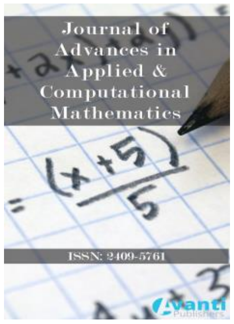
Published by Avanti Publishers

Journal of Advances in Applied & Computational Mathematics

ISSN (online): 2409-5761



Analysis of the Computational Cost of PolyFront: an Algorithm for Planar Triangulation

Nadaniela Egidi, Josephin Giacomini, Luciano Misici, Alessia Perticarini* and Riccardo Piergallini

School of Science and Technology, Mathematics Division, University of Camerino, via Madonna delle Carceri 9, Camerino 62032, Italy

ARTICLE INFO

Article Type: Research Article

Academic Editor: Ahmed Saad Rashed

Keywords:
Mesh generation
Polygon offsetting
Computational cost
Two-dimensional delaunay triangulation

Timeline:
Received: May 29, 2023
Accepted: July 04, 2023
Published: September 22, 2023

Citation: Egidi N, Giacomini J, Misici L, Perticarini A, Piergallini R. Analysis of the computational cost of PolyFront: an algorithm for planar triangulation. J Adv App Comput Math. 2023; 10: 50-64.

DOI: https://doi.org/10.15377/2409-5761.2023.10.5

ABSTRACT

The triangulation of planar domains is a relevant and largely studied problem in many applied sciences. This paper analyzes the computational time of a triangulation algorithm for plane domains with holes, introduced in a previous paper. This algorithm is based on the normal offsetting technique starting from a polygonal approximation of the domain boundary. It is shown that the computational time is linear with respect to the number of vertices of the triangulation. Experimental results confirm the theoretical upper bound obtained for the computational time.

*Corresponding Author
Email: alessia.perticarini@unicam.it
Tel: +(39) 334 8053479

1. Introduction

The generation of a high-quality grid of plane domains is fundamental in many areas of applied sciences, such as engineering and architecture. In [1], a grid generation and optimization framework is proposed, which automatically generates triangular grids over complex surfaces for architectural design. In [2] and [3], a new method based on Coulomb's law is proposed with the same purposes. The construction of high-quality triangulations of plane domains is also fundamental for solving problems in many areas of applied mathematics [4, 5]. In fact, several numerical simulations are based on such triangulations and the quality of the results heavily depends on the quality of the triangulation [6, 7]. In particular, [8] proposes an algorithm generating a guaranteed quality mesh for the curvilinear triangulation of planar domains with piecewise polynomial boundary. In [9], we introduced a new triangulation algorithm named PolyFront (*PF*) based on polygon offsetting, starting from a polygonal approximation of the domain boundary. The constructed offset polygons are used to insert vertices and triangles, in particular, triangles are inserted subsequently to the current mesh as in the paving method [10, 11], and in the advancing front method [12-15]. In the same paper [9], we experimentally showed that this algorithm *PF* produces a good-quality mesh with reduced computational time. An interactive graphical user interface for *PF* was presented and illustrated in [16]. Moreover, the proposed method has been implemented in a software package including also the graphical user interface, that can be downloaded from https://docenti.unicam.it/public/Cad_2D.zip.

In this paper, we analyze the *PF* algorithm and we prove that the computational time of *PF* is $O(n_e n_v)$ when the domain has a polygonal boundary approximation made of n_e edges and the generated triangulation has n_v vertices.

The paper is organized as follows. The main proof is in Sections 2 and 3; in particular, in Section 2 we report and discuss the algorithm *PF* proposed in [9], for the reader's convenience; in Section 3, we analyze the *PF* algorithm and theoretically estimate its computational cost. Then, in Section 4, we confirm by experiments the upper bound of the computational time. Finally, in Section 5, we give some final remarks and conclusions.

2. The Algorithm

For the reader's convenience, we report *PF*, the triangulation algorithm proposed in [9], and we give the computational cost of some steps of *PF*. *PF* constructs a triangulation of a planar connected domain $\Omega \subset \mathbb{R}^2$ with holes, this construction starts from a polygonal approximation of the boundary $\partial\Omega$ of the domain. So, in the following Ω is supposed to have a polygonal boundary, otherwise, it is replaced by a domain having a boundary equal to a polygonal approximation of $\partial\Omega$. Moreover, the proposed algorithm constructs triangulations where the length of the triangle edges is as much as possible near a desired length d. The constructed triangulation is defined by $\mathcal{S}$, the set of triangulation vertices, and $\mathcal{T}$, the set of its triangles.

We begin with some notations. We denote with n_{bv} the number of the triangulation vertices chosen in $\partial\Omega$. A simple polygon P with $k(\geq 3)$ edges is represented by an ordered list of its vertices $P = (v_0, v_1, \dots, v_{k-1})$. The interior of the polygon is denoted with $I(P)$. We associate to each edge E of P a positive real number d_E that gives the distance between two consecutive triangulation vertices chosen on E, of course, d_E is chosen as near as possible to the desired length d. Moreover, we denote with L_E the length of E. If $\Omega \subset \mathbb{R}^2$ has h holes, then it is represented by a list of polygons $(P^E, P_1^I, \dots, P_h^I)$, where $\partial I(P^E)$ is the external boundary $\underline{\text{of } \Omega \text{ and } \partial I(P_i^I),}$ $i = 1,2, \dots, h$ are the boundaries of its h holes. We note that with this representation we have $\Omega = I(P^E) \setminus \cup_{i=1}^{h} I(P_i^I)$, and so we use the following notation $\Omega = D(P^E, P_1^I, \dots, P_h^I)$. Moreover, the vertices of a given polygon P are ordered counterclockwise when P is an external polygon and clockwise otherwise. In particular, when Ω has no hole ($h = 0$) then in its representation we have only the external polygon and so $\Omega = D(P^E)$. For example, Fig. (**1a**) shows a domain Ω with one hole, its representation is $\Omega = D(P^E, P_1^I)$ where the external polygon is $P^E = (v_0, \dots, v_6)$ and the unique internal polygon is $P_1^I = (u_0, \dots, u_3)$.

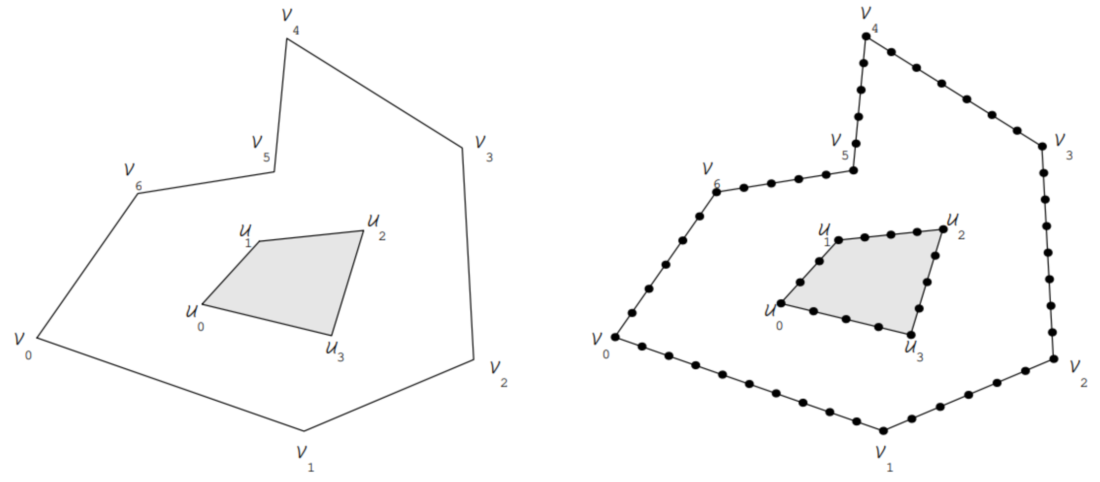


Figure 1: An example: (a) The domain Ω with a quadrilateral hole. (b) The vertices of the triangulation were chosen on $\partial\Omega$.

