An elasticity-based smoothing post-processing algorithm for the quality improvement of quadrilateral elements

Cesar Blecua and Antonio Falcó

Departamento de Matemáticas, Física y Ciencias Tecnológicas, Universidad CEU Cardenal Herrera, CEU Universities, San Bartolomé 55, 46115 Alfara del Patriarca, Spain;
and
ESI International Chair@CEU-UCH, Universidad Cardenal Herrera-CEU, CEU Universities San Bartolomé 55, 46115 Alfara del Patriarca, Spain
cblecua@uchceu.es, afalco@uchceu.es

Abstract

Post-processing meshing algorithms are widely used to achieve the desired quality in quadrilateral meshes. Assuming that the mesh quality depends on the distortion and the size error of each of its convex quadrilaterals, its deficiencies arises by consider solutions based in minimizing either the distortion or the size error. To solve this undesirable situation, in this paper we propose a new smoothing post-processing meshing algorithm. This procedure provides a good compromise between the distortion and the size of each element on the mesh. It is formulated by using an elasticity-based argument and allows to be implemented either in sequential or parallel form. Moreover, it provides a good quality output compared with some of the usual smoothing post-processing meshing algorithms.

Keywords: Mesh Quality, Mesh Smoothing, Quadrilateral Element, Unstructured Mesh
2010 MSC: 00-01, 99-00

1. Introduction

A quadrilateral meshing algorithms provide a result that usually in order to achieve a target quality [1, 2] requires the use of a post-processing procedure that can fall in one of following two categories. The first one, contains the so-called the *clean-up* algorithms. They are able to modify the mesh topology by creating, removing, and reconnecting nodes and elements. The second category, contains the *smoothing* algorithms. They proceed modifying the coordinates of the interior nodes and maintaining the mesh topology. It is well-known (Cita requerida) that the use of a *smoothing* post-processing procedure cannot achieved a high quality output over a mesh which is far from a mesh endowed with a topology that only contains squares.

The goal of this paper is to introduce a meshing post-processing algorithm for the quality enhancement of quadrilateral elements. Along this paper we assume that a *structured mesh* is a mesh whose interior nodes, that is, the nodes that are outside the domain's boundary, are the vertices of exactly four sides. Our starting point is the quadrilateral meshing algorithm introduced by Sarrate and Huerta [3]. The aforementioned procedure provides a satisfactory topology (the obtained output is either a structured mesh, when it is used over in a suitable contour, or an unstructured mesh otherwise). This fact allows to propose a *smoothing*-only post-processing, avoiding the use of a *clean-up* method. We also use the following characteristics and assumptions to construct our procedure:

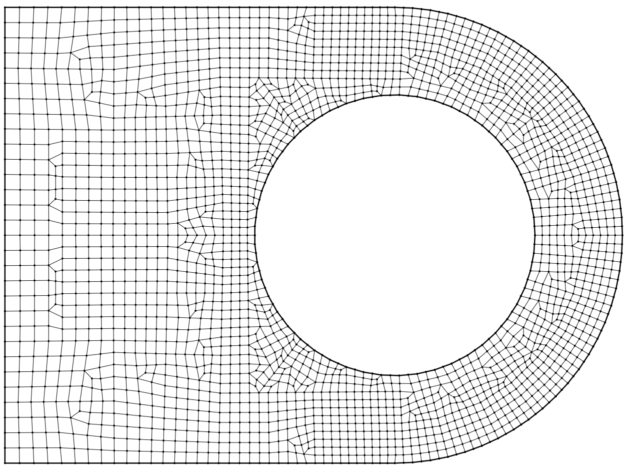
- It doesn't require an initial triangle mesh (it's a *direct* method).
- It is a recursive subdivision algorithm. It chooses the best splitting line by assigning a cost line to each possible candidate and then picking the optimal one (hence, it is easy to be implemented).
- It avoids the use of specific treatments with the only exception of 6 sided contours. However, following [4], a simple mapping is performed for subdividing them with some minor modifications.
- We avoid to solve complex geometrical situations (and hence difficult cases that frequently arise in *advancing front* and *paving* [5] approaches).

Figure 1a shows an output obtained by a meshing algorithm implemented following the above considerations. It is applied to a two dimensional region with a boundary combining rectangular and circular areas in order to test the behaviour from straight to curved perimeters. Moreover, it has a gradation over the elements size (near the curved border they have a side length around the half of the opposite side straight border). To conclude, observe that the contour is symmetric with respect to the horizontal axis. Thus, when the initial contour is symmetric the procedure provides a symmetric mesh.

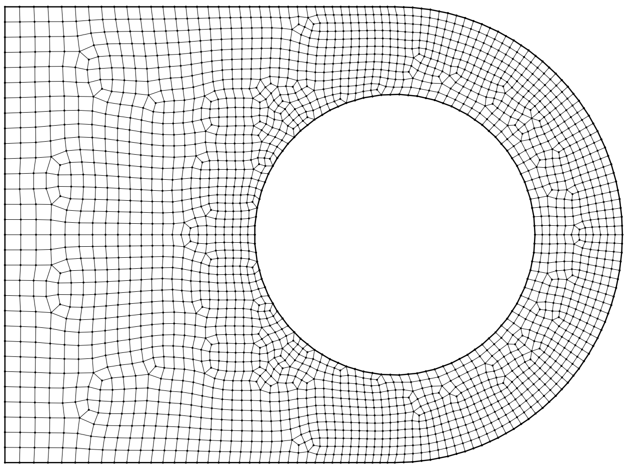
The result given in Figure 1a satisfies some of the expected requirements of a quadrilateral mesh with of a topology endowed by square elements: (a) whenever is possible it favours a structured meshing, (b) it respects element size gradation when it's variable across the meshing domain, (c) it only uses quadrilateral elements, (d) it works well over arbitrary contours, and (e) the elements tend to inherit the directions of the perimeter (in this example, he quadrilaterals arriving to the circular border acquire a radial directionality). We point out that the above requirements has been achieved without intervention of the end-user.

On the other hand, this mesh is still far from the shape quality that we shall expect. Some quadrilaterals are noticeably distorted, and not all the elements have their size accordingly to the place they are located within the domain. Even the resulting mesh topology is good, some node coordinates need to be modified for achieving the desired quality. More precisely, Figure 1b provides an example of the quality improvement that can result after applying a *smoothing* post-processing. Now, it seems to be clear that before to proceed we need to define a performance measure for the "quality" of the mesh topology. Some questions arising in this framework are the following. Can be the "quality" quantified? if the answer is positive, will depends it on one or in several criteria? and, if it depends on several criteria, will be they mutually diverging?

In a close look at Figure 1a, we can observe following two cases for poor quality elements (Figure 2):Either they are too distorted (Figure 2a) or have an inadequate size (Figure 2b). Thus, we shall get a mesh with the desired quality by applying a post-processing that fixes these two deficiencies. A natural procedure to improve the quality issues in Figures 2a and 2b would be minimizing the distortion of the involved quadrilaterals (we fix the distortion of a square to be zero), which would fix the deficiencies in Figure 2b, and minimizing the elements size error, which would address the issues from Figure 2a. Applying a brute force minimization strategy to both approaches (by evaluating a massive number of points) we



(a) Without post-processing.



(b) With smoothing post-processing.

Figure 1: Quality improvements from smoothing post-processing. Note that this example has a size gradation for elements (double side length in nodes at the left).

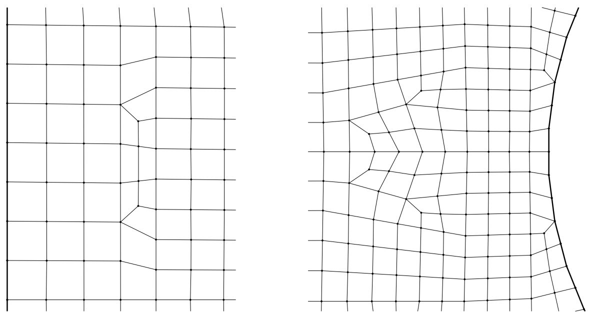
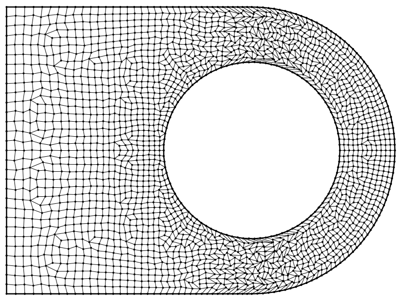

(a) Elements with a size quite different from the one they should have in that place. (b) Excessive distortion in some elements.

Figure 2: Quality deficiencies without post-processing.

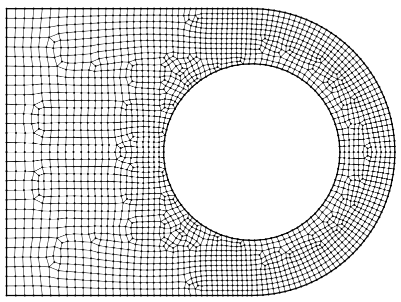
obtain as output the Figures 3a,3b and 3c. In particular, Figure 3a has been obtained by modifying the nodes coordinates under the constraint that the difference between the element and an "ideal square" should be minimal. This "ideal square" has a fixed size at each place in the domain. Thus, it allows to address the deficiencies related to the element initial size.. However, this approach leads us to an unacceptable distortion of the elements far worse than before the post-processing (compare Figure 3a with Figure 1a).

Now, in Figure 3b, we give the result obtained after minimizing the so-called Oddy distortion of the mesh (see 3.2.1 below). Even the mesh looks more pleasant to the eye, a more careful inspection finds that some issues related to wrong sizes of the elements have not been fixed. This is due to the fact that the minimization process ignore the element size. Figure 3c reveals the existence, after the distortion minimization, of quadrilaterals bigger and smaller than their neighbours in places where it seems desirable to have elements of constant size. Therefore, it's not possible to achieve a structured output addressing each of the two aforementioned quality deficiencies separately.

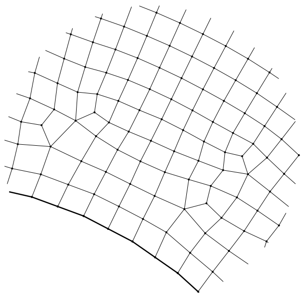
In order to answer the questions given above, we fix that the quality of a quadrilateral mesh depends to optimize: *the distortion and the size of the quadrilaterals*. However, the



(a) Minimization of the element size error.



(b) Minimization of the Oddy distortion.



(c) Close-up of the Oddy distortion minimization, showing elements notably bigger and smaller than their neighbours (in a place where all elements should have the same size).

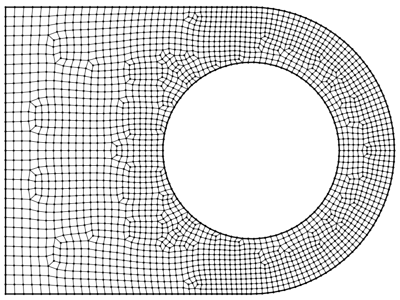
Figure 3: First approaches towards quality improvement.

fulfilment of both quantities is in general not possible. Thus, in order to find a quasi-optimal solution we need to fix a compromise between both criteria. A first approach to this strategy [3] is a *smoothing* post-processing algorithm introduced by Giuliani [6]. This procedure tends to minimize the distortion and hence to achieve an uniform element size. However, this last fact has the consequence of blurring or even losing the sizes gradient. Figure 4a shows how the size gradient in the original mesh (Figure 1a) has been noticeably lost applying the Giuliani algorithm. Indeed, due to this undesirable fact, Sarrate and Huerta proposed in [3] a simple variation for the Giuliani algorithm. They trying to respect the size of the mesh elements. Obviously this cannot fix the size errors in the mesh before post-processing (in fact the algorithm has no knowledge of the desired element size, so it cannot make any size corrections), but it does better respect size gradations (Figure 4b).

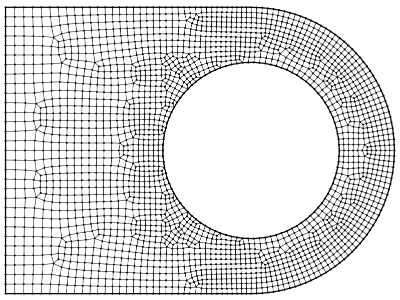
We must mention however that experience shows this variation of the Giuliani algorithm not only respects the size gradations, but tends to exaggerate them, introducing visually noticeable noise patterns throughout the mesh because it produces areas with elements smaller than the average size in their neighbourhood. At Figure 4b, it can be seen that this algorithm not only has not made the elements more uniformly sized, but it has even incremented the differences, and these noise patterns can be perceived. In [7], the authors introduce a new proposal for *smoothing* post-processing based on the distortion formulation by Knupp [8]. In this new development, the element size error is included in the function to be minimized, so that the post-processing not only reduces distortion, but also the size error. Minimization on each element, denoted by $elem$, is performed by a function which is the product of the distortion of the given element, denoted by η_{sh} and where $1 \leq \eta_{sh}(elem) < \infty$, by the size error, denoted by η_{si} and where $1 \leq \eta_{si}(elem) < \infty$), i.e.

$$\eta(elem) = \eta_{sh}(elem)\eta_{si}(elem). \tag{1}$$

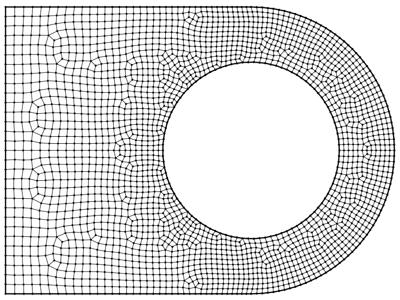
Now, we can understand the idea we outlined before: seeking optimal quality implies a compromise between distortion and size error (in this case, the compromise has been chosen as the product between the two). The development in [7] is based in the distortion formulation by Knupp [8] and it minimizes the arithmetic mean (rather than the maximum)



(a) Original Giuliani algorithm.



(b) Giulani algorithm as modified by Sarrate/Huerta.



(c) Minimization of the product of the size error by the average of the distortion.

