# Méthode multipole rapide pour un nuage de points

Gaétan Facchinetti
5 décembre 2016

*Université Paris-Saclay, Ecole Normale Supérieure de Cachan,*
*Ecole Nationale Supérieure des Techniques Avancées*

## Question 1

Nous avons créer une fonction permettant de renvoyer, pour une densité de point par longueur d'ondre $n_\lambda$ et une fréquence $f$ donnée, un tableau de coordonnées de l'ensemble des points du nuages ainsi que le nombre de $N$ points. Dans notre code ce nombre de points se calcule en fonction des paramètres par la formule,

$$N = 4s_a(s_b - 1) + 2(s_b - 2)^2 \tag{1}$$

Avec $s_a = \mathbb{E}(fn_\lambda L/c) + 1$ et $s_b = \mathbb{E}(fn_\lambda l/c) + 1$, où $L = 1$ (m) et $l = 0.5$ (m) sont les dimensions de la boite et $c$ la célérité de l'onde dans le milieu considéré Nous avons alors pu représenter en Fig. 1 les points de discrétisation.

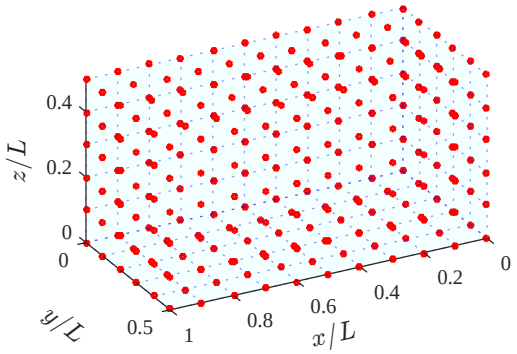


FIGURE 1 – Points de discrétisation suivant les trois coordonnées spatiales (rouge). $N = 252$, $n_\lambda = 10$, $f = c/L$. Pour faciliter la lecture les plans $x = 0$, $y = 0$ et $z = 0$ ont été représentés en cyan.

## Question 2

Notons, pour $i \in [\![1, N]\!]$, $\mathbf{x}_i$ le vecteur position du point du nuage indicé $i$. Introduisons alors la matrice de la fonction de Green G que nous definissons par :

$$\forall (i,j) \in [\![1,N]\!]^2 \quad G_{i,j} = \begin{cases} \frac{e^{ik|\mathbf{x}_i - \mathbf{x}_j|}}{|\mathbf{x}_i - \mathbf{x}_j|} & \text{si } i \neq j \\ 0 & \text{si } i = j \end{cases} \tag{2}$$

Nous notons $\tau_a$ le temps d'assemblage de cette matrice. Soit maintenant un vecteur $\boldsymbol{\rho}$ quelconque de $\mathbb{R}^N$. Nous notons $\tau_c$ le temps de calcul du produit matrice vecteur $\mathbf{V} = G\boldsymbol{\rho}$.

Nous pouvons remarquer qu'à partir de $N \sim 10000$ l'assemblade de la matrice est trop gourmand en mémoire et cela rend l'execution sous Matlab impossible. Nous avons donc fait varier $n_\lambda$ à $f$ fixé pour avoir, d'après Eq. (1), une valeur de $N$ maximale de 9002. Puis nous avons représenté en Fig. 2 et Fig. 3 l'évolution de $\tau_a$ et $\tau_c$ en fonction de $N$.

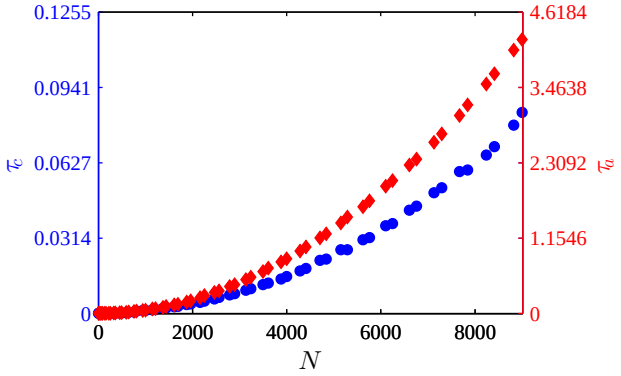


FIGURE 2 – Temps d'assemblage de $G$, $\tau_a$ (losanges rouge) et temps de calcul du produit matrice vecteurs $\tau_c$ (ronds bleu) en fonction de $N$. Les deux courbes ont une allure parabolique mais nous pouvons remarquer que, les echelles étant différentes, le temps d'assemblage est plus long

