Institut **S**upérieur d '**I**nformatique,
de **M**odélisation et de leurs **A**pplications

24, Avenue des Landais
BP 10 125
63173 AUBIERE Cedex

Rapport d'ingénieur
projet de 2e année
Filière F4 – Calculs et Modélisation Scientifiques

Implémentation de la triangulation de Delaunay en C++

Présenté par : Coatelen Jérémy et Falk Kevin

Responsables : Jonas Koko et Rachid Touzani

Vendredi 2 Avril 2010

REMERCIEMENTS

Nous tenons à remercier nos tuteurs messieurs Jonas Koko et Rachid Touzani pour leur soutient tout au long de l'élaboration du projet, leurs conseils et pour nous avoir donné l'opportunité d'étendre nos connaissances dans le domaine des éléments finis.

GLOSSAIRE

Le travail repose sur des notions mathématiques et informatiques. Aussi, il est nécessaire de poser quelques définitions pour saisir l'intégralité des termes utilisés par la suite.

Algorithme itératif : un algorithme itératif est composé d'une succession d'instructions.

Algorithme récursif : un algorithme récursif est un processus qui s'appelle lui même.

Cercle circonscrit à un triangle : étant donné un triangle non plat (i.e dont les sommets ne sont pas alignés), le cercle circonscrit à ce triangle est l'unique cercle passant par les sommets de ce triangle.

Complexité algorithmique : il s'agit d'un ordre de grandeur du nombre d'opération à effectuer pour atteindre le résultat en fonction du nombre de paramètres (ici le nombre de points à trianguler). Si n est le nombre de points, une complexité quasi-linéaire signifie un nombre d'opérations ne l'ordre de $n \times \log(n)$ et une complexité quadratique est de l'ordre de n^2.

Convexité : un ensemble E est dit convexe si pour tout couple d'éléments (x,y) de E, tous les points du segment [x,y] appartiennent à E.

Dichotomie : procédé de recherche itératif par lequel on découpe en deux parties l'ensemble de recherche pour le restreindre à chaque étape.

Divide And Conquer (diviser pour régner) : il s'agit d'une stratégie algorithmique visant à réduire récursivement un problème en deux sous-problèmes du même type.

Discrétisation : procédé par lequel on réduit un milieu continu à un ensemble dénombrable de grandeurs de ce milieu.

Maillage : il s'agit de la discrétisation d'un milieu continu. D'un point de vue structure de données, c'est un ensemble de sommets connectés par des facettes polygonales.

Mailleur : outil servant à générer un maillage.

Points cocycliques : on dit que des points sont cocycliques s'ils appartiennent à un même cercle.

Pseudo-langage : langage utilisé pour décrire un algorithme de manière claire et non technique.

Templates : en langage C++, il s'agit d'un patron définissant un modèle de fonction ou de classe dont certaines parties sont des paramètres (type traité, taille maximale).

Triangulation : c'est un maillage dont les facettes sont des triangles.

Simplexe : enveloppe convexe de (n+1) points formant un repère affine d'un espace Euclidien de dimension n, en dimension 3 il s'agit d'un tétraèdre et en dimension 2 il s'agit d'un triangle.

RÉSUMÉ

Le calcul de solutions numériques pour des problèmes physiques est nécessaire dans tous les domaines de la physique moderne. Il peut s'agir de l'étude de la déformation d'une surface, d'évolution de la chaleur ou de la propagation d'onde. Pour obtenir de tels résultats, un des outils les plus utilisés est la méthode des éléments finis. Son principe est de discrétiser l'espace étudié en morceaux pour approximer la solution. La triangulation est donc nécessaire pour obtenir des solutions aux problèmes assez proches de la réalité.

Notre travail a consisté à implémenter en C++ un programme générant une triangulation de Delaunay en deux dimensions, possédant des propriétés avantageuses pour les calculs. Ce programme adapte ensuite le maillage à une forme quelconque.

Le cahier des charges est majoritairement respecté. Une optimisation du temps d'exécution est envisageable pour une utilisation plus efficace.

Mots clés : triangulation de Delaunay, mailleur, éléments finis, diviser pour régner, triangle

ABSTRACT

The calculation of numeric solutions for physical problems is necessary in all the domains of the modern physics. It can be the study of the deformation of a surface, the evolution of the heat or the propagation of electronic waves. To obtain such results, one of the most used tools is the method of finite elements. Its principle is to divide the space in pieces to approximate the solution. So the triangulation is necessary to obtain solutions of problems rather close to the reality.

Our work consisted in implementing a C++ program generating a Delaunay's triangulation in two dimensions, which has advantageous properties for the calculations. This program adapts then the mesh to a given shape.

The specifications are mainly respected. An optimization of the time of execution is possible for a more efficient use.

Keywords: Delaunay's triangulation, mesher, finite elements, divide and conquer

TABLE DES MATIÈRES

TABLE DES ILLUSTRATIONS

I INTRODUCTION

Les outils de mesure actuels concernant l'observation et l'analyse des milieux continus sous contraintes (flexion d'une poutre, déformation d'un pont, envoi d'un liquide sur une surface, etc.) utilisent pour la plupart la méthode des éléments finis. En effet, utilisant un découpage du milieu et définissant ainsi des domaines beaucoup plus simples pour la résolution des équations (ici des triangles), cette méthode peu donner une très bonne solution approchée du problème.

Le découpage du milieu est fait grâce à un maillage c'est à dire en suivant les arêtes formées par liaison d'un ensemble fini de points. Les points sont rarement uniformément répartis sur le domaine car on essaie d'augmenter la densité en points sur les zones d'intérêt (zones de flexion, de déformation plus grande, etc.) afin d'augmenter la qualité de la solution approchée. La forme du domaine définissant le problème peut avoir une forme quelconque, il a donc été nécessaire de prendre en compte ce critère.

L'objectif pour ce projet est de développer un logiciel portable permettant de générer un maillage de qualité, exploitable et convenable pour des domaines en deux dimensions de forme quelconque.

Une première partie est consacrée à la présentation des objectifs du projet, de ses aspects théoriques et de notre méthode de travail. Puis sont détaillées les méthodes d'implémentation que nous avons utilisées. Nous expliquons aussi les algorithmes principaux. Enfin, un exemple d'application est traité pour visualiser le résultat et une analyse de complexité est réalisée.

I PRÉSENTATION ET RECHERCHES

I.1 Cahier des charges

Mr Jonas Koko et Mr Rachid Touzani disposent actuellement d'un mailleur sous Matlab et le désirent sous un environnement indépendant de ce logiciel. Le programme à réaliser doit respecter cette indépendance, le langage choisi pour son développement est le langage C++.

Le programme doit donc générer un maillage exploitable sur un domaine.

Le temps d'exécution du programme doit être minimisé. Le maillage doit respecter certaines contraintes qui sont les suivantes : les triangles générés doivent respecter la condition de Delaunay (voir chapitre suivant), et être resserrés sur les zones d'intérêt. Le programme doit être portable et doit pouvoir utiliser la librairie OFELI permettant la résolution de problèmes par la méthode des éléments finis.

I.2 Triangulation de Delaunay

I.2.1 Présentation

Il existe un très grand nombre de méthodes théoriques et pratiques pour générer une triangulation à partir d'un ensemble de points. Mais toutes ne sont pas exploitables. En effet, on cherche à construire un ensemble de triangles qui soient arrangés de manière assez uniforme et régulière [2].

Par exemple, les triangles générés doivent être les moins allongés possible et former un ensemble homogène.

Dans le cas contraire, les calculs effectués par la méthode des éléments finis seraient moins représentatifs et les résultats moins proches de la réalité.

Le mathématicien russe Boris Delaunay (1890 – 1980) a énoncé pour la première fois la définition de la triangulation qui porte son nom en 1934 dans son œuvre « Sur la sphère vide » [1].