Figure 4: Some smoothing post-processing algorithms.

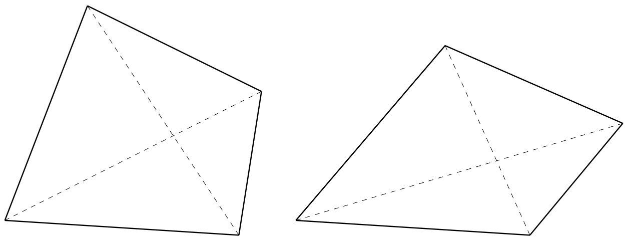


Figure 5: The sides of both quadrilaterals have the same length, but their diagonals vary, altering the distortion.

of the distortion at the element vertices, and because of this it cannot be directly compared to the results of the algorithm that we propose. Thus, we have made a reinterpretation of [7], implementing it with the same distortion formulation that our algorithm uses, in order to compare it with our results. The output from this reinterpretation is shown in Figure 4c. Considering that the elements are quadrilaterals, it's obvious that their diagonals have a great responsibility on the quadrilateral distortion, while the sides affect mainly its size. Of course these are not completely independent responsibilities: diagonals have also an impact on the quadrilateral size (keeping the distortion constant, diagonals will be longer in a bigger quadrilateral), and the side lengths also affect distortion (the more uniform the side lengths, the more likely to achieve a low distortion). But it's clear that if we keep the sides with a fixed length, the diagonals get a crucial importance in distortion (see Figure 5). And if we wish to modify the size without altering the distortion, we need to apply the same scale factor to all the sides and the diagonals.

The paper is organised as follows COMPLETAR DISTRIBUCION

2. A elasticity-based smoothing algorithm for a quadrilateral mesh

The aim of this section is to give some preliminary definitions that we will use to describe our quadrilateral mesh smoothing algorithm for a given two dimensional region and also to explain the smoothing strategy used to each vertex in the mesh. As we said above, we'll be taking a compromise between the lengths of the sides and the diagonals over each quadrilateral in the mesh. Formulating a balance of lengths suggests the physical parallel of a

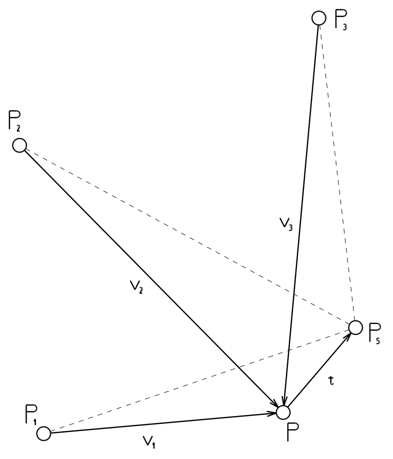


Figure 6: Equilibrium at a joint.

spring system, or the equilibrium of members subjected to axial force. There's previous work in spring-based smoothing post-processing (Lohner *et al* [9], as well as an implementation in the ANSYS software), but those developments don't take the quest for achieving minimum distortion in quadrilateral elements (their formulation assumes triangle meshes). This will require to introduce a distortion magnitude into the model, so that the diagonals tend to minimize it. Moreover, it shall be also necessary to take into account its quadrilateral sides in order to reach the desired compromise between size and distortion.

Let us consider a quadrilateral mesh $\mathfrak{Q}(\Omega) = \{\mathcal{Q}_\alpha : \alpha \in \mathfrak{I}\}$ of a given two dimensional connected region Ω. We will consider the set of mesh points given by the vertices of the individuals in $\mathfrak{Q}(\Omega)$, that is,

$$\mathcal{V}(\mathfrak{Q}(\Omega)) = \{\boldsymbol{P} \in \Omega : \boldsymbol{P} \text{ is a vertex of } \mathcal{Q} \text{ for some } \mathcal{Q} \in \mathfrak{Q}(\Omega)\}.$$

Next, we will explain for a generic fixed point $\boldsymbol{P} \in \mathcal{V}(\mathfrak{Q}(\Omega))$ (just before the smoothing post-processing) the strategy to compute its smoothing version. Let $\boldsymbol{P_S}$ be the displacing position for $\boldsymbol{P}$ after the smoothing post-processing. To compute $\boldsymbol{P_S}$, we assume that $\{\boldsymbol{P_i} \in \mathcal{V}(\mathfrak{Q}(\Omega)) : 1 \leq i \leq n\}$ are the initial n mesh points connected to $\boldsymbol{P}$ by either a side or a diagonal of a quadrilateral $\mathcal{Q} \in \mathfrak{Q}(\Omega)$. More precisely, given $\boldsymbol{P}$ we will take into account the set

$$\mathcal{C}(\boldsymbol{P}) := \{\boldsymbol{P_i P} : \boldsymbol{P_i P} \text{ is either a side or a diagonal for some } \mathcal{Q} \in \mathfrak{Q}(\Omega),\ 1 \leq i \leq n\}.$$

(see Figure 6). Let be denote by n_s (respectively, n_d) the number of sides in $\mathcal{C}(\boldsymbol{P})$ (respectively, the number of diagonals in $\mathcal{C}(\boldsymbol{P})$). Thus, $n = n_s + n_d$. We also assume the existence of a map

$$L : \mathcal{C}(\boldsymbol{P}) \longrightarrow (0, \infty), \quad \boldsymbol{P_i P} \mapsto L(\boldsymbol{P_i P}) := L_i,$$

where L_i is a previously fixed ideal length for $\boldsymbol{P_i P}$ (in Section 3 below we explain the practical construction of L).

Let $\boldsymbol{v_i} := (v_i^x, v_i^y)$ be direction vector associated to the corresponding segment $\boldsymbol{P_i P}$. Thus, $\|\boldsymbol{v_i}\|$ is the length of $\boldsymbol{P_i P}$. Let $\boldsymbol{t} = (t^x, t^y)$ be the displacement vector $\boldsymbol{P_S P}$ joining $\boldsymbol{P}$ together its smoothed position, $\boldsymbol{P_S}$. It will be our goal variable that needs to be calculated (see also Figure 6).

In order to define an optimal value for $\boldsymbol{t}$, we consider that $\boldsymbol{P_i P}$ is a bar with a length L_i and associated cross sectional area A. We assume that L_i is given and A is a fixed constant independent of i that needs to be determined. Consider that $\boldsymbol{P_i P}$ is subjected to equal and opposite axial forces $N_{S,i}$ pulling at the ends so the bar is under tension. Then we assume that $\boldsymbol{P_i P}$ suffers a stress producing a distortion length $\|\boldsymbol{v_i} + \boldsymbol{t}\|$. In consequence, the strain will be given by the ratio:

$$\frac{\|\boldsymbol{v_i} + \boldsymbol{t}\| - L_i}{L_i}.$$

Let $E_i(\boldsymbol{t})$ be a given non-linear function of $\boldsymbol{t}$ that represents a modulus of elasticity for $\boldsymbol{P_i P} \in \mathcal{C}(\boldsymbol{P})$ (it will be different depending if $\boldsymbol{P_i P}$ is either a side or a diagonal and we will give its construction below in Section 3). Now, following the relationship between stress-strain and the modulus of elasticity, we can write

$$\frac{N_{S,i}}{A} = E_i(\boldsymbol{t})\,\frac{\|\boldsymbol{v_i} + \boldsymbol{t}\| - L_i}{L_i}, \tag{2}$$

We consider that the direction vector representing the axial force direction after smoothing is:

$$\boldsymbol{u_{S,i}} = \frac{\boldsymbol{v_i} + \boldsymbol{t}}{\|\boldsymbol{v_i} + \boldsymbol{t}\|} \tag{3}$$

and, hence the i-th axial force after smoothing exerted at point $\boldsymbol{P_S}$, yields:

$$\boldsymbol{F_i} = \boldsymbol{u_{S,i}} \cdot N_{S,i} = \frac{\boldsymbol{v_i} + \boldsymbol{t}}{\|\boldsymbol{v_i} + \boldsymbol{t}\|} \cdot \frac{(\|\boldsymbol{v_i} + \boldsymbol{t}\| - L_i)E_i(\boldsymbol{t})A}{L_i}. \tag{4}$$

Assuming an equilibrium of forces at the smoothed joint $\boldsymbol{P_S}$:

$$\sum_{i=1}^{n} \boldsymbol{F_i} = \boldsymbol{0}, \tag{5}$$

leads to:

$$\sum_{i=1}^{n} \frac{\boldsymbol{v_i} + \boldsymbol{t}}{\|\boldsymbol{v_i} + \boldsymbol{t}\|} \cdot \frac{(\|\boldsymbol{v_i} + \boldsymbol{t}\| - L_i)E_i(\boldsymbol{t})}{L_i} = \boldsymbol{0} \tag{6}$$

which is a system of equations in $\boldsymbol{t}$ (independent of the constant A). In order to solve (6) we consider the function

$$\boldsymbol{f_P}(\boldsymbol{t}) := \sum_{i=1}^{n} \boldsymbol{f_P^{(i)}}(\boldsymbol{t}) = \sum_{i=1}^{n} (f_1^{(i)}(\boldsymbol{t}), f_2^{(i)}(\boldsymbol{t})). \tag{7}$$

where

$$\boldsymbol{f_P^{(i)}}(\boldsymbol{t}) := \frac{\boldsymbol{v_i} + \boldsymbol{t}}{\|\boldsymbol{v_i} + \boldsymbol{t}\|} \cdot \frac{(\|\boldsymbol{v_i} + \boldsymbol{t}\| - L_i)E_i(\boldsymbol{t})}{L_i} \text{ for } 1 \leq i \leq n.$$

Observe, that we can write the above function as

$$\boldsymbol{f_P^{(i)}}(\boldsymbol{t}) = \boldsymbol{g_P^{(i)}}(\boldsymbol{t})E_i(\boldsymbol{t})$$

where

$$\boldsymbol{g_P^{(i)}}(\boldsymbol{t}) := (\boldsymbol{v_i} + \boldsymbol{t}) \cdot \frac{(\|\boldsymbol{v_i} + \boldsymbol{t}\| - L_i)}{L_i\|\boldsymbol{v_i} + \boldsymbol{t}\|},$$

and we have the following lemma.

Lemma 2.1. *Given $\boldsymbol{v_i} \in \mathbb{R}^2$ consider the open set $\mathcal{O}_i = \{\boldsymbol{t} \in \mathbb{R}^2 : \boldsymbol{t} + \boldsymbol{v_i} \neq \boldsymbol{0}\}$. Then the function $\boldsymbol{g_P^{(i)}} : \mathcal{O}_i \longrightarrow \mathbb{R}^2$ is $\mathcal{C}^1(\mathcal{O}_i)$ and its derivative is given by*

$$D\boldsymbol{g_P^{(i)}}(\boldsymbol{t}) = \frac{(\|\boldsymbol{v_i} + \boldsymbol{t}\| - L_i)}{L_i\|\boldsymbol{v_i} + \boldsymbol{t}\|} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{1}{\|\boldsymbol{v_i} + \boldsymbol{t}\|^3} \begin{pmatrix} (v_i^x + t^x)^2 & (v_i^x + t^x)(v_i^y + t^y) \\ (v_i^x + t^x)(v_i^y + t^y) & (v_i^y + t^y)^2 \end{pmatrix}.$$

Proof. See Appendix A. ∎

Now, we have all ingredients to propose the following:

Individual Smoothing Procedure (ISP): Given $\boldsymbol{P} \in \mathcal{V}(\mathfrak{Q}(\Omega))$ construct $\boldsymbol{f_P}$ and compute $\boldsymbol{t}^*$ satisfying $\boldsymbol{f_P}(\boldsymbol{t}^*) = \boldsymbol{0}$. Return $\boldsymbol{P_S}$.

It allows us to introduce the following two smoothing procedures for $\mathfrak{Q}(\Omega)$.

Sequential Procedure:

```
1. Put $\mathcal{V}_{NS}(\mathfrak{Q}(\Omega)) \leftarrow \mathcal{V}(\mathfrak{Q}(\Omega))$;
2. Repeat until $\mathcal{V}_{NS}(\mathfrak{Q}(\Omega)) = \emptyset$ :
    (a) Take $\boldsymbol{P} \in \mathcal{V}_{NS}(\mathfrak{Q}(\Omega))$;
    (b) Run (ISP) to compute $\boldsymbol{P_S}$;
    (c) Put $\mathcal{V}(\mathfrak{Q}(\Omega)) \leftarrow (\mathcal{V}(\mathfrak{Q}(\Omega)) \setminus \{\boldsymbol{P}\}) \cup \{\boldsymbol{P_S}\}$;
    (d) Put $\mathcal{V}_{NS}(\mathfrak{Q}(\Omega)) \leftarrow \mathcal{V}_{NS}(\mathfrak{Q}(\Omega)) \setminus \{\mathbf{P}\}$;
3. Return $\mathcal{V}(\mathfrak{Q}(\Omega))$.
```

Let $D = \{1, 2, \ldots, d\}$ be the number of available computational devices to implement (SP) in parallel.

