
L_p Lloyd's Energy Minimization for Quadrilateral Surface Mesh Generation

Tristan Carrier Baudouin¹, Jean-François Remacle¹, Emilie Marchandise¹,
and Jonathan Lambrechts¹

Institute of Mechanics, Materials and Civil Engineering, Université catholique de
Louvain, Avenue Georges-Lemaître 4, 1348 Louvain-la-Neuve, Belgium
{tristan.carrier,jean-francois.remacle,emilie.marchandise,
jonathan.lambrechts}@uclouvain.be

Summary. Indirect methods recombine the elements of triangular meshes to produce quadrilaterals. The resulting quadrilaterals are usually randomly oriented, which is not desirable. However, by aligning the vertices of the initial triangular mesh, precisely oriented quads can be produced. Levy's algorithm is a non-linear optimization procedure that can align points according to a locally defined metric. It minimizes an energy functional based on the L_p distance in the local metric. The triangulation of a set of vertices smoothed with Levy's algorithm is mainly composed of right-angled triangles, which is ideal for quad recombination. An implementation of Levy's algorithm for the purpose of finite element computation has been developed. The implementation can create quads of desired size and orientation. The algorithm has been tested on two-dimensional geometries as well as parametrized curved surfaces. The results show an improvement of the quads alignment.

Keywords: centroidal Voronoi tessellations, non-linear optimization, quad mesh generation.

1 Introduction

For finite element analysis, quad meshes are advantageous compared to triangular meshes [1]. For example, in computational fluid mechanics, they accelerate grid convergence [2][3][4] and they capture boundary layers with a higher precision [5]. They are also very useful in structural mechanics. They are not subject to numerical locking [6] and they allow schemes to remain stable under inexact integration [7][8][9]. In the context of high order methods, quad meshes can be curved more robustly [10].

However, quad mesh generation techniques are not as mature as triangular ones. The indirect approach looks like a promising solution. It consists of combining two by two the elements of a triangular mesh in order to create quads. Triangular mesh generators are usually designed to produce near-equilateral triangles. Combining these triangles yields randomly oriented quads, which

is not ideal. However, if the vertices of the initial triangular mesh are well aligned, an indirect algorithm like Blossom-Quad [11] can create high quality, precisely oriented quads. Blossom-Quad optimizes the mean quality of the resulting quads. Unlike triangles, quads are always oriented. It is desirable to prescribe this orientation.

Levy and Liu developed a method based on centroidal Voronoi tessellations in L_p norm for aligning points in a rectangular manner [12]. The method minimizes an energy functional equal to the sum of the L_p moment of inertia of the Voronoi cells. The L_p distances are measured with respect to a locally defined metric. This metric field controls the orientation of the quads. Levy's algorithm uses the limited-memory Broyden-Fletcher-Goldfarb-Shanno optimization procedure (LBFGS) to minimize the energy functional. LBFGS has the advantage of only requiring the value of the functional and its gradient. Low energy solutions are sets of points having rectangular Voronoi cells. When the Voronoi cells are rectangular, the points are also aligned in a rectangular way. In other words, Levy's algorithm optimizes the shape of the Voronoi cells. This process is illustrated on Fig. 1 and 2. Fig. 1(a) is the initial triangular mesh and Fig. 1(b) is the Voronoi diagram of the vertices. Fig. 2(a) is the final quad mesh obtained with Levy's and Blossom-Quad algorithms. Fig. 2(b) is the corresponding Voronoi diagram.

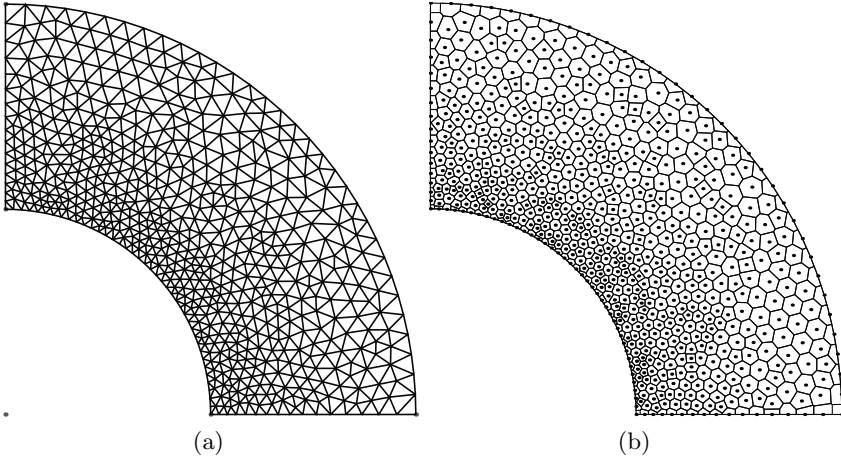


Fig. 1. The initial triangular mesh (a) and its corresponding Voronoi diagram (b).

