INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2010; **00**:1–6 *Prepared using* nmeauth.cls *[Version: 2002/09/18 v2.02]*

Blossom-Quad: a non-uniform quadrilateral mesh generator using a minimum cost perfect matching algorithm

J.-F. Remacle[1*], J. Lambrechts[1], B. Seny[1], E. Marchandise[1], A. Johnen[2] and C. Geuzaine[2]

[1] *Université catholique de Louvain, Institute of Mechanics, Materials and Civil Engineering (iMMC), Bâtiment Euler, Avenue Georges Lemaître 4, 1348 Louvain-la-Neuve, Belgium*
[2] *Université de Liège, Department of Electrical Engineering and Computer Science, Montefiore Institute B28, Grande Traverse 10, 4000 Liège, Belgium*

SUMMARY

A new indirect way of producing all-quad meshes is presented. The method takes advantage of a well known algorithm of the graph theory, namely the Blossom algorithm that computes the minimum cost perfect matching in a graph in polynomial time. The new Blossom-Quad algorithm is compared with standard indirect procedures. Meshes produced by the new approach are better both in terms of element shape and in terms of size field efficiency. Copyright © 2010 John Wiley & Sons, Ltd.

KEY WORDS: quadrilateral meshing; surface remeshing; graph theory; optimization; perfect matching

1. Introduction

Quadrilateral surface meshes are sometimes considered as superior to triangular meshes for finite element simulations. Discussions about if and why quadrilaterals are better than triangles are usually passionate in the finite element community. We will not try to argue about that thorny question here—but we assume that quadrilateral meshes are indeed useful and in this paper we present a new way of generating such meshes.

Let us first briefly recall which kinds of methods can be used to build non uniform quadrilateral meshes in an automatic manner. There are essentially two categories of methods.

In *direct methods*, the quadrilaterals are constructed at once, either using some kind of advancing front technique [1] or using regular grid-based methods (quadtrees). Advancing front methods for quads are considered to be non robust and quadtree methods usually produce low quality elements close to the boundaries of the domain and are unable to fulfill general size constraints (anisotropy, strong variations).

In *indirect methods*, a triangular mesh is built first. Triangle-merge methods then use the triangles of the initial mesh and recombine them to form quadrangles [2, 3]. Other more sophisticated indirect methods use a mix of advancing front and triangle merge [4].

*Correspondence to: jean-francois.remacle@uclouvain.be

Received
Revised

The method we present here is an indirect approach to quadrilateralization. We make use of a famous algorithm of the theory of graphs: the Blossom algorithm, proposed by Edmonds in 1965 [5, 6], which allows to find the minimal cost perfect matching of a given graph. The new method has some clear advantages: (i) it provides a mesh that is guaranteed to be quadrilateral only, (ii) it is optimal in a certain way and (iii) it is fast.

2. Mesh quality measures

The aim of the mesh generation process described in this paper is to build a mesh made of quadrilaterals that has controlled element sizes and shapes. We are interested in generating non uniform quadrilateral meshes. Local information about sizes is given through the definition of a mesh size field that returns, for every point $\vec{x}$ in the domain, a "characteristic" length $h(\vec{x})$ that has to be fulfilled by the mesh.

Let $\vec{a}$ and $\vec{b}$ be two points of $\mathbb{R}^3$. The adimensional length of the vector $\overrightarrow{ab}$ with respect to the non uniform size field h is defined as [7, 3]:

$$l^h(\overrightarrow{ab}) = \left\|\overrightarrow{ab}\right\| \int_0^1 \frac{1}{h(\overrightarrow{a} + t\,\overrightarrow{ab})}\, dt. \tag{1}$$

An optimum mesh in term of the size is a mesh for which every edge i is of adimensional size l_i^h equal to one. It is of course impossible to have such a perfect unit mesh. Here, we define the *efficiency index* [7] τ of a mesh as the exponential of the mean value of the difference between each edge length l_i^h and one:

$$\tau[\%] = 100 \exp\left(\frac{1}{n_e}\sum_{i=1}^{n_e} d_i\right), \tag{2}$$

with $d_i = l_i^h - 1$ if $l_i^h < 1$, $d_i = \dfrac{1}{l_i^h} - 1$ if $l_i^h > 1$ and n_e the number of edges in the mesh. Surface mesh algorithms usually produce triangular meshes with typical values of τ around $\tau = 85\%$, i.e., with non-dimensional sizes around $1/\sqrt{2} \leq l_i^h \leq \sqrt{2}$.

Having the right sizing for the mesh is not enough: mesh generators should also provide meshes with controlled element qualities. We then define a quality measure for quadrilateral elements. Consider a quadrilateral element q and its the four internal angles α_k, $k = 1, 2, 3, 4$. We define the quality $\eta(q)$ of q as:

$$\eta(q) = \max\left(1 - \frac{2}{\pi}\max_k\left(\left|\frac{\pi}{2} - \alpha_k\right|\right), 0\right). \tag{3}$$

This quality measure is 1 if the element is a perfect quadrilateral and is 0 if one of those angles is either ≤ 0 or $\geq \pi$. In what follows, we will present statistics for the quadrilateral meshes:

- The efficiency index τ, which measures the adequacy of the mesh with the mesh size field. The index τ is smaller or equal to one, and should be as close as possible to $\tau = 100\%$.
- The average element quality $\bar{\eta}$ as well as the worst element quality η_w, which can be important in the context of finite element simulations.

3. Indirect quadrilateralization using a non-optimal matching algorithm

In §1 we have made the distinction between direct and indirect methods for the construction of quadrilateral meshes. In the case of indirect methods, a triangular mesh is first constructed. Then, triangles are recombined in order to produce quadrangles.

Consider a triangular mesh made of n_t triangles t_i, $i = 1, \dots n_t$. In what follows, we consider internal edges e_{ij} of the mesh that are common to triangles t_i and t_j. We define a cost function $c(e_{ij}) = 1 - \eta(q_{ij})$ that is associated to each graph edge e_{ij} of the mesh and that is defined as the mesh quality of the quadrilateral q_{ij} that is formed by merging the two adjacent triangles t_i and t_j. Usual indirect quadrilateralization procedures work as follows [3]. Edges e_{ij} of the graph are sorted with respect to their individual cost functions. Then, the two triangles that are adjacent to the best edge e_{ij} of the list are recombined into a quadrilateral. Triangles t_i and t_j are tagged in order to prevent other edges that are adjacent either to t_i or to t_j to be used for another quadrilateral forming. Then, the algorithm processes the ordered list of edges, forming quadrilaterals with triangles adjacent to an edge as long as none of those adjacent triangles are tagged. Fig. 1 shows an illustration of this procedure for a rectangular domain of size 1×3 and a mesh size field defined by

$$h(x, y) = 0.1 + 0.08 \sin(3x) \cos(6y).$$

Isolated triangles inevitably remain in the mesh and the resulting mesh is not made of quadrilaterals only. The mesh is then said to be *quad-dominant*. In the example of Fig. 1, the resulting mesh is made of 836 quads and 240 triangles.

A mesh composed of quadrilaterals can be build subsequently using a uniform mesh refinement procedure [3]. Every quadrilateral of the quad-dominant mesh is split into four sub-quadrilaterals and every triangle is split into three sub-quadrilaterals (see Fig. 2). In order to fulfill the size criterion $h(\vec{x})$, the initial triangular mesh should thus be built using a size field with twice the value (i.e. $2h$) that is expected in the final mesh.

The recombination process just described is sub-optimal. It does not provide the best set of edges to be recombined with respect to some general cost function. Indeed, the only optimality property of this algorithm is that it ensures that the best triangle pair will be recombined.

The second part of the algorithm, namely the mesh refinement step, also has some drawbacks. Splitting every element of the mesh produces a mesh that has half the size of the initial mesh. It is of course possible to generate an initial mesh with double the required size. Yet, with real geometries, the new vertices will have to be added on the geometry, which is not trivial. On the other hand, the refinement step does not allow a sharp control of the mesh size. On Fig. 2, the procedure ends with an efficiency index of 79%, which cannot be considered as good.

In [2] the authors propose a scheme for recombining triangular meshes that does not always require the refinement step, using a kind of advancing front technique. The merging of triangles starts at the boundary; when a front closes, the algorithm attempts to maintain an even number of triangles on any sub-front. Again, this approach is sub-optimal because the result depends on the ordering of elements and on the choice of the initial front.

