Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2014) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

23rd International Meshing Roundtable (IMR23)

# Hex-dominant meshing approach based on frame field smoothness

P.-E. Bernard[a], J.-F. Remacle[a], N. Kowalski[a], C. Geuzaine[b]

[a] *Université catholique de Louvain, Institute of Mechanics, Materials and Civil Engineering (iMMC), Bâtiment Euler, Avenue Georges Lemaître 4, 1348 Louvain-la-Neuve, Belgium*

[b] *Université de Liège, Department of Electrical Engineering and Computer Science, Montefiore Institute B28, Grande Traverse 10, 4000 Liège, Belgium*

**Abstract**

We propose here an indirect approach for building hex-dominant meshes, using the recombination of tetrahedra to create hexahedra. The efficiency of this recombination depends on the location of the initial points. A two-dimensional frame field is obtained on the boundaries by solving an elliptic PDE. We first propose a procedure to extend the two dimensional field inside the volume and to obtain a smooth three-dimensional frame field, used to insert new points. Then, we propose a point insertion algorithm based on a frame field smoothness estimator: new points are preferentially inserted in smooth frame field regions. The meshes obtained clearly exhibit a larger ratio of hexahedra, compared to more straightforward approaches.

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

*Keywords:* hexahedral meshing ; mixed hexahedral meshes ; tetrahedra recombination

## 1. Background

Hexahedral meshes are known to present interesting properties and are sometimes prefered to tetrahedral in finite element analysis. The main advantage resides in the fact that a lower number of elements is required for the same amount of vertices, compared to tetrahedra. In computational fluid dynamics, hexahedral meshes offer for instance good results along boundary layers. Indeed, the anisotropic refinement of tetrahedra is known to produce poor quality elements [1], while this operation does not affect the quality of hexahedra. In the field of solid mechanics, tetrahedra may also lead to some issues as inaccuracy or locking problems [2]. Understanding precisely why an element type is better than another in some situations may be still open for debate, but the fact is that isotropic mesh generation based on (linear) tetrahedra almost became a trivial problem, while the automatic generation of quality all-hexahedra meshes is still an open issue.

The mesh generation process represent a significant part of engineering computations. If the generation of tetrahedra may now be considered as fully automated, the generation of hex-dominant meshes often requires time consuming user interactions. Our purpose here is to develop a fully automated procedure for maximizing the amount of hexahe-

*E-mail address:* paul-emile.bernard@uclouvain.be


Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

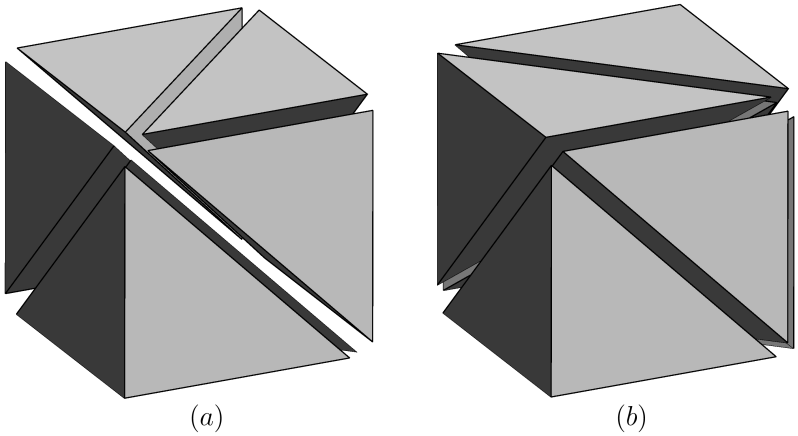


Fig. 1. Two examples of recombination patterns.

dra, in both volume and number, in non-uniform conformal isotropic meshes on arbitrary geometries.

The meshing procedure, based on the work of *Carrier-Baudouin et al.* [3], is decomposed in two different stages. First, bulk points are created inside the domain and are subsequently tetrahedralized. Then, tetrahedra are recombined to create a mixed mesh containing a maximum amount of quality hexahedra. The way points are distributed in the domain is of paramount importance for obtaining a hex-dominant mesh. The method is said *indirect*, since relying on a intermediate tetrahedral mesh. The point insertion in [3] was a bit more direct, based on standard advancing front techniques [4], placing point from the boundaries to the center of the domain. We propose here a new point insertion procedure, based on a scalar estimation of the geometry smoothness. We observe that this smoothness function allows one to identify the geometric singularities in the domain, as done in domain decomposition techniques (i.e. [5]). However, we never reconstruct the domain skeleton, connecting these singularities.
In section 2, we present our tetrahedra recombination procedure. The point insertion method is developped in section 3, while some two- and three-dimensional results are considered in section 4.

## 2. From tetrahedra to hexahedra

### *2.1. Recombination patterns*

Tetrahedra are recombined into hexahedra using the recombination patterns proposed in [6]. An hexahedron can be decomposed into five, six, or seven tetrahedra. Thus, for each tetrahedron, we check the adjacent tetrahedra to create an hexahedron according to the patterns proposed in [6]. Two patterns are illustrated in Figure 1: all tetrahedra in 1($a$) share an hexahedron diagonal as common edge, while at least one tetrahedron shares three edges with the final hexahedron in 1($b$). This procedure yields a set of potential hexahedra. Of course, with the aim to obtain a conformal mesh, many potential hexahedra present incompatibilities. The final set of hexahedra will consist in a small sub-set of these potential hexahedra.