La triangulation de Delaunay impose une contrainte : pour chaque triangle du maillage, son cercle circonscrit ne doit contenir aucun élément de l'ensemble des sommets comme le montre la figure suivante (Illustration 1) :

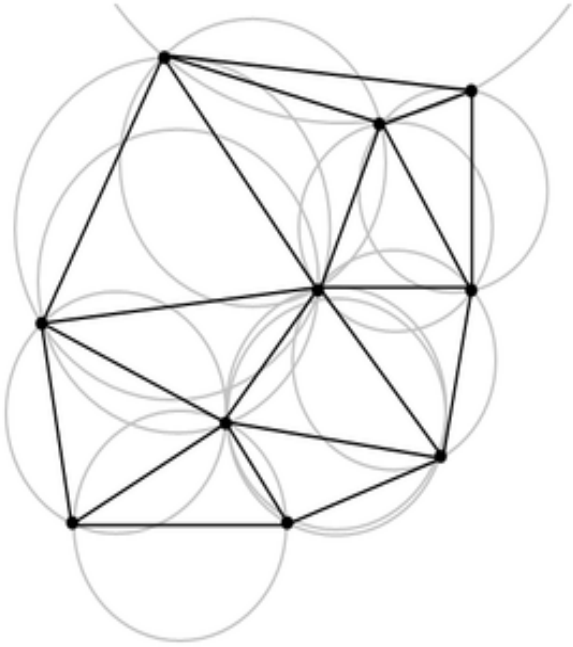

Illustration 1: Représentation de la propriété de Delaunay

Cela implique une construction assez naturelle du maillage, et « arrange » les triangles pour uniformiser leurs dimensions. Les seuls triangles plats qui peuvent advenir se situent aux frontières de l'ensemble de points considéré.

I.2.2 Propriétés

Cette triangulation possède des propriétés intéressantes pour construire un algorithme de génération de maillage [4]. On se restreint ici aux propriétés de la triangulation dans un espace à deux dimensions.

Soit n le nombre de points :

- La triangulation est unique s'il n'y a pas trois points alignés ou quatre points cocycliques.
- La triangulation contient au plus n simplexes.
- Chaque sommet est connecté en moyenne à 6 triangles.
- L'union des simplexes de la triangulation forme l'enveloppe convexe de l'ensemble des points.

I.3 Algorithmes

Il existe deux types d'algorithmes pour générer une triangulation : les algorithmes itératifs d'une part, et récursifs d'autre part.

Dans les deux cas, on part d'un ensemble de points du plan et le résultat obtenu est un ensemble de triangles (la triangulation).

L'approche itérative tend à partir d'un ensemble constitué d'un unique triangle et à ajouter à cet ensemble les nouveaux triangles obtenus pour finir avec l'ensemble désiré.

L'approche récursive adopte une stratégie diviser pour régner. Une division par dichotomie de l'ensemble de points est effectuée avant fusion des ensembles.

I.3.1 Méthode incrémentale

L'ensemble de points étant fini dans le plan, il est inclus dans un intervalle de $\mathbb{R}^2$. On peut ainsi définir deux triangles initiaux formant cet intervalle et englobant l'ensemble des points (*Illustration 2*-1).

Puis, on choisit un point dans l'ensemble initial (Illustration 2-2). On peut localiser le triangle dans lequel il se trouve et créer les nouveaux triangles formés entre le point et les sommets de ce triangle *(Illustration 2*- 3 et 4).

On continue de la sorte jusqu'à ne plus avoir de point à choisir. On obtient la triangulation de Delaunay en sortie.

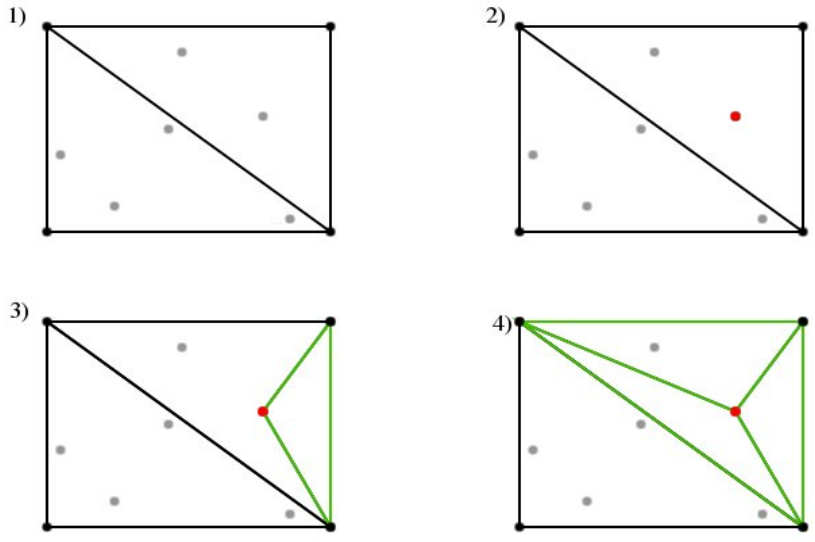

Illustration 2: Première itération de l'algorithme incrémental

La complexité théorique de ce genre d'algorithme est quadratique.

I.3.2 Méthode récursive

Les créateurs de cet algorithme sont Lee et Schachter. Plus tard, il a été amélioré par Guibas et Stolfi et finalement par Dwyer.

On part de l'ensemble initial de points. On impose un ordonnancement aux points selon l'axe des abscisse puis selon l'axe des ordonnées (Illustration 3-1).

Puis on divise cet ensemble en deux sous-ensembles en gardant le même nombre de points (à l'unité près) dans les deux sous-ensembles. On réitère la division sur ces deux ensemble et ainsi de suite jusqu'à obtenir au final des ensembles d'au plus trois points (Illustration 3-2).

Puis on fusionne chaque couple d'ensembles pour obtenir la triangulation de Delaunay relative à ces deux ensembles en supprimant certaines arêtes en cas de nécessité (Illustration 3-3). On obtient ainsi un nouvel ensemble constitué de triangles. On réitère la fusion entre deux ensembles et cela jusqu'à ce qu'il n'y ait plus d'ensemble à fusionner (Illustration 3-4).

Par construction, on obtient la triangulation de Delaunay de l'ensemble de points initial.

La complexité théorique de cet algorithme est quasi-linéaire.

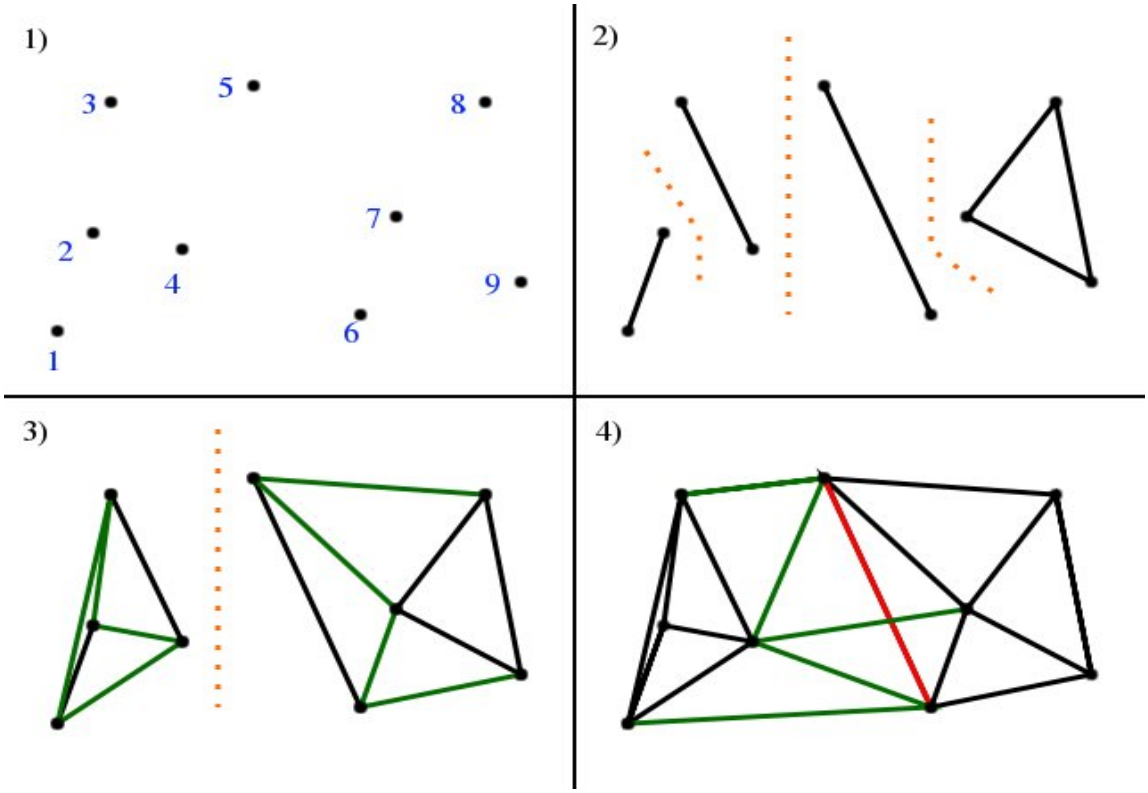

Illustration 3: Division des ensembles puis fusion

Le schéma ci-après résume la structure de l'algorithme en pseudo-langage (Schéma 1).

