**cerca**
Centre de Recherche en Calcul Appliqué

ÉCOLE POLYTECHNIQUE MONTRÉAL

# How to Subdivide Pyramids, Prisms and Hexahedra into Tetrahedra

Julien Dompierre  Paul Labbé  Marie-Gabrielle Vallet  Ricardo Camarero

Centre de recherche en calcul appliqué (CERCA)
5160, boul. Décarie, suite 400, Montréal, (Québec), H3X 2H9, Canada
[julien|paul|vallet|ricardo]@cerca.umontreal.ca

Rapport CERCA R99–78
24 août 1999

Présenté au
8th International Meshing Roundtable,
Lake Tahoe, Californie,
10–13 octobre 1999.

# HOW TO SUBDIVIDE PYRAMIDS, PRISMS AND HEXAHEDRA INTO TETRAHEDRA

Julien Dompierre, Paul Labbé, Marie-Gabrielle Vallet, Ricardo Camarero

*Centre de recherche en calcul appliqué (CERCA)*
*5160, boul. Décarie, bureau 400, Montréal, QC, H3X 2H9, Canada.*
*[julien|paul|vallet|ricardo]@cerca.umontreal.ca*

## ABSTRACT

This paper discusses the problem of subdividing meshes containing tetrahedra, pyramids, prisms or hexahedra into a consistent set of tetrahedra. This problem occurs in computer graphics where meshes with pyramids, prisms or hexahedra must be subdivided into tetrahedra to use efficient algorithms for volume rendering, iso-contouring and particle advection. Another application is for the use of some tetrahedral finite element solvers on a non tetrahedral mesh, or even an hybrid mesh. Arbitrary splitting of quadrilateral faces into two triangles has two major drawbacks: it can lead to discontinuities across element faces, resulting in non conformal meshes, and the subdivision of some elements into tetrahedra may need to introduce new vertices. The algorithms presented in this paper split each quadrilateral face of pyramids, prisms and hexahedra in a consistent way that preserves the conformity of the mesh. Elements are subdivided into tetrahedra without introducing new vertices. The algorithms are fast, generic, robust, and local, i.e. they do not need any neighboring information.

**Keywords: pyramid, prism, hexahedron, tetrahedron, tetrahedralization, subdivision, conformity.**

## 1. INTRODUCTION

This paper discusses the problem of subdividing meshes containing tetrahedra, pyramids, prisms or hexahedra into a consistent set of tetrahedra, while preserving the overall mesh conformity. A mesh is conformal if it assumes the continuity of piecewise linear solutions at interfaces between elements. It is assumed that the initial mesh of tetrahedra, pyramids, prisms and hexahedra is conformal. Its subdivision into a tetrahedral mesh stays conformal if all internal quadrilateral faces are split the same way for both elements sharing it.

One of the main applications is in finite element method, when an unstructured tetrahedral solver is used to solve a problem on an hybrid mesh. Non tetrahedral elements, for example a skin of prismatic elements, must be subdivided into tetrahedra [4, 13, 17, 18]. Another targeted application is for two-dimensional space-time finite element schemes, when a two-dimensional unstructured triangular mesh is extruded in temporal direction in a space-time slab composed of prismatic elements that must be subdivided into tetrahedra to allow two-dimensional space-time unstructured mesh adaptation [6, 9, 15]. This problem is also a major concern in computer graphic when non tetrahedral mesh must be subdivided into tetrahedra to use efficient algorithms for volume rendering, iso-contouring and particle advection that exist for mesh topologies including only tetrahedra [1, 14].

The problem of subdividing non tetrahedral elements can be decomposed in two steps [1]. The first step is to split all the quadrilateral faces into triangles in a consistent manner. The second step is to tetrahedralize each element according to quadrilateral faces splitting.

Some of the algorithms found in the literature for the first step [1, 4, 13, 14, 17, 18] are iterative, all of them need some neighboring information and some memory to store temporary data (this is a minor problem). Also, they are tested on layer of elements, not on generic unstructured mixed tetrahedral, pyramidal, prismatic and hexahedral meshes. The proposed solution is an algorithm which is direct (no iteration), that does not need to construct neighboring information, does not need additional memory and is fully generic.

When addressing the second step, the same authors have to deal with elements that are not tetrahedralizable without adding a new vertex. Adding new vertices during the elements subdivision step is a little more complex to implement, but it also leads to a larger number of elements. It is shown in

this paper that these untetrahedralizable configurations never arise using the presented algorithms.

Even if there is still research being done and recent publications about different algorithms that attempt to solve parts of this problem, the solution of this problem already exists since a few years but is not widely known! This solution is direct, it does not need to construct neighboring information, it does not need additional memory, it is fully generic and it avoids the introduction of new vertices. In his *"Thèse d'habilitation à diriger des recherches"*, Hecht [11] wrote a paragraph about this problem and the solution was given in one sentence which can be translated as *"...it is sufficient, for example, to split all quadrilateral faces by a diagonal that goes through the vertex with the greatest global number."* This algorithm was coded in 1986 in the module `DTRI3D` of the subroutines library `Modulef`. This software is now publicly available at `http://www-rocq.inria.fr/modulef/` and there is some documentation on the module `DTRI3D` in the manuals.

This paper gives this solution and *proves by construction* that there is always a tetrahedralization without introducing new vertices. All the subtleties of the algorithms are fully explained. Section 2 gives the method outline based on quadrilateral face splitting according to vertex numbering. Sections 3,4 and 5 detail the algorithm for a pyramid, a prism and a hexahedron, respectively. As the proof of the existence of a tetrahedralization without introducing new vertices is done by construction, this paper is more or less a technical note. Some comments on efficient computer implementation are given in section 6. Limitations of the method are given in section 7. Section 8 provides two examples to illustrate results obtained with the method.

## 2. METHOD OUTLINE

Let a mesh be composed of tetrahedra, pyramids, prisms and hexahedra. The mesh is supposed to be *conformal*, i.e., for any pair of two different elements, they can share 1) nothing, 2) a vertex, 3) an entire edge, 4) an entire (triangular or quadrilateral) face. For example, a mesh is not conformal when a quadrilateral face of an hexahedron is connected to two triangular faces of tetrahedra or prisms. If the mesh is not conformal, the algorithms of this paper will probably fail. The initial mesh is supposed to be conformal, and the purpose is to get a conformal mesh only composed of tetrahedra.

The method is based on the use of vertex identifiers. In computer graphics, in finite element or volume method, in mesh generation, each vertex of a mesh has an identifier. Usually, this identifier is an integer value which is the global number of the vertex. It can also be a pointer which is the memory address where the vertex is stored. Whatever the identifier used, two characteristics are mandatory for the proposed method: first, each vertex must have an *unique and constant identifier*; second, the set of vertex identifiers has to be a *totally ordered set*, i.e. a relation *is lower than* is defined between the identifiers. For any set of vertices, these two characteristics permit to define the *smallest vertex* of the set which is the vertex with the smallest identifier. In the following, the notation $V_i < V_j$ means that the identifier of vertex $V_i$ is smaller than the identifier of the vertex $V_j$. The notation $(V_i,\ V_j) < (V_k,\ V_\ell)$ means that the smallest identifier of the vertices $V_i$ and $V_j$ is smaller than the smallest identifier of the vertices $V_k$ and $V_\ell$.

