Sujets des projets (version 23/10/2019)
Initiation au C++ 4M016
Université Pierre et Marie Curie

F. Hecht, X. Claeys

Master I / session 1/S1, 2019/2020

Table des matières

Introduction

Pour tous le projet , il faut faire un rapport d'une quinzaine de pages informatique : LibreOffice, Latex, web , ...
Les Rapports et Programmes et un fichier explication sont à envoyer par mèl à `mailto: frederic.hecht@upmc.fr`. Tous ces fichiers sont à envoyer dans une archive `tar` compressé sans les compilés, executable. de nom : `votrenom-4m016.tar.gz` , Pour cela cree un dossier "votrenom-4m016", mettez les fichier dans le dossier, Puis dans le terminal et dans le dossier parent pour cree l'archive dans votre `home` faire

```
tar cvfz ~/votrenom-4m016.tar.gz votrenom-4m016
# verififier (afficher) le contenue  faire:
tar tvfz ~/votrenom-4m016.tar.gz
```

Ces projets sont a réaliser par binôme pour les étudiants de **M1**, le choix des projets est faite par les enseignants, une fois le binôme déclarer.

1 Recherche rapide d'un triangle contenant un point dans un maillage

Un maillage conforme formé d'un ensemble de n_T triangles $\mathscr{T}_h$ telle que l'intersection de 2 triangles distincts soit une arête commune, un sommet commun ou rien, où les l'ensemble des n_S sommets de $\mathscr{T}_h$ est noté $\mathscr{S}_h$. x

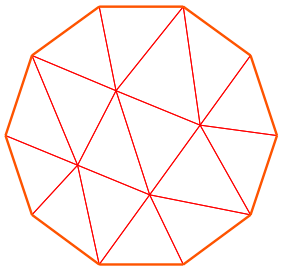


FIGURE 1 – exemple de maillage

Le but est d'écrire un algorithme de recherche d'un triangle K dans un maillage convexe $\mathscr{T}_h$ contenant un point (x,y) en $O(n_T^{1.5})$.

1.1 Le Projet 1 (promenade)

Algorithme 1

Partant du triangle K tournant dans sens trigonométrique, pour recherche le triangle contenant le point p, faire : Pour les 3 arêtes $(a_i,b_i), i = 0,1,2$ du triangle K, tournant dans le sens trigonométrique, calculer l'aire des 3 triangles (a_i,b_i,p). Si les trois aires sont positives alors $p \in K$ (stop), sinon nous choisirons comme nouveau triangle K l'un des triangles adjacents à l'une des arête associée à une aire négative (les ambiguïtés sont levées aléatoirement).

1. Construire une classe maillage formée d'un tableau de sommets et d'un tableau de triangles, qui lit le fichier maillage. donné (les fichier `.msh` de
   `https://www.ljll.math.upmc.fr/hecht/ftp/4M016/sujet-projet/` .
   Rappel : l'aire signé d'un triangle de sommets $ABC \in \mathbb{R}^2$) est $det(AB,AC)/2$ et le signe est positif si et seulement si le triangle tourne dans le sens direct.
2. Construire une méthode dans la classe maillage donnant le triangle K' adjacent d'un triangle K opposé à son sommet $i \in \{0,1,2\}$. La complexité de cette fonction doit être constante après bien sur une unique initialisation en $O(n_T)$ ou $O(n_T log_2(n_T))$.
   Pour cela vous pourrez utiliser le chapitre chaîne et chaînage ou les map de la STL.

3. Programmez l'algorithme 1 en partant d'un triangle quelconque (ajoutez une nouvelle méthode de la classe maillage).
4. afficher votre trajet avec le logiciel `gnuplot`.
5. Puis trouver pour chaque sommet d'un autre maillage donné du même domaine, le triangle le contenant. Afin de briser la complexité algorithmique, il faut utiliser l'assertion suivante : les sommets d'un triangle sont proche car les triangles sont petits.

2 Visualisation et optimisation de maillage

Le but de ce projet de visualiser une fonction f avec gnuplot du carré $]-1,1[^2$ à valeur dans $\mathbb{R}$ avec une précision de ε donnée.

Il faut donc construire un maillage telle que l'erreur L^2 sur chaque triangle du maillage soit inférieure à ε.