Parallel Procedure :

```
1. Put $\mathcal{V}_{NS}(\mathfrak{Q}(\Omega)) \leftarrow \mathcal{V}(\mathfrak{Q}(\Omega))$ and $\mathcal{V}_S(\mathfrak{Q}(\Omega)) \leftarrow \emptyset$;
2. Repeat until $\mathcal{V}_{NS}(\mathfrak{Q}(\Omega)) = \emptyset$ :
    (a) Take $\boldsymbol{P}^{(1)}, \ldots, \boldsymbol{P}^{(d)} \in \mathcal{V}_{NS}(\mathfrak{Q}(\Omega))$;
    (b) Run (ISP) in device $1 \leq i \leq d$ to compute $\boldsymbol{P}_{\boldsymbol{S}}^{(i)}$;
    (c) Put $\mathcal{V}_S(\mathfrak{Q}(\Omega)) \leftarrow \mathcal{V}_S(\mathfrak{Q}(\Omega)) \cup \{\boldsymbol{P}^{(1)}, \ldots, \boldsymbol{P}^{(d)}\}$;
    (d) $\mathcal{V}_{NS}(\mathfrak{Q}(\Omega)) \leftarrow \mathcal{V}_{NS}(\mathfrak{Q}(\Omega)) \setminus \{\boldsymbol{P}^{(1)}, \ldots, \boldsymbol{P}^{(d)}\}$;
3. Return $\mathcal{V}_S(\mathfrak{Q}(\Omega))$.
```

The difference between the above two procedures is 2.(c). Recall that the set $\mathcal{V}(\mathfrak{Q}(\Omega))$ is used to compute $\mathcal{C}(\boldsymbol{P})$ that allows to construct $f_{\boldsymbol{P}}$. In the Sequential Procedure 2.(c) we modify the set $\mathcal{V}(\mathfrak{Q}(\Omega))$ leaving the vertex $\boldsymbol{P}$ and introducing smoothing one $\boldsymbol{P_S}$. Thus, there are vertices $\boldsymbol{P}' \in \mathcal{V}(\mathfrak{Q}(\Omega))$ such that before we run (SP) it holds $\boldsymbol{P}'\boldsymbol{P} \in \mathcal{C}(\boldsymbol{P}')$, however after running (SP) we have $\boldsymbol{P}'\boldsymbol{P} \notin \mathcal{C}(\boldsymbol{P}')$ (because in the Sequential Procedure 2.(c) we leave $\boldsymbol{P}$ from $\mathcal{V}(\mathfrak{Q}(\Omega))$. On the other hand, in the Parallel Procedure the set $\mathcal{V}(\mathfrak{Q}(\Omega))$ remains invariant and, in consequence, we compute each $\boldsymbol{P_S}$ in an independent way allowing to parallelize the algorithm.

3. The Individual Smoothing Procedure (ISP) function

In this section we develop in detail the construction of the Individual Smoothing Procedure (ISP) function $\boldsymbol{f_P}$ used to compute the vector $\boldsymbol{t}^*$ satisfying $\boldsymbol{f_P}(\boldsymbol{t}^*) = \boldsymbol{0}$. First at all, we decompose the map $\boldsymbol{f_P}$ as follows. Let

$$\boldsymbol{f_P}(\boldsymbol{t}) = \boldsymbol{f}_{\boldsymbol{P}}^{(s)}(\boldsymbol{t}) + \boldsymbol{f}_{\boldsymbol{P}}^{(d)}(\boldsymbol{t}) = \sum_{j=1}^{n_s}(f_1^{(j)}(\boldsymbol{t}), f_2^{(i)}(\boldsymbol{j})) + \sum_{k=1}^{n_d}(f_1^{(k)}(\boldsymbol{t}), f_2^{(k)}(\boldsymbol{t})),$$

where the map $\boldsymbol{f}_{\boldsymbol{P}}^{(s)}(\boldsymbol{t})$ takes into account the sides in $\mathcal{C}(\boldsymbol{P})$ and the map $\boldsymbol{f}_{\boldsymbol{P}}^{(d)}(\boldsymbol{t})$ takes into account the diagonals. Next we give the L_i values and the functions E_i used to construct $\boldsymbol{f_P}$. In particular, section 3.1 is devoted to the case for the quadrilateral sides and section 3.2 for the quadrilateral diagonals.

3.1. The Individual Smoothing Procedure function over the quadrilateral sides

As we discussed earlier, the quadrilateral sides have their part of responsibility for achieving the desired element size. In this situation we assume that for each $\boldsymbol{P} \in \mathcal{V}(\mathfrak{Q}(\Omega))$ there exists a previously fixed ideal length, denoted by $\ell_{\boldsymbol{P}} > 0$. Then, for each side $\boldsymbol{P_i P} \in \mathcal{C}(\boldsymbol{P})$ we will define

$$L_i = \frac{\ell_{\boldsymbol{P_i}} + \ell_{\boldsymbol{P}}}{2}.$$

Now, we model the modulus of elasticity for the quadrilateral sides as

$$E_i(\boldsymbol{t}) = 1 + e^{c\left(\left|1 - \frac{L_i}{\|\boldsymbol{v_i} + \boldsymbol{t}\|}\right| - d\right)} \tag{8}$$

where c, d are parameters for adjusting the exponential function as appropriate (our software implementation uses $c = 1$ and $d = 0$ with acceptable results). Therefore, the elements of the

Jacobian matrix $D\boldsymbol{f}_{\boldsymbol{P}}^{(s)}(\boldsymbol{t})$ corresponding to quadrilateral sides can be computing by using that

$$D\boldsymbol{f}_{\boldsymbol{P}}^{(i)}(\boldsymbol{t}) = E_i(\boldsymbol{t})D\boldsymbol{g}_{\boldsymbol{P}}^{(i)}(\boldsymbol{t})E_i(\boldsymbol{t}) + \boldsymbol{g}_{\boldsymbol{P}}^{(i)}(\boldsymbol{t})DE_i(\boldsymbol{t}),$$

Lemma 2.1 and together some algebra to obtain:

$$\begin{aligned} \frac{\partial f_1^{(i)}(\boldsymbol{t})}{\partial t^x} &= \frac{\left[\|\boldsymbol{v_i}+\boldsymbol{t}\|^3 + r - L_i\left(v_i^y + t^y\right)^2\right]\cdot E_i(\boldsymbol{t}) - r}{L_i\|\boldsymbol{v_i}+\boldsymbol{t}\|^3}, \\ \frac{\partial f_1^{(i)}(\boldsymbol{t})}{\partial t^y} = \frac{\partial f_2^{(i)}(\boldsymbol{t})}{\partial t^x} &= \frac{\left[E_i(\boldsymbol{t}) + s(E_i(\boldsymbol{t})-1)\right](v_i^x+t^x)(v_i^y+t^y)}{\|\boldsymbol{v_i}+\boldsymbol{t}\|^3}, \\ \frac{\partial f_2^{(i)}(\boldsymbol{t})}{\partial t^y} &= \frac{\left[\|\boldsymbol{v_i}+\boldsymbol{t}\|^3 + q - L_i\left(v_i^x + t^x\right)^2\right]\cdot E_i(\boldsymbol{t}) - q}{L_i\|\boldsymbol{v_i}+\boldsymbol{t}\|^3}, \end{aligned} \tag{9}$$

where

$$s = c\left|1 - \frac{L_i}{\|\boldsymbol{v_i}+\boldsymbol{t}\|}\right|, \quad ; \quad q = L_i s(v_i^y+t^y)^2, \quad ; \quad r = L_i s(v_i^x+t^x)^2,$$

and $1 \le i \le n_s$. The non-linearity of $E_i(\boldsymbol{t})$ will be essential as we want members whose length is farther from their goal length, to react with exponentially growing axial force (if E_i was linear, a very deformed member wouldn't be able to counteract a big set of better sized neighbouring members, leading to situations where badly shaped elements appear surrounded by a crowd of well shaped elements: non-linear behaviour avoids those situations).

3.2. The Individual Smoothing Procedure function over the quadrilateral diagonals

In the previous section, we supposed that the goal length for the diagonals, L_i is previously known. The different impact on the quality of a quadrilateral of its sides versus its diagonals, allows us to fix a compromise between the quadrilateral distortion and size. This fact invites to formulate a compromise between the lengths of the sides and the diagonals by means the notion of the distortion of a quadrilateral element. There are several proposals for its quantification and, in particular, Oddy *et al* [10] suggest a simple approach that can be applied to parallelograms and it can be extended to arbitrary quadrilaterals. Lee and Lo [11] use a distortion measure to get well shaped quadrilaterals by merging triangles (their algorithm generates quadrilaterals by grouping triangles from an already existing mesh).

Canann *et al* [12] use the Lee and Lo approach, with some modifications, for their implementation in the ANSYS software. Knupp [8] proposes a general formulation by consider a set of properties that affect the elements quality. Amongst all the available distortion evaluation formulations, we have chosen the Oddy *et al* one. This choice is motivated to have an easy implementation and to have good results at assessing quadrilaterals quality. However, the algorithm we'll described in the previous section is independent on it, and can be applied by using others distortion evaluation approaches. In this paper use of the Oddy distortion measure allows to give a collection of simplified equations that can be easily implemented in a computer program. Next, we will introduce the definition of the Oddy distortion measure for a quadrilateral.

3.2.1. The Oddy distortion measure for a quadrilateral

Following [13], the *Oddy distortion* of a given parallelogram $\mathcal{P}$ is defined by

$$D_{\text{Oddy}}(\mathcal{P}) = 2(Q^2_{\text{Oddy}}(\mathcal{P}) - 1), \tag{10}$$

here $Q_{\text{Oddy}}(\mathcal{P})$ is the *geometrical efficiency*:

$$Q_{\text{Oddy}}(\mathcal{P}) = \frac{l_1^2 + l_2^2}{2A}, \tag{11}$$

and where l_1 and l_2 are the lengths of any of the two adjacent sides in $\mathcal{P}$, and A is its area.

A way for generalizing this expression to arbitrary quadrilaterals consists in evaluating it at the four parallelograms that can be constructed from each of the vertices in the quadrilateral $\mathcal{Q}$ (drawing parallel lines to the both sides meeting at the vertex). In [13], this leads an expression for the Oddy distortion at a given vertex in an arbitrary quadrilateral. Let (x_j, y_j) be the coordinates of the j-th vertex of a given local quadrilateral $\mathcal{Q}$, and put

$$[i] = k \text{ for some } 1 \leq k \leq 4 \text{ if and only if } [i] \equiv k \mod 4.$$

Then the distortion for each vertex $i = 1, 2, 3, 4$ is computed by considering

$$l^2_{[i]} = (x_{[i+1]} - x_{[i]})^2 + (y_{[i+1]} - y_{[i]})^2$$

$$l^2_{[i+3]} = (x_{[i+3]} - x_{[i]})^2 + (y_{[i+3]} - y_{[i]})^2$$

$$\begin{aligned} A_{[i]} &= (x_{[i+1]} - x_{[i]})(y_{[i+3]} - y_{[i]}) - (x_{[i+3]} - x_{[i]})(y_{[i+1]} - y_{[i]}) \\ Q_{\text{Oddy},i}(\mathcal{Q}) &= Q_{\text{Oddy},i}(x_{[i+3]}, x_{[i+1]}, x_{[i]}, y_{[i+3]}, y_{[i+1]}, y_{[i]}) = \frac{l^2_{[i]} + l^2_{[i+3]}}{2A_{[i]}} \\ D_{\text{Oddy},i}(\mathcal{Q}) &= 2(Q^2_{\text{Oddy},i}(\mathcal{Q}) - 1). \end{aligned} \tag{12}$$

Observe that the function $Q_{\text{Oddy},i}$ is $\mathcal{C}^2$-differentiable in the open set

$$U_{[i]} := \left\{(x_{[i+3]}, x_{[i+1]}, x_{[i]}, y_{[i+3]}, y_{[i+1]}, y_{[i]}) \in \mathbb{R}^6 : A_{[i]} \neq 0\right\},$$

and hence also $D_{\text{Oddy},i}(\mathcal{Q}) \in \mathcal{C}^2(U_{[i]})$. Given that we want a single scalar value for measuring the global distortion of the quadrilateral, the means for avoiding neglecting local shape problems is to take the maximum distortion value amongst all the vertices in the quadrilateral. Thus,

$$D_{\text{Oddy}}(\mathcal{Q}) = \max_{1 \leq i \leq 4}(D_{\text{Oddy},i}(\mathcal{Q})), \tag{13}$$

is non-differentiable, and therefore compute a minimum value by classical smooth methods it is not possible. The solution in [13] to have a differentiable function is to use the average value from the distortions at the four vertices. However, the authors point out, this can result in a distortion value lower than the one that should be assigned to the quadrilateral. In our approach, it is not necessary as we explain below and allow us to consider a function based on the maximum rather than on the average. The map $D_{\text{Oddy}}(\mathcal{Q})$ is in fact a piecewise $\mathcal{C}^2$-function in $\mathbb{R}^8$ (see Chapter 4 in [14]) by considering that each $D_{\text{Oddy},i}(\mathcal{Q})$ is defined over $(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4) \in \mathbb{R}^8$ for $1 \leq i \leq 4$. In consequence, the $D_{\text{Oddy}}(\mathcal{Q})$ is a semi-smooth function (see Section 2.5.3 in [15]) and hence a non-smooth version of the classical Newton method is available (see Chapter 3 in [15]).

This strategy helps us to detect local anomalies that appears at some vertices. On the other hand, a direct implementation of equation (10) can generate asymptotes (in the case when $A = 0$) or provide relatively low distortion values in degenerate quadrilaterals (in the case that the quadrilateral has concave vertices). This last possibility is due that raising

Q_{Oddy} to two makes it lose its sign, which is equal to the sign of A (note that a vertex is concave if and only if the area A of the local parallelogram at the vertex is negative (assuming counter-clockwise vertex order, a requirement that we impose to construct our meshes). These cases are used in [13] to propose modifications to the D_{Oddy} definition. However, our algorithm imposes that $A > 0$ for all local parallelograms, which lets us to consider the original formulation for D_{Oddy}.

Next we will provide an analytical formulation to compute the diagonal goal length L_i with respect a fixed vertex. This value, as we will explain below, will be optimal for the Oddy distorsion measure.

3.2.2. An optimal diagonal length L_i associated to a quadrilateral $\mathcal{Q}$ with respect a fixed vertex

Let $\mathcal{Q}$ be a quadrilateral with vertices $\boldsymbol{P}, \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}$ in counter-clockwise order, namely $\mathcal{Q} = \{\boldsymbol{P}, \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}\}$ (see Figure 7). Assume that $\boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}$ are fixed points and we wish to construct a point $\boldsymbol{P_O}$ displacing $\boldsymbol{P}$ in the diagonal direction $\boldsymbol{PP_i}$ such the quadrilateral $\delta\mathcal{Q} = \{\boldsymbol{P_O}, \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}\}$ satisfies $D_{\text{Oddy}}(\delta\mathcal{Q}) < D_{\text{Oddy}}(\mathcal{Q})$ Our goal is to obtain a quadrilateral $\delta\mathcal{Q}$ with a diagonal length $L_i = \|\boldsymbol{P_O}\boldsymbol{P_i}\|$ for which the $D_{\text{Oddy}}(\delta\mathcal{Q})$ is a minimum. More precisely, we prove the following proposition.