Each triangular face of the mesh remains untouched. Each quadrilateral face of the mesh must be subdivided into two triangular faces. There are two different ways to subdivide a quadrilateral face. The method is based on the following rule:

> *A quadrilateral face is subdivided into two triangular faces by the diagonal issuing from the smallest vertex of the face.*

This rule can be applied locally, element by element, and does not need any information on the neighboring elements. However, it ensures that two adjacent elements will split their common quadrilateral face the same way, so long as the vertices have a unique and constant identifier. Let $V_1$, $V_2$, $V_3$ and $V_4$ be the vertices counterclockwise of a quadrilateral face. The face will be cut with the diagonal $V_1$ to $V_3$ if $(V_1,\ V_3) < (V_2,\ V_4)$ or will be cut with the diagonal $V_2$ to $V_4$ if $(V_2,\ V_4) < (V_1,\ V_3)$. This test is cheap to compute, there is no roundoff error and there is always a decision, i.e. $(V_1,\ V_3)$ cannot be equal to $(V_2,\ V_4)$. If the way to split a quadrilateral face is to use the shortest diagonal [1], then it is more expensive, subject to roundoff errors and ambiguous as what to do if the two diagonals have the same length. The subdivision rule introduced in this paragraph is the kernel of the paper, nothing being perfect, see section 7 for some limitations of the method.

Now that quadrilateral faces are split into triangular faces in a way that ensures conformity across the mesh, elements must be tetrahedralized according to the triangular faces. The simplest method should be to insert a vertex at each element center and to build tetrahedra by connecting it to all triangular faces. But the resulting tetrahedra would be numerous (a pyramid is split into six tetrahedra, a prism into eight, and a hexahedron into twelve), would not be particularly well shaped and there would be new vertices to manage.

The optimal solution should be to subdivide the elements with the minimum number of tetrahedra, and to avoid adding new vertices. The problem is that there exists some configurations of faces splitting such that they are not tetrahedralizable, except by adding a vertex. *An advantage of the present quadrilateral face subdivision rule is that those problematic configurations never arise*. This is shown in following sections. It is possible to tetrahedralize all elements without adding any new vertex.

A note about local numbering of the vertices of the elements. If the local numbering used in this paper is not the same to the one used by the reader, it is easy to understand the outline of the subdivision algorithms and to rewrite them according to the local numbering used. Also, if the local numbering of the vertices differs, a layer of indirection can be added.

## 3. PYRAMID

It is supposed that the local numbering $V_1$, $V_2$, $V_3$, $V_4$ and $V_5$ of the vertices of the pyramid is given by the Fig. 1.

The problem of subdividing a pyramid into tetrahedra is simple because there are only two possible configurations to

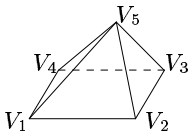


**Figure 1: Local numbering of the vertices $V_1$ to $V_5$ of a pyramid.**

analyze with obvious subdivision into tetrahedra. For the quadrilateral face of the pyramid, either $(V_1, V_3) < (V_2, V_4)$ or $(V_2, V_4) < (V_1, V_3)$. These two configurations can be subdivided into two tetrahedra that are illustrated in Fig. 2 and given in Table 1.

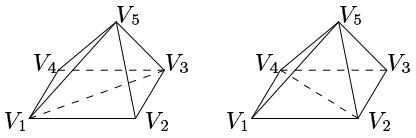


**Figure 2: The two manners to split the quadrilateral face of the pyramid and to subdivide the pyramid into two tetrahedra.**

**Table 1: The two tetrahedra that subdivide the pyramid in function of how the quadrilateral face is split.**

| If | Tetrahedra |
|---|---|
| $(V_1, V_3) < (V_2, V_4)$ | $\triangle V_1V_2V_3V_5$<br>$\triangle V_1V_3V_4V_5$ |
| $(V_2, V_4) < (V_1, V_3)$ | $\triangle V_2V_3V_4V_5$<br>$\triangle V_2V_4V_1V_5$ |

## 4. PRISM

It is supposed that the local numbering $V_1$, $V_2$, $V_3$, $V_4$, $V_5$ and $V_6$ of the vertices of the prism is given by the Fig. 3.

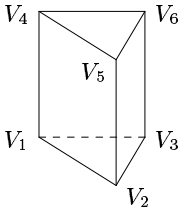


**Figure 3: Local numbering of the vertices $V_1$ to $V_6$ of a prism.**

Each quadrilateral face of the prism can be split into triangles in two different manners, giving a total number of eight different configurations to analyze. It is well known that among these eight configurations, six of them can be subdivided into three tetrahedra (see Fig. 4) and two of them cannot be subdivided into three tetrahedra (see Fig. 5). These two cases are named Schönhart polyhedra and can have a tetrahedralization only if a Steiner vertex is added in the prism. These two cases are characterized by the fact that the diagonals of the three quadrilateral faces do not share any vertex of the prism. For the six other cases, they are two vertices of the prism that share two diagonals of the quadrilateral faces.

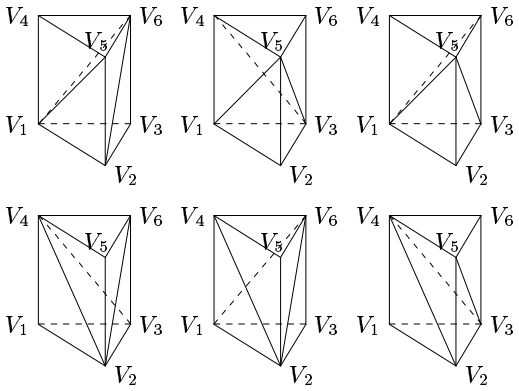


**Figure 4: The six manners to split the quadrilateral faces of the prism such that it can be subdivided into three tetrahedra.**

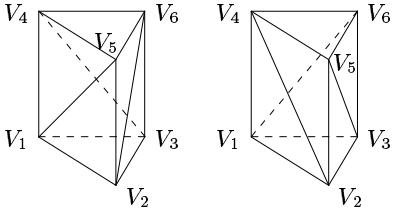


**Figure 5: The two manners to split the quadrilateral faces of the prism such that it cannot be subdivided into three tetrahedra.**

Suppose that $V_i$, $i \in \{1, 2, \ldots, 6\}$ is the smallest vertex of the six vertices of the prism. By a rotation of the prism, this smallest vertex can be located at the lower left corner (see Fig. 6).