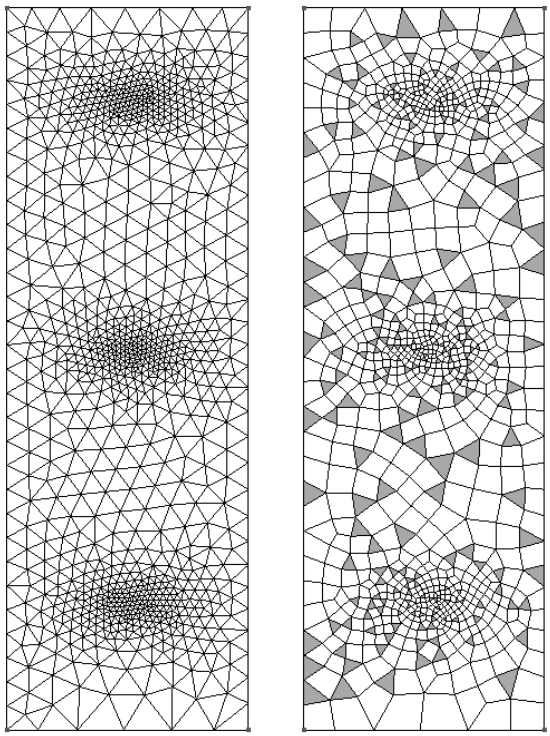


Figure 1. Illustration of the quad-dominant algorithm. Left mesh is the initial triangular mesh and right mesh is the quad dominant mesh, after smoothing (triangles are in grey).

4. The new Blossom-Quad algorithm

Here, our aim is to build a mesh generation scheme that starts with a triangular mesh and attempts to find the set of pairs of triangles that form the best possible quadrilaterals with the constraint of not leaving any remaining triangle in the mesh.

4.1. Blossom: a minimum cost perfect matching algorithm

Let us consider $G(V, E, c)$ an undirected weighted graph. Here, V is the set of n_V vertices, E is the set of n_E undirected edges and $c(E) = \sum c(e_{ij})$ is an edge-based cost function, i.e., the sum of all weights associated to every edge $e_{ij} \in E$ of the graph. A *matching* is a subset $E' \subseteq E$ such that each node of V has at most one incident edge in E'. A matching is said to be perfect if each node of V has exactly one incident edge in E'. As a consequence, a perfect matching contains exactly $n_{E'} = n_V/2$ edges. A perfect matching can therefore only be found for graphs with an even number of vertices. A matching is optimum if $c(E')$ is minimum among all possible perfect matchings.

In 1965, Edmonds [8, 5] invented the *Blossom algorithm* that solves the problem of optimum

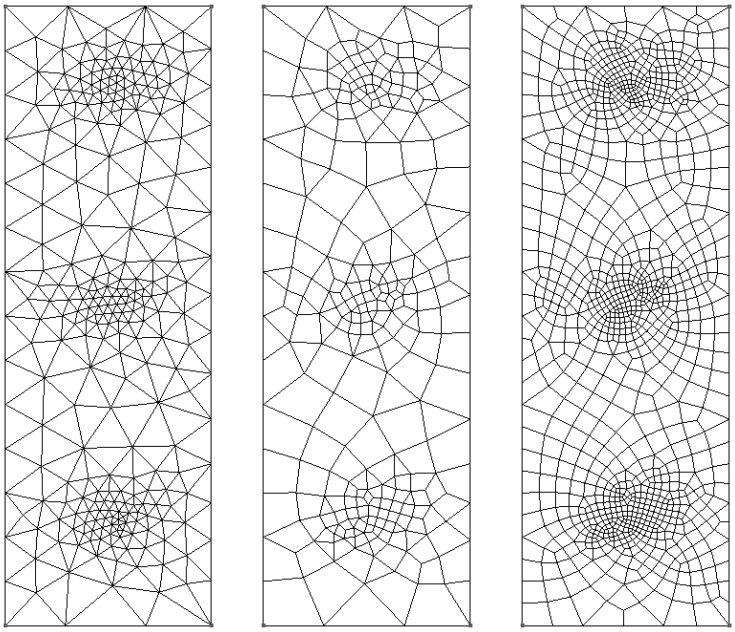


Figure 2. A quad dominant algorithm using $h^{\text{algo}} = 2h$ followed by a one mesh refinement procedure leads to a reduction of the efficiency index τ.

perfect matching in polynomial time. A straightforward implementation of Edmonds's algorithm requires $\mathcal{O}(n_V^2 n_E)$ operations.

Since then, the worst-case complexity of the Blossom algorithm has been steadily improving. Both Lawler [9] and Gabow [10] achieved a running time of $\mathcal{O}(n_V^3)$. Galil, Micali and Gabow [11] improved it to $\mathcal{O}(n_V n_E \log(n_V))$. The current best known result in terms of n_V and n_E is $\mathcal{O}(n_V(n_E + \log n_V))$ [12].

There is also a long history of computer implementations of the Blossom algorithm, starting with the Blossom I code of Edmonds, Johnson and Lockhart [6]. In this paper, our implementation makes use of the Blossom IV code of Cook and Rohe [13]†, which has been considered for several years as the fastest available implementation of the Blossom algorithm.

4.2. Optimal triangle merging

Consider now a mesh made of n_t triangles and n_v vertices. Consider a specific weighted graph $G(V, E, c)$ that is build using triangle adjacencies in the mesh. Here, every vertex of the graph is a triangle t_i of the mesh and every edge of the graph is an internal edge e_{ij} of the mesh that connects two neighboring triangles t_i and t_j. Fig. 3 shows a simple triangular mesh with

†Computer code available at http://www2.isye.gatech.edu/~wcook/blossom4/.

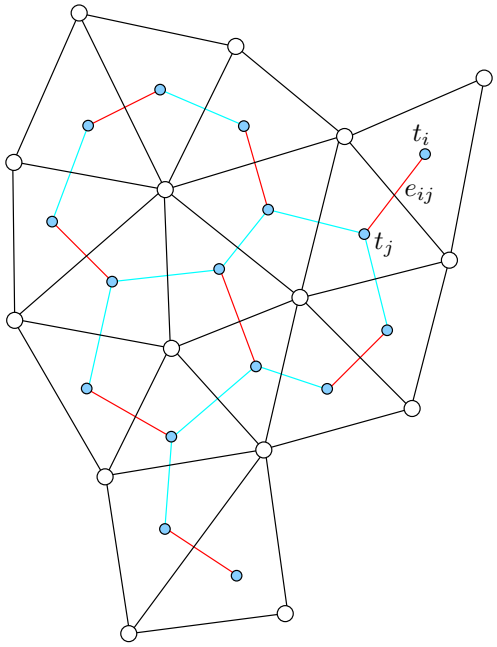


Figure 3. A mesh (in black) and its graph (in cyan and red). The set of graph edges colored in red forms a perfect matching.

its graph and one perfect matching.

Let us come back first to the non-optimal triangle merging algorithms of §3. In term of what has just been defined, the subset E' of edges that have been used for triangle merging in the approach of [3] is a matching that is very rarely a perfect matching. The one of [2] is usually a perfect matching, but not necessarily the optimal one.

Here, we propose a new indirect approach to quadrilateral meshing that takes advantage of the Blossom algorithm of Edmonds. To this end we apply the Blossom IV algorithm to the graph of the mesh. We intend to find the optimum perfect matching with respect to the following total cost function

$$c = \sum_{e \in E'} (1 - \eta(q_{ij})), \tag{4}$$

that is, the sum of all elementary cost functions (or "badnesses") of the quadrilaterals that result in the merging of the edges of the perfect matching E'.