Remarques : Triangle(E) renvoie un triangle formé des trois points de E.

```
Entrée : E, ensemble considéré
Division(E) :
        Si taille(E) <= 3 Alors
                Renvoyer Triangle(E);
        Sinon
                Diviser E en deux :  E1, E2;
                Renvoyer( Fusion( Division(E1), Division(E2) ) );
Fin;
```

Schéma 1: Algorithme diviser pour régner

I.4 Outils utilisés

I.4.1 OFELI (Object Finite Element Library)

Le programme réalisé est écrit en C++ et les classes utilisées appartiennent à la librairie de calculs par méthode d'éléments finis OFELI [5] écrite en C++. Elle constitue un ensemble d'outils spécifiques pouvant être incorporés dans un programme. Elle inclut des méthodes de traitement de maillages, des solveurs de problèmes linéaires ainsi que des classes servant à des problèmes mécaniques, dynamiques et électromagnétiques.

Cette librairie implémente beaucoup de classes de base pour la triangulation comme Mesh, Element ou encore Point que nous avons utilisé.

Une documentation en ligne est disponible, ce qui rend aisée la compréhension de la structure des classes.

I.4.2 Gnuplot et Gmsh

Lors de l'implémentation du programme, il était plus confortable d'avoir un aperçu visuel du résultat. Pour détecter les erreurs, les cas particuliers à traiter ou encore observer l'évolution temporelle de la construction de la triangulation, nous avons principalement utilisé gnuplot.

Fourni de base dans la plupart des distributions Linux, gnuplot est un outil léger, performant et simple d'utilisation pour l'affichage de graphes (Illustration 4). Les données à afficher (points et

triangles dans notre cas) sont stockées dans un fichier, sous un format simple et naturel.

Gmsh est un outil de génération de maillage 3D rapide et puissant. Les paramètres d'entrée sont modifiables par l'utilisateur. Il possède aussi un outil d'affichage du maillage créé (Illustration 5).

Gmsh nous a été utile pour la comparaison de nos résultats avec ceux qu'il produisait avec les mêmes données.

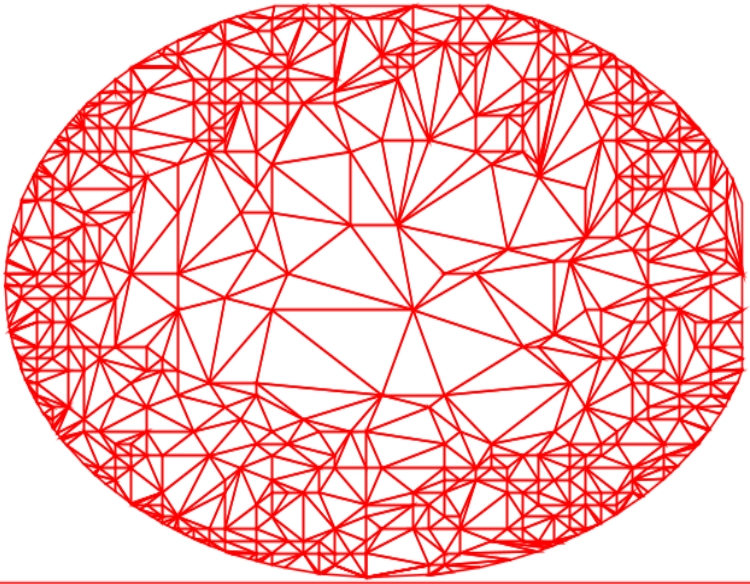

Illustration 4: Triangulation de Delaunay affiché par gnuplot

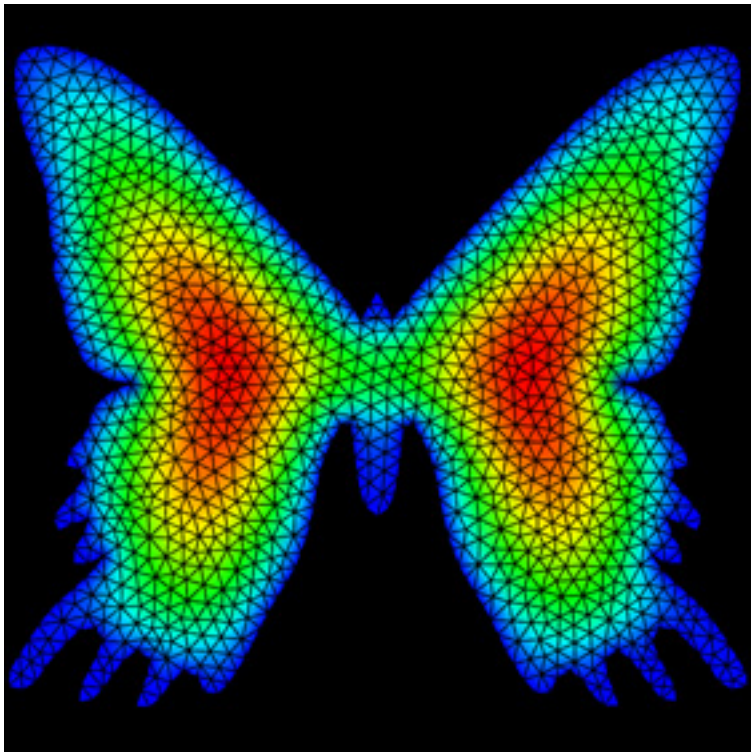

Illustration 5: Affichage d'un maillage généré par Gmsh

I.5 Organisation du travail

Le développement du programme s'est effectuée en deux grandes étapes.

La première, la phase de recherche, consistait à essayer d'élaborer des structures de données, des méthodes, mais aussi des algorithmes adaptés au travail demandé. À partir de sources internet et bibliographiques, nous nous sommes renseignés sur les différentes méthodes de programmation de l'algorithme de triangulation de Delaunay. Nous avons commencé à programmer le premier algorithme, de complexité quadratique. Puis nous avons découvert l'algorithme diviser pour régner, plus efficace.

Nous programmions en parallèle et indépendamment sur les mêmes parties, avant de réunir et d'analyser nos travaux. Le partage des fichiers s'effectuait depuis un serveur Subversion, accessible sur internet. Cela nous permettait de mettre à jour notre propre version à distance et observer les différences entre les deux codes. Nous analysions nos travaux respectifs avant d'adopter la meilleure solution et fusionner les recherches.

Puis, une fois les classes déterminées, les structures définies et les différentes fonctions établies, nous nous sommes partagés le travail. La deuxième étape était donc de programmer en parallèle sur des parties différentes.

II DÉVELOPPEMENT

II.1 Structure du programme

II.1.1 Description

Pour générer une triangulation, il est nécessaire d'avoir un ensemble de points repérés par leurs coordonnées dans le plan.

En entrée, notre programme prend :

- Un quadruplet de réels (Xmin,Ymin,Xmax,Ymax) qui représente l'intervalle I de $\mathbb{R}^2$ dans lequel on veut générer la triangulation.
- Une fonction densité sur cet intervalle. Il s'agit d'une fonction d : I → [0,1] qui représente la probabilité qu'un point de I existe en fonction de ses coordonnées. Cette densité aura une valeur proche de 1 sur les zones où l'on veut créer beaucoup de points (zone de grand intérêt) et une valeur proche de 0 dans une zone où le nombre de points est désiré faible.
- Une fonction distance définie sur I. Il s'agit d'une fonction d' : I → $\mathbb{R}$ qui représente la distance signée d'un point à un ensemble. Plus concrètement, cette fonction aide à définir une forme quelconque sur l'intervalle comme l'objet à mailler. Étant donné un point P de I, d'(P) sera positif si le P se situe en dehors du domaine défini par l'ensemble, négatif s'il se situe à l'intérieur et nul à sa frontière.

