partial (\rho \vec{U})}{\partial t} + \nabla.(\rho \vec{U} \otimes \vec{U}) + \nabla p & = & \mathcal{S}_{\rho \vec{U}} \,, \\ \dfrac{\partial (\rho E)}{\partial t} + \nabla.((\rho E + p)\vec{U}) & = & \mathcal{S}_{\rho E} \,, \end{array} \right.$$

where ρ denotes the density, $\vec{U} = (u, v)$ the velocity vector, $E = T + \frac{\|\vec{U}\|^2}{2}$ the total energy and $p = (\gamma - 1)\rho T$ the pressure with $\gamma = 1.4$ the ratio of specific heats and T the temperature. $\mathcal{S} = {}^t(\mathcal{S}_\rho, \mathcal{S}_{\rho u}, \mathcal{S}_{\rho v}, \mathcal{S}_{\rho E})$ is a source term depending on the problem. These equations could be symbolically rewritten:

$$\frac{\partial W}{\partial t} + \nabla \cdot F(W) = \mathcal{S} \,, \tag{4}$$

where $W = {}^t(\rho, \rho u, \rho v, \rho E)$ is the conservative variables vector and the vector F represents the convective operator.

The Euler system is solved by means of a Finite Volume technique on unstructured triangular meshes. The proposed scheme is vertex-centered and uses a particular edge-based formulation with upwind elements. This formulation consists in associating to each vertex of the mesh a control volume (or finite-volume cell) built by the rule of medians. This flow solver employes a HLLC approximate Riemann solver to compute the numerical fluxes, see [14].

A high-order scheme is derived according to a MUSCL (Monotone Upwind Schemes for Conservation Laws) type method using downstream and upstream tetrahedra [15]. This approach is compatible with vertex-centered and edge-based formulations, allowing rather easy

and, importantly, inexpensive higher-order extensions of monotone upwind schemes. The flux integration based on the edges and their corresponding upwind elements (crossed by the edge) is a key-feature in order to preserve the positivity of the density for vertex-centered formulation as demonstrated in [16]. Appropriate β-schemes are used for the variable extrapolation which gives us a third-order space-accurate scheme for the linear advection on cartesian triangular meshes, see [17]. This approach provides low diffusion second-order space-accurate scheme in the non-linear case. The MUSCL type method is combined with a generalization of the Superbee limiter with three entries to guarantee the TVD (Total Variation Diminishing) property of the scheme [16].

An explicit time stepping algorithm is used by means of a 4-stage, 3-order strong-stability-preserving (SSP) Runge-Kutta scheme which allows us to use a CFL coefficient up to 2 [18]. Such time discretization methods have non linear stability properties which are particularly suitable for the integration of system of hyperbolic conservation laws where discontinuities appear. These schemes verify the TVD property. In practice, we consider a CFL equal to 1.8.

More details can be found in [19].

8.2. Spherical blast

This example is a spherical Riemann problem between two parallel walls simulating a blast proposed by Langseth and LeVeque [20]. Initially, the gas is at rest with density $\rho_{out} = 1$ and pressure $p_{out} = 1$ everywhere except in a sphere centered at $(0, 0, 0.4)$ with radius 0.2. Inside the sphere the parameters are $\rho_{in} = 1$ and $p_{in} = 5$. For both regions, we have $\gamma = 1.4$. As mentioned in [20], the initial pressure jump results in a strong outward moving shock wave, an outward contact discontinuity and an inward moving rarefaction wave. The main feature of the solution are the interactions between these waves. Another significant feature is the development of a low density region in the center of the domain with the development of instabilities along the contact discontinuity.

As the solution remains cylindrically symmetric throughout the simulation, it is possible to formulate it as a two-dimensional problem with a source term where $\mathcal{S}$ is given by:

$$\mathcal{S} = -\frac{\text{sign}(x)}{r}{}^t\big(\rho u, \rho u^2, \rho uv, u(\rho E + p)\big)\,,$$

where $r = \sqrt{x^2 + (y - 0.4)^2}$ represents the distance to the center of the sphere. The sign of x is considered in the source term as we solve the problem on the entire domain $[-1.5, 1.5] \times [0, 1]$. The solution is computed until a-dimensioned time $t = 0.7$.

Error threshold $\mathcal{E}$	$\mathbf{P}^1$	$\mathbf{P}^1$-conservative
0.05	6,360	7,702
0.04	10,424	12,102
0.03	19,003	24,297
0.02	65,740	78,099
0.015	118,929	158,121

Table III. *Number of vertices of the final adapted meshes obtained with the unsteady mesh adaptation algorithm for the* $\mathbf{P}^1$ *and the* $\mathbf{P}^1$*-conservative interpolation for several error thresholds.*

For this problem, the $\mathbf{L}^2$-norm of the density interpolation error is controlled. The global simulation is split into 20 time periods of time length $\Delta t = 0.035$ for which a transient fixed-point problem is solved throughout 5 internal loop iterations. This simulation has been performed for five error thresholds, ranging from $\varepsilon = 0.05$ to $\varepsilon = 0.015$. The number of metric intersections in time varies between 10 and 20 depending on the error threshold. Two series of adaptations have been performed: one with the classic $\mathbf{P}^1$ interpolation and the other one with the $\mathbf{P}^1$-conservative interpolation scheme. The statistics of the final ($t = 0.7$) adapted meshes are given in Table III.

In the following, we will analyze the impact of the conservative interpolation on the solution accuracy. We designate by *reference solution* the adaptive solution at time $t = 0.7$ computed with an error threshold $\varepsilon = 0.015$ and the $\mathbf{P}^1$-conservative interpolation. Figure 17 shows a schlieren picture representing the reference solution density at $t = 0.7$. This picture imitates a photographic technique used in physical experiments. The final adapted mesh at $t = 0.7$ associated to the reference solution is depicted in Figure 18 (top). A close up view of the adapted mesh in a region where instabilities occur is given in Figure 20 (right). Notice that in some regions the accuracy of the mesh attains $h = 2.\, e^{-4}$.

Several items illustrate the gain in accuracy of the conservative interpolation with respect to the classical one. These items point out that the solution has been less diffused during the interpolation stage and it thus results in a more accurate solution.

The gain in accuracy is demonstrated in Figure 20 (left) where the density errors in $\mathbf{L}^2$-norm with respect to the reference solution are drawn for the adaptive simulations with an error threshold from $\varepsilon = 0.05$ to $\varepsilon = 0.02$ for the conservative and the classical interpolations. We notice that the error has been lowered with the conservative interpolation.