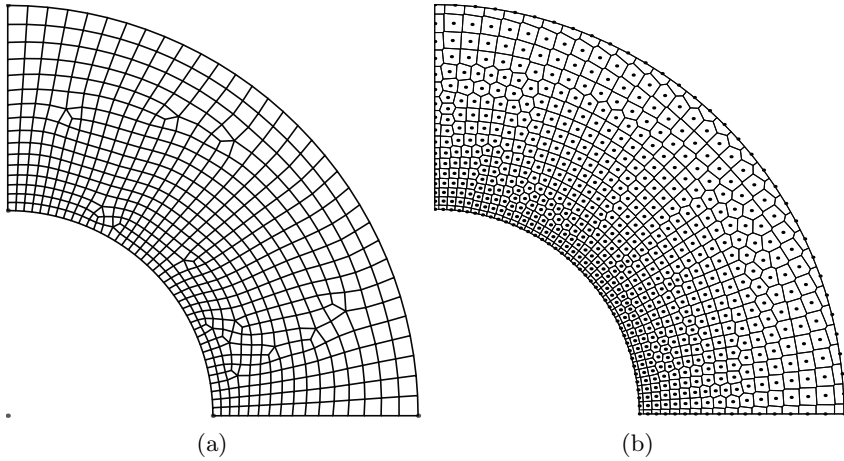


Fig. 2. The final quad mesh (a) and its corresponding Voronoi diagram (b) (A Laplacian smoothing has been applied, which explain the slight differences between the two sets of vertices).

A two-dimensional implementation of Levy's algorithm adapted to the field of finite element computation is presented in this paper. It can take into input metric and density fields. These metric and density fields define the orientation and the size of the quads. By using a parametrization, the algorithm can be applied to curved surfaces. The energy and the gradient are computed with Gauss integration techniques. The Alglib library¹ is used for the LBFGS optimization part.

2 Levy's Algorithm

The implementation of Levy's algorithm described in this article can only find local minimums. It cannot find global minimums even for simple geometries such as squares and rectangles. It can nevertheless align points in a very satisfactory manner.

The clipped Voronoi diagram is the part of the Voronoi diagram that is inside the domain, as shown in Fig. 1(b) and 2(b). It is an essential part of Levy's algorithm [12]. Without it, computing accurate values for the energy and the gradient is impossible. The method used here is inspired from [13].

Subsections 2.1 and 2.2 introduce the energy functional and the gradient. Subsection 2.3 shows how the domain is divided into triangular elements. Subsection 2.4 explains how to compute the energy and the gradient with Gauss integration techniques. The addition of a metric and a density is discussed in subsection 2.5.

¹ ALGLIB (www.alglib.net), Sergey Bochkanov and Vladimir Bystritsky.

The main difference between Levy's implementation and the one described in this article is the way of computing the energy and the gradient. Like it was said before, the present implementation uses Gauss integration techniques, while Levy employs analytical formulas.

2.1 Energy Functional

The energy functional minimized by Levy's algorithm uses the L_p distance defined here [12]:

$$||\mathbf{y} - \mathbf{x}||_p^p = |y_1 - x_1|^p + |y_2 - x_2|^p \tag{1}$$

Each curve on Fig. 3 contains a set of points equidistant to the origin according to its particular L_p distance. The curves become more and more rectangular as p increases. For any p higher than two, the L_p distance become anisotropic.

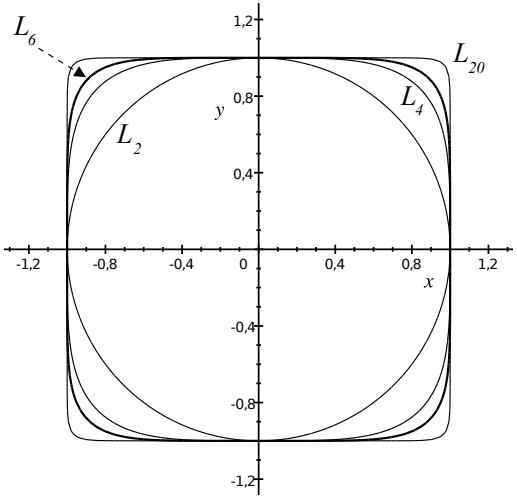


Fig. 3. Unit circles in various L_p distances.