An obvious requirement for the final mesh to be quadrilateral only is that the initial triangular mesh contains an even number of triangles (i.e., an even number of graph vertices). Euler's formula for planar triangulations states that the number of triangles in the mesh is

$$n_t = 2(n_v - 1) - n_v^b, \tag{5}$$

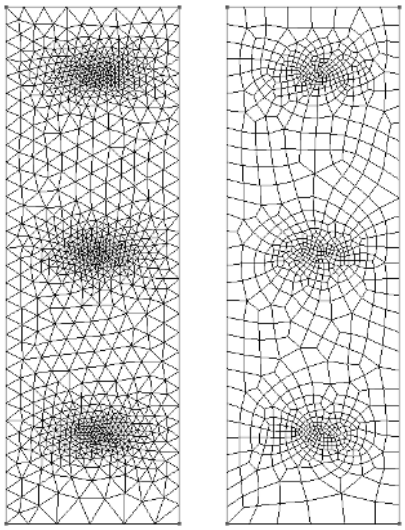
where n_v^b is the number of mesh nodes on its boundary. So, the number of mesh points on the boundary n_v^b should be even. Here our algorithms are applied to general solid models that

have a boundary representation (BRep) [14]. This means that model surfaces are bounded by connected model edges that form edge loops and that the model edges are bounded by model vertices. The mesh vertices of a model edge $n_v^{b_i}$ are defined as the mesh vertices on that edge minus the model vertices. The total number of mesh points on the boundary n_v^b can thus be written as:

$$n_v^b = \sum_{i=0}^{N_E} (1 + n_v^{b_i}). \tag{6}$$

It is then easy to see that for n_v^b to be even, it is sufficient for $n_v^{b_i}$ to be odd. This means that a sufficient condition for having an even number of triangles in the mesh is to have every model edge b_i discretized with an odd number of mesh vertices.

Fig. 4 shows the same illustrative example as Figures 1 and 2 using the Blossom algorithm for recombining the triangles together with the optimization procedure that will be described in §6. The final result is not only much better with respect to the efficiency index (with $\tau = 83\%$), but also with respect to worst element quality ($\eta_w = 0.405$ instead of $\eta_w = 0.310$ for the mesh of Fig. 2). The average quality is better as well.



$\tau = 88.9\%$ $\tau = 83.9\%$

Figure 4. Illustration of the Blossom-Quad algorithm.

4.3. Alternative cost functions

Quadrilateral meshes are not isotropic by definition: this is an important difference between quadrilateral and triangular meshes. In principle, the orientation of the mesh can be defined through a "cross field", i.e., a field of orthogonal tangent vectors to the surface. The cross field can be used to orient the edges of the quadrilateralization [15].

It is possible to define an alternative cost function c that is based on the information contained in the cross field. Consider a graph edge e_{ij} that is tangent to the surface with a unit vector $\vec{e}_{ij}$. Assume that the cross field at the mid point of e_{ij} is defined through two orthogonal unit tangents $\vec{t}_1$ and $\vec{t}_2$. Edges that are aligned with one of the directions of the cross field should not be used for triangle merging. This leads to the the alternative cost function

$$c(e_{ij}) = \frac{1}{1-\sqrt{2}/2}\left(\max(|\vec{e}_{ij}\cdot\vec{t}_1|,|\vec{e}_{ij}\cdot\vec{t}_2|)-\sqrt{2}/2\right). \tag{7}$$

This edge cost function is maximum ($c(e_{ij}) = 1$) for graph edges aligned with one of the directions of the cross field and is minimal ($c(e_{ij}) = 0$) for graph edges that are aligned with the bisector of the cross field. For this minimal value of edge cost function, the mesh edges will then be aligned with the orthogonal unit tangents $\vec{t}_1$ and $\vec{t}_2$. Fig. 5 compares two meshes of

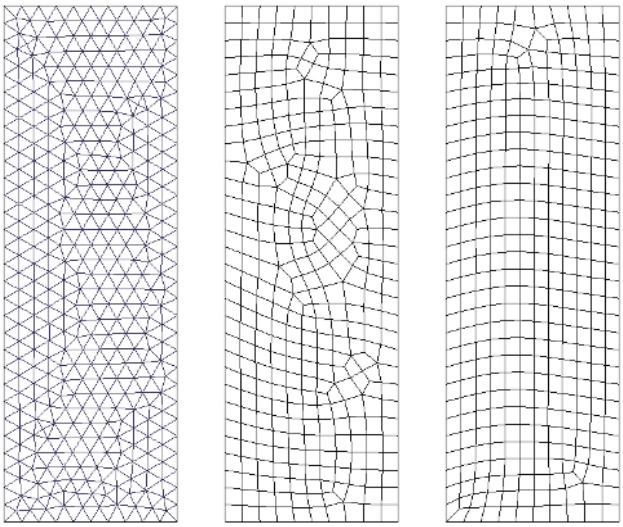


Figure 5. Comparison of meshes generated by Blossom-Quad using cost functions (4) and (7). Initial triangular mesh is on the left. Middle mesh has been generated using (4) and right mesh uses (7).

the rectangle with a uniform size field $h = 0.1$. The first mesh makes use of the cost function (4). The elements of this mesh are mainly oriented with the x an y axes. Yet, some patterns of elements are oriented differently. The use of the alternative cost function (7) with a cross field that is aligned with the coordinate axes allows to have a quasi perfect orientation of the mesh. A mix of both cost functions could also be an alternative.

One can use curvature for aligning the quadrilaterals: it is well known that quadrilateral meshes can be aligned in an optimal way for approximating surfaces, depending on the sign of the two eigenvalues of the local curvature tensor. Again, it is possible to find an objective function that takes that into account.

The issue of choosing both an optimal cost function and a suitable cross field are currently under investigation, and we see in these choices many perspectives for further improvements of the Blossom-Quad algorithm.

5. Existence of perfect matchings

If for some graphs it is possible to find different perfect matchings, there is in general no guarantee that even one single perfect matching exists in a given graph. Consider the meshes of Fig. 6. It is obvious that no perfect matching exists for the coarsest one. The following result, known as Tutte's theorem, proves that none of the two meshes of Fig. 6 contains a perfect matching.

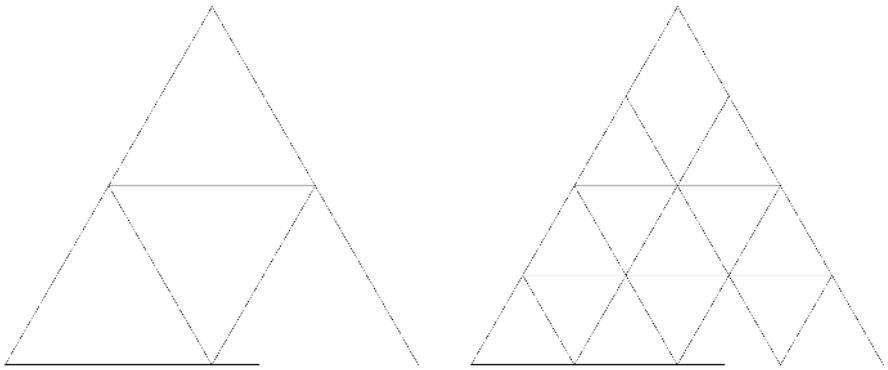


Figure 6. Triangulations that have no perfect matching.

Tutte's theorem : A graph $G = (V, E)$ has no perfect matching if and only if there is a set $S \subseteq V$ whose removal results in more odd-sized components than the cardinality n_S of S, i.e., the number of elements in S [16, 17].

In other words, Tutte's theorem says that there is no perfect matching in a triangulation if and only if it is possible to remove n_S triangles from the mesh and create more that n_S non connected regions that have an odd-sized number of triangles. Let us use Tutte's theorem to prove that none of the two meshes of Fig. 6 has a perfect matching. Let us consider the set S of triangles that have their tip pointed downwards. In the coarsest mesh, $n_S = 1$ and 3 odd-sized components are created, which proves that no perfect matching exists. In the second one, 6 triangles are removed and 10 odd-sized components are created. This simple pattern can be repeated to produce meshes of arbitrary sizes that have no perfect matchings.

The general problem of counting the number of perfect matchings in a general graph is #P-complete[‡]. In other words, there is no hope to find the number of perfect matchings in a general graph. (There is a way to find out, in polynomial time, wether a perfect matching exists by detecting a breakdown in the Blossom algorithm.)

There are however some interesting special cases.

[‡]Sharp p-complete, i.e. much harder than NP-complete.

5.1. Planar Graphs

A graph is said to be planar if it can be drawn in the 2D plane in such a way that its edges intersect only at its vertices. There exists an efficient algorithm (i.e., in polynomial time) that counts perfect matchings in a planar graph. In planar graphs, graph edges form closed non overlapping loops that form the graph faces. Let G be a planar graph. Then G can be oriented efficiently so that each face has an odd number of lines oriented clockwise (this orientation is called a Pfaffian orientation of G) [18]. It can be proven that counting the number of perfect matchings can be done by computing the determinant of the Kasteleyn matrix K:

$$(\#\text{ of perfect matchings of } G)^2 = \det(K), \tag{8}$$

where the Kasteleyn matrix $K(G)$ is an adjacency matrix defined as follows. Consider an edge e_{ij}. If e_{ij} is oriented positively, then $K_{ij} = 1$ and $K_{ji} = -1$. If e_{ij} is oriented negatively, then $K_{ij} = -1$ and $K_{ji} = 1$. If no edge exists between i and j, then $K_{ij} = K_{ji} = 0$.

