Rapport TP AMS304
TP4

Aurélien Valade

1 Introduction

Le but de ce TP est la mise en place d'un code de calcule efficace d'interaction entre différents points d'une sphère plongée dans un espace à trois dimensions. On va pour cela considérer une méthode multipôle ainsi qu'une aproximation par décomposition en ondes planes pour la fonction de Green qui modélise l'intéraction mentionnée. Ces approximations sont valables *i.e.* donnent de bons résultats sous certaines conditions qui seront précisées plus tard.

On donne la fonction de Green complexe

$$G(x,y) = \frac{\exp(ik|x-y|)}{4\pi|x-y|}, \tag{1}$$

et un maillage de la sphère[1] dont les sommets seront notés par la suite les $\{x_i\}_{1,N}$. En considérant le vecteur aléatoire $\rho \in \mathbb{R}^N$, nous allons essayer de calculer de la manière la plus rapide possible une approximation de $V \in \mathbb{R}^N$ défini par

$$V_i = \sum_{j \neq i} G(x_i, x_j)\rho_j. \tag{2}$$

Remarque 1. *Cette fonction se réécrit avec un unique argument $r > 0$*

$$G(r) = \frac{\exp(ikr)}{4\pi r}.$$

2 Attaque brutale du problème

Dans un premier temps on cherche à calculer la quantité recherchée de manière "naïve". On va pour cela construire la matrice $\mathbb{G} \in \mathcal{M}_N(\mathbb{R})$ telle que pour un maillage donné $\{x_i\}_{i \leq N}$,

$$\mathbb{G}_{ij} = G(x_i, x_j)$$

et qui nous permettra de calculer le produit matriciel classique présenté en Equation 2.

Les fonctions MATLAB utilisée font le plus appel possible à la vectorialisation. Des résultats sont présentés en Table 1, mais ce qui est essentiel de retenir est que le calcul est très rapidement infaisable à cause de la taille en mémoire des matrices. On remarque que l'explosion de la demande en mémoire à partir du mesh 2 est peut être aussi due à une mauvaise gestion de la mémoire par MATLAB directement, car on supposerait des valeurs bien plus faibles (comme indiquées en rouge dans la colonne mémoire utilisée).

On observe en revanche aucune variation dans ces catégories quand k varie.

[1] En réalité on en prendra quatre différents, avec quatre tailles de mailles.

# maillage	nb points	temps de calcul (s)	mémoire utilisée
0	642	$6{\cdot}10^{-2}$	6 Mo
1	2562	1.1	100 Mo
2	10242	?	$\geq$ 8 Go (sup. 2.5 Go)
3	40962	?	$\geq$ 8 Go (sup. 10 Go)

Table 1: Temps de calcul et ressources utilisées en fonction de la taille du problème.

3 Approximation harmonique de la fonction d'onde

On peut montrer par le calcul que la fonction d'onde peut se réécrire sous la forme d'une somme de contributions harmoniques donnée en 3 sous condition que les points x et y soient *assez* éloignés. Considérons un ensemble de points $\{x_i\}_{1,N_x}$ au voisinage de x_0 et un second ensemble de points $\{y_i\}_{1,N_y}$ dans le voisinage de y_0. On peut donner comme condition suffisante pour définir le voisinage que à k donné, il faut

$$\max_{i\in[1,N_x]} \left(|x_i - x_0|\right) \geq 0.3\frac{2\pi}{k}$$

et de même pour les $\{y_i\}_{1,N_y}$. On définit alors $r_0 = y_0 - x_0$ et $r = y - x - r_0$.

$$G(x,y) = \lim_{L\to\infty} \int_{\hat{s}\in S^2} e^{ik\hat{s}\cdot r}\mathcal{G}_L(\hat{s}, r_0) \tag{3}$$

ou $\mathcal{G}_L(\hat{s}, r_0)$ désigne la somme

$$\mathcal{G}_L(\hat{s}, r_0) = \frac{ik}{16\pi^2}\sum_{p=0}^{L}(2p+1)i^p h_p^{(1)}(k|r_0|)P_p(\cos(\hat{s}, r_0)) \tag{4}$$

avec $h_p^{(1)}$ la première fonction de Hankel sphérique et P_p le polynôme de Legendre d'ordre p.