The energy functional is defined as [12]:

$$F(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = \sum_{i=1}^N \underbrace{\int_{R_i} ||\mathbf{y} - \mathbf{x}_i||_p^p d\mathbf{y}}_{I^{R_i}(\mathbf{x}_i)} \tag{2}$$

The index i runs over all mesh vertices $\mathbf{x}_i$. R_i is the domain corresponding to the i -th voronoi cell generated by mesh vertex $\mathbf{x}_i$. Evidently, the Voronoi diagram is computed with the L_2 metric. In what follows, the energy integral will be denoted by $I^{R_i}(\mathbf{x}_i)$.

2.2 Energy Gradient

The energy gradient is the total derivative of F with respect to the position of each non-boundary mesh vertex [12]. Boundary vertices are considered unmovable. It is assumed that $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ are non-boundary vertices and that $\mathbf{x}_{n+1}, \mathbf{x}_{n+2}, \dots, \mathbf{x}_N$ are boundary vertices. The energy gradient is given by:

$$\frac{d}{d\mathbf{x}_k} F(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = \frac{d}{d\mathbf{x}_k} \sum_{i=1}^N I^{R_i}(\mathbf{x}_i) = \sum_{i=1}^N \frac{dI^{R_i}(\mathbf{x}_i)}{d\mathbf{x}_k} \quad k = 1, \dots, n$$

The total derivative on the right hand side can be rewritten in terms of partial derivatives.

$$\frac{d}{d\mathbf{x}_k} F(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = \sum_{i=1}^N \frac{\partial I^{R_i}(\mathbf{x}_i)}{\partial \mathbf{x}_k} + \sum_{i=1}^N \frac{\partial I^{R_i}(\mathbf{x}_i)}{\partial R_i} \frac{dR_i}{d\mathbf{x}_k}$$

The partial derivative of $I^{R_i}(\mathbf{x}_i)$ with respect to $\mathbf{x}_k$ is non-null only when $i = k$.

$$\frac{d}{d\mathbf{x}_k} F(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = \frac{\partial I^{R_k}(\mathbf{x}_k)}{\partial \mathbf{x}_k} + \sum_{i=1}^N \frac{\partial I^{R_i}(\mathbf{x}_i)}{\partial R_i} \frac{dR_i}{d\mathbf{x}_k}$$

R_i can be expressed in terms of the Voronoi vertices $\mathbf{C}_{i1}, \mathbf{C}_{i2}, \dots, \mathbf{C}_{iM_i}$ of the i -th Voronoi cell. All Voronoi cells do not have the same number of vertices M_i . Fig. 4 shows a Voronoi cell with six Voronoi vertices.

The gradient can be rewritten in terms of the position of the Voronoi vertices.

$$\frac{d}{d\mathbf{x}_k} F(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = \frac{\partial I^{R_k}(\mathbf{x}_k)}{\partial \mathbf{x}_k} + \sum_{i=1}^N \sum_{j=1}^{M_i} \frac{\partial I^{R_i}(\mathbf{x}_i)}{\partial \mathbf{C}_{ij}} \frac{d\mathbf{C}_{ij}}{d\mathbf{x}_k} \quad (3)$$

Most of the terms $\frac{d\mathbf{C}_{ij}}{d\mathbf{x}_k}$ are null. They are non-null only when the mesh vertex is very close to the Voronoi vertex, as detailed in the next section.

2.3 Gradient Assembly

In order to evaluate the integrals with Gauss techniques, the Voronoi cells are divided into triangular elements. Each triangular element is composed of a mesh vertex and two successive Voronoi vertices, as shown in Fig. 4. In Fig. 4 to 7, the black dots are assumed to be mesh vertices and the hollow dots are assumed to be Voronoi vertices. The dotted segments are Delaunay edges.

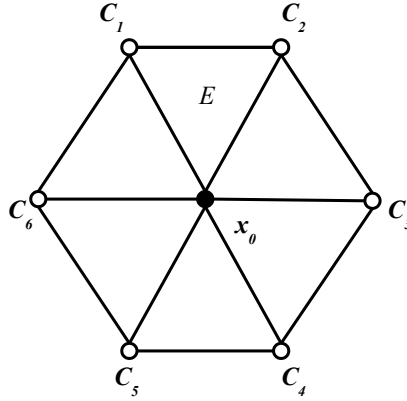


Fig. 4. A Voronoi cell divided into triangular elements.