Here, the computation of the determinant can be done in polynomial time so that it is quite easy to count matchings in a triangular mesh. It is therefore possible to compute wether a perfect matching exists in any planar graph. Yet, finding out that no perfect matching exists does not help us a lot at this point. Moreover, the mesh of a whole torus does not lead to a planar graph[§].

5.2. Cubic Graphs

Cubic graphs, also called trivalent graphs, are graphs for which every node has exactly 3 adjacent nodes. Every cubic graph has at least one perfect matching [19]. It can be proven that the number of perfect matchings in a cubic graph grows exponentially with n_V.

In a finite element triangulation, most of the triangles of the mesh have three neighbors. Only the triangles that are on the boundary of the domain have less than three neighbors. Thus, in general, a finite element mesh is close to be trivalent. We then expect intuitively that perfect matchings will exist in most finite element triangulations. Even though most of the triangulations that we have tried have a perfect matching, the Blossom algorithm has encountered a break down for some of the meshes we have tried.

Since cubic graphs always have many perfect matchings, we propose in the Blossom-Quad algorithm to add some extra-edges to the graph with the aim of creating a graph topology that is close to be trivalent, and thus increase the chance of finding perfect matchings.

5.3. Extra-edges

In our approach, we propose to add edges (that we call "extra-edges" in what follows) in the graph of the triangulation in a way that maximizes the chance of existence of a perfect matching. Consider two successive mesh edges on the boundary of the domain. If those edges are like e_1 and e_2 (Fig. 7), their neighboring triangles are not connected in the graph. At this point, we add an extra-edge for every pair of triangles that are adjacent to those edges that are successive in the boundary of the domain. Those new connexions are represented as dotted cyan lines in Fig. 7.

[§]The mesh of a complete sphere is planar, even though it is not intuitive.

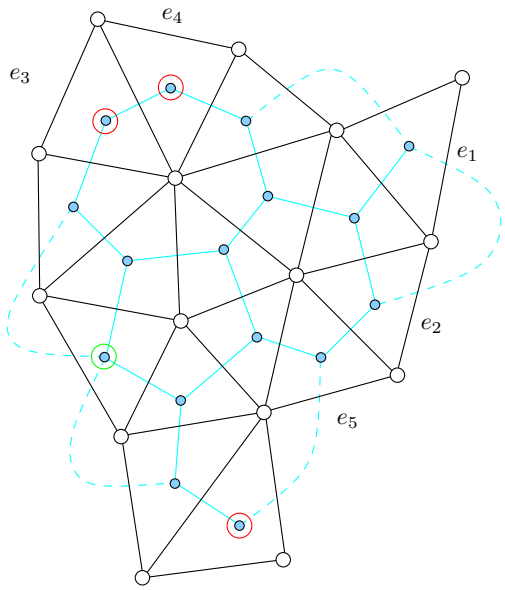


Figure 7. A mesh (in black) and its graph (in cyan). Dotted cyan lines are the extra-graph edges that have been added in order to ensure that the graph has perfect matchings.

Some of those successive mesh edges correspond to triangles that are already connected, like e_3 and e_4, or e_2 and e_5 in Fig. 7. This implies that some of the graph nodes have two neighbors; those ones are rounded in red in Fig. 7. Some others have four neighbors: those are rounded in green in Fig. 7. Yet, there is no node of the graph that has only one adjacent node.

We have not been able to prove that the graph with those extra-edges has always a perfect matching. Yet, in our experience, the addition of those extra-edges allows to find a perfect matching in every example that we have tried.

Technically, we assign a high value (typically 1000) to the "badness" of those extra-edges so that an extra edge belongs to the optimal perfect matching only if there exists no perfect matching in the original graph. We propose two manners of post-processing those extra-edges when they belong to the perfect matching.

5.3.1. Edge swap. The first algorithm is applied essentially when the extra-edge connects two triangles t_1 and t_3 that surround one single triangle t_3 (see Fig. 8). Edges that are colored in red in the graph belong to the perfect matching. If e_{13} belongs to the matching, then edge e_{24} belongs to the matching as well. It is therefore possible to swap the mesh edge that connects t_2 and t_4 and build an all-quad configuration. Note that the concave quadrangle that has been created will be removed through topological optimization (see §6).

5.3.2. Vertex duplication. The second algorithm consists in duplicating the boundary vertex (see Fig. 9). This algorithm is applied when the two triangles that are connected by the extra-edge are surrounded by more than one quad.

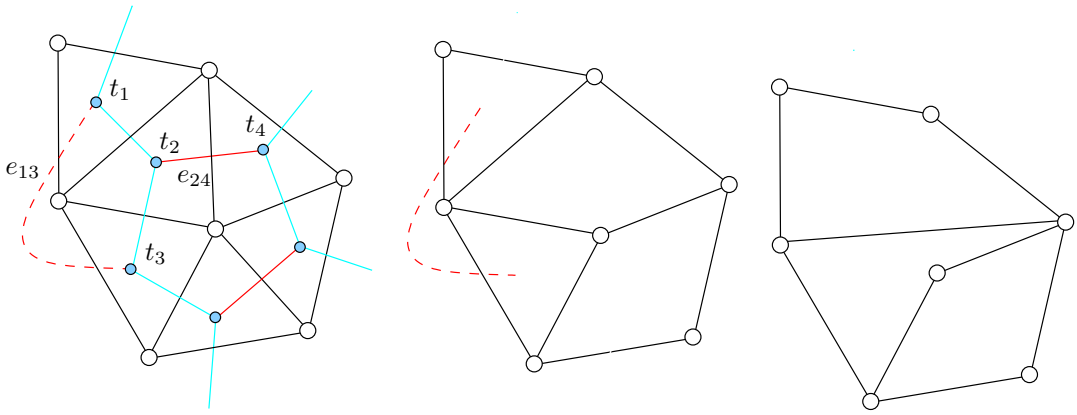


Figure 8. Edge swap algorithm for building an all-quad mesh when an extra edge such as e_{13} belongs to the matching. Example for a configuration with two triangles and two quads.

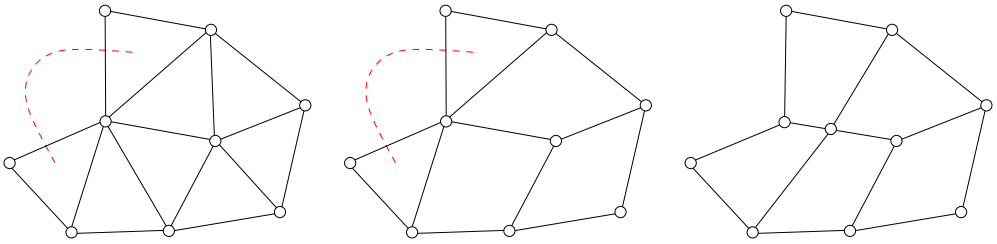


Figure 9. Vertex duplication algorithm for building an all-quad mesh when an extra-edge is in the matching.

In the next paragraph, we will show how to optimize the quality of the quadrangles of those all-quad meshes using local mesh modifications.

6. Optimization

In order to enhance the quality of the final mesh, we first apply a standard vertex smoothing procedure [20] to the nodes of the mesh, taking into account the gradation of the size field. Next, we apply two topological optimization operators specifically tailored for quadrilateral meshes.

The topological optimization operators are local deletion operators: a quad-vertex-merge (see Fig. 10) and the doublet collapse (see Fig. 11) operation [21]. Those operators allow to remove local mesh structures that have a bad topology. More precisely, the quad-vertex-merge operator replaces two mesh nodes that have 3 quadrilateral neighbors by one mesh node with 4 neighbors and the doublet collapse removes a vertex that has two neighbors.

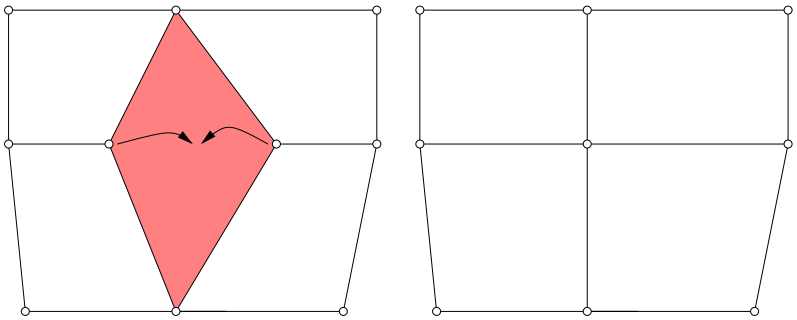


Figure 10. Illustration of quad-vertex-merge optimization operation.