### *2.2. Potential hexahedra as a graph*

Consider the following undirected graph $G$ : each node of $G$ is a potential hexahedron $H_i$. Hexahedra $H_i$ and $H_j$ are connected in $G$ if they are compatible i.e. they can exist simultaneously in a finite element mesh. Any subset $S = \{H_{i_1}, \ldots, H_{i_m}\}$ of $m$ hexahedra for which $H_{i_k}$ and $H_{i_l}$, with $k, l = 1, \ldots, m$, are compatible is a good set of hexahedra. In the language of graph theory, $S$ is a clique, i.e. a subgraph such that any pair of nodes are connected

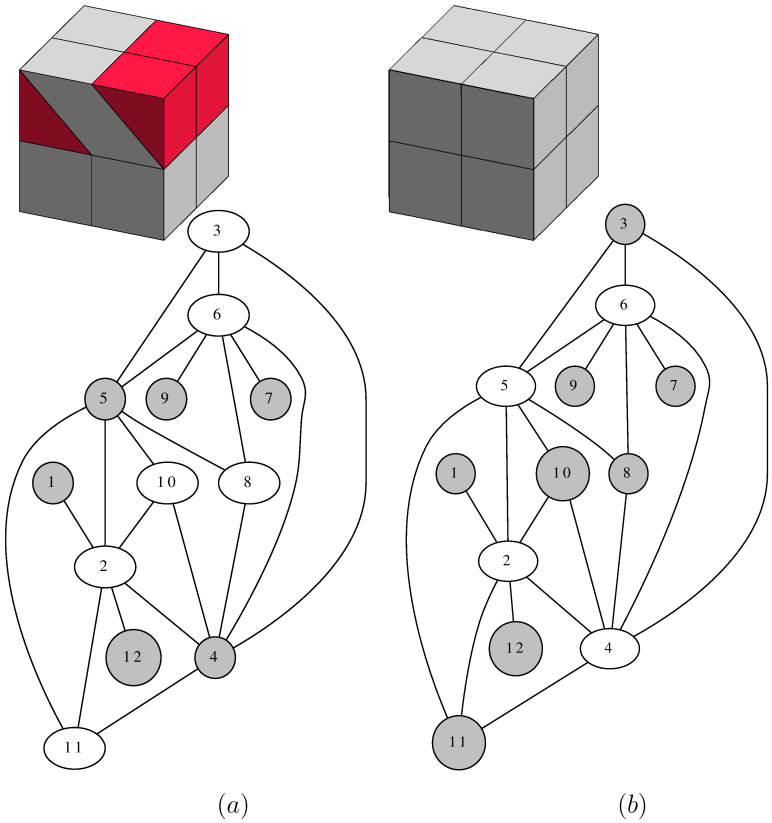


Fig. 2. The maximum clique, on the right, contains 8 hexahedra, while another clique of 6 hexahedra yields the mesh on the left, with 4 prisms in red. The graphs depicted here are incompatibility graphs: the independent sets corresponding to the meshes are highlighted in gray.

nodes. To maximize the number of hexahedra in the final mesh, we thus need to find the largest clique possible.
Note that, asymptotically, nearly all hexahedra are compatible with each other which means that the number of edges in $G$ is close to $n^n$, which is huge.It may be convenient to consider the dual graph i.e. a graph $G'$ with the same nodes but where an edge exists between $H_i$ and $H_j$ if and only if it was not present in $G$. This graph that links incompatible elements contains $O(n)$ edges. In graph theory, it is well known that finding the maximal clique is equivalent to finding the maximal independent set of the dual graph.
The maximal independent set can be defined as the larger subgraph $S$, i.e. the independent set with the highest $m$. Weights $w_j$ can also be defined on the nodes of the graph and the maximum independent set can be defined as the one that maximizes $\sum_{j=1}^{m} w_j$.

Unfortunatly, the general problem of the maximum clique/independent set is known to be NP-hard. The complexity for finding all the maximum cliques, in the worst-case, varies like $O(\alpha^n)$ for $n$ nodes.

In the algorithm proposed in [7], all the maximum cliques are found. At some point, the decision criteria to choose a node is a function $f$, equal to $n_c$, the number of compatible adjacent nodes. Of course, one could change this function to be maximized and make it depend on other criteria. For instance, one could choose a weighted sum including the element quality and the boundary proximity to create this function to maximize.

On Figure 2, the algorithm from [7] has been used to find the maximum clique (depicted on Figure 2(b)) on a very simple cubic domain. The optimal solution made of 8 hexahedra is found. The corresponding incompatibility graphs of potential hexahedra are shown, with highlighted independent sets in gray on Figure 2. We observe that 12 potentiels hexahedra have been found, while 16 were found by the recombination patterns, which means that 4 were