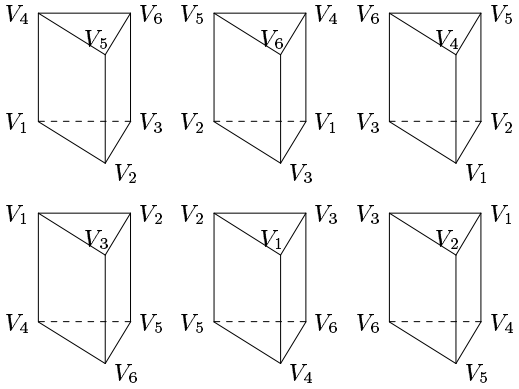


**Figure 6: The six possible rotations of the prism such that the smallest vertex is located at the lower left corner.**

In computer science, there is a common saying that stipulates that many problems can be solve by adding a layer of

indirection. Let $V_{I_1}$, $V_{I_2}$, $V_{I_3}$, $V_{I_4}$, $V_{I_5}$ and $V_{I_6}$ be the six vertices of the rotated prism where $I_1$ to $I_6$ are given by Table 2.

**Table 2: Values of the indirection $I_1$ to $I_6$ in function of the smallest vertex $V_i$.**

| Smallest vertex | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ | $I_6$ |
|---|---|---|---|---|---|---|
| $V_1$ | 1 | 2 | 3 | 4 | 5 | 6 |
| $V_2$ | 2 | 3 | 1 | 5 | 6 | 4 |
| $V_3$ | 3 | 1 | 2 | 6 | 4 | 5 |
| $V_4$ | 4 | 6 | 5 | 1 | 3 | 2 |
| $V_5$ | 5 | 4 | 6 | 2 | 1 | 3 |
| $V_6$ | 6 | 5 | 4 | 3 | 2 | 1 |

Now, considering only the prism numbered $V_{I_1}$ to $V_{I_6}$. As $V_{I_1}$ is the smallest vertex of the six vertices of the prism, it is necessarily the smallest vertex of the adjacent quadrilateral face $\square V_{I_1} V_{I_2} V_{I_5} V_{I_4}$ and of the adjacent face $\square V_{I_3} V_{I_1} V_{I_4} V_{I_6}$. The first adjacent face will be split with the diagonal $V_{I_1}$ to $V_{I_5}$, the second adjacent face will be split with the diagonal $V_{I_1}$ to $V_{I_6}$. No matter how the third quadrilateral face is split into two triangular faces, from $V_{I_2}$ to $V_{I_6}$ or from $V_{I_3}$ to $V_{I_5}$, the manner that the two adjacent quadrilateral faces are split ensures that prisms will have a tetrahedralization into three tetrahedra. See Fig. 7.

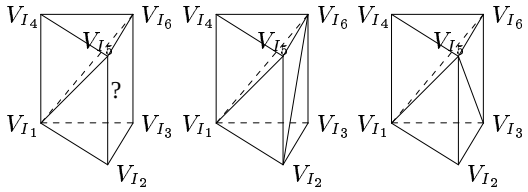


**Figure 7: Left: The smallest vertex $V_{I_1}$ of the rotated prism is also the smallest vertex of its two adjacent quadrilateral faces. Center and right: The two manners to split into two triangles the remaining quadrilateral face.**

Considering again the prism numbered $V_{I_1}$ to $V_{I_6}$, there are only two possible configurations to analyze and to subdivide into tetrahedra. Either $(V_{I_2}, V_{I_6}) < (V_{I_3}, V_{I_5})$ or $(V_{I_3}, V_{I_5}) < (V_{I_2}, V_{I_6})$. These two configurations can be subdivided into three tetrahedra that are given in Table 3.

**Table 3: The three tetrahedra that subdivide the prism in function of how the third quadrilateral face is split.**

| If | Tetrahedra |
|---|---|
| $(V_{I_2}, V_{I_6}) < (V_{I_3}, V_{I_5})$ | $\triangle V_{I_1} V_{I_2} V_{I_3} V_{I_6}$<br>$\triangle V_{I_1} V_{I_2} V_{I_6} V_{I_5}$<br>$\triangle V_{I_1} V_{I_5} V_{I_6} V_{I_4}$ |
| $(V_{I_3}, V_{I_5}) < (V_{I_2}, V_{I_6})$ | $\triangle V_{I_1} V_{I_2} V_{I_3} V_{I_5}$<br>$\triangle V_{I_1} V_{I_5} V_{I_3} V_{I_6}$<br>$\triangle V_{I_1} V_{I_5} V_{I_6} V_{I_4}$ |

## 5. HEXAHEDRON

The subdivision of the hexahedron into five or six tetrahedra is more complex than for the prism, but the underlying idea is the same. Each of the six quadrilateral faces of the hexahedron can be split in two manners, giving a total number of 64 possible configurations. It is listed in [1] that 46 configurations can be subdivided into five or six tetrahedra and 18 configurations cannot. With the face splitting based on the vertex of the quadrilateral face with the smallest identifier, this section will show that these configurations never arise.

It is supposed that the local numbering $V_1$, $V_2$, $V_3$, $V_4$, $V_5$, $V_6$, $V_7$ and $V_8$ of the vertices of the hexahedron is given by the Fig. 8.

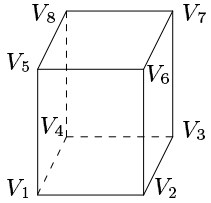


**Figure 8: Local numbering of the vertices $V_1$ to $V_8$ of an hexahedron.**

Suppose that $V_i$, $i \in \{1, 2, \ldots, 8\}$ is the smallest vertex of the eight vertices. By a rotation of the hexahedron, $V_i$ can be located at the lower front left corner. The rotation is not unique and, by convention, the vertex $V_j$ at bottom front right corner is the one corresponding to the lowest value of $j$ (see Fig. 9).

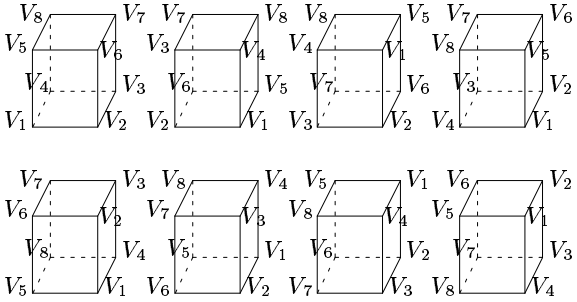

**Figure 9: The eight rotations of the hexahedron putting the smallest vertex at the lower front left corner.**

This rotation can be coded as an indirection. Let $V_{I_1}$, $V_{I_2}$, $V_{I_3}$, $V_{I_4}$, $V_{I_5}$, $V_{I_6}$, $V_{I_7}$ and $V_{I_8}$ be the eight vertices of the rotated hexahedron where $I_1$ to $I_8$ are given by Table 4.