Remarque 2. *Dans la suite on notera $G_L(r, r_0)$ la somme partielle de la série convergent vers $G(r)$:*

$$G_L(r, r_0) = \int_{\hat{s}\in S^2} e^{ik\hat{s}\cdot r}\mathcal{G}_L(\hat{s}, r_0)$$

Remarque 3. *Une écriture plus proche de l'application*

$$G(x,y) = \lim_{L\to\infty} \int_{\hat{s}\in S^2} \underbrace{e^{ik\hat{s}\cdot(x_0-x)}}_{Contribution}\underbrace{\mathcal{G}_L(\hat{s}, r_0)}_{Transfert}\underbrace{e^{ik\hat{s}\cdot(y-y_0)}}_{Répartition}$$

permet de mieux comprendre comment l'on va procéder dans la suite. On ne va en effet pas considérer tous les couples $(x,y) \in C_x \times C_y$ avec C_x et C_y deux cellules disjointes, cet ensemble étant de taille $|C_x| \times |C_y| \approx \mathcal{O}(N^2)$. Cela reviendrait à essayer de calculer la contribution de chaque point x sur chaque point y, ce que l'on veut justement éviter de faire. On va plutôt prendre tous les x séparément des y, on a donc un ensemble de couples de cardinal $|C_x| + |C_y| \approx \mathcal{O}(N)$.

Trois grandes difficultés se présentent :

1. il faudra faire une quadrature sur la sphère unité pour calculer l'intégrale ;
2. la fonction $\mathcal{G}_L$ est relativement compliquée ;
3. il faudra faire un découpage adapté de la sphère avec de bons voisinages pour que l'approximation reste correcte.

4 Quadrature sur la sphère unité

Tout point $\hat{s}$ de la sphère peut être paramétré par les angles θ et φ. On peut donc définir la quadrature générale suivante pour une fonction $f : \mathbb{R}^3 \to \mathbb{R}$

$$\int_{S^2} f = \sum_{i,j} w_i^{\varphi} w_j^{\theta} f(x(\theta_j, \varphi_i), y(\theta_j, \varphi_i), z(\theta_j, \varphi_i))$$

reste à savoir quels points et quels poids prendre.

Parmis plusieurs quadratures possibles, dont la plus simple serait une quadrature uniforme sur les deux angles φ et θ, on choisie une quadrature optimale pour les harmoniques sphériques, c'est à dire qui intègre exactement les harmoniques sphériques pour $l < 2L$. Soit $L \in \mathbb{N}$, on pose $I = 2L+1$ et $J = L+1$.

- On fait une quadrature uniforme d'ordre I sur la variable φ :

$$\begin{cases} \varphi_i = \frac{2\pi i}{I} \\ w_i^{\varphi} = 1/I \end{cases} \quad \forall i \in [1, I].$$

- On fait une quadrature de Gauss-Legendre d'ordre J sur la variable θ. Cette quadrature permet d'intégrer exactement des polynômes d'ordre $2J+1$ sur le segment $[-1, 1]$.

Pour trouver les points et les poids de la quadrature de Gauss-Legendre plusieurs solutions s'offrent à nous. On décide d'utiliser la diagonalisation de la matrice symétrique $\mathbb{T} \in \mathcal{M}_J(\mathbb{R})$

$$\mathbb{T} = \begin{pmatrix} 0 & \frac{1}{\sqrt{3}} & 0 & & \\ \frac{1}{\sqrt{3}} & 0 & \ddots & 0 & \\ 0 & \ddots & \ddots & \frac{i}{\sqrt{4i^2-1}} & 0 \\ & 0 & \frac{i}{\sqrt{4i^2-1}} & 0 & \ddots \\ & & 0 & \ddots & \ddots \\ & & & & \end{pmatrix}$$

diagonalisable en

$$\mathbb{T} = PDP^{-1}$$

avec $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_J)$ et $P = (V_1 \ \ldots \ V_J)$. On peut déduire les valeurs nécessaires pour la quadrature:

$$\begin{cases} \theta_j = \arccos(\lambda_j) \\ w_j^{\theta} = 2V_j^2 \end{cases} \quad \forall j \in [1, J].$$

