Projets MPE , Master 2, UPMC , C++

Frédéric hecht

22 jan. 2020 (version 5)

1 Introduction

Liste des sujets par binôme et étudiant:

pj2 BEN SALEM–KNAPP LOUISE, ANCOURT KEVIN

pj3 SQUARCIONI ALEXIS, TALATIZI ADRIEN

pj2 ASSOGBA KENNETH , DOTSE KOKOU

pj2 ROBERT DE BEAUCHAMP ZACHARIE, LEPREVOST ANTONIN

pj1 WANG XIAOYUAN, ABDULVAGABOV MAKHATCH

pj1 AIT DAHMANE ASSIA, DEHNI MANISSA

pj1 VALBON FLORENT, NANA KONGUEP CHARLY

pj1 CHELHAB KATIA, WANG-MA DAVID

pj3 HERVE PAUL, RUEDA ALEXANDRE

pj4 RICOLLEAU PAUL, CISSÉ ALY

pj2 BEN SLIMANE TINHINAN, HMIDY YASSINE

pj4 MARIN ALEXANDRE FOPA NOTEMI LÉONCE HMIDI YASSINE.

pj2 NDOYE MAMADOU, THIAM HAMIDOU

Le travail est faire en binôme. Il faut rendre les programmes commenter, un petit rapport 10 à 20 pages , dans une présentation vidéo projetée dans fichier archive à votre deux nom à m'envoyer pour le Jeudi 30 janvier 2020 8h00..

Les soutenances de 15mn auront lieu les 3,4,5 février après midi en monôme.

3/02 les apprenties de 17h00 à 18h30: .

4/02 le 14h00 à 17h00, n étudiants:

5/02 le 14h00 à 17h00, m étudiants:

Projet 1: la correlation entre 2 produits achetés

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

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

```
id_client ; id_produit
```

Donc après la numérotation des clients et des produits (voir remarque TD 7).), 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 produit 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 quatre fichiers tests: data-100.csv, data-1000.csv, data-10000.csv, data-500000.csv.

2 Projet 2:Recherche rapide d'un triangle contenant un point dans un maillage

Un maillage conforme formé d'un ensemble de n_T triangles $\mathcal{T}_h$ tels 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 $\mathcal{T}_h$ est noté $\mathcal{S}_h$.

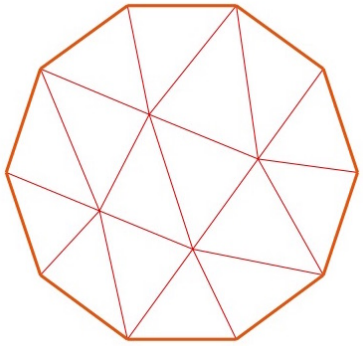

Figure 1: exemple de maillage

Le but est d'écrire un algorithme de recherche d'un triangle K dans un maillage convexe $\mathcal{T}_h$ contenant un point (x, y) en $O(log_2(n_T))$.

Rappel: l'aire signée d'un triangle (A, B, C) est définie par $det(\overrightarrow{AB}, \overrightarrow{AC})/2$

L'algorithme de recherche:

Algorithme 1

Partant du triangle K,
Pour les 3 arêtes $(a_i, b_i), i = 0, 1, 2$ du triangle K, tournant dans le sens trigonométrique, calculer l'aire signée 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 arêtes associées à une aire négative (les ambiguïtés sont levées aléatoirement).

2.1 Le Projet

1. Construire une classe maillage formée d'un tableau de sommets et d'un tableau de triangles, qui lit le fichier maillage (voir les classes du cours C++-4).

2. Construire une méthode dans la classe maillage donnant le triangle K' adjacent d'un triangle K par une arête qui est opposé à au 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.

3. Programmez l'algorithme 1 en partant d'un triangle quelconque (ajoutez une nouvelle méthode de la classe maillage).

4. Puis pour chaque sommet d'un autre maillage donné du même domaine, trouvez le triangle le contenant. Afin de briser la complexité algorithmique, il faut utiliser l'assertion suivante: les sommets d'un triangle sont proches car les triangles sont petits.

Vous pourrez utilise les script freefem++ cercle.edp pour générer des maillages comme suit

```
FreeFree++ cercle.edp --nn 100 -R 0.9
```

où ici on obtient le maillage d'une boule de rayon 0.9 avec 100 points sur un cercle .