L'algorithme de raffinement de maillage est très simple : il consiste à couper les triangles d'erreur trop grand en deux parties égales par rapport à une des 3 arêtes en choisissant l'arête qui générera l'erreur minimal.

Donc l'algorithme est :

1. insérer les triangle avec une erreur trop grande la queue
2. tant que la queue n'est pas vide faire :
   (a) Prendre le triangle de la queue et découper le triangle en deux en choissisant la bonne arête, et ajouter les nouveaux triangles dans la queue si nécessaire.

L'erreur sur un triangle K sera définie par :

$$E_K = \int_K |f - \Pi_K(f)|^2$$

Où $\Pi_K(f)$ est la projection $L_2(K)$ de f sur l'espaces $P_1(K)$ des fonctions polynomes de degre 1 de K a valeur dans $\mathbb{R}$, c'est-à-dire :

$$\int_K \Pi_K(f) - f = 0; \qquad \int_K (\Pi_K(f) - f)x = 0 \qquad \int_K (\Pi_K(f) - f)y = 0$$

Et vous utiliserez des formules d'integration pour évaluer les intégrales qui sont définies dans `http://arxiv.org/abs/math/0501496` Several new quadrature formulas for polynomial integration in the triangle de Mark A. Taylor, Beth A. Wingate, Len P. Bos. les sources sont dans `http://arxiv.org/e-print/math/0501496v2`

Remarque sur les formules intégration sur un triangle $K = (A,B,C)$ soit F_K la transformation affine de $\hat{K} = ((0,0),(1,0),(0,1))$ dans K qui est défini par :

$$F_K : (\hat{x},\hat{y}) \mapsto (1-\hat{x}-\hat{y})A + \hat{x}B + \hat{y}C$$

Les N points $\{\hat{z}_1,\hat{z}_2,\ldots,\hat{z}_N\}$ défini sur le triangle $\hat{K}$ ry les poids associés $\{w_1,w_2,\ldots,w_N\}$ sont définis dans le fichier `coords.txt` pour une formule integration à l'ordre d. La formule de quadrature sur K est défini par

$$\int_K g \simeq \frac{|K|}{2} \sum_{j=1}^{N} w_j\, g(F_K(\hat{z}_i)),$$

Cette formule est exact pour les fonctions g qui sont des polynômes de degré d.

2.1 Projet 2

Programmer l'algorithme precédent pour visualiser avec gnuplot les fonctions

$$\begin{aligned} f_1(x,y) &= x^2+y^2 \\ f_2(x,y) &= x^2+y^3+tanh(5sin(2(y+x)) \end{aligned}$$

3 Optimisation d'un trajet sur un graphe

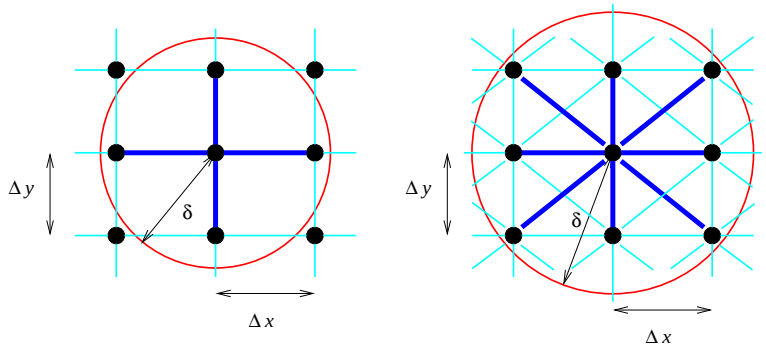
Le principe est d'optimiser le trajet d'un véhicule se déplaçant d'un point à un autre sur un terrain ayant une topographie quelconque. Cette optimisation devra être réalisée en minimisant deux quantités : la distance entre le point de départ et le point d'arrivée et la pente (positive et négative) du trajet. Il faudra donc trouver le chemin optimal pour que le véhicule ait le moins de distance à parcourir et qu'il ait le moins à monter et descendre possible.

