

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 créé une fonction permettant de renvoyer, pour une densité de point par longueur d'onde n_λ et une fréquence f donnée, un tableau de coordonnées de l'ensemble des points du nuage ainsi que le nombre de N points. Dans notre code ce nombre de points se calcule en fonction des paramètres par la formule,

$$N = 4s_a(s_b - 1) + 2(s_b - 2)^2 \quad (1)$$

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 boîte et c la célérité de l'onde dans le milieu considéré. Nous avons alors pu représenter en Fig. 1 les points de discrétisation.

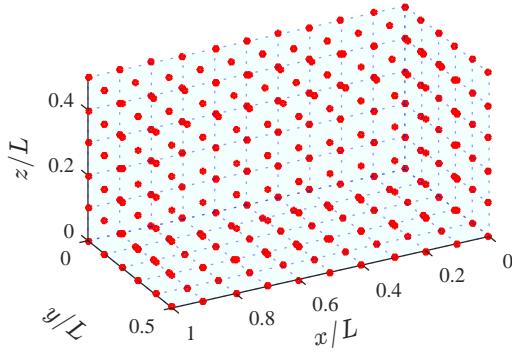


FIGURE 1 – Points de discrétisation suivant les trois coordonnées spatiales (rouge). $N = 252$, $n_\lambda = 10$, $f = c/L$. Pour faciliter la lecture les plans $x = 0$, $y = 0$ et $z = 0$ ont été représentés en cyan.

Question 2

Notons, pour $i \in \llbracket 1, N \rrbracket$, \mathbf{x}_i le vecteur position du point du nuage indicé i . Introduisons alors la matrice de la fonction de Green G que nous définissons par \mathbb{G} avec pour tout $(i, j) \in \llbracket 1, N \rrbracket^2$:

$$\mathbb{G}_{i,j} = \begin{cases} G(\mathbf{x}_i, \mathbf{x}_j) = \frac{e^{ik|\mathbf{x}_i - \mathbf{x}_j|}}{|\mathbf{x}_i - \mathbf{x}_j|} & \text{si } i \neq j \\ 0 & \text{si } i = j \end{cases} \quad (2)$$

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

Nous pouvons remarquer qu'à partir de $N \sim 10000$ l'assemblage de la matrice est trop gourmand en mémoire et cela rend l'exécution sous Matlab impossible. Nous avons donc fait varier f à n_λ 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 τ_a et τ_c en fonction de N .

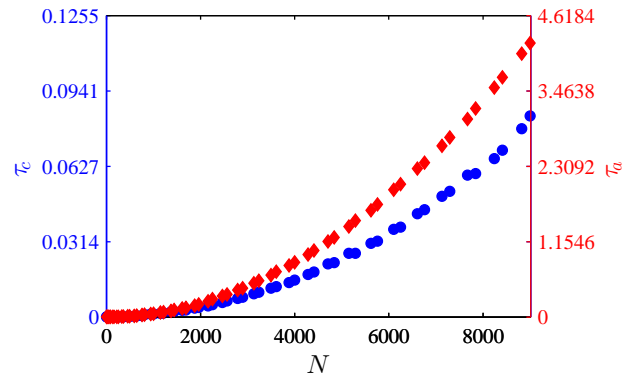


FIGURE 2 – Temps d'assemblage de \mathbb{G} , τ_a (losanges rouge) et temps de calcul du produit matrice vecteur τ_c (ronds bleu) en fonction de N . Les deux courbes ont une allure parabolique mais nous pouvons remarquer que, les échelles étant différentes, le temps d'assemblage est plus long

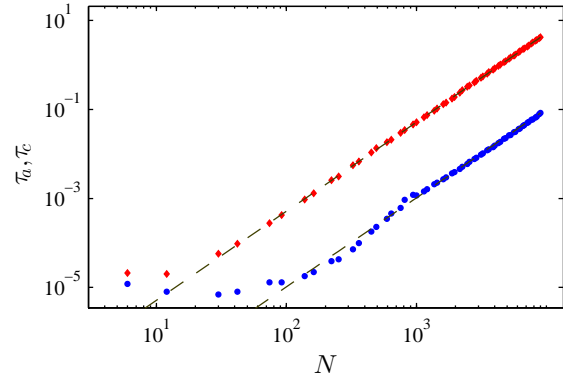


FIGURE 3 – Temps d'assemblage de \mathbb{G} , τ_a (losanges rouge) et temps de calcul du produit matrice vecteur τ_c (ronds bleu) en fonction de N . Asymptotiquement $\log(\tau_a/c) \simeq 2 \log(N) + K_{a/c}$ représentées en pointillé vert avec $K_{a/c}$ constante.