3 Projet 3: Calcul de la fonction distance signé à une isovaleur zero d'une fonction

La fonction distance est la solution de viscosité d'une équation Eikonale : Trouver un champ scalar u dans le domaine, Ω of $\mathbb{R}^2$ tel que:

$$||\nabla u|| = F \text{ dans } \Omega, \quad (1)$$
$$u = g \text{ sur } \Gamma \quad (2)$$

où F est une fonction donnée suffisamment régulière positive, et Γ est un frontière interne (l'isovaleur), et g une fonction définie sur Γ. La fonction distance à Γ dans Ω est la solution pour $F = 1$ et $g = 0$.

Cette méthode de "Fast Marching" est un algorithme pour approcher la solution de viscosité de l'équation Eikonale. Nous voulons faire un calcul approximatif de cette solution sur un maillage triangulaire conforme T_h du domaine Ω dans V_h l'espace des fonctions affines par triangle du maillage T_h et globalement continues. Cette méthode est reliée aux principe de Huyghens qui correspond au développement de fronts d'ondes et à l'algorithme de Dijkstra qui calcule le chemin le plus court dans un graphe.

Notre problème discret est de calculer une approximation de la solution de viscosité de l'équation Eikonale dans l'espace V_h, c'est à dire en chaque sommet du maillage T_h. Où Γ_h est l'isovaleur 0 d'une fonction $f_h \in V_h$ donnée, et où la fonction g est nulle sur Γ_h.

L'on connait des solutions exactes de l'équation Eikonale dans un triangle K, et si Γ_h est un segment de droite $[A, B]$ qui peut être dégénéré en un point, l'équation Eikonale locale devient:

$$||\nabla u_K|| = 1 \text{ dans } K, \quad (3)$$
$$u_K = g \text{ sur } \Gamma \text{ avec } \Gamma = [A, B] \text{ ou } \Gamma = \{A\}. \quad (4)$$

et la solution u_K en P notée $u_K(P) = \mathcal{E}_K([\Gamma \cap K, g], P)$ est donnée par

$$\mathcal{E}_K([\Gamma \cap K, g], P) = \min_{Q \in [A,B]} g(Q) + ||QP|| \quad (5)$$

et si Γ est réduite au point A alors $Q = A$ et la solution est

$$\mathcal{E}_K([\Gamma \cap K, g], P) = g(A) + ||AP||. \quad (6)$$

3.1 L'algorithme

Les notations utilisées dans l'algorithme sont:
– $\mathcal{V}(K)$ l'ensemble des sommets des triangle K de T_h,
– $Adj(K, E)$ l'ensemble formée des éléments contenant E et différents de K, si E est sur le bord du domaine alors $Adj(K, E) = \emptyset$, sinon le cardinal de $Adj(K, E)$ est 1 et si $K' \in Adj(K, E)$ alors $K \cap K' = E$.
– ∂K le bord de K qui est un ensemble d'arêtes de K ,
– $Opp(K, E)$ le sommet du triangle K opposé à E et ou non dans l'arête E de K.
– $Iso(K, fh)$ le segment, point ou l'ensemble vide définissant l'isovaleur 0 de f_h dans K.
– $\mathcal{E}_K([\Gamma \cap K, g], Q)$ la solution locale de viscosité de l'équation Eikonale en Q dans K.

L' algorithme est :

Algorithme 2

– P_Q *la priority queue* $(v, (K, Q))$ *vide*
– $n = 0$
– $T_0 = \{K \in T_h; Iso(K, fh) \neq \emptyset\}$
$= \{K \in T_h; \quad \min_{Q \in \mathcal{V}(K)} f_h(Q) \leq 0 \leq \max_{Q \in \mathcal{V}(K)} f_h(Q)\}$
– $\forall Q \in \mathcal{V}(T_h); u(Q) = \infty$;
– $\forall K \in T_0$, $\forall Q \in \mathcal{V}(K)$; $u(Q) = \min(u(Q), \mathcal{E}_K([Iso(K, fh), 0], Q))$;
– $\forall K \in \mathcal{K}(T_0)$, $\forall E \in \partial K$, $\forall K' \in Adj(K, E)$ *tel que* $K' \notin T_0$,
mettre $(v, (K', Q))$ *dans* P_Q *avec* $v = \mathcal{E}_{K'}([E, u], Opp(K', E))$.