Ce genre de quadrature se génère avec une complexité en $I \times J^2 \approx \mathcal{O}(L^3)$. Le terme en J^2 se justifie par la recherche des valeurs propres pour calculer les points et les poids. Dans la suite, nous allons faire quelques vérifications sur les harmoniques sphériques

$$Y_{lm}(\hat{s}) = \sqrt{\frac{(2l+1)(l-m)!}{4\pi(l+m)!}} P_l^m(\cos(\theta)) e^{im\varphi} \quad \forall l \geq 0, \ \forall -l \leq m \leq l$$

dont on connaît la formule d'intégration théorique

$$\int_{S^2} Y_{lm} = \sum_{i,j} w_i^{\varphi} w_j^{\theta} Y_{lm}(\theta_j, \varphi_i) = \begin{cases} 2\sqrt{\pi} & \text{si } l = m = 0 \\ 0 & \text{sinon.} \end{cases}$$

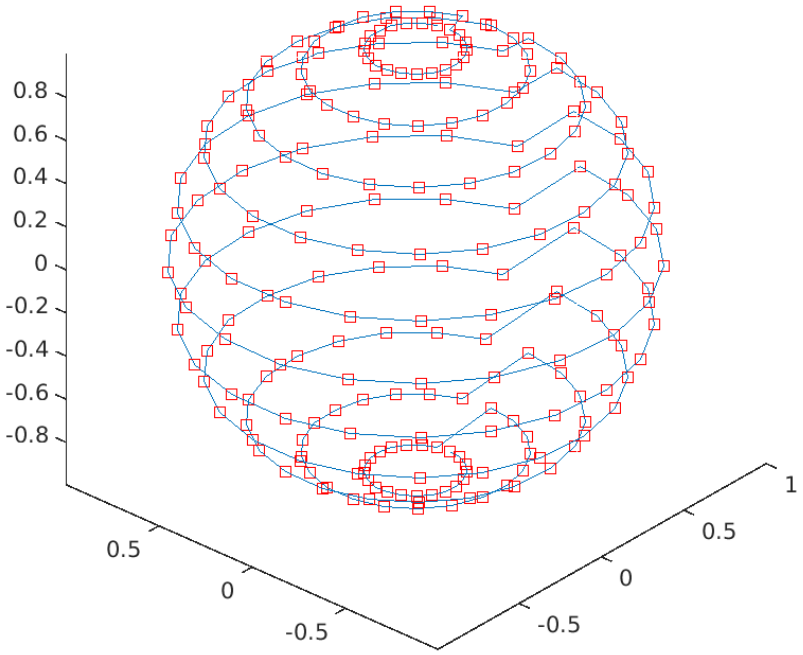

Figure 1: Représentation de la sphère pour $L = 10$.

On obtient en effet les valeurs attendues avec des nombres complexes dont la norme est d'environs 10^{-16} pour les valeurs non nulles de l, m et est proche de $2\sqrt{\pi}$ pour $(l, m) = (0, 0)$ à la précision flottante près.

Cependant, on aimerait en savoir plus sur comment se comporte la valeur de cette intégrale en fonction du L. On observé qualitativement dans un premier temps que pour un l fixé et pour tout $m \in [0, l]$, il y a une valeur seuil de L à partir de laquelle la valeur de l'intégrale chute de plusieurs ordres de grandeur en passant de l'unité à 10^{-16}. La dépendance de ce L_{seuil} par rapport au paramètre l est tracée en Figure 2, où on peut supposer une relation linéaire avec un coefficient d'environ $1/2$, c'est donc bien la loi des $l < 2L$ que l'on retrouve.

On ne s'attend pas à avoir un comportement identique plus tard mais on peut espérer trouver des similarités du fait de la présence de polynômes de Legendre dans les deux cas, sinon cette étude nous aura au moins permis de savoir de quel ordre de grandeur on doit choisir L pour des cas simples.