Comme montré en Fig. 3, l'évolution du logarithme de τ_a avec N tend asymptotiquement vers une droite de pente 2. Il en est de même pour le logarithme de τ_c avec N . Nous avons pu vérifier cette évolution quadratique à l'aide d'une régression linéaire, 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 diagonalisant la matrice tridiagonale définie dans l'énoncé. La méthode utilisée pour obtenir les points de quadrature est la méthode de Golub-Welsh¹. 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 $\deg P \leq 2L_q - 1$,

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

Ceci est équivalent, par un changement de variable, à

$$\int_0^\pi P(\cos(t)) \sin(t) dt = \sum_{i=1}^{L_q} \omega_i P(\cos(\theta_i)) \quad (4)$$

Nous avons testé cette quadrature avec $L_q = 3$ en comparaison de celle à 3 points développée 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 quadrature effectuée par Matlab et I_v la valeur vraie pour l'intégration 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 mêmes résultats 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 polynômes de degré 4 et 5, puisque la formule est bien exacte jusqu'au degré $2L_q - 1$, ce que nous n'avions pas avant.

Question 4

Notons, pour $l \in \mathbb{N}$ et $m \in \llbracket -l, l \rrbracket$:

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

Nous avons, par définition,

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

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),

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

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

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

En particulier, la quadrature sur ϕ doit vérifier :

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

Nous avons réalisé une quadrature à $L_q + 1$ points en θ et $2L_q + 1$ points en ϕ . Nous notons une intégration exacte, à erreur d'arrondi machine de l'ordre de 10^{-16} , pour toute les harmoniques sphériques avec $l \leq 2L_q$. Cependant pour $l = 2L_q + 1$ nous pouvons commencer à remarquer des écarts (dus à la quadrature sur ϕ) puisque le code nous donne une valeur numérique $I_{GL-Y_{lm}} = 2.576$ alors que $I_{Y_{lm}} = 0$ pour $Y_{l=9, m=9}$ en utilisant $L_q = 4$. Pour l encore supérieur la quadrature sur θ donne aussi des valeurs éronnées et les résultats ne sont plus satisfaisants.

Nous aurions pu utiliser toute autre quadrature de Gauss, de Simpsons, etc. L'avantage de cette quadrature ci est 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 $l = 2L_q$ et 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

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

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

$$\cos(\hat{\mathbf{s}}, \mathbf{r}_0) = \frac{1}{|\mathbf{r}_0|} \left\langle \begin{pmatrix} \sin(\theta) \cos(\phi) \\ \sin(\theta) \sin(\phi) \\ \cos(\theta) \end{pmatrix}, \mathbf{r}_0 \right\rangle \quad (12)$$

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 sphère unité est alors calculée par la quadrature déterminée dans la question précédente. En effet, la largeur de bande de l'équation à intégrer étant de $2L_G$ comme nous intégrons quasiment exactement avec une quadrature de L_q en θ et $2L_q + 1$ et ϕ les harmoniques sphériques Y_{lm} avec $l \leq 2L_q$ nous prenons ici une quadrature de $L_G + 1$ points en θ et $2L_G + 1$ points en ϕ .

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 $(z, p) \in \mathbb{R}_+^* \times \mathbb{N}$,

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

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 jusqu'à l'ordre L .

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

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 récupérer les valeurs approximativement 10 fois plus vite qu'avec la fonction de Matlab. Pour $(x, p) \in \mathbb{R} \times \mathbb{N}$,

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

Enfin nous noterons aussi que pour réaliser des calculs rapides en vectorialisant au maximum les opérations effectuées nous avons beaucoup 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,