Proposition 3.1. *Let $\mathcal{Q}$ be a quadrilateral with vertices $\boldsymbol{P}, \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}$ in counter-clockwise order, denoted as $\mathcal{Q} = \{\boldsymbol{P}, \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}\}$ and consider the displacements of a point $\boldsymbol{P_O}$ along the diagonal $\boldsymbol{PP_i}$, that is, $\boldsymbol{P_O}(m) = \boldsymbol{P} + m \cdot \boldsymbol{PP_i}$. Then the function $D_{Oddy}(m) = D_{Oddy}(\mathcal{Q}(m))$, where*

$$\mathcal{Q}(m) = \{\boldsymbol{P_O}(m), \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}\}.$$

is the quadrilateral with vertices $\boldsymbol{P_O}(m), \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}$ in counter-clockwise order, has an global minimum.

Proof. See Appendix B. ■

From the proof of Proposition 3.1 we can explicit compute the absolute minimum value m^* for the Oddy distortion measure. Thus substituting the value m^* in $\boldsymbol{P_O}(m)$ we a ob-

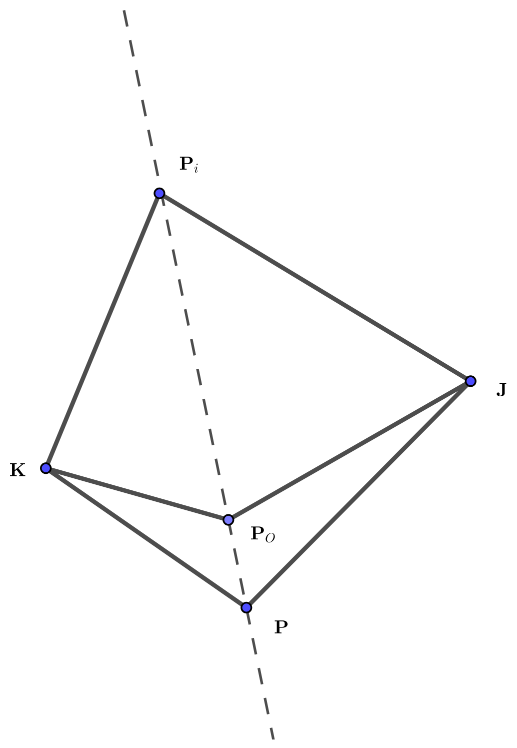


Figure 7: Quadrilateral with $\boldsymbol{P}, \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}$ vertices in counterclockwise order. Point $\boldsymbol{P}$ is the new position for $\boldsymbol{P}_O$ on the diagonal $\boldsymbol{P}\boldsymbol{P_i}$.

tain a vertex $\boldsymbol{P_O}(m^*)$ and conclude that the length $L_i = \|\boldsymbol{P_O(m^*)P_i}\|$ is associated to a quadrilateral $\mathcal{Q}^* := \{\boldsymbol{P_O}(m^*), \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}\}$ having an diagonal with an optimal length for the set of quadrilaterals $\mathcal{Q}(m) = \{\boldsymbol{P_O}(m), \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}\}$ that are obtaining by displacing $\boldsymbol{P}$ along the diagonal $\boldsymbol{PP_i}$.

Next, we will define an elasticity modulus $E_i(\boldsymbol{t})$ for a quadrilateral diagonal by using of the Oddy distortion measure.

3.2.3. The modulus of elasticity E_i for a quadrilateral diagonal

The non-linear strain-stress behaviour of diagonals is implicitly introduced by formulating their $E_i(\boldsymbol{t})$ modulus as a function of the Oddy distortion, turning it unnecessary to adopt an exponential function as we do for the quadrilateral sides, as follows. In a similar way as above, let $\mathcal{Q} = \{\boldsymbol{P}, \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}\}$ be a quadrilateral. Now, let $\boldsymbol{P_S}(\mathbf{t}) := \boldsymbol{P} + \mathbf{t}$ a and consider the parametrized quadrilaterals $\mathcal{Q}(\mathbf{t}) := \{\boldsymbol{P_S}(\mathbf{t}), \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}\}$, where $\mathcal{Q}(\mathbf{0}) = \mathcal{Q}$. Then we define modulus of elasticity for a quadrilateral diagonal as

$$E_i(\boldsymbol{t}) = m_d \cdot D_{\text{Oddy}}(\mathcal{Q}(\mathbf{t})) + 1, \tag{14}$$

where m_d is a scalar to equilibrate the compromise between element size quality and distortion quality. A greater m_d, the more weight the diagonals will have in the final result, and vice versa. In our experience, $m_d = 0.5$ provides a satisfactory compromise and optimal results. Taking smaller m_d-values reduce deviations from the desired element size at the cost of increasing the distortion. On the other hand, taking greater m_d-values can slightly reduce distortion at the cost of increasing the element size error. Although the diagonals are mainly responsible for the distortion variation, they also have an impact in the variations of the quadrilateral size. Thus, it is convenient to scale their goal length L_i by a factor equal to the ratio between the desired quadrilateral size and the actual quadrilateral size. In this way, diagonals not only reduce distortion, but also help the sides to reach the desired quadrilateral size. We must also take into account that we are following the original formulation for the Oddy distortion, as previously stated. Thus, we must consider the sign of the area of the local parallelogram when evaluating the distortion at each vertex in the quadrilateral. In other words, if we obtain a null or negative area while computing (12), we shall give an arbitrarily large distortion value to that vertex. That way, degenerate quadrilaterals will have a large $E_i(\boldsymbol{t})$ modulus for its diagonals, due to the great need of such diagonals achieving their goal length, and thus fixing the quadrilateral degeneracy. As we are detecting the cases with null or negative area, we avoid accepting the distortion of degenerate quadrilaterals, as well as we make it impossible for asymptotes to appear from division by a null area. Since quadrilateral diagonals have a different modulus of elasticity that its corresponding sizes, we need to study $D_{\text{Oddy}}(\mathcal{Q}(\boldsymbol{t}))$ as a function of $\boldsymbol{t} = (t^x, t^y)$ in order to understand the nature of the function

$$\boldsymbol{f}_{\boldsymbol{P}}^{(i)}(\boldsymbol{t}) := \frac{\boldsymbol{v_i} + \boldsymbol{t}}{\|\boldsymbol{v_i} + \boldsymbol{t}\|} \cdot \frac{(\|\boldsymbol{v_i} + \boldsymbol{t}\| - L_i)E_i(\boldsymbol{t})}{L_i} = \boldsymbol{g}_i(\boldsymbol{t})E_i(\boldsymbol{t})$$

that we can write as the product of a differentiable function $\boldsymbol{g}_i(\boldsymbol{t}) = \frac{\boldsymbol{v_i}+\boldsymbol{t}}{\|\boldsymbol{v_i}+\boldsymbol{t}\|} \cdot \frac{(\|\boldsymbol{v_i}+\boldsymbol{t}\|-L_i)}{L_i}$ and function $E_i(\boldsymbol{t})$ for each $1 \leq i \leq n_d$.

To this end, assume that $\boldsymbol{P_S}(\mathbf{t}) = (P^x + t^x, P^y + t^y), \boldsymbol{J} = (J^x, J^y), \boldsymbol{P_i} = (P_i^x, P_i^y), \boldsymbol{K} = (K^x, K^y)$. Then, to evaluate $D_{\text{Oddy}}(\mathcal{Q}(\boldsymbol{t}))$ we will use the following four functions (see (12)):

1. $D_{\text{Oddy},\boldsymbol{P_S}}(P_i^x, J^x, P^x + t^x, P_i^y, J^y, P^x + t^y)$,
2. $D_{\text{Oddy},\boldsymbol{J}}(K^x, P_i^x, J^x, K^y, P_i^y, J^y)$,
3. $D_{\text{Oddy},\boldsymbol{P_i}}(P^x + t^x, K^x, P_i^x, P^y + t^y, K^y, P_i^y)$,
4. $D_{\text{Oddy},\boldsymbol{K}}(J^x, P^x + t^x, K^x, J^y, P^y + t^y, K^y)$.

Thus, $D_{\text{Oddy}}(\mathcal{Q}(t^x, t^y))$ is computing as the maximum value against the above four functions, where 1,3 and 4 depends on (t^x, t^y) and 2 is a constant function. In consequence, $D_{\text{Oddy}}(\mathcal{Q}(t^x, t^y))$ is a $\mathcal{C}^2$-piecewise smooth function [14] and hence it is semi-smooth [15]. Since $D_{\text{Oddy}}(\mathcal{Q}(\mathbf{t}))$ is a semi-smooth function then we can conclude that $E_i(\mathbf{t})$ is also semi-smooth and hence $\boldsymbol{f}_{\boldsymbol{P}}^{(i)}(\boldsymbol{t}) = \boldsymbol{g}_i(\boldsymbol{t})E_i(\boldsymbol{t})$ is a semi-smooth function for each $1 \leq i \leq n_d$.

3.3. Concluding remark

In this section we introduce the Individual Smoothing Procedure (ISP) function $\boldsymbol{f_P}$ as a sum of two class of functions: $\boldsymbol{f}_{\boldsymbol{P}}^{(j)}$ that are $\mathcal{C}^2$-smooth functions for $1 \leq j \leq n_s$ and $\boldsymbol{f}_{\boldsymbol{P}}^{(k)}$, that are semi-smooth for $1 \leq k \leq n_d$. Thus, we can conclude that the Individual Smoothing Procedure (ISP) function $\boldsymbol{f_P}$ is a semi-smooth function.

4. On the practical implementation of the elasticity-based smoothing algorithm

After having presented the values and conditions upon which the proposed smoothing algorithm is based, in this section we proceed to describe how it was implemented.

The smoothing is performed with an iterative approach. At each iteration, vertices interior to the meshing are moved to their smoothed position. The procedure ends when the vertices displacements are below a given threshold in an iteration. A high quality is often already achieved within the first iterations, which permits to perform a partial smoothing from a fixed and low number of iterations. However, the quality of the elements size gets improved as more iterations are completed. Therefore, by iterating until the vertices displacement threshold is met, we can reach optimum quality. It's possible to displace all vertices simultaneously at the end of each iteration (thus allowing a parallel implementation) or, alternatively, for sequential implementations only, each vertex can be displaced immediately once its equilibrium is solved (this makes the rest of vertices in the same iteration to benefit from considering previously solved vertices in their smoothed position, helping reduce the number of needed iterations). The achieved quality is almost identical in the two approaches, although the result can be different (displacing all the vertices in parallel maintains symmetry if the mesh is symmetrical, while displacing vertices one by one as they are solved can lead to lose the symmetry in some cases). At each iteration, we locate all the springs that are connected to each vertex. For example, in figure 8, vertex **A** is shared by 5 quadrilaterals, and so we must consider its equilibrium from 10 springs (5 springs are quadrilateral sides, and the other 5 springs are diagonals connected to the vertex).

The equilibrium at the vertex is equation (6), where $f(\boldsymbol{t})$ is generated by adding the contribution of all springs connected to the vertex (both the ones belonging to sides and the ones from diagonals, expression (7)), with the only difference that $E_i(\boldsymbol{t})$ is taken from (8) for quadrilateral sides, and from (14) for diagonals, and also with L_i being the user-specified desired element size in the case of sides, or the ideal length in the case of diagonals (obtained from the best candidate in (B.16)). From equation (6) we get the $\boldsymbol{t}$ value that transforms the vertex into its smoothed position. This equation can be solved by *Newton-Raphson*, building the Jacobian matrix as the sum as the contribution from sides springs -expression (9)- and from diagonals springs -expression (15). Proceeding in this way for all vertices, we complete an iteration, and we continue until all $\boldsymbol{t}$ vectors for all vertices are below a threshold. Pseudocode for the method is shown at algorithm 1 for the sequential version in which each vertex is displaced as soon as its equilibrium is solved, and at algorithm 2 for the parallel version that displaces all the vertices simultaneously at the end of each iteration.

Now, in order to implement the proposed elasticity-based smoothing algorithm we need to compute for each $\boldsymbol{P} \in \mathcal{V}(\mathfrak{Q}(\Omega))$ the value $\boldsymbol{t}^*$ for which $\boldsymbol{f}_{\boldsymbol{P}}(\boldsymbol{t}^*) = \boldsymbol{0}$. Since the map $\boldsymbol{f}_{\boldsymbol{P}}$ is a semi-smooth function then in order to use the Newton method to compute $\boldsymbol{t}^*$we need to substitute the classical derivative by a generalized one. More precisely, we use the so-called the B-subdifferential ("B" for Bouligard) and the Clarke's generalized Jacobian (see Section 2.1 in [15]) that are defined below.

Definition 4.1. *Let $V \subset \mathbb{R}^n$ be an open set and $\boldsymbol{f} : V \longrightarrow \mathbb{R}^m$ be Lipschitz continuous near $\boldsymbol{x} \in V$. The set*

$$\partial_B \boldsymbol{f}(\boldsymbol{x}) = \left\{ M \in \mathbb{R}^{n\times m} : \ \textit{exists } (\boldsymbol{x}_k) \subset \mathrm{Dom}(\boldsymbol{f}) : \boldsymbol{x}_k \to \boldsymbol{x} \textit{ and } D\boldsymbol{f}(\boldsymbol{x}_k) \to M \right\}$$

is called the B-subdifferential of $\boldsymbol{f}$ at $\boldsymbol{x}$. Moreover, Clarke's generalized Jacobian of $\boldsymbol{f}$ at $\boldsymbol{x}$ is the convex hull of the B-subdifferential of $\boldsymbol{f}$ at $\boldsymbol{x}$, that is,

$$\partial \boldsymbol{f}(\boldsymbol{x}) = \mathrm{co}(\partial_B \boldsymbol{f}(\boldsymbol{x})).$$

It is possible to prove that if $\boldsymbol{f}$ is continuously differentiable in a neighbourhood of $\boldsymbol{x}$ then

$$\partial \boldsymbol{f}(\boldsymbol{x}) = \partial_B \boldsymbol{f}(\boldsymbol{x}) = \{D\boldsymbol{f}(\boldsymbol{x})\}$$

(see Proposition 2.2 (e) in [15]).