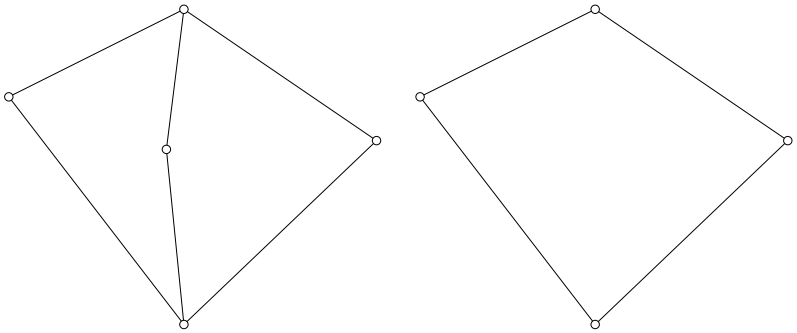


Figure 11. Illustration of doublet collapse optimization operation.

7. The Blossom-Quad algorithm

In this section, we summarize the different steps of the new Blossom-Quad algorithm. This algorithm has been implemented in the open source mesh generator Gmsh [14][¶] and examples of how to use it can be found on the Gmsh wiki[‖].

1. Starting from a solid model with a BRep representation, mesh every model edge b_i with an odd number of mesh vertices $n_v^{b_i}$ (6). Then mesh the model faces with any 2D triangulation algorithm. According to the Euler equation (5), there will then be an even number of mesh triangles.

2. From the produced mesh, build a weighted graph $G(V, E, c)$ (see Fig. 3) where the cost function associated to each graph edge $c(e_{ij})$ is given either by (4) or by (7). This weighted graph has then an even number of graph nodes which is a necessary condition

[¶] http://geuz.org/gmsh/
[‖] https://geuz.org/trac/gmsh/wiki (username: gmsh and password: gmsh)

for a perfect matching to exist.

3. Enrich the graph with extra-edges such as explained in §5.3. Those extra edges are given a very high value of cost function: $c(e_{ij}) = 1000$.
4. Run the Blossom algorithm to find the perfect matching for the given graph.
5. If the perfect matching contains no extra-edges, go to step 6. If it contains some, apply the edge swap algorithm and the vertex duplication algorithm (see §5.3).
6. Optimize the resulting all-quad mesh as explained in §6.

Fig. 12 shows the global Blossom-Quad procedure applied to an initial triangular mesh.

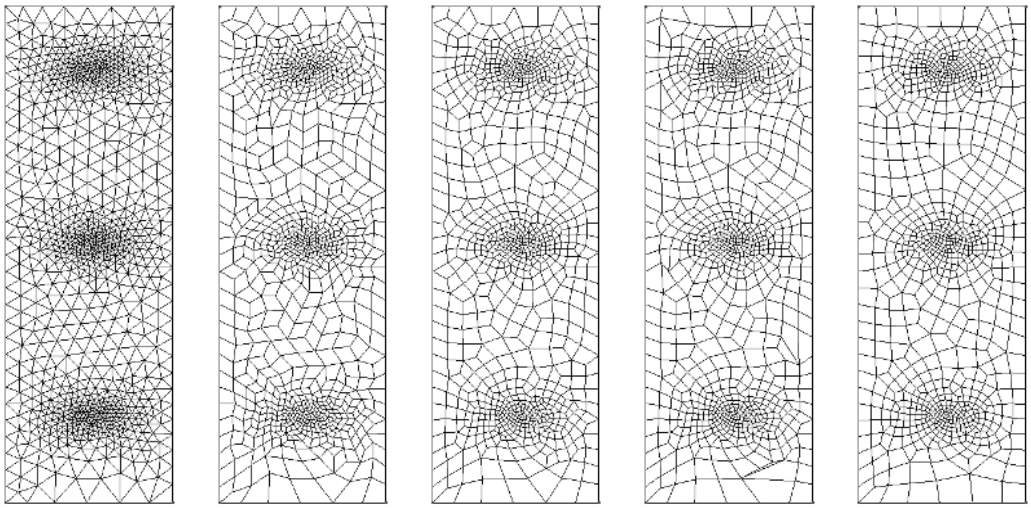


Figure 12. Illustration of the whole Blossom-Quad algorithm.

8. Examples

In this section we present the results obtained by applying the new Blossom-Quad algorithm in different contexts. First, we present meshes of simple planar geometries using analytical mesh size fields. Then, we present quadrilateral meshes generated over complex solid models, defined either by a CAD** or an STL†† triangulation. Finally, we show a complex quadrilateral mesh used for multiscale ocean modeling.

**Computer Aided Design
††stereolithography

8.1. Planar quadrilateral meshes with analytical isotropic size fields

This test case has been proposed by [3]. The domain is a unit square with a circular hole of radius 0.15 centered at $(0.75, 0.75)$. The mesh size field is taken to have a value of $h(x, y) = 0.003$ along the medial axis of the domain and to have a linear grow-up from the medial axis to the interior of the domain.

The uniform refinement step that is applied in the recombination algorithm of [3] has two consequences. On the one hand, it naturally creates a mesh that has a better connectivity. On the other hand, it reduces the efficiency index τ of the mesh. As the recombination algorithm of [3] includes a mesh refinement step, we propose here to generate three meshes with the Blossom-Quad algorithm. The first one applies Blossom to a triangular mesh of size h. The second one applies the Blossom algorithm to a mesh of size $2h$, with one subsequent uniform refinement. The last one applies the Blossom algorithm to a mesh of size $4h$, with two uniform refinements.

Fig. 13 compares the quad-mesh obtained by [3] and the three meshes obtained with the presented Blossom-Quad algorithm. For the problem with size h, the Blossom-Quad algorithm takes 1.38 s on a MacBook Pro clocked at 2.66 GHz. The new algorithm provides meshes that are of better quality close to the boundaries of the domain. The new algorithm also provides a smoother gradation of the mesh. This is due to the fact that no mesh refinement phase is required in the new algorithm.

Table I compares the quality of the four meshes. We also present some statistics about the degree of the vertices d_i, i.e., the number of quads surrounded by a vertex. The new Blossom-Quad algorithm always produces meshes with better element qualities. The application of the Blossom-Quad procedure to the mesh of size h is the best in terms of the efficiency τ. Yet, it has less nodes with the optimal topology, i.e., with 4 neighbors. Applying the Blossom-Quad to a mesh of size $2h$ seems to be, at least for this test case, a good compromise. The efficiency is still acceptable and the quality of the mesh is optimum, both in term of topology and element quality. Using an initial triangular mesh of size $4h$ does not seem to be a good option, especially in term of the efficiency τ.

Algorithm	Mesh size	Quad quality		Degree vertices			Efficiency
	h^{algo}	η_w	$\bar{\eta}$	$d_4(\%)$	d_{min}	d_{max}	τ
Borouchaki [3]	2h	0.30	0.73	91	3	6	–
Blossom-Quad	h	0.32	0.77	72	3	6	85.6%
Blossom-Quad	2h	0.39	0.85	94	3	6	82.3%
Blossom-Quad	4h	0.31	0.87	98	3	7	75.4%

Table I. Quality of the quad meshes for the test case with the medial-axis based mesh size field h. The algorithms are run with an initial mesh size h^{algo} that is subsequently refined to reach the prescribed mesh size field h. We present values for the minimum quality η_w, mean quality $\bar{\eta}$, percentage of vertices of degree 4, minimal and maximal values for the degree of vertices, and efficiency index τ.

8.2. Quadrilateral mesh generation applied to STL models

In this section, we present quad meshes for complex solid models represented only by a triangulation in STL format.