This is also illustrated in Figure 19 where the pressure isolines are represented for the reference solution (top) and the adaptive solutions with an error of $\varepsilon = 0.03$ for the $\mathbf{P}^1$-conservative interpolation (middle) and the $\mathbf{P}^1$-interpolation (bottom). The solution with the $\mathbf{P}^1$-conservative interpolation is slightly more detailed showing that it is less dissipative.

This impact is also emphasized by the mesh size of the final adapted of each simulation. For a given error threshold, we remark that using the $\mathbf{P}^1$-conservative interpolation results

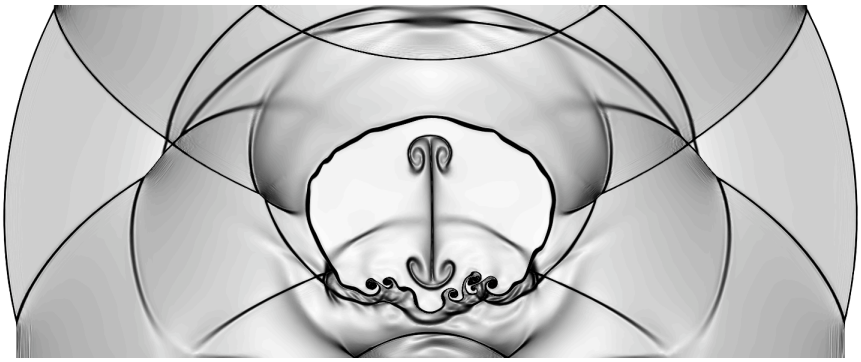

Figure 17. *A schlieren type picture representing the final density at* $t = 0.7$ *on the final adapted mesh containing* $158,121$ *vertices obtained from an error level of* 0.015 *and the* $\mathbf{P}^1$*-conservative interpolation.*

in a larger final mesh. As the solution is less dissipated during the interpolation stage, more details (phenomena) of the solution are preserved and thus the adaptive process generates larger meshes.

Notice that for this example, it is impossible to examine the impact on the mass conservation due to the source term that modifies the mass of each variable at each iteration.

As regards the cpu time, both interpolation schemes have been compared on several couples of meshes on a Intel Core 2 at 2.8 GHz. All the cases are summarized in Table IV. It follows that the $\mathbf{P}^1$-conservative interpolation is approximately 5 times slower than the $\mathbf{P}^1$ interpolation. Nevertheless, the cost of the interpolation stage (a few seconds) is always negligible as compared to the solver cpu time.

# vertices $\mathcal{H}_i^{back}$	# vertices $\mathcal{H}_i^{new}$	$\mathbf{P}^1$-cons cpu time in sec.	$\mathbf{P}^1$ cpu time in sec.	ratio
$7,682$	$7,702$	0.235	0.046	5.1
$11,789$	$12,102$	0.298	0.061	4.9
$22,964$	$24,297$	0.564	0.104	5.4
$71,735$	$78,099$	2.192	0.293	7.5
$140,603$	$158,121$	4.839	0.971	5.0

Table IV. *Cpu time comparison between the $\mathbf{P}^1$-conservative interpolation and the $\mathbf{P}^1$ interpolation on several couples of meshes on a Intel Core 2 at 2.8 GHz. The cpu time is expressed in seconds.*

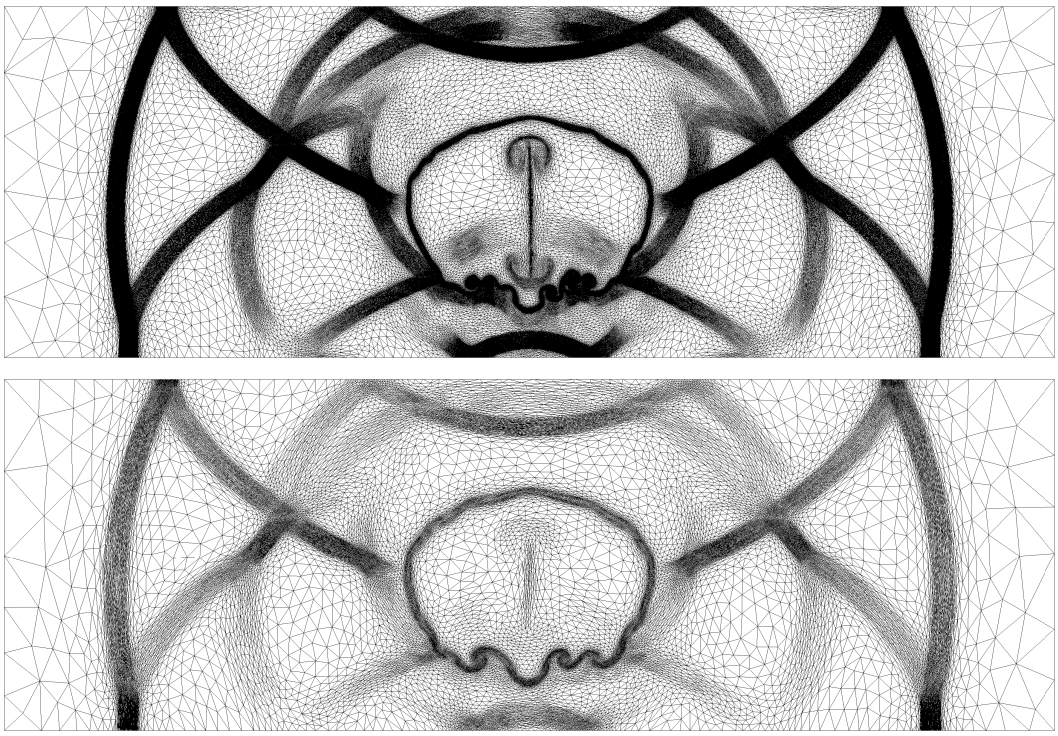

Figure 18. *Final adapted meshes using the* $\mathbf{P}^1$*-conservative interpolation at time* $t = 0.7$ *for errors equal to* 0.015 *(top) and* 0.03 *(bottom). The upper mesh contains* $158,121$ *vertices and the lower one* $24,297$*.*