As $V_{I_1}$ is the smallest vertex of the eight vertices of the hexahedron, it is necessarily the smallest vertex of the adjacent quadrilateral faces $\square V_{I_1} V_{I_4} V_{I_3} V_{I_2}$, $\square V_{I_1} V_{I_2} V_{I_6} V_{I_5}$ and $\square V_{I_1} V_{I_5} V_{I_8} V_{I_4}$. The first adjacent face will be split by the diagonal $V_{I_1}$ to $V_{I_3}$, the second adjacent face will be split by the diagonal $V_{I_1}$ to $V_{I_6}$ and the third adjacent face will be split by the diagonal $V_{I_1}$ to $V_{I_8}$. See Fig. 10. The three remaining quadrilateral faces can still be split by either the two possible diagonals, which leads to eight configurations to analyze.

**Table 4: Values of the indirection $I_1$ to $I_8$ in function of the smallest vertex $V_i$.**

| Smallest vertex | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ | $I_6$ | $I_7$ | $I_8$ |
|---|---|---|---|---|---|---|---|---|
| $V_1$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $V_2$ | 2 | 1 | 5 | 6 | 3 | 4 | 8 | 7 |
| $V_3$ | 3 | 2 | 6 | 7 | 4 | 1 | 5 | 8 |
| $V_4$ | 4 | 1 | 2 | 3 | 8 | 5 | 6 | 7 |
| $V_5$ | 5 | 1 | 4 | 8 | 6 | 2 | 3 | 7 |
| $V_6$ | 6 | 2 | 1 | 5 | 7 | 3 | 4 | 8 |
| $V_7$ | 7 | 3 | 2 | 6 | 8 | 4 | 1 | 5 |
| $V_8$ | 8 | 4 | 3 | 7 | 5 | 1 | 2 | 6 |

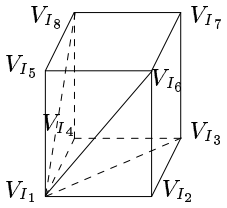


**Figure 10: The smallest vertex $V_{I_1}$ of the rotated prism is also the smallest vertex of the its three adjacent quadrilateral faces.**

These eight configurations can be reduced to four with another tricky rotation. Let the three remaining faces be numbered $\square_1 = \square V_{I_2} V_{I_3} V_{I_7} V_{I_6}$, $\square_2 = \square V_{I_3} V_{I_4} V_{I_8} V_{I_7}$ and $\square_3 = \square V_{I_5} V_{I_6} V_{I_7} V_{I_8}$. The eight configurations can be encoded with three bits. The first bit is for face $\square_1$ and is equal to one if the diagonal goes through vertex $V_{I_7}$, i.e. $(V_{I_2}, V_{I_7}) < (V_{I_3}, V_{I_6})$, zero otherwise. The second bit is for face $\square_2$ and is equal to one if the diagonal goes through vertex $V_{I_7}$, i.e. $(V_{I_4}, V_{I_7}) < (V_{I_3}, V_{I_8})$, zero otherwise. The third bit is for face $\square_3$ and is equal to one if the diagonal goes through vertex $V_{I_7}$, i.e. $(V_{I_5}, V_{I_7}) < (V_{I_6}, V_{I_8})$, zero otherwise. Summing the number of bit equal to one gives the number of diagonals that goes through the vertex $V_{I_7}$.

The other rotation of the hexahedron needed is a topological rotation of an angle of $0°$ (the identity), $120°$ or $240°$ around the axis that goes through vertices $V_{I_1}$ (South pole) and $V_{I_7}$ (North pole). The rotation of $120°$, counterclockwise around the North pole is shown in Fig. 11. This rotation can be coded by adding another layer of indirection. It can also by done with a circular affectation of vertices $V_{I_2}$, $V_{I_4}$ and $V_{I_5}$ around vertex $V_{I_1}$ and another circular affectation of vertices $V_{I_6}$, $V_{I_3}$ and $V_{I_8}$ around vertex $V_{I_7}$ as follows:

$$\begin{array}{ll} temp = I_2, & temp = I_6, \\ I_2 = I_5, & I_6 = I_8, \\ I_5 = I_4, & I_8 = I_3, \\ I_4 = temp, & I_3 = temp. \end{array}$$

The rotation of $240°$, counterclockwise around the North pole is shown in Fig. 12. It can be performed by adding a layer of indirection, by applying twice the permutations for a $120°$ rotation or by doing the two following circular affectations of vertices:

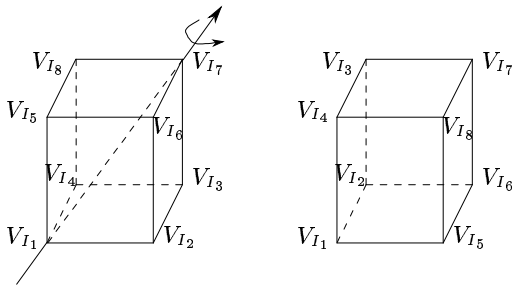


**Figure 11: Topological rotation of $120°$ around the axis that goes through vertices $V_{I_1}$ and $V_{I_7}$.**

$$\begin{array}{ll} temp = I_2, & temp = I_6, \\ I_2 = I_4, & I_6 = I_3, \\ I_4 = I_5, & I_3 = I_8, \\ I_5 = temp, & I_8 = temp. \end{array}$$

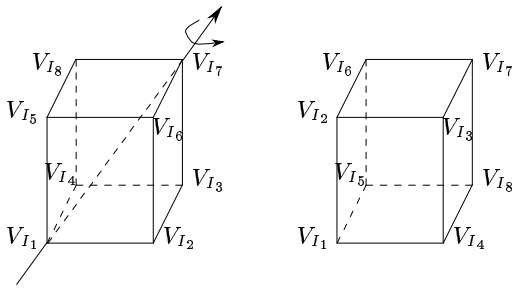


**Figure 12: Topological rotation of $240°$ around the axis that goes through vertices $V_{I_1}$ and $V_{I_7}$.**

Which rotation to perform? If the number of diagonals that goes through the vertex $V_{I_7}$ is zero or three, there is no rotation. If the number of diagonals that goes through the vertex $V_{I_7}$ is one, the rotation to perform is the one that puts this face on the right of the hexahedron. If the number of diagonals that goes through the vertex $V_{I_7}$ is two, the rotation to perform is the one that puts the third one on the right of the hexahedron. With the bitwise encoding of the three faces $\square_1$, $\square_2$ and $\square_3$, the rotation to perform is given by Table 5. Remember that a bit to one means that the diagonal goes through the vertex $V_{I_7}$.

Finally, after this rotation of $0°$, $120°$ or $240°$, there remains only four configurations to analyze and to subdivide into tetrahedra.

### 5.1. First Configuration

There is no diagonal that goes through vertex $V_{I_7}$ and no rotation around the axis that goes through vertices $V_{I_1}$ and $V_{I_7}$ was performed. The six quadrilateral faces of the hexahedron are split into triangles according to Fig. 13.

