A Quadratic Running Time Example for Ruppert's Refinement Algorithm

Technical Report, Computer Science Department, USC

Jernej Barbič — University of Southern California

Gary Miller — Carnegie Mellon University

May 10, 2013

Abstract

We present an example where Ruppert's refinement algorithm for quality 2D Delaunay mesh generation runs in time quadratic in the size of the output mesh. For any $n > 1$, input and output mesh sizes of our example are $\Theta(n)$, the bounding box dimensions are $1 \times \Theta(n)$, and the algorithm running time is $\Theta(n^2)$. Because $\Theta(n^2)$ is a theoretical upper bound on the running time of Ruppert's refinement algorithm, our example is tight.

1 Introduction

Mesh generation is important technology with applications in scientific computing, engineering, computer graphics and other areas of computer science. Given a planar set of input points and edges, the goal of a 2D meshing algorithm is to find a triangulation such that (1) the input points and edges appear in the triangulation, (2) triangles with small angles (*skinny triangles*) are avoided, and (3) the number of output mesh vertices and triangles is as small as possible. One very common strategy for quality 2D mesh generation is to use Delaunay triangulations. Delaunay triangulations have the property that the minimum mesh angle is maximal over all triangulations of a given set of points. Delaunay-based meshing algorithms typically first compute the Delaunay triangulation of the input geometry and then iteratively add new vertices (*Steiner vertices*) to the mesh, until the Delaunay triangulation is free of triangles with small angles. Given an input set of points and edges, how many Steiner vertices are necessary to avoid small angles, and where should they be positioned? In 1995, Jim Ruppert [2] proposed a strategy of splitting skinny triangles, i.e., the strategy of picking the next Steiner point to be the circumcenter of a skinny triangle, and terminating when there are no more skinny triangles in the Delaunay triangulation. Ruppert proved that, assuming the minimum allowed angle is smaller than 20°, the algorithm always terminates. He also gave a tight asymptotical bound on the number of vertices in the output mesh in terms of the geometry of the input mesh. The number of output vertices is at most (up to a constant, depending only on the minimum angle) [2]

$$\int_{\mathcal{B}} \frac{1}{\mathrm{lfs}^2(x)} dx, \tag{1}$$

where $\mathcal{B}$ denotes the mesh bounding box and lfs is the local feature size function [2]. Ruppert also proved that, for any minimum angle $\alpha_{\min} < 20^\circ$, the number of vertices in the output mesh generated by his algorithm is smaller than a constant factor times the theoretically minimum possible number of vertices under *any* set of Steiner vertices.

After inserting a Steiner vertex, some triangles are created and some existing triangles cease to exist in the Delaunay triangulation. A practical implementation of Ruppert's algorithm must therefore maintain a data structure containing all the currently existing triangles. The necessary work to update the data structure after inserting a Steiner vertex is proportional to the number of triangles that have appeared or disappeared in the mesh. Therefore, one can measure the running time of Ruppert's algorithm by counting the number of triangles that were created or disappeared during the lifetime of the algorithm. Since in a planar graph, adding a new vertex increases the total number of triangles by 2, one can measure the work in terms of only the newly created triangles.

Ruppert's algorithm works well in practice. It has been implemented in the popular Triangle package [1] and is widely used. Ruppert's refinement algorithm, however, does not specify *which* skinny triangle to split among all the skinny triangles available at a certain point during the algorithm. We demonstrate that the running time of the algorithm can depend significantly on the particular choice of skinny triangles to split. We do so by providing a specific input set of points (no edges are needed), where there exists an inefficient strategy of selecting skinny triangles. Our strategy results in a quadratic number of triangles created during the course of the algorithm, whereas the output mesh size is linear in the input mesh size. Specifically, for any $n \geq 1$, and any minimum angle $\alpha_{\min} > 0$, we demonstrate an input set of points of size $n + 4$, with a bounding box of dimensions $1 \times \Theta(n)$, and a particular strategy of selecting skinny triangles, so that the total number of triangles created is $\Theta(n^2)$, even though the input and output number of points and triangles is only $\Theta(n)$.

