Assignment 2 : Geometry Processing

Todoskova Darya

June 2023

My project consists in a free-surface 2D fluid solver using incompressible Euler's equations coded in C++.

1 Code Organization

The provided code consists of three files: "main.cpp," "polygon.cpp," and "polygon.h."

- **main.cpp** : initializes the clipPolygon and subjectPolygon objects by creating instances of the Polygon class (with vertices are represented as 2D points). Calls the "clipPolygonfunction" function to perform polygon clipping. Also generates random vertices for the subjectPolygon and assigns weights to each vertex. The "voronoi" function is called to generate a Voronoi diagram using the clipPolygon and subjectPolygon vertices (the Voronoi diagram represents the division of space into regions based on the proximity to the vertices).
- **polygon.cpp** : this file defines the Vector, Edge, and Polygon classes used in the main code. The Polygon class represents a polygon and contains a list of vertices and edges. It contains methods for calculating the polygon's area, checking if a point is inside the polygon, and performing clipping operations (using the Sutherland-Hodgman algorithm).

2 Polygon clipping

I started by implementing a Voronoï diagram using Voronoï Parallel Linear Enumeration (Sec. 4.3.3, lab 6) with polygon clipping(lab 6). To do so, I coded the Sutherland-Hodgman algorithm (p.88) in **polygon.cpp**. By following the lecture notes, I coded two functions *intersect* and *inside*, in order to compute the intersection point between the (finite) edge [prevVertex, curVertex] and the (infinite) line. Here is how it works:

- The Sutherland-Hodgman algorithm takes two polygons, a subject polygon, and a clip polygon, and clips the subject polygon against the clip polygon.
- It iterates over each edge of the clip polygon and clips the subject polygon accordingly.
- For each edge of the clip polygon, it checks if the subject polygon's vertices are inside or outside the clip polygon.
- If a vertex is inside the clip polygon, it adds the vertex to the output polygon.

- If a vertex is outside the clip polygon, it calculates the intersection point between the edge of the clip polygon and the line formed by the previous and current vertices of the subject polygon. It adds the intersection point to the output polygon.
- The resulting output polygon is the clipped polygon, which represents the portion of the subject polygon that lies inside the clip polygon.

I also added the function given in the sample code to save .svg files, to display results:

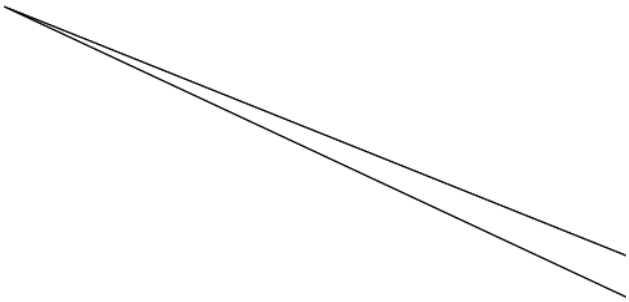

Figure 1: Result after clipping

3 Voronoï diagram

We get a Voronoi diagram using the "voronoi()" function. It uses the clipPolygon and subjectPolygon vertices, with their weights, to create the Voronoi cells. For each point in the subject polygon, the function calculates the Voronoi cell by computing the intersection of half-spaces formed by the midpoints between each pair of points. The resulting Voronoi cells are stored as separate polygons and can be seen in the final Voronoi diagram: Figure 5.

I tried to perform the optimal transport using gradient descent algorithm to find optimal weights for points in a given clip polygon. So I saved several .svg files every 50 iterations til around 2500, and turned it into a GIF (c.f. github repository under the name *50to2500.gif*). I don't think it worked very well.

I then decided to use the L-BFGS library.