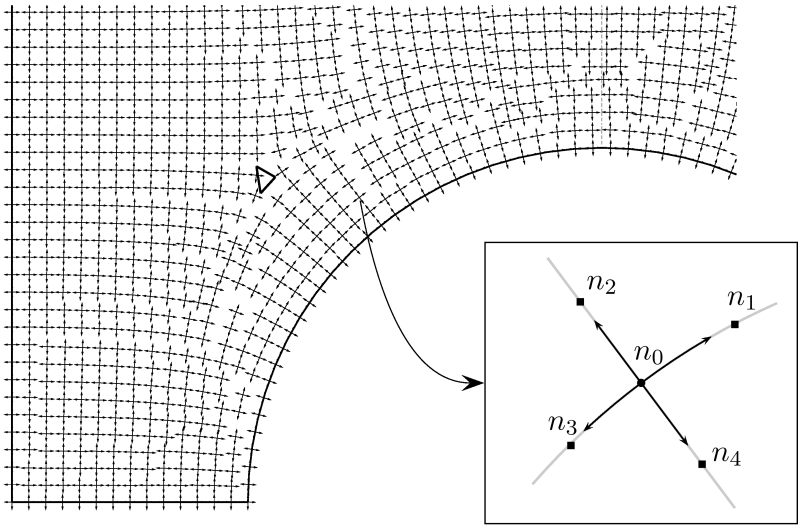


Fig. 3. Cross field on a part of the 2D domain depicted on Figure 5. The cross field directions around point $n_0$ are represented with gray curves. For such a smooth cross field, four additional points $n_i$, $i = 1 \dots 4$, are correctly computed. On the contrary, the bold black triangle contains the geometric singularity, which may lead to inaccurate additional points.

discarded due to non-conformity.

However, since such a clique algorithm presents a high computational cost, we use hereafter a simple *greedy algorithm* for meshes involving a large amount of elements, while keeping the idea of the function to maximize. The greedy algorithm consists in sorting all the potential hexahedra, according to this weighted function, and choosing in priority the ones with the higher values. Note that if the optimal solution from the clique problem is obviously the optimal solution for the greedy algorithm in the 8 hexahedra cube, this is of course not a general result.
Let us finally mention that, depending on the kind of graph involved in this mesh generation problem, one could consider using appropriate heuristics for these graphs. Such heuristics could allow one to use faster algorithms for solving the maximum clique (or approximate maximum) problem.

## 3. Point insertion

The tetrahedra recombination is knwon to be highly sensitive on the initial points potition. Here, we do not consider any post-smoothing of the points positions (as, for instance, in [8,9]) to improve the mesh quality, but we propose a pre-computation to directly improve the points location.
In the frontal point insertion from [3], the boundary points are inserted in a “first in, first out” queue. Each node produces six (or four, in two dimensions) potential additionnal surrounding nodes (as depicted in Figure 3), provided that they lie in the domain, and are not too close to any other point. These additional nodes are then inserted in the “first in, first out” queue.
First, this queue system implies that new points are inserted from the boundaries to the inner domain: this may lead to some artifacts, as visible on Figure 5(a). Moreover, the computation of surrounding nodes is correct only if the cross field is smooth enough. Indeed, around singularities (as the four singularities depicted on Figure 5(c)), this method is clearly not optimal and surrounding points should first be inserted where the cross field is smoother. On Figure 3, the bold black triangle is the location of a singularity: we observe large variation of the cross field around this point. Using the cross field in this region may lead to large inaccuracy.
Therefore, we consider here another queue system, based on a *smoothness* function: we first insert points where the underlying cross field is smooth. Two issues must be handled at this point. First, the definition of smoothness itself. Then, the fact that a good point insertion algorithm is useless if based on a poor 3D cross field: it must be smoothed

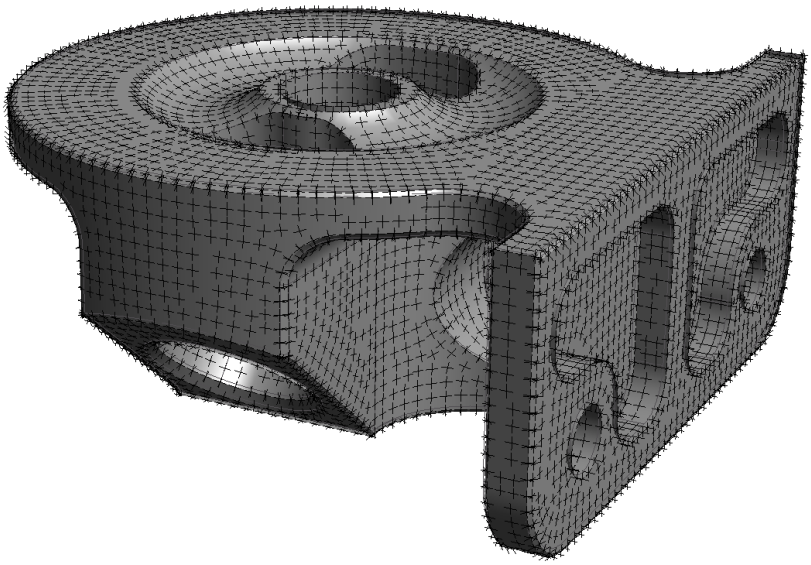

Fig. 4. Illustration of boundary cross field.

as well.