Moreover, in the case of piecewise differentiable maps it can be shown the following result (see Proposition 4.3.1 in [14]).

Proposition 4.2. *Let V be an open set in $\mathbb{R}^n$ and $\boldsymbol{f} : V \longrightarrow \mathbb{R}^m$ is a piecewise $\mathcal{C}^1$-function with $\mathcal{C}^1$-selection functions $\boldsymbol{f}_i : \mathcal{O} \longrightarrow \mathbb{R}^m$ $1 \leq i \leq k$ at $\boldsymbol{x}_0 \in \mathcal{O} \subset V$, that is, $\boldsymbol{f}(\boldsymbol{x}) \in \{\boldsymbol{f}_1(\boldsymbol{x}), \ldots, \boldsymbol{f}_k(\boldsymbol{x})\}$ for all $\boldsymbol{x} \in U$. Then*

$$\partial \boldsymbol{f}(\boldsymbol{x}_0) = \text{co}\{D\boldsymbol{f}_i(\boldsymbol{x}_0) : i \in I_{\boldsymbol{f}}(\boldsymbol{x}_0)\},$$

where

$$I_{\boldsymbol{f}}(\boldsymbol{x}_0) = \{1 \leq i \leq k : \boldsymbol{x}_0 \in \text{cl}(\text{int}\{\boldsymbol{x} \in \mathcal{O} : \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}_i(\boldsymbol{x})\})\}.$$

We have also available some basic calculus rules (see 2.3 in [16]).

295 **Proposition 4.3.** *Let V be an open set in $\mathbb{R}^n$ and $f, g : V \longrightarrow \mathbb{R}$ Assume that f and f are Lipschitz near $\boldsymbol{x} \in V$. Then the following statements hold.*

(a) $\partial(s\,f)(\boldsymbol{x}) = s\,\partial f(\boldsymbol{x})$ for any scalar s.

(b) $\partial(f+g)(\boldsymbol{x}) \subset \partial f(\boldsymbol{x}) + \partial g(\boldsymbol{x})$ and if either f or g is differentiable at $\boldsymbol{x}$ then $\partial(f+g)(\boldsymbol{x}) = \partial f(\boldsymbol{x}) + \partial g(\boldsymbol{x})$ holds.

300 *(c) The map $(f \cdot g)(\boldsymbol{x}) = f(\boldsymbol{x})g(\boldsymbol{x})$ is Lipschitz near $\boldsymbol{x} \in V$ and $\partial(f \cdot g)(\boldsymbol{x}) \subset f(\boldsymbol{x})\,\partial g(\boldsymbol{x}) + g(\boldsymbol{x})\,\partial f(\boldsymbol{x})$.*

(d) If $g(\boldsymbol{x}) \neq 0$ then the map $\left(\frac{f}{g}\right)(\boldsymbol{x}) = \frac{f(\boldsymbol{x})}{g(\boldsymbol{x})}$ is Lipschitz near $\boldsymbol{x} \in V$ and

$$\partial\left(\frac{f}{g}\right)(\boldsymbol{x}) \subset \frac{g(\boldsymbol{x})\,\partial f(\boldsymbol{x}) - f(\boldsymbol{x})\,\partial g(\boldsymbol{x})}{g^2(\boldsymbol{x})}.$$

From Proposition 4.3 (b) and (c) we obtain the following.

Corollary 4.4. *The Individual Smoothing Procedure (ISP) function $\boldsymbol{f}_{\boldsymbol{P}}$ satisfies*

$$\partial \boldsymbol{f}_{\boldsymbol{P}}(\boldsymbol{t}) = \{D\boldsymbol{f}_{\boldsymbol{P}}^{(s)}(\boldsymbol{t})\} + \partial \boldsymbol{f}_{\boldsymbol{P}}^{(d)}(\boldsymbol{t}).$$

Moreover, since $\boldsymbol{f}_{\boldsymbol{P}}^{(d)}(\boldsymbol{t}) = \sum_{k=1}^{n_d} \boldsymbol{g}_i(\boldsymbol{t}) E_i(\boldsymbol{t})$ we have

$$\partial \boldsymbol{f}_{\boldsymbol{P}}^{(d)}(\boldsymbol{t}) \subset \sum_{k=1}^{n_d} E_i(\boldsymbol{t})\,\{D\boldsymbol{g}_i(\boldsymbol{t})\} + \boldsymbol{g}_i(\boldsymbol{t})\,\partial E_i(\boldsymbol{t})$$

This corollary allows to introduce the semi-smooth Newton method to compute $\mathbf{t}^*$ such that $\boldsymbol{f_P}(\boldsymbol{t}^*) = \mathbf{0}$ holds.

305 **Semi-smooth Newton Method**

1. Choose an initial $\boldsymbol{t}_0$ and set $k = 0$.
2. If $\boldsymbol{f_P}(\boldsymbol{t_k}) = \mathbf{0}$ then STOP.
3. Choose $M_k \in \partial \boldsymbol{f}_{\boldsymbol{P}}^{(d)}(\boldsymbol{t_k})$ and compute $\boldsymbol{s_k}$ from $(D\boldsymbol{f}_{\boldsymbol{P}}^{(s)}(\boldsymbol{t_k}) + M_k)\,\boldsymbol{s_k} = -\boldsymbol{f_P}(\boldsymbol{t_k})$.
4. Set $\boldsymbol{t_{k+1}} = \boldsymbol{t_k} + \boldsymbol{s_k}$ and goto to step 2.

In our practical implementation of the semi-smooth Newton method we use a matrix M_k constructed as follows. Assuming that $E_i(\boldsymbol{t})$ is a differentiable function of $\boldsymbol{t}$ then the Jacobian matrix $D\boldsymbol{f}_{\boldsymbol{P}}^{(d)}(\boldsymbol{t})$ is constructed by using

$$D\boldsymbol{f}_{\boldsymbol{P}}^{(d)}(\boldsymbol{t}) = \sum_{k=1}^{n_d} E_i(\boldsymbol{t})\, D\boldsymbol{g}_i(\boldsymbol{t}) + \boldsymbol{g}_i(\boldsymbol{t})\, DE_i(\boldsymbol{t}) \tag{15}$$

where

$$DE_i(\boldsymbol{t}) = \begin{pmatrix} \frac{\partial E_i(\boldsymbol{t})}{\partial t^x} & \frac{\partial E_i(\boldsymbol{t})}{\partial t^y} \end{pmatrix}$$

Then we use $M_k := M(\boldsymbol{t_k})$ for a function M defined by

$$M(\boldsymbol{t}) := \sum_{k=1}^{n_d} E_i(\boldsymbol{t})\, D\boldsymbol{g}_i(\boldsymbol{t}) + \boldsymbol{g}_i(\boldsymbol{t})\, \delta D_{\text{Oddy}}(\boldsymbol{t})\, \boldsymbol{t}^T,$$

where $\delta D_{\text{Oddy}}(\boldsymbol{t})$ is defined as follows. From Proposition 3.1 we known the existence of a minimum value $D_{\text{Oddy}}(\mathcal{Q}(\boldsymbol{m}^*))$ associated to the diagonal $\boldsymbol{P_O(m^*)P_i}$ used to compute the value L_i, in the diagonal case. Now, we consider the quadrilateral

$$\mathcal{Q}(\boldsymbol{t}) = \{\boldsymbol{P_S}(\boldsymbol{t}), \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}\},$$

recall that $\boldsymbol{P_S}(\boldsymbol{t}) = \boldsymbol{P} + \boldsymbol{t}$, then we define

$$\delta D_{\text{Oddy}}(\boldsymbol{t}) := \frac{D_{\text{Oddy}}(\mathcal{Q}(\boldsymbol{m}^*)) - D_{\text{Oddy}}(\mathcal{Q}(\boldsymbol{t}))}{\|\boldsymbol{P_O(m^*)P_S}(\boldsymbol{t})\|},$$

310 by using that $\boldsymbol{P_O(m^*)}$ provides a minimum distortion, and this lets us use $\delta D_{\text{Oddy}}(\boldsymbol{t})$ as a measure of the distortion variation in $\boldsymbol{t}$. Our implementation uses the following semi-smooth method:

Implemented semi-smooth Newton Method

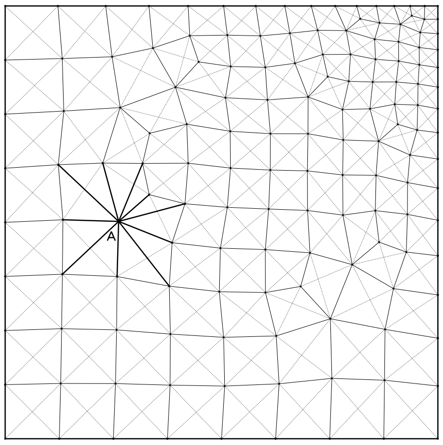


Figure 8: Smoothing algorithm. The springs considered at the equilibrium in A have been drawn with a wider width.

1. Choose an initial $\boldsymbol{t}_0$ and set $k = 0$.
2. If $\boldsymbol{f_P}(\boldsymbol{t_k}) = \mathbf{0}$ then STOP.
3. Compute $\boldsymbol{s_k}$ from $(D\boldsymbol{f}^{(s)}_{\boldsymbol{P}}(\boldsymbol{t_k}) + M(\boldsymbol{t_k}))\,\boldsymbol{s_k} = -\boldsymbol{f_P}(\boldsymbol{t_k})$.
4. Set $\boldsymbol{t_{k+1}} = \boldsymbol{t_k} + \boldsymbol{s_k}$ and goto to step 2.

4.1. A comparative analysis from different smoothing algorithms

We show a comparative analysis of the results obtained from different smoothing algorithms. A set of examples can be seen in figures 9 to 16.

We have measured the Oddy distortion, the side size error (relative in percentage respect of the desired size), and the growth error (this last one in areas with structured meshing only, as it's truly meaningful in consecutive colinear edges —which occur mainly when there are four edges per vertex, that is, structured meshing). The growth error has been measured by comparing the ratio between the lengths of two consecutive colinear edges, and the ratio that they should have if their lengths were the desired ones. The result is expressed as the growth angle error, in degrees. Table 1 shows all these quality measurements, for the original mesh without smoothing (figures 9 and 10), the original Giuliani algorithm [6] (figures 11a, 12a, and 13a), the Giuliani algorithm as modified by Sarrate and Huerta [3] (figures 11b, 12b,

```
Version displacing each vertex as soon as its equilibrium is solved

 input: {vertex list},
        {quadrilateral list},
        {desired element size (per vertex)}

 repeat
   foreach P ← interior vertex in {vertex list}
     t ← [0,0]

       repeat
         f ← [0,0]
         J ← [0,0,0,0]
         foreach A ← quadrilateral side connected to P
           $L_i$ ← desired A length (from user input)
           f ← f + f(A,$L_i$,t) —evaluating equation (7)
           J ← J + J(A,$L_i$,t) —evaluating equation (9)
         foreach D ← quadrilateral diagonal connected to P
           m ← best $m_i$ candidate from (B.16)
           $L_i$ ← Ideal length for D according to scalar m
           (optional) $L_i$ ← $L_i$·mean($L_i$ of sides)/mean(side lengths)
           f ← f + f(D,$L_i$,t) —evaluating equation (7) with (14)
           J ← J + J(D,$L_i$,t) —evaluating equation (15)
         $t_{new}$ ←  solve the system J(t)·($t_{new}$ − t) = −f(t)
         step ← $t_{new}$ − t
         t ← $t_{new}$
       while (norm(f) > $threshold_{sol}$) AND (norm(step) > $threshold_{step}$)

     P ← P + t

 while max( norm(t) ) > $threshold_{smoothing}$
```

Listing 1: Proposed smoothing algorithm (sequential).

```
Version with simultaneous displacement of all vertices

 *input*: {vertex list},
         {quadrilateral list},
         {desired element size (per vertex)}

 Ps ← {smoothed vertex list} ← {vertex list}

 *repeat*
   *foreach* P ← interior vertex *in* {vertex list}
     t ← [0,0]

       *repeat*
         f ← [0,0]
         J ← [0,0,0,0]
         *foreach* A ← quadrilateral side connected to P
           $L_i$ ← desired A length (from user *input*)
           f ← f + f(A,$L_i$,t) —evaluating equation (7)
           J ← J + J(A,$L_i$,t) —evaluating equation (9)
         *foreach* D ← quadrilateral diagonal connected to P
           m ← best $m_i$ candidate from (B.16)
           $L_i$ ← Ideal length for D according to scalar m
           (optional) $L_i$ ← $L_i$·mean($L_i$ of sides)/mean(side lengths)
           f ← f + f(D,$L_i$,t) —evaluating equation (7) with (14)
           J ← J + J(D,$L_i$,t) —evaluating equation (15)
         $t_{new}$ ←  *solve* the system J(t)·($t_{new}$ − t) = −f(t)
         step ← $t_{new}$ − t
         t ← $t_{new}$
       *while* (norm(f) > $threshold_{sol}$) AND (norm(step) > $threshold_{step}$)

     Ps[P] ← P + t

   *foreach* P ← interior vertex *in* {vertex list}
     P ← Ps[P]

 *while* max( norm(t) ) > $threshold_{smoothing}$
```

Listing 2: Proposed smoothing algorithm (parallel).