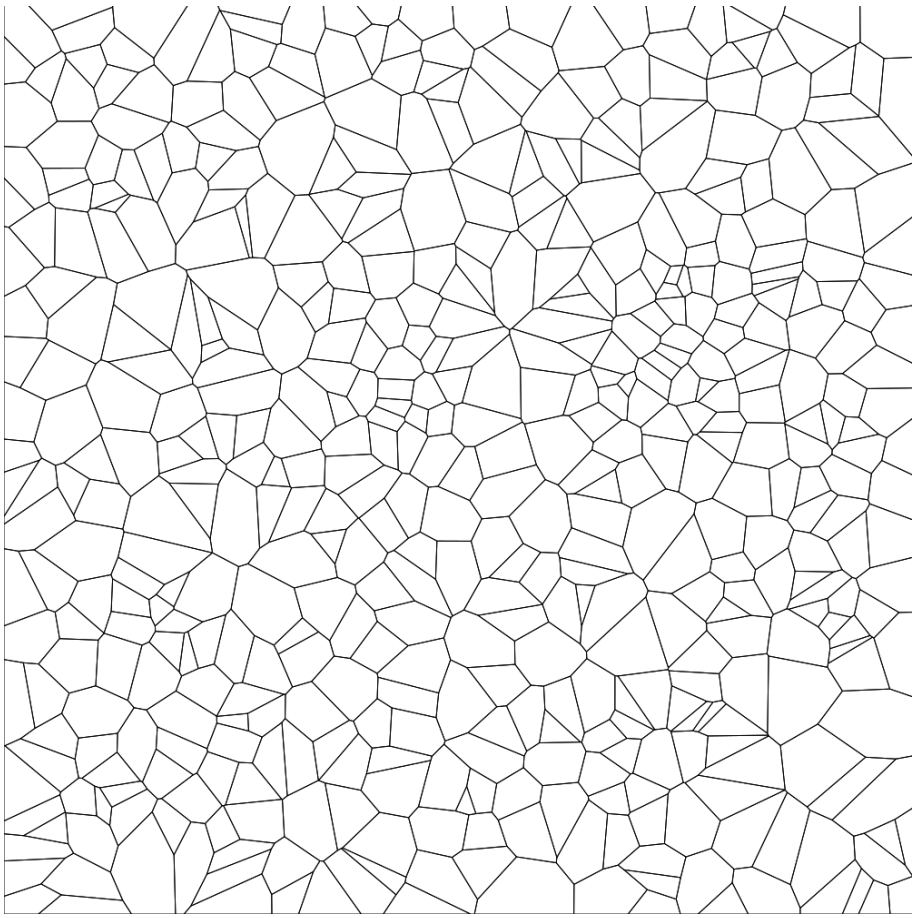

Figure 2: my voronoi diagram (polygon with 500 random vertices)

4 Power diagram

Indeed, I added the power diagram by partitioning cells based on the distances to a set of points and their associated weights (Diracs) (sections 4.4.3 and 4.4.4). Thanks to the libLBFGS library I followed their sample code to get the *static lbfgsfloatval_t evaluate(...)* function.