### 3.1. Cross field on boundaries

The first step consists in computing a so-called *cross field*, i.e. a field indicating the principal geometrical directions in the domain. To achieve this, we first create a mesh of triangles (or tetrahedra in 3D), called *background mesh*. Cross fields were used for instance in computer graphics applications in [10,11], or in [3,12] for mesh generation.
Let us briefly recall the main idea for computing the cross field. A 2D cross field is composed of two orthogonal unit vector oriented along the curvature of the domain. This information can be, in 2D, resumed to one information: the angle $\theta$ between the cross field and a reference basis. The cross field is found by solving an elliptic PDE, to propagate the boundary conditions of $\theta$ into the surface $\Omega$:

$$\begin{aligned} \nabla^2 a(\theta) = 0 \quad \text{on } \Omega, \qquad a(\theta) = \bar{a} \quad \text{on } \partial\Omega \\ \nabla^2 b(\theta) = 0 \quad \text{on } \Omega, \qquad b(\theta) = \bar{b} \quad \text{on } \partial\Omega \end{aligned} \tag{1}$$

with $a(\theta) = \cos(4\theta)$, $b(\theta) = \sin(4\theta)$, and $\bar{a}, \bar{b}$ the boundary conditions set in such a way that the cross field is aligned on the normal vector to the domain. The cross field is eventually given by

$$\theta = \frac{1}{4}\text{atan2}(b, a).$$

Details about cross field computation can be found in [3,12].
On Figures 3 and 4 are depicted some examples of 2D cross fields. The next steps consist in extending this 2D cross field inside the domain, and in inserting new points, well suited for triangle (or tetrahedra in 3D) recombination, according to the cross field.

### 3.2. Scalar smoothness function

If the 3D cross field is based only on the nearest boundary node's cross field, the points position will present the same artifacts than using the *fifo* frontal insertion procedure. Smoothing this cross field requires somehow to introduce

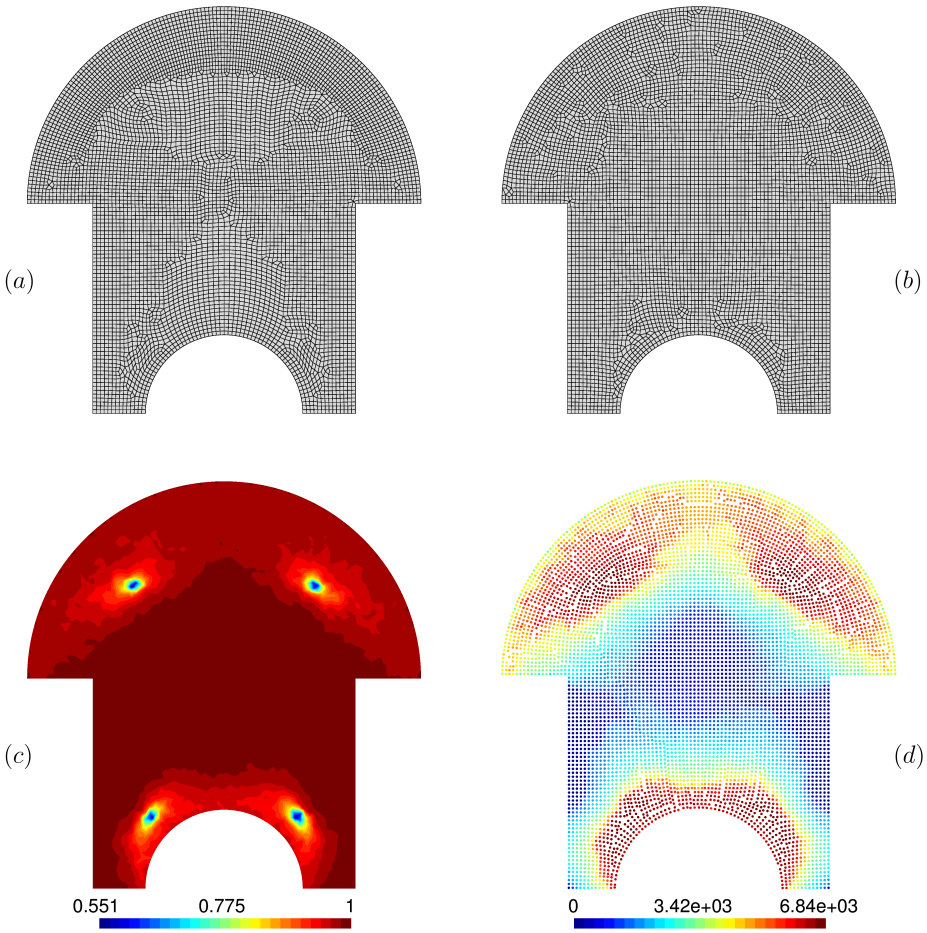


Fig. 5. Comparison between the new approach (b) and the frontal points insertion (a). The cross field smoothness depicted on (c) is used to choose which points are inserted in priority. These inserted points are visible on (d), the value scale showing their insertion order.

a measure of smoothness.

Let us consider the cross field $\boldsymbol{C}$ made of vectors $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$:

$$\boldsymbol{C} = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \tag{2}$$

and compare it to the identity cross field made of $(\boldsymbol{e_x}, \boldsymbol{e_y}, \boldsymbol{e_z})$. First, let us apply $\boldsymbol{u}$ on $\boldsymbol{e_x}$: the rotation axis $\boldsymbol{a_1}$ is the vectorial product

$$\boldsymbol{a_1} = (\boldsymbol{u} \times \boldsymbol{e_x}) \text{ signof}(\boldsymbol{u} \cdot \boldsymbol{e_x})$$

while the rotation angle $\alpha_1$ is given by

$$\alpha_1 = \arccos\left(|\boldsymbol{u} \cdot \boldsymbol{e_x}|\right)$$

and is always positive. Then, applying the corresponding rotation matrix

$$\boldsymbol{R} = \begin{bmatrix} a_{1x}^2 + (1 - a_{1x}^2)c & a_{1x}a_{1y}(1-c) - a_{1z}s & a_{1x}a_{1z}(1-c) + a_{1y}s \\ a_{1x}a_{1y}(1-c) + a_{1z}s & a_{1y}^2 + (1 - a_{1y}^2)c & a_{1z}a_{1y}(1-c) - a_{1x}s \\ a_{1x}a_{1z}(1-c) - a_{1y}s & a_{1z}a_{1y}(1-c) + a_{1x}s & a_{1z}^2 + (1 - a_{1z}^2)c \end{bmatrix} \tag{3}$$