Most of the terms inside the double summation found in equation (3) vanish. By considering the relationship between the mesh vertices and the Voronoi vertices, the relevant contributions can be identified. Each Voronoi vertex falls into one of three categories.

1. A Voronoi vertex can be the center of the circle circumscribing a Delaunay element. Voronoi vertex C_1 from Fig. 5 belongs to this category. As long as the displacements are infinitesimal, C_1 depends only on x_0 , x_1 and x_2 .

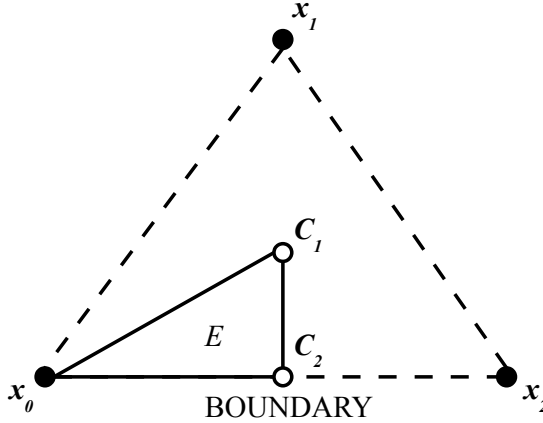


Fig. 7. C_2 is the median point between x_0 and x_2 .

The following formulas were derived from equation (3). They apply to Fig. 5. It is assumed that x_0 , x_1 , x_2 and x_3 are non-boundary mesh vertices. A $+=$ sign is used because there will be contributions from other elements as well.

$$\frac{dF}{dx_0} += \frac{\partial I^E(x_0)}{\partial x_0} + \frac{\partial I^E(x_0)}{\partial C_1} \frac{dC_1}{dx_0} + \frac{\partial I^E(x_0)}{\partial C_2} \frac{dC_2}{dx_0} \quad (4)$$

$$\frac{dF}{dx_1} += \frac{\partial I^E(x_0)}{\partial C_1} \frac{dC_1}{dx_1} \quad (5)$$

$$\frac{dF}{dx_2} += \frac{\partial I^E(x_0)}{\partial C_1} \frac{dC_1}{dx_2} + \frac{\partial I^E(x_0)}{\partial C_2} \frac{dC_2}{dx_2} \quad (6)$$

$$\frac{dF}{dx_3} += \frac{\partial I^E(x_0)}{\partial C_2} \frac{dC_2}{dx_3} \quad (7)$$

2.4 Gauss Integration

The terms inside equations (4) to (7) are integrals on the element E . A linear transformation T will be used in order to go from the reference triangle E' to the triangle E , as shown in Fig. 8. It will then become possible to evaluate the various integrals with Gauss techniques.

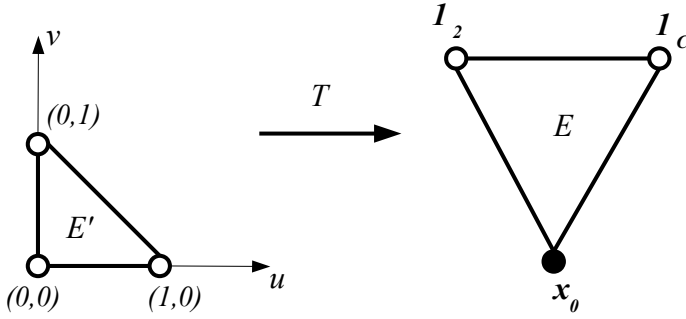


Fig. 8. The linear transformation T .

$$\mathbf{y} = \mathbf{T}(u, v) = \mathbf{x}_0(1 - u - v) + \mathbf{C}_1 u + \mathbf{C}_2 v$$

Equation (2) defines $I^E(\mathbf{x}_0)$ as the energy contribution of element E . J is the Jacobian of the transformation T :

$$I^E(\mathbf{x}_0) = \int_E \|\mathbf{y} - \mathbf{x}_0\|_p^p d\mathbf{y} = \int_{E'} \|\mathbf{T}(u, v) - \mathbf{x}_0\|_p^p J du dv$$