2 The Example with Quadratic Running Time

For any $n \geq 1$, and any minimum allowed angle $\alpha_{\min} > 0$, consider the following set of input points (see Figure 2):

$$\mathcal{I} = \left\{ (\frac{i}{n}, 0) \,|\, i = 0, 1, \ldots, n \right\} \cup \left\{ (0, a), (1, a), (\frac{1}{2n}, h) \right\}, \tag{2}$$

where r is the smallest positive integer such that

$$\frac{1}{(2r+1)(2r-1)} < \tan^2(\alpha_{\min}), \tag{3}$$

and

$$h = \frac{1}{2n}\sqrt{(2r+1)(2r-1)}, \tag{4}$$

$$a = h + \frac{1}{h} = \Theta(n). \tag{5}$$

The values of r, h, a were chosen in a way that establishes a recursive, "broom"-like pattern of triangles while adding Steiner points (Figure 3), as we demonstrate in the rest of this report.

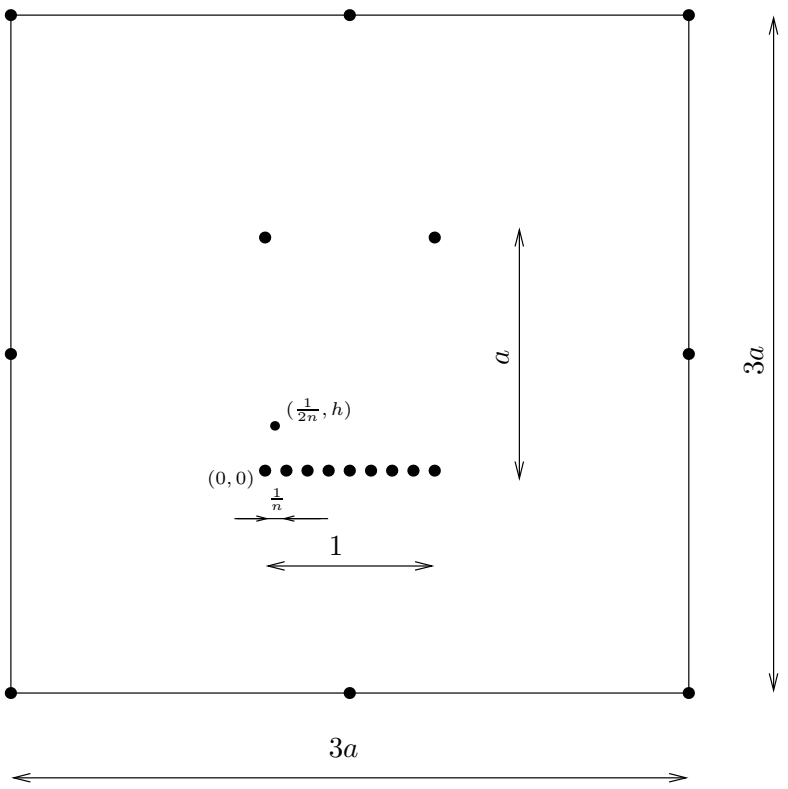

Figure 1: The $(n+4)$ input points, together with the bounding box, after splitting the four boundary segments. Quantities h and a are defined in Equations 4 and 5. Note the intentional presence of point $(\frac{1}{2n}, h)$, which triggers a recurring pattern of skinny triangles.

Ruppert's algorithm begins by bounding the input with a box of size three times the maximum of vertical and horizontal span, which is $3a$ in our example. The center of the bounding box is at $(1/2, a/2)$. The four box corner points are added to the input, together with the four boundary edge segments, which are immediately split by inserting the four edge midpoints, so that they become unencroached. The algorithm is then run on this extended input. Because our input consists of points, the eight boundary segments are the only constrained edges that appear in the extended input, or at any point during the execution of the algorithm.

To prove the running time of our example, we must first prove two geometric lemmas. Lemma 1 will assert that after splitting triangle CDP, the new triangle CDS is congruent to ABP (see Figure 2). Also, all the triangles to the right of CDS in the new mesh are congruent to the triangles to the right of ABP in the old mesh, establishing a recursive pattern.

Lemma 1 *For any $i \geq 0$ and $n \geq 1$, let A, B, C, D, P be points with coordinates*

$$A\big(\frac{i}{n}, 0\big),\ B\big(\frac{i+1}{n}, 0\big),\ C\big(\frac{i+r}{n}, 0\big),\ D\big(\frac{i+r+1}{n}, 0\big),\ P\big(\frac{i}{n} + \frac{1}{2n}, h\big) \qquad \text{(see Figure 2)}, \quad (6)$$

where r and h are defined as in Equations 3 and 4. Then, the angle θ' at P in triangle ABP is twice the angle θ at P in triangle CDP. Triangle CDP is skinny. Let S be the circumcenter of