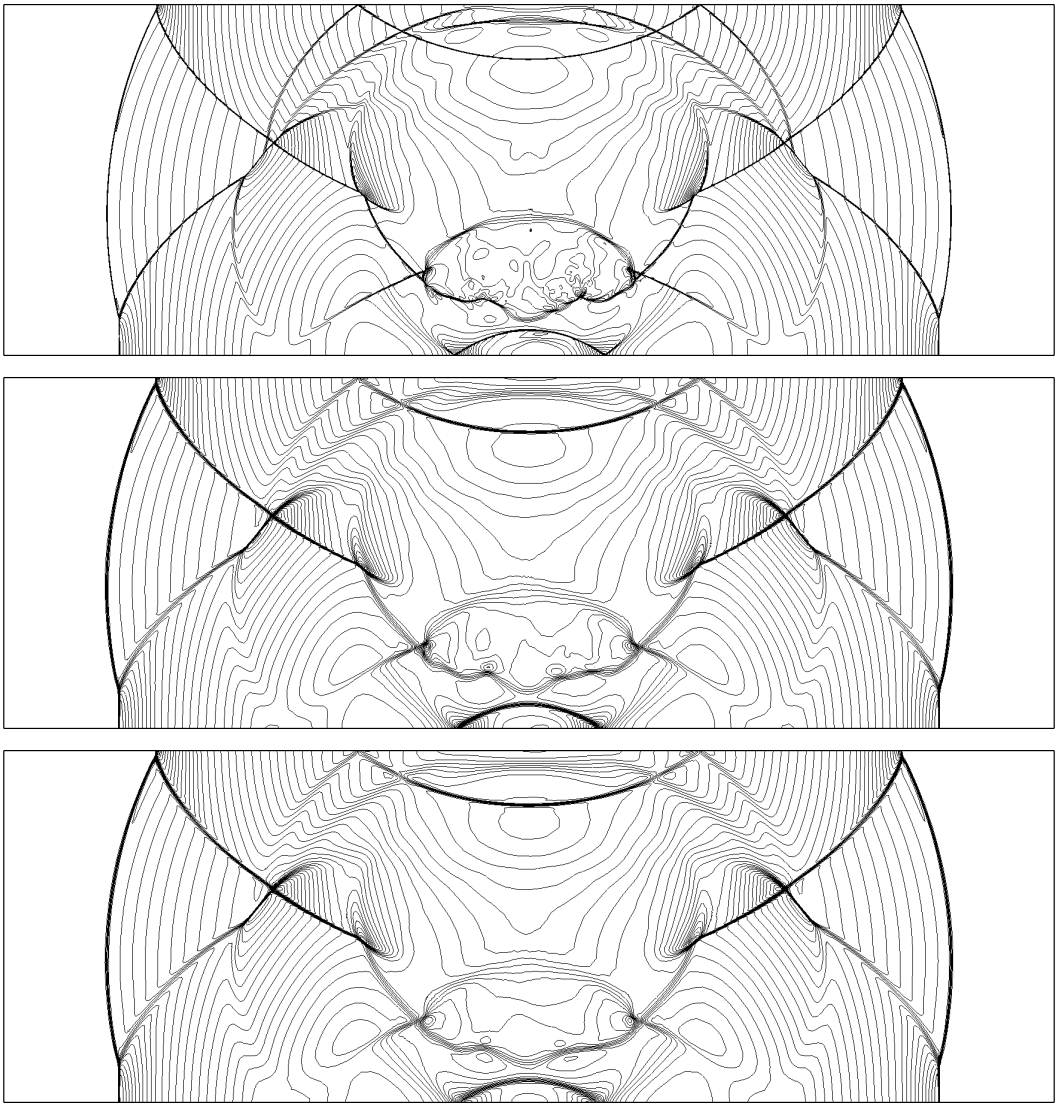

Figure 19. *Pressure isolines from* 0.715 *to* 1.695 *with an increment of* 0.0245 *at time* $t = 0.7$. *Top, the reference solution,* i.e., $\varepsilon = 0.015$ *and the* $\mathbf{P}^1$*-conservative interpolation. Middle, adapted solution obtained for* $\varepsilon = 0.03$ *with the* $\mathbf{P}^1$*-conservative interpolation. Bottom, adapted solution obtained for* $\varepsilon = 0.03$ *with the* $\mathbf{P}^1$*-interpolation.*

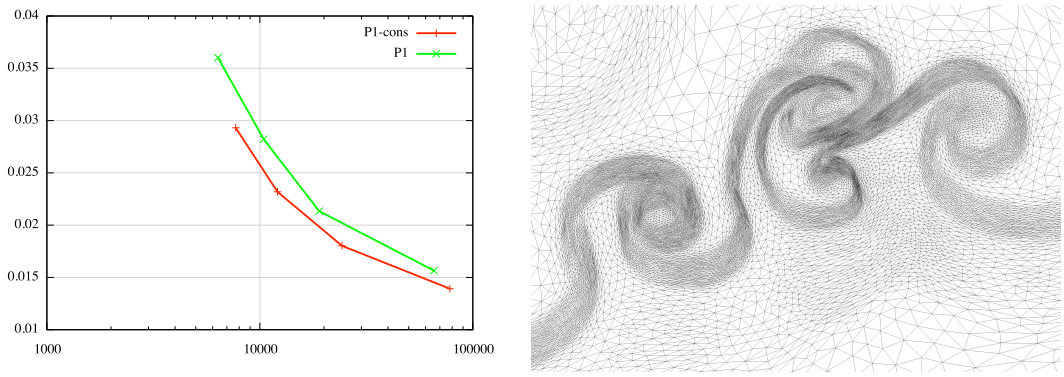

Figure 20. *Left,* $\mathbf{L}^2$*-norm error of the density with respect to the reference solution for adaptive simulations for* ε *from* 0.05 *to* 0.02*. In red, adaptive simulations with the* $\mathbf{P}^1$*-conservative interpolation and, in green, with the* $\mathbf{P}^1$ *interpolation. Right, a close up view of the adapted mesh of Figure 18 (top) in a region where instabilities occur.*

8.3. A blast in a town

The second example is a blast in a geometry representing a city plaza that has been proposed in [6]. This simulation is a multi-dimensional generalization of the Sod Riemann problem [21] in a geometry. The main feature of this problem is related to the random character of the shock wave propagation due to a large number of waves reflexions on the geometry and of shock waves interactions.

The computational domain size is 150×90 m^2. Initially, the gas representing the ambient air is at rest with a density $\rho_{out} = 0.125$ and $p_{out} = 0.1$. To simulate the blast, a high pressure and density region is introduced in a quarter-circle centered at $(6.5, 0)$ with a radius 0.25. In this region, the relevant parameters are $\rho_{in} = 1$, $p_{in} = 1$ and $u_{in} = v_{in} = 0$. For both regions, we have $\gamma = 1.4$. The solution is computed until physical time $t = 0.2$ seconds.