Remarque 4. *Suite aux bugs décrits plus loin, d'autres tests ont été menés avec d'autres fonctions d'intégration :*

$$\begin{aligned} f_x : &\mathbb{R}^3 \to \mathbb{R} \\ &(x, y, z) \to x \end{aligned}$$

$$\begin{aligned} f_y : &\mathbb{R}^3 \to \mathbb{R} \\ &(x, y, z) \to y \end{aligned}$$

$$\begin{aligned} f_z : &\mathbb{R}^3 \to \mathbb{R} \\ &(x, y, z) \to z \end{aligned}$$

qui sont toutes d'intégrale théoriquement nulle. C'est en effet le cas numériquement, ce qui confirme notre quadrature.

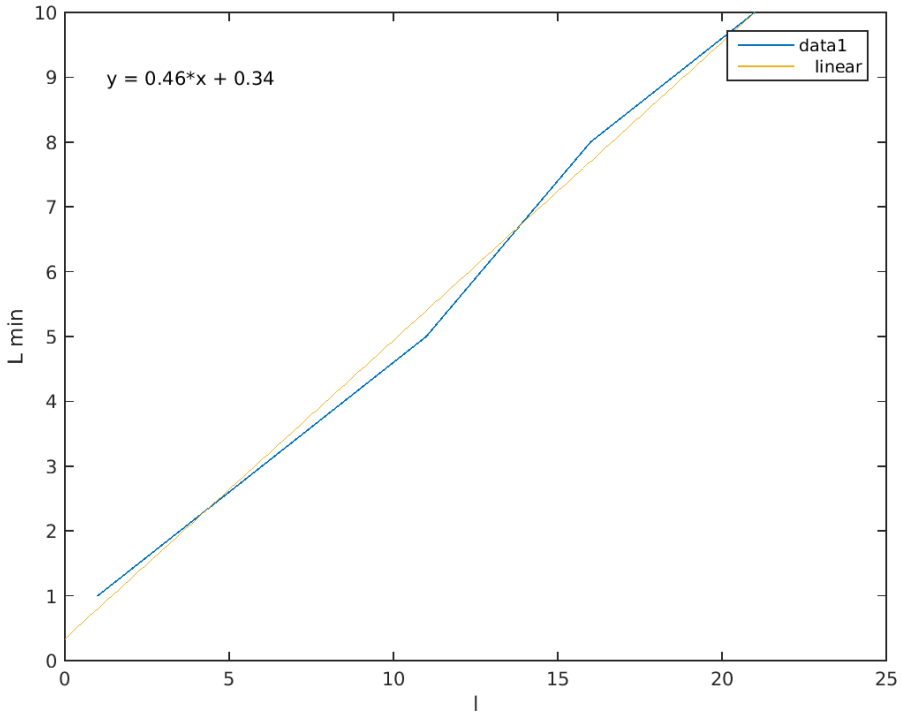

Figure 2: Valeur de L_{seuil} à partir de laquelle l'intégrale est supposée juste en fonction de l.

5 Éléments de calculs de la fonction $\mathcal{G}_L$

Remarque 5. *Dans la suite on note $\circ$ le produit terme à terme (aussi appelé produit de Schur).*

On rappelle que les fonctions Hankel sphériques $h_n^{(1)}$ s'obtiennent à partir des fonctions de Hankel $H_n^{(1)}$ cartésiennes implémentées dans MATLAB avec la formule

$$h_n^{(1)}(z) = \sqrt{\frac{\pi}{2z}} H_{n+1/2}^{(1)}(z), \quad \forall z \in \mathbb{C}, \forall n \in \mathbb{N}$$

Le calcul de G_L se fait en trois étapes indépendantes qui peuvent séparées dans un but d'optimisation bien que l'ordre soit fixé

1. Calcul des points de la quadrature ;
2. Calcul de la fonction $\mathcal{G}_L$;
3. Calcul des exponentielles complexes et de l'intégrale.