For this configuration, there exists *one* subdivision of the hexahedron into *five* tetrahedra that is given in Table 6.

**Table 6: All possible subdivisions of the hexahedron $V_{I_1}$ to $V_{I_8}$ into five or six tetrahedra in function of the number $n$ of diagonals through the vertex $V_{I_7}$.**

| $n$ | § | $\triangle_1$ | $\triangle_2$ | $\triangle_3$ | $\triangle_4$ | $\triangle_5$ | $\triangle_6$ |
|---|---|---|---|---|---|---|---|
| 0 | §5.1 | $\triangle V_{I_1}V_{I_2}V_{I_3}V_{I_6}$ | $\triangle V_{I_1}V_{I_3}V_{I_8}V_{I_6}$ | $\triangle V_{I_1}V_{I_3}V_{I_4}V_{I_8}$ | $\triangle V_{I_1}V_{I_6}V_{I_8}V_{I_5}$ | $\triangle V_{I_3}V_{I_8}V_{I_6}V_{I_7}$ | Nil |
| 1 | §5.2 | $\triangle V_{I_1}V_{I_6}V_{I_8}V_{I_5}$<br>$\triangle V_{I_1}V_{I_6}V_{I_8}V_{I_5}$ | $\triangle V_{I_1}V_{I_2}V_{I_8}V_{I_6}$<br>$\triangle V_{I_1}V_{I_2}V_{I_7}V_{I_6}$ | $\triangle V_{I_2}V_{I_7}V_{I_8}V_{I_6}$<br>$\triangle V_{I_1}V_{I_7}V_{I_8}V_{I_6}$ | $\triangle V_{I_1}V_{I_8}V_{I_3}V_{I_4}$<br>$\triangle V_{I_1}V_{I_8}V_{I_3}V_{I_4}$ | $\triangle V_{I_1}V_{I_8}V_{I_2}V_{I_3}$<br>$\triangle V_{I_1}V_{I_8}V_{I_7}V_{I_3}$ | $\triangle V_{I_2}V_{I_8}V_{I_7}V_{I_3}$<br>$\triangle V_{I_2}V_{I_1}V_{I_7}V_{I_3}$ |
| 2 | §5.3 | $\triangle V_{I_1}V_{I_5}V_{I_6}V_{I_7}$<br>$\triangle V_{I_1}V_{I_3}V_{I_4}V_{I_7}$ | $\triangle V_{I_1}V_{I_4}V_{I_8}V_{I_7}$<br>$\triangle V_{I_1}V_{I_5}V_{I_7}V_{I_8}$ | $\triangle V_{I_1}V_{I_8}V_{I_5}V_{I_7}$<br>$\triangle V_{I_1}V_{I_7}V_{I_4}V_{I_8}$ | $\triangle V_{I_1}V_{I_2}V_{I_3}V_{I_6}$<br>$\triangle V_{I_1}V_{I_2}V_{I_3}V_{I_6}$ | $\triangle V_{I_1}V_{I_4}V_{I_7}V_{I_3}$<br>$\triangle V_{I_1}V_{I_7}V_{I_5}V_{I_6}$ | $\triangle V_{I_1}V_{I_7}V_{I_6}V_{I_3}$<br>$\triangle V_{I_1}V_{I_3}V_{I_7}V_{I_6}$ |
| 3 | §5.4 | $\triangle V_{I_1}V_{I_3}V_{I_4}V_{I_7}$<br>$\triangle V_{I_1}V_{I_2}V_{I_7}V_{I_6}$<br>$\triangle V_{I_1}V_{I_2}V_{I_7}V_{I_6}$ | $\triangle V_{I_1}V_{I_4}V_{I_8}V_{I_7}$<br>$\triangle V_{I_1}V_{I_7}V_{I_8}V_{I_5}$<br>$\triangle V_{I_1}V_{I_2}V_{I_3}V_{I_7}$ | $\triangle V_{I_1}V_{I_8}V_{I_5}V_{I_7}$<br>$\triangle V_{I_1}V_{I_6}V_{I_7}V_{I_5}$<br>$\triangle V_{I_1}V_{I_3}V_{I_4}V_{I_7}$ | $\triangle V_{I_1}V_{I_6}V_{I_7}V_{I_5}$<br>$\triangle V_{I_1}V_{I_7}V_{I_2}V_{I_3}$<br>$\triangle V_{I_1}V_{I_6}V_{I_7}V_{I_5}$ | $\triangle V_{I_2}V_{I_6}V_{I_7}V_{I_1}$<br>$\triangle V_{I_1}V_{I_8}V_{I_7}V_{I_4}$<br>$\triangle V_{I_1}V_{I_7}V_{I_4}V_{I_8}$ | $\triangle V_{I_2}V_{I_7}V_{I_3}V_{I_1}$<br>$\triangle V_{I_1}V_{I_7}V_{I_3}V_{I_4}$<br>$\triangle V_{I_1}V_{I_7}V_{I_8}V_{I_5}$ |

**Table 5: Angle of the topological rotation around the axis that goes through vertices $V_{I_1}$ and $V_{I_7}$ in function of the bitwise encoding of the quadrilateral faces $\square_1$, $\square_2$ and $\square_3$.**

| Code | Rotation |
|---|---|
| 000 | 0° |
| 001 | 120° |
| 010 | 240° |
| 011 | 0° |
| 100 | 0° |
| 101 | 240° |
| 110 | 120° |
| 111 | 0° |

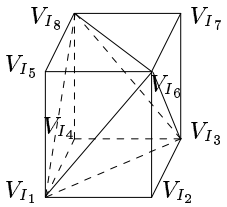


**Figure 13: First configuration with no diagonal that goes through vertex $V_{I_7}$.**

### *5.2. Second Configuration*

There is one diagonal that goes through vertex $V_{I_7}$ and the appropriate rotation puts that face on the right of the hexahedron. The six quadrilateral faces of the hexahedron are split into triangles according to Fig. 14.

For this configuration, there exists *two* subdivisions of the hexahedron into *six* tetrahedra. These two subdivisions are given in Table 6.

### *5.3. Third Configuration*

There are two diagonals that go through vertex $V_{I_7}$ and the appropriate rotation puts the face with the diagonal that does not go through vertex $V_{I_7}$ on the right of the hexahedron.

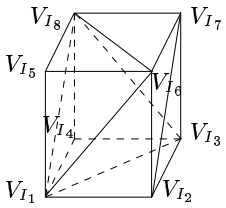


**Figure 14: Second configuration with one diagonal that goes through vertex $V_{I_7}$ and rotated such that it is on the right.**

The six quadrilateral faces of the hexahedron are split into triangles according to Fig. 15.