The partial derivative of $I^E(\mathbf{x}_0)$ with respect to the position of mesh vertex $\mathbf{x}_0$ can be written as:

$$\frac{\partial I^E(\mathbf{x}_0)}{\partial \mathbf{x}_0} = \frac{\partial}{\partial \mathbf{x}_0} \int_E \|\mathbf{y} - \mathbf{x}_0\|_p^p d\mathbf{y} = \int_{E'} \frac{\partial \|\mathbf{T}(u, v) - \mathbf{x}_0\|_p^p}{\partial \mathbf{x}_0} J du dv$$

The partial derivative of $I^E(\mathbf{x}_0)$ with respect to the position of Voronoi vertex $\mathbf{C}_1$ is given by:

$$\begin{aligned} \frac{\partial I^E(\mathbf{x}_0)}{\partial \mathbf{C}_1} &= \frac{\partial}{\partial \mathbf{C}_1} \int_E \|\mathbf{y} - \mathbf{x}_0\|_p^p d\mathbf{y} = \frac{\partial}{\partial \mathbf{C}_1} \int_{E'} \|\mathbf{T}(u, v) - \mathbf{x}_0\|_p^p J du dv \\ &= \int_{E'} \frac{\partial \|\mathbf{T}(u, v) - \mathbf{x}_0\|_p^p}{\partial \mathbf{T}(u, v)} \frac{\partial \mathbf{T}(u, v)}{\partial \mathbf{C}_1} J + \|\mathbf{T}(u, v) - \mathbf{x}_0\|_p^p \frac{\partial J}{\partial \mathbf{C}_1} du dv \end{aligned}$$

A Voronoi vertex can sometimes depend on three mesh vertices, as in Fig. 5. The following matrix is the derivative of Voronoi vertex $\mathbf{C}_1$ with respect to mesh vertex $\mathbf{x}_0$ [12]. The derivatives of $\mathbf{C}_1$ with respect to mesh vertices $\mathbf{x}_1$ and $\mathbf{x}_2$ can be obtained by replacing $\mathbf{x}_0$ with $\mathbf{x}_1$ or $\mathbf{x}_2$.

$$\frac{d\mathbf{C}_1}{d\mathbf{x}_0} = \begin{bmatrix} (\mathbf{x}_1 - \mathbf{x}_0)^T \\ (\mathbf{x}_2 - \mathbf{x}_0)^T \end{bmatrix}^{-1} \begin{bmatrix} (\mathbf{C}_1 - \mathbf{x}_0)^T \\ (\mathbf{C}_1 - \mathbf{x}_0)^T \end{bmatrix}$$

A Voronoi vertex can instead depend on two mesh vertices and one boundary line segment, as in Fig. 6. The following matrix needs to be used in this situation [12]. The derivative of $\mathbf{C}_1$ with respect to mesh vertex $\mathbf{x}_1$ can be obtained by replacing $\mathbf{x}_0$ with $\mathbf{x}_1$. $\mathbf{N}$ is the normal vector to the boundary line segment. It can be multiplied by any non-zero constant without affecting the value of the derivative.

$$\frac{d\mathbf{C}_1}{d\mathbf{x}_0} = \begin{bmatrix} (\mathbf{x}_1 - \mathbf{x}_0)^T \\ \mathbf{N}^T \end{bmatrix}^{-1} \begin{bmatrix} (\mathbf{C}_1 - \mathbf{x}_0)^T \\ \mathbf{0} \end{bmatrix}$$

More details about the procedure used to obtain these matrices can be found in Levy’s article [12]. It is important to recall that it is only the derivatives with respect to non-boundary mesh vertices that need to be calculated.

2.5 Non-uniform Metric and Density

The addition of a metric M and a density ρ allows varying orientation and quad sizes. The following energy functional takes into account these two parameters [12][14].

$$F(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N) = \sum_{i=1}^N \int_{R_i} \rho(\mathbf{y}) ||M(\mathbf{y} - \mathbf{x}_i)||_p^p d\mathbf{y}$$

The following formula illustrates the relationship between the density ρ and the mesh size h . The procedure that has been used to obtain it is similar to the one described in another article [14].

$$\rho \sim \frac{1}{h^{p+2}}$$

The metric can be considered constant by element. It is preferable to use a continuous density however.

3 Results

In this section, different examples will be presented in order to show that Levy’s algorithm can create high quality quad surface meshes that have well defined orientations.

Five steps are necessary to produce quad meshes :

1. Compute a conformal mapping that maps the 3D surface to a 2D parametric space [15].
2. Create a triangular mesh in the parametric space using a 2D mesh generator.
3. Use this mesh to compute an alignment direction. The alignment direction is already known at the boundary. By using the heat transfer equation, it can be obtained everywhere else inside the domain [16].

