VENTURINI Riccardo

LORES Clément

TP – Enveloppe convex de points

Objectif du TP :

L'objectif de ce TP était de mettre en œuvre des algorithmes efficaces pour calculer l'enveloppe convexe d'un ensemble de points.

2. La première étape de ce TP était de faire une petite interface graphique pour pouvoir manipuler le nombre de point que nous voulions. Avec gtk, nous avons donc créer un input pour récupérer le nombre de point que nous voudrons ajouter dans notre tableau. A l'aide de la formule mathématiques suivante :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \frac{\sqrt{2}}{4} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

Nous avons pu ajouter nos points dans une forme de losange. En code, cette formule ce traduit comme suivant :

```
Point p;
Point pLos;

p.x = 2.0 * (rand() / (double)RAND_MAX) - 1.0;
p.y = 2.0 * (rand() / (double)RAND_MAX) - 1.0;
pLos.x = p.x * (sqrt(2) / 4) + p.y * (sqrt(2) / 4);
pLos.y = p.x * -(sqrt(2) / 4) + p.y * (sqrt(2) / 4);

TabPoints_ajoute(ptrp, pLos);
```

3. Le balayage de Graham. Tout d'abord nous avons bien du comprendre comment fonctionne graham pour l'implémenter correctement. Nous avons utilisé un tri rapide pour faire fonctionner l'algorithme car rapide et efficace pour les tableaux qui commence à être grand. Nous avons ajouter une structure de pile avec des méthodes pour pouvoir la manipuler

```
typedef struct SPilePoint {
  int taille;
  int nb;
  Point* points;
} PilePoints;

void  PilePoints_init( PilePoints* pile );
int   PilePoints_estVide( PilePoints* pile );
void  PilePoints_empile( PilePoints* pile, Point p );
void  PilePoints_depile( PilePoints* pile );
Point PilePoints_sommet( PilePoints* pile );
// Récupère l'élément juste sous le sommet de la pile.
Point PilePoints_deuxiemeSommet( PilePoints* pile );
void  PilePoints_agrandir( PilePoints* pile );
void  PilePoints_termine( PilePoints* pile );
int   PilePoints_nb( PilePoints* pile );
Point PilePoints_get( PilePoints* pile, int idx );
```

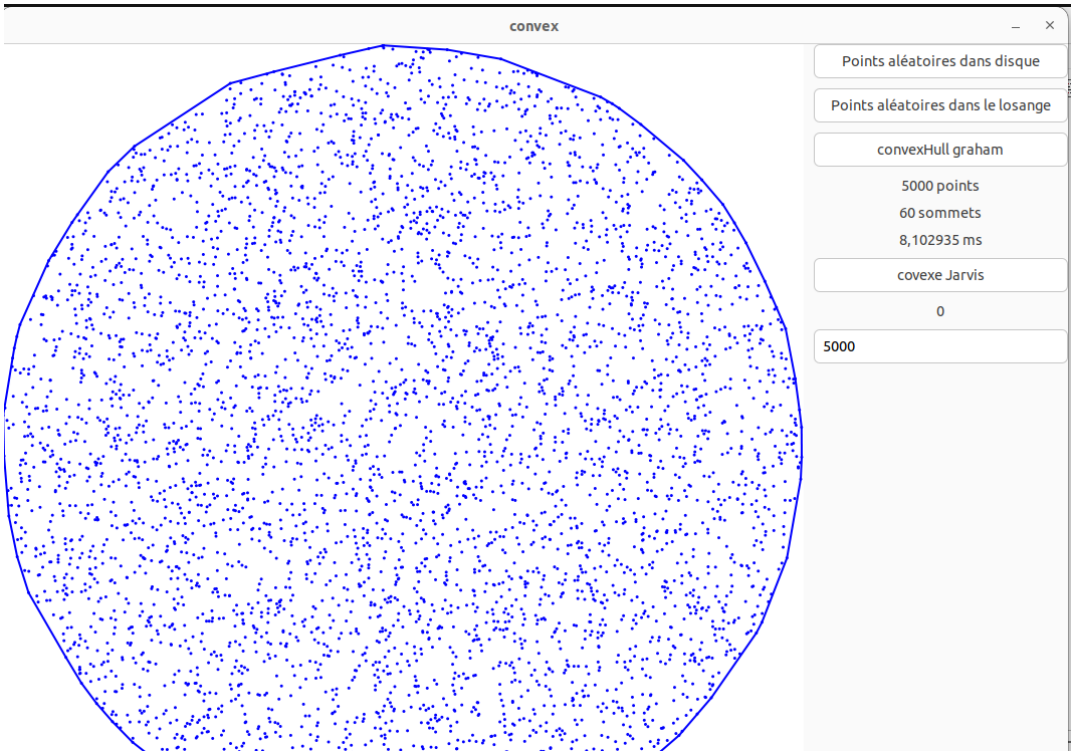
Et enfin, implémenté l’algorithme au complet pour dessiné une ligne qui enveloppe tout les points.