	Average distortion	Distortion 99th percentile	Average size error	Sides with error $\leqslant 10\%$	Maximum growth error	Absolute value average of growth error
No smoothing	0.31	4.27	8.69%	68%	18.98^o	0.60^o
Giuliani smoothing	0.16	1.56	7.77%	72%	3.75^o	0.39^o
Giuliani smoothing (with modifications from [3])	0.14	1.65	9.33%	64%	4.97^o	0.72^o
Product minimization (reinterpreting [7])	0.18	0.90	8.10%	69%	5.30^o	0.92^o
Proposed method	0.15	1.04	7.35%	75%	2.65^o	0.37^o

Table 1: Comparative analysis of quality measures after each studied algorithm. The proposed algorithm is optimal, showing a remarkable reduction at the growth error (higher fidelity to size gradients required by the user). It also reduces the size error from the other algorithms, and achieves an optimal Oddy distortion (significantly better than the Giuliani-based methods, and on a par with the distortion-size product minimization method).

and 13b), the method with the distortion by size error product minimization (reinterpreting Gargallo, Roca and Sarrate [7] with the Oddy distortion, figures 14a, 15a, and 16a), as well as our proposed spring-based algorithm (figures 14b, 15b, and 16b). The results show that the proposed algorithm provides an optimal response to our requirements. Regarding Oddy distortion, the obtained quality is better than that of the Giuliani-based algorithms (the average is similar, but the 99th percentile of the distortion gets reduced in about 40% —an important aspect, as those other algorithms should achieve an extraordinarily low distortion given that they don't take care of meeting the user desired element size). The distortion from the proposed method is very similar to the one provided by the product minimization algorithm (the average is reduced from 0.18 to 0.15, while the 99th percentile is slightly increased from 0.90 to 1.04). At the edge size error, our algorithm gets the best result of all the studied methods, with an average relative error of 7.35%, and managing that 75% of edges have a relative error below 10%. The other methods don't achieve these results. The low value of the growth error is also noteworthy, being about the half of the maximum error from the other algorithms (let's recall that the example used in the comparison has a size gradient required by the user, and therefore the growth error is also an important quality measurement). With a maximum error of 2.65° and with 0.37° as the average of the absolute value of the error, it is the algorithm that best maintains desired size gradients. A quick glimpse at table 1 could give the false impression that the original Giuliani algorithm provides a quality not too far away from our proposed method, even if its results are worse[1]. However, as said before, it tends to make all elements of uniform size, and this doesn't substantially affect the quality measurements but can be visually appreciated at figure 11a (the size gradient desired by the user has lost its presence to a large degree). In conclusion, it can be observed that the proposed algorithm provides an excelent response to the required quality criteria, as we wished to decrease as much as possible the size error, respecting the size gradients desired by the user, and with an as low as possible quadrilateral distortion. The

[1]With the exception of the average distortion, which is slightly reduced.

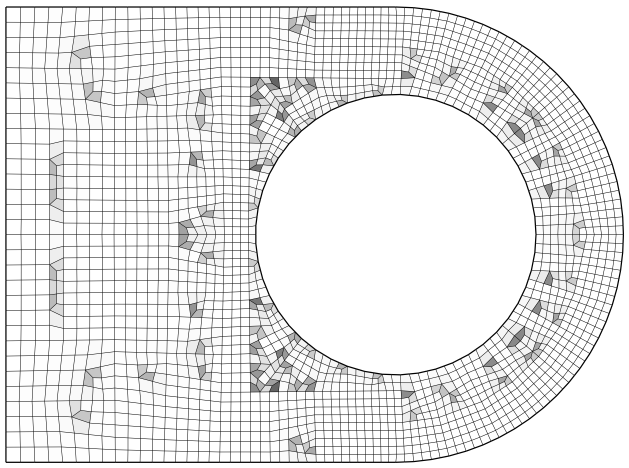
springs model, with the diagonals controlling the element distortion, and the edges taking care of the element size, achieves a great result meeting all these requirements, without neglecting any of them. Moreover, the algorithm also removes the "noise patterns" like those introduced by methods like Giuliani with the modifications from [3] (areas with elements sizes substantially different from the size they should have, creating visually noticeable patterns). By comparing figures 11b and 14b, it can be observed how the proposed algorithm (second figure) has eliminated the presence of these patterns, which are noticeable in the first figure. In addition to quality, the algorithm can be implemented very efficiently both in *CPU* and in *GPU*, thanks to its parallelization possibilities. In each iteration, the smoothing of each joint can be made independent from the rest of joints, which allows for a massive parallel implementation without communication between processing cores (communication is only necessary at the end of each iteration).

5. Conclusions and future work

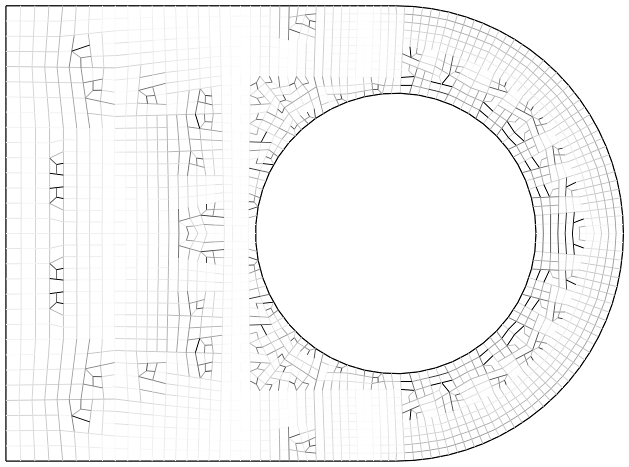
The proposed algorithm for *smoothing* post-processing, based on a spring system, provided optimal results in our experiments, being the one delivering the highest quality measurements from all the alternatives that we tested. It's suitable for both sequential and parallel execution, and it's easy to implement. We based the formulation on the *Oddy distortion*, but the development of the algorithm allows for adaptations to other distortion criteria, by just modifying the ideal length expressions for the quadrilateral diagonals accordingly (expressions (B.1) to (B.16)). The simplicity of its parallel implementation (algorithm 2) suggests promising future work in the area of multi-core *CPU* and *GPU* programming, and our day-to-day use of the algorithm in our mesh-based simulations may open opportunities for improvement.

6. Bibliography

[1] S. J. Owen, A survey of unstructured mesh generation technology, in: Proceedings, 7th International Meshing Roundtable, 1998, pp. 239–267.



(a) Oddy distortion in the mesh before smoothing. The greater the distortion, the darker the shading (average distortion=0.31; 99th percentile distortion=4.27).



(b) Relative side size error in mesh before smoothing. The greater the error, the darker the shading (average relative error=8.69%; the 68% of edges have a relative error lower or equal than 10%).

Figure 9: Quality measurements in the original mesh.

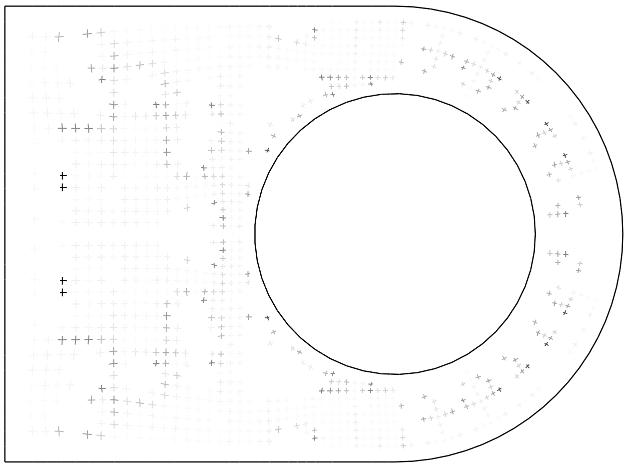


Figure 10: Error of the size variation in structured mesh areas, before smoothing, expressed as growth slope error in degrees. The greater the error, the darker the shading (maximum=18.98°; average of absolute value=0.60°).

[2] S. H. Lo, Finite Element Mesh Generation, CRC Press, 2015.

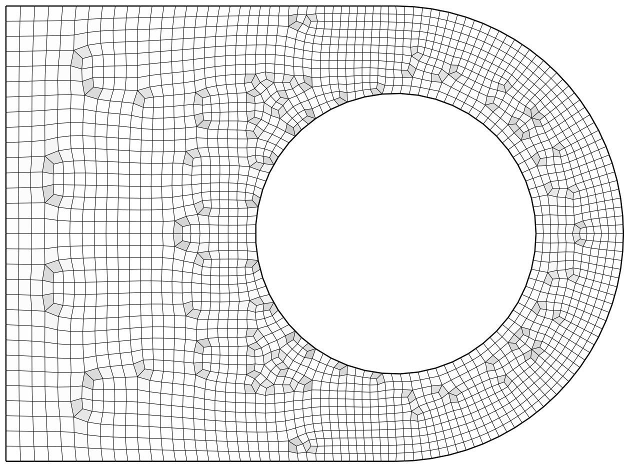
[3] J. Sarrate, A. Huerta, Efficient unstructured quadrilateral mesh generation, International Journal for Numerical Methods in Engineering 49 (10) (2000) 1327–1350.

[4] M. Bastian, B. Q. Li, An efficient automatic mesh generator for quadrilateral elements implemented using C++, Finite Elements in Analysis and Design 39 (9) (2003) 905–930.

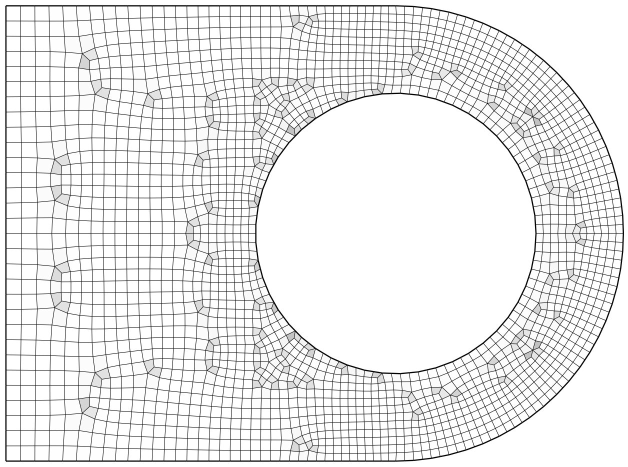
[5] T. D. Blacker, M. B. Stephenson, Paving: A new approach to automated quadrilateral mesh generation, International Journal for Numerical Methods in Engineering 32 (4) (1991) 811–847.

[6] S. Giuliani, An algorithm for continuous rezoning of the hydrodynamic grid in arbitrary lagrangian-eulerian computer codes, Nuclear Engineering and Design 72 (2) (1982) 205–212.

[7] A. Gargallo, X. Roca, J. Sarrate, A surface mesh smoothing and untangling method independent of the CAD parameterization, Computational mechanics 53 (4) (2014) 587–609.

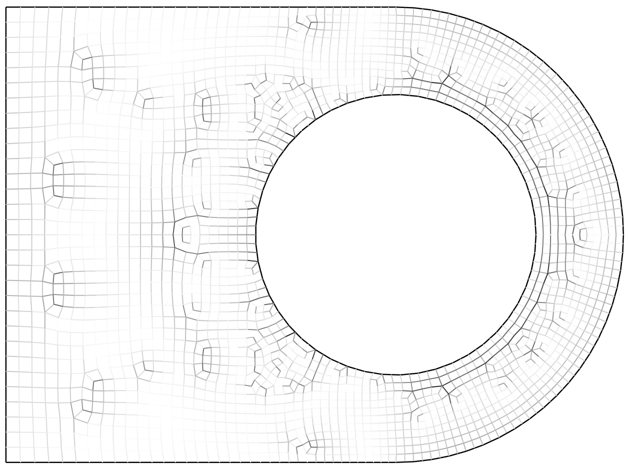


(a) Oddy distortion after Giuliani smoothing. The greater the distortion, the darker the shading (average distortion=0.16; 99th percentile distortion=1.56).

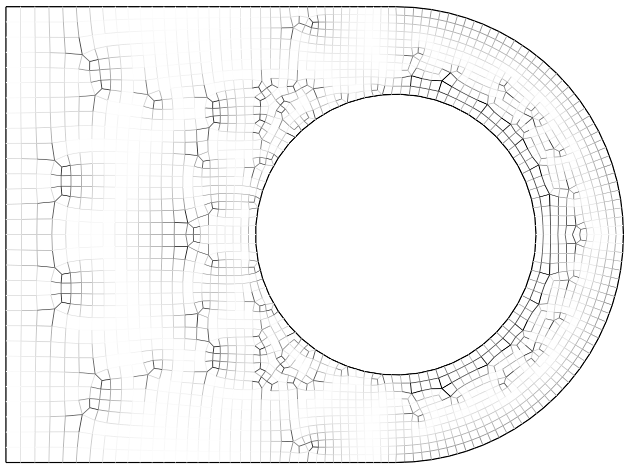


(b) Oddy distortion after Giuliani smoothing with the modifications by [3]. The greater the distortion, the darker the shading (average distortion=0.14; 99th percentile distortion=1.65).

Figure 11: Oddy distortion after smoothing methods that ignore element size.

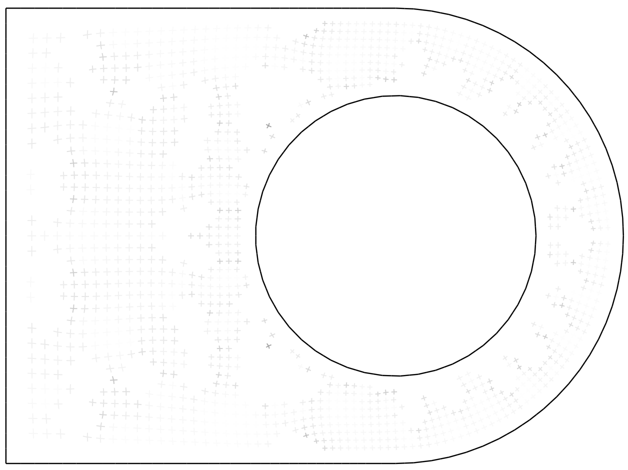


(a) Relative side size error after Giuliani smoothing. The greater the error, the darker the shading (average relative error=7.77%; the 72% of edges have a relative error lower or equal than 10%).



(b) Relative side size error after Giuliani smoothing with the modifications by [3]. The greater the error, the darker the shading (average relative error=9.33%; the 64% of edges have a relative error lower or equal than 10%).

Figure 12: Relative side size error after smoothing methods that ignore element size.