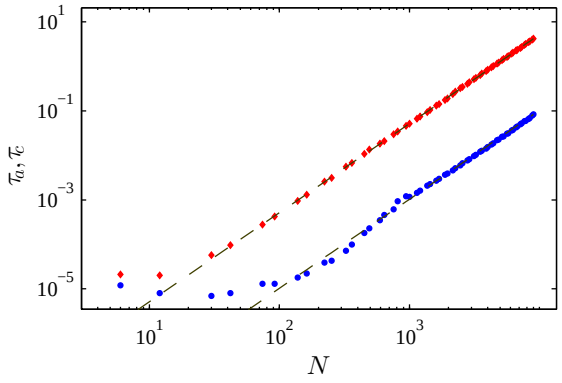


FIGURE 3 – Temps d'assemblage de $G$, $\tau_a$ (losanges rouge) et temps de calcul du produit matrice vecteurs $\tau_c$ (ronds bleu) en fonction de $N$. Nous pouvons remarquer ici qu'asymptotiquement $\log(\tau_{a/c}) \simeq 2\log(N) + K_{a/c}$ représentées en pointillé vert avec $K_{a/c}$ constante.

Comme il l'est montré en Fig. 3, l'évolution du logarithme de $\tau_a$ avec $N$ tend asymptotiquement vers une droite de pente 2. Il est en cd même pour le logarithme de $\tau_c$ avec $N$. Ceci confirme l'évolution en $O(N^2)$ du temps d'assemblage et de produit matrice vecteur.

# Question 3

Nous avons calculé la quadrature de Gauss-Legendre à $L_q$ points en diagonsalisant la matrice tridiagonale définie dans l'énoncé. La methode utiliséé pour obtenir les points de quadrature est la méthode de Golub-Welsh [1]. Avec les notations du TP et $L_q$ le nombre de points de quadrature (à ne pas confondre avec $L$ la longueur du pavé) nous calculons, pour P polynôme tel que deg$P \leq 2L_q - 1$,

$$\int_{-1}^{1} P(t)dt = \sum_{i=1}^{L_q} \omega_i P(\lambda_i) \tag{3}$$

Ceci est équivalent, par un changement de variable, à

$$\int_{0}^{\pi} P(\cos(t))\sin(t)dt = \sum_{i=1}^{L_q} \omega_i P(\cos(\theta_i)) \tag{4}$$

Nous avons testé cette quadrature avec $L_q = 3$ en comparaison de celle à 3 points dévleoppée lors du premier TP. Nous écrivons $I_{GL}$ le résultat par la quadrature de Gauss Legendre, $I_1$ le résultat pour la quadrature à trois points du premier TP, $I_M$ le résultat de la qudrature effectuée par Matlab et $I_v$ la valeur vraie pour l'integration de la fonction polynomiale $P$.

| $P$ | $I_{GL}$ | $I_1$ | $I_M$ | $I_v$ |
|---|---|---|---|---|
| $x \mapsto x$ | 0.00 | 0.00 | 0.00 | 0 |
| $x \mapsto x^2$ | 0.667 | 0.667 | 0.667 | 2/3 |
| $x \mapsto x^4$ | 0.400 | 0.667 | 0.400 | 2/5 |

Table 1 – Résultat des quadratures numériques

Comme attendu nous observons que pour la quadrature de Gauss-Legendre donne les même resultats pour des polynômes de degré inférieur ou égal à 3 que la quadrature du premier TP. En revanche, comme nous pouvions nous y attendre cette nouvelle quadrature nous permet d'avoir une solution correcte pour des polynomes de degré 4 et 5, puisque la formule est bien exacte jusqu'au degé $2L_q - 1$, ce que nous n'avions pas avant.