À partir de ces données, le programme quadrille l'intervalle et créé une grille. Sur cette grille sont générés l'ensemble de points qui servira de base pour la triangulation (Illustration 6).

À cet ensemble de points sont retirés ceux qui ne respectent pas la densité. Ainsi, on obtient un nuage de points qui représente globalement la densité souhaitée (Illustration 7).

Puis, l'algorithme de triangulation de Delaunay est appliqué sur ce nuage de points triés. On obtient ainsi un maillage sur tout l'intervalle I (Illustration 8).

Enfin, le programme adapte ce maillage en utilisant la fonction distance d'. Pour chaque triangle du maillage, on calcule les distances des ses sommets avec la fonction d'. Ainsi, on sait quels sont ses sommets qui se trouvent à l'extérieur de l'ensemble. Si tous ses sommets sont à l'extérieur, cela implique que le triangle tout entier est à l'extérieur et on le supprime du maillage final. Dans le cas particulier où un triangle chevauche la frontière de l'ensemble, il faut projeter les sommets situés à l'extérieur sur la frontière (Illustration 9).

En sortie, le programme génère un fichier contenant l'ensemble des triangles de la triangulation de Delaunay restreint à l'ensemble désiré. Le format des triangles est du type :

Sommet1

Sommet2

Sommet3

Sommet1

<u>Remarque</u> : on écrit une nouvelle fois le Sommet1 pour simplifier le tracé des triangles avec gnuplot. En effet, gnuplot trace un segment entre deux sommets. Pour afficher le triangle entier, il faut tracer les segments [Sommet1,Sommet2], [Sommet2,Sommet3] et [Sommet3,Sommet1].

II.1.2 Schéma général

Les illustrations suivantes représentent les différentes étapes de l'algorithme.