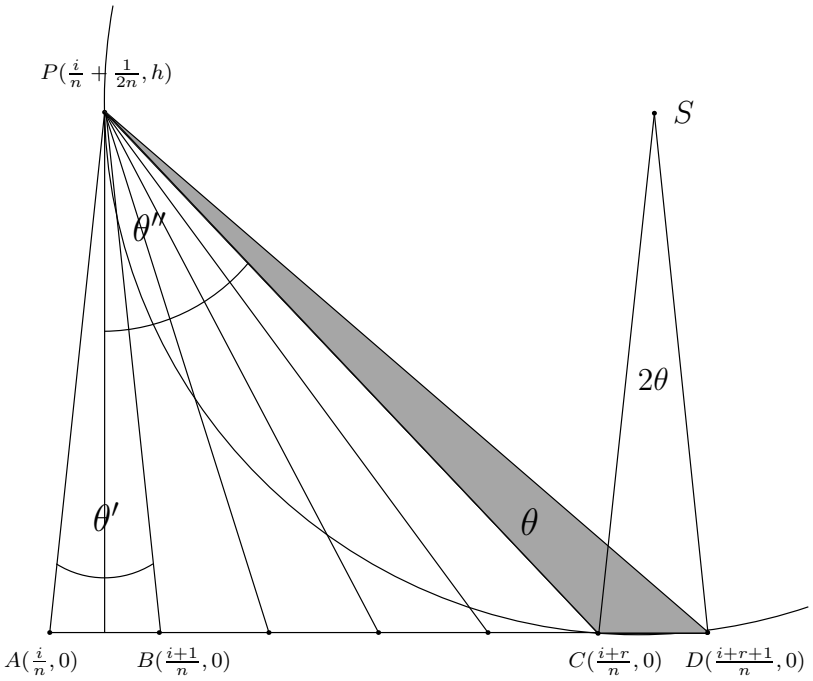

Figure 2: Illustration of Lemma 1, for $r = 5$. Shaded triangle CDP is skinny and will be split by Ruppert's algorithm.

triangle CDP. Then, the angle at S in triangle CDS is 2θ, coordinates of S are $(\frac{i+r}{n}+\frac{1}{2n},h)$, and triangle CDS is congruent to ABP.

Proof:

The angles θ'' (see Figure 2) and θ satisfy equations

$$\mathrm{tg}\,(\theta'') = \frac{r-\frac{1}{2}}{nh}, \qquad \mathrm{tg}\,(\theta+\theta'') = \frac{r+\frac{1}{2}}{nh}. \tag{7}$$

Using the tangent subtraction formula and the definition of h, it follows that

$$\mathrm{tg}\,(\theta) = \frac{\mathrm{tg}\,(\theta+\theta'')-\mathrm{tg}\,(\theta'')}{1-\mathrm{tg}\,(\theta+\theta'')\mathrm{tg}\,(\theta'')} = \frac{1}{2nh} = \mathrm{tg}\,(\frac{\theta'}{2}), \qquad \text{and} \qquad \theta' = 2\theta. \tag{8}$$

By Equation 3, it follows that $\theta < \alpha_{\min}$, therefore CDP is skinny. Clearly, CDP's circumcenter $S(x_S, y_S)$ has to lie on the perpendicular bisector of the segment CD, hence the x-coordinate of S is $\frac{i+r}{n}+\frac{1}{2n}$. By Thales's theorem, the angle at S in CDS is 2θ. Therefore, triangles ABP and CDS are congruent, and the y-coordinate of S must therefore equal h. ■

We now prove that after adding a Steiner point at S, a "broom" of triangles is created, spreading from S all the way to the right.