For this problem too, the $\mathbf{L}^2$-norm of the density interpolation error is controlled. The global simulation is split into 30 time periods of time length $\Delta t = 0.0066$ for which a transient fixed-point problem is solved throughout 5 internal loop iterations. This simulation has been performed for five error thresholds, from $\varepsilon = 0.007$ to $\varepsilon = 0.003$. The number of metric intersections in time varies between 10 and 20 depending on the error threshold. Two series

Error threshold $\mathcal{E}$	$\mathbf{P}^1$	$\mathbf{P}^1$-conservative
0.007	15,532	17,795
0.006	21,343	23,767
0.005	43,001	51,104
0.004	73,219	84,049
0.003	152,503	170,020

Table V. *Number of vertices of the final adapted meshes obtained with the unsteady mesh adaptation algorithm for the* $\mathbf{P}^1$ *and the* $\mathbf{P}^1$*-conservative interpolation for several error thresholds.*

of adaptations have been performed: one with the classic $\mathbf{P}^1$ interpolation and the other one with the $\mathbf{P}^1$-conservative interpolation scheme. The statistics of the final ($t = 0.2$) adapted meshes are given in Table V.

Like in the previous case, the impact of the conservative interpolation on the solution accuracy is studied. We designate by *reference solution* the adaptive solution at time $t = 0.2$ seconds computed with an error threshold $\varepsilon = 0.003$ and the $\mathbf{P}^1$-conservative interpolation. Figure 21 shows a schlieren picture representing the reference solution density. The associated adapted mesh is given in Figure 23 (top). Close up views of this mesh are provided in Figure 22. These views illustrate the anisotropy of the mesh.

The gain in accuracy of the conservative interpolation with respect to the classical one is illustrated similarly to the previous example. Figure 25 (left) demonstrates the gain in accuracy by depicting density errors in $\mathbf{L}^2$-norm with respect to the reference solution for simulations with an error threshold between $\varepsilon = 0.007$ and $\varepsilon = 0.004$. We notice that the error has been lowered with the $\mathbf{P}^1$-conservative interpolation. The reduced diffusion is also pointed out in the same figure (right) where density profiles are plotted along an arbitrary line for solutions with an error $\varepsilon = 0.006$.

Figure 24 represents the density isolines for the reference solution (top) and the adaptive solutions with an error of $\varepsilon = 0.006$ for the $\mathbf{P}^1$-conservative interpolation (middle) and the $\mathbf{P}^1$-interpolation (bottom). We notice that the solution with the $\mathbf{P}^1$-conservative interpolation is a little more accurate. Once again, this gain of accuracy results in a larger number of vertices obtained for the series of adapted meshes with the $\mathbf{P}^1$-conservative interpolation.

The density relative mass variation during the simulation for all the cases is presented in

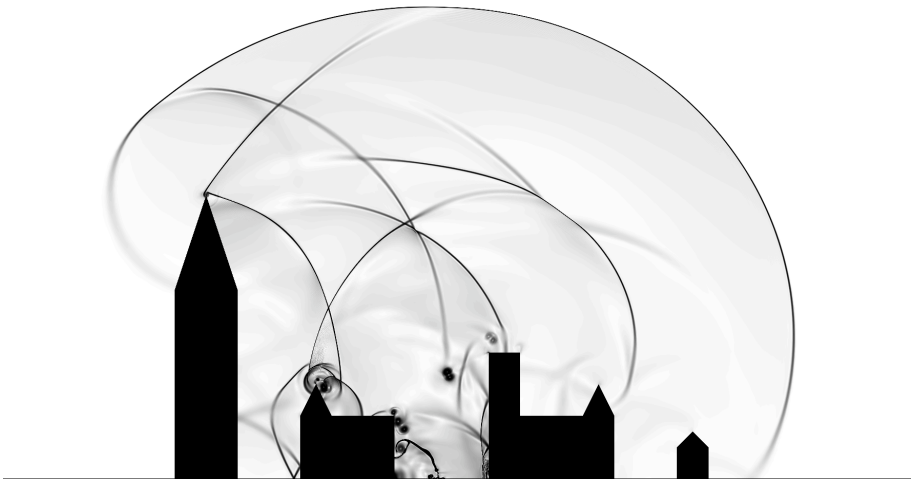

Figure 21. *A schlieren type picture representing the final density at* $t = 0.2$ *sec on the final adapted mesh containing* $170,020$ *vertices obtained from an error level of* 0.003 *and the* $\mathbf{P}^1$*-conservative interpolation.*

Figure 26. For the $\mathbf{P}^1$ interpolation, the mass has varied of almost 0.01% at the end of the simulation. We remark that the mass variation increases during the simulation and that it decreases with the adapted mesh accuracy. We also notice that the mass variation is small as compared to the one obtained for the analytical examples. This is due to the use of adapted meshes.

Concerning the $\mathbf{P}^1$-conservative interpolation, the mass variation is very low: 10^{-7} at the end of the simulation. It increases during the simulation. Notice that this growth is to some extent due to small variations of the domain area while remeshing the domain during the computation.

As regards the cpu time, both interpolation schemes have been compared on several couples of meshes on a Intel Core 2 at 2.8 GHz. All the cases are summarized in Table VI. It follows that the $\mathbf{P}^1$-conservative interpolation is, in this case too, approximately 5 times slower than the $\mathbf{P}^1$ interpolation. Nevertheless, the cost of the interpolation stage (a few seconds) is still negligible comparing to the solver cpu time.

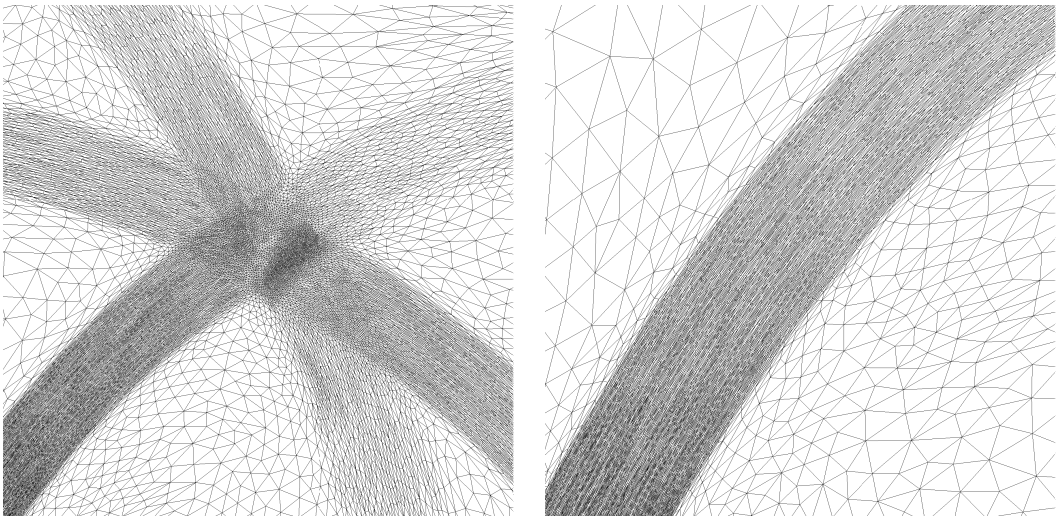

Figure 22. *Close up views of the final adapted mesh containing* $170,020$. *These views illustrate that all the regions where shock waves progress during a given period are anisotropically refined.*