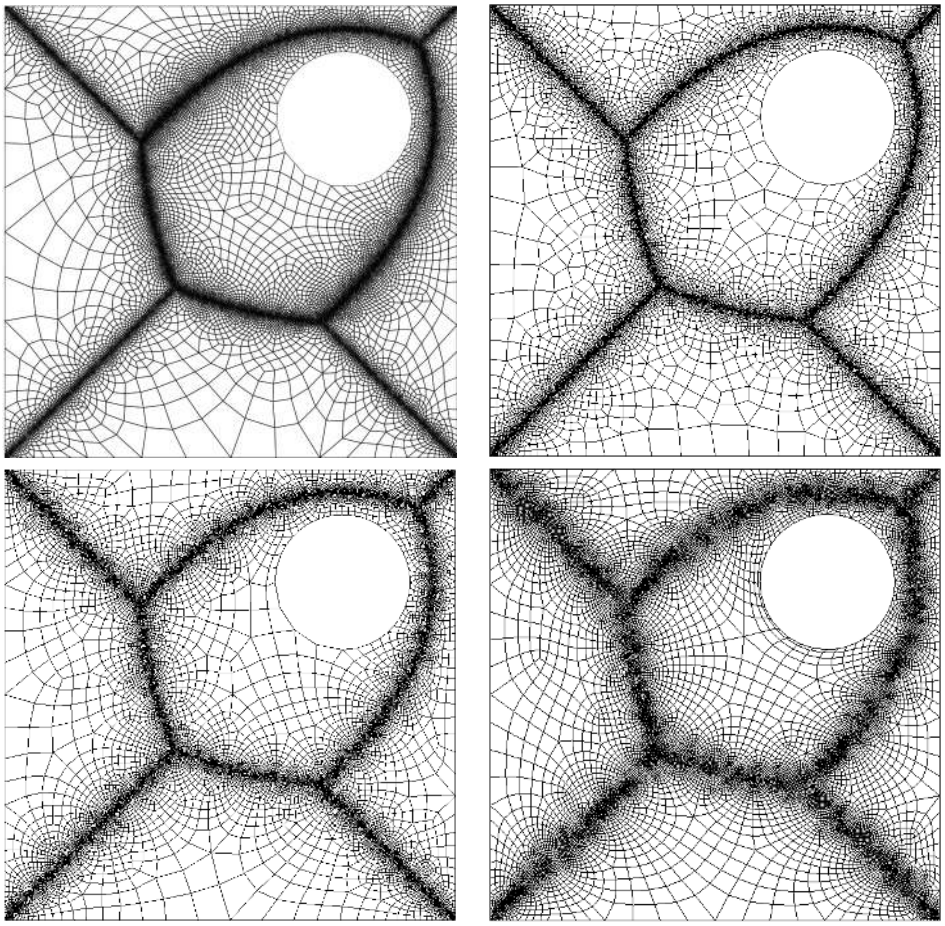


Figure 13. Comparison of both the quadrilateral mesh generation algorithm of [3] (top,left) and the Blossom-Quad algorithm (other meshes). The top-right mesh has been done using Blossom on a triangular mesh of size h. The bottom-left mesh has been generated with the Blossom algorithm applied to a triangular mesh of size $2h$, with one subsequent uniform refinement. The last mesh uses the Blossom algorithm on a mesh of size $4h$, with two uniform refinements.

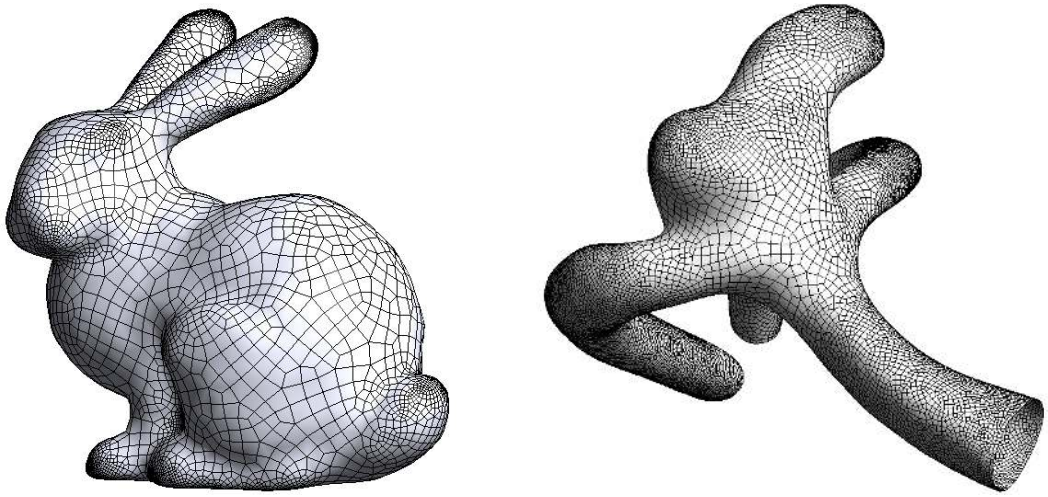
The first triangulation represents the solid model of the Stanford bunny model and the second represents a cerebral aneurysm[‡‡]. The latter triangulation is the output of an image segmentation procedure performed from medical data (CT-scan). For the quad-remeshing procedure, we first compute an automatic triangular remeshing procedure based on a conformal parametrization as described in [22] and then run the presented Blossom-Quad algorithm.

[‡‡]The STL triangulation of the aneurysm can be found on the INRIA web site http://www-roc.inria.fr/gamma/gamma/download.

Fig. 14 shows two curvature-adapted remeshed STL surfaces and Fig. 15 presents the quality histograms of those meshes. The curvature-adapted meshes are computed by defining the mesh size $h(\vec{x})$ as follows:

$$h(\vec{x}) = \frac{2\pi R(\vec{x})}{N_p}, \quad \text{with } R(\vec{x}) = \frac{1}{\bar{\kappa}(\vec{x})} \tag{9}$$

where $\bar{\kappa}(\vec{x})$ is the mean curvature that is computed from the initial nodes of the STL triangulation with the algebraic point set surface method (based on the local fitting of algebraic spheres [23]) and N_p is the number of points chosen for the radius of curvature ($N_p = 50$) .



$\tau = 84.3\%$ $\tau = 85.6\%$

Figure 14. Isotropic mean-curvature based quad-meshes of 13,000 elements for the stl bunny model (left) and 17,000 elements for the aneurysm model (right).

The overall remeshing procedure for both STL examples takes $21s$: $16s$ for the automatic triangular remeshing procedure and only $5s$ for the blossom-Quad algorithm. The remeshing was performed on a MacBook Pro clocked at 2.66 GHz. This quad-remeshing procedure is extremely fast compared to the quad-dominant meshes of Levy et al. [24] for which the remeshing of the Stanford bunny takes $271s$.

8.3. Quadrilateral mesh generation applied to parametric CAD models

We consider the solid model of a human bone. The model contains 11 NURBS surfaces. The mesh size field is based on surface curvature: we use formula (9) with $N_p = 35$. The total time for surface meshing was 26 seconds for generating 10,134 quadrilateral elements. The time for computing all 11 perfect matchings and doing optimization does not exceed 10 seconds. The average quality of the finite element mesh is $\bar{\eta} = 73.3\%$ and the efficiency is $\tau = 80.2\%$. The model still contains 1 element of bad quality, with $\eta = 0.01$ (see Fig. 16).

Gmsh allows direct access to the CAD model, allowing to compute exactly the principal directions of curvature together with minimal and maximal curvature. This allows to define an

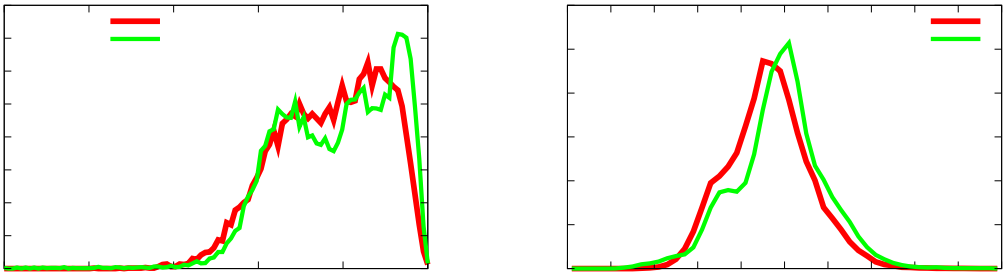


Figure 15. Quality histograms (element quality η (3) and normalized edge length l^h (1)) for the Blossom-Quad remeshing of the bunny and the aneurysm models.