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

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

$$G(\mathbf{x}, \mathbf{y}) = ik h_0^{(1)}(k|\mathbf{r}_0|) \quad (17)$$

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 . Nous notons $G_{\simeq}(\mathbf{x}, \mathbf{y})$ l'approximation de la fonction de Green calculée aux points \mathbf{x} et \mathbf{y} . Nous introduisons aussi

$$\varepsilon = \frac{|G(\mathbf{x}, \mathbf{y}) - G_{\simeq}(\mathbf{x}, \mathbf{y})|}{|G(\mathbf{x}, \mathbf{y})|} \quad (18)$$

$$\text{i.e. } \varepsilon = \|\mathbf{x} - \mathbf{y}\|_2 |G(\mathbf{x}, \mathbf{y}) - G_{\simeq}(\mathbf{x}, \mathbf{y})| \quad (19)$$

k	L_G	\mathbf{x}/L	\mathbf{y}/L	\mathbf{x}_0/L	\mathbf{y}_0/L	$G(\mathbf{x}, \mathbf{y})$	$G_{\simeq}(\mathbf{x}, \mathbf{y})$	ε (%)
4π	2	(0 0 0)	(1.5 0.5 0.5)	(0 0 0)	(1.5 0.5 0.5)	$-0.25 + 0.55i$	$-0.25 + 0.55i$	10^{-18}
4π	10	(0 0 0)	(1 0.5 0.5)	(0 0 0)	(1.5 0.5 0.5)	$-0.25 + 0.55i$	$-0.25 + 0.55i$	10^{-18}
4π	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
4π	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
4π	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
4π	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
4π	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
4π	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

Nous pouvons commencer par remarquer que dans le cas $\mathbf{x} = \mathbf{x}_0$ et $\mathbf{y} = \mathbf{y}_0$ nous obtenons un résultat exact même pour $L_G = 2$ qui est la plus petite valeur accessible

dans notre code.

Question 6

1. Partitionnement

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 réalisé, 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 beaucoup des tableaux de **partition** sont creux nous avons utilisé une définition en tant que *sparse* de ces matrices/tableaux.

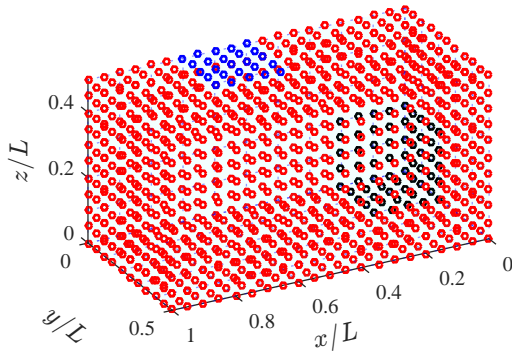


FIGURE 4 – Représentation de deux paquets de points appartenant à deux boîtes de taille d différentes en sortie de notre algorithme de partition (représentés en bleu et en noir). Celle en bleu contient moins de points car elle ne contient que des éléments du haut de la boîte alors que celui en noir regroupe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

2. Algorithme FMM

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

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

3. Résultats

Nous avons regardé l'erreur et norme $\| \cdot \|_2$ et en norme $\| \cdot \|_\infty$, entre le résultat de l'algorithme FMM et le résultat par le produit matrice vecteur classique en fonction de L_G . Nous notons plus précisément aussi \mathbb{G}_\simeq la matrice associée comme en Eq. (2) à la fonction de Green précédemment définie G_\simeq et de plus :

$$\eta_2 = \frac{\|(\mathbb{G} - \mathbb{G}_\simeq)\rho\|_2}{\|\mathbb{G}\rho\|_2} \quad \text{et} \quad \eta_\infty = \frac{\|(\mathbb{G} - \mathbb{G}_\simeq)\rho\|_\infty}{\|\mathbb{G}\rho\|_\infty} \quad (20)$$

La figure Fig. 5 montre, pour $\rho = \mathbf{1}$ (vecteur ayant toutes ces composantes à 1), l'évolution de ces coefficients avec L_G .

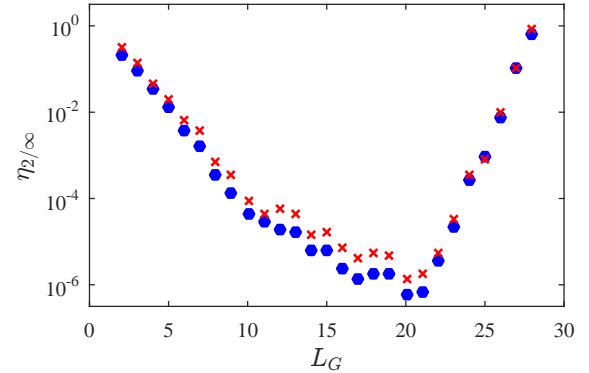


FIGURE 5 – Evolution des écarts η_2 (points bleus) et η_∞ (croix rouges) en fonction de L_G représentant le nombre de points de quadrature et échelle logarithmique. $f = 4c/L$, $n_\lambda = 10$ et $d = 0.3\lambda$ ce qui donne $N = 4002$ et un nombre de 162 partitions

Nous remarquons qu'il y a un point de minimum. Avec la formule du cours nous avons une estimation de la valeur de L_G à choisir pour se placer à ce minimum :

$$L_G = \sqrt{3}kd + 7.5 \log(\sqrt{3}kd + \pi) \quad (21)$$

Nous trouvons qu'alors $L_G = 17$ au point. Sur notre graphique nous remarquons un minimum en $L_G = 20$ ce qui est convenablement proche. Avec des erreurs relatives qui sont alors très faibles.