Le calcul vectorialisé de la valeur de $\mathcal{G}_L(\hat{s}, r_0)$ avec $N_{\hat{s}}$ points de quadrature se fait comme suit

1. Créer le vecteur $F \in \mathbb{C}^L$ tel que $F_p = (2p+1)i^p, \forall 1 \leq p \leq L$
2. Créer le vecteur $H \in \mathbb{C}^L$ tel que $H_p = h_p^{(1)}(k|r_0|)$
3. Calculer le vecteur $K \in \mathbb{C}^L$ tel que $K = \frac{ik}{16\pi^2} F \circ H$
4. Créer la matrice $\mathbb{P} \in \mathcal{M}_{N_{\hat{s}} \times L}(\mathbb{C})$ telle que $\mathbb{P}_{sp} = P_p(\cos(\hat{s}_s, r_0))$
5. Calculer le vecteur $G \in \mathbb{C}^{N_{\hat{s}}}$ tel que $G = \mathbb{P}K$.

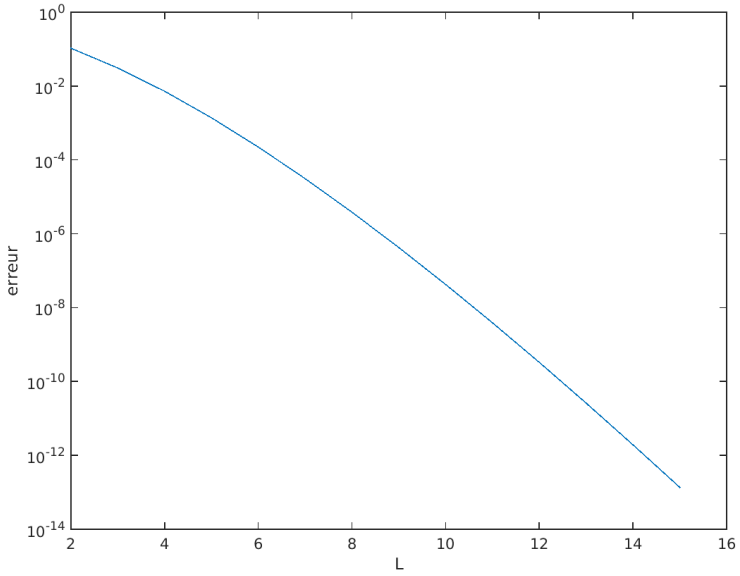

Figure 3: Erreur en fonction de L. On a ici $k = 2,\ x_0 = (10, 0, 0),\ y_0 = (1, 0, 0),\ N_x = N_y = 50,\ d_x = d_y = 1$.

Le vecteur G est le résultat vectorialisé pour les tous les points de la sphère : $G_s = \mathcal{G}_L(\hat{s}_s, r_0),\ \forall s < N_{\hat{s}}$.

Pour vérifier que cette fonction donne de bons résultats, on les compare avec ceux de $G(r)$ pour plusieurs paramètrisations de L, r_0 et k. Pour cela on fixe un r_0 relativement grand. On pose ensuite un ensemble de N_x (*resp.* N_y) points aléatoires dans une boule de rayon d_x (*resp.* d_y) tel que $\max(d_x, d_y) \ll |r_0|$. On fixe ensuite k tel que

$$k \geq 0.3 \cdot 2\pi/|r_0| \approx 2.1/|r_0|. \tag{5}$$

Cependant... Malgré une bonne dizaine d'heures de recherche, impossible de débugger la fonction qui calcule $\mathcal{G}_L$... Les résultats, bien que du bon ordre de grandeur ne sont pas continus en r_0 dans la majorités des cas où l'on s'attendrait à avoir une décroissance en $1/|r_0|$. On ne retrouve pas non plus de forme en $\sin(x)$ ou $\cos(x)$ sur les parties réelles et imaginaires comme on pourrait l'espérer. Voici ce que j'aurais écrit si le code ne s'était pas mis à marcher à 22h la veille de rendre le rapport... Moyennant "bricolage" (coefficient -1...) on retrouve maintenant de bons résultats.