We define edge of $\Omega = D(P^E, P_1^I, \dots, P_h^I)$ as an edge of a polygon in its representation. Let n_e be the number of edges of Ω and let E_j, $j = 1,2,\dots,n_e$ its edges. The length of the boundary of Ω is given by $p_\Omega = \sum_{j=1,\ n_e} L_{E_j}$. We define subsequent domains of a given domain Ω the domains contained in Ω such that their edges are parallel to the edges of Ω and are at a fixed distance from them. Fig. (**2a**) shows the result obtained by using the construction of the subsequent domains starting from the domain Ω shown in Fig. (**1a**), in particular, the domain $\Omega = \Omega_0 = D((v_0,\dots,v_6),(u_0,\dots,u_3))$ has a unique subsequent domain $\Omega_1 = D((w_0,\dots,w_6),(z_0,\dots,z_3))$, the domain Ω_1 has three subsequent domains $\Omega_2 = D((y_0,y_1,y_2))$, $\Omega_3 = D((x_0,x_1,x_2))$ and $\Omega_4 = D((t_0,\dots,t_3))$, and these last three domains have no subsequent domain.

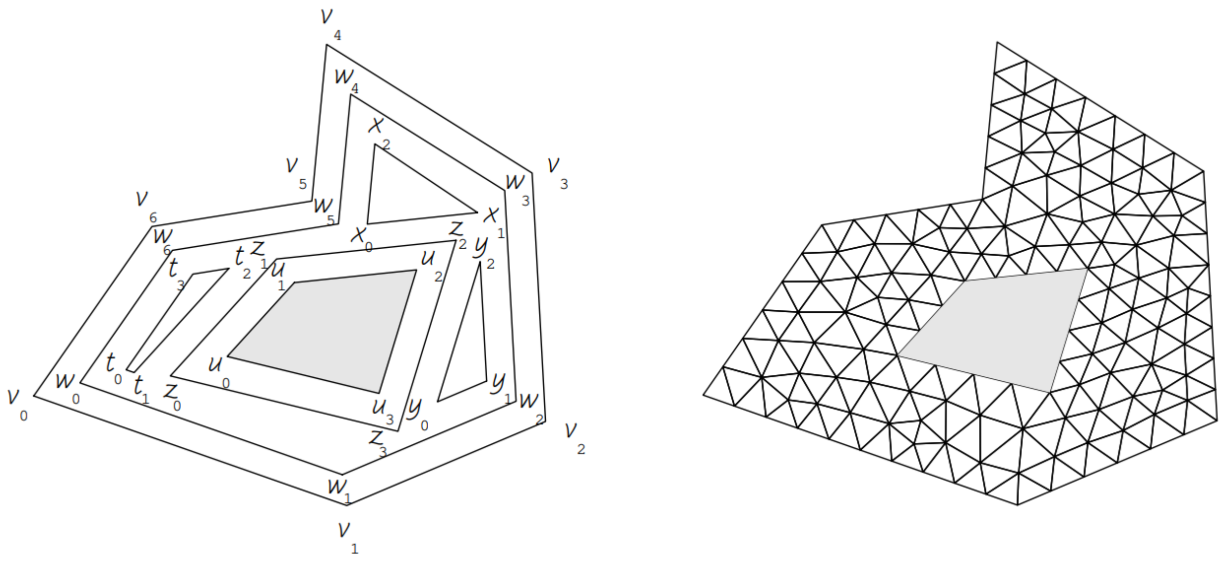


Figure 2: (a) The construction of the subsequent domains. (b) The triangulation obtained by applying the *PF* algorithm on domain Ω of Fig. (**1a**).

In the following, we report the *PF* algorithm that has been proposed in [9] and computes a triangulation of a generic domain $\Omega \subset \mathbb{R}^2$. *PF* uses the construction of the subsequent domains starting from a given domain Ω to compute a triangulation of the domain itself. In Fig. (**2b**) we have the resulting triangulation obtained on the domain shown in Fig. (**1a**) by using *PF*.

Algorithm 1 (*PF*)

Let $\Omega \subset \mathbb{R}^2$ be a given domain, let d be the desired length for a triangulation on Ω; construct a triangulation on Ω by computing $\mathcal{S}$ and $\mathcal{T}$ as follows:

- S1 Initialization: $\Omega_0 = \Omega$, $i = 0$, $j = 0$, $\mathcal{S} = \emptyset$, $\mathcal{T} = \emptyset$.
- S2 Insert in $\mathcal{S}$ the vertices chosen on $\partial\Omega$.
- S3 Create the $h(i) \geq 0$ subsequent domains of Ω_i; denote these domains with $\Omega_{j+1}, \Omega_{j+2}, \dots, \Omega_{j+h(i)}$; set $j = j + h(i)$.
- S4 Insert in $\mathcal{S}$ the vertices chosen on the boundary of the subsequent domains of Ω_i.
- S5 Insert in $\mathcal{T}$ the triangles obtained by triangulating the polygonal domain between Ω_i and its subsequent domains, if $h(i) = 0$ triangulate Ω_i.
- S6 Set $i = i + 1$ and if $i \leq j$ go to S3.
- S7 Compute the mesh optimization.