where $c$ stands for $\cos(\alpha_1)$, $s$ for $\sin(\alpha_1)$, to the cross field, we obtain a new cross field $\boldsymbol{C'} = \boldsymbol{RC}$. The vectors $\boldsymbol{v'}$ and $\boldsymbol{w'}$ are in the plane defined by $\boldsymbol{e_y}, \boldsymbol{e_z}$. A second rotation is required to fully apply $\boldsymbol{C}$ on the identity: the second rotation angle $\alpha_2$ is given by $\min\Big(\arccos(|\boldsymbol{v} \cdot \boldsymbol{e_y}|), \arccos(|\boldsymbol{v} \cdot \boldsymbol{e_z}|)\Big)$, while the second rotation axis is $\pm\boldsymbol{e_x}$.
We then consider the sum of the two rotation angles, $s_1 = \alpha_1 + \alpha_2$. Instead of applying $\boldsymbol{u}$ on $\boldsymbol{e_x}$, one could apply $\boldsymbol{v}$ or $\boldsymbol{w}$, leading to different values of $s_2$ and $s_3$. We choose as smoothness measure the sum $s = \min(s_i), \quad i \in [1, 2, 3]$, i.e. somehow a minimum angular distance between two cross fields.

In two dimensions, the problem is much simpler: we just compute the mean angle between a cross field and all its direct neighbors.

### 3.3. Smoothing the 3D cross field

The cross field values on the boundaries are set as the solution of elliptic PDE (1).

We use the following iterative procedure for smoothing the field. For each node, $n_0$, we consider its $N$ adjacent nodes $n_i, i \in [1 \dots N]$. A smoothness value $s_i$ is available for each node. For every adjacent node $n_i$, we compare the cross fields of $n_i$ and $n_0$, as explained above, and choose the minimum angular distance transformation. We have two rotation axes and two rotation angles, which can be reduced to one single rotation axis and angle, $\boldsymbol{a}_i$ and $\alpha_i$ respectively. We wish to *align* the cross field of $n_0$ on the adjacent cross fields. We consider the vector

$$\boldsymbol{a}^* = \sum_{i=1}^{N} \boldsymbol{a}_i \alpha_i \ . \tag{4}$$

To align the cross field of $n_0$, we apply a rotation given by the angle $\|\boldsymbol{a}^*\|$ and the axis $\dfrac{\boldsymbol{a}^*}{\|\boldsymbol{a}^*\|}$.

However, a key point has to be mentionned here: we want to extend the information from the boundaries in a smooth way, which means that we want to minimize the impact of the singularity points (or lines). To prevent the information to propagate from these singularities, we use different weights $c_i(s_i)$ depending on the local smoothness. Equation (4) thus eventually reads:

$$\boldsymbol{a}^* = \sum_{i=1}^{N} c_i(s_i) \boldsymbol{a}_i \alpha_i \ . \tag{5}$$

The values of coefficients $c_i(s_i)$ are typically 1 if $s_i$ is larger than some given threshold, $10^{-3}$ otherwise.
This *Gauss-Seidel*-like iterative procedure progressively extends the cross field information from the boundaries to the center of the domain, minimizing the impact of the singularities while applying some smoothing.
We iterate on this procedure until the cross field becomes stable on the whole domain.

### 3.4. Insertion procedure

On the background mesh composed of tetrahedra, the cross field is smoothed according to the procedure described above, and a scalar smoothness is computed at each node. All boundary nodes are stored in an ordered set, the order being the local scalar smoothness at the node. Another set $S$ is initially empty, and will eventually contain all the new nodes strictly inside the volume of the final mesh. Then, the following loop is executed until the ordered set is empty:

1. In the ordered set, choose the node $n_0$ with the higher smoothness.
2. Remove $n_0$ from the ordered set.
3. Recover the cross field at this node $n_0$, and create six new nodes $n_i$ in the three directions of the cross field.
4. For each node $n_i$, if it is in the domain and compatible with the other nodes of $S$ (not too close), add $n_i$ to $S$ and insert $n_i$ into the ordered set, according to the smoothness value at $n_i$.