Question 7

Nous montrons en Fig. 6 l'évolution du temps de calcul τ_{FMM} par la méthode FMM et τ_{class} par la méthode classique du produit matrice vecteur en fonction de N . Pour augmenter le nombre de points nous fixons $n_\lambda = 10$ et nous faisons varier la fréquence entre $0.2c/L$ et $5c/L$. Nous fixons $d = 0.5\lambda$ à chaque itération pour cette figure et $L = 10$ afin d'avoir une précision correcte sans pour autant trop alourdir le calcul déjà long que sa conception nécessite.

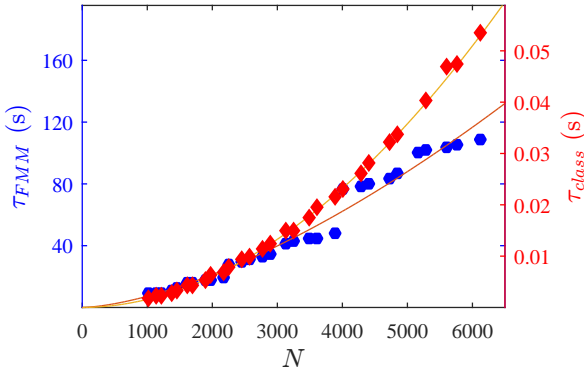


FIGURE 6 – Evolution du temps de calcul du produit matrice vecteur par la méthode classique τ_{class} (rouge) et par la méthode FMM τ_{FMM} (bleu) avec N . Les courbes jaunes et vertes correspondent aux modélisations en loi de puissance réalisés avec Matlab.

Nous remarquons que, même si cela ne paraît pas évident à cause des échelles différentes mises en jeu sur la figure, la courbe de τ_{FMM} décroît moins vite que celle de τ_{class} . Afin de vérifier cette tendance sont représentés en en jaune et vert les courbes de meilleures interpolations avec les équations suivantes : $\tau_{FMM} \simeq 1.63 \times 10^{-4} \times N^{1.55}$ (s) et $\tau_{class} \simeq 1.97 \times 10^{-9} N^{1.95}$ (s). Même en utilisant l'option `-singleCompThread` de Matlab la complexité du produit

classique est légèrement en dessous de la complexité théorique ceci peut s'expliquer par le fait qu'il nous aurait fallu peut être plus de points à N plus grand pour ne prendre en compte que le comportement asymptotique. Malgré cela nous ne sommes pas loin de la complexité théorique et ceci nous permet de voir que notre méthode FMM se rapproche aussi de sa complexité théorique en $O(N^{3/2})$.

Cependant, nous observons une courbe qui évolue par paliers. Sachant que sur chaque palier le nombre de partition est constant nous pouvons en conclure qu'en augmentant la fréquence, nous rajoutons par étapes des partitions et ainsi nous obtenons une augmentation par petits sauts du temps.

Question 8

Nous pouvons remarquer que bien que malgré une complexité plus avantageuse et nos efforts pour le rendre le plus rapide possible ce nouveau algorithme met plus de temps à s'exécuter que le produit réalisé classiquement pour le nombre de points (qui reste relativement faible) que nous avons pu tester. En effet nous avons pour $N = 6012$, un temps $\tau_{FMM} = 108$ s et $\tau_{class} = 0.05$ s. Nous pouvons dire, en comparant les deux courbes de la question précédente et de la question 2 qu'il faudrait atteindre $N \sim 10^7$ pour que l'utilisation de la FMM mono-niveau soit, dans ce cas, utile si l'on prend en compte le temps d'assemblage de la matrice dans la méthode classique. Ce nombre est trop malheureusement important et il serait nécessaire de réussir à optimiser le temps d'exécution du code de manière générale ou de passer par un algorithme multi-niveaux.