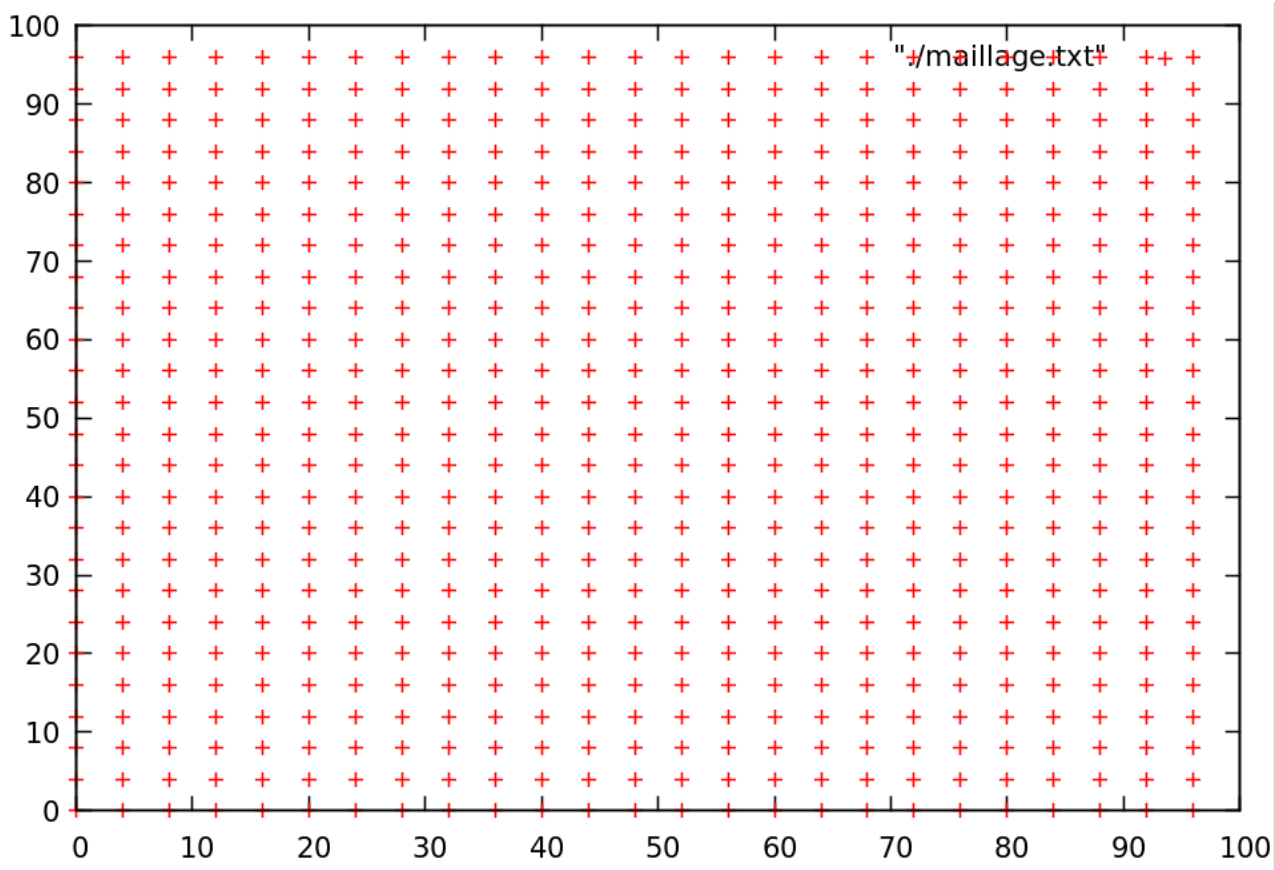

Illustration 6: Génération de la grille

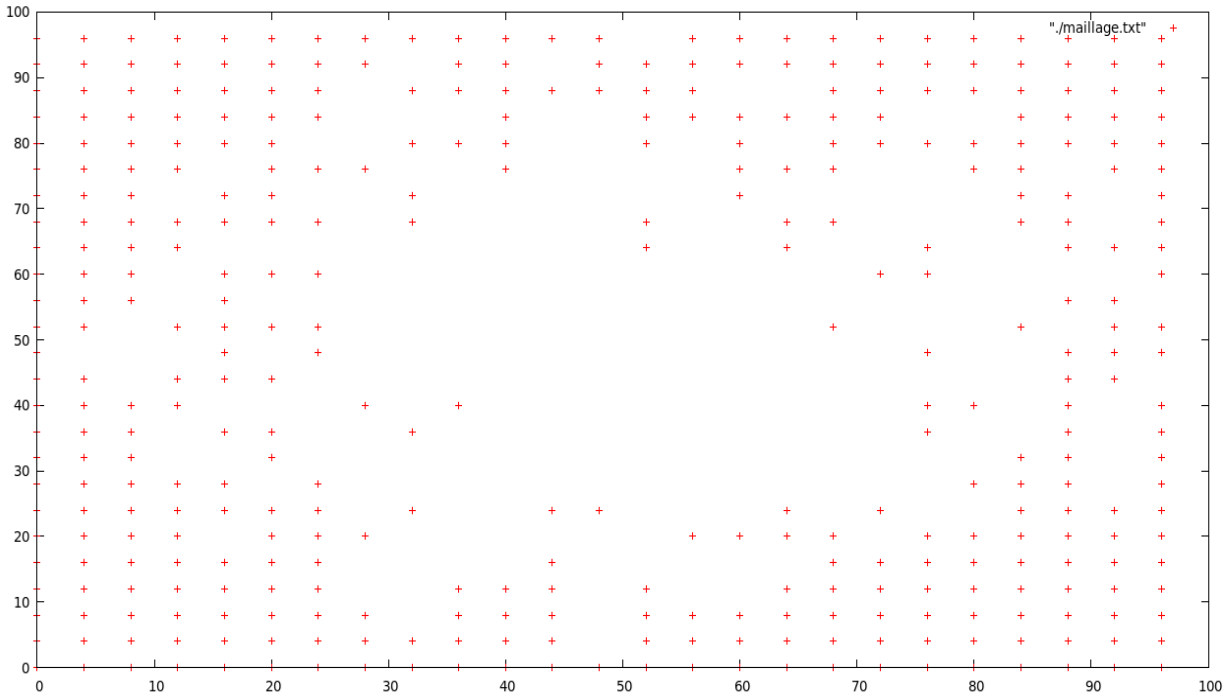

Illustration 7: Suppression des points grâce à la fonction densité

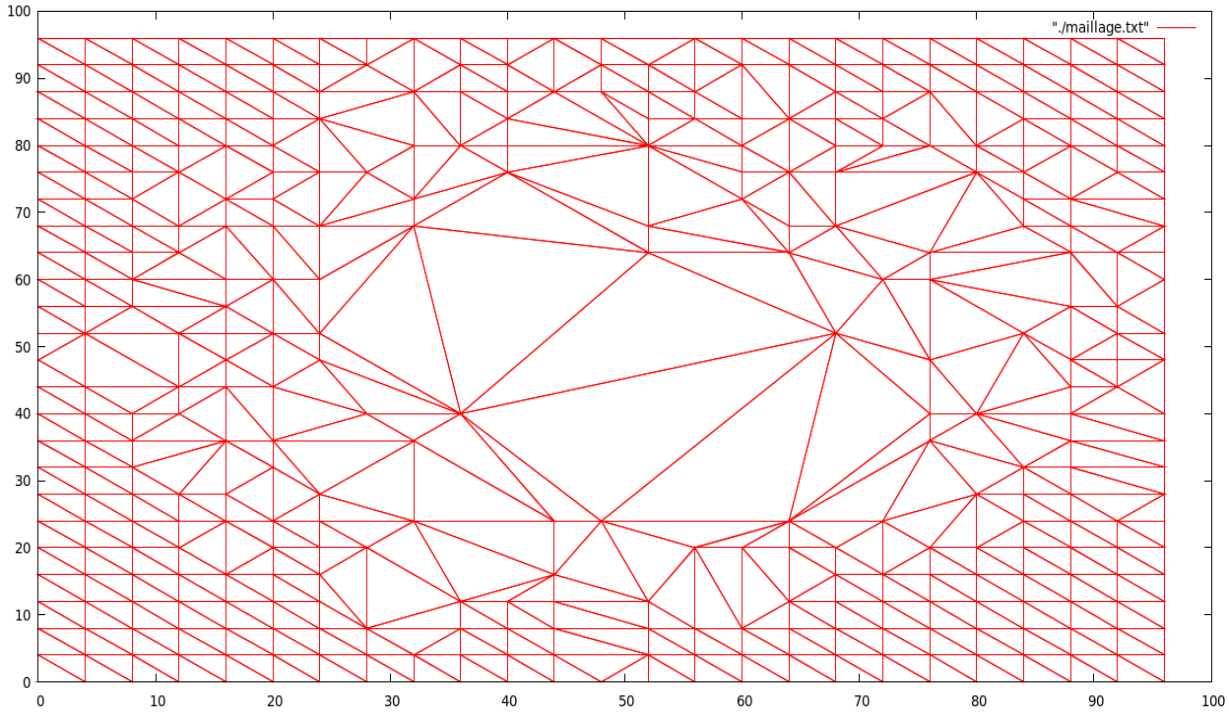

Illustration 8: Triangulation de Delaunay

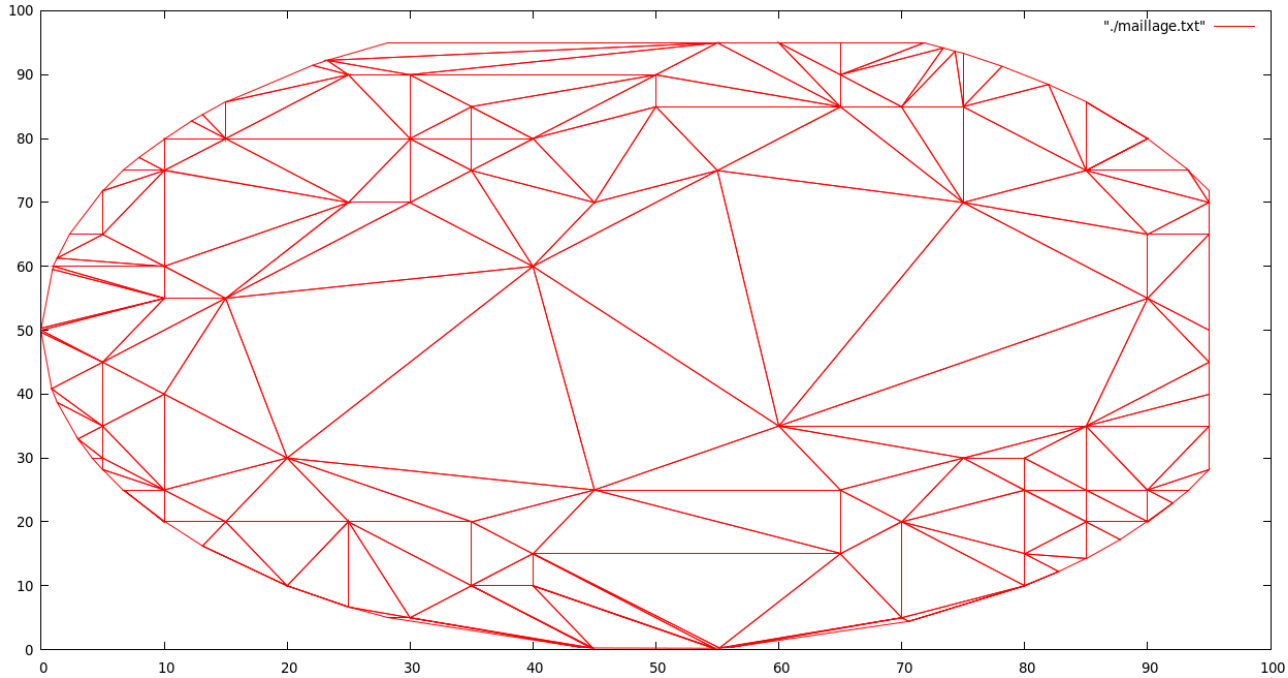

Illustration 9: Adaptation à la forme grâce à la fonction distance

II.2 Triangulation de Delaunay

II.2.1 Schéma général des classes

Voici la structure générale des classes utilisées (Illustration 10). La plupart hérite de classes appartenant à la librairie OFELI.

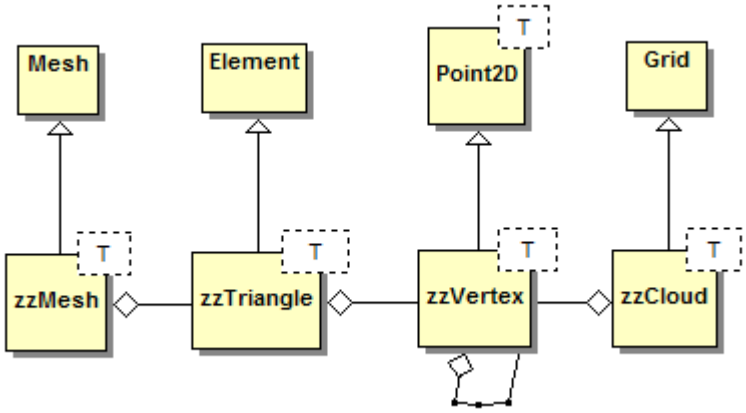

Illustration 10: Diagramme de classes UML

La classe Mesh permet de stocker et de manipuler des maillages d'éléments finis. zzMesh est une classe qui hérite de Mesh que nous avons implémenté. Son rôle est de gérer toutes les actions relatives au maillage comme l'ajout et la suppression d'un triangle.

Grid est une classe qui permet de gérer une grille. zzCloud en hérite et peut ainsi gérer l'ensemble des points d'une grille.

Element a pour rôle de gérer et de stocker élément fini. zzTriangle, classe fille de Element, est la classe qui représente un élément fini te type triangle.

Point2D définit un point dans le plan. zzVertex en hérite et comprend toutes les méthodes relatives à la gestion d'un point du plan.

II.2.2 Déroulement détaillé de la fusion

Le fichier « delaunay.cpp » contient les fonctions relatives à la triangulation.

Comme nous l'avons vu précédemment, l'algorithme réalise d'abord une division récursive de l'ensemble des points avant de fusionner ces ensembles deux à deux [3].

Pour fusionner deux ensembles, l'algorithme n'a besoin de connaître que leur enveloppe convexe et l'ensemble des triangles. Le principe est de partir du segment le plus bas reliant les ensembles et de le faire remonter en construisant au fur et à mesure des triangles respectant la condition de Delaunay.

L'algorithme de fusion est détaillé dans le schéma ci-après (Schéma 2).