Dans la Figure 3 on voit l'erreur calculée comme la norme de Foebus de la différences entres les matrices obtenues par méthodes directe et par méthode du kernel :

$$err = \sum_{ij} (\mathbb{G}_{ij} - (\mathbb{G}_L)_{ij})^2.$$

On voit que celle ci chute plus vite que polynomialement. On s'intéresse en Figure 4 à la même erreur commise en fonction de $|r_0|$ à k constant. On voit encore une fois une décroissance très forte puisqu'elle semble exponentielle alors que le graphe est en semi log. Les petits sauts d'erreurs pouvant être dûs au tirage aléatoire des points x pour chaque valeur de x_0. Enfin en Figure 5 on observe le temps de calcul en fonction du nombre de points $N_x = N_y = N$. On voit que malgré une grosse ordonnée à l'origine, la courbe de temps de la méthode kernel devient rentable pour N grand. Ce phénomène est amplifié par le fait que les demandes en ressources aux environs de 10^4 points explosent et font passer une partie des données sur la mémoire `swap` ce qui augmente énormément les temps de calcul.

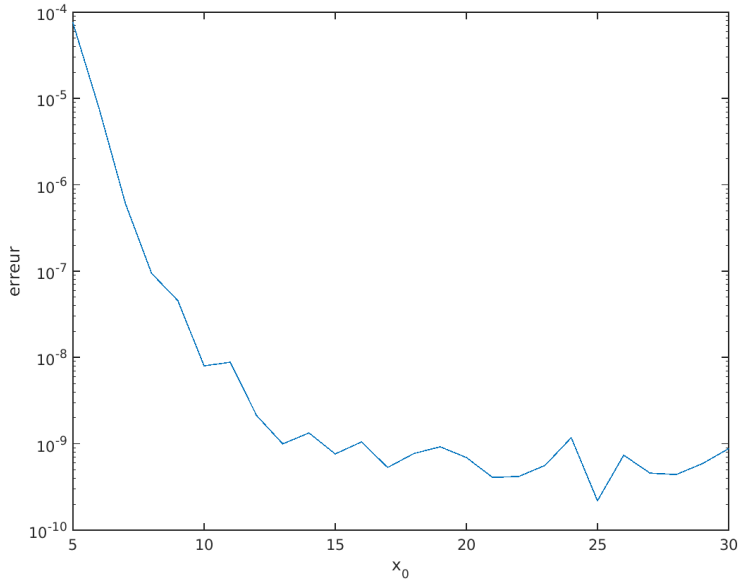

Figure 4: Erreur en fonction de $|r_0|$. On a ici $k = 2,\ L = 12,\ y_0 = (1, 0, 0),\ N_x = N_y = 50,\ d_x = d_y = 1$.

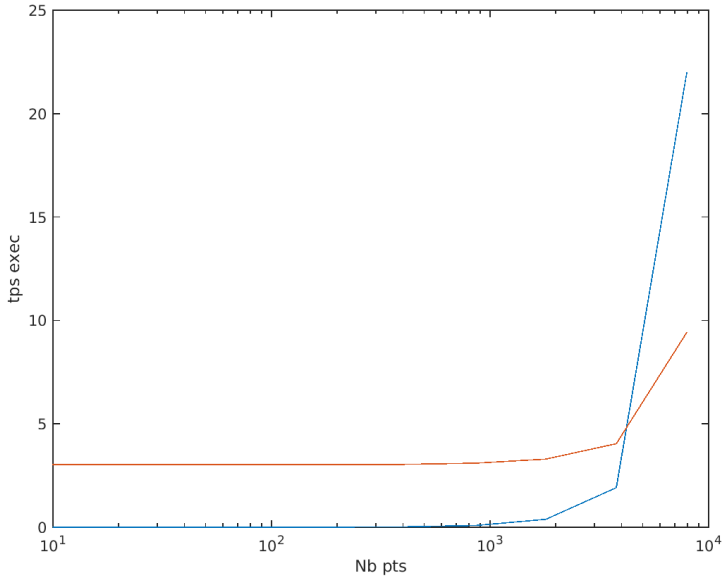

Figure 5: Temps de calcul en fonction de $N_x = N_y = N$. On a ici $k = 1,\ L = 12,\ x_0 = (10, 0, 0),\ y_0 = (1, 0, 0),\ d_x = d_y = 1$. En rouge on voit la courbe pour la méthode kernel, en bleu la méthode classique. Les temps sont donnés en secondes.