# Question 4

Notons, pour $l \in \mathbb{N}$ et $m \in [\![-l, l]\!]$ :

$$C_{l,m} = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} \tag{5}$$

Nous avons, par définition,

$$Y_{lm}(\theta,\phi) = C_{l,m} P_l^m(\cos(\theta))\, e^{im\phi} \tag{6}$$

En notant $I_{Y_{lm}}$ l'intégrale de $Y_{lm}$ sur la sphère unité (nous noterons $I_{GL,Y_{lm}}$ le résultat numérique de la quadrature),

$$I_{Y_{lm}} = C_{l,m}\int_0^{\pi} P_l^m(\cos(\theta))\sin(\theta)d\theta \int_0^{2\pi} e^{im\phi}d\phi \tag{7}$$

Ainsi si $m \neq 0$ l'integrale sur $\phi$ donne directement $I = 0$. De plus, d'après d'autres propriétés des polynomes de Legendre, la quadrature totale doit alors satisfaire les égalités suivantes :

$$\sum_{i,j} \omega_i\omega_j Y_{lm}(\theta_j,\phi_i) = \begin{cases} 0 & \text{si } m \neq 0 \text{ ou } l \neq 0 \\ \sqrt{4\pi} & \text{si } m = 0 \text{ et } l = 0 \end{cases} \tag{8}$$

En particulier, la quadrature sur $\phi$ doit vérifier :

$$\sum_i \omega_i e^{im\phi_i} = \begin{cases} 0 & \text{si } m \neq 0 \\ 2\pi & \text{si } m = 0 \end{cases} \tag{9}$$

Nous avons réalisé une quadraturre à $L_q + 1$ points en $\theta$ et $2L_q + 1$ points en $\phi$. Nous notons une intégration exacte, à erreur d'arrondi machine de l'ordre de $10^{-8}$, pour toute les harmoniques sphériques avec $l \leq 2L_q + 1$. Cependant pour $l$ supérieur nous pouvons commencer à remarquer des écarts puisque la quadrature nous donne une valeur numérique $I_{GL-Y_{lm}} = 2.576$ pour $Y_{l=9,m=9}$ en utilisant $L_q = 4$.

Nous aurions pu utiliser toute autre quadrature de Gauss, de Simpsons, etc. L'avantage de cette quadrature ci et d'intégrer exactement les harmoniques sphériques et de donner donc de très bon résultats pour nos expressions qui peuvent se décomposer avec une très bonne approximation sur une base constituée d'un nombre limité d'entre elles. Le deuxième avantage qui apparait est qu'il faut un nombre de points $(L_q+1)(2L_q+1)$ pour pouvoir intégrer exactement des harmoniques sphériques jusqu'à l'ordre $l = 2L_q + 1$ et donc ainsi utiliser un nombre de points restreint.

# Question 5

## 1. Mise en pratique

Nous souhaitons calculer la décomposition en onde plane de la fonction de green G. Posons $\mathbf{x} - \mathbf{y} = (\mathbf{y}_0 - \mathbf{y}) + (\mathbf{x}_0 - \mathbf{y}_0) - (\mathbf{x}_0 - \mathbf{x}) = \mathbf{r} + \mathbf{r}_0$. Nous utilisons

$$G(\mathbf{x},\mathbf{y}) = \frac{\exp(ik|\mathbf{x}-\mathbf{y}|)}{|\mathbf{x}-\mathbf{y}|} \simeq \int_{S^2} e^{ik\hat{\mathbf{s}}\mathbf{r}}\mathcal{G}_L d\hat{\mathbf{s}} \tag{10}$$

$$\mathcal{G}_{L_G} = \frac{ik}{4\pi}\sum_{p=0}^{L_G}(2p+1)i^p h_p^{(1)}(k|\mathbf{r}_0|)P_p(\cos(\hat{\mathbf{s}},\mathbf{r}_0)) \tag{11}$$

$$\cos(\hat{\mathbf{s}},\mathbf{r}_0) = \frac{1}{|\mathbf{r}_0|} < \begin{pmatrix} \sin(\theta)\cos(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{pmatrix}, \mathbf{r}_0 > \tag{12}$$

1. *Calculation of Gauss Quadrature Rules*, G.H. Golub and J.H. Welsh , Math. Comp. 23 (1969), 221-230 , (Apr., 1969)

L'intégrale sur la sphére unité est alors calculée par la quadrature déterminée dans la question précédente. En effet, la largeur de bande de l'équation a intégrer étant de $2L_G$ comme nous intégrons quasiment exactement avec une quadrature de $L_q$ en $\theta$ et $2L_q+1$ et $\phi$ les harmoniques sphériques $Y_{lm}$ avec $l \neq 2L_q+1$ nous prenons ici une quadrature de $L_G+1$ points en $\theta$ et $2L+1$ points en $\phi$.

