Performance comparison of Delaunay Triangulation and refinement algorithms

Kevin van As
4076311
MSc Applied Physics

Laurent Verweijen
4030281
MSc Media and Knowledge

ABSTRACT

We study the performance of several deterministic and randomized algorithms for primality in terms of runtime and error. The tested algorithms are "trial division" (TD), "Wheel-Sieve" (WS), "Solovay-Strassen" (SS), "Miller-Rabin" (MR) and "Angrawal-Kayal-Saxena" (AKS).

Amongst the determinstic algorithms, WS had the best runtime, followed by TD. While AKS proves that primality testing is in P, it performed slowest by far, because it is extremely slow on prime input. As the input grows larger, SS and MR will eventually outperform all deterministic algorithms in terms of runtime. In terms of error, they retain an error, but this error grows negligibly small as the input size increases.

1. INTRODUCTION

In computational science, one wishes to resolve the physics around complex geometries (see e.g. Fig. 1). To do so, one must divide the computational space into computational cells. In the case of complex geometries, people frequently use an unstructured grid, consisting of triangles. In order not to introduce too large of a numerical error as a consequence of the used grid, these triangles must satisfy certain conditions. I.e., the triangles should not have too large of a surface area, since in that case we cannot fully resolve all relevant scales. Also, the triangles should be as close to an equilateral triangle as possible, such that we can properly approximate the gradients of quantities on boundaries of adjacent cells without resorting to expensive algorithms. The latter condition is frequently translated to the requirement that the minimum angle within any triangle should be above a certain criterion.

In this report, we will analyse several Delaunay triangulation and refinement algorithms in terms of their complexity and runtime. In the case of the triangulation algorithms, the problem is given by a set of points and the goal is to find the Delaunay triangulation belonging to that set of points". Furthermore, the solution can be constrained by a set PSLG

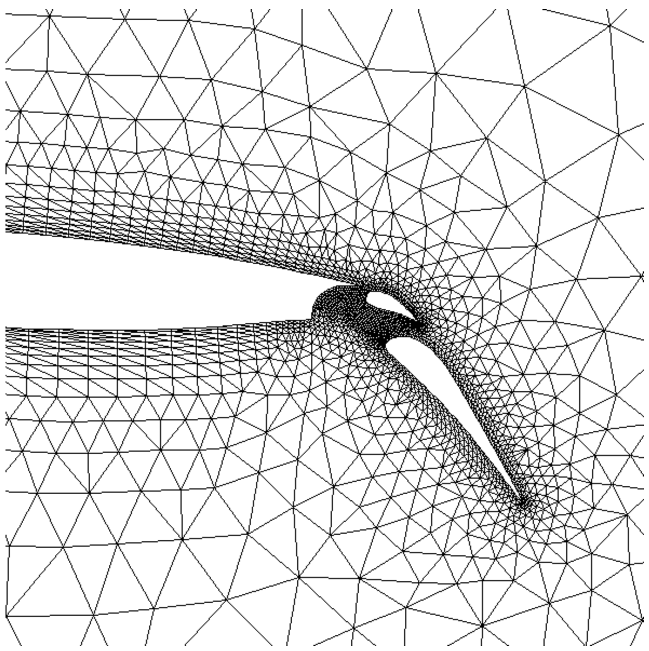

Figure 1: Example of an unstructured grid around some airfoils consisting of solely triangles [3].

(plane straight line graph) of boundary vertices, which may not be crossed and whose vertices must occur in the solution.

For the refinement algorithms the input is specified as a set of PSLG boundary vertices and the goal is to triangulate the area with triangles with a certain minimum quality while respecting the boundary". The Delaunay refinement algorithms that we will discuss are build on top of a Delaunay triangulation algorithm.

A Delaunay triangulation (as introduced by Delaunay in 1934 [2]) is defined as follows:

"Given a set of vertices $P = \{p_1 \ldots p_n\}$, a Delaunay triangulation, D, is a triangulation, T, of P iff for each triangle $t \in T$ the circumcircle contains no vertex $p \in P$ in its interior." In our definition, it is allowed for other vertices to lie on the circumcircle, which imposes an ambiguity on D when more than three points like on the same circle. In this case, any permutation of D will do.

Another way to define the Delaunay triangulation is as the dual graph of the Voronoi diagram. If we let $cell(p_i)$ be the points in P that are closer to p_i than to any other point of P, those cells together form the Voronoi diagram of P.

Firstly, we will look at the Delaunay triangulation algorithms. Afterwards, we will look at refinement algorithms.

2. DELAUNAY TRIANGULATION

We describe two common algorithms for generated a Delaunay triangulation of a given set of points. Both algorithms are incremental.

2.1 Bowyer-Watson

The Bowyer-Watson (B-W) algorithm was independently introduced by Bowyer [1] and Watson [5] in 1981. When incrementally inserting a vertex, B-W removes all triangles that violate the Delaunay property after insertion of the new vertex. Then, the boundary of the created cavity is connected to the new vertex, which results in a new Delaunay triangulation. This algorithm has an expected runtime of $O(n)$ [4].