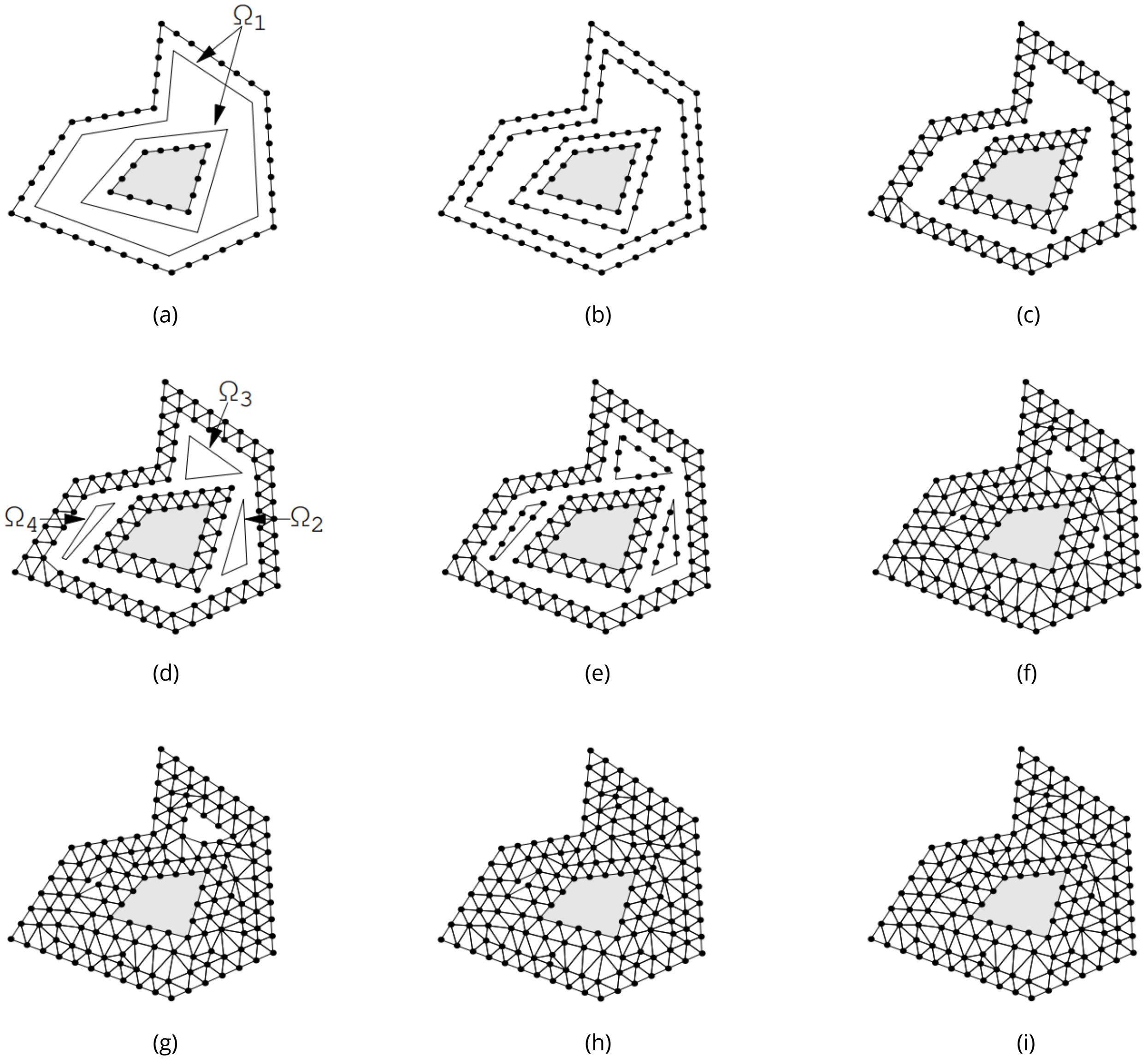


Figure 3: The steps of the *PF* algorithm for domain Ω of Fig. (**1a**). (a) Step S3 for $i = 0$; (b) Step S4 for $i = 0$; (c) Step S5 for $i = 0$; (d) Step S3 for $i = 1$; (e) Step S4 for $i = 1$; (f) Step S5 for $i = 1$; (g) Step S5 for $i = 2$; (h) Step S5 for $i = 3$; (i) Step S5 for $i = 4$.

In step S2 we insert in $\mathcal{S}$ some points chosen on the boundary of Ω in the following way. For each edge E of Ω, we define n_E as the largest integer such that $|L_E - n_E d| \leq \frac{d}{2}$ and we define $d_E = \frac{L_E}{n_E}$, then we insert, in $\mathcal{S}$, $n_E + 1$ uniformly distributed points of E such that two consecutive points have distance equal to d_E, note that for each edge E we have:

$$\frac{d}{2} \leq d_E < \frac{3d}{2}. \tag{1}$$

With this construction, we have that the number n_{bv} of the triangulation vertices chosen in $\partial\Omega$ is

$$n_{bv} = \sum_{j=1}^{n_e} n_{E_j} = \sum_{j=1}^{n_e} \frac{L_{E_j}}{d_{E_j}}, \tag{2}$$

The computational time necessary for this construction is $O(n_{bv})$, and the folowing inequalities hold

$$\frac{2p_\Omega}{3d} \leq n_{bv} \leq \frac{2p_\Omega}{d}. \tag{3}$$

Fig. (**1b**) shows the n_{bv} triangulation vertices chosen in $\partial\Omega$, note that two consecutive vertices chosen in an edge E of $\partial\Omega$ have a fixed distance d_E that is as near as possible to d.

In step S3, at the generic i-th iteration, we construct $\Omega_{j+1}, \Omega_{j+2}, \dots, \Omega_{j+h(i)}$ the subsequent domains of Ω_i by using a parallel offsetting technique; these domains are further processed in the next iterations of the algorithm. This not trivial construction of the subsequent domains of a given domain Ω_i is carried on in such a way that each edge E' of the subsequent domains of Ω_i is parallel to an edge E of Ω_i, is inside Ω_i and the distance between E' and E is $\frac{\sqrt{3}}{2} d_E$, that is the height of an equilateral triangle with edge length d_E. Moreover, we put $d_{E'} = d_E$. Figs. (**2a**, **3a**, **3d**), show similar examples of this construction, see [9] for more details. At each i-th iteration the computational time required for this construction is $O(n_e^2)$, in fact for each i it depends quadratically on the number of edges of Ω_i that is $O(n_e)$.

In step S4, at the generic i-th iteration, we insert in $\mathcal{S}$ some points chosen on the boundary of $\Omega_{j+1}, \Omega_{j+2}, \dots, \Omega_{j+h(i)}$ the subsequent domains of Ω_i. These points are chosen in such a way that, together with the triangulation vertices previously placed along the boundary of Ω_i, they determine as many as possible equilateral triangles in the strip $\Omega_i \backslash \cup_{h=j+1}^{j+h(i)} \Omega_h$. In particular, when E' is an edge of a subsequent domain of Ω_i parallel to edge E of Ω_i, we insert in $\mathcal{S}$ points of E' uniformly distributed that satisfy the following two properties: 1) the distance between two consecutive points is equal to $d_{E'} = d_E$; 2) at least one of these points is on the axis of the segment defined by two consecutive triangulation vertices of E previously inserted in $\mathcal{S}$. When necessary, we insert also the endpoints of E'. We note that when Ω_i has no subsequent domains, i.e. $h(i) = 0$, we do not insert other points in $\mathcal{S}$. At each i-th iteration the computational time required for this construction is $O(n_{bv})$ in fact this is the order of the number of triangulation vertices chosen in the edges of the subsequent domains of Ω_i. Once we inserted in $\mathcal{S}$ the points corresponding to each subsequent domain of Ω_i, we control if two of the inserted points are too close to each other, in this case, the points are modified. At each i-th iteration, this check can be made by cycling on the edges of Ω_i and the edges of its subsequent domains, and so its computational cost is $O(n_e^2)$. In Figs. (**3b**, **3e**) some examples are shown.

In step S5, at the generic i-th iteration, we triangulate the strip $R_i = \Omega_i \backslash \cup_{h=j+1}^{j+h(i)} \Omega_h$, between a domain Ω_i and its subsequent domains, by using the triangulation vertices inserted in step S4, see Figs. (**3c**) and (**3f**) as examples of the triangulation obtained with this step. Note that, if the domain Ω_i has no subsequent domains, i.e. $h(i) = 0$, we triangulate Ω_i, see Figs. (**3g**, **3h**, **3i**) as examples of the triangulation obtained with this step when $h(i) = 0$. In particular, the triangulation of the strip R_i is performed by two different phases: in the first phase we construct the equilateral triangles whose union gives a region $S_i \subset R_i$, every one of these equilateral triangles has at least one vertex on $\partial\Omega_i$ and at least one vertex on $\partial(\cup_{h=j+1}^{j+h(i)} \Omega_h)$; in the second phase, the Delaunay triangulation of each connected component of the critical region $R_i \backslash S_i$ is computed by using the Watson algorithm [17], possibly after having added a negligible number of Steiner points, depending on a certain closeness criterion.

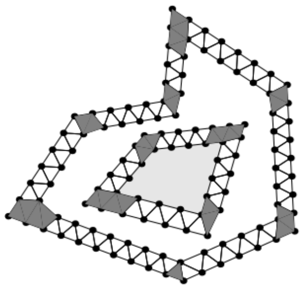


Figure 4: The first phase of step S5 of the *PF* algorithm for domain Ω of Fig. (**1**) inserts in S only the equilateral triangles. The more dark areas are triangulated in the second phase.