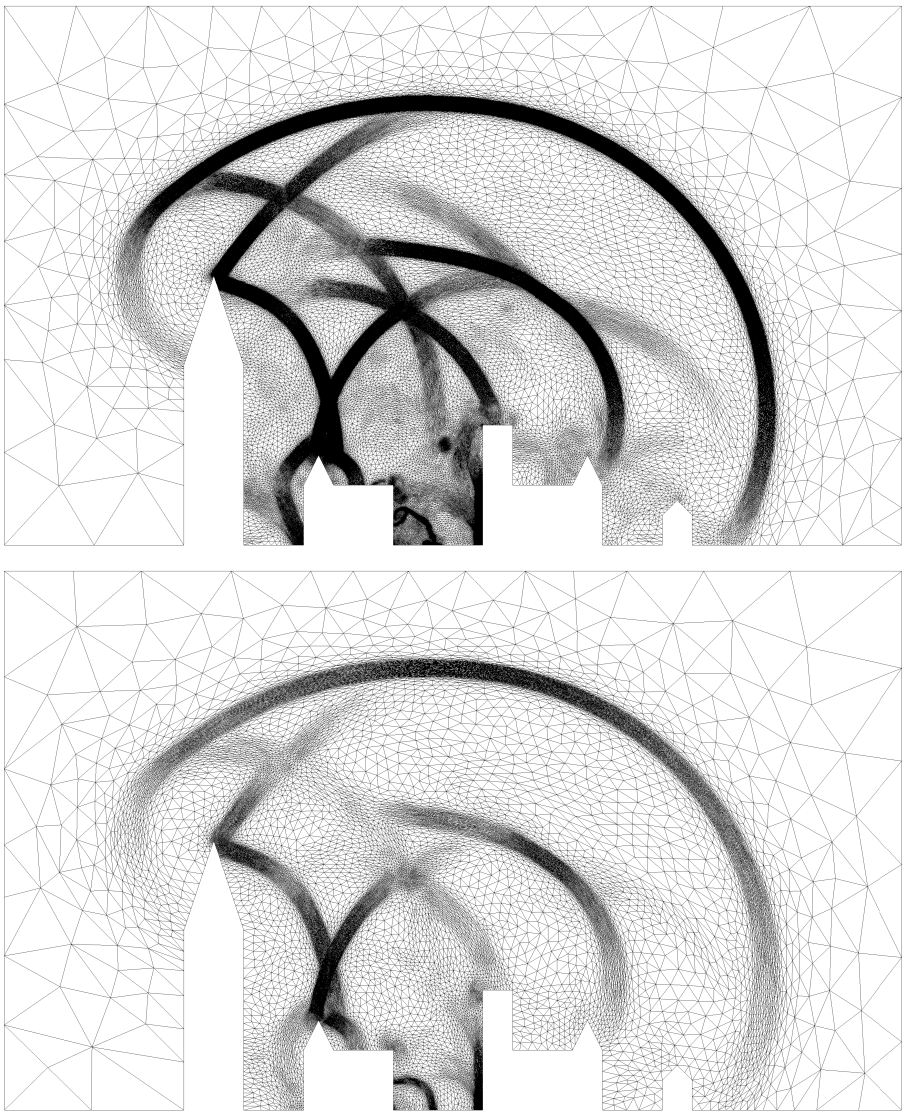

Figure 23. *Final adapted meshes using the* $\mathbf{P}^1$*-conservative interpolation at time* $t = 0.2$ *seconds for errors equal to* 0.003 *(top) and* 0.006 *(bottom). The upper mesh contains* $170,020$ *vertices and the lower one* $23,767$.