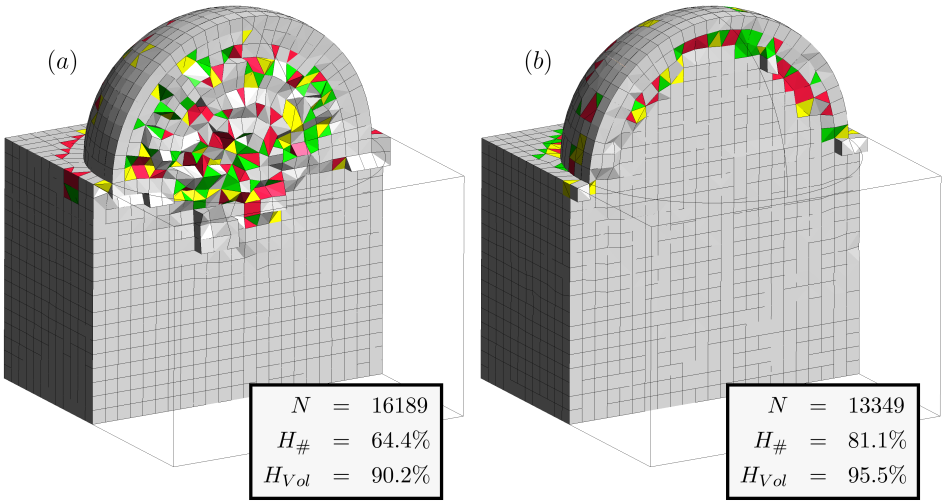


Fig. 6. Cut views of hybrid meshes obtained with the frontal approach (a) and the cross field-based approach (b). $N$, $H_{\#}$ and $H_{Vol}$ are the total number of elements, the hexahedra ratio in number of elements and in volume respectively.

At the end of the loop, the boundary nodes and the nodes in $S$ are the final mesh nodes, used to create a mesh of tetrahedra to recombine.

Note that the same procedure is applied for the creation of the 2D mesh on the boundaries, except for the cross field smoothing. Indeed, the 2D cross field comes from the resolution of the elliptic PDE and does not require any smoothing.

## 4. Numerical results

The following results illustrate the efficiency of the method on two- and three-dimensional cases. For three-dimensional examples, comparison is made with the point insertion method from [3].

### *4.1. 2D example: testing the point insertion*

The simple domain depicted in Figure 5 is a good example to illustrate the advantage of the method. Both computations use the same 2D cross field, only the point insertion algorithm changes. Full quadrilateral meshes were obtained using the Blossom-Quad algorithm [13]. On Figures 5 (a) and (b) are depicted the final quad meshes obtained with the "frontal" insertion and the new insertion method, respectively. We clearly observe many artifacts on Figure 5 (a), due to the fact that the new points are inserted from the boundaries, independently of the cross field regularity. On Figure 5 (c) and (d) are depicted the scalar smoothness estimation of the cross field, and the order of points insertion with the new approach, respectivelly. The first points to be inserted correspond to the smooth part of the domain. Even with the simple scalar smoothness estimation, we clearly observe the position of four singularities, as computed for instance in domain partitioning methods as [5].

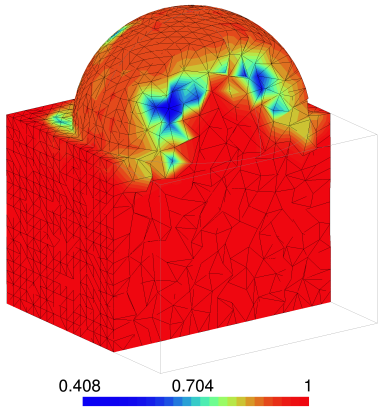


Fig. 7. Cut view of the cross field smoothness estimation on the background mesh.

### *4.2. 3D examples: point insertion and cross field smoothing*

In the following meshes on Figures 6, 8 and 9, hexahedra are colored in gray, while the colors yellow, red and green correspond to tetrahedra, prisms and pyramids respectively. On Table 1 are summarized the ratio of hexahedra for every 3D example. For each computation using the new approach, 20 iterations were performed for smoothing the cross field.

The new cross field-based approach is particularly efficient on the domain on Figure 6, composed of a cube and a half-sphere. The cross field has been efficiently smoothed, as depicted on Figure 7: the cross field in the center of the half-sphere and in the cube is clearly not affected by the surrounding boundaries, while it is strongly affected if set to the value of the nearest neighbor on the boundary. Once the cross field has been correctly smoothed, the new insertion point algorithm itself can be evaluated. The new approach clearly fills the half-sphere with hexahedra (Figure 6(b)), while a frontal algorithm leads to many remaining tetrahedra (Figure 6(a)). Note the 20 iterations for smoothing the cross field (using a naive code implementation) took 30.2 seconds.

Table 1. Hexahedra ratio in number and volume for the three-dimensional cases. *(a)* figures correspond to the frontal point inseretion, while *(b)* figures are the new cross field-based approach. The duration $t$ is the time took in seconds for each smoothing iteration, on a single Core i7 2012 laptop computer.

| Figure | # vertices | # elements | t ($s$) | Hex ratio (%) | |
|---|---|---|---|---|---|
| | | | | # | Vol |
| 6 (a) | 17080 | 16189 | 1.51 | 64.4 | 90.2 |
| (b) | 16576 | 13349 | | 81.1 | 95.5 |
| 8 (a) | 50159 | 42761 | 3.11 | 63.7 | 88.1 |
| (b) | 49997 | 38664 | | 74.7 | 92.4 |
| 9 (a) | 114590 | 172429 | 6.7 | 40.9 | 76.6 |
| (b) | 112087 | 160688 | | 45.0 | 79.4 |

Figures 8 and 9 compare the two approaches for some mechanical pieces. On Figure 8, we observe that the domain clearly presents some large regions, compared to the element size, where the cross field-based approach is able to bring a clear advantage: the hexahedra are well organized in these regions, leading to an hexahedra ratio 4% larger in volume, and more than 10% larger in number. On the contrary, the filter mount on Figure 9 presents thiner parts (compared to the element size used), explaining why the advantage of the new approach is much less obvious. Indeed, one can observe on the cut view of Figure 9(b) that the hexahedra concentration is larger around the central hole,