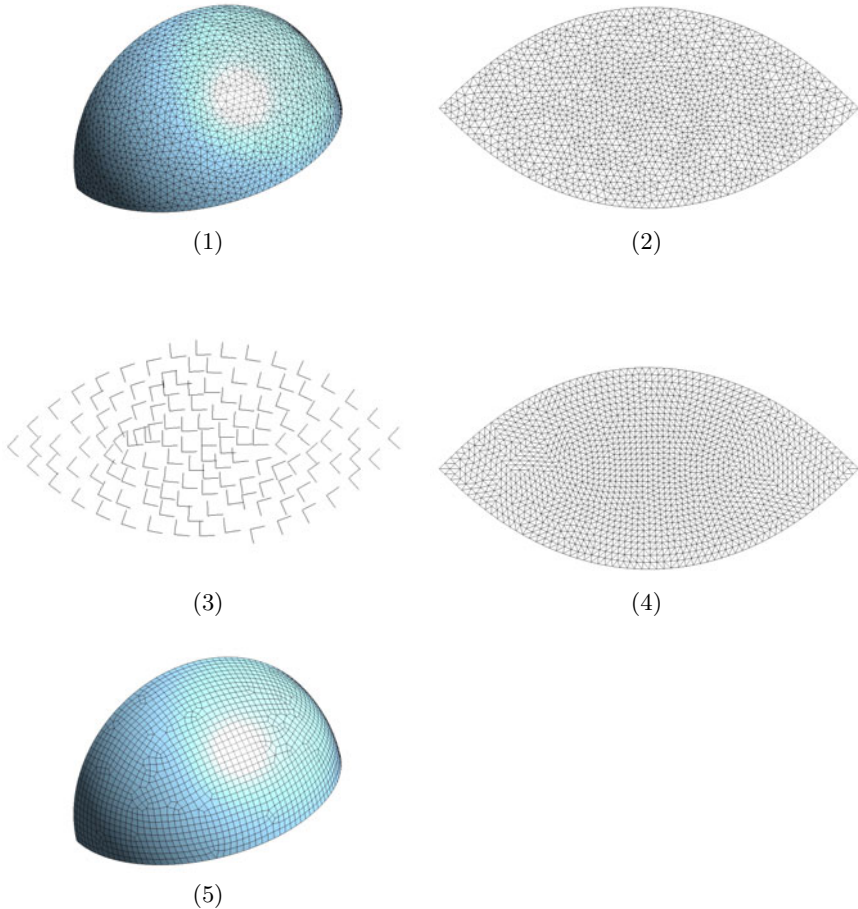


Fig. 9. The different steps necessary to produce quad meshes.

4. Use Levy's algorithm to optimize the location of the mesh vertices. L_6 is a good compromise between speed and squareness.
5. Apply the Blossom-Quad algorithm described in [11] to combine the triangles into quads. Blossom-Quad uses the well-known Blossom algorithm in order to find a perfect matching between triangles. The matching also optimizes the mean quality of the quads.

Fig. 9 illustrates the different steps of the global process. (1) is the triangular mesh of the geometry in the three-dimensional space. (2) is the triangular mesh of the geometry in the parametric space. (3) shows the cross-field determining the orientation of the quads. (4) is the triangular mesh in the parametric space after the application of Levy's algorithm. (5) is the final quad mesh.

A car hood is depicted on Fig. 10. (A) is the triangular mesh obtained with the *mesh adapt* algorithm from Gmsh [17]. It contains 1740 triangles. (AB) is the quad mesh obtained by applying Blossom-Quad directly on (A). (AB) contains randomly oriented quads. (ABL) is the quad mesh obtained by applying Levy's algorithm before the recombination. Most of the quads are now oriented in the specified direction.

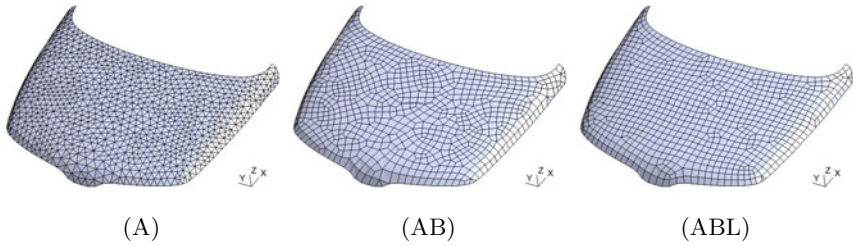


Fig. 10. Different meshes of a car hood. (A) is a triangular mesh, (AB) is a quad mesh created only with Blossom-Quad algorithm and (ABL) is a quad mesh created with both Levy's and Blossom-Quad algorithms.