– *While* (P_Q *n'est pas vide*)
– *Prendre* $(v, (K, Q))$ *la plus petit valeur* P_Q *et la retirer de* P_Q,
– *if(* $K \notin T_n$*) then*
$T_{n+1} = T_n \cup \{K\}$
$u(Q) = v$
$n = n + 1$;
$\forall E \in \partial K$, $\forall K' \in Adj(K, E)$ *tel que* $K' \notin T_n$, $v = \mathcal{E}_{K'}([E, u], Opp(K', E))$,
ajouter (v, K', Q) *à* P_Q

3.2 Le Projet

Pour programmer cet algorithme, vous allez utilisez la class `Mesh2d` du cours (c.f. les classes du cours C++-3). Voilà le découpage du travail:

1. Ajouter dans la classe `Mesh2d` une méthode `int Adj(int K,int E,int &Ep);` qui retourne le numero de élément adjacent si il existe ou -1 sinon, et la variable `Ep` retournera le numéro de l'aréte dans l'element adjacent si il existe. La complexité de cette fonction doit être constante ou en $\log(nt)$ (où nt) est le nombre de triangles du maillage. Il faut appeler une fois une méthode `Initalisation()` pour la construction des données utilisées par `int Adj(int K,int E,int & Ep);` et la complexité de cette méthode de construction doit être au pire en $nt\ \log(nt)$.

2. Ecrire une function `int IsoK(const R2 *Pk,const double *fhk,R2 *A);` qui calcule $iso(K, fh)$ sur $K = (Pk[i])_{i=0}^2$ et où les 3 valeurs de f_h sont stockées dans le tableau fhk, cette fonction retourne le nombre de points définissant l'intersection (0,1 ou 2) et les points seront stockés dans le tableau A.

3. Ecrire une fonction `double EikonalK(int n, R2 *A, R *g,R2 P);` qui calcule pour $n = 1$ ou 2: $\mathcal{E}_K([\mathcal{C}(A), g_A], P)$, où $\mathcal{C}(A)$ est le convexcifié des points $\{A[i], i = 0, .., n-1\}$, c'est-à-dire

$$\mathcal{C}(A) = \{P = \sum_{i=0}^{n-1} p_i A[i],\ p_i \geq 0,\ \text{et}\ \sum_{i=0}^{n-1} p_i = 1\}; \qquad \text{et}\ g_A(P) = \sum_{i=0}^{n-1} p_i g[i].$$

   Attention seul le cas où $n = 2$ est non trivial, vous pourrez utiliser comme repère orthonormé $(O, \overrightarrow{e_1}, \overrightarrow{e_2})$ d'origine le projeté de P sur la droite passant par les points $A[0], A[1]$ et tel que le vecteur $\overrightarrow{A[0]A[1]}$, soit le vecteur $\overrightarrow{e_1}$, puis exprimer (5) dans ce repère, et conclure.

   c'est à dire précisement:

   Notons $p = ||\overrightarrow{PO}||_2, a = ||\overrightarrow{A[0]A[1]}||_2$, soit $Q = (1-\ell)A[0] + \ell A[0]$ le point de l'equation (5) sur la droite $A[0], A[1]$, notons ℓ_O le paramètre telle que $O = A[0] + \ell_O \overrightarrow{A[0]A[1]}$, comme $(\overrightarrow{PO}, \overrightarrow{A[0]A[1]}) = 0$ on a

$$\ell_O = -\frac{(\overrightarrow{A[0]P}, \overrightarrow{A[0]A[1]})}{a^2}.$$

   Remarquons que la fonction $g(\ell) = g[0] + d\ell$ avec $d = g[1] - g[0]$, et que l'on a

$$||\overrightarrow{PQ}|| = \sqrt{||\overrightarrow{PO}||^2 + ||\overrightarrow{OQ}||^2} = \sqrt{p^2 + (\ell - \ell_O)^2},$$

   donc en faisant le changement de variable $x = \ell - \ell_O$, on a $||\overrightarrow{PQ}|| = \sqrt{p^2 + a^2x^2}$ et $g(x) = g[0] - d\ell_O + dx$, il faut donc minimise J défini par:

$$Q = O + x\,\overrightarrow{A[0]A[1]}, \qquad J(Q) = J(x) = g[0] - d\,\ell_O + d\,x + \sqrt{p^2 + a^2x^2}.$$

   Le minimum est atteint quand $J'(x) = 0$, ou en $A[0]$ ou $A[1]$. On a $J'(x) = d + xa^2/\sqrt{p^2 + a^2x^2} = 0$ donc

$$d\sqrt{p^2 + a^2x^2} = -a^2x$$

   ce qui implique que

$$(d^2p^2 + d^2a^2x^2) = a^4x^2$$

   d'où $d^2p^2 = a^2(a^2 - d^2)x^2$ donc si $|d| > a$ il n'y a pas de solution, et le minimun est donc atteint en $Q \in \{A[0], A[1]\}$, sinon $x = -dp/(a\sqrt{a^2 - d^2})$ et $\ell = x + \ell_O$ et si $\ell \in [0, 1]$ alors $Q = (1-\ell)A[0] + \ell A[0]$ sinon on a $Q \in \{A[0], A[1]\}$.