3.1 Discrétisation et graphe

Soit $\Omega = [0,1] \times [0,1] \subset \mathbb{R}^2$ le domaine dans lequel le véhicule pourra évoluer. Soit la fonction $f : \Omega \mapsto \mathbb{R}$ associant à un point $X = (x,y)$ son altitude $f(X)$.

La première étape est la discrétisation de Ω. On se donne deux entiers N_x et N_y et on définit les points $X_{ij} = (x_i, y_j)$ avec $i = 1,...,N_x$ et $j = 1,...,N_y$. Si on note $\Delta x = (b-a)/(N_x - 1)$ et $\Delta y = (d-c)/(N_y - 1)$, on choisira N_x et N_y tels que $\Delta x \simeq \Delta y$. Les points $(X_{ij})_{i,j}$ sont les sommets du graphe associé à Ω (on notera $\mathscr{X}$ l'ensemble des sommets $(X_{ij})_{i,j}$ du graphe).

On définit ensuite l'ensemble $\mathscr{A}$ des arêtes du graphe. Pour cela, on se donne un réel $\delta > \max(\Delta x, \Delta y)$. Les arêtes du graphe sont définies par les segments $\sigma_{X_n X_m} = (X_n, X_m)_{1 \leq n,m \leq N_x N_y}$ vérifiant $0 < |X_n - X_m| \leq \delta$. Voici deux exemples de graphes obtenus suivant différentes valeurs de δ.



$\max(\Delta x, \Delta y) < \delta < \sqrt{\Delta x^2 + \Delta y^2}$ $\qquad$ $\sqrt{\Delta x^2 + \Delta y^2} \leq \delta < 2\min(\Delta x, \Delta y)$

La dernière étape pour la construction du graphe est d'affecter une valeur à chacune des arêtes. Soit l'arête $\sigma_{X_n X_m}$ reliant les points X_n et X_m. La valeur $c_{X_n X_m}$ (ou le *coût*) associée à cette arête sera une fonction qui dépendra à la fois de la distance $|X_n - X_m|$ ainsi que la valeur absolue de la pente $|f(X_m) - f(X_n)|/|X_n - X_m|$. Le coût $c_{X_n X_m}$ pour aller de X_m à X_n (ou bien de manière équivalente de X_n à X_m) par l'arête $\sigma_{X_n X_m}$ sera d'autant plus grand que la distance entre X_m et X_n sera importante et que la valeur absolue de la pente sera grande. Par exemple, on pourrait définir le coût ainsi :

$$c_{X_n X_m} = \alpha \frac{|X_n - X_m|}{\max(\Delta x, \Delta y)} + (1-\alpha)\frac{|f(X_m) - f(X_n)|}{|X_n - X_m|}, \quad 0 \leq \alpha \leq 1.$$

3.2 Détermination du plus court chemin

Soit $A \in \mathscr{X}$ le point de départ du véhicule et $B \in \mathscr{X}$ son point d'arrivée. On définit un *chemin* $\mathscr{C}$ allant de A à B comme une suite de sommets de $\mathscr{X}$ $\mathscr{C} = \{A = X_0, X_1, X_2, ..., X_{p-1}, X_p = B\}$ tels que pour tout $n = 0, ..., p-1$ on ait $\sigma_{X_n X_{n+1}} \in \mathscr{A}$, c'est-à-dire que tout segment (X_n, X_{n+1}) corresponde à une arête du graphe. On associe au chemin $\mathscr{C}$ son coût : $c_{\mathscr{C}} = \sum_{n=0}^{p} c_{X_n X_n + 1}$. Le problème d'optimisation du trajet du véhicule revient donc à trouver le chemin allant de A à B dont le coût est le plus faible. Ce chemin est appelé *le plus court chemin*. Pour le déterminer, on utilisera l'algorithme de Dijkstra.