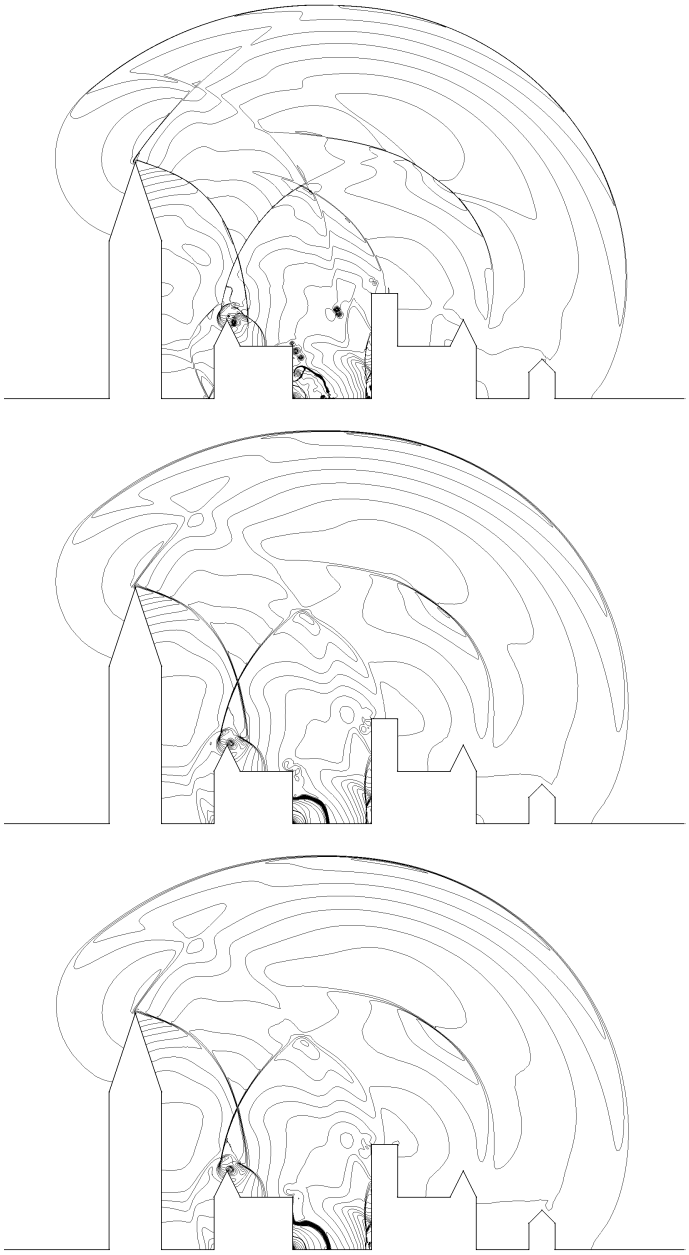

Figure 24. *Density isolines from* 0.085 *to* 0.19 *with an increment of* $2.65\,e^{-3}$ *at time* $t = 0.2$ *seconds. Top, the reference solution,* i.e., $\varepsilon = 0.003$ *and the* $\mathbf{P}^1$*-conservative interpolation. Middle, adapted solution obtained for* $\varepsilon = 0.006$ *with the* $\mathbf{P}^1$*-conservative interpolation. Bottom, adapted solution obtained for* $\varepsilon = 0.006$ *with the* $\mathbf{P}^1$*-interpolation.*

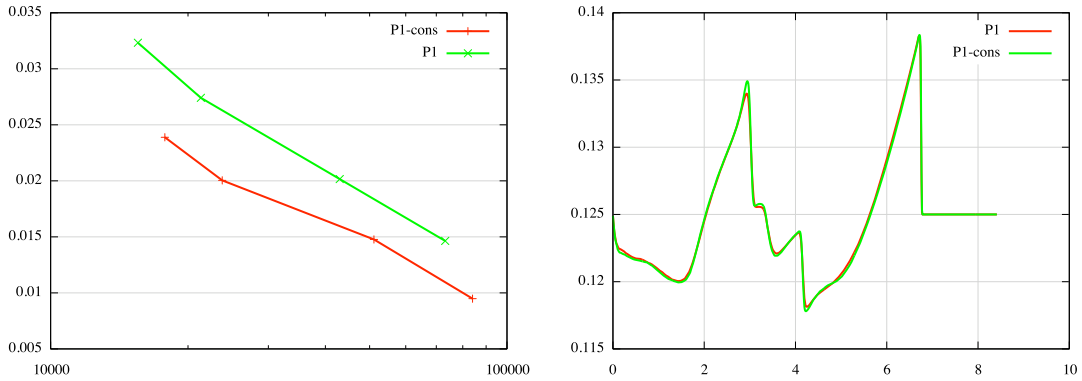

Figure 25. *Left,* $\mathbf{L}^2$*-norm error of the density with respect to the reference solution for adaptive simulations for* ε *from* 0.007 *to* 0.004. *In red, adaptive simulations with the* $\mathbf{P}^1$*-conservative interpolation and, in green, with the* $\mathbf{P}^1$ *interpolation. Right, comparison of the density profile at time* $t = 0.2$ *seconds along the line of equation* $y = 3.07692x - 17.4615$ *for the solutions obtained with an error threshold* $\varepsilon = 0.006$.

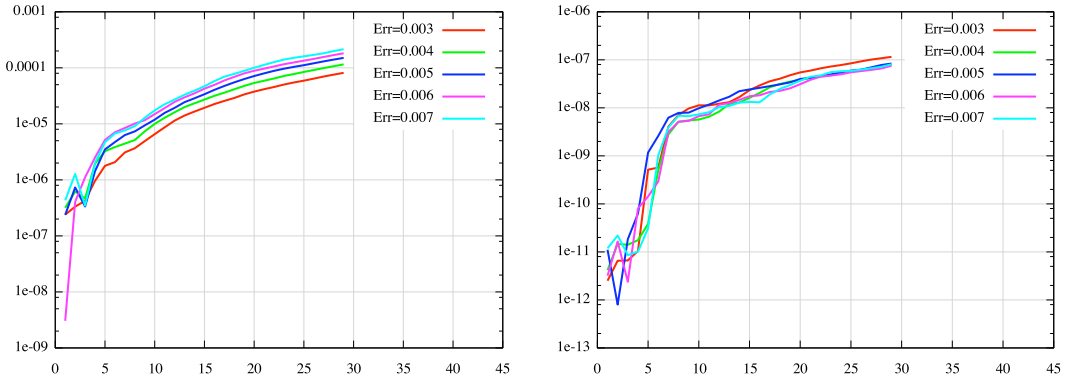

Figure 26. *Density relative mass variation during the simulation for all cases. The abscissae represent the iterations of the mesh adaptation algorithm. Left, mass variation for the* $\mathbf{P}^1$ *interpolation. Right, mass variation for the* $\mathbf{P}^1$*-conservative interpolation.*

# vertices $\mathcal{H}_i^{back}$	# vertices $\mathcal{H}_i^{new}$	$\mathbf{P}^1$-cons cpu time in sec.	$\mathbf{P}^1$ cpu time in sec.	ratio
$17,180$	$17,795$	0.468	0.114	4.1
$22,384$	$23,767$	0.630	0.126	5.0
$47,232$	$51,104$	1.295	0.243	5.3
$75,104$	$84,049$	2.103	0.328	6.4
$153,502$	$170,020$	4.463	1.225	3.6

Table VI. *Cpu time comparison between the $\mathbf{P}^1$-conservative interpolation and the $\mathbf{P}^1$ interpolation on several couples of meshes on a Intel Core 2 at 2.8 GHz. The cpu time is expressed in seconds.*

8.4. Some remarks about the numerical results

In all the presented examples, the benefits due to the conservative interpolation have been illustrated. Analytical examples have clearly demonstrated the advantages of the conservative interpolation. However, for the two blast simulations, the gain in accuracy seems to be small as regards the cost. But, it is important to note that the number of mesh adaptations, and thus of interpolations, that have been performed in both simulations is small (less than 30). More complex problem or problem with a long time will require to perform more adaptations and therefore the difference between the P^1-interpolation and the P^1-conservative interpolation should be more significative. Moreover, we have to remember that the conservation property can be mandatory for some applications.

9. CONCLUSION

In this work, we have proposed a $\mathbf{P}^1$-conservative interpolation operator that satisfies the maximum principle. This operator is based on local mesh intersections and local operations that make it efficient in terms of cpu time and memory requirement. Numerical examples show that it is approximately 5 times slower than the classical linear interpolation. The properties of this new operator have been verified numerically on analytical examples and adaptive simulations. These examples also point out a better accuracy of the conservative interpolation compared to the classical one.