4. Progammer l'algorithme.

5. A ce niveau, vous avez la fonction distance non signée, pour la signer, il suffit de coder la phase suivante: le signe de distance sera $-$ le signe de la fonction f_h donnée.

6. Valider votre algorithme en choisissant des cas où vous pouvez calculer cette fonction distance signe.

Vous pourrez utilise les script freefem++ distance.edp ou distance2.edp pour générer des exemples et valider votre programe (cf. fig 2)

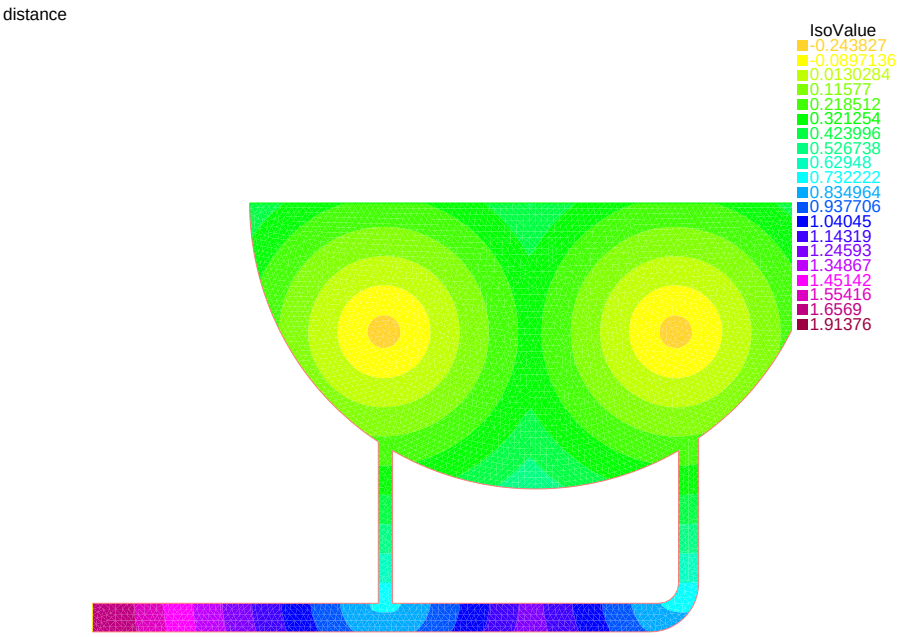

Figure 2: Isovalue de la fonction distance à 2 cercles dans un maillage complique

4 Problème de Plateau (surface minimale)

On s'intéresse au calcul du problème de surface minimale.

Le problème de surface minimale est: soit Γ une courbe fermée de $\mathbb{R}^3$, le but est de trouver la surface S d'aire minimale de bord la courbe Γ.

Ce problème n'est pas simple théoriquement, mais si on le regarde du point de vue discret, il est relativement simple.

Donc notre problème est de trouver

$$\underset{S_h \in \Sigma_h}{\texttt{arg min}}\, J = \sum_{K \in S_h} |K| \tag{7}$$

où Σ_h est un l'ensemble des surfaces discrètes admissibles, approchant Γ donné.

Dans le problème de Plateau on considère les surfaces sont le graphe d'une fonction. C'est-à-dire que l' on prendra comme ensemble de surface Σ_h, l'ensemble des maillages générés à partir d'un maillage 2d T_h du domain Ω de bord Γ_h et où l' on ajoute une troisième coordonnée z_s à chaque sommet du maillage T_h.

Nous noterons

1. $|K|$ l'aire d'un triangle K de $\mathbb{R}^3$,
2. $\partial_{z_i}|K|$ la dérivé partielle de l'aire par rapport à la cote (z_i) d'un sommet i du triangle K ,
3. $S = S_0 \cup S_\Gamma$ l'ensemble des sommets, avec S_0 l'ensemble des sommets interne et S_Γ l'ensemble des sommets frontières,

L'equation d'Euler Lagrange (7) est trouver les $z_i / i \in I_0$ tel que

$$\forall i \in I_0, \quad \sum_{K \in T_h} \partial_{z_i}|K| = 0 \tag{8}$$

Ce problème est non linéaire, vous utilisez un algorithme de point fixe, pour cela il faut réécrire la formulation.