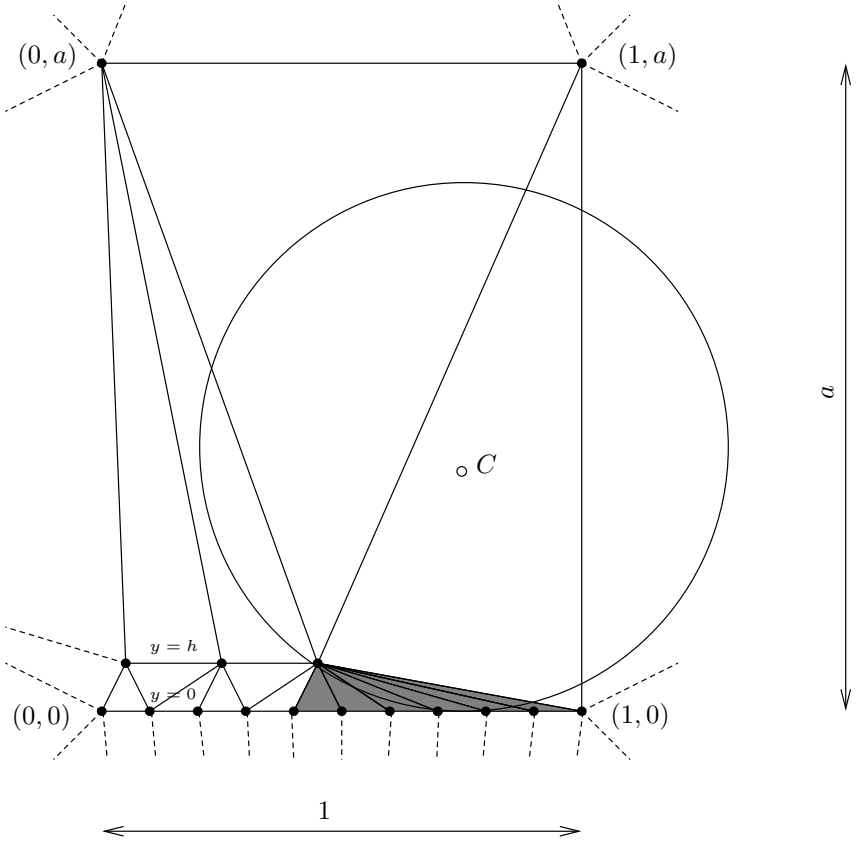

Figure 3: Illustration of Lemma 2 for $r = 2$, $n = 10$, $k = 2$, showing the Delaunay triangulation in the region $[0, 1] \times [0, a]$. The triangles to which the lemma applies (i.e. the "broom") are shaded and the empty circumcircle is shown for one of them. Bounding box is not shown, for clarity. Delaunay edges connecting to a point on the bounding box are shown dashed.

Lemma 2 *For any $n \geq r$, and any k, such that $0 \leq k \leq \lfloor \frac{n}{r} \rfloor$, let $\mathcal{P}$ be the set*

$$\mathcal{P} = \mathcal{E} \ \cup \ \Big\{ \Big(\ell \frac{r}{n} + \frac{1}{2n}, h \Big) \mid 0 < \ell \leq k \Big\}, \tag{9}$$

where $\mathcal{E}$ is the extended input point set, and r and h are defined as in Equations 3 and 4. Then, for all m, such that $kr \leq m < n$, the triangle with vertices $(\frac{m}{n}, 0), (\frac{m+1}{n}, 0), (\frac{kr}{n} + \frac{1}{2n}, h)$ (call it Δ) appears in the Delaunay triangulation of the point set $\mathcal{P}$.

Proof: Analytically, we can compute the circumcenter C of Δ to be at

$$\Big(x_C, y_C \Big) = \Big(\frac{m}{n} + \frac{1}{2n}, \ \frac{1}{8hn^2} \big(4(m - kr)^2 + 4h^2 n^2 - 1 \big) \Big). \tag{10}$$

We must show that the only points from $\mathcal{P}$ contained in the circumcircle of triangle Δ are the vertices of Δ. Because two vertices of Δ lie at $y = 0$, other points at $y = 0$ cannot lie inside the circumcircle. Similarly, points $(\ell \frac{r}{n} + \frac{1}{2n}, h)$ for $\ell < k$ lie outside of the circumcircle. Circumcircle radius R is largest when $k = 0$ and $m = n - 1$, when we obtain

$$R_{\max} = \frac{1}{8hn^2} \Big(4(n-1)^2 + 4h^2 n^2 - 1 \Big) < \frac{1}{2}(h + \frac{1}{h}) = \frac{a}{2}. \tag{11}$$

Because $y_C < R < \frac{a}{2}$, it follows that the circumcircle cannot contain points $(0, a)$ or $(1, a)$. Similarly, because the distance from C to the eight boundary points is at least a, these points are outside the circumcircle. ■

We can now prove our main theorem which will establish the $\Theta(n^2)$ runtime of Ruppert's algorithm under a particular choice of skinny triangles. Lemmas 1 and 2 assert that after adding a Steiner vertex from Lemma 1, a completely new "broom" of $\Theta(n)$ triangles appears to the right of the newly inserted Steiner point. The pattern is recursive: the triangles to the right of a new Steiner point are congruent to the triangles to the right of the last Steiner point in the previous mesh. There are n/r such Steiner points, each time generating $\Theta(n)$ triangles, resulting in $\Theta(n^2)$ generated triangles. The following theorem proves this formally.