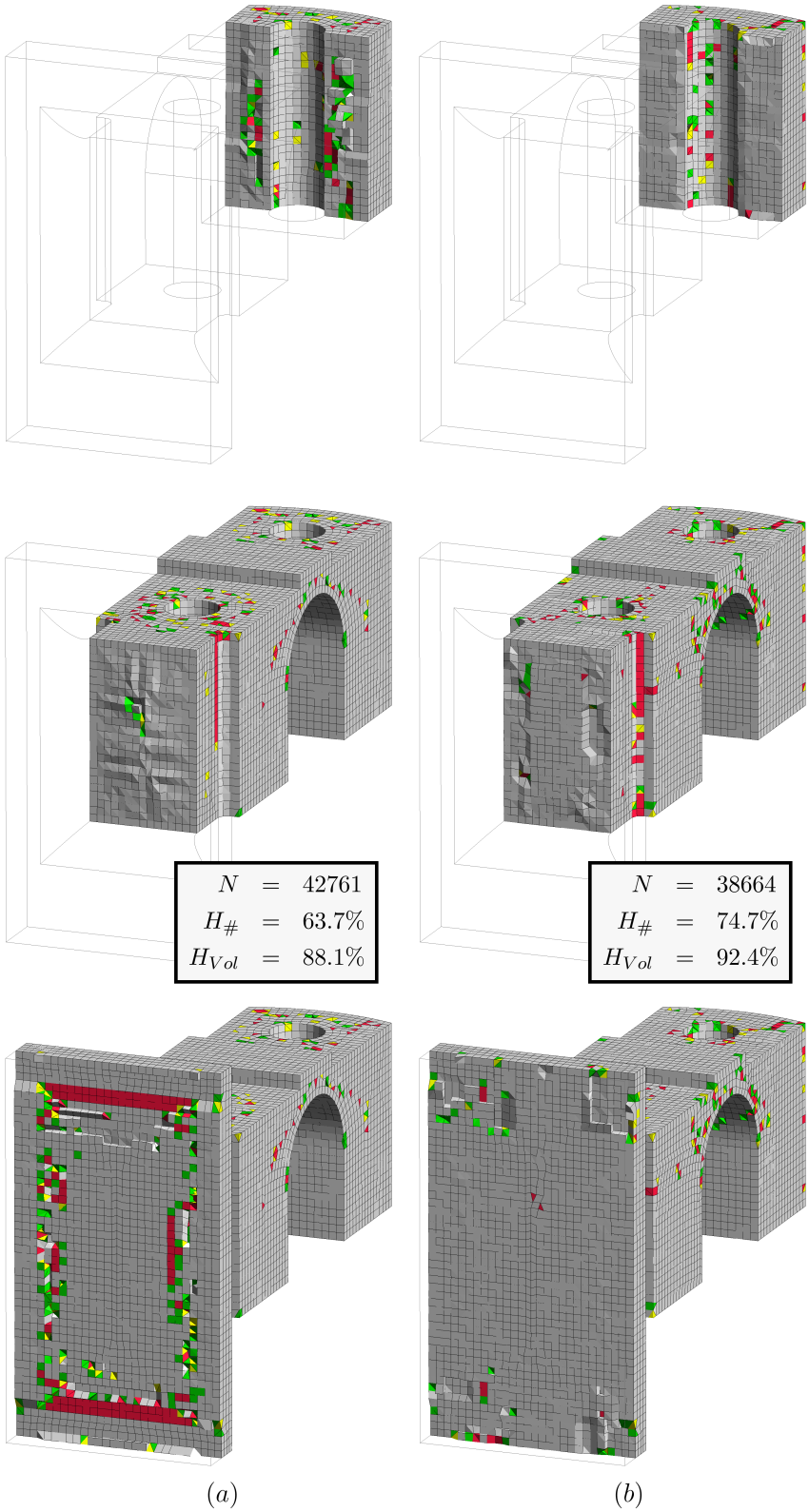


Fig. 8. Cut views of some mechanical piece with frontal approach (a) and cross field-based approach (b). $N$, $H_{\#}$ and $H_{Vol}$ are the total number of elements, the hexahedra ratio in number of elements and in volume respectively.