Entrées : ***E1, E2, les enveloppes convexes des ensembles à fusionner.***

Mesh, l'ensemble des triangles existants.

Sortie : ***E, l'enveloppe convexe après fusion de E1 et E2.***

```
Tant Que non Stop
    Soit (BG,BD) la base inférieure de la fusion
    Soit (HG,HD) la base supérieure de la fusion
    Rechercher le candidat CG dans E1
    Rechercher le candidat CD dans E2
    Supprimer les triangles qui croisent (CG,BG,BD) et ceux qui croisent (CD,BG,BD)
    Si CG = HG et CD = HD : Stop
    Sinon:
        Si CD est dans le cercle circonscrit à (CG,BG,BD) :
            Ajouter (CD,BG,BD) à Mesh
            BD ← CD
        Sinon
            Ajouter (CG,BG,BD) à Mesh
            BG ← CG
Fin Tant Que
```

Schéma 2: Algorithme Fusion

<u>Remarques</u> : La base inférieure de la fusion est le couple de points (BG,BD) (pour Bas Gauche et Bas Droit) tels que tout point de E1 et tout point de E2 se trouvent au dessus de la droite (BG,BD) . Symétriquement, la base supérieure est le couple de points (HG,HD) (pour Haut Gauche et Haut Droit) tels que tout point de E1 et tout point de E2 se trouvent au dessous de la droite (HG,HD) (Illustration 11).

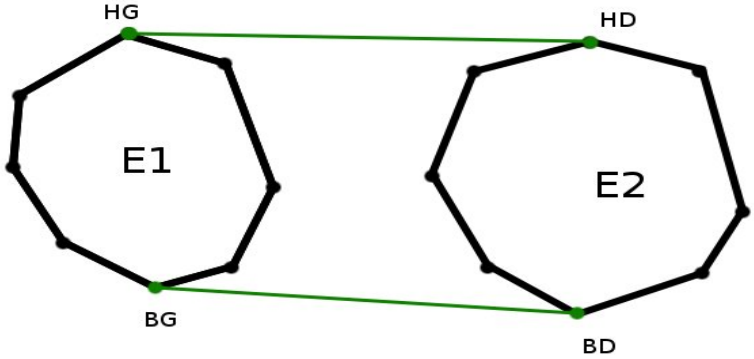

Illustration 11: Bases inférieure et supérieure de la fusion

Pour « monter », la base doit déterminer un candidat parmi les voisins de BG et ceux de BD. Cette détermination du candidat dans l'ensemble de gauche se fait symétriquement de la même manière que dans l'ensemble de droite. Le schéma suivant (Schéma 3) décrit l'algorithme de recherche de candidat pour l'ensemble de gauche, donc à partir de BG :

Entrées : ***BD, autre point de la base***

Sortie : ***CG, candidat de l'ensemble de gauche***

Initialisation : ***Angle_min ← 180°***

Pour chaque voisin V de BG :

Si $\widehat{BD, BG, V}$ ***< Angle_min :***

Angle_min ← $\widehat{BD, BG, V}$

CG = V

Fin Pour

Angle_min←180°

Pour chaque voisin V de BG :

Si $\widehat{CG, BG, V}$ ***< Angle_min :***

Angle_min ← $\widehat{CG, BG, V}$

C2 ← V

Fin Pour

Si C2 appartient au cercle circonscrit à (CG,BG,BD), alors CG ← C2

Schéma 3: Algorithme de détermination du candidat dans l'ensemble de gauche

La base (BG,BD) évolue ainsi en remontant vers (HG,HD). Les nouveaux triangles créés sont ajoutés à la triangulation et les anciens triangles chevauchant les nouveaux sont supprimés. Il faut aussi noter que les points situés sur les faces en regard des deux ensembles à fusionner ne seront pas ajoutés à l'enveloppe convexe de l'ensemble final (Illustration 12).

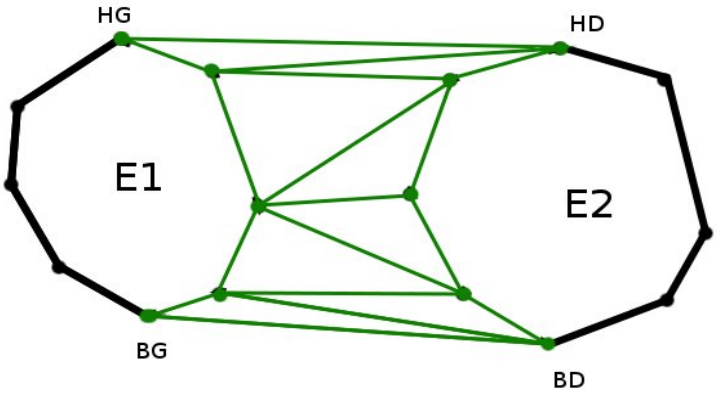

Illustration 12: Fusion des deux ensembles

II.3 Adaptation du maillage

Il reste à adapter le maillage à la forme définie par la fonction distance. L'algorithme d'adaptation choisi est le suivant (Schéma 4):

Pour chaque triangle (A,B,C) de T

Si les 3 sommets sont à l'extérieur : Supprimer (A,B,C) de T.

Si 2 sommets A et B sont à l'extérieur : Soit PA et PB les projections respectives sur la frontière de A et B selon $\overrightarrow{AC}$ ***et*** $\overrightarrow{BC}$ ***. Supprimer (A,B,C) à T. Ajouter (PA,PB,C) à T.***

Si le sommet A seulement est à l'extérieur : Soit PA1 et PA2 les projections sur la frontière de A selon respectivement $\overrightarrow{AB}$ ***et*** $\overrightarrow{AC}$ ***. Supprimer (A,B,C) à T. Ajouter (PA1,B,C) et (PA2,B,C) à T.***

Sinon : Ne rien faire.

Fin Pour

Schéma 4: Algorithme d'adaptation

On voit ici que la difficulté repose sur le chevauchement des triangles avec la frontière.

Dans ce cas, les deux situations à envisager (deux point à l'extérieur ou un unique à l'extérieur) utilisent une projection selon une direction (Illustration 13, Illustration 14). Comme on veut que le point extérieur soit projeté sur la frontière, en notant A le point à l'extérieur et B le point à l'intérieur, la direction de projection est $\overrightarrow{AB}$. Voici le schéma de principe de la projection (Schéma 5):

Entrée : Point A externe au domaine. Point B interne au domaine.

Sortie : Point A projeté sur la frontière

Faire

Soit I le milieu de [A, B].

Si d(I)>0 : A←I

Sinon : B←I

Tant que |d(I)| > ε

Schéma 5: Algorithme de projection d'un point sur la frontière selon une direction

On remarque qu'il s'agit d'une recherche d'un point sur un segment. Ce type de recherche est rapide et précise. L'epsilon choisi dans le programme est de l'ordre de 10^{-1} .

Il est important de noter que les triangles modifiés sur la frontière peuvent ne plus respecter la condition de Delaunay. Cependant, cette restructuration ne peut pas altérer la qualité de la triangulation car cela concentre les points sur la frontière qui est une zone de grand intérêt.