De plus, il faut être prudent et utiliser ici les fonctions de hankel sphériques et non celles définies sous Matlab avec la denomination *besselh*. Ces fonctions sont définies par, pour $(z,p) \in \mathbb{R}_+^* \times \mathbb{N}$,

$$h_p^{(1)}(z) = e^{iz} \sum_{k=0}^{p} \frac{(p+k)!}{2^k (p-k)!} \frac{1}{z^{k+1}} \tag{13}$$

Pour être plus rapide dans le code ces fonctions sont en réalité calculées à l'aide de la formule de récurrence suivante. Ceci est pratique puisque nous avons besoin de toutes les fonctions de hankel jusuqu'à l'ordre $L$.

$$h_{p+1}^{(1)}(z) = \frac{2p+1}{z} h_p^{(1)}(z) - h_{p-1}^{(1)}(z) \tag{14}$$

De même, par souci de rapidité, nous n'utilisons pas non plus la fonction *legendreP* intégrée au logiciel pour récupérer la valeur des polynômes de Legendre. Nous procédons aussi avec une formule de récurrence, ceci nous ayant permis d'en récupéréer les valeurs approximativement 10 fois plus vite qu'avec la fonction de Matlab. Pour $(x,p) \in \mathbb{R} \times \mathbb{N}$,

$$P_{p+1}(x) = \frac{2p+1}{p+1} x P_p(x) - \frac{p}{p+1} P_{p-1}(x) \tag{15}$$

Enfin nous noterons aussi que pour réaliser des calculs rapides en vectorialisant au maximum les opérations effectuées nous avons beacoup utilisé la fonction *bsxfun(@times, ..., ...)* du logiciel.

## 2. Un exemple de cas test

Pour tester nos codes nous avons utilisé le cas test $\mathbf{r} = 0$ et $\mathbf{r}_0 = \mathbf{x} - \mathbf{y}$. En effet dans ce cas,

$$G(\mathbf{x},\mathbf{y}) = \frac{ik}{4\pi} \sum_{p=0}^{L} (2p+1) i^p h_p^{(1)}(k|\mathbf{r}_0|) \times \int_{S^2} P_p(\cos(\hat{\mathbf{s}}, \mathbf{r}_0)) d\hat{\mathbf{s}} \tag{16}$$

Cependant pour $p \in [\![0, L]\!]$ comme notre quadrature intègre exactement les polynomes de degrés inférieur ou égal à $2L+1$ et d'après l'ortogonalité des polynômes de Legendre, il vient,

$$G(\mathbf{x},\mathbf{y}) = ikh_0^{(1)}(k|\mathbf{r}_0|) \tag{17}$$

Nous avons un résultat analytique très pratique pour vérifier le code. Ceci nous permet en outre de confirmer le choix des fonctions de Hankel et le bon fonctionnement de la quadrature sur les polynômes de Legendre.

## 3. Resultats

Pour nous assurer du bon fonctionnement de notre code nous avons calculé différentes valeurs de la fonction de green en différents points en faisant varier $k$ et $L_q$.

| $k$ | $L_q$ | $\mathbf{x}/L$ | $\mathbf{y}/L$ | $\mathbf{x}_0/L$ | $\mathbf{y}_0/L$ | $G_{ex}(\mathbf{x},\mathbf{y})$ | $G_{\sim}(\mathbf{x},\mathbf{y})$ | $\varepsilon$ (%) |
|---|---|---|---|---|---|---|---|---|
| $4\pi$ | 1 | (0 0 0) | (1 0.5 0.5) | (0 0 0) | (1.5 0.5 0.5) | 0 | 0 | 0 |
| $4\pi$ | 5 | (0 0 0) | (1 0.5 0.5) | (0 0 0) | (1.5 0.5 0.5) | 0 | 0 | 0 |
| $4\pi$ | 5 | (0.1 0.1 0.1) | (1.1 0.4 0.6) | (0 0 0) | (1.5 0.5 0.5) | 0 | 0 | 0 |
| $4\pi$ | 10 | (0.1 0.1 0.1) | (1.1 0.4 0.6) | (0 0 0) | (1.5 0.5 0.5) | 0 | 0 | 0 |
| $4\pi$ | 5 | (0.1 0.1 0.1) | (0.3 0.3 0.3) | (0 0 0) | (0.3 0.3 0.3) | 0 | 0 | 0 |

Table 2 – Résultat de la décomposition en ondes planes avec L longueur max

Nous pouvons commencer par remarquer que dans le cas $\mathbf{x} = \mathbf{x}_0$ et $\mathbf{y} = \mathbf{y}_0$ nous obtenons un resultat exact même pour $L_q = 1$.