For example, Fig. (**4**) shows the triangulation computed with the first phase on the strip $R_0 = \Omega_0 \backslash \Omega_1$ shown in Fig. (**3b**), moreover, the more dark areas denote the critical region $R_0 \backslash S_0$ where in the second phase of step S5 we use the Watson algorithm to triangulate such a region and in Fig. (**3c**) it is shown the final result. In Figs. (**5**) and (**11**), we have the triangles inserted in $\mathcal{T}$ by using only the first phase of step S5 at each iteration i. Let Γ be the union of all the critical regions $R_i \backslash S_i$ obtained at each iteration i-th of the *PF* algorithm, in Fig. (**5**) Γ is the untriangulated part of the regular polygon, in Fig. (**11**) Γ is the more dark area. We note that, in this region the *PF* algorithm at each i-th iteration applies the Watson algorithm on connected subdomains to compute the Delaunay triangulation of the triangulation vertices inside Γ. Moreover, at each i-th iteration the computational time required for the first phase of step S5 is $O(n_{bv})$ because it depends on the number of triangulation vertices of the triangulation chosen on the boundary of Ω_i. The analysis of the computational cost of the second phase of step S5 will be made in Section 3.

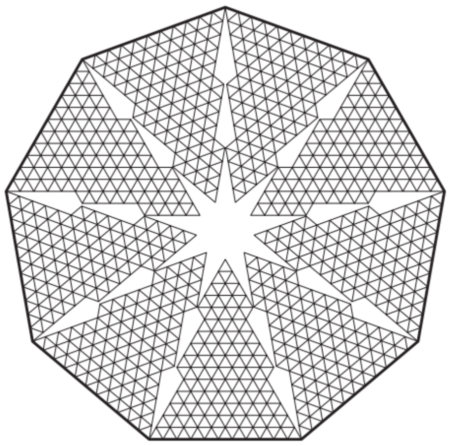


Figure 5: The untriangulated part is the union of all the critical regions.

Step S6 allows the recursively application of steps from S3 to S5 to each of the domains arising from step S3; this process ends since at iteration i-th each subsequent domain of Ω_i is contained in Ω_i Moreover, by construction, the distance between each edge E' of any subsequent domain of Ω_i and the parallel edge E of Ω_i is equal to $\frac{\sqrt{3}}{2} d_E$ but $d_E > \frac{d}{2}$. So, since the distance between the boundary of Ω_i and the boundary of its subsequent domain is greater than $\frac{\sqrt{3}d}{4}$ after $I = O(n_{bv})$ iterations there is no other subsequent domain. So the number of iterations of steps S3-S5 performed by the algorithm is $I = O(n_{bv})$.

We remark that the quality of the triangulation produced by the above procedure is already good. However, we optionally apply a mesh optimization to improve it. In step S7, at the end of the iterations, the triangulation constructed in the previous steps is optimized with respect to the minimal angle criterion and the maximal angle criterion, see [9] for more details. In Fig. (**2b**) we have the mesh obtained after optimization over the domain of Fig. (**1a**). We note that the optimization process is needed only for the triangles whose vertices are in Γ.

3. Computational Time

Let Ω be a domain with polygonal boundary, n_e be the number of edges of Ω, n_v be the number of triangulation vertices generated by the *PF* algorithm, I be the number of iterations of steps S3-S5, n_{bv} be the number of triangulation vertices chosen by the *PF* algorithm on $\partial\Omega$, $\Gamma = \cup_{i=1}^{I} (R_i \backslash S_i)$ be the union of the critical regions arising when we apply the *PF* algorithm on Ω. Then the iteration for $i = 1, \ldots, I$ of steps S3-S5 with the exception of the second phase of step S5 gives the triangulation of $\Omega\backslash\Gamma$, see Figs. (**5**) and (**11**) for examples.

Theorem 1. The computational time of the *PF* algorithm for constructing the triangulation of $\Omega\backslash\Gamma$ is $O(n_e n_v)$.

Proof: We note that the number n_v of the vertices of the triangulation constructed with the *PF* algorithm is $n_v = O(n_{bv}^2)$. Hence, for each iteration $i = 1, \ldots I$ the computational time of all the steps S3-S5 with the exception of the second phase of step $S5$ is $O(n_e^2) + O(n_{bv}) = O(n_e n_{bv})$, moreover, $I = O(n_{bv})$ so that the computational time necessary for the construction of the triangulation of $\Omega\backslash\Gamma$, is $O(n_e n_{bv}^2) = O(n_e n_v)$.

The Delaunay triangulation of the remaining part Γ is computed in the second phase of step S5 by applying the Watson algorithm in each critical region separately, and so its computational time is less than $O(K \log K)$ where K is the number of the triangulation vertices that are inside Γ, in fact this is the computational time of the Watson algorithm applied in whole Γ.

In the following we prove that $K = O(\sqrt{n_v})$, in particular, we prove an elementary lemma concerning the plane tessellation $\mathcal{T}_d$ made up of equilateral triangles with edge length d, (Fig. **6**). Then we give two lemma where we evaluate the number of the triangulation vertices in Γ when Ω is a regular polygon and when Ω is a general domain.

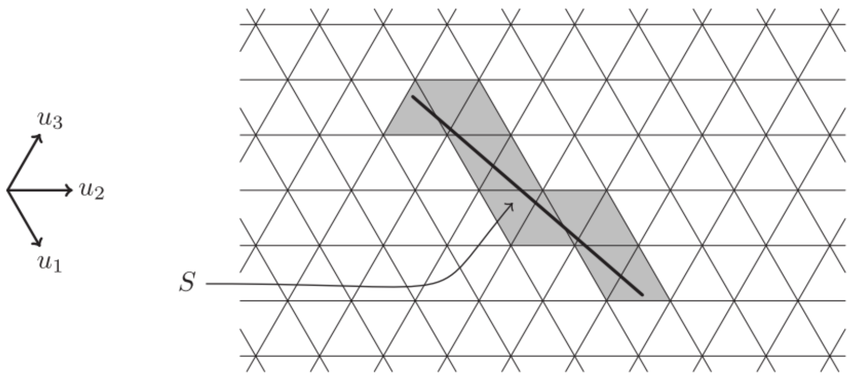


Figure 6: The segment S meets at most $4(\sqrt{3}L_S/(3d) + 1)$ interiors of triangles in tessellation $\mathcal{T}_d$.

Lemma 1. Given any segment S of length L_S, the number $N(S)$ of triangles of $\mathcal{T}_d$ whose interiors meet S satisfies the following inequality

$$N(S) \leq 4\left(\frac{L_S}{d} + 1\right), \tag{4}$$

and the union of these $N(S)$ triangles is a polygonal disk with $N(S) + 2$ vertices.

Proof: Let $u_1 = (1/2, -\sqrt{3}/2)$, $u_2 = (1,0)$, and $u_3 = (1/2, \sqrt{3}/2)$ unit vectors parallel to the edges of $\mathcal{T}_d$ and $0 \leq \theta_i \leq \pi/2$ be the angle between S and the direction determined by u_i. Up to symmetries of $\mathcal{T}_d$, we can assume $\theta_1 \leq \theta_2 \leq \theta_3$ (cf. Fig. **6**). Then, we have $\theta_1 \leq \pi/6 \leq \theta_2 \leq \pi/3 \leq \theta_3$.