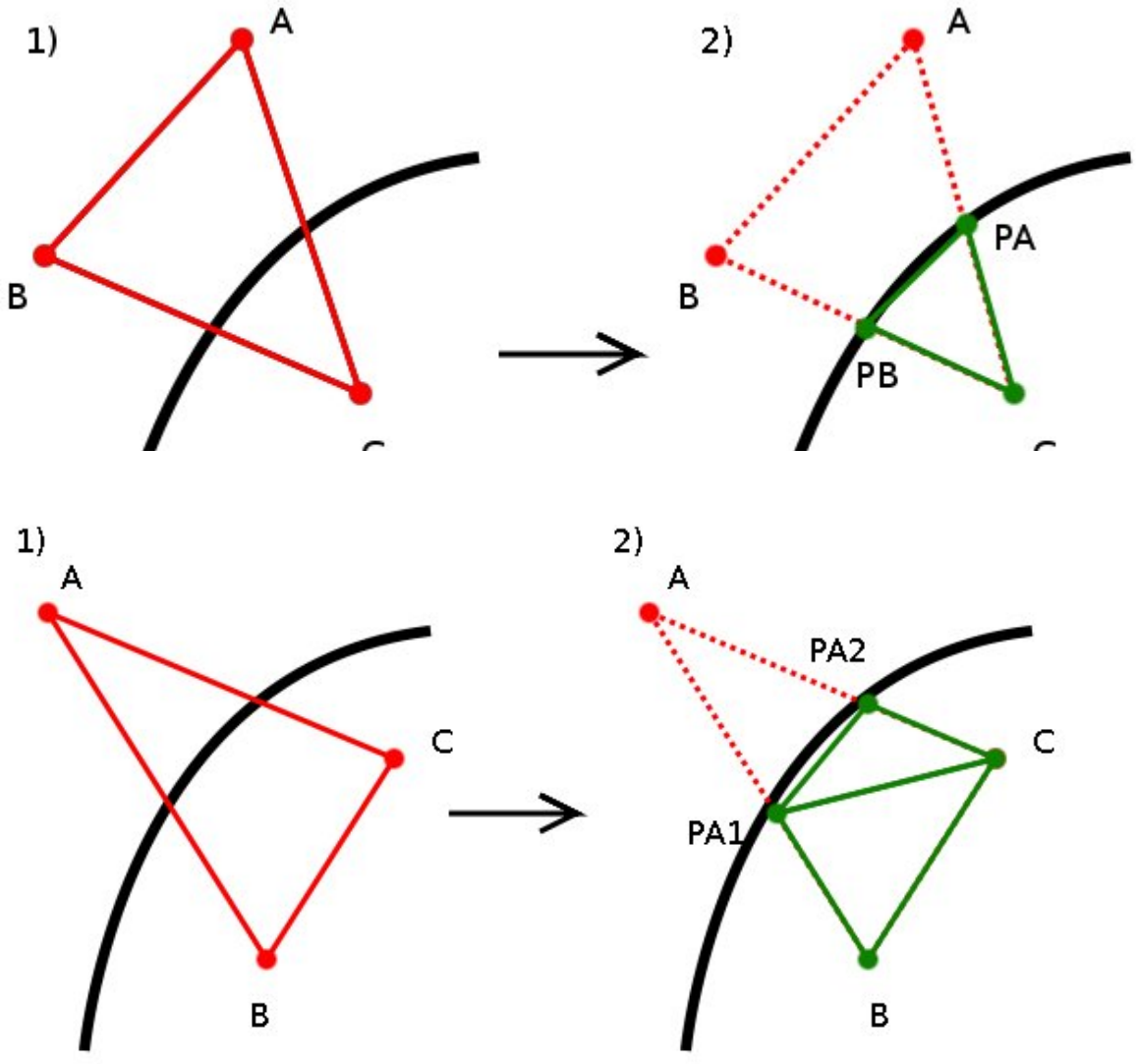

Illustration 14: Projection dans le cas où un seul sommet est à l'extérieur

III RÉSULTATS ET ANALYSES

III.1 Application sur un exemple

Nous allons maintenant utiliser le programme dans une étude concrète. L'objet considéré est une pièce circulaire et on désire étudier sa déformation sous l'effet d'une force appliquée sur une zone de son contour (Illustration 15).

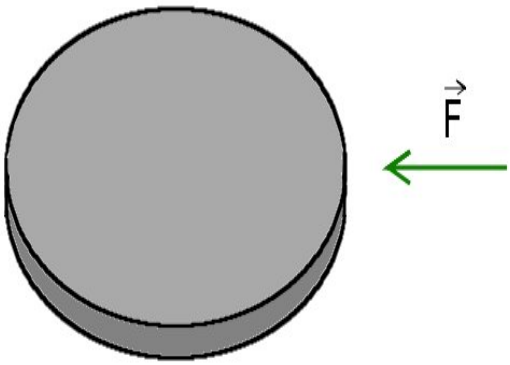

Illustration 15: Pièce subissant une force latérale

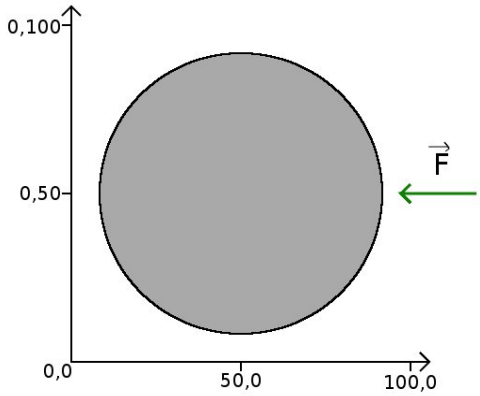

Illustration 16: Pièce dans le plan

On se restreint à une étude de la déformation dans le plan de la pièce. Il nous faut tout d'abord définir un intervalle du plan. Prenons le pavé I=[0,100]×[0,100] (Illustration 16). Le quadruplet (Xmin,Ymin,Xmax,Ymax) est donc (0,0,100,100).

Dans le repère créé, la pièce a pour centre le point (50,50) et pour rayon 40.

Dans cet intervalle, compte tenu de la géométrie de la pièce, il est aisé de définir la fonction distance qui la représente. Ici :

Pour tout (x,y) de I, d'(x,y) = $(x-50)^2 + (y-50)^2 - 40^2 = (x-50)^2 + (y-50)^2 - 1600$

Il faut aussi choisir le nombre de points que la grille initiale comportera, par exemple : 50 points sur l'axe des abscisses et 50 points sur l'axe des ordonnées.

Il nous faut définir une fonction densité pour la génération des points du maillage. Comme la zone d'intérêt se situe autour du point d'application de la force (90,50), la densité doit être plus importante dans cette région. Prenons la fonction suivante :

Pour tout (x,y) de I, d(x,y) = $\dfrac{(x-10)^2+(y-50)^2}{10000}$

Autour du point d'intérêt (90,50), la valeur de d est très proche de 1 et plus on s'éloigne de ce point, plus la densité décroit. Une division par une grande valeur est nécessaire pour que les valeurs soient comprises entre 0 et 1. Ici, cette valeur a été déterminée empiriquement pour avoir une bonne

concentration de points autour de la zone d'intérêt et peu de points ailleurs (à gauche de la pièce).

Les paramètres étant déterminés, on peut les entrer dans le programme avant de le lancer.

On peut observer (Illustration 17) le résultat obtenu.

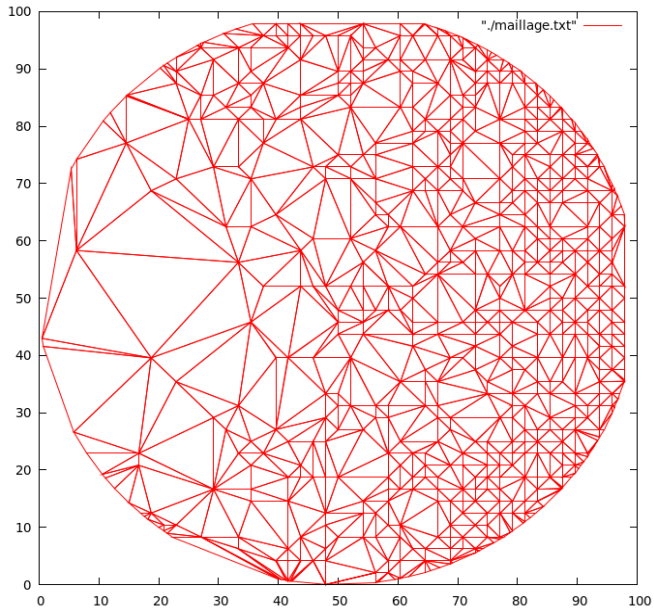

Illustration 17: Affichage du résultat par gnuplot

À partir de la liste des triangles générée, l'utilisateur peut appliquer une méthode de résolution par éléments finis pour calculer la déformation causée par la force F sur la pièce.

III.2 Analyses

III.2.1 Évaluation de la complexité

Il est intéressant d'analyser la complexité du programme pour évaluer son efficacité.

Le programme mêle des fonctions itératives et récursives et il devient très vite compliqué d'évaluer avec précision sa complexité théorique. Cependant une rapide analyse de la structure du programme permet d'en donner une estimation.

La génération de la grille est fonction de la densité et du nombre de points initiaux voulus. Avec m le nombre de points sur l'axe des abscisses et n le nombre de points sur l'axe des ordonnées, la génération de la grille est de l'ordre de mxn calculs.