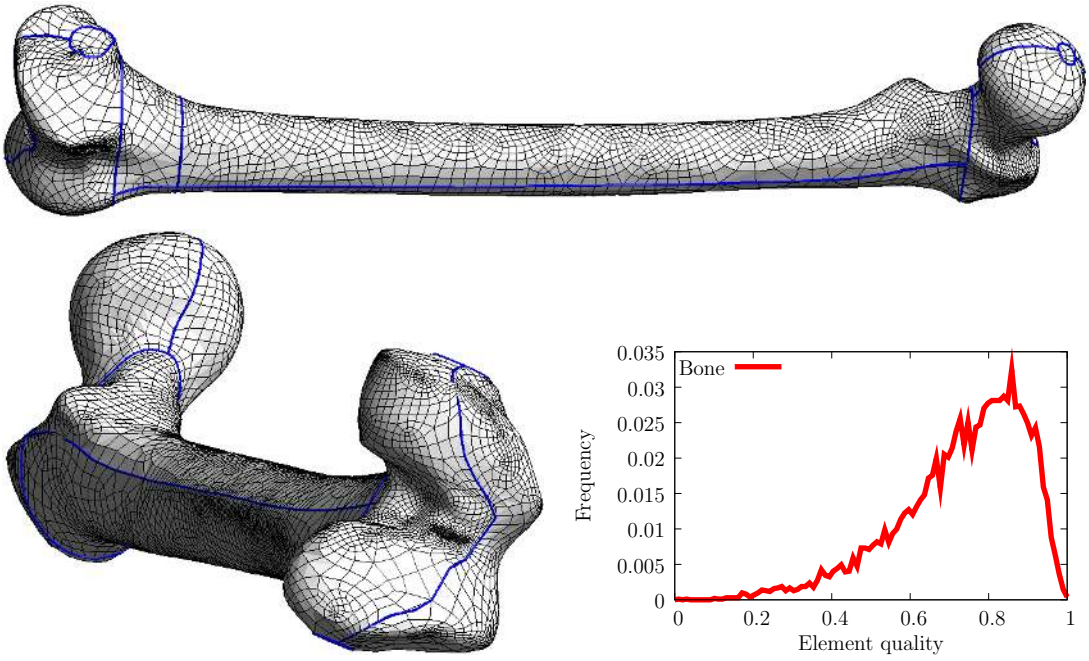


Figure 16. Isotropic quad mesh of the bone geometry with a quality plot.

anisotropic mesh size field [7]. Here, we use formula (9) with $N_p = 35$ in each of the directions of principal curvature. An anisotropic triangular mesh is initially built using the anisotropic metric provided by the curvature. Then, the Blossom-Quad algorithm is applied as it. The resulting mesh is presented in Fig. 17. The quadrilateral mesh naturally aligns itself to the principal directions of curvature, allowing to build an anisotropic quadrilateral mesh without

effort.

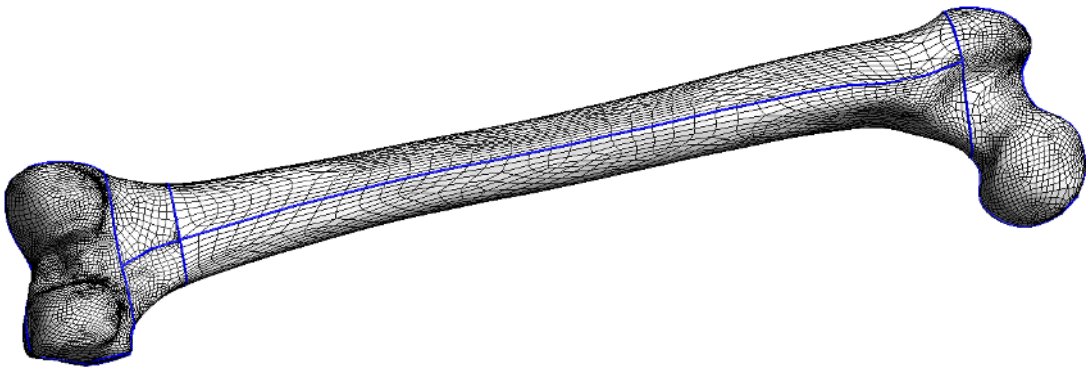


Figure 17. Anisotropic mesh of the bone geometry.

8.4. Quadrilateral meshes for ocean modeling

Our research team has developed the first multiscale hydrodynamic model of the whole Great Barrier Reef. The Great Barrier Reef is on the continental shelf of the Australian northeastern coastline and contains over 2500 coral reefs in a strip that is about 2600 km in length and 200 km in width. The mesh size field $h(\vec{x})$ is defined as a function of the bathymetry and the distance to the shore.

We have performed 24 hours of simulations of the water circulation on the Great Barrier Reef shelf. The physical model is described in [25]. The equations are descretized with P_1^{DG} discontinuous finite elements combined with a second order multirate explicit Runge-Kutta temporal integrator as in [26]. A plot of the velocity vectors and the sea surface elevation is presented on Fig. 18. Tidal jets and eddies due to the interaction of the flow with the topography near the open-sea boundary are clearly visible. The same simulation on a triangular mesh ($\simeq$ 250,000 triangles) was 2.7 times slower than on the corresponding Blossom-Quad quadrilateral mesh ($\simeq$ 119,000 quads).

9. Conclusions

The main contribution of this paper is to have introduced a well known result of graph theory (the Blossom algorithm for minimum cost perfect matching) in the domain of unstructured quadrilateral mesh generation. We have presented a new algorithm—dubbed Blossom-Quad—that takes advantage of this result to produce high-quality quadrilateral meshes in a robust and efficient manner, and we have applied it in different contexts: from planar geometries to parametric CAD models to STL remeshing to multiscale ocean models.

Possible further improvements to the proposed algorithm are numerous. For example, the cost function that has been used could be modified in order to align the quadrilateral mesh with some preferred directions. (We have already presented a simple example to that effect in

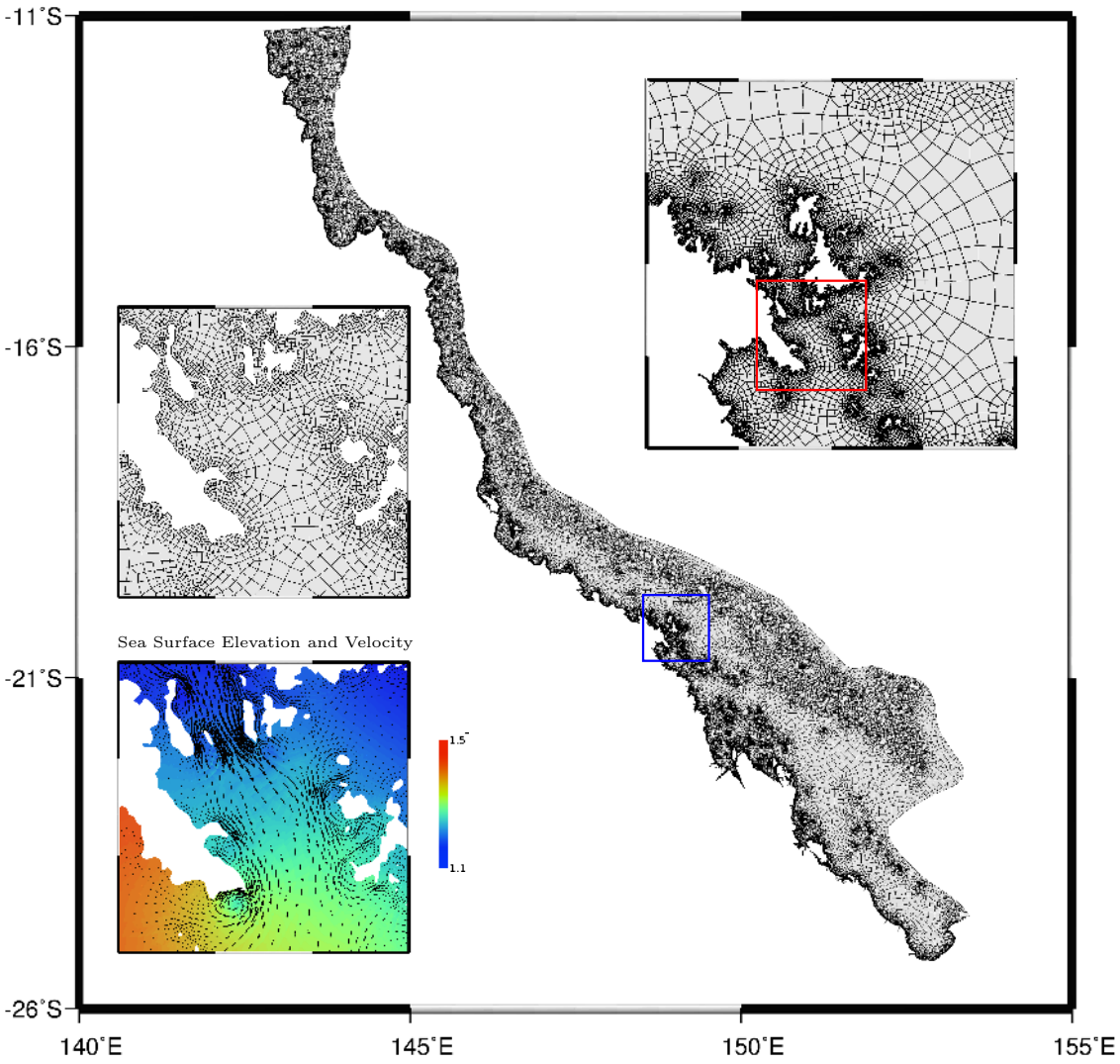


Figure 18. Quadrilateral mesh of the Great Barrier Reef. Two successive zooms of the Withsundays Inslands Archipelago. Sea surface elevation (*color levels*) and bidimensional velocity field (*arrows*).