# Question 6

Nous avons réalisé le partitionnement, correspondant à un niveau de ce qu'aurait été l'octree dans un algorithme multi-niveau dans un fonction nous renvoyant une structure nommée **partition**. Les attributs essentiels de cette structure Matlab sont trois tableaux :

— du numéro des noeuds dans chaque boite.
— du nombre de noeuds dans chaque boite.

— des coordonnées du centre de chaque boite.

Même si l'on ne prend pas en compte cette étape dans le temps de calcul du produit matrice vecteur il est bon d'essayer de ne pas la rendre trop longue. Pour ce faire nous n'avons réalis,é pour la création des boites de la partition et l'extraction des noeuds qui y appartiennent, que des boucles sur au maximum le nombre de boite et jamais sur le nombre de noeuds total qui est bien plus important. De plus pour ne pas surcharger la mémoire et comme beacoup des tableaux de **partition** sont creux nous avons utilisé une définition en tant que *sparse* de ces matrices.

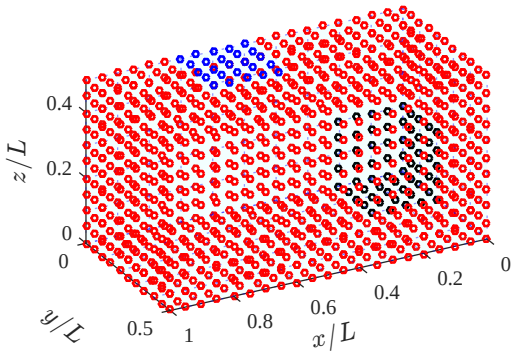


FIGURE 4 – Représentation de deux paquets deux points appartenants à deux boites de taille $d$ différentes en sortie de notre algorithme de partition (représentés en bleu et en noir). Celle en bleu contient moins de points car elle ne contient que des éléments du haut de la boite alors que celui en noir regroupe des points sur trois faces. $f = 2c/L$, $n_\lambda = 10$, $d = 0.3\lambda$.

Après avoir réalisé le partitionnement nous nous sommes attaqués à l'écriture totale du code de résolution FMM à un noveau en réutilisant toutes les fonctions déjà écrites. Nous pouvons résumé ici la structure de l'algorithme final de la sorte :

1: Initialisation des données du problèmes
2: Maillage du domaine
3: Initialisation des points de quadrature
4: Creation des partitions
5: Calcul du produit pour les noeuds non voisins
6: Calcul des contributions des voisins

Le coeur du problème se trouve alors dans le calcul du produit matrice vecteur sur les noeuds non voisins. Pour cela nous avons réalisé ...

## Question 7

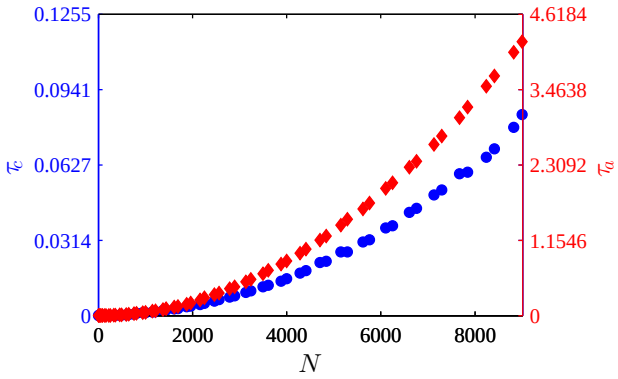


FIGURE 5 – Evolution du temps de calcul du produit matrice vecteur par la méthode classique $\tau_{class}$ (rouge) et par la méthode FMM $\tau_{FMM}$ (bleu) avec $N$.

Nous avons pu aussi regarder les erreurs totales obtenues en fonction de $L_q$ utilisé pour une certaine configuration donnée.

## Question 8

Nous pouvons remarquer que bien que malgrès une complexité plus avantageuse et nos effort pour le rendre le plus rapide possible ce nouveau algorithme met plus de temps à s'executer que le produit réalisé classiquement pour le nombre de points (qui reste relativement faible) que nous aons pu tester. En effet nous avons pour $N = ...$, un temps $\tau_{FMM} = ...$s et $\tau_{class} = ...$s. Nous pouvons dire, en comparant les deux courbes de la quesiton précédente qu'il faudrait atteindre $N = ...$ pour que l'utilisation de la FMM mono-niveau soit, dans ce cas, utile.