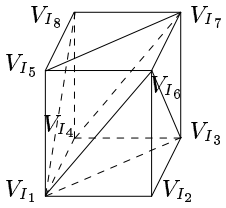


**Figure 15: Third configuration with two diagonals that go through vertex $V_{I_7}$ and rotated such that it is not on the right.**

For this configuration, there exists *two* subdivisions of the hexahedron into *six* tetrahedra. These two subdivisions are given in Table 6.

### *5.4. Fourth Configuration*

There are three diagonals that go through vertex $V_{I_7}$ and no rotation around the axis that goes through vertices $V_{I_1}$ and $V_{I_7}$ was performed. The six quadrilateral faces of the hexahedron are split according to Fig. 16.

For this configuration, there exists *three* subdivisions of the hexahedron into *six* tetrahedra. These three subdivisions are given in Table 6.

It is now proved by construction that hexahedra always have

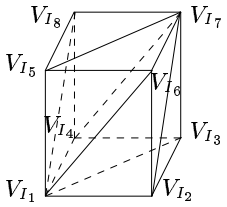


**Figure 16: Fourth configuration with three diagonals that goes through vertex $V_{I_7}$.**

a tetrahedralization in five or six tetrahedra without introducing new vertices if quadrilateral faces are split according to the smallest vertex criterion.

## 6. COMPUTER IMPLEMENTATION

For computer implementation, there are three alternatives. If the tetrahedral mesh is really needed for further computations, for each non tetrahedral element, substitute the connectivity of this element by the connectivity of the tetrahedra obtained with the algorithms of this paper.

The second alternative, if the tetrahedral mesh is not really needed, is to perform the subdivision algorithms on the fly. For example, for the prism, five tests are needed to find the vertex of the prism with the smallest identifier. The rotation is performed using the Table 2 statically stored in computer memory. Three more tests are needed to determine if $(V_{I_2}, V_{I_6}) < (V_{I_3}, V_{I_5})$ or if $(V_{I_3}, V_{I_5}) < (V_{I_2}, V_{I_6})$. So, eight tests are needed and the algorithm returns the vertices of the three tetrahedra that subdivide the prism.

The third alternative, if the tetrahedral mesh is not really needed, is to perform all the tests once, for all non tetrahedral elements, and to store the results. There are two possible diagonals along which a quadrilateral face can be split into triangles. According to Albertelli and Crawfis [1], the diagonal direction can be encoded in a one bit entity. For the prism, for example, the first bit is for face $\square V_1V_2V_5V_4$, zero indicating that $(V_1, V_5) < (V_2, V_4)$, one otherwise. The second bit is for face $\square V_2V_3V_6V_5$, zero indicating that $(V_2, V_6) < (V_3, V_5)$, one otherwise. The third bit is for face $\square V_3V_1V_4V_6$, zero indicating that $(V_3, V_4) < (V_1, V_6)$, one otherwise. A total of nine tests are needed to find the code of a prism. The six configurations of Fig. 4 correspond, from left to right and from top to bottom, to 001, 010, 011, 100, 101 and 110. The two configurations of Fig. 5 correspond, from left to right, to 000 and 111, configurations that can not be obtained with the face splitting algorithm of this paper. This code can be computed once, stored for each prism. When the tetrahedralization of a prism is needed, retrieve the code of this prism and, with a switch on this code, the three tetrahedra are given by Table 7.

The idea is the same for an hexahedron, the diagonal direction of the six quadrilateral faces can be encoded with six bits as done in Albertelli and Crawfis [1]. For each code that can be obtained with the face splitting algorithm, the subdivision into tetrahedra can be stored. This is long, tedious and error prone, but once done, it will fly.

**Table 7: The three tetrahedra that subdivide a prism in function of the encoding of the face splitting.**

| Code | $\triangle_1$ | $\triangle_2$ | $\triangle_3$ |
|---|---|---|---|
| 000 | Nil | Nil | Nil |
| 001 | $\triangle V_1V_2V_6V_5$ | $\triangle V_1V_2V_3V_6$ | $\triangle V_1V_5V_6V_4$ |
| 010 | $\triangle V_1V_2V_3V_5$ | $\triangle V_1V_5V_3V_4$ | $\triangle V_4V_5V_3V_6$ |
| 011 | $\triangle V_1V_2V_3V_5$ | $\triangle V_1V_5V_6V_4$ | $\triangle V_1V_5V_3V_6$ |
| 100 | $\triangle V_1V_2V_3V_4$ | $\triangle V_2V_3V_4V_6$ | $\triangle V_2V_6V_4V_5$ |
| 101 | $\triangle V_1V_2V_3V_6$ | $\triangle V_1V_2V_6V_3$ | $\triangle V_2V_4V_5V_6$ |
| 110 | $\triangle V_1V_2V_3V_4$ | $\triangle V_2V_5V_3V_4$ | $\triangle V_3V_5V_6V_4$ |
| 111 | Nil | Nil | Nil |

## 7. LIMITATIONS OF THE METHOD

The proposed method subdivides elements according to topological considerations. In input, there is one list of five (pyramid), six (prism) or eight (hexahedron) identifiers. In output, there will be two (pyramid), three (prism), five or six (hexahedron) lists of four identifiers. Nowhere the geometric information on vertices (the $x$, $y$, $z$ coordinates) is taken into account. This has some advantages: no roundoff error, robustness, simplicity, speed, etc, but has also some limitations.

### *7.1. Element Orientation*

Element orientation is a geometrical property which can be defined as the sign of a signed volume measure. In numerical method as in vizualisation, it is preferred that all elements have the same orientation (all signed volumes are positive). The algorithms of this paper can lead to badly oriented tetrahedra even if the initial non tetrahedral mesh is well oriented, in particular when quadrilateral faces are concave or highly skewed or when non tetrahedral elements are concave. In that situation, we suggest to retrieve the tetrahedral orientation in a post-processing step. In fact, the mesh is topologically correct and it can be geometrically optimized by any unstructured tetrahedral mesh optimizer, the simplest one being an iterative vertex centering.

### *7.2. Element Quality*

If a non tetrahedral element is skewed, its subdivision into tetrahedra will probably produce skewed tetrahedra. The algorithms of this paper for the subdivision of non tetrahedral elements into tetrahedra are not designed to increase the quality of the mesh. The quality of the resulting tetrahedral mesh will be as good or as bad as was the initial one.

Geometric information can be taken into account to select a particular subdivision when several are possible. That arises only for hexahedra: the algorithm may give two or three possible subdivisions into tetrahedra. A geometric criterion that uses the $x$, $y$, $z$ coordinates of the vertices to compute the minimum of the tetrahedral volumes, the minimum of the tetrahedral shape measures [7], etc, can be used. Nevertheless, geometric criterion can never be used to split the quadrilateral faces, otherwise all the advantages of the method are lost.