The proposed conservative interpolation scheme extends easily to P^k-representation of the solution if solutions are defined at the elements. However, the extension to P^k-representation of the solution defined at vertices requires more work and is under consideration at the time.

Finally, the proposed algorithm is extendable to three dimensions even if the mesh intersection is more difficult to handle. This version is under development.

Appendix

We give here the proof of Propositions 5.3 and 5.4.

Proposition 5.3. Let $S \in \{\text{I,II}\}$. The reconstruction u_K^S satisfies the maximum principle, is linear preserving and is conservative. Moreover, we have:

$$u_K(P_0) \leq u_K(P_1) \leq u_K(P_2) \quad \Longrightarrow \quad u_K^S(P_0) \leq u_K^S(P_1) \leq u_K^S(P_2)$$

and

$$u_{\min} \leq u_K(P_i) \leq u_{\max} \text{ for } i = 0,1,2 \quad \Longrightarrow \quad u_K^S(P_i) = u_K(P_i) \text{ for } i = 0,1,2.$$

Proof of Proposition 5.3.

- We first consider the case where $S = I$.

From the definition of $u_K(G)$, we have $u_{\min} \leq u_K(G) \leq u_{\max}$ where G is the barycenter of the triangle. Now, it suffices to prove that $u_{\min} \leq u_K^I(P_i) \leq u_{\max}$, for $i = 0,1,2$. We have at first $0 \leq \varphi_K(P_i)$, since $u_{\min} \leq u_K(G) \leq u_{\max}$. If $u_K(P_i) - u_K(G) > 0$, we then get

$$0 \leq u_K^I(P_i) - u_K(G) \leq \varphi_K(P_i)|u_K(P_i) - u_K(G)| \leq u_{\max} - u_K(G).$$

On the other hand, if $u_K(P_i) - u_K(G) < 0$, we have

$$0 \geq u_K^I(P_i) - u_K(G) \geq -\varphi_K(P_i)|u_K(P_i) - u_K(G)| \geq u_{\min} - u_K(G).$$

Finally, if $u_K(P_i) = u_K(G)$, we have $u_K^I(P_i) = u_K(G)$.

Consequently, for all the cases, we have $u_{\min} \leq u_K^I(P_i) \leq u_{\max}$ and therefore $u_{\min} \leq u^I(P_i) \leq u_{\max}$. This proves the maximum principle.

Now, we suppose that u is affine. We have to prove that $u_K^I(P_i) = u(P_i)$, for $i = 0,1,2$. Since u is affine, we have $u_{\min} \leq u(P) = u_K(P) \leq u_{\max}$ and thus $\varphi_K(P_i) = 1$ for $i = 0,1,2$. We then have $u_K^I(P_i) = u_K(G) + \nabla u_K \cdot \overrightarrow{GP_i} = u_K(P_i) = u(P_i)$.

Since $\int_K \overrightarrow{GP} \mathrm{d}P = 0$, we obtain that $\int_K u_K^I = \int_K u$. We then have

$$\begin{aligned}
\sum_K \int_K u &= \sum_K |K| \frac{u_K^I(P_0) + u_K^I(P_1) + u_K^I(P_2)}{3} = \sum_P \sum_{K_i \ni P} |K_i| u_{K_i}^I(P) \\
&= \sum_P \Big(\sum_{K_i \ni P} |K_i| \Big) u^I(P) = \sum_K |K| \frac{u^I(P_0) + u^I(P_1) + u^I(P_2)}{3} \\
&= \sum_K \int_K u^I,
\end{aligned}$$

and thus the reconstruction is conservative.

For the last two properties, as we have

$$u_K(P_0) \leq u_K(P_1) \leq u_K(P_2) \Leftrightarrow \nabla u_K \cdot \overrightarrow{GP_0} \leq \nabla u_K \cdot \overrightarrow{GP_1} \leq \nabla u_K \cdot \overrightarrow{GP_2},$$

and since $\varphi_K(P_i) \geq 0$, we have the first result. If $u_{\min} \leq u_K(P_i) \leq u_{\max}$, we get $\varphi_K(P_i) = 1$ and thus the last property is also true.

- Now, we consider the case where $S = II$.

We first prove that

$$u_K^M(P_0) \leq u_K^M(P_1) \leq u_K^M(P_2) \leq u_{\max}.$$

If $u_K(P_2) \leq u_{\max}$, we have $u_K^M(P_i) = u_K(P_i),\ i = 0, 1, 2$, and thus have the inequality.
Otherwise, we have $u_K^M(P_2) = u_{\max}$ and $u_K^M(P_1) = \min(u_K(P_1) + \frac{u_K(P_2) - u_{\max}}{2}, u_{\max})$.
Therefore, if $2u_K(P_1) + u_K(P_2) \leq 3u_{max}$, we have $u_K^M(P_i) = u_K(P_i) + \frac{u_K(P_2) - u_{\max}}{2}$, for $i = 0, 1$ and we get the result.
Finally, if $2u_K(P_1) + u_K(P_2) > 3u_{max}$, we have $u_K^M(P_1) = u_{\max}$ and $u_K^M(P_0) = 3u_K(G) - 2u_{\max} \leq u_{\max}$ and we obtain also the desired inequality.

We can similarly prove that

$$u_{\min} \leq u_K^{II}(P_0) \leq u_K^{II}(P_1) \leq u_K^{II}(P_2).$$

It remains to prove that $u_K^{II}(P_2) \leq u_{\max}$. We already know that $u_K^M(P_2) \leq u_{\max}$. If $u_K^M(P_0) \geq u_{\min}$, we have it. Otherwise, if $2u_K^M(P_1) + u_K^M(P_0) \geq 3u_{\min}$, we have

$$u_K^{II}(P_2) = u_K^M(P_2) + \frac{u_K^M(P_0) - u_{\min}}{2} \leq u_K^M(P_2) \leq u_{\max}\,.$$

And in the other case, we have $u_K^{II}(P_2) = 3u_K(G) - 2u_{\min} \leq u_{\max}$. It finishes to prove the maximum principle.

If u is affine, we have $u_K(P_0) = u_{\min}$ and $u_K(P_2) = u_{\max}$. We then get $u_K^M(P_i) = u_K(P_i)$ and $u_K^{II}(P_i) = u_K(P_i)$, for $i = 0, 1, 2$ that provide the $\mathbf{P}^1$-exactness.

Since $u_K^{II}(P_2) = 3u_K(G) - u_K^{II}(P_0) - u_K^{II}(P_1)$, we obtain the conservation: $\int_K u_K^{II} = \int_K u$.

The last properties can be checked from the formulae.