Theorem 3 *For any minimum allowed angle $\alpha_{\min} > 0$, define r and h as in Equations 3 and 4. Then, for any $n \geq r$, start Ruppert's algorithm from input $\mathcal{I}$, and consider the following strategy for splitting skinny triangles. After triangulating the extended input, split consecutively triangles Δ_k, defined by vertices $(\frac{kr+r}{n}, 0), (\frac{kr+r+1}{n}, 0), (\frac{kr}{n} + \frac{1}{2n}, h)$, for $k = 0, \ldots, \lfloor \frac{n}{r} \rfloor - 1$. The number of Steiner vertices inserted in this procedure is $\lfloor \frac{n}{r} \rfloor = \Theta(n)$, and the total number of triangles created is at least*

$$\frac{n^2}{2r} - 2n = \Theta(n^2). \tag{12}$$

Note that r only depends on $\alpha_{\min}$, and not on n.

Proof: First, note that all the eight boundary segments remain unencroached as all the Steiner points lie inside the box $[0, 1] \times [0, a]$, which is outside diametral circles of the boundary segments. Therefore, the algorithm need not remedy encroached segments and can split skinny triangles

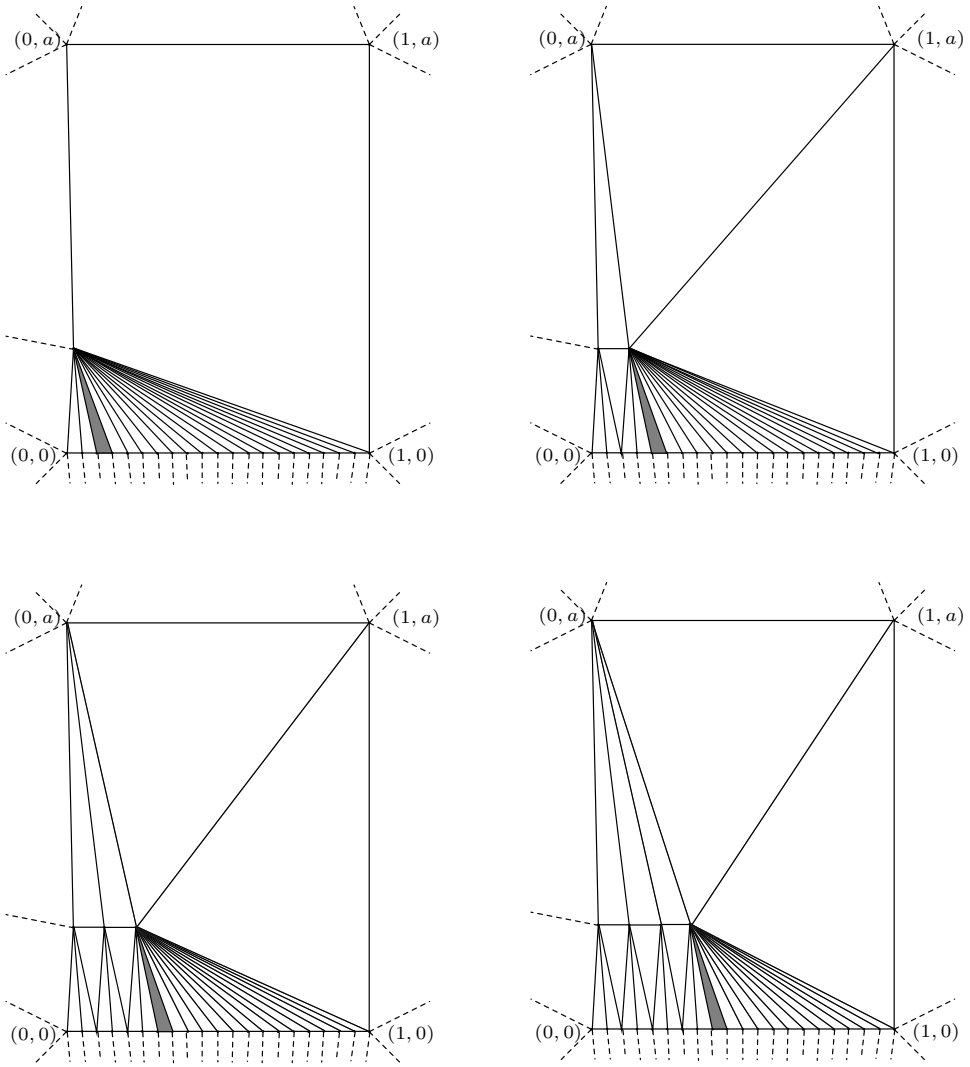

Figure 4: Illustration of Theorem 5 for $r = 2$, $n = 19$, showing the first four splitting steps. The triangles $\Delta_0, \Delta_1, \Delta_2, \Delta_3$ are shaded. The newly added point in a picture is circumcenter of the shaded triangle from the previous picture. To make the picture readable, Steiner points (circumcenters) are drawn much higher than in they lie in reality, i.e., real brooms are much flatter (see the broom in Figure 3).

instead. By Lemma 2 (case $k = 0$), triangle Δ_0 is in the initial mesh. By Lemma 1, triangle Δ_k is skinny for all k, and hence it is a legal choice for the algorithm to split it. By Lemma 2, triangle Δ_{k+1} appears in the mesh after splitting Δ_k, therefore, the pattern is recurring and can be applied consecutively for $k = 0, \ldots, \lfloor \frac{n}{r} \rfloor - 1$. By Lemma 2, triangles

$$\left\{ \left((\frac{m}{n}, 0), (\frac{m+1}{n}, 0), (\frac{(k+1)r}{n} + \frac{1}{2n}, h) \right) \mid (k+1)r \leq m < n \right\} \tag{13}$$

appear in the mesh after splitting triangle Δ_k. These triangles did not exist in the mesh prior to splitting Δ_k, because they all have Δ_k's circumcenter as a vertex, which previously did not exist in the mesh. Hence, the total number of new triangles created at step k is at least $n - r(k+1)$. The total number of triangles created is therefore at least

$$\sum_{k=0}^{\lfloor \frac{n}{r} \rfloor - 1} \Big(n - r(k+1) \Big) \geq \frac{n^2}{2r} - 2n = \Theta(n^2). \tag{14}$$