Soit N le nombre de points restants après génération de la grille. La division de Delaunay se fait de manière récursive et admet une complexité théorique en N×log(N).

La fusion de deux ensembles convexes, appelée à chaque étape de la division, requiert :

- Une recherche des bases inférieure et supérieure entre les ensembles : avec p et q les

nombres respectifs de points du premier et du second ensemble convexe, la complexité est de l'ordre de $p^2 + q^2$.

- Pour chaque point considéré, il faut rechercher parmi ses voisins le candidat suivant pour déterminer la nouvelle base. Cela et linéaire en fonction du nombre de voisins. Cette recherche est effectuée 4 fois par étape.
- Lorsqu'un candidat est trouvé et qu'un triangle est ajouté, il faut chercher parmi tous les triangles déjà construits s'il n'y a pas chevauchement de triangle. Cela est linéaire en fonction du nombre de triangles existants.
- La création de l'ensemble convexe pour l'étape suivante se fait par réunion des deux ensembles convexes traités. Avec p et q les nombres respectifs de points du premier et du second ensemble convexe, le nombre de calculs peut être estimé de l'ordre de $\frac{p}{2} + \frac{q}{2}$.

L'adaptation de la triangulation à une forme parcourt tous les triangles une fois. Pour chaque triangle, dans les cas de chevauchement avec la frontière de la forme, une recherche dichotomique du(des) point(s) d'intersection s'effectue entre le point externe A et le point interne B. Avec k le rapport entre la longueur du segment [A,B] et l'epsilon choisi pour la précision de la distance à la frontière, cette recherche est de l'ordre de $k \times \log(k)$.

Finalement, trop de paramètres indéterminés sont à considérer pour le calcul exact de la complexité comme le nombre moyen de voisins d'un point, le nombre de points moyens contenus dans l'enveloppe convexe d'un ensemble, etc.

III.2.2 Mesure de la complexité réelle

Cependant, une mesure du temps d'exécution en fonction de nombre de point entrés initialement peut indiquer comment se comporte le programme quand le nombre de points augmente (Illustration 18).

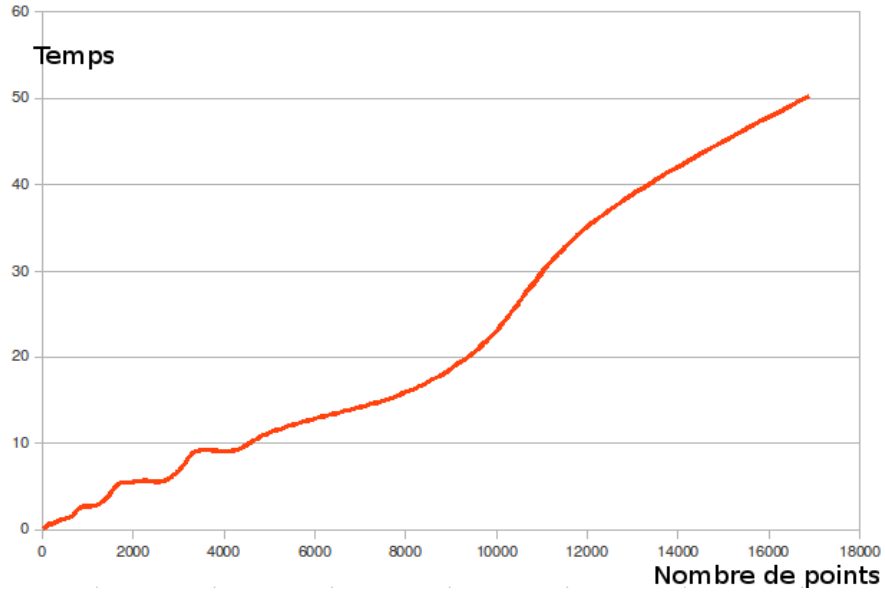

Illustration 18: Temps d'exécution en fonction du nombre de points

Cette mesure a été faite pour un maillage sur l'intervalle [0,100]×[0,100], avec une densité

$$d = \frac{(x-50)^2+(y-50)^2}{10000} \quad .$$

Les temps d'exécution variant considérablement entre deux mêmes lancements du programme et sur deux ordinateurs différents avec les mêmes paramètres, les valeurs obtenues ne sont pas significatives. Seule l'allure de la courbe est représentative du fonctionnement du programme.

On observe ici que la complexité réelle n'atteint pas la complexité théorique en n×log(n) prévue par l'algorithme de Delaunay et le comportement du programme avec un nombre de point supérieur à 100000 devient légèrement longue. Cependant, le résultat n'est pas insatisfaisant, compte-tenu de la difficulté à adapter les structures de données offertes par un langage de programmation tel que le C++ à un algorithme tel que celui-ci.

III.2.3 Problèmes rencontrés et améliorations à envisager

Certaines parties de notre programme peuvent être optimisées en adoptant une autre stratégie.

Par exemple, la fonction checkIntersect(A,B) qui parcourt tous les triangles existant pour supprimer ceux qui possèdent un côté qui croise le segment [A,B], est très gourmande en calculs. Cette fonction aurait pu être évitée en mettant à jour un voisinage de triangles pour chaque triangle. Seul le parcourt de la liste des voisins aurait été utile.

Le fait que les points soient générés sur une grille induit l'apparition de triangles plats. Ces triangles particuliers sont autant de sous-cas à traiter indépendamment des autres. Par exemple, le test d'appartenance au cercle circonscrit ne peut pas être appliqué comme habituellement puisque le cercle n'est pas défini. C'est une des raisons pour lesquelles la fonction de recherche de candidat lors

de la fusion arbore une structure si particulière. En effet, beaucoup de tests sont à effectuer pour tester l'alignement de sommets, et de calculs de produits scalaires dans ces cas précis. Cela a pour conséquence une complexification considérable du code et cela le rend difficile à interpréter. Une clarification de cette fonction est envisageable pour augmenter sa lisibilité et son efficacité.

La première implémentation du programme n'utilisait pas la librairie OFELI. L'incorporation de nos classes dans les classes pré-existantes de la librairie nous a posé quelques problèmes, notamment lors de la généralisation des types utilisés. En effet, les points étaient repérés par des coordonnées de type 'double'. Or la librairie OFELI utilise des templates pour une plus grande généralisation. Nous avons donc restructuré l'ensemble de notre code.

IV CONCLUSION

Tout d'abord, la triangulation générée respecte la condition de Delaunay. C'est lors de l'adaptation à la forme par la fonction distance que peuvent se former des triangles aplatis sur la frontière ne respectant plus la condition. La qualité du maillage est ainsi favorisée et les calculs ultérieurs pourront être effectués sur un bon support.

Le temps de calcul n'est cependant pas optimal. En effet, sur un ensemble de points dont le nombre dépasse les 100000, la triangulation se créé après un temps allant de 18 secondes à 30 secondes. Ce temps varie selon la densité entrée, la fonction distance et de l'ordinateur sur lequel le programme est lancé.

Le programme est portable, utilise certaines classes de la librairie OFELI, et son code est commenté pour une lisibilité maximale.

Le projet en lui-même fut très enrichissant. Il nous a permis d'étendre nos connaissances dans le domaine des éléments finis et d'étudier plus en détail les méthodes relatives à la résolution de problèmes physiques sur des surfaces en deux dimensions.

Le fait que la finalité du projet ait une utilité pour nos tuteurs et soit intégré à une partie de leurs travaux nous a motivé pour son élaboration.

Bien qu'imposant certains problèmes techniques, l'implémentation nous a été instructive car nous avons, grâce à elle, pu nous perfectionner dans le langage C++. La phase de recherche d'algorithmes, l'étude de complexité et l'utilisation de nouvelles structures de données ont suscité chez nous beaucoup d'intérêt.

BIBLIOGRAPHIE

[1] B. Delaunay, « Sur la sphère vide », *Izvestia Akademii Nauk SSSR*, *Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793-800, 1934

[2] Frey P. et George P., « Le maillage facile », Hermès Science Publications Paris, Paris, 2003

[3] Description de l'algorithme diviser pour régner appliqué à Delaunay, par Samuel Peterson : http://www.geom.uiuc.edu/~samuelp/del_project.html

[4] Description des propriétés de la triangulation de Delaunay : http://fr.wikipedia.org/wiki/Triangulation_de_Delaunay

[5] Site officiel d'OFELI : http://www.opheli.com