Xinyi DAI

Assignment 2

Introduction

In lab 6, We implement the Voronoï Parallel Linear Enumeration algorithm in 2D with Sutherland-Hodgman polygon clipping algorithm and KDTree.
In lab 7, We add weight parametres to our Voronoi diagram and use L-BFGS to solve the semi-discrete optimal transport.

Context:

Nodesets.h :

in this header file, we define the Node for KDtree and some operator for vector<double> which we use to store k-dim values. in every Node we stop the left child, right child , parents nodes and value of k-dim vector, axis records the axis for current node, idx to mark every vector(use when solving the k-nn problem) .

Polygon.h:

Here we dine the Point class for vector in 2-dim, Line class and Polygon class. the intersection function intersect(for line with line) and internorm (for points and line) and the side determine function side and side2

SutherlandHodgman.h:

this includes the very basic Sutherland-Hodgman algorithm for 2 polygons

```
Polygon SutherlandHodgman(Polygon subjectPolygon, Polygon clipPolygon){
```

SH2 for clipping with points with weight.

```
vector<Polygon> SHDiagram(vector<Point> sites,vector<double> ws)
```

SHDiagram to generate voronoi diagram

KDtree.h:

cmp is used to compare vector at certain dimension
PwithD is the structure to store vectors with distance(to certain point, don't have to calculate every time)
knear is the class for the k-nn problem. It's a queue ordered by distance to some point, we also provide .add(), .pop() and .popfront() function for insert or delete elements.
KDTree is the class to search vectors. in this class we have

```
Node* find(vector<double> d)
```

to find the nearest (not d itself) node of point d.

```
knear FindKnearest(vector<double> p, int k)
```

to find the k nearest neighbors of point p(p itself not included)

Voronoi2D.h

it is a voronoi diagram generator using kdtree.
we have vectoP and Pmtovec these two transformer between Point and vector<double>[3]
The function clips is almost the same as in SH, which is used to clip with points
voronoicell is the function to avoid clip with far away points.
voronoiDiagram is the function to generate voronoi Diagram with KDtree.

gfunction.h

It is the g-function and g-gradient function using voronoi diagram for L-BFGS.

optimal.h

It is the optimal function

lloyd.h

It is to generate uniform spread random sites.

fluid.h

It is the stimulation of water drop.

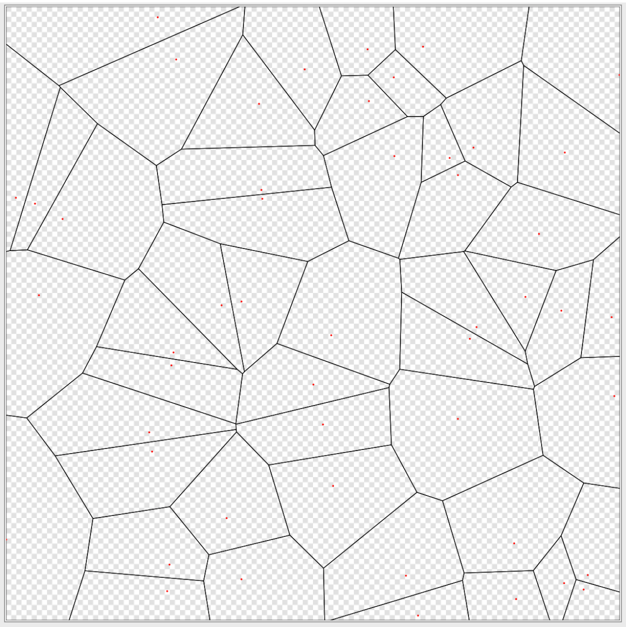
gfluid.h

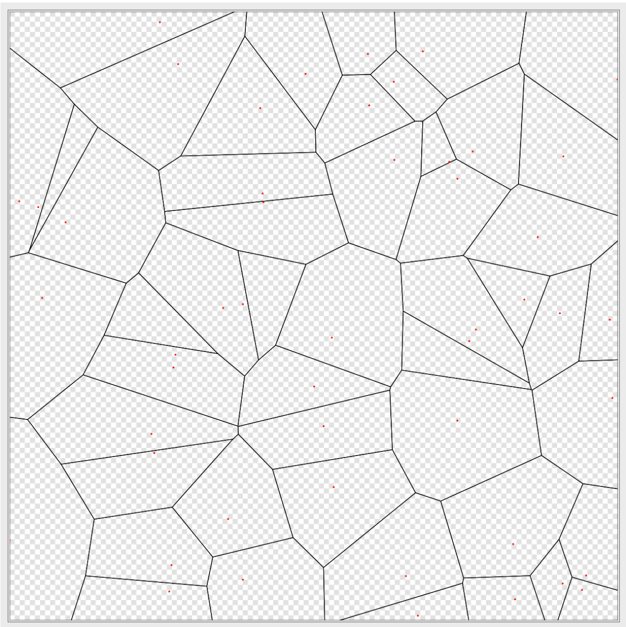
g function and gradient function of L-BFGS for simulation

optfluid.h

L-BFGS for fluid simulation

In lab 6, we generate the voronoi diagram with SH algorithm and KTtree. the following left one is the diagram generated by 50 random sites. (see image/voronoi_KD50.svg)