Soit $\mathscr{R}$ l'ensemble des sommets restant à visiter et $\mathscr{P}$ l'ensemble des sommets déjà parcourus. On définit aussi $d(X)$ comme le coût du plus court chemin reliant X à A et $p(X)$ le prédécesseur de X dans le plus court chemin le reliant à A. L'algorithme de Dijkstra :

— Initialisation :
  $\mathscr{R} = \mathscr{X} \setminus \{A\}$,
  $\mathscr{P} = \{A\}$,
  $d(A) = 0$, $d(X) = c_{AX}$ si $\sigma_{AX} \in \mathscr{A}$ et $d(X) = +\infty$ sinon.
— Tant que $B \notin \mathscr{P}$, Faire :

- — Trouver X le sommet réalisant le minimum de $d(.)$ sur $\mathscr{R}$.
- — Ajouter X à l'ensemble $\mathscr{P}$.
- — Enlever X à l'ensemble $\mathscr{R}$.
- — Pour tout $Y \in \mathscr{R}$ tel que $\sigma_{XY} \in \mathscr{A}$, Faire :
  - — Si $d(X)+c_{XY} < d(Y)$ Alors $d(Y) = d(X)+c_{XY}$, $p(Y) = X$; Fin Si.
- — Pour obtenir le plus court chemin reliant A et B, il suffit alors de cheminer à l'envers : on regarde B, puis on sauvegarde son prédécesseur $p(B)$, puis le prédécesseur de son prédécesseur $p(p(B))$, ... jusqu'à arriver à A.

3.3 But du projet 3

Le but de ce projet est donc de réaliser un programme utilisant permettant d'afficher a l'aide de gnuplot le graphe et le chemin pour quelque cas bien choisi.

4 Problème du voyageur de commerce

Données : n villes numérotées $1,2,...,n$

Soit d_{ij} le coût (ici ce sera la distance) entre les villes i et j. On cherche le tour de coût minimum qui passe une fois et une seule par toutes les villes (cycle hamiltonien).

Pour les tests , on tirera aléatoirement suivant une loi uniforme les positions des villes dans un carré 100×100.

Si (x_i, y_j) sont les coordonnées de i et j

$d_i j = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ (distance euclidienne)

Algorithme à tester

4.1 Solution initiale

A1 Prendre les villes dans l'ordre $1,2,...,n$

A2 Partir d'une ville et choisir la plus proche puis la plus proche etc ... Si (1) et (k) sont les premières et dernières villes, refermer le chemin en ajoutant l'arc $(k,1)$.

A3 Partir de 3 villes ({1,2,3} par exemple). Supposons à une itération qu'on ait les villes $(i_1, i_2, ..., i_k)$ formant un cycle. Choisir une ville j non encore étudiée et la placer entre 2 villes $i_r, i_r + 1$ successives du circuit actuel $(1 \leq r < k)$ ou (i_k, i_1) de la manière la moins coûteuse.

A4 Construire l'arbre de poids minimum et parcourir 2 fois cet arbre puis enlever les villes parcourues 2 fois.

$1 \to 2 \to 3 \to 4 \to 3 \to 5 \to 3 \to 2 \to 6 \to 7 \to 6 \to 8 \to 6 \to 2 \to 1$

ce qui donne en fait [1 2 3 4 5 6 7 8 1]

4.2 Méthodes d'amélioration

B1 Echange de 2 arcs sur la solution actuelle (voir schma)

Soient 2 arcs $e = (i, i+1), f = (j, j+1)$ non consécutifs utilisés par la solution courante H (c.a.d.) les 4 indices $i, j, i+1, j+1$ sont distincts).

La nouvelle solution H_1 consiste à prendre j comme sommet suivant de i, de parcourir les sommets $(j-1, j-2, ...i+1)$ puis d'aller en $j+1$ et de compléter la tournée comme dans la solution H. Le coût $C(H_1)$ de H_1 se déduit de celui de $C(H)$ de H en ajoutant le coût des arcs $(i, j)(i+1, j+1)$ et en retirant le coût des arcs $(i, i+1)(j, j+1)$.

On dira que H_1 améliore H si $C(H_1) < C(H)$.

Si H_1 améliore H, remplacer H par H_1 et rechercher une nouvelle améloration Si aucune amélioration n'est possible à partir de H_1 pour tous les choix possibles de 2 arcs e et f de H, fournir H comme solution de l'algorithme.

B2 Enlever une ville i d'une solution actuelle et la replacer par un endroit plus intéressant entre $(j, j+1)$, c'est-à-dire les arcs $(i-1, i)$, $(i, i+1)$ et $j, j+1$ sont supprimés et les arcs $(i-1, i+1)$, (j, i) et $(i, j+1)$ sont créés. Pour faire cette transformation, il faut que deux arcs $(i, i+1)$, $(j, j+1)$ n'est aucun sommet en commun (c'est à dire que les 4 indices $i, i+1, j, j+1$ sont distincts) utilisés par la solution actuelle H.