(a) Error of the size variation in structured mesh areas after Giuliani smoothing, expressed as growth slope error in degrees. The greater the error, the darker the shading (maximum=3.75°; average of absolute value=0.39°).

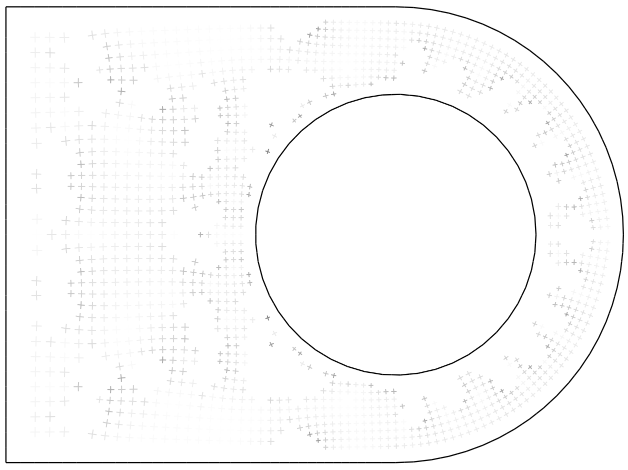
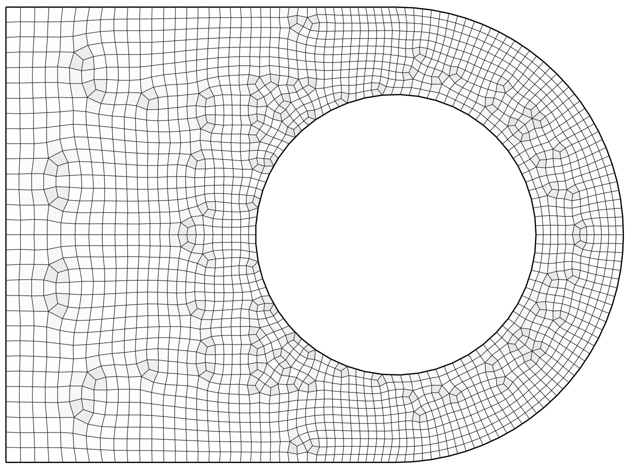

(b) Error of the size variation in structured mesh areas after Giuliani smoothing with the modifications by [3], expressed as growth slope error in degrees. The greater the error, the darker the shading (maximum=4.97°; average of absolute value=0.72°).

Figure 13: Error of the size variation after smoothing methods that ignore element size.


(a) Oddy distortion after smoothing by minimization of the product of the area size error by the distortion average. The greater the distortion, the darker the shading (average distortion=0.18; 99th percentile distortion=0.90).

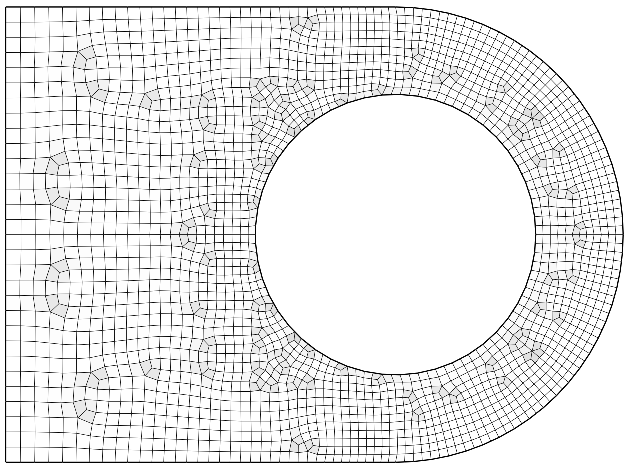
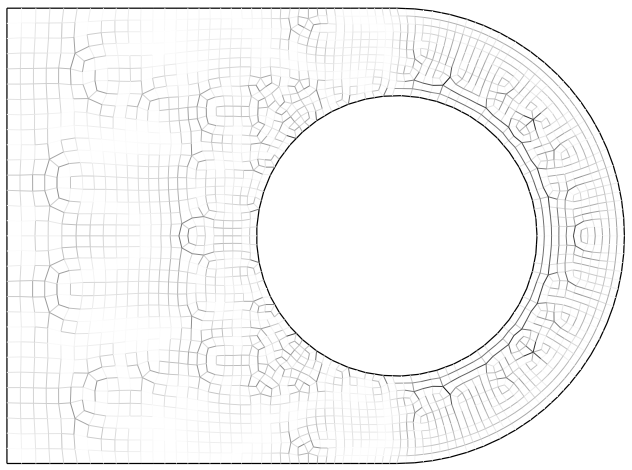

(b) Oddy distortion after smoothing by the proposed algorithm. The greater the distortion, the darker the shading (average distortion=0.15; 99th percentile distortion=1.04).

Figure 14: Oddy distortion after smoothing methods that take care of element size.


(a) Relative side size error after smoothing by minimization of the product of the area size error by the distortion average. The greater the error, the darker the shading (average relative error=8.10%; the 69% of edges have a relative error lower or equal than 10%).

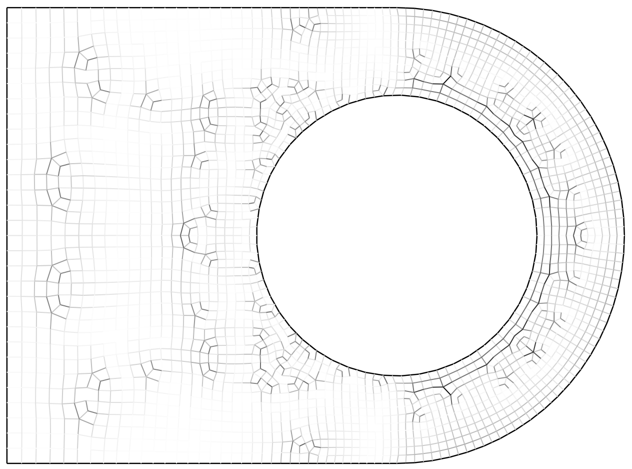
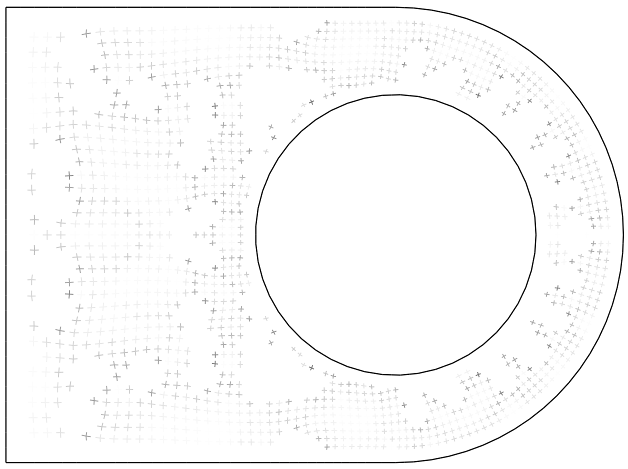

(b) Relative side size error after smoothing by the proposed algorithm. The greater the error, the darker the shading (average relative error=7.35%; the 75% of edges have a relative error lower or equal than 10%).

Figure 15: Relative side size error after smoothing methods that take care of element size.


(a) Error of the size variation in structured mesh areas after smoothing by minimization of the product of the area size error by the distortion average, expressed as growth slope error in degrees. The greater the error, the darker the shading (maximum=5.30°; average of absolute value=0.92°).

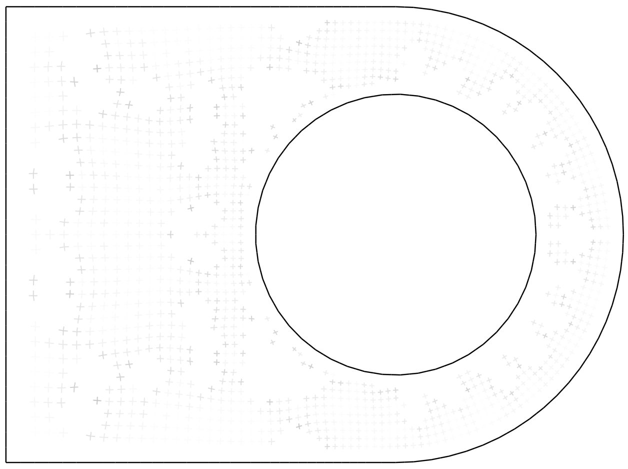

(b) Error of the size variation in structured mesh areas after smoothing by the proposed algorithm, expressed as growth slope error in degrees. The greater the error, the darker the shading (maximum=2.65°; average of absolute value=0.37°).

Figure 16: Error of size variation after smoothing methods that take care of element size.

[8] P. M. Knupp, Algebraic mesh quality metrics, SIAM J. Sci. Comput. 23 (1) (2001) 193–218.

[9] R. Lohner, K. Morgan, O. C. Zienkiewicz, Adaptive grid refinement for the compressible Euler equations, in: Accuracy estimates and adaptive refinements in finite element computations, Wiley, 1986, pp. 281–297.

[10] A. Oddy, J. Goldak, M. McDill, M. Bibby, A distortion metric for iso-parametric finite elements, CSME Transactions 12 (4) (1988) 213–217.

[11] C. K. Lee, S. H. Lo, A new scheme for the generation of a graded quadrilateral mesh, Computers & Structures 52 (5) (1994) 847–857.

[12] S. A. Canann, J. R. Tristano, M. L. Staten, An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes, in: Proceedings of the 7th International Meshing Roundtable, 1998, pp. 479–494.

[13] J. Sarrate, A. Coll, Minimización de la distorsión de mallas formadas por cuadriláteros o hexaedros, Revista internacional de métodos numéricos para cálculo y diseño en ingeniería 23 (1) (2007) 55–76.

[14] S. Sholtes, Introduction to Piecewise Differentiable Equations, springer briefs in optimization Edition, Springer-Verlag, 2012.

[15] M. Ulbrich, Semismooth Newton Methods for Variational Inequalities and Constrained Optimization Problems in Function Spaces, MPS-SIAM Series on Optimization, Society for Industrial & Applied Mathematics, 2011.

[16] F. H. Clarke, Optimization and Nonsmooth Analysis, Society for Industrial and Applied Mathematics, 1990.

Appendix A. Proof of Lemma 2.1

Recall that

$$\boldsymbol{g}_{\boldsymbol{P}}^{(i)}(\boldsymbol{t}) = (\boldsymbol{v_i}+\boldsymbol{t})\cdot\frac{(\|\boldsymbol{v_i}+\boldsymbol{t}\|-L_i)}{L_i\|\boldsymbol{v_i}+\boldsymbol{t}\|},$$

then

$$\begin{aligned} D\boldsymbol{g}_{\boldsymbol{P}}^{(i)}(\boldsymbol{t}) &= \frac{(\|\boldsymbol{v_i}+\boldsymbol{t}\|-L_i)}{L_i\|\boldsymbol{v_i}+\boldsymbol{t}\|}D((\boldsymbol{v_i}+\boldsymbol{t})) + (\boldsymbol{v_i}+\boldsymbol{t})D\left(\frac{(\|\boldsymbol{v_i}+\boldsymbol{t}\|-L_i)}{L_i\|\boldsymbol{v_i}+\boldsymbol{t}\|}\right)\\ &= \frac{(\|\boldsymbol{v_i}+\boldsymbol{t}\|-L_i)}{L_i\|\boldsymbol{v_i}+\boldsymbol{t}\|}\begin{pmatrix}1&0\\0&1\end{pmatrix} + \frac{1}{L_i}(\boldsymbol{v_i}+\boldsymbol{t})D\left(\frac{(\|\boldsymbol{v_i}+\boldsymbol{t}\|-L_i)}{\|\boldsymbol{v_i}+\boldsymbol{t}\|}\right).\end{aligned}$$

Now,

$$D\left(\frac{(\|\boldsymbol{v_i}+\boldsymbol{t}\|-L_i)}{\|\boldsymbol{v_i}+\boldsymbol{t}\|}\right) = \frac{\|\boldsymbol{v_i}+\boldsymbol{t}\|D(\|\boldsymbol{v_i}+\boldsymbol{t}\| + (\|\boldsymbol{v_i}+\boldsymbol{t}\|-L_i)D(\|\boldsymbol{v_i}+\boldsymbol{t}\|)}{\|\boldsymbol{v_i}+\boldsymbol{t}\|^2} = \frac{L_iD(\|\boldsymbol{v_i}+\boldsymbol{t}\|)}{\|\boldsymbol{v_i}+\boldsymbol{t}\|^2}.$$

To conclude the proof, we take into account that

$$D(\|\boldsymbol{v_i}+\boldsymbol{t}\|) = D\left(\sqrt{(\boldsymbol{v_i}+\boldsymbol{t})^T(\boldsymbol{v_i}+\boldsymbol{t})}\right) = \frac{(\boldsymbol{v_i}+\boldsymbol{t})^T}{\|\boldsymbol{v_i}+\boldsymbol{t}\|},$$

hence

$$D\boldsymbol{g}_{\boldsymbol{P}}^{(i)}(\boldsymbol{t}) = \frac{(\|\boldsymbol{v_i}+\boldsymbol{t}\|-L_i)}{L_i\|\boldsymbol{v_i}+\boldsymbol{t}\|}\begin{pmatrix}1&0\\0&1\end{pmatrix} + (\boldsymbol{v_i}+\boldsymbol{t})\frac{(\boldsymbol{v_i}+\boldsymbol{t})^T}{\|\boldsymbol{v_i}+\boldsymbol{t}\|^3}.$$

This proves the lemma.