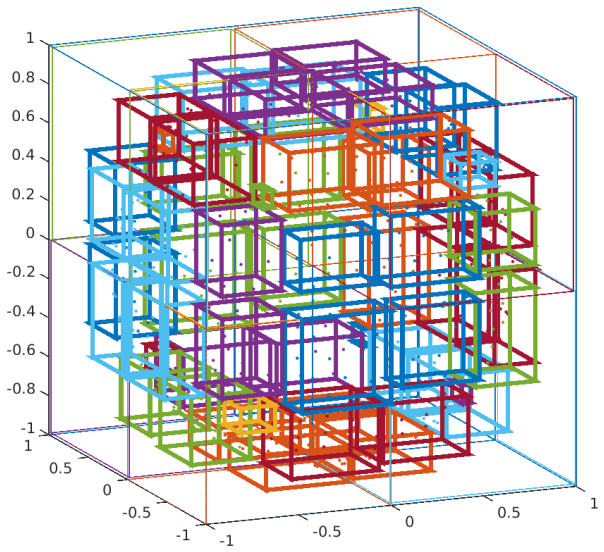

Figure 6: Découpage de la sphère

6 Découpage du maillage en boîtes adaptées

Il s'agit maintenant de faire un partitionnement du maillage de la sphère qui soit cohérent avec les approximations. Pour cela on a récupéré sur le net un objet MATLAB nommé `OcTree`[2] ou les routines de bases pour partitionner sont codées. Parmi les paramètres à fixer on choisie d'empêcher plus de 2 niveaux de profondeurs dans l'arbre. De plus on donne une taille minimale aux cubes en fonction de k avec la règle énoncée en Equation 5.

Remarque 6. *On ne travaillera qu'au niveau feuille, mais les niveaux 0 et 1 étant particuliers il faut monter au niveau 2 de profondeur pour que l'algorithme de partionnement fonctionne correctement.*

Une fois qu'on a filtré les cubes de sorte à n'avoir que les feuilles de l'arbre, il faut ajouter une méthode pour sauvegarder les voisinages, puisque les méthodes de kernel ne sont applicables que dans les partitions lointaines *i.e.* à plus d'un cube de distance. On ne peut en effet pas appliquer la méthode kernel sur les cubes voisins car pour un point de notre partition à la frontière avec la boîte voisine, les points de la seconde sont bien trop proches pour que la méthode kernel soit adaptée. On code cette information sous la forme d'une matrice symétrique $\mathbb{V} \in \mathcal{M}_{N_\mathrm{p}}(\{0,1\})$ telle que

$$\mathbb{V}_{ij} = \begin{cases} 1 & \text{si les boîtes } i \text{ et } j \text{ sont voisines ou bien si } i = j \\ 0 & \text{sinon.} \end{cases}$$

On peut voir un de ces découpages dans la Figure 6, qui semble coloré néanmoins complexe.

La routine principale pour faire la FMM est cependant buggée et je n'ai plus le temps de corriger les problèmes...

A Description des fonctions

[2] Le lien vers la source.

Fonction	Description
`besselh_Sph.m`	Calcul des fonctions de Hankel sphériques à partir des fonctions de Hankel cartésiennes.
`calc_quad.m`	Calcul des points et des poids de la quadrature nécessaire.
`dist.m`	Calcul la distance point à point entre deux vecteurs (renvoie donc une matrice).
`FMM_G.m`	Fait le nécessaire pour le calcul de Equation 2 par la méthode la FMM. Construction de l'arbre, recherche des voisins, calcul de la quadrature, calcul des valeurs de G_L locales et produits. Pas tout à fait finie pour être honnête, ρ n'apparaît pas dans cette version, on construit donc actuellement l'ensemble de la matrice...
`G_devel_G_L.m`	Calcul $\mathcal{G}_L$ pour L, k, et une quadrature données.
`G_devel.m`	Calcul de G_L avec $\mathcal{G}_L$ et une quadrature données.
`G.m`	Calcul classique de la matrice $\mathbb{G}$.
`integ_SphHarmo.m`	Calcul de l'intégrale d'une harmonique sphérique.
`main.m`	Ensemble des scripts de plot présentés dans ce rapport.
`OcTree.m`	Objet MATLAB qui permet le découpage spatial de la sphère.
`read_meshfile.m`	Méthode pour lire un fichier msh.

Table 2: Description des fonctions utilisées.