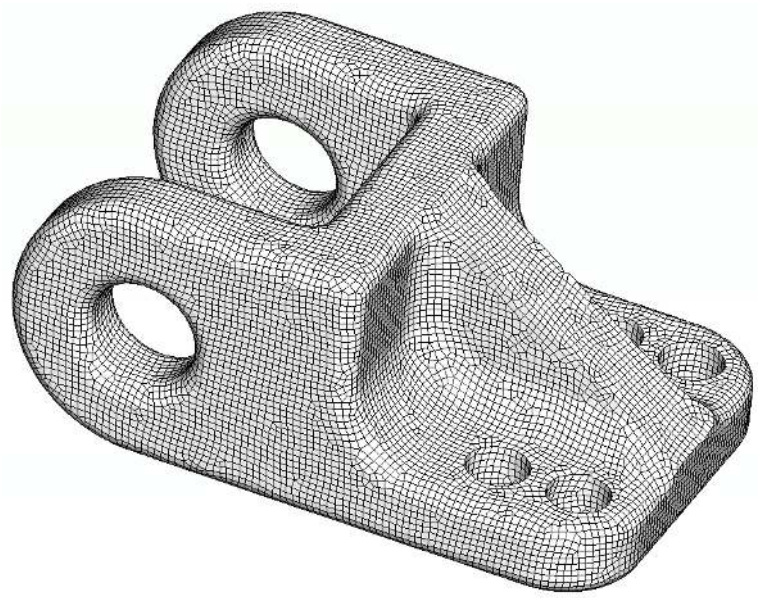


Figure 19. Blossom-Quad algorithm applied to a triangular surface mesh that has been smoothed using the Lp centroidal Voronoï tesselation technique of [24].

§4.3.) Also, in this paper we have used as input to the recombination procedure triangulations with vertices distributed fairly uniformly. Recent developments [24, 27] allow to align those vertices with some prescribed directions. Coupling both approaches could potentially lead to even higher quality quadrilateral meshes. As an example, we have applied the Blossom-Quad algorithm (without optimization) to a triangular surface mesh that has been smoothed using the Lp centroidal Voronoï tesselation technique of [24]. The resulting mesh is presented on Fig. 19. There, quadrangles do not form patches that are randomly aligned but follow a regular pattern.

In the longer run, the very challenging problem of automatic generation of hexahedral-dominant meshes could be approached using an indirect technique of this kind. In this case, more than two tetrahedra have to be merged in order to form one hexahedron. Here again, we think that graph theory could maybe help us in finding some kind of optimal matching.

Acknowledgements

This work has been partially supported by the Belgian Walloon Region under WIST grants ONELAB 1017086 and DOMHEX 1017074.

Authors gratefully thanks F. Glineur and J. Hendricks from the Applied Math. Department of the Université catholique de Louvain for the discussions and hints about graph theory.

Authors also acknowledge P. Frey of Université Paris VI for authorizing us to include one of the illustrations of his paper [3] (Fig. 13, top left).

Authors finally deeply acknowledge B. Levy of LORIA Nancy for providing us with the triangulation and the smoothing algorithm used to produce Fig. 19.

REFERENCES

1. Blacker TD, Stephenson MB. Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1991; **32**:811–847.
2. Lee CK, Lo SH. A new scheme for the generation of a graded quadrilateral mesh. *Computers and Structures* 1994; **52**:847–857.
3. Borouchaki H, Frey P. Adaptive triangular–quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1998; **45**(5):915–934.
4. Owen SJ, Staten ML, Canann SA, Saigal S. Q-morph: An indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering* 1999; **9**:1317–1340.
5. Edmonds J. Maximum matching and a polyhedron with 0-1 vertices. *J. of Research at the National Bureau of Standards* 1965; **69B**(125–130).
6. Edmonds J, Johnson EL, Lockhart SC. Blossom I: A computer code for the matching problem. IBM T. J. Watson J. Edmonds, E. L. Johnson, and S. C. Lockhart. IBM T. IBM T.J. Watson Research Center, Yorktown Heights, New York 1969.
7. Frey P, George PL. *Mesh Generation - Application To Finite Elements*. Wiley, 2008.
8. Edmonds J. Paths, trees, and flowers. *Canad. J. Math* 1965; **17**:449–467.
9. Lawler EL. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, and Winston, New York, NY, 1976.
10. Gabow H. Implementation of algorithms for maximum matching on nonbipartite graphs. PhD Thesis, Stanford University 1973.
11. Gabow H, Galil Z, Micali S. An o(ev log v) algorithm for finding a maximal weighted matching in general graphs. *SIAM J. Computing,* 1986; **15**(120–130).
12. Gabow HN. Data structures for weighted matching and nearest common ancestors with linking. *In Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms,*, 434-443 (ed.), 1990.
13. Cook W, Rohe A. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing,* 1999; **11(2)**(138–148).
14. Geuzaine C, Remacle JF. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering* 2009; **79**(11):1309–1331.
15. Bommes D, Zimmer H, Kobbelt L. Mixed-integer quadrangulation. *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, ACM: New York, NY, USA, 2009; 1–10, doi:http://doi.acm.org/10.1145/1576246.1531383.
16. Tutte WT. A family of cubical graphs. *Proc. Cambridge Philos. Soc.* 1947; **43**:459–474.
17. Pemmaraju S, Skiena S. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica* ®. Cambridge University Press: New York, NY, USA, 2003.
18. Kasteleyn PW. Dimer Statistics and Phase Transitions. *Journal of Mathematical Physics* Feb 1963; **4**:287–293, doi:10.1063/1.1703953.
19. Oum S. Perfect Matchings in Claw-free Cubic Graphs. *ArXiv e-prints* Jun 2009; .
20. Sarrate J, Huerta A. An improved algorithm to smooth graded quadrilateral meshes preserving the prescribed element size. *Communications in numerical methods in engineering* 2001; **17**(2):89–99.
21. Daniels J II, Silva CT, Cohen E. Localized quadrilateral coarsening. *SGP '09: Proceedings of the Symposium on Geometry Processing*, Eurographics Association: Aire-la-Ville, Switzerland, Switzerland, 2009; 1437–1444.
22. Marchandise E, de Wiart C, Vos W, Geuzaine C, Remacle J. High-quality surface remeshing using harmonic maps–part ii: Surfaces with high genus and of large aspect ratio. *International Journal for Numerical Methods in Engineering* 2011; **86**:1303–1321.
23. Guennebaud G, Germann M, Gross M. Dynamic sampling and rendering of algebraic point set surfaces. *Computer Graphics Forum* 2008; **27**:653–662.
24. Lévy B, Liu Y. Lp centroidal voronoi tesselation and its applications. *ACM Transactions on Graphics (SIGGRAPH conference proceedings)*, 2010.
25. Lambrechts J, Hanert E, Deleersnijder E, Bernard PE, Legat V, Wolanski JFRE. A high-resolution model of the whole great barrier reef hydrodynamics. *Estuarine, Coastal and Shelf Science* 2008; **79(1)**:143–151. Doi 10.1016/j.ecss.2008.03.016.
26. Constantinescu EM, Sandu A. Multirate timestepping methods for hyperbolic conservation laws. *Journal of Scientific Computing* 2007; **33**:239–278, doi:10.1007/s10915-007-9151-y.

27. Hausner A. Simulating decorative mosaics. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 2001; 573–580.