□

Proposition 5.4. Suppose that $u_K(P_0) \leq u_K(P_1) \leq u_K(P_2)$ and that $u_{\max} < u_K(P_2)$. Then, we have

$$\sum_{i=0}^{2} |u_K(P_i) - u_K^M(P_i)|^2 \leq \sum_{i=0}^{2} |u_K(P_i) - v_K(P_i)|^2$$

for all the linear reconstructions v_K satisfying $v_K(P_2) = u_{\max}$, $v_K(P_i) \leq u_{\max}$ for $i = 0, 1$ and $\int_K v_K = \int_K u_K$.

Proof of Proposition 5.4. We set $\varepsilon_i = v_K(P_i) - u_K(P_i)$, for $i = 0, 1, 2$. Since we have $\int_K v_K = \int_K u$, we deduce that $\varepsilon_0 + \varepsilon_1 + \varepsilon_2 = 0$. On the other hand $\varepsilon_2 = u_{\max} - u_K(P_2)$, and $\varepsilon_i \leq u_{\max} - u_K(P_i)$ for $i = 0, 1$. We thus have to minimize $\varepsilon_0^2 + \varepsilon_1^2$, for all the numbers

$\varepsilon_0, \varepsilon_1$ satisfying $\varepsilon_i \leq u_{\max} - u_K(P_i)$ for $i = 0, 1$, and $\varepsilon_0 + \varepsilon_1 = u_K(P_2) - u_{\max}$.
By setting $\varepsilon_0 = \frac{1}{2}(u_K(P_2) - u_{\max}) + \eta$, we have to minimize

$$(\frac{1}{2}(u_K(P_2) - u_{\max}) + \eta)^2 + (\frac{1}{2}(u_K(P_2) - u_{\max}) - \eta)^2,$$

for η belonging to the interval

$$I = [u_K(P_1) - u_{\max} + \frac{1}{2}(u_K(P_2) - u_{\max}), u_{\max} - u_K(P_0) - \frac{1}{2}(u_K(P_2) - u_{\max})].$$

The minimum is given for $\eta = 0$, if $0 \in I$ or for one bound of I otherwise. Notice that $0 \in I$ if and only if $2u_K(P_1) + u_K(P_2) \leq 3u_{\max}$. Thus, if $2u_K(P_1) + u_K(P_2) \leq 3u_{\max}$, we check that for $\eta = 0$, $v_K(P_1) = u_K^M(P_1)$. On the other hand, if $2u_K(P_1) + u_K(P_2) > 3u_{\max}$, we can check that

$$(u_{\max} - u_K(P_1))^2 + (u_K(P_2) - 2u_{\max} + u_K(P_1))^2 \\ \leq (u_{\max} - u_K(P_0))^2 + (u_K(P_2) - 2u_{\max} + u_K(P_0))^2,$$

so that the minimum is given for the left bound and we get for this value of η that $v_K(P_1) = u_K^M(P_1)$. Thus, in all cases, we have $v_K(P_1) = u_K^M(P_1)$, and this concludes the proof.

□

REFERENCES

1. Dukowicz J, Kodis J. Accurate conservative remapping (rezoning) for arbitrary Lagrangian - Eulerian computations. *SIAM J. Sci. Statist. Comput.* 1987; **8**(3):305–321.
2. Garimella R, Kucharik M, Shashkov M. An efficient linearity and bound preserving conservative interpolation (remapping) on polyhedral meshes. *Comput. & Fluids* 2007; **36**(2):224–237.
3. Margolin L, Shashkov M. Second-order sign-preserving conservative interpolation (remapping) on general grids. *J. Comp. Phys.* 2003; **184**(1):266–298.
4. Cheng J, Shu CW. A high order accurate conservative remapping method on staggered meshes. *Appl. Numer. Math.* 2008; **58**(7):1042–1060.
5. Grandy J. Conservative remapping and regions overlays by intersecting polyhedra. *J. Comp. Phys.* 1999; **148**(2):433–466.
6. Alauzet F, Frey P, George PL, Mohammadi B. 3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations. *J. Comp. Phys.* 2007; **222**:592–623.
7. George PL, Borouchaki H. *Delaunay triangulation and meshing. Application to finite elements.* Herms: Paris, 1998.
8. Frey P, George PL. *Mesh generation. Application to finite elements.* 2nd edn., ISTE Ltd and John Wiley & Sons, 2008.
9. Löhner R. *Applied CFD techniques. An introduction based on finite element methods.* John Wiley & Sons, Ltd: New York, 2001.
10. Ern A, Guermond JL. *Theory and Practice of Finite Elements*, *Applied Mathematical Series*, vol. 159. Springer: New York, 2004.
11. Frey P, Alauzet F. Anisotropic mesh adaptation for CFD computations. *Comput. Methods Appl. Mech. Engrg.* 2005; **194**(48-49):5068–5082.
12. Loseille A, Dervieux A, Frey P, Alauzet F. Achievement of global second-order mesh convergence for discontinuous flows with adapted unstructured meshes. *AIAA paper* 2007; **2007-4186**.
13. Leservoisier D, George PL, Dervieux A. Métrique continue et optimisation de maillage. *RR-4172*, INRIA Apr 2001. (in French).
14. Batten P, Clarke N, Lambert C, Causon D. On the choice of wavespeeds for the HLLC riemann solver. *SIAM J. Sci. Comput.* 1997; **18**(6):1553–1570.

15. Stoufflet B, Periaux J, Fezoui L, Dervieux A. Numerical simulation of 3-D hypersonic Euler flows around space vehicles using adapted finite element. *AIAA Paper* 1987; **87-0560**.
16. Cournède PH, Koobus B, Dervieux A. Positivity statements for a Mixed-Element-Volume scheme on fixed and moving grids. *European Journal of Computational Mechanics* 2006; **15**(7-8):767–798.
17. Debiez C, Dervieux A. Mixed-Element-Volume MUSCL methods with weak viscosity for steady and unsteady flow calculations. *Computer & Fluids* 2000; **29**:89–118.
18. Spiteri R, Ruuth S. A new class of optimal high-order strong-stability-preserving time discretization methods. *SIAM J. Numer. Anal.* 2002; **40**(2):469–491.
19. Alauzet F. Adaptive sonic boom sensitivity analysis. *Proc. of ECCOMAS CFD*, 2006.
20. Langseth J, LeVeque RJ. A wave propagation method for three-dimensional hyperbolic conservation laws. *J. Comp. Phys.* 2000; **165**:126–166.
21. Sod G. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *J. Comput. Phys.* 1978; **27**:1–31.