Nous n’avons pas mesurer le temps d’exécution.

4. Algorithme de jarvis,

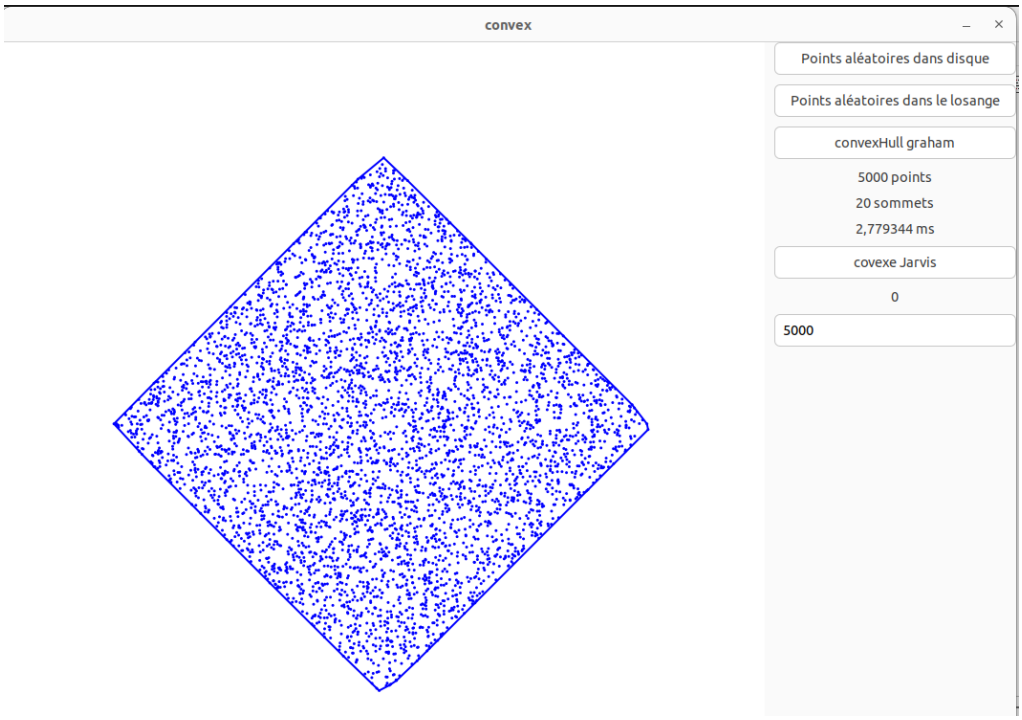
Après avoir implémenté l’algorithme qui dans l’execution est un peu plus simple que graham nous avons pu tester ces performances :

Prenom 5000 points que nous mettons dans un disque puis nous faisons l’enveloppe avec jarvis :



On peut voir que son temps d’exécution est d’environ 8ms.

Testons maintenant avec 5000 points mais dans un losange cette fois ci,



On peut voir que l’algorithme met 4x moins de temps à s’exécuter pour le même nombre de point que dans l’exemple précédent. Nous avons pu aussi tester que plus nous mettons de points, plus l’écart de vitesse entre le temps d’execution du disque et du losange augmente pour jarvis. Avec 100 000 points jarvis était 10x plus rapide dans le losange que dans le disque.

Conclusion :

En conclusion, ce TP nous a permis de mettre en pratique des algorithmes pour calculer l'enveloppe convexe d'un ensemble de points. Nous avons implémenté avec succès les algorithmes de Graham et de Jarvis et mesuré leur temps d'exécution pour différents ensembles de points. Nous avons également créé une interface graphique pour manipuler facilement le nombre de points à traiter.