La partie $\mathbb{R}^2$ des sommets est constante et noterons u_i la troisième composantes des sommets A^i nous noterons la normal $N^K = (A^j - A^i) \wedge (A^k - A^i)$ au triangle de A^i, A^j, Aj^k , on a $|K| = ||N^K||/2$, on allons

étudier la dependence en u de N^K , c'est à dire que nous noterons $N^K(u)$ le vecteur normal du triangle K qui a comme troisième coordonné $z_l = u_\ell$ pour $\ell \in \{i, j, k\}$. Notre problème de minimisation est donc:

$$\arg\min_u J = \sum_{K \in S_h} ||N^K(u)|| \tag{9}$$

Il est facile de voir que cette dépendance $N^K(u)$ est affine en u donc la différentielle $DN^K(u)$ est donné par $v \mapsto N^K(v) - N^K(0)$. De plus, DN est dans le plan et pour $K = (P^0, P^1, P2)$ dans le plan, nous avons

$$DN^K(u) = \sum_{i=0}^{2} u_{i_K} (P^{i+2} - P^{i+1})^\perp$$

où les operations dans $0, 1, 2$ son modulo 3 et où i_K est le numero du sommet global du sommet i de K. On a aussi

$$N^K(0) = [0, 0, det(P^1 - P^0, P^2 - P^0)]$$

Et comme $||A|| = \sqrt{A.A}$ où l'operateur . est le produit scalaire de $\mathbb{R}^3$, la différentielle de $||N^K(u)||$ est $D||N^K(u)|| : v \mapsto (DN^K(v).N^K(u))/||N^K(u)||$ et l'equation d'Euler Lagrange peut ce réécrire comme:

Trouver $u \in \mathbb{R}^I$ telle que $u_i = g_i$ si $i \in I_\Gamma$ et telle que

$$\forall v \in \mathbb{R}^{I_0}, \quad \sum_K (DN^K(v).N^K(u))/||N^K(u)|| = 0$$

Comme par construction on a $N^K(u) = DN^K(u) + N^K(0)$, on a

$$\forall v \in \mathbb{R}^{I_0}, \quad \sum_K (DN^K(v).DN^K(u))/||N^K(u)|| = -\sum_K (DN^K(v).DN^K(0))/||N^K(u)||.$$

Bien sur ce problème est non linéaire nous utiliserons un méthode de points fixe pour la résoudre
Nous allons basé sur le calcule du problem linéaire (Pa) suivant:

Trouver $u \in \mathbb{R}^I$ tel que $\forall i \in I_\Gamma, u_i = g_i$ (les g_i sont des données) , tel que le vecteur $w \in \mathbb{R}^I$ soit aussi donné, tel que

$$\forall v \in \mathbb{R}^{I_0}, \quad \sum_K (DN^K(v).DN^K(u))/||N^K(w)|| = -\sum_K (DN^K(v).DN^K(0))/||N^K(w)|| \tag{10}$$

Il est trivial de remarquer que ce problème est symétrique positif, nous utiliserons donc une méthode de gradient conjugué pour le résoudre, vous pourrez utilisez les classes du cours (c.f. les classes du cours C++-4).

4.1 Le travail

Q0) Petite recherche bibliographique afin de pouvoir valider votre projet.

Q1) Calculer u^0 comme solution P_a avec $w = 0$,vous utiliser l'algorithme du gradient conjugue pour résoudre le problème (10).

Validez votre programme sur un cas en choisissant $\#I_\Gamma = 1$ et $g_i = 1$.

Q3) Ecrire l'algorithme du point fixe en faisant la boucle suivante $w = u^i$ et calculer u^{i+1} solution P_a.

Q4) Valider votre programme sur la caténoide (choisir un domaine correct).

Q5) Tester sur le domaine $\Omega =]-\pi, \pi[^2$ maillé en 20×20 régulièrement avec FreeFem++, l'interpolée de la fonction $(x, y) \mapsto cos(x)cos(y)$ sur votre maillage.

Q6) Utiliser votre programme sur d'autre cas personnel, et commenter.

Il faut faire un rapport électronique de 5 à 10 pages (latex, LibreOffice, Page Web , ...), pour les visualisations des résultats vous utilisez gnuplot. Le rapport et les programmes sont à rendre pour