■

Corollary 4 *The number of output points under Ruppert's refinement, for input $\mathcal{I}$ and under the particular splitting strategy given by Theorem 3, is $\Theta(n)$. The number of triangles generated during the course of the algorithm is $\Theta(n^2)$.*

Proof: By applying the bound from Equation 1, one obtains that the number of output points is at most $\Theta(n)$ (up to a constant, depending only on $\alpha_{\min}$). This can be seen as follows: the local feature size in the region $0 < y < 1/n$ is $1/n$ (up to a constant factor). Therefore, this region contributes $\Theta(n)$ to the integral in Equation 1. The local feature for points with $y > 1/n$ is y (up to a constant factor), therefore the contribution of the region $y > 1/n$ is $\Theta(n)$. The input point at $(\frac{1}{2n}, h)$ does not asymptotically change the integral, which evaluates to $\Theta(n)$. The construction from Theorem 3 created $\Theta(n)$ Steiner vertices already, therefore, the number of output points is $\Theta(n)$.

The number of generated triangles is at least $\Omega(n^2)$ by Theorem 3. Because there are $\Theta(n)$ points in the mesh at every step of the algorithm, we can create at most $\Theta(n)$ new triangles when introducing a new Steiner point. Because we add $\Theta(n)$ Steiner points, the algorithm can create at most $\Theta(n^2)$ triangles during its lifetime. Therefore, in our example, we generate exactly $\Theta(n^2)$ triangles. ■

3 Conclusion

We have shown that for our particular input and output of size n, and a box of dimensions $1 \times \Theta(n)$, there exist a strategy of picking skinny triangles such that Cn^2 triangles are created during the lifetime of the algorithm. The constant C depends only on the minimum allowed angle $\alpha_{\min}$.

In our example, input and output sizes are asymptotically equal, $\Theta(n)$. It remains an open question if quadratic examples can be constructed when the output size is much larger than the input size. Another open question is whether examples can be constructed on input boxes with $\Theta(1)$ aspect ratio, e.g., $[0, 1] \times [0, 1]$.

References

[1] J. R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulato. *Lecture Notes in Computer Science: Applied Computational Geometry: Towards Geometric Engineering*, 1148:203–222, 1996.

[2] J. Ruppert. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. *Journal of Algorithms*, 18(3):548–585, 1995.