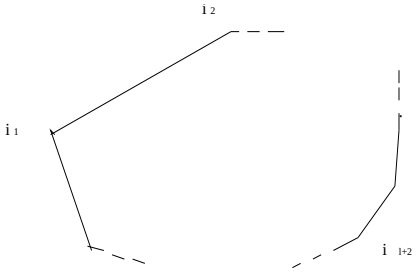


FIGURE 2 – figure A3

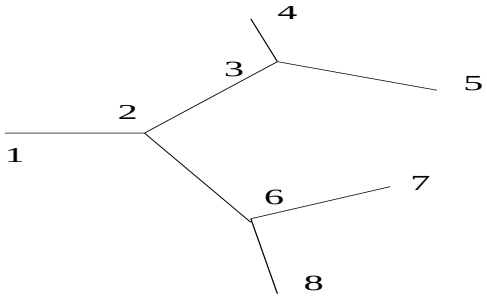


figure A4

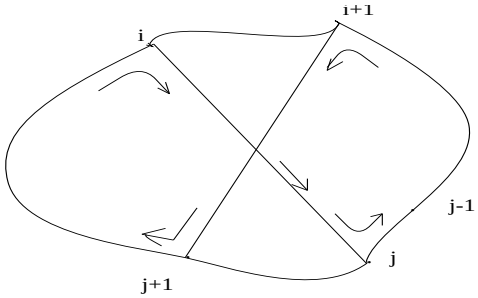


FIGURE 3 – Amélioration figure B1

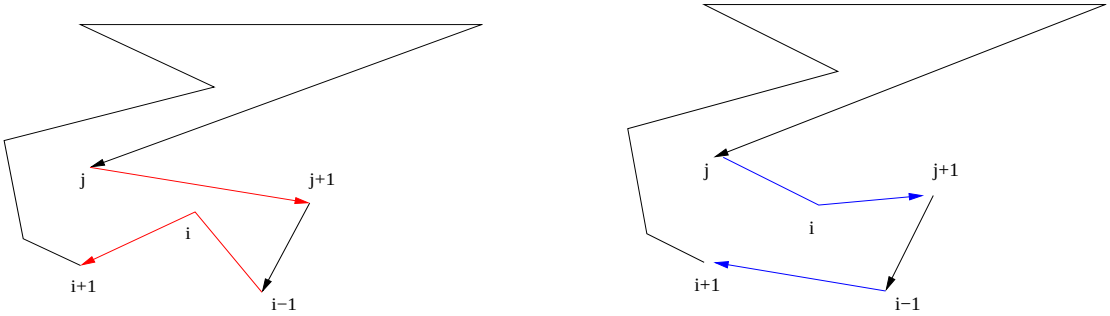


FIGURE 4 – Amélioration B2

4.3 Le projet 4

programmer les quatre initialisations et le deux méthodes d'amélioration, faite une analyse numérique de la complexité des algorithmes.

Résolution d'une EDP par la méthode des différences finis et visualisation

Le but est de résoudre numériquement l' équation suivante :
Trouver u une fonction régulière de $\Omega =]0,1[^2$ dans $\mathbb{R}$ tel que

$$-\Delta u = f, \qquad \text{dans} \quad \Omega \tag{1}$$

et telle que u soit nulle sur le bord du carré Ω et où l'opérateur Δ est défini par $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$.

Pour cela nous utiliserons une méthode aux différences finis. Nous allons calculé une approximation de la fonction u aux points $x_{n,m}$, noté $u_{n,m}$, où les points $x_{n,m}$ sont $(h_1 n, h_2 m)$ avec $h_1 = 1/N$ et $h_2 = 1/M$, pour $n = 0, ..., N$, et $m = 0, ..., M$.