The (AB) mesh has a mean quality of $\bar{\eta} = 0.79$. The (ABL) mesh has a mean quality of $\bar{\eta} = 0.88$. The quality of a quadrilateral $\eta(q)$ is defined by the values of its four angles α_k , $k = 1, 2, 3, 4$ [11]:

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

If the element is a perfect square, $\eta(q)$ is equal to one.

In an ideal quad mesh, each non-boundary mesh vertex is connected to four neighbors. In (AB), 62% of the non-boundary mesh vertices are 4-valent. In (ABL), this number reaches 77%.

Fig. 11 shows the decrease of the energy in function of the number of iterations for the car hood problem. Voronoi diagrams at various steps of the optimization process are also shown. As the number of iteration increases, the Voronoi cells become more and more rectangular.

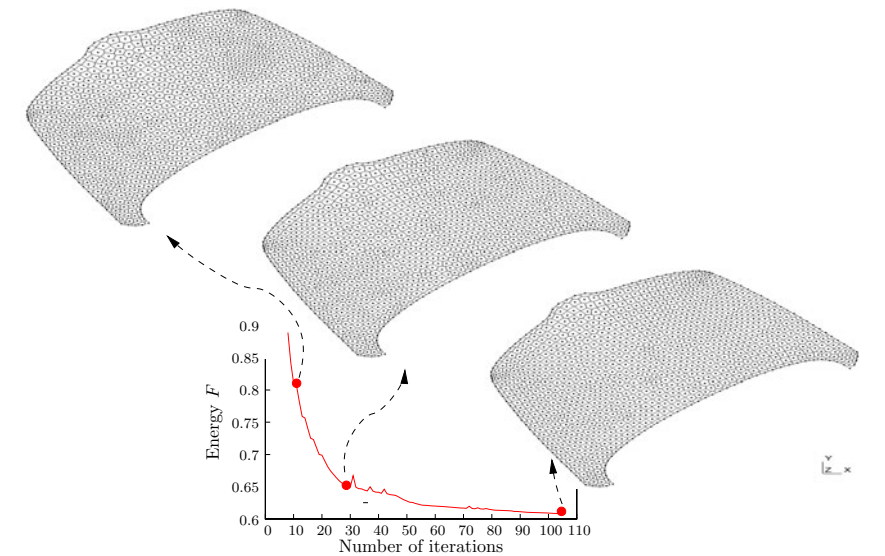


Fig. 11. Convergence curve of the car hood problem. The Voronoi diagrams in the parametric space for three different iterations (12, 26, 106) are shown.

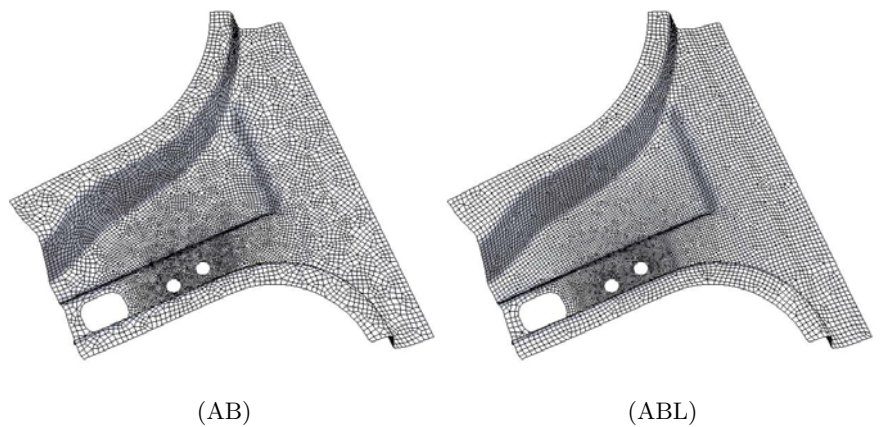


Fig. 12. (AB) is a quad mesh created only with Blossom-Quad algorithm and (ABL) is a quad mesh created with both Levy's and Blossom-Quad algorithms.

