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 'given a set of points, find the Delaunay triangulation belonging to that set of points'. For the refinement algorithms the problem is given by 'given a

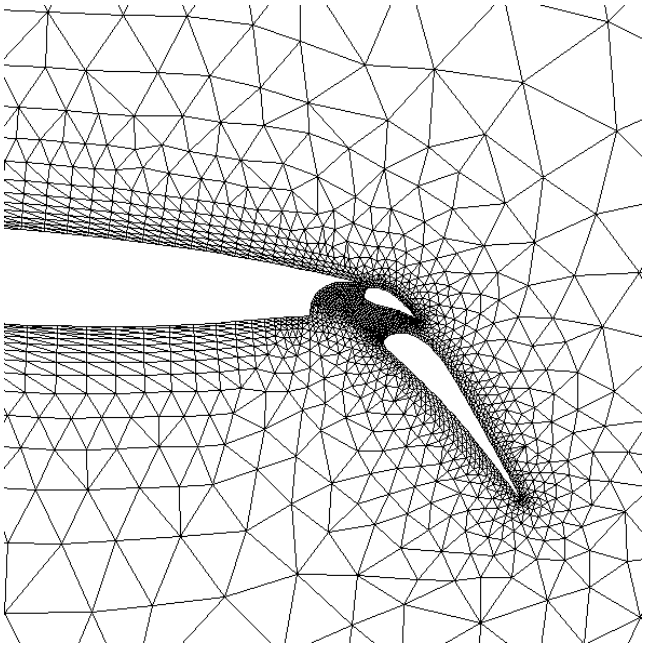

Figure 1: Example of an unstructured grid around some airfoils consisting of solely triangles [1].

set of PSLG (plane straight line graph) boundary vertices, triangulate the area with triangles with a certain minimum quality while respecting the boundary'.

A Delaunay triangulation is defined as follows: "Given a set of vertices $P = \{p_1 \dots p_n\}$, 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.

We will first look at methods to find and maintain a delaunay triangulation of a set of points. Then we will look at delaunay refinement algorithms.

2. DELAUNAY TRIANGULATION

There are 2 common algorithms to obtain a delaunay triangulation. Both algorithms are incremental.

2.1 Bowyer-Watson

Algorithm 1 Bowyer-Watson

function TRIANGULATE(P)
  Intialize T with a single triangle that contains all vertices of P
  for all $p \in P$ **do**
    Let D be the set all triangles in T whose circumcircle contains p
    Remove triangles in D from T
    Split the triangle that contains p
    Retriangulate the vertices in D
  end for
  Remove the initial triangle **return** T
end function

This algorithm has an expected time of $O(n)$.

2.2 Lawson

Algorithm 2 Lawson

function TRIANGULATE(P)
  Intialize T with a single triangle that contains all vertices of P
  for all $p \in P$ **do**
    Locate the triangle t that contains p and connect the edges of t to p
    TODO
  end for
  Remove the initial triangle **return** T
end function

3. RESULTS

Text

4. CONCLUSION

Text

5. REFERENCES

[1] J.-D. Müller. Image extracted from Delaundo: cerfacs.fr.