We denote by Q the smallest parallelogram among the ones which contain S, are unions of triangles of $\mathcal{T}_d$ and have edges parallel to u_2 and u_3 (cf. Fig. **7**). We think of Q as the union of L_2 strips parallel to u_3 made up of triangles of $\mathcal{T}_d$, where dL_2 is the length of the edges of Q parallel to u_2.

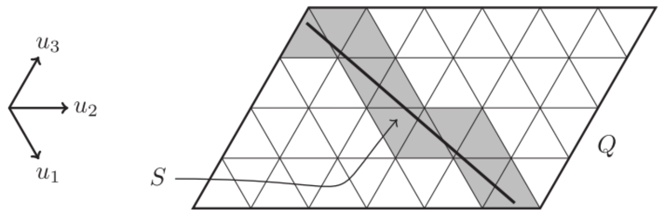


Figure 7: The smallest parallelogram Q containing S.

Because of the assumptions made on the θ_i's, S can meet at most two triangle interiors inside each strip, note that the union of these triangles gives a polygonal disk with $N(S) + 2$ vertices. Hence, we have $N(S) \leq 2L_2$ but $dL_2 \leq (L_S/\cos\theta_2 + 2d) \leq (2L_S + 2d)$ so $N(S) \leq 4(L_S/d + 1)$.

Now, we use the previous lemma in order to obtain the claimed upper bound on the number of triangulation vertices in Γ which arise when we perform the *PF* algorithm on a regular polygon.

Lemma 2. The union Γ of all the critical regions arising when we perform the *PF* algorithm on a regular polygon Ω, with n_e edges of length $n \cdot d$, to get a triangulation with preferred edge length d, has at most $4n_{bv}n_e/5 + 8n_e$ triangulation vertices, where $n_{bv} = n_e \cdot n$ is the number of triangulation vertices on the boundary of the regular polygon.

Proof: It suffices to prove that inside each triangle T bounded by an edge of the polygon and two radii of its circumcircle, there are at most $4nn_e/5 + 8$ triangulation vertices of the critical regions. Looking at T and denoting by R one of its radial edges, we see that the number of such triangulation vertices inside T is at most $N(R) + 4$, where $N(R)$ is the number of triangles of the tessellation whose interiors meet R (cf. Fig. **8**). In fact, by Lemma 1 the union of the triangles whose interiors meet R is a polygonal disk with $N(R) + 2$ vertices, but at most half of them plus one lies inside T. Then, $N(R) + 4$ is obtained by considering the contribute of the other radial edge of T.

Now, by Lemma 1 and taking into account that $n_e \geq 3$ and $\sin x \geq \frac{3\sqrt{3}x}{2\pi}$ when $0 \leq x \leq \frac{\pi}{3}$, we have the inequalities

$$N(R) \leq 4\left(\frac{n}{2\sin(\pi/n_e)} + 1\right) \leq 4\left(\frac{n}{3\sqrt{3}/n_e} + 1\right) \leq \frac{4nn_e}{5} + 4\ ,$$

which complete the proof.

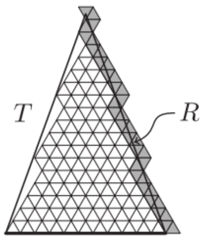


Figure 8: The triangulation vertices of the critical regions inside the triangle T.

Lemma 3. Let $\Omega \subset \mathbb{R}^2$ be a domain with polygonal boundary $\partial\Omega$. Let n_e be the number of its edges and let n_{bv} be the number of triangulation vertices chosen on $\partial\Omega$. The union Γ of all the critical regions arising when we perform the *PF* algorithm on Ω to get a triangulation with desired edge length d, has at most $6n_{bv}n_e + 12n_e^2 - 16n_e$ triangulation vertices.

Proof: Let E_j, $j = 1, \dots, n_e$ be the edges of Ω. Let Ω_i, $i = 0, \dots, I$, be the domains constructed in the *PF* algorithm. For $i = 0, \dots I$, let E be an edge of domain Ω_i and let $E_1^{'}, \dots, E_L^{'}$ be the edges of the subsequent domains of Ω_i that are parallel to E and have distance $\frac{\sqrt{3}}{2}d_E$ from E. Then there exist $L+1$ unique segments $S_1, \dots, S_{L+1}$ that join end points of $E_1^{'}, \dots, E_L^{'}$ and E such that $\left(\cup_{l=1}^{L} E_l^{'}\right) \cup \left(\cup_{l=1}^{L+1} S_l\right) \cup E$ is a quadrilateral (cfr. Fig. **9**).

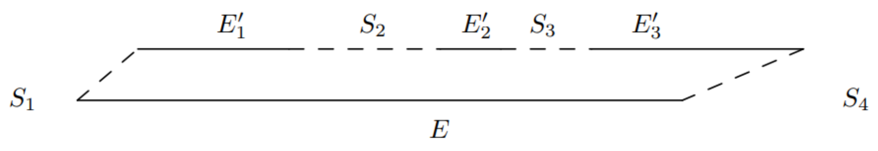


Figure 9: The edge E and the corresponding edges $E_j^{'}$ at distance $\frac{\sqrt{3}}{2}d_E$, joining their endpoints with suitable segments S_j we obtain a quadrilateral.

We denote with $\mathcal{F}$ the set whose elements are either the segments above defined or edges of Ω_i, $i = 0, \dots, I$. In Fig. (**10**) we can see an example of the set $\mathcal{F}$. By using the *PF* algorithm in the domain Ω, we can identify n_e connected regions T_j, $j = 1, \dots, n_e$ (an example is shown in Fig. (**11**) characterized by the following properties.

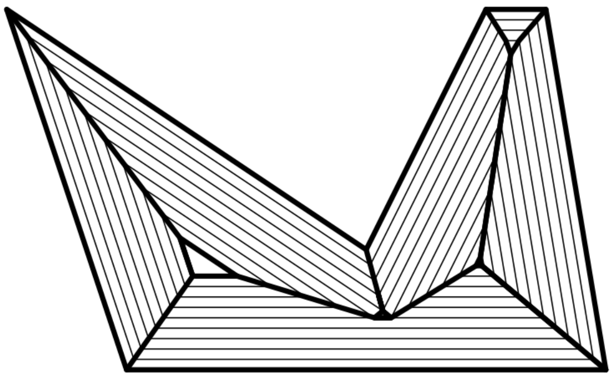


Figure 10: The segments in $\mathcal{F}$ form a graph.

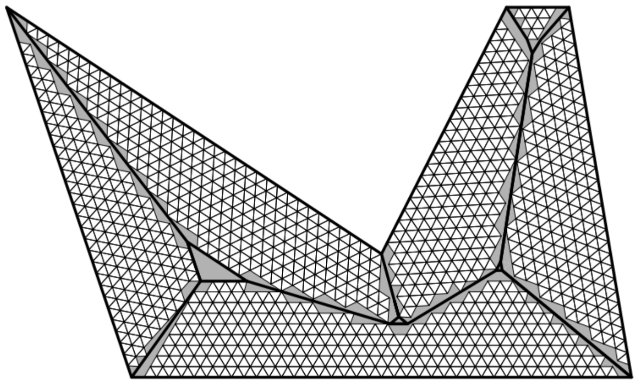


Figure 11: In the domain Ω there are 6 connected regions T_j containing a tessellation $\mathcal{T}_j$ made up of equilateral triangles, the gray area is the critical region Γ.