However, once the non tetrahedral mesh is tetrahedralized in a topologically conformal way, three-dimensional unstructured tetrahedral mesh optimizers can be used to improved the geometric quality of the mesh. See [3, 5, 7, 8, 10, 12, 16] among many others for three-dimensional unstructured tetrahedral mesh optimization.

### 7.3. Vertices not Uniquely Defined

Another limitation is that vertices must have a *unique* and *constant* identifier, otherwise the tetrahedralization is not necessarily conformal. The most common case where vertices have multiple identifiers is with multiblock grids. At the interface between two or more blocks, the same vertices with the same $x$, $y$, $z$ coordinates belong to each block. These vertices can have two or more identifiers, one per block they belong to. If the method is applied without any change, each block will be subdivided into a conformal block, but nothing ensures the conformity at blocks interface. To overcome that problem, a global list of vertices can be built with only one occurrence of each one. An indirection table has to be built from the local identifier of the vertices inside a block to the corresponding vertices in the global list of unique vertices.

## 8. RESULTS

This section presents two three-dimensional meshes where non tetrahedral elements are subdivided into tetrahedra. The first example is for a layer of prisms, the second one is for a structured multiblock hexahedral grid.

### 8.1. Layer of Prisms

For two-dimensional space-time finite elements [9, 15], a two-dimensional space unstructured triangular mesh (see Fig. 17 on the top) is extruded into prisms by an height $\Delta t$ in the time direction (see Fig. 17 in the center). This three-dimensional mesh for two-dimensional space-time finite elements is named a space-time slab. To adapt the mesh on the top of the space-time slab and to avoid interpolation between the space-time slabs, prisms are subdivided into tetrahedra (see Fig. 17 at the bottom). Once done, all the capabilities of three-dimensional unstructured tetrahedral mesh adaptation and optimization with mesh refinement and coarsening, diagonal and face swapping and vertex relocation can be applied [6].

### 8.2. Structured Multiblock Hexahedral Grid

In this example, a twelve blocks structured hexahedral grid is subdivided into tetrahedra. See Fig.18. The problem is that vertices at the boundaries between two blocks are not unique, they exist in both blocks. As explained in section 7.3, all blocks are parsed to build a global list of unique vertices and to build an indirection from the local identifier of the vertices inside a block to the corresponding vertices in the global list of unique vertices. Then, the hexahedra can be subdivided into tetrahedra using vertex identifiers of the global list of unique vertices.

## 9. CONCLUSION

This paper presents the problem of subdividing structured and unstructured, monoblock and multiblock meshes containing tetrahedra, pyramids, prisms and hexahedra into a consistent set of only tetrahedra, while preserving the overall mesh conformity. The proposed algorithm is to split each quadrilateral faces with a diagonal from the vertex with the smallest identifier towards the opposite vertex. This is a purely topological method that ensure that 1) the quadrilateral faces are consistently split in two triangles, 2) it is always possible to subdivide the non tetrahedral elements into tetrahedra without introducing new vertices. Inside some reasonable limitations, this method is direct, fast, generic, local, i.e. do not need any neighboring information and it is robust.

## REFERENCES


[1] G. Albertelli and R. A. Crawfis. Efficient subdivision of finite-element datasets into consistent tetrahedra. In R. Yagel and H. Hagen, editors, *IEEE Visualization*, pages 213–220, Phoenix, AZ, November 1997.

[2] M. Bern and D. Eppstein. *Computing in Euclidean Geometry*, chapter Mesh Generation and Optimal Triangulation, pages 47–123. World Scientific, D.-Z. Du and F. K. Hwang Eds., 2nd edition, 1995.

[3] É. Brière de L'Isle and P.-L. George. Optimisation de maillages tridimensionnels. Technical Report 2046, Institut National de Recherche en Informatique et en Automatique, France, September 1993.

[4] S. D. Connell and M. E. Braaten. Semi-structured mesh generation for 3D Navier-Stokes calculations. In *12th AIAA Computational Fluid Dynamics Conference*, number AIAA–95–1679–CP, pages 369–380, San Diego, CA, June 1995. AIAA.

[5] T. Coupez. *Grandes transformations et remaillage automatique*. PhD thesis, École Nationale Supérieure des Mines de Paris, November 1991.

[6] J. Dompierre, P. Labbé, A. Garon, and R. Camarero. Unstructured tetrahedral mesh adaptation for 2-D space-time finite element. In *38th Aerospace Sciences Meeting and Exhibit*, number AIAA–00–0810, Reno, NV, January 2000. AIAA.

[7] J. Dompierre, P. Labbé, F. Guibault, and R. Camarero. Proposal of benchmarks for 3D unstructured tetrahedral mesh optimization. In *7th International Meshing Roundtable*, pages 459–478, Dearborn, MI, October 1998. Sandia National Laboratories.

[8] L. A. Freitag and C. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering*, 40:3979–4002, 1997.

[9] A. Froncioni, P. Labbé, A. Garon, and R. Camarero. Interpolation-free space-time remeshing for the Burgers equation. *Communication in Numerical Methods in Engineering*, 13:875–884, 1997.