Appendix B. Proof of Proposition 3.1

Recall that, according to equation (13), the Oddy distortion will be the greatest of the local distortions at each of the four vertices in the quadrilateral. Since $\boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}$ remain in a fixed position, the Oddy distortion at the $\boldsymbol{P_i}$ vertex remains constant, because the angle between the $\boldsymbol{JP_i}$ and the $\boldsymbol{P_iK}$ sides is kept constant. Therefore, to find the goal length for the $\boldsymbol{P_O(m)P_i}$ – *diagonal*, we can ignore the Oddy distortion at $\boldsymbol{P_i}$. Thus we only need to evaluate $D_{\text{Oddy},i}(m) := D_{\text{Oddy},i}(\{\boldsymbol{P_O}(m), \boldsymbol{J}, \boldsymbol{P_i}, \boldsymbol{K}\})$ for $i \in \{\boldsymbol{P}, \boldsymbol{J}, \boldsymbol{K}\}$. As we shall see, the variation of the Oddy distortion at each vertex in the set $\{\boldsymbol{P_O}(m), \boldsymbol{J}, \boldsymbol{K}\}$ as we move m

yields in a cubic polynomial in m. Each cubic polynomial can have a feasible or an unfeasible minimum (because in some cases the minimum can correspond to an absurd geometry). Since the distortion is the maximum from the distortions evaluated at the vertices, we need to compute the intersection of the three cubic curves, in order to take into account the envelope of the maximum distortion.

In conclusion, the optimal position $\boldsymbol{P_O}(m^*)$ will take place at exactly one of these points:

(i) A local minimum from one of the three curves.

(ii) An intersection point between them.

Figure B.17 shows an example in which the minimum Oddy distortion occurs at the intersection between the curves associated to vertices $\boldsymbol{J}$ and $\boldsymbol{K}$. This example illustrates the reason to find the local minima for the three curves and its intersection points in order to obtain the minimum distortion for the quadrilateral. To provide an explicit expression for $D_{\text{Oddy},i}(m)$ for $i \in \{\boldsymbol{P}, \boldsymbol{J}, \boldsymbol{K}\}$ we introduce the following notation.

- We put $\boldsymbol{PP_i} = (PP_i{}^x, PP_i{}^y)$, $\boldsymbol{PJ} = (PJ^x, PJ^y)$, $\boldsymbol{PK} = (PK^x, PK^y)$, $\boldsymbol{JP_i} = (JP_i{}^x, JP_i{}^y)$, $\boldsymbol{KP_i} = (KP_i{}^x, KP_i{}^y)$, $\boldsymbol{JK} = (JK^x, JK^y)$, $\boldsymbol{PJ} = (PJ^x, PJ^y)$ and $\boldsymbol{PK} = (PK^x, PK^y)$.

By using (12) together $\boldsymbol{P_O}(m)$, we can evaluate the Oddy distortion at each vertex as a function of the m or, in other words, as a function of the translation of $\boldsymbol{P}$ along the diagonal. We can arrive to the following expressions, which are the equations of the three curves plotted at figure B.17:

$$D_{\text{Oddy},P}(m) = 2\left[\frac{\left[(PJ^x - mPP_i{}^x)^2 + (PK^x - mPP_i{}^x)^2 + (PJ^y - mPP_i{}^y)^2 + (PK^y - mPP_i{}^y)^2\right]^2}{4\left[m\left[PP_i{}^x\left(PJ^y - PK^y\right) + PP_i{}^y\left(PK^x - PJ^x\right)\right] + PJ^xPK^y - PJ^yPK^x\right]^2} - 1\right], \tag{B.1}$$

$$D_{\text{Oddy},J}(m) = 2\left[\frac{\left[(PJ^x - mPP_i{}^x)^2 + (PJ^y - mPP_i{}^y)^2 + \|\boldsymbol{JP_i}\|^2\right]^2}{4\left(-mPP_i{}^xJP_i{}^y + mPP_i{}^yJP_i{}^x - JP_i{}^xPJ^y + JP_i{}^yPJ^x\right)^2} - 1\right], \tag{B.2}$$

$$D_{\text{Oddy},K}(m) = 2\left[\frac{\left[(PK^x - mPP_i{}^x)^2 + (PK^y - mPP_i{}^y)^2 + \|\boldsymbol{KP_i}\|^2\right]^2}{4\left(mPP_i{}^xKP_i{}^y - mPP_i{}^yKP_i{}^x + KP_i{}^xPK^y - KP_i{}^yPK^x\right)^2} - 1\right]. \tag{B.3}$$

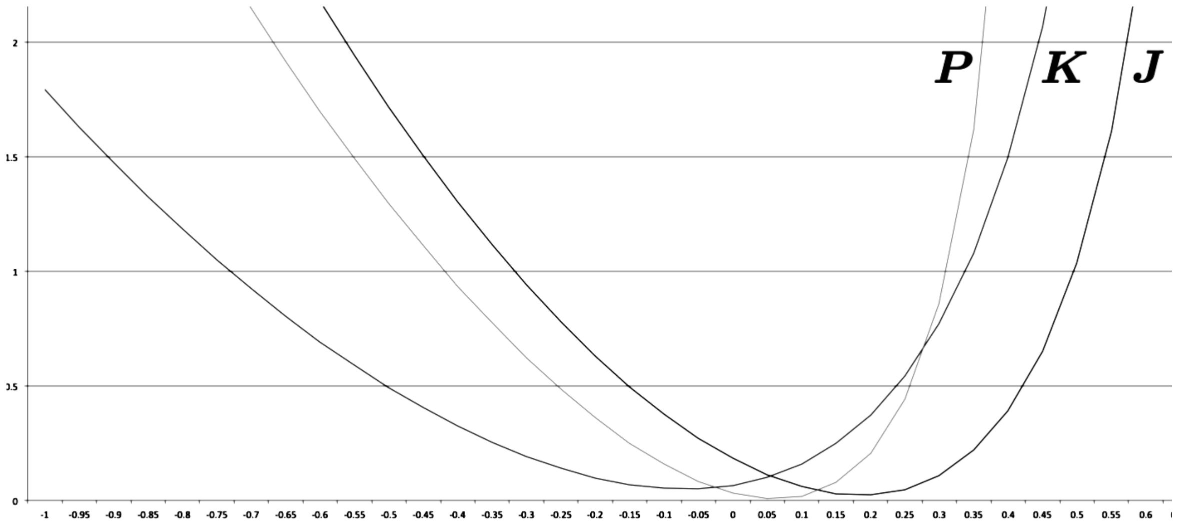


Figure B.17: Variation of the Oddy distortion at the $\boldsymbol{P}, \boldsymbol{J}, \boldsymbol{K}$ vertices as we translate $\boldsymbol{P}$ along the $\boldsymbol{PP_i}$ direction. In this example, the minimum of the maximum distortion occurs at the intersection between curves $\boldsymbol{J}$ and $\boldsymbol{K}$. The horizontal axis represents the m scalar from equation (**??**).

Computing the intersection points

In order to find the candidate values for m that correspond to the intersection of the curves for the local distortion variation at $\boldsymbol{J}$ and $\boldsymbol{K}$, we need to solve the equation

$$D_{\text{Oddy},J}(m) = D_{\text{Oddy},K}(m) \tag{B.4}$$

which, after the substitution of expressions (B.2) y (B.3), implies finding the roots of a cubic equation in the form

$$e_{JK}m^3 + f_{JK}m^2 + g_{JK}m + h_{JK} = 0 \tag{B.5}$$

where

$$e_{JK} = \|\boldsymbol{PP_i}\|^2 (s - r)$$

$$f_{JK} = 2(br - as) + \|\boldsymbol{PP_i}\|^2 (t - u)$$

$$g_{JK} = 2(bu - at) - dr + cs$$

$$h_{JK} = ct - du$$

$$a = PP_i{}^x PJ^x + PP_i{}^y PJ^y$$

$$b = PP_i^x PK^x + PP_i^y PK^y$$

$$c = \|\boldsymbol{JP_i}\|^2 + \|\boldsymbol{PJ}\|^2$$

$$d = \|\boldsymbol{KP_i}\|^2 + \|\boldsymbol{PK}\|^2$$

$$r = PP_i^y JP_i^x - PP_i^x JP_i^y$$

$$s = PP_i^x KP_i^y - PP_i^y KP_i^x$$

$$t = KP_i^x PK^y - KP_i^y PK^x$$

$$u = JP_i^y PJ^x - JP_i^x PJ^y$$

equation that can have three distinct real roots, three multiple real roots, or one real and two non-real roots.

Analogously, we can obtain the m candidates belonging to the intersection between the $\boldsymbol{J}$ and $\boldsymbol{P}$ distortion variation curves, solving the equation

$$D_{\text{Oddy},J}(m) = D_{\text{Oddy},P}(m) \tag{B.6}$$

which leads to finding the roots of another cubic equation

$$e_{JP}m^3 + f_{JP}m^2 + g_{JP}m + h_{JP} = 0 \tag{B.7}$$

where

$$e_{JP} = \|\boldsymbol{PP_i}\|^2 (z - 2r)$$

$$f_{JP} = 2(ar + br - az) + \|\boldsymbol{PP_i}\|^2 (w - 2u)$$

$$g_{JP} = 2(au + bu - aw) + cz - r\left(\|\boldsymbol{PJ}\|^2 + \|\boldsymbol{PK}\|^2\right)$$

$$h_{JP} = cw - u\left(\|\boldsymbol{PJ}\|^2 + \|\boldsymbol{PK}\|^2\right)$$

$$w = PJ^x PK^y - PJ^y PK^x$$

$$z = PP_i^y JK^x - PP_i^x JK^y$$

taking a, b, c, r, u the same values as in equation (B.5). And we also add the m candidates due to the intersection for the $\boldsymbol{K}$ and $\boldsymbol{P}$ curves:

$$D_{\text{Oddy},K}(m) = D_{\text{Oddy},P}(m) \tag{B.8}$$

resulting in the cubic equation

$$e_{KP}m^3 + f_{KP}m^2 + g_{KP}m + h_{KP} = 0 \tag{B.9}$$

where

$$e_{KP} = \|\boldsymbol{PP_i}\|^2 (z - 2s)$$

$$f_{KP} = 2(as + bs - bz) + \|\boldsymbol{PP_i}\|^2 (w - 2t)$$

$$g_{KP} = 2(at + bt - bw) + dz - s\left(\|\boldsymbol{PJ}\|^2 + \|\boldsymbol{PK}\|^2\right)$$

$$h_{KP} = dw - t\left(\|\boldsymbol{PJ}\|^2 + \|\boldsymbol{PK}\|^2\right)$$

taking a, b, d, s, t, w, z the same values as in equations (B.5) and (B.7). It can be observed that equations (B.7) and (B.9) take symmetrical expressions against each other, while equation (B.5) is notably different. This was expected, because (B.7) and (B.9) are the intersections between the distortion curves at $\boldsymbol{P}$ (moving point) and the (fixed) vertex located at one side of the diagonal ($\boldsymbol{J}$ and $\boldsymbol{K}$, respectively), whereas (B.5) is the intersection of the distortion curves at the two fixed vertices, $\boldsymbol{J}$ and $\boldsymbol{K}$. For the moment, considering just the intersection points between the distortion variation curves, we get a maximum of 9 candidate values for m (if the equations (B.5), (B.7) and (B.9) have three distinct real roots each), and a minimum of 3 candidates (if there's a total of three real roots only).

Computing the local minimum

We still have to consider the m candidates that come from the local minima of each curve. To do this we proceed as follows. Assume first that $a^2 - c\|\boldsymbol{PP_i}\|^2 \geqslant 0$, then the curve $D_{\text{Oddy},J}(m)$ reaches local extrema at:

$$m_J^{\pm} = \frac{a \pm \sqrt{a^2 - c\|\boldsymbol{PP_i}\|^2}}{\|\boldsymbol{PP_i}\|^2} \tag{B.10}$$

otherwise, it reaches them at:

$$m_J^{\pm} = 1 \pm \frac{\sqrt{-2a + c + \|\boldsymbol{PP_i}\|^2}}{\|\boldsymbol{PP_i}\|} \tag{B.11}$$

We now consider the case when $b^2 - d\,\|\boldsymbol{PP_i}\|^2 \geqslant 0$ holds. Under this assumption the curve $D_{\text{Oddy},K}(m)$ reaches local extrema at:

$$m_K^{\pm} = \frac{b \pm \sqrt{b^2 - d\,\|\boldsymbol{PP_i}\|^2}}{\|\boldsymbol{PP_i}\|^2} \tag{B.12}$$

otherwise, it reaches them at:

$$m_K^{\pm} = 1 \pm \frac{\sqrt{-2b + d + \|\boldsymbol{PP_i}\|^2}}{\|\boldsymbol{PP_i}\|} \tag{B.13}$$

To conclude we consider the case when $(a+b)^2 - 2\,\|\boldsymbol{PP_i}\|^2 \left(\|\boldsymbol{PJ}\|^2 + \|\boldsymbol{PK}\|^2\right) \geqslant 0$. In this case the curve $D_{\text{Oddy},P}(m)$ reaches local extrema at:

$$m_P^{\pm} = \frac{a + b \pm \sqrt{(a+b)^2 - 2\,\|\boldsymbol{PP_i}\|^2 \left(\|\boldsymbol{PJ}\|^2 + \|\boldsymbol{PK}\|^2\right)}}{2\,\|\boldsymbol{PP_i}\|^2} \tag{B.14}$$

otherwise, it reaches them at:

$$m_P^{\pm} = \frac{-2w \pm z\sqrt{\frac{2z\left[2w(a+b)+z\left(\|\boldsymbol{PJ}\|^2+\|\boldsymbol{PK}\|^2\right)\right]+4\|\boldsymbol{PP_i}\|^2 w^2}{\|\boldsymbol{PP_i}\|^2 z^2}}}{2z} \tag{B.15}$$

Collecting all the candidate values found, we are finally able to obtain the goal length for the diagonal. From equations (B.5), (B.7), (B.9), (B.10), (B.11), (B.12), (B.13), (B.14) and (B.15) we obtain a set of up to 15 candidate values for m (9 from the curves intersections if all roots are distinct and real, and 6 from the local minima). Thus we can conclude that there exists $m^* \in \mathbb{R}$ and $i \in \{P, J, K\}$ such that

$$D_{\text{Oddy}}(m^*) = D_{\text{Oddy},i}(m^*) \leq D_{\text{Oddy}}(m) \tag{B.16}$$

for all $m \in \mathbb{R}$. This concludes the proof of the proposition.