- T_j contains the regions where with the *PF* algorithm we have inserted equilateral triangles that come from edge E_j of $\partial\Omega$, these triangles give a tessellation $\mathcal{T}_j$ made up of equilateral triangles with edge length d_{E_j}.
- The triangulation $\mathcal{T}$, resulting from applying the *PF* algorithm without the smoothing step, is the union of $\mathcal{T}_j$, $j = 1, \dots, n_e$, and of the triangulation of the vertices in Γ.
- T_j has polygonal boundary whose edges are union of segments belonging to $\mathcal{F}$.
- The edges of T_j are E_j and other k_j edges E_j^k, $k = 1, \dots, k_j$, moreover by construction we have

$$\sum_{k=1}^{k_j} L_{E_j^k} \le p_\Omega. \quad (5)$$

First of all, we note that

$$k_j \leq 3n_e - 4, \tag{6}$$

that is a consequence of the following facts. For each $k \neq j$, $T_k \cap T_j$ is a graph with at most two edges, in fact $T_k \cap T_j$ is the union of segments of $\mathcal{F}$ that are all aligned with eventually the exception of one segment. So that T_j has at most $2(n_e - 1)$ edges in common with the other $n_e - 1$ regions T_k, $k \neq j$, and these edges are joined by at most $n_e - 2$ segments in $\mathcal{F}$.

By construction, each triangulation vertex in Γ belongs to a region $T_j, j = 1, \dots, n_e$. So to prove the lemma it suffices to see that inside each region T_j, $j = 1, \dots, n_e$ there are at most $6n_{bv} + 12n_e - 16$ triangulation vertices inside Γ. Looking at T_j we see that each vertex inside $\Gamma \cap T_j$ is a vertex of a triangle in $\mathcal{T}_j$ whose interior meets E_j^k for some $k = 1, \dots, k_j$.

Moreover, for each k we have at most $N(E_j^k)$ triangles of tessellation $\mathcal{T}_j$ whose interiors meet E_j^k, and the union of these triangles is a polygonal disk with at most $N(E_j^k) + 2$ triangulation vertices, so that at most half of them plus one lies inside T_j. Then inside T_j there are at most $\sum_{k=1}^{k_j} \left(\frac{N(E_j^k)}{2} + 2\right)$ triangulation vertices chosen in Γ.

Now, by Lemma 1 and equations (1), (3), (5) we have the inequalities

$$\sum_{k=1}^{k_j} \left(\frac{N(E_j^k)}{2} + 2\right) \leq \sum_{k=1}^{k_j} 2\left(\frac{L_{E_j^k}}{d_{E_j}} + 1\right) + 2k_j \leq$$

$$\leq 2\left(\frac{p_\Omega}{d_{E_j}} + k_j\right) + 2k_j \leq 6n_{bv} + 4k_j,$$

which, with (6), completes the proof.□

Theorem 2. For domains with polygonal boundary approximation having n_e edges, the computational time of the *PF* algorithm is $O(n_e n_v)$.

Proof: By Theorem 1, the computational time of steps S1-S6 is $O(n_e n_v) + O(K \log K)$ where K is the number of triangulation vertices in Γ. By Lemma 2 when the domain is a regular polygon and by Lemma 3, when the domain is a general domain with a polygonal boundary approximation with n_e edges, we have that $K = O(n_e n_{bv})$. Then, from the quite trivial fact that $n_v = O(n_{bv}^2)$ we have that the computational time of steps S1-S6 is $O(n_e n_v)$. Moreover, the optimization of the resulting mesh can be performed by cycling on the K triangulation vertices in Γ, and so its computational cost is $O(n_e n_{bv})$.

We remark that in the worst case $n_e = n_{bv}$ the computational cost of *PF* is $O(n_v^{3/2})$, instead in the more frequent case $n_e << n_{bv}$ the computational cost of *PF* is $O(n_v)$.

4. Experimental Results

In the experimental results, only the computational cost of the mesh construction without optimization has been considered, for two reasons: 1) the news of the *PF* algorithm is in the mesh construction; 2) even if the optimization is only needed on triangles lying on Γ and therefore its computational cost is $O(n_e n_{bv})$, in the present version of the optimization is made on all triangles. Tests have been performed on a Pentium 4 CPU 2.66GHz, RAM 480 Mb.

The three domains shown in Fig. (**12**) have been considered in the numerical simulations: $\Omega^{(1)}$ is a polygon with 6 edges and without holes; $\Omega^{(2)}$ has two holes and 14 edges, $\Omega^{(3)}$ is a domain with one hole and curved boundary, the polygonal discretization of its boundary has 55 edges of which 12 are edges of the hole. In Fig. (**13**) we can see examples of mesh generated by the *PF* algorithm on these domains, larger meshes generated by *PF* have the same aspect.

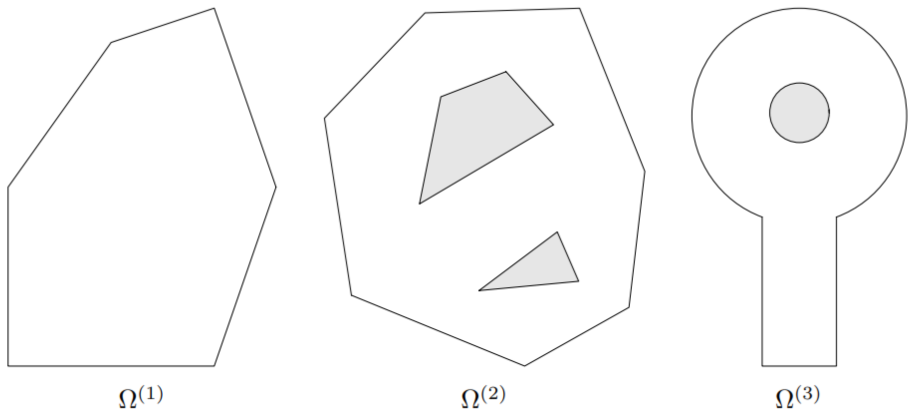


Figure 12: $\Omega^{(1)}$ is a polygon with 6 edges and without a hole; $\Omega^{(2)}$ has two holes and 14 edges; $\Omega^{(3)}$ has one hole and curved boundary whose discretization has 55 edges.

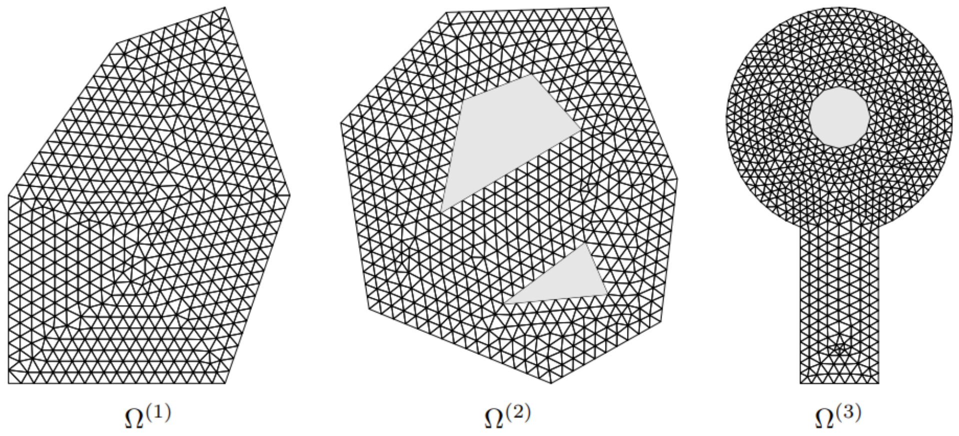


Figure 13: Mesh generated by *PF* on $\Omega^{(1)}$ has $nv = 571$ vertices. Mesh generated by *PF* on $\Omega^{(2)}$ has $nv = 650$. Mesh generated by *PF* on $\Omega^{(3)}$ has $nv = 788$.