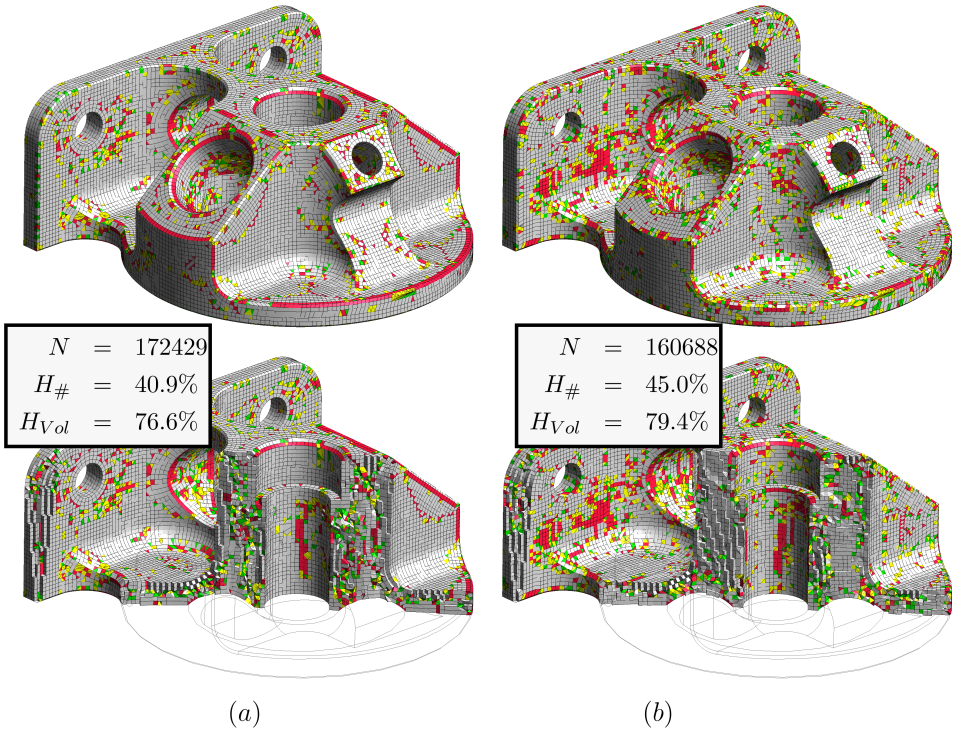


Fig. 9. Cut views of a filter mount with frontal approach (a) and cross field-based approach (b). $N$, $H_{\#}$ and $H_{Vol}$ are the total number of elements, the hexahedra ratio in number of elements and in volume respectively.

where the domain is thicker, while presenting less hexahedra on the boundaries of thiner parts.

## 5. Conclusions

In the framework of an automated hex-dominant mesh generation based on tetrahedra recombination, we proposed here a new point insertion method based on a smoothness criteria of the cross field. The simple smoothness estimation is sufficient to localize the geometric singularity points in the domain. We also proposed a basic iterative procedure to smooth the 3D cross field. In two dimensions, we visualy observe that the artifacts from a frontal point insertion from the boundaries disappear. In three dimensions, numerical experiments suggest that the proposed approach leads to a larger ratio of hexahedra for a sufficiently small mesh size. However, it requires a cross field smoothing and the computation of a smoothness estimation.
Future works include the extension from a scalar smoothness function to a vectorial one. Such an information could lead to results similar to sweeping meshing techniques [14,15], particularly efficient for two-and-one-half dimensional geometries.

## References

[1] R. Biswas, R. C. Strawn, Tetrahedral and hexahedral mesh adaptation for cfd problems, Applied Numerical Mathematics 26 (1998) 135 – 151.
[2] M. A. Puso, J. Solberg, A stabilized nodally integrated tetrahedral, International Journal for Numerical Methods in Engineering 67 (2006) 841–867.
[3] T. Carrier Baudouin, J.-F. Remacle, E. Marchandise, F. Henrotte, C. Geuzaine, A frontal approach to hex-dominant mesh generation, Advanced Modeling and Simulation in Engineering Sciences 1 (2014) 1–30.

[4] T. D. Blacker, M. B. Stephenson, Paving: A new approach to automated quadrilateral mesh generation, International Journal for Numerical Methods in Engineering 32 (1991) 811–847.
[5] N. Kowalski, F. Ledoux, P. Frey, A pde based approach to multidomain partitioning and quadrilateral meshing, in: X. Jiao, J.-C. Weill (Eds.), Proceedings of the 21st International Meshing Roundtable, Springer Berlin Heidelberg, 2013, pp. 137–154.
[6] S. Yamakawa, K. Shimada, Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells, Int J Numer Meth Eng 57 (2003) 2099–2129.
[7] E. Tomita, A. Tanaka, H. Takahashi, The worst-case time complexity for generating all maximal cliques and computational experiments, Theoretical Computer Science 363 (2006) 28 – 42.
[8] B. Levy, Y. Liu, $l_p$ centroidal voronoi tessellation and its applications, in: H. Hoppe (Ed.), ACM Transactions on Graphics, Los Angeles, 2010.
[9] T. Carrier Baudouin, J.-F. Remacle, E. Marchandise, J. Lambrechts, F. Henrotte, Lloyd's energy minimization in the lp norm for quadrilateral surface mesh generation, Engineering with Computers 30 (2012) 97–110.
[10] B. Lévy, S. Petitjean, N. Ray, J. Maillot, Least squares conformal maps for automatic texture atlas generation, ACM Transactions on Graphics 21 (2002) 362–371.
[11] D. Bommes, H. Zimmer, L. Kobbelt, Mixed-integer quadrangulation, in: SIGGRAPH '09: ACM SIGGRAPH 2009 papers, ACM, New York, NY, USA, 2009, pp. 1–10. doi:http://doi.acm.org/10.1145/1576246.1531383.
[12] J.-F. Remacle, F. Henrotte, T. Carrier-Baudouin, E. Bechet, E. Marchandise, C. Geuzaine, T. Mouton, A frontal delaunay quad mesh generator using the l norm, International Journal for Numerical Methods in Engineering 94 (2013) 494–512.
[13] J.-F. Remacle, J. Lambrechts, B. Seny, E. Marchandise, A. Johnen, C. Geuzaine, Blossom-quad: a non-uniform quadrilateral mesh generator using a minimum cost perfect matching algorithm, Int J Numer Meth Eng accepted (2011).
[14] P. M. Knupp, Next-generation sweep tool: A method for generating all-hex meshes on two-and-one-half dimensional geometries., in: IMR, 1998, pp. 505–513.
[15] M. L. Staten, S. A. Canann, S. J. Owen, Bmsweep: locating interior nodes during sweeping, Engineering with Computers 15 (1999) 212–218.