Fig. 12 shows another car body part. Again, it compares two quad meshes, one created with Levy's algorithm and one without. Both meshes contain 9954 quads. It took 6 minutes and 22 seconds to perform the 202 iterations of Levy's algorithm on a standard 2010 laptop.

(AB) has a mean quality of $\bar{\eta} = 0.79$ and (ABL) has a mean quality of $\bar{\eta} = 0.90$. The percentage of 4-valent vertices is equal to 71% in (AB). It is equal to 89% in (ABL).

4 Conclusion

A two-dimensional implementation of Levy's algorithm has been described in this article. When used in combination with an indirect algorithm like Blossom-Quad, it is able to create well-oriented quads of varying size. By taking advantage of parametrization techniques, curved surfaces can also be meshed. The only apparent drawback of this method is the execution time. It is much more complex than the traditional Lloyd's algorithm. Optimizing large meshes can take very long.

Indirect methods can also be used to create hexahedra. However, in order to create good quality hex meshes aligned in precise directions, a three-dimensional version of Levy's algorithm would be necessary. The implementation described in this article can be used as a starting point. Computing the energy and the gradient would not be particularly more difficult in three-dimension. Nevertheless, clipping a Voronoi diagram in three-dimension would be much more complex, but feasible.

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

References

1. Remacle, J.-F., Marchandise, E., Geuzaine, C., Béchet, E.: The domhex proposal (2010)
2. Prakash, S., Ethier, C.R.: Requirements for mesh resolution in 3d computational hemodynamics. *Journal of Biomechanical Engineering* 123, 134–144 (2001)
3. Vinchurkar, S., Longest, P.W.: Effect of mesh style and grid convergence on particle deposition in bifurcating airway models with comparisons to experimental data. *Medical Engineering and Physics* 29, 350–366 (2007)
4. Vinchurkar, S., Longest, P.W.: Evaluation of hexahedral, prismatic and hybrid mesh styles for simulating respiratory aerosol dynamics. *Computers and Fluids* 37, 317–331 (2008)
5. Ferziger, J., Peric, M.: *Computational Methods for Fluid Dynamics*. Springer, Berlin (1999)
6. Roache, P.: *Computational Fluid Dynamics*. Hermosa, Albuquerque (1992)
7. Zienkiewicz, O.C., Taylor, R.L.: *The Finite Element Method. The Basis*, vol. 1. Butterworth-Heinemann, Oxford (2000)

8. Hughes, T.: The Finite Element Method - Linear Static and Dynamic Finite Element Analysis. Dover Publications, New York (2000)
9. Zienkiewicz, O.C., Rojek, J., Taylor, R.L., Pastor, M.: Triangles and tetrahedra in explicit dynamic codes for solids. *International Journal for Numerical Methods in Engineering* 43, 565–583 (1998)
10. Sherwin, S.J., Peiró, J.: Mesh generation in curvilinear domains using high-order elements. *International Journal for Numerical Methods in Engineering* 53, 207–223 (2002)
11. Remacle, J.-F., Lambrechts, J., Seny, B., Marchandise, E., Johnen, A., Geuzaine, C.: Blossom-quad: a non-uniform quadrilateral mesh generator using a minimum cost perfect matching algorithm. *International Journal for Numerical Methods in Engineering* (submitted 2011)
12. Lévy, B., Liu, Y.: l_p centroidal voronoi tessellation and its applications. In: Hoppe, H. (ed.) *ACM Transactions on Graphics*, Los Angeles, USA (2010)
13. Yan, D.-M., Wang, W., Lévy, B., Liu, Y.: Efficient Computation of 3D Clipped Voronoi Diagram. In: Mourrain, B., Schaefer, S., Xu, G. (eds.) *GMP 2010*. LNCS, vol. 6130, pp. 269–282. Springer, Heidelberg (2010)
14. Du, Q., Wang, D.: Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations. *International Journal for Numerical Methods in Engineering* 56, 1355–1373 (2003)
15. Marchandise, E., Crosetto, P., Geuzaine, C., Remacle, J.F., Sauvage, E.: Quality open source mesh generation for cardiovascular flow simulations. Springer, Heidelberg (2011)
16. Remacle, J.-F., Henrotte, F., Carrier Baudouin, T., Béchet, E., Marchandise, E., Geuzaine, C., Mouton, T.: A frontal delaunay quad mesh generator using the l^∞ norm. *International Journal for Numerical Methods in Engineering* (in preparation, 2011)
17. Geuzaine, C., Remacle, J.-F.: Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering* 79, 1309–1331 (2009)