Table 1: Computational time t, in seconds $[s]$, needed for mesh generation with the algorithm *PF* without optimization for the domain $\Omega^{(1)}$, shown in Fig. (12); $n_e = 6$ is the number of edges of the domain boundary, and n_v is the number of vertices of the mesh constructed by *PF*. Also the value of $\frac{t}{n_e n_v}$ is reported.

n_v	$t\ [s]$	$\frac{t}{n_e n_v}$
50029	0.641	2.14(−6)
100031	1.231	2.05(−6)
150105	1.815	2.02(−6)
200232	2.452	2.04(−6)
250228	2.858	1.90(−6)
300358	3.492	1.94(−6)
350075	3.904	1.86(−6)
399935	4.411	1.84(−6)
449890	4.978	1.84(−6)
500138	5.473	1.82(−6)

In Tables **1-3** we report the computational time t, in seconds $[s]$, needed for the mesh generation with the *PF* algorithm without optimization for each of the domains $\Omega^{(i)}$, $i = 1,2,3$, shown in Fig. (**12**), for different values of the number of vertices n_v. In the same tables also the values of $\frac{t}{n_e n_v}$ are reported. For each domain $\Omega^{(i)}$, $i = 1,2,3$, in Fig. (**14**) there are reported the values of n_v and of $\frac{t}{n_e n_v}$ on the x -axes and the y -axes, respectively. In addition, in Fig. (**15**) we reported the values of n_v and of t on the x-axes and the y-axes, respectively.

Table 2: Computational time t, in seconds $[s]$, needed for mesh generation with the algorithm *PF* without optimization for the domain $\Omega^{(2)}$, shown in Fig. (12); $n_e = 14$ is the number of edges of the domain boundary, and n_v is the number of vertices of the mesh constructed by *PF*. Also the value of $\frac{t}{n_e n_v}$ is reported.

n_v	$t\ [s]$	$\frac{t}{n_e n_v}$
50006	0.799	1.14(−6)
100001	1.374	9.81(−7)
150036	1.981	9.43(−7)
200022	2.545	9.09(−7)
250063	3.219	9.19(−7)
300006	3.649	8.69(−7)
349944	4.403	8.99(−7)
399975	4.759	8.50(−7)
450009	5.448	8.65(−7)
500017	6.137	8.77(−7)

Table 3: Computational time t, in seconds $[s]$, needed for mesh generation with the algorithm *PF* without optimization for the domain $\Omega^{(3)}$, shown in Fig. (12); $n_e = 55$ is the number of edges of the discretization of the domain boundary, and n_v is the number of vertices of the mesh constructed by *PF*. Also the value of $\frac{t}{n_e n_v}$ is reported.

n_v	$t\ [s]$	$\frac{t}{n_e n_v}$
49993	1.471	5.35(−7)
99449	3.195	5.84(−7)
150510	3.806	4.60(−7)
200998	4.527	4.10(−7)
249349	5.483	4.00(−7)
299032	6.806	4.14(−7)
349306	8.481	4.41(−7)
400670	8.911	4.04(−7)
449653	9.951	4.02(−7)
500218	10.363	3.77(−7)

From Tables **1-3** and Fig. (**14**) we can see that $\frac{t}{n_e n_v}$ is near constant. In addition, from Tables **1-3** and Fig. (**15**) we can conclude that $t = O(n_v)$. We note that in the considered domains $n_e << n_{bv}$.

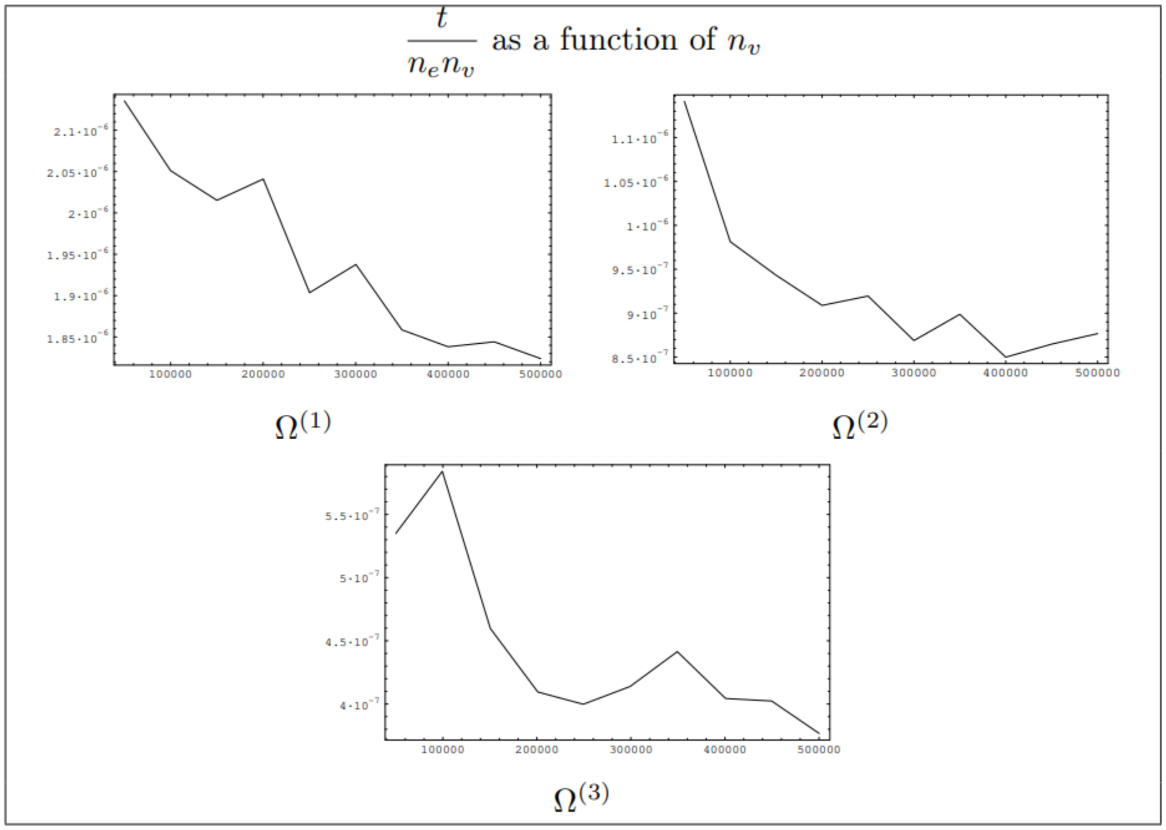


Figure 14: In the x-axis there are displayed the values of n_v and in the y-axis there are displayed the values of $\frac{t}{n_e n_v}$ obtained by using *PF* to generate a mesh on domains $\Omega^{(i)}$, $i = 1,2,3$.