Algorithm 1 Bowyer/Watson

```
function TRIANGULATE($P$)
    Initialize $T$ with a single large triangle in which all vertices of $P$ are contained
    for all $p \in P$ do
        Let $T'$ be the set of all triangles in $T$ whose circumcircle contains $p$
        Remove triangles in $T'$ from $T$
        Find the total boundary, $B$, of the triangles in $T'$
        For each edge $e \in B$, create the triangle connecting $e$ to $p$ and add it to $T$
    end for
    Remove the initial triangle
    return $D = T$, the Delaunay Triangulation of $P$
end function
```

The Bowyer-Watson algorithm can be extended to create a constrained Delaunay triangulation. In this case, the segments connecting p and each of the edges of the boundary, B, may not intersect with any segment in the PSLG. Ultimately, this boils down to having to check for each PSLG edge in the cavity, T', if this condition is violated. If violated, then the adjacent triangle (the one distant from p) should not be added to the cavity. By keeping a list of all PSLG edges and checking whether a to-be-checked edge is in the PSLG this may be implemented relatively cheaply. Though, the algorithm adaptation is more complicated than for Lawson's algorithm.

2.2 Lawson

Lawson's algorithm makes use of the principle of edge-flips. A theorem states that "either two triangles are both locally Delaunay or their common edge may be flipped and the result is two locally Delaunay triangles". Here, 'locally Delaunay' is defined as 'Delaunay', while only taking these two triangles into account. Lawson's algorithm is easily implemented in 2 dimensions, but is harder to generalize to more dimensions. This algorithm has an expected runtime of $O(n)$ [4].

Algorithm 2 Lawson

```
function TRIANGULATE($P$)
    Initialize $T$ with a single large triangle in which all vertices of $P$ are contained
    for all $p \in P$ do
        Locate the triangle $t$ that contains $p$ and connect the edges of $t$ to $p$ creating 3 new triangles, $T'$
        Remove $t$ from $T$ and add $T'$ to T
        Call Edge-flip on the three edges in $T'$ that do not contain $p$.
    end for
    Remove the initial triangle
    return $D = T$, the Delaunay Triangulation of $P$
end function
function EDGE-FLIP(edge $e$)
    if $e$ is not locally Delaunay then
        Flip $e$ to replace the two adjacent triangles with two new triangles
        Call Edge-Flip on all outer edges of the two adjacent triangles
    end if
end function
```

Lawson's algorithm can be extended to create a constrained Delaunay triangulation. To do so, we must append the if-statement of the edge-flip function by the condition that e is not in the PSLG. Also, the initial triangulation, T, should not violate the PSLG.

3. DELAUNAY REFINEMENT

Delaunay refinement algorithms are algorithms that generate Delaunay triangulations, D, of a given PSLG by inserting points in its interior. The triangles in D must satisfy certain quality criterion, which serve as an input to the algorithm. The most common criteria are the minimum angle present in each triangle and the surface area of each triangle. These should be above resp. below a certain treshold criterion specified by the user. Note that not all refinement algorithms can ensure termination for all tresholds. An optimal refinement algorithm would create as few triangles as possible satisfying the specified criteria.

We will discuss two algorithms for accomplishing this task: Ruppert's refinement algorithm resp. Chew's second refinement algorithm. The algorithms take a Delaunay triangulation satisfying the PSLG as their input and then refine it until all criterea are satisfied. Since these algorithms insert (and possibly remove) points into the triangulation, T, they make use of the algorithms discussed in the previous section to make sure that T maintains the Delaunay property (and thus is equal to D) [4].

3.1 Ruppert's refinement algorithm

In Ruppert's algorithm, two actions may occur. Either the algorithm will split an edge, $e \in$ PSLG into half, or it will insert the circumcenter of a poor quality triangle into the triangulation. An edge is split when its encroached by another vertex. An edge is encroached by a vertex when the vertex is in its diametrical circle. [4]

Note that in each insertion step, the resulting triangulation must remain Delaunay, which can be ensured by the algorithms of the previous section.

Algorithm 3 Ruppert

```
function Refinement(PLSG)
    Start with a Delaunay triangulation, $D$, satisfying the PSLG
    while there is a poor quality triangle, $t$ do
        if $t$'s circumcenter encroaches a segment then
            Split that segment into half
        else
            Insert the circumcenter of $t$ in $D$
        end if
    end while
end function
```

3.2 Chew's second refinement algorithm

Chew's second refinement algorithm also works by repeatedly inserting the circumcenter of a bad triangle. Unlike Ruppert's algorithm, if the circumcenter and the triangle are separated from each other by a PSLG segment, that segment is split instead. Also, any previously-inserted circumcenters in the diametrical circle of that (unsplitted) segment are removed from the triangulation. [4]

Algorithm 4 Chew

```
function Refinement(PLSG)
    Triangulate the vertices bounded by PLSG
    while there is a triangle $t$ that doesn't meet the requirement do
        if a triangle $t$ is separated from its circumcenter $c$ by a segment $s$ then
            Remove all previously inserted circumcenters that are in the interior of the diametercircle of $s$
            Split $s$ in 2 subsegments of equal size
        else
            Insert c in the triangulation
        end if
    end while
end function
```

4. RESULTS

Text

5. CONCLUSION

Text

6. REFERENCES

[1] A. Bowyer. Computing dirichlet tessellations. *Computer Journal*, 24(3):162–166, 1981.

[2] B. N. Delaunay. Sur la Sphère Vide. *Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Matem-aticheskii i Estestvennyka Nauk*, 7:793–800, 1934.

[3] J.-D. Miller. Image extracted from Delaundo: cerfacs.fr.

[4] J. R. Shewchuk. Delaunay refinement mesh generation. Technical report, DTIC Document, 1997.

[5] D. F. Watson. Computing the n-dimensional Delaunay Tessellation with Application to Voronoi Polytopes. *Computer Journal*, 24(3):167–172, 1981.