Le schéma aux différences finis pour approcher Δ par Δ_d au point interne $x_{n,m}$ (c'est-à-dire n différent de 0 ou N et m différent de 0 ou M) , c'est à dire :

$$\forall (n,m) \in \{1,...,N-1\} \times m = \{1,...,M-1\} \quad \Delta_d u_{n,m} = \frac{u_{n-1,m} + u_{n+1,m} - 2u_{n,m}}{h_1^2} + \frac{u_{n,m-1} + u_{n,m-1} - 2u_{n,m}}{h_2^2} = f(x_{n,m}).$$

Remarque $u_{n,m}$ est connue et nulle sur les point du bord, c'est-à-dire $n \in \{0, N\}$ ou $m \in \{0, M\}$.

5 Projet 5/ UMFPACK

Le projet est de numéroté les inconnues du systèmes linéaire, pour construire la matrice creuse du système linéaire (où l'on ne stocke que les éléments non nuls, on pourra utilise un map de la STL comme stucture intermédiaire).

Il existe plusieurs bibliothèque sur le toile (WEB) pour résoudre de grand système linéaire creux. Vous utiliserez le logiciel UMFPACK, qu'il faut trouver, compiler, tester, valider.

Afin de valider programme problème, il faut trouver des solutions du problème (1), le plus simple de de construire une solution manufacture, c'est à dire choisir un fonction analytique ϕ nulle sur le bord de Ω, et de calculer $f = -\Delta\phi$.

Pour, finir, vous utiliserez gnuplot, pour visualiser la représentation 3D de votre solution.

Il faut tester votre programme, pour une famille de N, M allant 5 à 100, faire des courbes de temps calcul, etc ...

6 Projet 6/ SuperLU

Le projet est de numéroté les inconnues du systèmes linéaire, pour construire la matrice creuse du système linéaire (où l'on ne stocke que les éléments non nuls, on pourra utilise un map de la STL comme stucture intermédiaire).

Il existe plusieurs bibliothèque sur le toile (WEB) pour résoudre de grand système linéaire creux. Vous utiliserez le logiciel SuperLU, qu'il faut trouver, compiler, tester, valider.

Afin de valider programme problème, il faut trouver des solutions du problème (1), le plus simple de de construire une solution manufacture, c'est à dire choisir un fonction analytique ϕ nulle sur le bord de Ω, et de calculer $f = -\Delta\phi$.

Pour, finir, vous utiliserez gnuplot , pour visualiser la représentation 3D de votre solution.

Il faut tester votre programme, pour une famille de N, M allant 5 à 100, faire des courbes de temps calcul, etc ...

7 Découpe d'un cube en tétraèdre et visualisation

Le but du projet 7 est très simple, trouver et coder un algorithme qui génère les 74 découpes du cube en tétraèdres, et les visualiser ces découpes avec gnuplot.

Les découpes sont des partitions du cube telle que :

1. les sommets des tétraèdres sont des sommets du cube.
2. le maillage est conforme, c'est à dire que l'intersection de deux tétraèdres différents (supposés fermés) est soit
   (a) une face commune aux 2 tétraèdres
   (b) une arête commune aux 2 tétraèdres
   (c) un sommet commun aux 2 tétraèdres
   (d) l'ensemble vide.

Pour ce faire, il est conseiller de construire l'ensemble de tétraèdre possible, et de décrire les relations de compatibilité entre les faces.

Question bonus, faire la même chose en dimension 4 (dur).

8 projet 8 : les corrélations entre 2 produits achetés

Le but de ce projet est très simple : trouver et coder un algorithme qui donne la correlation entre 2 produits acheté par un même client.

C'est à dire que nous disposons d'un fichier achat des clients de n_l ligne dans ensemble de ligne I_l, formé des 2 mots

```
id_client ; id_produit
```

Donc après la numérotation consecutive des clients et des produits, nous disposons I_c, $n_c = \#I_c$, (resp. I_p, $n_p = \#I_p$) ensemble des numéros de clients (resp. produits) et des 2 tableaux c, p de taille n_l donnant pour chaque achat le numéro de client et le numéro du produit.