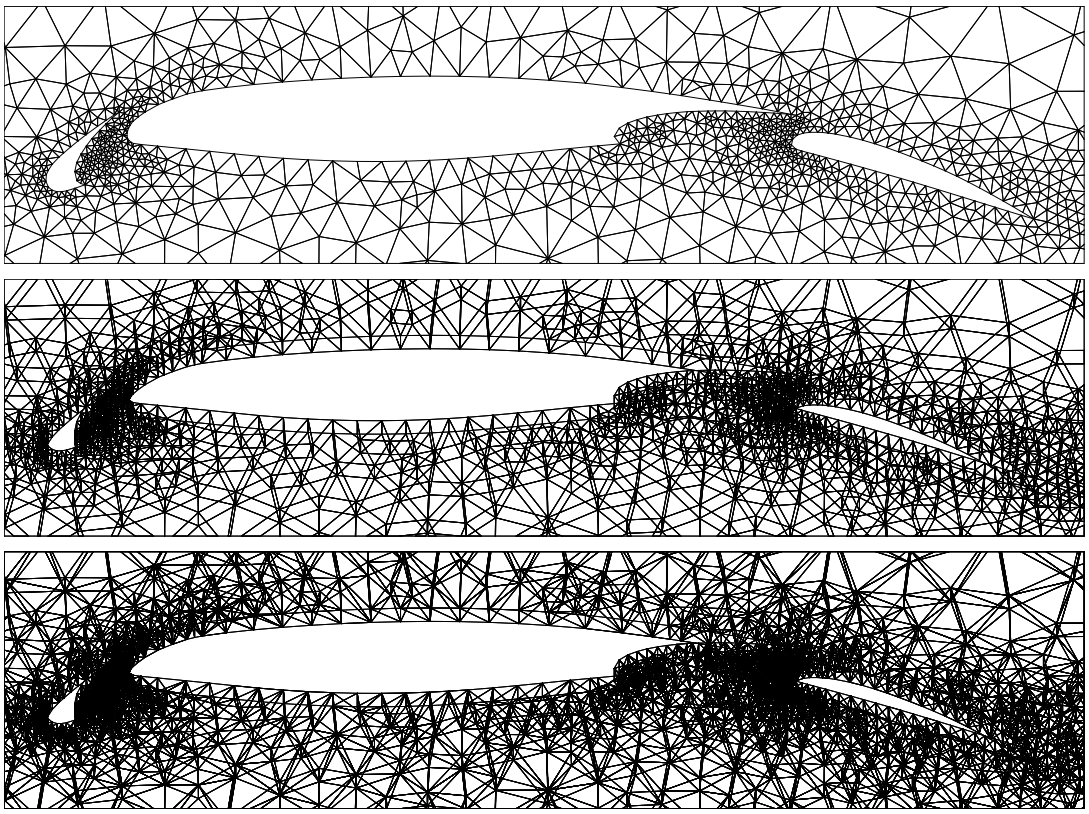

**Figure 17: *Top*: Initial triangular mesh around a three element airfoil. *Center*: The initial triangular mesh extruded in prisms by a height $\Delta t$ for two-dimensional space-time finite elements. *Bottom*: Prisms are subdivided into three tetrahedra, each with the algorithm of § 4.**


[10] P.-L. George and H. Borouchaki. *Delaunay Triangulation and Meshing. Applications to Finite Elements.* Hermès, Paris, 1998.

[11] F. Hecht. Outils et algorithmes pour la méthode des éléments finis. Thèse d'habilitation à diriger des recherches, Université Pierre et Marie Curie, Paris VI, June 1992.

[12] S. H. Lo. Optimization of tetrahedral meshes based on element shape measures. *Computers and Structures*, 63(5):951–961, 1997.

[13] R. Löhner. Matching semi-structured and unstructured grids for Navier-Stokes calculations. In *11th AIAA Computational Fluid Dynamics Conference*, number AIAA–93–3348–CP, pages 555–564, Orlando, FL, July 1993. AIAA.

[14] N. Max, B. Becker, and R. Crawfis. Flow volumes for interactive vector field visualization. In *IEEE Visualization*, Los Alamitos, CA, October 1993. IEEE.

[15] D. N'dri, A. Garon, and A. Fortin. Stable space-time formulation for the Navier-Stokes equations. In *AIAA 38th Aerospace Sciences Meeting and Exhibit*, number AIAA–00–0394, Reno, NV, January 2000.

[16] C. F. Ollivier-Gooch. An unstructured mesh improvement toolkit with application to mesh improvement, generation and (de-)refinement. In *AIAA 36th Aerospace Sciences Meeting & Exhibit*, number AIAA–98–0218, Reno, NV, January 1998.

[17] S. Pirzadeh. Unstructured viscous grid generation by advancing layers method. In *11th AIAA Applied Aerodynamics Conference*, number AIAA–93–3453, Monterey, CA, August 1993. AIAA.

[18] S. Pirzadeh. Three-dimensional unstructured viscous grids by the advancing-layers method. *American Institute of Aeronautics and Astronautics Journal*, 34(1):43–49, January 1996.

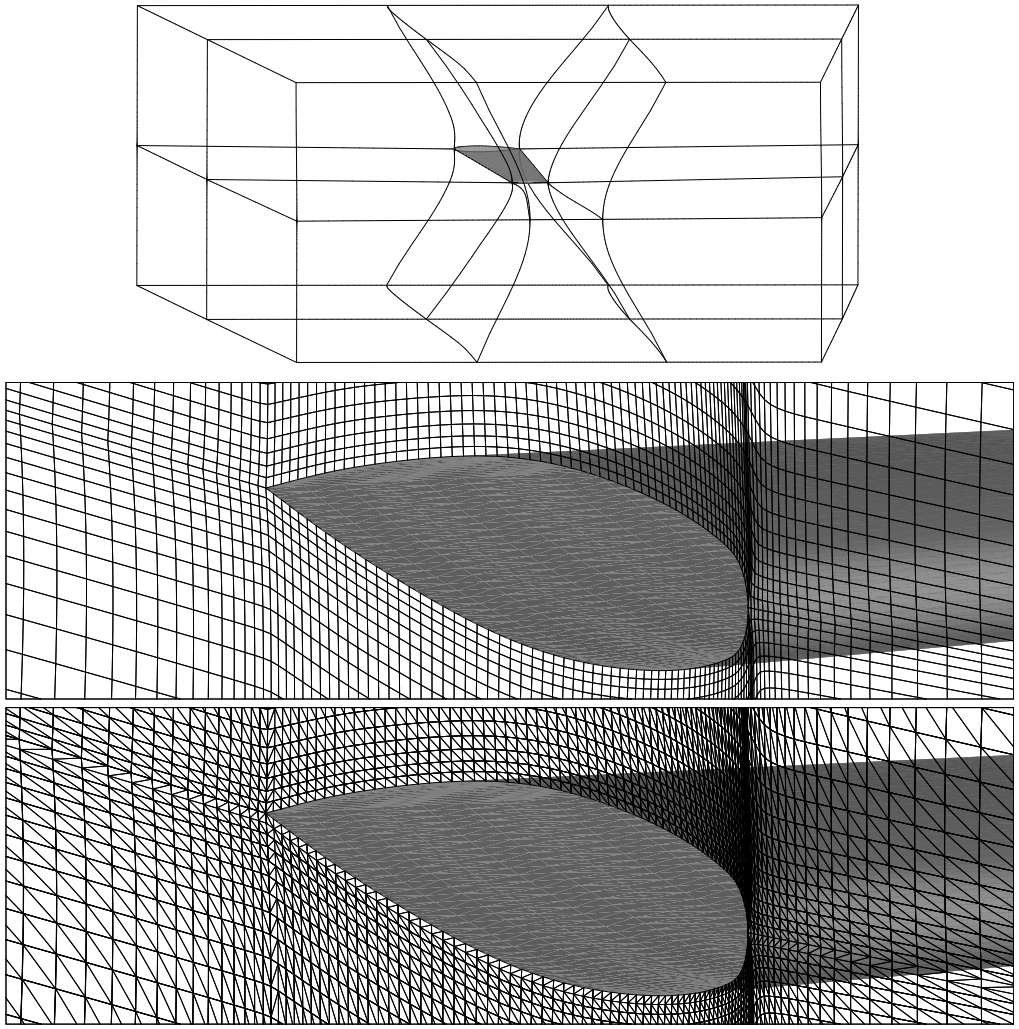

**Figure 18:** ***Top*****: Twelve blocks for a structured hexahedral grid around the Onera M6 wing.** ***Center*****: Partial view of the hexahedral structured multiblock at the symmetry plane around the wing.** ***Bottom*****: Hexahedra are subdivided into five or six tetrahedra, each with the algorithm of § 5.**