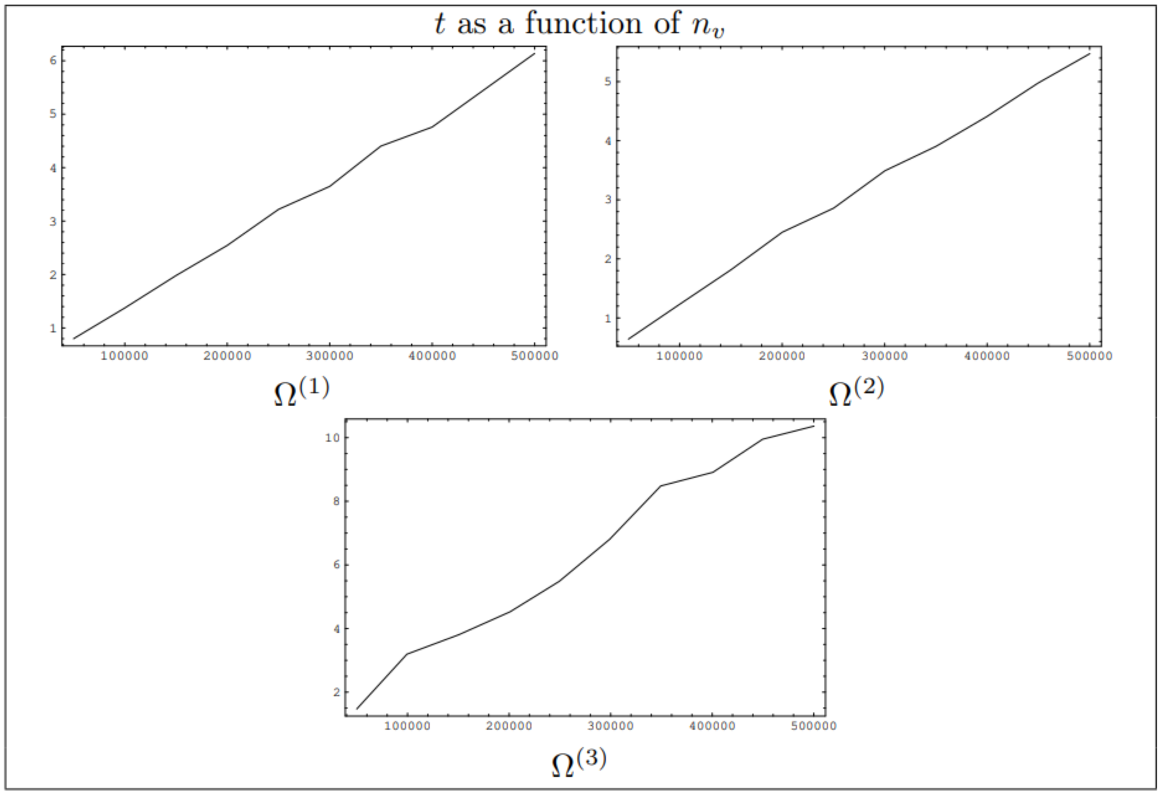


Figure 15: In the x-axis there are displayed the values of n_v and in the y-axis there are displayed the values of t obtained by using *PF* to generate a mesh on domains $\Omega^{(i)}$, $i = 1,2,3$.

5. Final Remarks

In this work, we analyze the computational time of *PF*, that is a triangulation algorithm for 2D domains with holes. It is based on polygon offsetting starting from the domain boundary. In Theorem 2, we have shown that *PF* requires a computational time of $O(n_e n_v)$. In particular, it is linear with respect to the number of the triangulation vertices n_v, when n_e, the number of the edges of the domain boundary approximation, is much smaller than n_{bv}, the number of vertices of the triangulation chosen in domain boundary, because $n_v = O(n_{bv}^2)$. Note that this situation is the most common one, even if the domain has a curved boundary, and experimental results confirmed the upper bound $O(n_e n_v)$.

We remark that the proof of Theorem 2 is essentially based on the fact that the critical regions are concentrated near to a finite graph. This seems to be supported by the experimental results. Moreover, Lemma 2 gives an upper bound for the number of triangulation vertices of the union of all critical regions. Actually, the *PF* algorithm applies the Delaunay method to each critical region separately. This argument does not allow us to improve the theoretical upper bound of Theorem 2, but in practice reduces the computational time of the *PF* algorithm as it results from experiments.

This study of the computational time of *PF*, together with the goodness of the generated mesh quality shown in [9], suggests that *PF* should be developed towards nonuniform meshes or a three-dimensional version of the proposed method.

Conflict of Interest

The authors declare no potential conflict of interest.

Acknowledgments

We would like to show our gratitude to University of Camerino, and in particular to Roberto Pennesi and Paolo Mancinelli for the technical support provided.

References

[1] Wang Q, Ye J, Wu H, Gao B, Shepherd P. A triangular grid generation and optimization framework for the design of free-form gridshells. Comput Aid Des. 2019; 113: 96–113. https://doi.org/10.1016/j.cad.2019.04.005

[2] Liu F, Feng R. Automatic triangular grid generation on a free-form surface using a particle self-organizing system. Eng Comput. 2020; 36: 377–89. https://doi.org/10.1007/s00366-019-00705-4

[3] Liu F, Feng R, Tsavdaridis KD. A novel progressive grid generation method for free-form grid structure design and case studies. J Build Eng. 2021; 34: 101866. https://doi.org/10.1016/j.jobe.2020.101866

[4] Liseikin V. Grid generation methods. vol. 1, Springer; 1999, p. 1–29. https://doi.org/10.1007/978-3-662-03949-6_1

[5] Thompson JF. Handbook of grid generation. 1st Edition. Boca Raton: CRC Press; 1999. https://doi.org/10.1201/9781420050349

[6] George P-L, Houman Borouchaki. Delaunay triangulation and meshing. vol. 1. Paris: 1998.

[7] Parthasarathy VN, Graichen CM, Hathaway AF. A comparison of tetrahedron quality measures. Finite Elem Anal Des. 1994; 15: 255–61. https://doi.org/10.1016/0168-874X(94)90033-7

[8] Mandad M, Campen M. Guaranteed-quality higher-order triangular meshing of 2D domains. ACM Trans Graph. 2021; 40: 1–14. https://doi.org/10.1145/3476576.3476736

[9] Egidi N, Misici L, Piergallini R. PolyFront: an algorithm for fast generation of the high-quality triangular mesh. Eng Comput. 2011; 27: 357–72. https://doi.org/10.1007/s00366-010-0206-6

[10] Blacker TD, Stephenson MB. Paving: A new approach to automated quadrilateral mesh generation. Int J Numer Methods Eng. 1991; 32: 811–47. https://doi.org/10.1002/nme.1620320410

[11] van Rens BJE, Brokken D, Brekelmans WAM, Baaijens FPT. A two-dimensional paving mesh generator for triangles with controllable aspect

ratio and quadrilaterals with high quality. Eng Comput. 1998; 14: 248–59. https://doi.org/10.1007/BF01215978

[12] George JA. Computer implementation of the finite element method. Stanford University; 1971.

[13] Lo SH. A new mesh generation scheme for arbitrary planar domains. Int J Numer Methods Eng. 1985; 21: 1403–26. https://doi.org/10.1002/nme.1620210805

[14] Löhner R. Progress in grid generation via the advancing front technique. Eng Comput. 1996; 12: 186–210. https://doi.org/10.1007/BF01198734

[15] Peraire J, Vahdati M, Morgan K, Zienkiewicz OC. Adaptive remeshing for compressible flow computations. J Comput Phys. 1987; 72: 449–66. https://doi.org/10.1016/0021-9991(87)90093-3

[16] Egidi N, Misici L, Pennesi R. An algorithm for planar triangulations. A graphical user interface. Proceedings of MASCOT03-3rd Meeting on Applied Scientific Computing and Tools, Grid Generation, Approximation and Visualization, Roma: IMACS Series in Computational and Applied Mathematics; 2004, p. 71–80.

[17] Watson DF. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. Comput J. 1981; 24: 167–72. https://doi.org/10.1093/comjnl/24.2.167