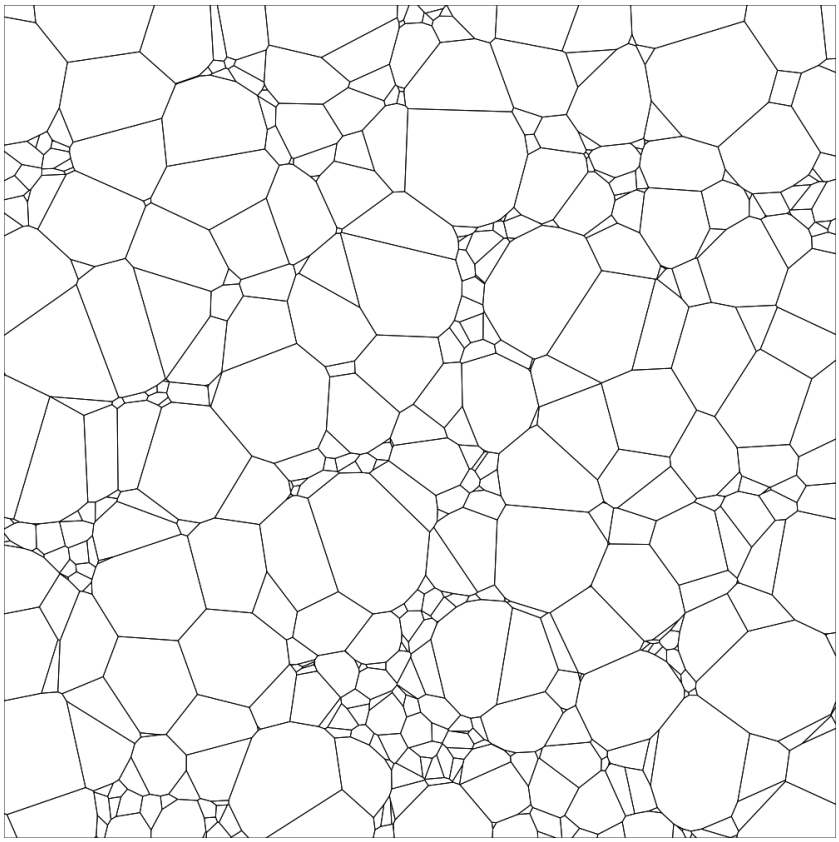

Figure 3: This is before the optimization

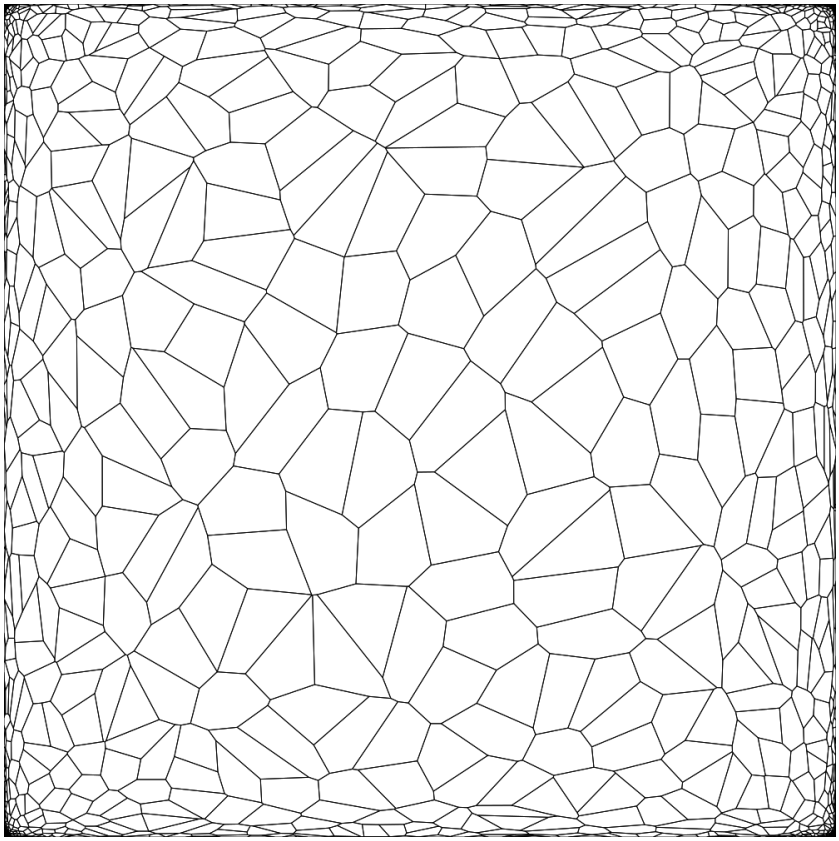

Figure 4: After optimization : this is what I once got, but can't recover :(

5 Stranger Things Vibes

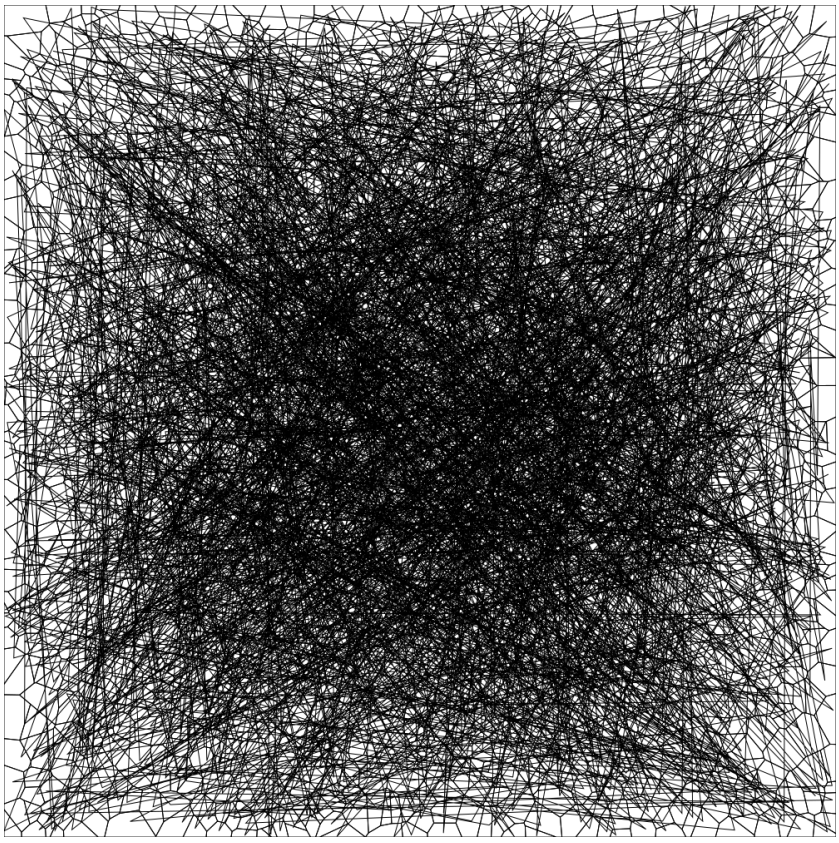

Figure 5: Here is what I get when I currently compile my code, I couldn't figure out where the bug was in time, unfortunately...

6 Course Feedback

I particularly enjoyed this course because of how much application there was, and how varied the assignments were. I just wish it wasn't in C++, but that is very personal due to my lack of ease in the language (hours to solve syntax issues). I also feel like the course is a bit dense for half a semester, but still doable. The lecture notes are also very well written.