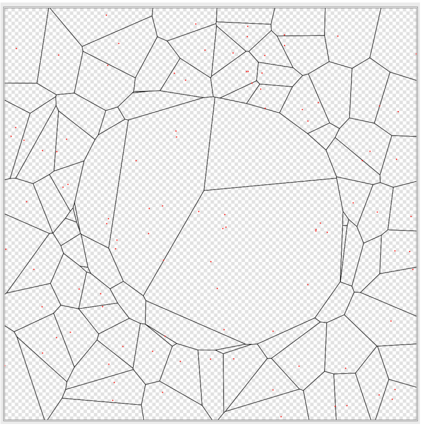


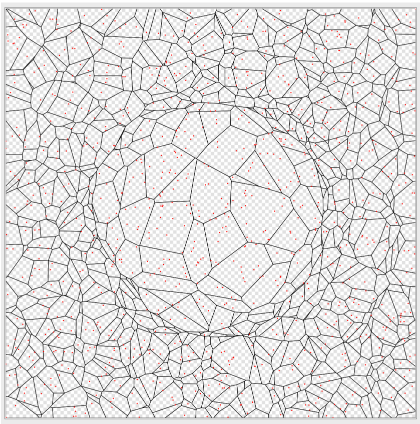
In lab7, we will generate power diagram. The right graph is generated with same sites but with weight of wi= i /25000. (see image/voronoi_50pow.svg)

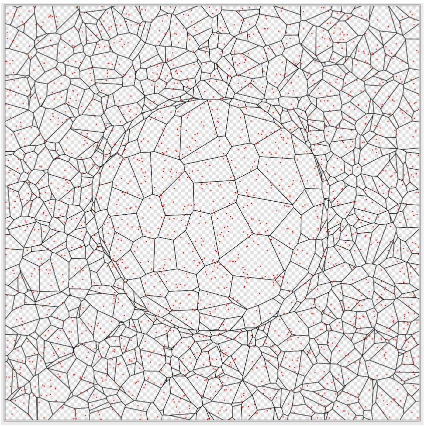
Until now, i tried several sites/random weights using the two method and gets the same diagram. So I suppose both of them are right.

Then we tried to generate optimal transport.

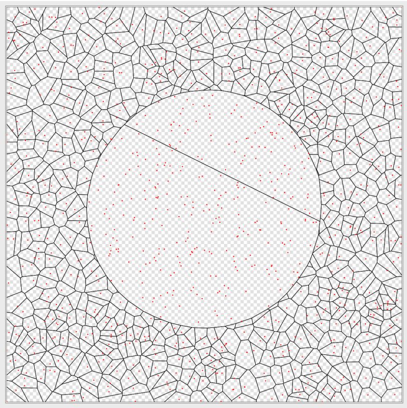
We use $\lambda_i = exp(-\|y_i - C\|^2/0.02)$ the following image is n=100, n=1000, n=2000 from left to right (see image/opt_SH_100, svg,opt_SH_1000.svg, opt_SH_2000.svg)

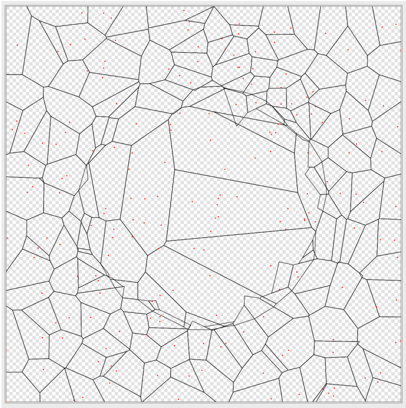


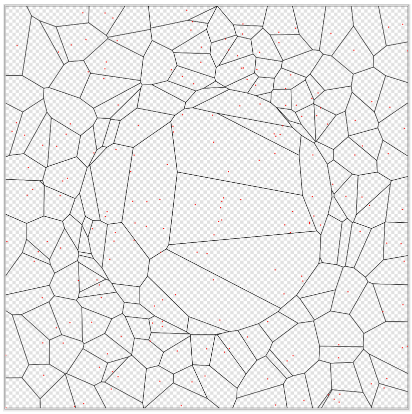




if we take lambda as lecture notes $\lambda_i = exp(-\|y_i - C\|/0.02)$ with n=1000 we have the following graph:(see image/opt_SH_norm)







If we use KDtree in this part, it will give some quite strange image. However, if I use KDtree in LFGBS and use the output weight with SH and KD to generate graph, I could get a good one with SH but wrong one with KD. (centre for KD and right one for SH, see image/opt_KD200, svg,opt_SH200.svg)
(I have spend lots of time debugging it and still haven't solve it. if you have time, could you please see my code?)
*change the #define SH and #define KD in the top of optimal.h and main.h to set the method.

Lab8 is the stimulation of water.
We first set uniformly spread sites then select a circle as initial water particles.
we set mass=200 for water particles and -20 for air particles(we assume that air only move by force from water)
see video/fluid200.svg fluid2000.svg