Soit C_i ensemble des clients achetant un produit i, c'est à dire que

$$C_i = \{j \in I_c / \exists k \in I_l, c[k] = j \text{ et } p[k] = i\} = \{c[k]; k \in p^{-1}(i)\}$$

où $p^{-1}(i)$ est l'image reciproque de la fonction $p : k \mapsto p[k]$. Soit C_{ij} ensemble des clients achetant deux produits i, j, c'est à dire que

$$C_{ij} = C_i \cap C_j.$$

On appellera la correlation en deux produits ij le cardinal de l'ensemble C_{ij}. le but est de trouver l'ensemble des 100 couples les plus corrélés, pour des données de taille type $n_l = 10^6$, $n_c = 5\,10^3$ et $n_p = 10^4$.

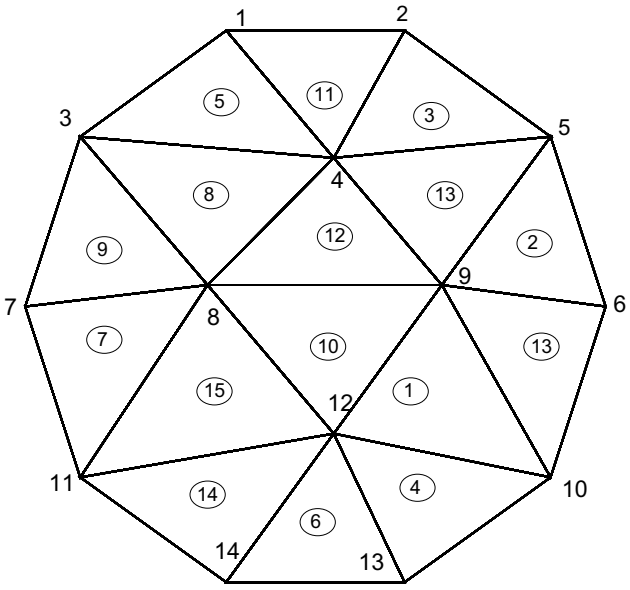
Le problème est : si n_p est trop grand (> 3000), il est déraisonnable de calculer tous les C_{ij}, il faudra donc utiliser une heuristique statistique pour résoudre ce problème.

Afin de test votre programme vous disposerez de trois fichiers tests : data-100.csv, data-1000.csv, data-10000.csv, data-500000.csv.

9 Annexe : Structure des fichiers maillage

We can read from Fig. 5 and "mesh_sample.msh" as in Table 1 where n_v is the number of vertices, n_t number of triangles and n_s the number of edges on boundary. For each vertex $q^i, i = 1, \cdots, n_v$, we denote by (q^i_x, q^i_y) the x-coordinate and y-coordinate.

Each triangle $T_k, k = 1, \cdots, 10$ have three vertices $q^{k_1}, q^{k_2}, q^{k_3}$ that are oriented in counterclockwise. The boundary consists of 10 lines $L_i, i = 1, \cdots, 10$ whose tips are q^{i_1}, q^{i_2}.



In the left figure, we have the following.

$n_v = 14, n_t = 16, n_s = 10$

$q^1 = (-0.309016994375, 0.951056516295)$
$\vdots \quad \vdots \quad \vdots$
$q^{14} = (-0.309016994375, -0.951056516295)$

The vertices of T_1 are q^9, q^{12}, q^{10}.
$\vdots \quad \vdots \quad \vdots$
The vertices of T_{16} are q^9, q^{10}, q^6.

The edge of 1st side L_1 are q^6, q^5.
$\vdots \quad \vdots \quad \vdots$
The edge of 10th side L_{10} are q^{10}, q^6.

FIGURE 5 – mesh by `buildmesh(C(10))`

Contents of file	Explanation
14 16 10	n_v n_t n_e
-0.309016994375 0.951056516295 1	q^1_x q^1_y boundary label=1
0.309016994375 0.951056516295 1	q^2_x q^2_y boundary label=1
$\ldots\ldots\vdots$	
-0.309016994375 -0.951056516295 1	q^{14}_x q^{14}_y boundary label=1
9 12 10 0	1_1 1_2 1_3 region label=0
5 9 6 0	2_1 2_2 2_3 region label=0
$\cdots$	
9 10 6 0	16_1 16_2 16_3 region label=0
6 5 1	1_1 1_2 boundary label=1
5 2 1	2_1 2_2 boundary label=1
$\cdots$	
10 6 1	10_1 10_2 boundary label=1

TABLE 1 – The structure of "mesh_sample.msh"